



```
public:
    RegLgraph G() {
        v=A; // начальная (входная) вершина
    }
    void gc() { c=cin.get(); }

    bool accept() {
        try {
            while(gc(), !cin.eof()) {
                switch(v) {
                    case A: if(c=='b') v=B;
                            else if(c!='a') throw c;
                            // else v=A;
                            break;
                    case B: if(c=='b') v=A;
                            else if(c!='a') throw c;
                            // else v=B;
                            break;
                }
            }
        } catch(char c) { while (gc(), !cin.eof()); return false; }
        return v==B; // успех при достижении заключительной
                    // (выходной) вершины
    }
}; //end of RegLgraph_G

int main() {
    cout<< RegLgraph_G().accept()<<endl;
    return 0;
}
```

И.А. Волкова, А.А. Вылиток, Л.Е. Карпов

СИСТЕМЫ ПРОГРАММИРОВАНИЯ



МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ имени М. В. ЛОМОНОСОВА
Факультет вычислительной математики
и кибернетики



И.А. Волкова, А.А. Вылиток, Л.Е. Карпов

Сборник экзаменационных задач по курсу

СИСТЕМЫ ПРОГРАММИРОВАНИЯ

(ООП на Си++ и методы трансляции
на основе формальных грамматик и автоматов)

Учебное пособие для студентов II курса



$$L(G_3) = \{x \in \{a, b\}^* \mid |x|_b \bmod 2 = 1\}$$

```
#include <iostream>
using namespace std;
```

```
class RegLgraph_G {
    // граф конечного автомата
    char c; // текущий символ
    typedef enum {A, B} Vertex;
    Vertex v; // текущая вершина
```

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М. В. ЛОМОНОСОВА



Факультет
вычислительной математики
и кибернетики



И. А. Волкова, А. А. Вылиток, Л. Е. Карпов

**Сборник экзаменационных задач
по курсу «Системы программирования»
Объектно-ориентированное программирование на Си++
и методы трансляции на основе формальных грамматик
и автоматов**

Учебное пособие для студентов II курса



МОСКВА – 2026

УДК 004.43:004.4'41(075.8)
ББК 32.973.2-018.1я73
В92



<https://elibrary.ru/ayofzl>

*Печатается по решению Редакционно-издательского совета
факультета вычислительной математики и кибернетики
МГУ имени М. В. Ломоносова*

Рецензенты:

И.В. Машечкин – д.ф.-м.н.;

С.Ю. Соловьев – д.ф.-м.н.

Волкова И.А., Вылиток А.А., Карпов Л.Е.

**В92 Сборник экзаменационных задач по курсу «Системы программирования» (ООП на Си++ и методы трансляции на основе формальных грамматик и автоматов) : учебное пособие для студентов II курса / И.А. Волкова, А.А. Вылиток, Л.Е. Карпов. – Москва: Издательский отдел факультета ВМК МГУ имени М.В. Ломоносова (лицензия ИД N 05899 от 24.09.2001 г.); МАКС Пресс, 2026 – 336 с.
e-ISBN-978-5-89407-664-5 (ВМК МГУ имени М. В. Ломоносова)
ISBN 978-5-317-07623-8 (МАКС Пресс)
<https://doi.org/10.29003/m5356.978-5-317-07623-8>**

В сборнике представлены задачи и упражнения по языку Си++, основам трансляции и формальным грамматикам, рекомендуемые студентам 2 курса для подготовки к коллоквиуму и экзамену в рамках курса «Системы программирования». Рассматриваемая версия языка Си++ соответствует стандарту ISO/IEC (1998), за исключением явно оговоренных случаев использования более поздних стандартов.

В сборнике собраны задачи на использование основных механизмов объектно-ориентированных языков программирования: инкапсуляции, наследования, полиморфизма, а также задачи по методам трансляции на основе формальных грамматик и автоматов. Задачи предлагались студентам в письменных проверочных работах в 2008–2024 гг.

Для студентов факультета ВМК в поддержку основного лекционного курса «Системы программирования», а также для преподавателей, ведущих практические занятия по этому курсу.

Авторы выражают благодарность преподавателям кафедры алгоритмических языков за участие в обсуждении и составлении задач.

УДК 004.43:004.4'41(075.8)
ББК 32.973.2-018.1я73

**e-ISBN-978-5-89407-664-5
ISBN 978-5-317-07623-8**

© И.А. Волкова, А.А. Вылиток, Л.Е. Карпов, 2026
© Факультет ВМК МГУ имени М. В. Ломоносова, 2026
© Оформление. ООО «МАКС Пресс», 2026

ОГЛАВЛЕНИЕ

План курса «Системы программирования»	4
2008. Коллоквиум	11
Экзамен	17
2009. Коллоквиум	21
Экзамен	37
2010. Коллоквиум	57
Экзамен	69
2011. Коллоквиум	93
Экзамен	109
2012. Коллоквиум	128
Экзамен	140
2013. Коллоквиум	152
Экзамен	164
2014. Коллоквиум	176
Экзамен	188
2015. Коллоквиум	200
Экзамен	216
2016. Коллоквиум	224
Экзамен	232
2017. Коллоквиум	240
Экзамен	248
2018. Коллоквиум	256
Экзамен	263
2019. Коллоквиум	271
Экзамен	277
2021. Экзамен	285
2022. Экзамен	289
2023. Коллоквиум	297
Экзамен	305
2024. Коллоквиум	313
Экзамен	321
Тематический указатель	329

План курса “Системы программирования”

Введение. Определение термина "система программирования"

1. Этапы жизненного цикла программного продукта
 - Программы и программные продукты
 - Фазы разработки, использования и сопровождения программных продуктов
2. Технологии разработки программного обеспечения
 - Процессы и этапы разработки и сопровождения
3. Компоненты СП, поддерживающие разработчика программного обеспечения
 - Требования к составу систем программирования
 - Основные компоненты системы программирования UNIX
 - Система программирования на основе совокупности приемов и решений RUP
4. ООП как технология (парадигма) программирования
5. Основные принципы ООП:
 - Абстракция
 - Инкапсуляция
 - Наследование
 - Полиморфизм
6. Обзор средств языка Си++, поддерживающих основные механизмы
 - Отличия языка Си++ от языка Си
 - Строгая типизация
 - Обобщенные константные выражения
 - Вывод типов
 - Абстрагирование
 - Инкапсуляция
 - Модульность программ, интерфейсные модули и модули реализации
 - Типизация
 - Иерархия
7. Виды отношений между классами
 - ER-диаграммы
 - Взаимодействие и иерархия классов
 - Ассоциация
 - Агрегация
 - Использование
 - Инстанцирование
 - Пример иерархии: одиночное наследование
 - Пример иерархии: множественное наследование
 - Примеры типизации, наследования и полиморфизма в Си++
8. Приёмы декомпозиции
 - Объектно-ориентированный анализ предметной области
 - Отображение сущностей предметной области в классы
 - Преимущества объектной модели
9. Концепция абстрактных типов данных и её реализация в Си++
10. Классы и объекты
11. Работа с состоянием объекта: селекторы и модификаторы
12. Указатели и ссылки на объекты, передача параметров по ссылке
13. Инициализация, конструкторы, копирование, деструкторы, указатель *this*.
 - Пример конструирования класса комплексных чисел

- 14. Работа с динамической памятью
 - Плоские и неплоские классы
 - Семантика переноса
 - Присваивание переносом
- 15. Статические и константные члены классов
- 16. Статический полиморфизм в Си++
 - Перекрытие и перегрузка имён
 - Перегрузка функций
 - Правила работы алгоритма выбора перегруженной функции
 - Перегрузка функций с несколькими параметрами
 - Проблемы отождествления ссылок, константных и неконстантных типов
- 17. Статический полиморфизм в Си++. Перегрузка операций
 - Различные способы перегрузки операций, друзья классов
 - Перегрузка бинарных операций
 - Перегрузка унарных операций
 - Примеры перегрузки операций
 - Индексирование
 - Безопасный массив
 - Операция косвенного доступа к объекту
 - Операция группирования параметров вызова функции
- 18. Одиночное наследование
 - Различия между наследованием и агрегацией, правила наследования
 - Иерархия классов при наследовании
 - Повторное использование программ
 - Видимость при наследовании, правила видимости, перекрытие имён
 - Перекрытие имён
 - Инициализация объектов внутри объектов, списки инициализации
 - Пример наследственной иерархии
 - Генеалогические преобразования указателей
- 19. Виды полиморфизма в Си++. Динамический полиморфизм
- 20. Виртуальные функции
 - Работа виртуальной функции
 - Правила создания виртуальных функций
 - Виртуальные деструкторы, спецификаторы *final* и *override*
- 21. Множественное наследование в Си++
 - Доступ к членам производного класса
 - Преобразование указателей
 - Виртуальные базовые классы
 - Неоднозначность из-за совпадающих имён в различных базовых классах
- 22. Абстрактные классы. Чистые виртуальные функции
- 23. Фабрики объектов. “Виртуальные” конструкторы
- 24. Принципы реализации возможностей ООП в существующих системах
 - Реализация одиночного наследования
 - Реализация множественного наследования
 - Реализация виртуальных функций

- 25. Пространства именования
- 26. Средства обработки ошибок. Исключения в Си++. Проблемы реализации.
- 27. Динамическая идентификация типа (RTTI), средства преобразования типов, проблемы реализации средств управления информацией о типах
- 28. Параметрический полиморфизм в Си++. Программирование с помощью шаблонов
 - Шаблоны функций и классов
 - Паттерны метапрограммирования
- 29. Стандартная библиотека языка Си++
- 30. Принципы дизайна и реализация стандартной библиотеки шаблонов Си++
- 31. Основные виды компонентов библиотеки шаблонов STL:
 - Контейнеры, квазиконтейнеры
 - Итераторы
 - Распределители памяти (аллокаторы)
 - Алгоритмы
 - Функциональные объекты, λ -функции
 - Адаптеры контейнеров, итераторов и функциональных объектов
 - Вывод сложных типов шаблонных классов
 - Шаблоны *vector* и *list*
 - Другие контейнеры
- 32. Достоинства и недостатки STL-подхода
- 33. Стандарты языка Си++ после 2011 года
- 34. Классическая система программирования
 - Общая схема работы систем программирования
 - Краткая характеристика отдельных компонентов систем программирования
- 35. Трансляторы, компиляторы, интерпретаторы
 - Схемы работы
 - Смешанная стратегия трансляции
- 36. Общая схема работы компилятора
 - Основные этапы компиляции
 - Начальные установки
 - Фазы анализа: лексический анализатор
 - Фазы анализа: синтаксический и семантический анализаторы
 - Внутреннее представление программ
 - Фазы оптимизации программ
 - Фазы синтеза: распределение памяти и регистров
 - Фазы синтеза: генерация команд и машинно-зависимая оптимизация
 - Однопроходный компилятор
- 37. Основные понятия теории формальных языков
 - Алфавит и цепочки символов
 - Лексика, синтаксис и семантика
 - Порождающие грамматики
 - Эквивалентность грамматик
- 38. Классификация грамматик и языков по Хомскому, соотношения между типами грамматик и языков
 - Типы грамматик
 - Регулярные выражения как альтернативный способ описания регулярных языков
 - Иерархии грамматик и языков
 - Примеры языков и грамматик

- 39. Задача разбора, дерево вывода
- 40. Проблемы однозначности и эквивалентности грамматик
- 41. Преобразования грамматик, группы и алгоритмы преобразования
 - Удаление бесплодных и недостижимых символов
 - Удаление правил с пустой правой частью
 - Циклический вывод и удаление цепных правил
 - Алгоритм приведения грамматики
- 42. Регулярные грамматики
 - Разбор цепочек символов языка на основе регулярных (леволинейных) грамматик
 - Детерминированные конечные автоматы
 - Алгоритм построения конечного автомата по леволинейной грамматике
 - Алгоритм построения леволинейной грамматики по конечному автомату
 - Диаграммы состояний, анализаторы
 - Недетерминированный конечный автомат
 - Недетерминированный разбор
 - Алгоритм преобразования НКА в ДКА
 - Праволинейные грамматики
 - Алгоритм построения конечного автомата по праволинейной грамматике
 - Алгоритм построения праволинейной грамматики по конечному автомату
 - Примеры преобразований конечных автоматов
- 43. Лексический анализ на основе регулярных грамматик
 - Задачи лексического анализа на основе регулярных грамматик
 - Определение границ лексем
 - Выполнение действий по преобразованию, диаграммы состояний с действиями
 - Объектно-ориентированный анализ и проектирование иерархии классов лексического анализатора модельного языка
 - Схема работы лексического анализатора модельного языка
 - Построение диаграммы состояний
 - Программирование по диаграмме состояний
- 44. Задачи и проблемы синтаксического анализа
- 45. Метод рекурсивного спуска (РС-метод)
 - Подкласс грамматик, к которому применим РС-метод
 - Достаточные условия применения РС-метода
 - Расширение подкласса КС-грамматик, к которому применим РС-метод
 - Устранение левой рекурсии
 - Левая факторизация
 - Подстановка нетерминалов (терминализация)
 - ε -альтернатива
 - Канонический вид грамматики для применения РС-метода (достаточные условия применимости метода рекурсивного спуска для грамматик с ε -альтернативами)
 - Критерий применимости РС-метода
 - Последовательность применения правил преобразования грамматик к виду, пригодному для работы РС-методом
 - Пример применения метода рекурсивного спуска к модельному языку

- Обработка ошибок при рекурсивном спуске
- 46.Альтернативные методы распознавания контекстно-свободных языков
 - Табличные методы анализа
 - Алгоритмы анализа, расходуящие линейное время
 - LL(k)- и LR(k)-грамматики
 - LALR(1)-грамматики
 - Граматики простого и операторного предшествования
- 47.Задачи семантического анализа
 - Проверка семантических соглашений и контекстных условий
 - Дополнение внутреннего представления
 - Проверка правил программирования
 - Распределение имён по пространствам именования
- 48.Семантический анализ описаний, выражений и операторов
 - Семантический анализ описаний
 - Граматики с действиями
 - Контроль контекстных условий в выражениях
 - Контроль типов в операторах
- 49.Внутреннее представление программ, свойства языков внутреннего представления
 - Связные списочные структуры
 - Многоадресный код с явно именуемым результатом (тетрады)
 - Многоадресный код с неявно именуемым результатом (триады)
 - Префиксная запись
 - Постфиксная (инверсная польская) запись
 - Язык ассемблера и машинные команды
- 50.ПОЛИЗ выражений, алгоритм Дейкстры, алгоритм интерпретации ПОЛИЗ
- 51.ПОЛИЗ операций и операторов языков программирования
 - Лево- и право-ассоциативные операции
 - Префиксные унарные операции увеличения и уменьшения
 - Постфиксные унарные операции увеличения и уменьшения
 - Условные операторы и операторы цикла
 - Семантика условного выражения
- 52.Синтаксически управляемый перевод
- 53.Синтаксический и семантический анализатор модельного языка
 - Проектирование объектной модели анализатора
 - Семантический анализ описаний
 - Контроль контекстных условий в выражениях модельного языка
 - Оператор присваивания
 - Условный оператор и оператор цикла
 - Оператор чтения данных
 - Оператор вывода данных
 - Контроль типов в операторах модельного языка
- 54.Генерация ПОЛИЗ программы на модельном языке
- 55.Интерпретация ПОЛИЗ программы на модельном языке
- 56.Генерация кода
 - Генерация кода как альтернатива непосредственной интерпретации
 - Виды объектных программ
 - машинные коды в абсолютных адресах
 - перемещаемый машинный код
 - язык ассемблера объектной машины
 - другой язык программирования высокого уровня

57. Оптимизация в компиляторах

Машинно-независимая оптимизация

Линейные участки

Передача параметров при вызовах функций, девиртуализация вызовов

Оптимизация циклов

Примеры оптимизирующих преобразований

Машинно-зависимая оптимизация

Учёт регистровой структуры вычислительной аппаратуры,

Удаление излишних команд,

Оптимизация потока управления и удаление недостижимых участков программ,

Снижение “стоимости” программы,

Использование машинных идиом,

Слияние, дробление и развёртывание циклов, иногда требующееся из-за технических особенностей аппаратуры,

Учёт векторных и конвейерных свойств архитектуры.

58. Основные методы распределения памяти

Стратегия статического распределения памяти

Стратегия динамического распределения

Стековая дисциплина распределения

Дисциплина произвольного распределения

Технологии динамического распределения памяти

Определение занятости блока с помощью счётчика ссылок

Определение занятости блоков памяти с помощью пометок

Процесс “сборки мусора”

59. Компоненты классической системы программирования, их роль в общей системе

Среда разработки

Режим работы с командной строкой

Концепция командных файлов

Интегрированные системы программирования

Редакторы текстов

Пакетные редакторы

Контекстное редактирование, макрообработка текстов

Диалоговые редакторы (строчные и экранные)

Основные функции редактора в рамках интегрированной среды разработки

Лексический анализ “на лету” как функция текстовых редакторов

Редакторы связей: назначение, принципы работы

Загрузчики: основные функции, принципы работы

Библиотеки (статические и динамические)

Средства конфигурирования

Системы управления версиями

Средства отладки программ

Средства и стратегии тестирования программ

Профилировщики

Справочные системы

60. Основные типы библиотек

Библиотеки функций, процедур и макроопределений

Библиотеки системных (стандартных) программ

- Пакеты прикладных программ
- Библиотеки классов
 - совокупности независимых классов
 - иерархии классов
 - иерархии шаблонов классов
- Библиотеки компонентов
- 61. Критерии проектирования стандартных библиотек
 - Требования по составу
 - Требования по свойствам компонентов
- 62. Автоматизация построения лексических анализаторов
 - Регулярные множества
 - Свойства регулярных выражений
 - Построение лексических анализаторов по регулярным выражениям (LEX)
- 63. Автоматизация построения синтаксических анализаторов (YACC)

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Есть ли ошибки в реализации функций `B::g()` и `main()`? Если есть, **объясните**, в чем они заключаются. Для всех правильных операторов этих функций укажите, из какой области видимости выбираются участвующие в их записи имена, используя операцию разрешения области видимости `::`.

```
#include <iostream>
using namespace std;
int x = 0;      void f (int a, int b){ x = a + b; }
class A {      int x;
public:        void f () { x = 2; }
};
class B: public A {
public:        void f (int a) { ::x = a; }
               void g ();
};
void B::g()    { f ();          f (1);          f (5 , 1);          x = 2;      }
int main ()   { B b;          f (5);          f ('+', 6);          return 0; }
```

2. Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то **объясните**, в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Что будет выдано в стандартный канал вывода при работе программы?

```
class X {
public: virtual int g(double x) { h(); cout << "X:g" << endl; return 1;}
public: virtual int f(int x)   {      cout << "T:f" << endl; return 0;}

               void h()          { t(); cout << "X:h" << endl; }
               virtual void t()   {      cout << "X:t" << endl; } };
class Z: public X {
public:        int g(double y) { h(); cout << "Z:g" << endl; return 3;}
               virtual void h()   { t(1);cout << "Z:h" << endl; }
               virtual void t(int k) {      cout << "Z:t" << endl; } };
int main() { X a;      Z b;      X * p = &b;
              p -> g(1.5);      p -> h();      p -> t(5); }
```

3. Является ли тип данных `A` из задания 1. абстрактным типом данных (пояснить ответ)? Перегрузить для этого класса префиксную операцию `«++»` в соответствии с принятым для нее в C++ смыслом.

4. Для объектов класса `B` из задания 1 определена функция `ret`:

```
B ret (B & x, B & y) { return x; }
```

Какие конструкторы и деструкторы и в каком порядке будут вызываться при работе следующего фрагмента программы

```
... ; { B b; b = ret (b, b); } ...
```

5. Описать прототипы двух перегруженных функций `g` из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
g(1);          g('+', '+');          g(2.3);          g(3, "str");
```

6. Для каких конструкций C++ существует понятие шаблона? Привести пример реализации какой-либо шаблонной конструкции.

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
class X;
void F(X & x, int n);
class X { public: X() { try { F(*this, -2); cout << 1 << endl; }
                    catch (X) { cout << 2 << endl; }
                    catch (int) { cout << 3 << endl; } }
        X (X &) { cout << 12 << endl; }
};
class Y: public X {
    public: Y () { cout << 4 << endl; }
           Y (Y & a) { cout << 5 << endl; }
           Y () { cout << 6 << endl; }
};
void F(X & x, int n) { try { if (n < 0) throw x;
                          if (n > 10) throw 1;
                          cout << 7 << endl; }
                    catch (int) { cout << 8 << endl; }
                    catch (X&) { cout << 9 << endl;
                                throw;
                              }
}
int main() { try { Y a; }
            catch (...) { cout << 10 << endl; }
            cout << 11 << endl;
}
```

8. Есть ли ошибки в тексте приведенной программы? Можно ли исправить описание класса, не вводя дополнительных членов, чтобы программа стала верной? Если да, то как?

```
class A {
    public: int y;
           void f () { cout << "f\n"; }
};
int A::y;
int main () { A::y = 1;
             const A a;
             a.f();
             return 0; }
```

9. Дать определение функции – друга класса. Привести пример класса, имеющего друга, описание этого друга, и пример его использования.

10. STL: Написать функцию, которая удваивает (добавляет еще один такой же) каждый элемент заданного контейнера-списка `list<int>`, а затем распечатывает его элементы в обратном порядке.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Есть ли ошибки в реализации функций `D::h()` и `main()`? Если есть, **объясните**, в чем они заключаются. Для всех правильных операторов этих функций укажите, из какой области видимости выбираются участвующие в их записи имена, используя операцию разрешения области видимости `::`.

```
#include <iostream>
using namespace std;
double a = 0;
void f (double x){a = x;}
struct B { double a;
          void f () {a = 2;}
};
class D: B {
public:    void f (int a){ ::a = a; }
          void h ();
};
void D::h() { f(1.2);      f();      a = 2;      }
int main () { D d;        f();        f(6);        return 0; }
```

2. Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то **объясните**, в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Что будет выдано в стандартный канал вывода при работе программы?

```
class T {
public: virtual int f(int x) { cout << "T::f" << endl; return 0; }
        void g() { f(1); cout << "T::g" << endl; }
        virtual void h() { g(); cout << "T::h" << endl; } };
class S: public T {
public: int f(double y) { cout << "S::f" << endl; return 2; }
        virtual void g() { f(1); cout << "S::g" << endl; }
        virtual void h() { g(); cout << "S::h" << endl; } };
int main() { T t; S s; T *p = &s;
             p -> f(1.5); p -> g(); p -> h(); }
```

3. Является ли тип данных `B` из задания 1. абстрактным типом данных (пояснить ответ)? Перегрузить для этого класса постфиксную операцию уменьшения `--` в соответствии с принятым для нее в C++ смыслом.

4. Для объектов класса `D` из задания 1. определена функция `empty`:

```
void empty (D & a, D b) { }
```

Какие конструкторы и деструкторы и в каком порядке будут вызываться при работе следующего фрагмента программы

```
... ; { D d; empty (d, d); } ...
```

5. Описать прототипы двух перегруженных функций `f` из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
f();          f("abc");          f(2);          f('+', 3);
```

6. Дать определение абстрактного класса в C++. Привести пример. Можно ли описывать конструкторы, вводить неконстантные члены-данные, заводить объекты абстрактного класса?

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
class A { public: A () { cout << 1 << endl; }
};
class B: public A {
public: B (int n) { try { if (n == 0) throw * this;
                      if (n > 11) throw 11; }
                      catch (int) { cout << 2 << endl; }
                      catch (B&) { cout << 3 << endl; throw; }
                      cout << 4 << endl; }
                      B (B&) { cout << 5 << endl; }
                      ~B () { cout << 6 << endl; }
};
int main() { try { B b(0); B c (3); }
            catch (...) { cout << 7 << endl; }
            cout << 8 << endl;
}
```

8. Есть ли ошибки в тексте приведенной программы? Можно ли исправить описание класса, не вводя дополнительных членов, чтобы программа стала верной? Если да, то как?

```
class X { public: void g () { cout << "g\n"; }
            int h (int n) { cout << "f\n"; return n; }
};
int main () { int k;
              const X x;
              X::g();
              k = x.h(5);
              return 0;
}
```

9. Описать суть работы операции *dynamic_cast*. Привести пример её использования.

10. STL: Написать функцию, которая удаляет каждый второй элемент заданного контейнера-вектора `v<char>`, а затем распечатывает его элементы в обратном порядке.

Коллоквиум, 2008 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка задачи – 10 баллов (всего 100 баллов)

1.

```
void B::g() { f() // ошибка, эта функция не видна
             B::f(1); f(5,1) // ошибка, эта функция не видна
             *--2 } // ошибка: доступ к x в базовом классе закрыт
int main () { B b; f(5) // ошибка, эта функция не видна
             ::f ('+', 6); return 0; }
```

Критерии: За каждую не найденную ошибку -2. За каждую лишнюю ошибку или неверно написанное с ":" имя (это строго взаимоисключающие ошибки) -2.

2.

```
int main() { X a; Z b; X * p = &b;
             p -> g(1.5); // печатается Z::t Z::h Z::g
             p -> h(); // печатается X::t X::h
             p -> t(5) } // ошибка, X::t - без параметров
```

Критерии: За каждую не найденную ошибку, лишнюю ошибку или неправильно вызванную функцию -2. (может быть больше 10 баллов, но причины ошибок пересекаются).

3. Да, так как не содержит открытых полей-данных. Метод класса A:

```
A & operator ++ () { x++; return * this; }
```

Критерии: Нет понимания концепции АТД -3. Нет ссылки в возвращаемом результате -1. Неверно реализована операция -7.

4.

```
A() B() A() B(B &) ~B() ~A() ~B() ~A()
```

Критерии: За каждую ошибку (пропущенную, лишнюю или неверную печать) -2.

5.

```
void g (int a, int b = 0 ); void g (int, const char * = 0 );
```


 Возможны варианты с **double**, **char...**

Критерии: За каждую ошибку -5. Функций не две -10.

6. Полный балл, если приведен пример шаблона класса, либо шаблона функции.

Критерии: Нет либо шаблона функций, либо классов -5, ошибки в примере существенные -4, несущественные -2.

7.

```
12 - 9 - 12 - 2 - 4 - 6 - 11 -
```

Критерии: За каждую неверную (лишнюю или пропущенную) печать -2.

8. Исправления:

```
class A { public: static int y;
               void f () const {cout << "f\n";} };
```

Критерии: Ответ "нет" -10, нет слова **static** -5, не добавлено слово **const** -5.

9. Друг класса – это функция, не являющаяся членом этого класса, но имеющая доступ к закрытым и защищенным членам класса.

```
class X { int a; public: friend void fff (X*, int); };
void fff ( X * p, int i ) { p -> a = i; }
void f() { X obj; fff (& obj, 10); }
```

Критерии: Нет нужного определения -8, существенные ошибки в примере -5.

10.

```
void g (list <int> &lst) {
    list <int> :: iterator p = lst.begin ();
    while ( p != lst.end()) { lst.insert(p, *p); p++; }
    list <int> :: reverse_iterator rp = lst.rbegin ();
    while ( rp!=lst.rend ()) { cout << *rp << ' '; rp++; }
    cout << endl;
}
```

Критерии: За каждую существенную ошибку -3.

Коллоквиум, 2008 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка задачи – 10 баллов (всего 100 баллов)

```
1. void D::h() { f(1.2); // D::f
                f();    // ошибка эта функция f не видна
                a = 2;   } // B::a = 2
int main () { D d;      f (); //ошибка
                f (6);   //::f
                return 0; }
```

Критерии: За каждую не найденную ошибку -2. За каждую лишнюю ошибку или неверно написанное с ":" имя (это строго взаимоисключающие ошибки) - 2.

```
2. int main() { T t; S s; T *p = &s; p -> f(1.5); // T::f
                p -> g(); // T::f T::g
                p -> h(); } // S::f S::g S::h
```

Критерии: За каждую не найденную ошибку, лишнюю ошибку или неправильно вызванную функцию – 2. (может быть > 10 баллов, но причины ошибок пересекаются).

3. Нет, так как содержит открытые поля-данные. Метод класса B:

```
B operator -- (int t) { B b = *this; a = a - 1; return b; }
```

Критерии: Нет понимания концепции АТД -3. Есть ссылка в возвращаемом результате -1. Неверно реализована операция -7.

```
4. B () D () B(B &) D (D &) ~D() ~B() ~D() ~B()
```

Критерии: За каждую ошибку (пропущенную, лишнюю или неверную печать) -2

```
5. void f (const char * a = 0 );
    void f (int a, int b = 0); // Возможны варианты с char, ...
```

Критерии: За каждую ошибку -5. Функций не две -10.

6. Абстрактным называется класс, в котором есть хотя бы одна чистая виртуальная функция. Нельзя создавать объекты абстрактного класса, но можно описывать указатели на абстрактный класс, использовать абстрактный класс в качестве базового класса, вводить неконстантные члены-данные, описывать конструкторы. В работе должен быть дан пример абстрактного класса.

Критерии: существенные ошибки в примере -4, в работе есть только пример -5, в работе есть только определение -5, за каждый неверный ответ на последний вопрос - 2.

```
7. 1 - 1 - 5 - 3 - 7 - 6 - 8 -
```

Критерии: За каждую неверную (лишнюю или пропущенную) печать -2.

```
8. Исправления: class X { public: static void g () {cout << "g\n";}
    int f (int n) const {cout << "f\n"; return n; } };
```

Критерии: Ответ "нет" -10, нет слова **static** -5, нет слова **const** -5.

9. Операция **dynamic_cast** реализует приведение указателей или ссылок на полиморфные типы в динамическом режиме. При неудачной попытке приведения типов результатом выполнения операции над указателем является 0. Если же операндом являлась ссылка, возбуждается исключительная ситуация **bad_cast**. Пример:

```
Base *bp, b_ob; Derived *dp, d_ob; bp = &d_ob;
dp = dynamic_cast <Derived *> (bp);
```

Критерии: Совсем неверное описание операции -8, нет полиморфного типа -4, неверный пример -5.

```
10. void f (vector <char> &v) {
    vector <char> :: iterator p = v.begin ();
    while (p != v.end()) { p++; if (p != v.end()) { v.erase(p); } }
    vector < char > :: reverse_iterator rp = v.rbegin ();
    while (rp != v.rend ()) { cout << *rp << ' '; rp++; }
    cout << endl;
}
```

Критерии: За каждую существенную ошибку -3.

1	2	3	4	5	6	7	8	9	10

1. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
class X;
void F (X& x, int n);
class X {
    public: X () { try { F (*this, -2); cout << 1 << endl; }
                catch (X&) { cout << 2 << endl; }
                catch (int n) {
                    F (*this, -n); cout << 3 << endl; } }
    X (X&) { cout << 4 << endl; }
    ~X () { cout << 5 << endl; }
};
void F (X & x, int n) { try { if (n < 0) throw x;
                             if (n > 1) throw n;
                             cout << 6 << endl; }
    catch (int) { cout << 7 << endl;
                 throw; }
    catch (X&) { cout << 8 << endl;
                 throw; }
}
void main() { try { X a; }
             catch (...) { cout << 9 << endl; }
             cout << 10 << endl;
}
```

Ответ: _____

2. Дать определение класса, являющегося другом некоторого другого класса. Привести пример класса, имеющего дружественный класс, описание этого друга, и пример его использования.

3. STL: Написать функцию, которая считает количество положительных элементов заданного контейнера-списка `list<int>`, а затем распечатывает это значение (выдает в стандартный поток `cout`).

4. Дать определение абстрактного класса. Привести небольшой пример, поясняющий использование абстрактных классов в программе на C++.

5. Есть ли ошибки в тексте приведенной программы? Можно ли исправить определение класса и функции, чтобы программа стала верной? Если да, то как?

```
#include <iostream.h>
class A { public: int y; void f (int t) { cout << "f\n"; } };
g () { const A a; a.f(A::y); return 0; }
```

6. Дана грамматика G :

$S \rightarrow Bc$
 $S \rightarrow dSc$
 $dBc \rightarrow dAc$
 $dA \rightarrow dAd$
 $A \rightarrow d$

(а) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$.

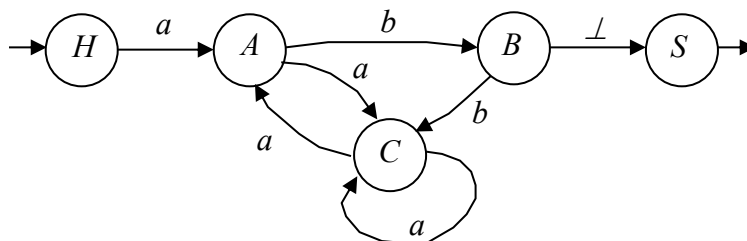
Ответ:

(б) Каким из перечисленных классов принадлежит язык $L(G)$? Ответ:

Класс $\mathfrak{Z}$	$L(G) \in \mathfrak{Z}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
регулярные языки	
языки типа 0	

(в) Определение неукорачивающей грамматики. Ответ:

7. Конечный автомат $\mathcal{A}$ задан в виде диаграммы состояний:



(а) Показать, что автомат $\mathcal{A}$ не является детерминированным.

(б) По автомату $\mathcal{A}$ с помощью соответствующего алгоритма построить эквивалентный детерминированный автомат $\mathcal{A}_D$.

(в) По автомату $\mathcal{A}_D$ построить эквивалентную левوليнейную грамматику G .

8. Дана грамматика:

$G_1 : S \rightarrow cAb$
 $A \rightarrow dS \mid d$

Вставить в G_1 действия вида $\langle cout \ll \dots \rangle$ по переводу цепочек языка $L_1 = L(G_1)$ в цепочки языка $L_2 = \{b(a)^{2k} \mid k \geq 1\}$ так, чтобы символов a в цепочке из L_2 было в два раза больше, чем символов d в соответствующей цепочке из L_1 . Ответ:

9. Записать в постфиксной записи оператор программы на Си++ (семантика присваивания как в модельном языке: только побочный эффект, без возвращаемого значения операции).

`if (i > 0) x = j; else if (j > 0) x = i; else if (i < j) x = j;`

Ответ:

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	

10. Какие виды отношений между классами поддерживают большинство объектно-ориентированных языков программирования? Коротко охарактеризуйте их.

Экзамен, 2008 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка задачи – 10 баллов (всего 100 баллов)

1. Ответ: в стандартный канал вывода будет выдано: 4 - 8 - 2 - 5 - 5 - 10 -

Критерии: За каждую неверную (лишнюю или пропущенную) печать: -2 балла. Если числа записаны в строчку без указания «перехода на новую строку», через пробел или через запятую, то за это не снижать).

2. Ответ: Дружественный класс – это класс, все методы которого являются друзьями данного класса.

Критерии: Нет определения – снижение на 5 баллов. Нет примера или пример не разъясняет нужное понятие – снижение на 5 баллов. Пример дан только на описание (без использования) дружественного класса – снижение на 3 балла.

3. STL: Функция f считает количество положительных элементов заданного контейнера-списка `list<int>`, а затем выдает это значение в стандартный поток `cout`.

Ответ:

```
void f (list <int> & l) { int count = 0;
    list <int> :: iterator p = l.begin ();
    while (p != l.end()) if (* p ++ > 0) count ++;
    cout << count << endl;
}
```

Критерии: За каждую существенную ошибку – снижение на 3 балла.

4. Ответ: Класс, в котором есть хотя бы одна чистая виртуальная функция, называется абстрактным классом. Нельзя (1) создавать объекты абстрактного класса, но можно (2) описывать указатели на абстрактный класс, (3) использовать абстрактный класс в качестве базового класса, (4) вводить неконстантные члены-данные, (5) описывать конструкторы.

Критерии: Ошибки в примере (существенные, то есть не показывающие перечисленных свойств абстрактных классов) – снижение на 1 балл за каждое пропущенное в примере свойство. Есть только пример – снижение на 5 баллов, только определение (выделенное шрифтом) – снижение на 5 баллов.

5. Ответ:

```
#include <iostream.h>
class A { public: int y; void f (int t) { cout << "f\n"; }
/* конструктор умолчания: */ A() {y = 1;} /* здесь любое значение */ };
int g () { A a; a.f(a.y); return 0; }
```

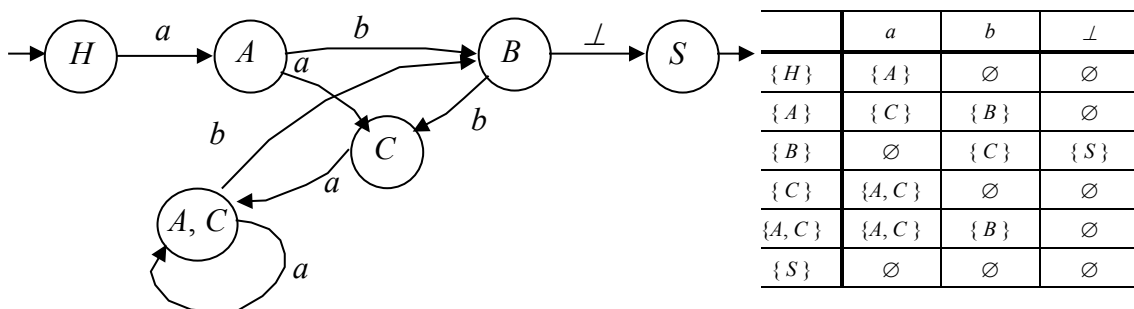
Критерии: За каждую не найденную или лишнюю ошибку – снижение на 3 балла.

6. Ответ: (а) $k = 1$ (третье правило не удовлетворяет типу 2), (б) $L(G) = \{ d^{n+k} c^n \mid n \geq 1, k \geq 1 \}$, КС – "да", КЗ – "да", языки типа 0 – "да", регулярные языки – "нет", (в) Грамматика $G = \{ V_T, V_N, P, S \}$ – *неукорачивающая*, если каждое правило из P имеет вид: $\alpha \rightarrow \beta$, где $\alpha \in (V_T \cup V_N)^+$, $\beta \in (V_T \cup V_N)^+$ и $|\alpha| \leq |\beta|$.

Дополнение. Если студент добавил к определению, что неукорачивающая грамматика может содержать, кроме прочих, единственное правило с пустой правой частью $S \rightarrow \varepsilon$, **но при этом** S не содержится в правых частях правил, такой ответ верен – баллы не снижать.

Критерии: (а) нет ответа; неверный ответ: -3 (обоснование не обязательно), (б) за каждый неверный или отсутствующий ответ: -1, то есть всего до -4 за этот пункт (обоснование и формула языка не обязательны), (в) определение отсутствует или с ошибками: -3.

7. **Ответ:** (а) Автомат $\mathcal{A}$ не является детерминированным, так как из состояния C по символу a возможен переход в более, чем одно состояние: (C, A) , (б) По автомату $\mathcal{A}$ с помощью соответствующего алгоритма можно построить эквивалентный детерминированный автомат $\mathcal{A}_D$ и таблицу переходов:



(в) Левополинейная грамматика G автомата $\mathcal{A}_D$.

G : $S \rightarrow B\perp$
 $B \rightarrow Ab \mid S_{\{A,C\}} b$
 $S_{\{A,C\}} \rightarrow S_{\{A,C\}} a \mid Ca$
 $C \rightarrow Bb \mid Aa$
 $A \rightarrow a$

Замечание. В качестве промежуточного этапа в грамматике допускается правило вида $H \rightarrow \varepsilon$, которое устраняется на последнем шаге построения. Результат должен совпасть с тем, что приведен слева (с точностью до обозначений нетерминалов).

Критерии: (а) нет ответа; неверное обоснование: **-3**, (б) нет ответа; неэквивалентный автомат; недетерминированный автомат: **-4**, (в) нет ответа; неэквивалентная автомату из пункта (б) грамматика: **-3**, эквивалентная, но не левополинейная грамматика: **-1**.

8. **Ответ:** Соглашение: запись $\langle a \rangle$ является сокращением записи $\langle cout \ll "a" \rangle$. Грамматика с действиями (возможные ответы):

(1) $S \rightarrow cAb \langle aa \rangle$ (2) $S \rightarrow cAb$ (3) $S \rightarrow cAb \langle a \rangle$
 $A \rightarrow dS \mid d \langle b \rangle$ $A \rightarrow dS \langle aa \rangle \mid d \langle baa \rangle$ $A \rightarrow dS(a) \mid d \langle ba \rangle$

Критерии: (а) нет ответа; неверное обоснование: **-3**, (б) нет ответа; неэквивалентный автомат; недетерминированный автомат: **-4**, (в) нет ответа; неэквивалентная автомату из пункта (б) грамматика: **-3**, эквивалентная, но не левополинейная грамматика: **-1**. За каждое неправильно вставленное (или пропущенное) действие вида $\langle cout \ll \dots \rangle$ **-4**. Если, кроме действий по печати символов $\langle cout \ll \dots \rangle$, есть иные вычисления ставить **0**.

9. **Ответ:**

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12		13		
	i	0	>	11	!F	<u>x</u>	j	=	29	!	j	0		>		
	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
	21	!F	<u>x</u>	i	=	29	!	i	j	<	29	!F	<u>x</u>	j	=	<

Критерии: Неправильное представление о ПОЛИЗ операторов, перестановка операций, замена одних операций на другие, оптимизация переходов, инвертирование условия, пропуск указания на адресность значения в присваивании – снижение до **0** баллов. Ошибка в подсчете адреса переходов **-4** балла. Переход по лжи разрешается без пояснений. Разрешается переход по истине, если правильно описана семантика этой операции.

10. **Ответ:** ассоциация (классы концептуально взаимодействуют друг с другом); наследование («частное – общее», «обобщение – специализация», «есть», «is a»); агрегация («часть – целое», «содержит», «has a»); использование (в функции класса используются локальные объекты, функции другого класса); инстанцирование (связь между шаблоном класса и классом-результатом генерации по шаблону).

Критерии: Нет понимания сути отношений между классами – снижение до **0** баллов. За пропуск любого из видов отношений **-2** балла, за пропуск пояснения к какому-либо виду отношений (когда указано только наименование) – снижение на **1** балл.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Есть ли ошибки в реализации функций `C::g()` и `main()`? Если есть, исправьте найденные ошибки, используя операцию разрешения области видимости `::`. Для всех операторов этих функций, используя операцию разрешения области видимости `::`, также укажите, из какой области видимости выбираются участвующие в их записи имена.

```
int x = 0;
int f (int a, int b)          {      return   x = a + b; }
class A { int x;
    public: A (int n = 1)          { x = n; }
           int f () {      return   ::x = x; }    };
class B { int x;
    public: B (int n = 2)          { x = n; }    };
class C: public A, public B { int x;
    public: int f (int a) { return   ::x = x; }
           void g ();
void C::g() { x = f ();
             f (3);
             x = f (4, 5);
             x = 6;
}
int main () { C c;
             c.f ();
             c.f (7);
             x = f ('8', 9);
             return -1;
}
```

2. Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Что будет выдано в стандартный поток вывода при работе программы?

```
class A { public:
    virtual void f (int x) { h (x); cout << "A::f," << x << endl; }
    void g ()          { h (0); cout << "A::g" << endl; }
    virtual void h (int k) {      cout << "A::h," << k << endl; } };
class B: virtual public A { public:
    void f (int y) { h (y); cout << "B::f," << y << endl; }
    void g ()          { h (1); cout << "B::g" << endl; }
    void h (int k) {      cout << "B::h," << k << endl; } };
int main() { A a; B b; A * p = & b;
            p -> f (2);
            p -> g ();
            p -> h ();
            p -> h (3); }
```

3. Описать прототипы двух перегруженных функций `f()` из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
f (0, 1);           f (1, 0);           f (0, "m");         f ("n", 0);
```

4. Для объектов из задания 1 определить, тела каких конструкторов и деструкторов (возможно пустые, если они не определены явно) и в каком порядке будут исполнены при работе следующего фрагмента программы:

```
int main () { C c; B b = c; A a = c; }
```

5. Для класса комплексных чисел `template<class T> class complex { T r; T i; ... };` написать два варианта реализации операции `"*"`, выполняющей умножение одного комплексного числа на другое методом класса и функцией-другом этого класса.

6. Дать определения абстрактного типа данных и абстрактного класса в языке Си++. Есть ли различия в этих понятиях? Можно ли создавать объекты и описывать конструкторы этих типов?

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct X; void f(X & x, int n);
struct X {      X() {      try { f(*this, -1); cout << 1 << endl; }
                catch (X)      { cout << 2 << endl; }
                catch (int)     { cout << 3 << endl; } }
        X (X &)      { cout << 4 << endl; }
        ~X ()        { cout << 5 << endl; } };
struct Y: X { Y () {      f(*this, -1); cout << 6 << endl; }
        Y (Y &)      { cout << 7 << endl; }
        ~Y ()        { cout << 8 << endl; } };
void f(X & x, int n) { try { if (n < 0) throw x;
                            if (n > 0) throw 1;
                            cout << 9 << endl; }
        catch (int)      { cout << 10 << endl; }
        catch (X& a)     { cout << 11 << endl;
                            f(a, 1);
                            cout << 12 << endl;
                            throw; } }

int main() { try { Y a; }
        catch (...)      { cout << 13 << endl;
                            return 0; }
        cout << 14 << endl;

        return 0; }
```

8. Есть ли синтаксические ошибки в тексте приведенной программы? Можно ли исправить описание класса, не вводя дополнительных членов и не убирая имеющиеся, чтобы программа стала верной? Если да, то как?

```
class A { static int i;
        static void f () { g (); cout << "f ()" << endl; }
        void g () { if (i >= 0) i = -1, f ();
                    cout << "g ()" << endl; } };

int A::i = 1;
int main () { A::i = 1;
        A a;
        a.f();
        a.i = 0; }
```

9. Описать последовательность отождествления типов параметров перегружаемых функций для случая функций с одним параметром. Приведите примеры для каждого из шагов последовательности.

10. Написать функцию *g()* с тремя параметрами: непустой контейнер-вектор типа *vector<int>*, непустой контейнер-список типа *list<int>*, целое число – шаг по первому контейнеру. Функция должна, последовательно проходя по списку от начала к концу, перезаписывать на место очередного его элемента соответствующий очередному шагу элемент вектора (сам вектор при этом не изменяется), а затем распечатывать элементы списка в обратном порядке. Функция возвращает количество изменённых элементов списка.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Для класса комплексных чисел `template<class T> class complex { T r; T i; ... };` написать реализацию операции `"=="`, выполняющей сравнение двух комплексных чисел методом класса и функцией-другом класса.

2. Объект x принадлежит классу X , для которого определены конструктор преобразования типа `int` в класс X и функция преобразования класса X в тип `int`. Описать прототипы двух перегруженных функций $f()$ из некоторой области видимости, для которых будут верны следующие обращения к ним:

`f ("p");      f (x, 0);      f (0, 0);      f (x, "q");      f (1, "r");`

3. Есть ли ошибки в реализации функции `main()`? Удалите неверные операторы из этой функции (если они есть). Для всех правильных операторов этой функции, используя операцию разрешения области видимости `"::"`, также укажите, из какой области видимости вызываются используемые в них функции $f()$.

```
int x = 0;
int f (int a) { return x = a; }
class A { int x;
public: A (int n = 1)          { x = n; }
      int f (int a)          { return x = a; } };
class B { int x;
public: B (int n = 2)          { x = n; }
      virtual int f ()        { return x = ::x; } };
class C: public A, public B { int x;
public: int f (int a) { return x = a; }
      int f ()        { return x = ::x; } };
int main () { A a; B b; C c;
             A * pa = & a; B * pb = & b; C * pc = & c;
             x = f (3);
             x = pa -> f ();
             x = pa -> f (4);
             pa = pc; x = pa -> f ();
             x = pa -> f (5); x = pb -> f ();
             x = pb -> f (6);
             pb = pc; x = pb -> f ();
             x = pb -> f (7);
             x = pc -> f ();
             x = pc -> f (8);
             return -1; }
```

4. Написать функцию $g()$ с тремя параметрами: непустой и неизменяемый контейнер-список типа `list<long int>`, непустой контейнер-вектор типа `vector<long int>`, целое число – шаг по второму контейнеру. Функция должна копировать отрицательные элементы списка с шагом, равным 1, в уже имеющийся контейнер-вектор, от его начала к концу с шагом, равным третьему параметру, а затем распечатывать элементы вектора в прямом порядке. Функция возвращает количество измененных элементов вектора.

5. Описать особенности процесса отождествления типов параметров перегружаемых функций с несколькими параметрами. Приведите по одному примеру перегрузки функции с несколькими параметрами, в которых (а) не удастся найти ни одной подходящей функции, (б) находится одна подходящая функция, (в) находится несколько подходящих функций.

6. Для объектов из задания 3 определить, тела каких конструкторов и деструкторов (возможно пустые, если они не определены явно) и в каком порядке будут исполнены при работе следующего фрагмента программы:

```
int main () { struct D { A a; C c; }; D d; }
```

7. Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то **объясните**, в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Что будет выдано в стандартный поток вывода при работе программы?

```
class C { public:
    virtual void f (int x) { h (x); cout << "C::f," << x << endl; }
    virtual void g ()      { h (0); cout << "C::g"      << endl; }
    virtual void h ()      { cout << "C::h"          << endl; }
    virtual void h (int k) { h (); cout << "C::h," << k << endl; } };
class D: public C { public:
    virtual void f (int y) { h (y); cout << "D::f," << y << endl; }
    virtual void g ()      { h (1); cout << "D::g"      << endl; }
    virtual void h ()      { cout << "D::h"          << endl; }
    virtual void h (int k) { h (); cout << "D::h," << k << endl; } };
int main() { C c; D d; C * p = &d;
            p -> f (2); p -> g (); p -> h (); p -> h (3); }
```

8. Дать определение полиморфизма. Описать различные виды полиморфизма в Си++. Привести необходимые примеры фрагментов программ.

9. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct X; void f(X & x, int n);
struct X { X() { try { f(*this, 0); cout << 1 << endl; }
                catch (X) { cout << 2 << endl; }
                catch (int) { cout << 3 << endl; } }
        X (X &) { cout << 4 << endl; }
        ~X () { cout << 5 << endl; } };
struct Y: X { Y () { f(*this, -1); cout << 6 << endl; }
            Y (Y &) { cout << 7 << endl; }
            ~Y () { cout << 8 << endl; } };
void f(X & x, int n) { try { if (n < 0) throw x;
                          if (n > 0) throw 1;
                          cout << 9 << endl; }
                catch (int) { cout << 10 << endl; }
                catch (X& a) { cout << 11 << endl;
                            f(a, 1);
                            cout << 12 << endl;
                            throw;
                            }
}
int main() { try { Y a; } catch (...) { cout << 13 << endl; return 0; }
            cout << 14 << endl; return 0; }
```

10. Есть ли синтаксические ошибки в тексте приведенной программы? Если можно исправить описание класса, не вводя дополнительных членов, чтобы программа стала верной, то как?

```
class A { static int i;
        void f () { if (i >= 0) i = -1, g ();
                    cout << "f ()" << endl; }
        void g () { f (); cout << "g ()" << endl; } };
int A::i = 1;
int main () { A::i = 1; const A a; a.f(); a.i = 0; }
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то **объясните**, в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Что будет выдано в стандартный поток вывода при работе программы?

```
class T {
public: virtual void f(int x) { h(); cout << "T::f," << x << endl; }
        void g() { h(); cout << "T::g" << endl; }
        virtual void h() { cout << "T::h" << endl; } };
class U:virtual public T {
public: void f(int y) { h(y); cout << "U::f," << y << endl; }
        virtual void g() { h(0); cout << "U::g" << endl; }
        void h(int k) { cout << "U::h," << k << endl; } };
int main() { T t; U u; T * p = &u;
            p -> f(1); p -> g(); p -> h(); p -> h(2); }
```

2. Есть ли ошибки в реализации функций `B::g()` и `main()`? Если есть, исправьте неверные операторы из этой функции, используя операцию разрешения области видимости `::`. Для всех операторов этих функций укажите, из какой области видимости вызываются используемые в них функций `f()` и `g()`.

```
int x = 0;
int f() { return x = 1; }
class A { int x;
public: A ( int n = 2) { x = n; }
        int f() { return x = 3; }
        int f(int a, int b) { return x = a % b; } };
class B: public A { int x;
public: int f(int a) { return x = a; }
        int f(int a, int b) { return x = a + b; }
        int g(A * a, B * b); };
int B::g(A * pa, B * pb) { x = f();
                          x = f(5);
                          x = f(6, 6);
                          x = A::f(5);
                          return -1; }
int main() { B a;
            x = a.f();
            x = a.f(7);
            return a.g(&a, &a); }
```

3. Привести по одному примеру перегрузки бинарной, унарной префиксной и постфиксной операций (всего 3 примера). Указать особенности перегрузки операций присваивания. Можно ли изменить приоритет операции при её перегрузке?

4. Для класса комплексных чисел `template<class T> class complex {T r; T i; ... };` написать два варианта реализации префиксной операции увеличения `"++"`, выполняющей увеличение на 1 действительной части комплексного числа методом класса и функцией-другом класса.

5. Данные каких типов могут подвергаться динамическому и статическому приведению? В чём проявляется отличие выполнения динамического приведения типов для указателей и ссылок? Какие свойства указателей и ссылок влияют на это отличие? Приведите примеры фрагментов программы с использованием подобных операций.

6. Написать функцию `g()` с тремя параметрами: непустой и неизменяемый контейнер-вектор типа `vector<double>`, непустой контейнер-список типа `list<double>`, целое число – шаг по первому контейнеру. Функция должна сравнивать элементы списка, выбираемыми от его начала с шагом, равным 1, с элементами вектора, выбираемыми от начала с шагом, равным третьему параметру. Если обнаруживается несовпадение очередной выбранной пары, то в список в текущем месте вставляется отсутствующий элемент. Изменённый список распечатывается в обратном порядке. Функция возвращает количество элементов, вставленных в список.

7. Для объектов из задания 2 определить, тела каких конструкторов и деструкторов (возможно пустые, если они не определены явно) и в каком порядке будут исполнены при работе следующего фрагмента программы:

```
int main () { A a; class C { B b; A a; public: C():b(), a(b){} }; C c; }
```

8. Описать прототипы двух перегруженных функций $f()$ из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
f (10000000000000);    f (1);    f ();    f (0, 0);    f ("t");    f (1, "u");
```

9. Есть ли синтаксические ошибки в тексте приведенной программы? Если можно исправить описание класса, не вводя дополнительных членов, чтобы программа стала верной, то как?

```
class A { static int i;
        void f () const { if (i < 0) g (i);
                          cout << "f ()" << endl; }
        void g (int & n) { i = n; f (); cout << "g ()" << endl; } };
int A::i = 1;
int main () { const A a; a.g (2); }
```

10. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct X; void f(X & x, int n);
struct X { X() { try { f(*this, -1); cout << 1 << endl; }
                  catch (X) { cout << 2 << endl; }
                  catch (int) { cout << 3 << endl; }
                }
        X (X &) { cout << 4 << endl; }
        ~X () { cout << 5 << endl; } };
struct Y: X { Y () { f(*this, 1); cout << 6 << endl; }
             Y (Y &) { cout << 7 << endl; }
             ~Y () { cout << 8 << endl; } };
void f(X & x, int n) { try { if (n < 0) throw x;
                           if (n > 0) throw 1;
                           cout << 9 << endl; }
                    catch (int) { cout << 10 << endl; }
                    catch (X& a) { cout << 11 << endl;
                                   f(a, 0);
                                   cout << 12 << endl;
                                   throw; } }

int main() { try { Y a; }
            catch (...) { cout << 13 << endl;
                          return 0; }
            cout << 14 << endl;
            return 0; }
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Есть ли синтаксические ошибки в тексте приведенной программы? Можно ли исправить описание класса, не вводя дополнительных членов, чтобы программа стала верной? Если да, то как?

```
class A { int i;
    void f (int & n) { if (i == n) cout << "f (& n)" << endl; }
    void f (int * n) { if (i == * n) cout << "f (* n)" << endl; } };
int main () {
    const A a;
    a.f (a.i);
    a.f (& a.i); }
```

2. Описать прототипы двух перегруженных функций $f()$ из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
f ("a");    f (1.0);    f ();    f (0, 0);    f ("b", 1);    f (1, "0.1");
```

3. Есть ли ошибки в определениях классов A , B и C , препятствующие правильной компиляции функций $B::g()$ и $main()$? Если такие ошибки есть, предложите такие исправления в классах, чтобы реализации указанных функций стали правильными. Для всех операторов этих функций, используя операцию разрешения области видимости `::`, укажите, из какой области видимости вызываются используемые в них функции $f()$ и $g()$.

```
int x = 0;
int f (int a) { return x = a; }
class A { int x;
    public:    A (int n = 1)          { x = n; }
              int f (int a) { return x = a; } };
class B { int x;    B (int n = 2)      { x = n; }
    public:    int f (int a) { return x = a; }
              void g ();
};
class C: public A, B { int x;
    public:    int f (int a) { return x = a; } };
void B::g () { x = f (3);
              x = f (4);
              x = f (5);
};
int main () { A a; B b; C c;
              A * pa = & a; B * pb = & b; C * pc = & c;
              x = pa -> f (7);
              pa = pc; x = pa -> f (8);    x = pb -> f (7);    x = f (8);
              pb = pc; x = pb -> f (7);    x = pc -> f (8);    c.g ();
              return -1;
}
```

4. Описать последовательность действий при возбуждении исключительной ситуации. Привести пример фрагмента программы.

5. Для класса комплексных чисел `template<class T> class complex {T r; T i; ... };` написать два варианта реализации постфиксной операции уменьшения `--`, выполняющей уменьшение на 1 действительной части комплексного числа методом класса и функцией-другом класса.

6. Какие виды конструкторов могут присутствовать в определениях классов Си++? Каково их назначение? Привести примеры определения и использования конструкторов разных видов.

7. Для объектов из задания 3, строящихся на основе исходных или (если надо) исправленных определений классов *A*, *B* и *C*, определить, тела каких конструкторов и деструкторов (возможно пустые, если они не определены явно) и в каком порядке будут исполнены при работе следующего фрагмента программы:

8. Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Что будет выдано в стандартный поток вывода при работе программы?

```
class R { public:
    virtual void f (int x) { g (); cout << "R::f," << x << endl; }
    void g () { h (); cout << "R::g" << endl; }
    virtual void h () { cout << "R::h" << endl; } };
class S: virtual public R { public:
    void f (int x) { g (); cout << "S::f," << x << endl; }
    void g () { h (); cout << "S::g" << endl; }
    void h () { cout << "S::h" << endl; } };
int main() { R r; S s; R * p = & s;
    p -> f (0); p -> g (); p -> g (1);
    p -> h (); p -> h (2);
}
```

9. Написать функцию *g()* с тремя параметрами: непустой и неизменяемый контейнер-вектор типа *vector<float>*, непустой контейнер-список типа *list<float>*, целое число – шаг по первому контейнеру. Функция должна исследовать элементы списка, выбираемые от его конца с шагом, равным 1, и элементы вектора, выбираемые от его начала с шагом, равным третьему параметру. Если обнаруживаются пары элементы разных знаков, то у текущего элемента списка должен меняться знак. Изменённый список распечатывается в прямом порядке. Функция возвращает общее количество неотрицательных элементов списка.

10. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct X; void f(X & x, int n);
struct X { X() { try { f(*this, 1); cout << 1 << endl; }
    catch (X) { cout << 2 << endl; }
    catch (int) { cout << 3 << endl; } }
    X (X &) { cout << 4 << endl; }
    ~X () { cout << 5 << endl; } };
struct Y: X { Y () { f(*this, -1); cout << 6 << endl; }
    Y (Y &) { cout << 7 << endl; }
    ~Y () { cout << 8 << endl; } };
void f(X & x, int n) { try { if (n < 0) throw x;
    if (n > 0) throw -1;
    cout << 9 << endl; }
    catch (int) { cout << 10 << endl; }
    catch (X& a) { cout << 11 << endl;
        f(a, 0);
        cout << 12 << endl;
        throw; } }
int main() { try { Y a; }
    catch (...) { cout << 13 << endl;
        return 0; }
    cout << 14 << endl;
    return 0; }
```

Коллоквиум, 2009 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
C::x = A::f ();
C::f (3);
C::x = ::f (4, 5);
C::x = 6;
c.A::f ();
c.C::f (7);
::x =::f ('8', 9);
```

Отмечены исправляющие (обязательные) изменения.

Критерии: За каждое не отмеченное или неверное уточнение области видимости (из 7) -2. За каждое не найденное исправление (из 3) еще -3.

2. Ответ: Ошибочным является вызов `p -> h()` (нет функции `h()` без параметров).

Будет выдано в поток: `B::h, 2` `B::f, 2` `B::h, 0` `A::g` `B::h, 3`

Критерии: За каждую не найденную ошибку, лишнюю ошибку или неправильно вызванную функцию – 2.

3. Ответ:

```
void f (const char *, int); void f (int, const char *);
```

Критерии: За каждую ошибку –5. Если приводятся прототипы более, чем двух функций –10. Пропуск слова `const` в обеих функциях одновременно считать за одну ошибку.

4. Ответ:

`A (), B (), C (), B (B &), A (A &), ~A (), ~B (), ~C (), ~B (), ~A ()`

Критерии: За каждую ошибку –3.

5. Ответ:

```
template<class T> complex<T> complex<T>::operator *
(const complex<T>& x)
{ complex<T> w; w.r = r * x.r - i * x.i;
  w.i = r * x.i + i * x.r; return w; }
template<class T> complex<T> operator *
(const complex<T>& x, const complex<T>& y)
{ complex<T> w; w.r = x.r * y.r - x.i * y.i;
  w.i = x.r * y.i + x.i * y.r; return w; }
```

Критерии: За отсутствие каждого примера или ошибки в нем: -5.

6. Ответ: (1) Абстрактный тип данных есть тип данных с полностью скрытым (инкапсулированным) внутренним устройством. Работа с объектами абстрактных типов данных осуществляется с помощью специальных функций (доступных методов). Описывать конструкторы и создавать объекты абстрактных типов данных можно.

(2) Абстрактный класс – это класс, содержащий чистые виртуальные функции. Объекты абстрактных классов создавать нельзя, но можно строить объекты классов, производных от абстрактных. В абстрактных классах могут быть конструкторы и деструкторы.

Критерии: За отсутствие каждого из определений: -5.

7. Ответ: 4 11 10 12 2 5 4 11 10 12 5 13 5

Критерии: За каждую неверную (лишнюю или пропущенную) печать -2.

8. Ответ: (1) Заменить слово `class` словом `struct`, либо вставить после открывающей фигурной скобки определения класса указание открытой зоны `public`:. (2) Сделать метод `g()` статическим.

Критерии: За пропуск ошибок доступа к закрытым членам класса: -5. За пропуск ошибки вызова нестатического метода: -5. За указание ошибки там, где ее нет: -5.

9. Ответ:

1. Точное отождествление (с учетом вариантов синтаксической записи):

```
void f (double); void f (int);  
void g () { ... f (1); ... } // f (int)
```

2. Безопасные расширения типов (`char, short, enum, --> int (unsigned int); bool --> int; float --> double; double --> long double`):

```
void f (float); void f (long double);  
void g () { ...f (1.0); ... } // f(long double)
```

3. Стандартные преобразования:

```
void f (char *); class A { ... }; void f (A);  
void g () { ... f (0); ... } // f (char *)
```

4. Пользовательские преобразования:

```
struct S { ... S (int); operator int (); ... };  
void e (int); void e (char *); void f (S); void f (char *);  
void g (S &a) { ... e (a); ... // e (int)  
               f (1); ... } // f (S)
```

Критерии: За пропуск какого-либо одного из шагов или непонимание его сущности: -4. За пропуск других шагов или непонимание их сущности: -2 за каждый шаг. Отсутствие примеров при указании самого шага: -1 за каждый пропуск.

10. Решение:

```
typedef vector<int> V;  
typedef list<int> L;  
int g (const V & vect, L & lst, int step)  
{ V:: const_iterator vp = vect.begin ();  
  int t = 0;  
  L:: iterator lp = lst.begin ();  
  do { * lp = * vp, ++ t;  
        if (vect.end() - vp <= step) break;  
        ++lp; vp += step;  
      } while (lp != lst.end ());  
  L:: reverse_iterator rp = lst.rbegin ();  
  while (rp != lst.rend ()) { cout << * rp << ' '; ++ rp; }  
  cout << endl;  
  return t;  
}
```

Критерии: За каждую существенную ошибку: -3. Если проверка на выход за границу вектора проводится в другом месте (например, в постусловии цикла) – не снижать.

Коллоквиум, 2009 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
template<class T> bool complex<T>::operator == (const complex<T>& x)
{ return r == x.r && i == x.i; }
template<class T> bool operator ==
    (const complex<T>& x, const complex<T>& y)
{ return x.r == y.r && x.i == y.i; }
```

Критерии: За отсутствие каждого примера или ошибки в нем: **-5.**

2. Ответ: `void f (int, const char *);` либо `void f (X, const char *);`
`void f (const char *);`

Критерии: За каждую ошибку **-5.** Если приводятся прототипы более, чем двух функций – **10.** Пропуск слова `const` в обеих функциях одновременно считать за одну ошибку.

3. Ответ:

```
                x = f (3); // ::f (3)
//            x = pa -> f ();
                x = pa -> f (4); // A::f (4)
pa = pc; // x = pa -> f ();
                x = pa -> f (5); // A::f (5)
                x = pb -> f (); // B::f ()
//            x = pb -> f (6);
pb = pc; x = pb -> f (); // C::f ()
//            x = pb -> f (7);
                x = pc -> f (); // C::f ()
                x = pc -> f (8); // C::f (8)
```

Комментариями закрыты неправильные вызовы.

Критерии: За каждое неправильное или неуказанное уточнение области видимости (из 7) **-2**, за найденную ошибку (из 4) **-3.**

4. Решение:

```
typedef vector<long int> VL;
typedef list<long int> LL;
int g (VL & vect, const LL & lst, int step)
{ VL:: iterator vp = vect.begin (); int t = 0;
  LL:: const_iterator lp = lst.begin ();
  do { if (*lp < 0) *vp = *lp, ++t;
      if (vect.end() - vp <= step) break; // если этого нет, не снижать
      ++lp; vp += step;
    } while (vect.end() > vp && lp != lst.end ());
  VL:: const_iterator rp = vect.begin ();
  while (rp != vect.end ()) { cout << *rp << ' '; ++rp; }
  cout << endl;
  return t;
}
```

Критерии: За каждую существенную ошибку: **-3.** Если проверка на выход за границу вектора проводится в другом месте (например, в постусловии цикла) – не снижать.

5. Ответ: (1) Отбор функций по количеству формальных параметров (с учетом параметров, имеющих значения по умолчанию, и многоточий), (2) Создание множеств пригодных функций при анализе каждого из параметров в отдельности, (3) Параметры функций, соответствующие многоточию в списке формальных параметров, считаются подходящими во всех случаях, (4) Вычисление пересечения построенных множеств, (5) Если в полученное пересечение входит ровно одна функция – она и есть искомый результат, в противном случае выдается сообщение об ошибке.

Пример:

```
class X { public: X (long); X (char *); ...};
void f (X, long); void f (X, double);
void g () { ... f(0, 0); // нет пригодной функции
           f(1L, 1L); // f(X, long)
           f("a", 1); ... } // неоднозначность при вызове
```

Критерии: Пропуск любого шага или непонимание его сущности: -2. Нет примера: -2.

6. Ответ: A(), A(), B(), C(), D(), ~D(), ~C(), ~B(), ~A(), ~A()

Критерии: За каждую ошибку -3.

7. Ответ: Ошибок нет. Будет выдано в поток: D::h D::h,2 D::f,2 D::h D::h,1
D::g D::h D::h D::h,3

Критерии: За каждую лишнюю ошибку или неправильно вызванную функцию -2.

8. Ответ: (1) Полиморфизм – использование одинакового интерфейса (имени) для различных действий (объектов). (2) Статический полиморфизм действует на этапе компиляции. Примеры – перегрузка функций и операций:

```
void f(double); void f(int);
void f(float); void f(long); void g() { ... f('a'); ... }
```

3. Динамический полиморфизм реализуется с помощью виртуальных функций и доступа к функциям по указателям или ссылкам, когда на этапе компиляции не удаётся однозначно определить вызываемую функцию, так как значение указателя (ссылки) в это время не известно.

```
class X {public: virtual void f(int x) {cout<<"X"<<endl; } };
class Y: public X{ public: void f(int y) {cout<<"Y"<<endl; } };
int main () { X * p = new Y b; ... p -> f (1); }
```

4. Параметрический (типовый) полиморфизм – использование шаблонов функций и классов.

```
template <class T> T max (T x, T y) { ... }
void f () { ... max (1, 2); ... }
template <class T> class Vector { T * p; int size;
                                public: explicit Vector (int); };
Vector<int> X (20);
```

Критерии: За отсутствие каждого из определений: -3. За отсутствие соответствующего примера: -1 за каждый пропуск.

9. Ответ: 9 1 4 11 10 12 5 13 5

Критерии: За каждую неверную (лишнюю или пропущенную) печать -2.

10. Ответ: (1) Заменить слово `class` словом `struct`, либо вставить после открывающей фигурной скобки определения класса указание открытой зоны `public:..` (2) В определениях методов `f()` и `g()` вставить после пустого списка формальных параметров слово `const`.

Критерии: За пропуск ошибок доступа к закрытым членам класса: -5. За пропуск одной ошибки вызова неконстантного метода `"f()"`: -6. За пропуск одной ошибки вызова неконстантного метода `"g()"`: -5. За пропуск обеих таких ошибок в методах `"f()"` и `"g()"` -8. За указание ошибки там, где ее нет: -5.

Коллоквиум, 2009 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: Ошибочным является вызов `p -> h(2)` (Функция `h(int)` из производного класса не замещает `h()` базового класса, так как имеет отличающийся набор параметров). Будет выдано в поток: `U::h,1 U::f,1 T::h T::g T::h`

Критерии: За каждую не найденную ошибку, лишнюю ошибку или неправильно вызванную функцию – 2.

2. Ответ:

```
x = A::f(); // A::f()
x = B::f(5); // B::f(5)
x = B::f(6, 6); // B::f(6, 6)
x = B::f(5); // B::f(5)
x = a.A::f(); // A::f()
x = a.B::f(7); // B::f(7)
return a.B::g(&a, &a);
```

Отмечены исправляющие (обязательные) изменения.

Критерии: За каждое не отмеченное уточнение области видимости (из 7) -2. За каждое не найденное исправление (из 3-х) еще -3.

3. Ответ: (а) Бинарная операция (сложение и присваивание):

```
Друг: complex & operator+=(complex & a, const complex & b)
{ a.re += b.re; a.im += b.im; return a; }
Метод: complex & complex::operator+=(const complex & a)
{ re += a.re, im += a.im; return *this; }
```

(б) Унарная префиксная операция:

```
Друг: complex & operator++(complex & a) { ++ a.re; return a; }
Метод: complex & complex::operator++() { ++ re; return *this; }
```

(в) Унарная постфиксная операция:

```
Друг: complex operator++(complex & a, int b)
{ complex temp = a; ++ a.re; ++ a.im; return temp; }
Метод: complex complex::operator++(int b)
{ complex temp = *this; ++ re; ++ im; return temp; }
```

(г) Операции присваивания могут перегружаться только нестатическими членами классов, но не функциями-друзьями. **(д)** При перегрузке операций не могут меняться их основные свойства – количество необходимых операндов, ассоциативность и приоритет.

Критерии: За отсутствие каждого из примеров: -1. За отсутствие описания особенностей операции присваивания: -2. Неправильный ответ на вопрос о свойствах операций: -2. На последний вопрос лаконичный ответ «нет» засчитывается как верный.

4. Ответ:

```
template<class T>complex<T>complex<T>::operator++() { ++ r; return *this; }
template<class T>complex<T>operator++(complex<T>&x) { ++ x.r; return x; }
```

Критерии: За отсутствие каждого примера или ошибки в нем: -5.

5. Ответ: Динамическое и статическое приведение типов возможно для объектов родственных полиморфных классов, относящихся к одной иерархии классов. Статическое приведение возможно также для свободных указателей (`void *`), которые могут преобразовываться в значения любых типов указателей, и для преобразований между арифметическими типами (отсутствие этого добавления не наказывается). Указатель может иметь нулевое значение, поэтому при динамическом приведении указателя в случае возникновения ошибки может возвращаться это нулевое значение, которое затем может быть проверено в программе. Ошибка при приведении ссылки всегда приводит к возбуждению исключительной ситуации `bad_cast`, никакого выделенного значения для ссылок не существует. Проверка правильности динамического приведения ссылок всегда выполняется перехватом исключительной ситуации.

```

Пример: class A { ... };
class B: public A { ... };
class C: public A { ... };
void f (A * p, A & r)
{ if (B* pp = dynamic_cast<B*>(p)) { ... } // использование указателя pp
  else { ... } // указатель pp не принадлежит нужному типу
  B & pr = dynamic_cast<B &> (r); ... // использование ссылки pr
}
void g ()
{ try { f (new B, * new B); // правильный вызов
        f (new C, * new C); // будет перехват (C - из другой иерархии,
    } // основанной на том же базовом классе)
    catch (bad_cast) { ... } } // обработка исключительной ситуации

```

Критерии: За неверное понимание сути операций приведения типов: **-4**. За неверное понимание приведения типа в одном из случаев: **-4**. За неверное понимание сути динамического приведения типа и для указателей и для ссылок: **-6**. Отсутствие одного из примеров (здесь приведен обобщенный пример, эквивалентный сразу двум простым) при демонстрации правильного понимания сути: **-2**.

6. Решение:

```

typedef vector<double> V;
typedef list<double> L;
int g (const V & vect, L & lst, int step)
{ V:: const_iterator vp = vect.begin ();
  L:: iterator lp = lst.begin (); int t = 0;
  do { if (* lp != * vp) lst.insert (lp, * vp), ++ t;
        if (vect.end()-vp<=step) break; // если этого нет, не снижать
        ++lp; vp += step;
    } while (vect.end () > vp && lp != lst.end ());
  L:: reverse_iterator rp = lst.rbegin ();
  while (rp != lst.rend ()) { cout << * rp << ' '; ++ rp; }
  cout << endl; return t; }

```

Критерии: За каждую существенную ошибку: **-3**. Если проверка на выход за границу вектора проводится в другом месте (например, в постусловии цикла) – не снижать

7. Ответ: A(), A(), B(), A(A &), C(), ~C(), ~A(), ~B(), ~A(), ~A()

Критерии: За каждую ошибку **-3**.

8. Ответ:

```

void f (double d = 0, const char * s = 0);
// long int и float засчитывать за правильное
void f (const char *); либо void f (const char *, int i = 0);

```

Критерии: За каждую ошибку **-5**. Если приводятся более, чем две функции **-10**. Пропуск слова *const* в обеих функциях одновременно считать за одну ошибку.

9. Ответ: (1) Заменить слово *class* словом *struct*, либо вставить после открывающей фигурной скобки определения класса указание открытой зоны *public*:. (2) В определении метода *g()* вставить перед типом формального параметра слово *const*. (3) В определении метода *g()* вставить после списка формальных параметров слово *const*.

Критерии: За пропуск ошибок доступа к закрытым членам класса: **-5**. За пропуск одной ошибки вызова метода "*g()*": **-5**. За пропуск обеих таких ошибок: **-8**. За указание ошибки там, где ее нет: **-5**.

10. Ответ: 4 11 9 12 2 5 10 6 8 5 14

Критерии: За каждую неверную (лишнюю или пропущенную) печать **-2**.

Коллоквиум, 2009 год. Ответы и критерии оценок.

Вариант 4.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (1) Заменить слово `class` словом `struct`, либо вставить после открывающей фигурной скобки определения класса указание открытой зоны `public`:. (2) В определении метода `f()` вставить перед типом формального параметра слово `const`. (3) В определении метода `f()` вставить после списка формальных параметров слово `const`.

Критерии: За пропуск ошибок доступа к закрытым членам класса: -5. За пропуск одной ошибки вызова метода "`f()`": -5. За пропуск обеих таких ошибок: -8. За указание ошибки там, где ее нет: -5.

2. Ответ: `void f (double d = 0, const char * s = 0);`
`void f (const char *, int i = 0);`
либо: `void f (double, const char * s = 0);`
`void f (const char * s = 0, int i = 0);`

Критерии: За каждую ошибку -5. Если приводятся прототипы более, чем двух функций -10. Пропуск слова `const` в обеих функциях одновременно считать за одну ошибку.

3. Ответ:

```
x = f (3);           // B::f (3)
x = f (4);           // B::f (4)
x = f (5);           // B::f (5)
x = pa -> f (7);      // A::f (7)
pa = pc; x = pa -> f (8); // A::f (8)
x = pb -> f (7);      // B::f (7)
x =          f (8);   // ::f (8)
pb = pc; x = pb -> f (7); // B::f (7)
x = pc -> f (8);      // C::f (8)
c. B::g ();
```

Конструктор класса *B* должен быть сделан открытым методом класса *B* (*public*). Наследование класса *B* классом *C* должно быть сделано открытым (*public*:).

Критерии: За каждое не найденное исправление (из 2) -4. За каждое не отмеченное уточнение области видимости (из 10) -2.

- 4. Ответ:** 1. Создание временного объекта (копии исключительной ситуации)
2. Выход из **try**-блока с освобождением памяти для локальных объектов (свёртка стека)
3. Выполнение составного оператора перехватчика (**catch**-блока) – статическая ловушка
4. Если статическая ловушка не сработала, выход в объемлющий блок с поиском подходящего обработчика (динамическая ловушка)
5. Если ни одна ловушка не сработала, выполнение функции `terminate()`.

Пример: `class A { char * p; int s; ... };
char & A::operator [] (int i)
{ if (i < 0) throw "A: размер < 0";
 if (i >= s) throw i;
 return p [i]; }
void g (int i)
{ ... try { ... A a (i), b (7); ... b [i] = a [i - 1]; ... }
 catch (int n) { ... }
 catch (const char * diagn) { ... }
}`

Критерии: За пропуск любого шага или непонимание его сущности: -2. Отсутствие примера: -2.

5. Ответ:

```
template<class T> complex<T> complex<T>::operator-- (int p)
{ complex<T> w; w = * this; -- r; return w; }
template<class T> complex<T> operator-- (complex<T>& x, int p)
{ complex<T> w; w = x; -- x.r; return w; }
```

Критерии: За отсутствие каждого примера или ошибки в нем: -5.

6. Ответ: 1. Конструктор умолчания (без параметров):

```
class complex { ... complex () { re = im = 0; } };
```

2. Конструктор преобразования (с одним параметром):

```
class complex { ... complex (const double a) { re = a; im = 0; } };
```

3. Конструктор копирования (с параметром-ссылкой на этот класс):

```
class complex { ... complex (complex & a) { re = re.a; im = re.im; } };
```

4. Обычный конструктор:

```
class complex { ... complex (double r, double i) { re = r; im = i; } };
```

Критерии: За отсутствие какого-либо из видов: -2. За отсутствие примера, если сам конструктор упомянут: -1.

7. Ответ: `A(), B(), C(), B(B &), D(), ~D(), ~B(), ~C(), ~B(), ~A()`

Критерии: За каждую ошибку -3.

8. Ответ: Ошибочными являются вызовы `p -> g(1)` и `p -> h(2)` (нет функций с таким набором параметров). Будет выдано в поток:

```
S::h    S::g    S::f,0    S::h    R::g    S::h
```

Критерии: За каждую не найденную ошибку, лишнюю ошибку или неправильно вызванную функцию -2.

9. Решение:

```
typedef vector<float> V;
typedef list<float> L;
int g (const V & vect, L & lst, int step)
{ V:: const_iterator vp = vect.begin ();
  L::reverse_iterator lp = lst.rbegin (); int t = 0;
  do { if (* lp * * vp < 0) * lp = - * lp;
        if (vect.end() - vp <= step) break;
        ++ lp;    vp += step;
    } while (lp != lst.rend ());
  L:: iterator rp = lst.begin ();
  while (rp != lst.end ())
  { cout << * rp << ' ';
    if (* rp >= 0) t ++;
    ++ rp;
  }
  cout << endl;
  return t;
}
```

Критерии: За каждую существенную ошибку: -3. Если проверка на выход за границу вектора проводится в другом месте (например, в постусловии цикла) – не снижать.

10. Ответ: 10 1 4 11 9 12 5 13 5

Критерии: За каждую неверную (лишнюю или пропущенную) печать -2.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow Aa \mid cSad \mid \varepsilon$
 $cAad \rightarrow cBad$
 $B \rightarrow c$
 $cB \rightarrow cBc$

(а) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$.

(б) Каким из перечисленных классов принадлежит язык $L(G)$? Ответ:

Класс $\mathfrak{L}$	$L(G) \in \mathfrak{L}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
языки типа 0	
регулярные языки	

(в) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

2. (а) Дать определение *недостижимого* символа.

(б) Эквивалентны ли данные КС-грамматики G_1 и G_2 ? Ответ обосновать.

$G_1 : S \rightarrow Baa$

$A \rightarrow Aa \mid a \mid \varepsilon$

$B \rightarrow b$

$G_2 : S \rightarrow E \mid BAa$

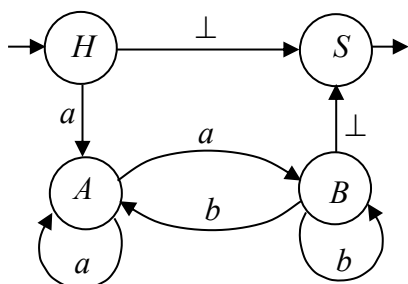
$D \rightarrow dDd \mid c$

$E \rightarrow EaD \mid Ed$

$A \rightarrow Aa \mid ADE \mid a \mid \varepsilon$

$B \rightarrow DbE \mid b$

3. По леволинейной грамматике G была построена диаграмма состояний $\mathcal{A}$:



(а) Восстановить по диаграмме $\mathcal{A}$ грамматику G :

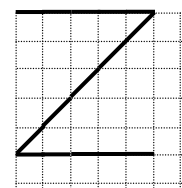
(б) Детерминирован ли разбор по данной диаграмме? Объясните свой ответ.

(в) Если разбор по диаграмме состояний $\mathcal{A}$ недетерминированный (то есть сам конечный автомат $\mathcal{A}$ — недетерминированный), то с помощью соответствующего алгоритма построить эквивалентный детерминированный автомат $\mathcal{A}_D$, иначе — написать для $\mathcal{A}$ программу-анализатор на языке C++.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вниз. В левом верхнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{a, b\}$: a — переместить перо на одну клетку вправо; b — переместить перо по диагонали на одну клетку влево-вниз.

Например, по последовательности $aaaaabbbbbbaaaaa$ интерпретатор построит букву «Z» размера 5×5 , изображенную на рисунке.

Построить грамматику с терминальным алфавитом $\{a, b\}$, описывающую буквы «Z» любого размера $n \times n$, $n \geq 1$. В грамматике должно быть не более 7 правил вывода (считая альтернативы).



5. (а) Сформулируйте достаточное условие применимости метода рекурсивного спуска с учетом возможной ε -альтернативы («канонический вид» КС-грамматики).

(б) Выполняется ли это условие для грамматики G ? Обоснуйте ответ.

$G: S \rightarrow aAS \mid bB \mid \varepsilon$

$A \rightarrow aAca \mid bBb$

$B \rightarrow cB \mid \varepsilon$

6. Описать грамматику, порождающую язык $L_{ex} = \{ d^m a^n \mid m, n \geq 0 \}$, и вставить в нее действия вида $\langle cout << 'СИМВОЛ'; \rangle$ так, чтобы для любых целых $m, n \geq 0$ в процессе анализа рекурсивным спуском входной цепочки $d^m a^n$ печаталась бы цепочка $cb^n a^m$.

7. Дайте определение системы программирования. Перечислите основные компоненты классической системы программирования на базе ОС UNIX (язык программирования Си). Кратко опишите назначение каждого из основных компонент.

8. Каковы два основных способа подключения библиотек функций? Чем они отличаются? Назовите основные достоинства этих способов.

9. Записать результат перевода в постфиксную запись (ПОЛИЗ) фрагмента программы на языке Си:

```
b = i = 0; do {S = (a < i*(a + b)? b++ : ++i);} while (i < 100);
```

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Соглашения: (1) Адреса в постфиксной записи можно обозначать подчеркиванием ($\underline{a} \equiv \&a$), (2) Префиксный $++$ обозначен в постфиксной записи как $\#\#$. (3) Различение префиксных и постфиксных $++$ разрешается на уровне интерпретации. (4) "Мусор" в стеке после выполнения присваивания (т.е. значение присваивания) можно игнорировать или "убрать" с помощью операции $;$; $"$ в ПОЛИЗе).

10. Чем характерны машинно-зависимые оптимизирующие преобразования? Перечислите их основные типы, дав краткое пояснение. На каком этапе трансляции производится большая часть этих преобразований?

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow A \mid \varepsilon$
 $A \rightarrow cdAa \mid cB$
 $cdcBa \rightarrow cdcDa$
 $Da \rightarrow Daa$
 $D \rightarrow a$

(а) Классификация:
найти такое целое k ,
что G является
грамматикой типа k
и не является грам-
матикой типа $k+1$

(б) Каким из перечисленных классов принадлежит язык $L(G)$? Ответ:

Класс $\mathfrak{Z}$	$L(G) \in \mathfrak{Z}$? (да/нет)
регулярные языки	
контекстно-зависимые языки	
контекстно-свободные языки	
языки типа 0	

(в) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

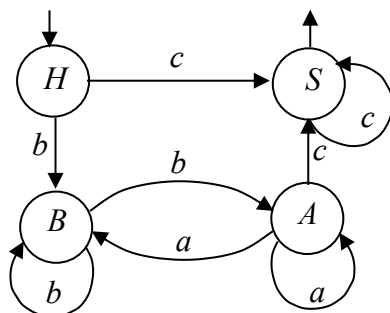
2. (а) Дать определение *бесплодного* символа.

$G_1 : S \rightarrow BAaC \mid BC$
 $A \rightarrow Aa \mid a \mid \varepsilon$
 $B \rightarrow b$

(б) Эквивалентны ли данные КС-грамматики G_1 и G_2 ? Ответ обосновать.

$G_2 : S \rightarrow E \mid BAa \mid AA$
 $D \rightarrow bDc \mid a$
 $E \rightarrow EaD \mid Ed$
 $A \rightarrow AaE \mid ADE \mid \varepsilon$
 $B \rightarrow DbE \mid bE$

3. По левoliniейной грамматике G была построена диаграмма состояний $\mathcal{A}$:



(а) Восстановить по диаграмме $\mathcal{A}$ грамматику G :

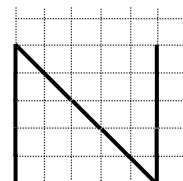
(б) Детерминирован ли разбор по данной диаграмме? Объясните свой ответ.

(в) Если разбор по диаграмме состояний $\mathcal{A}$ недетерминированный (т.е. конечный автомат $\mathcal{A}$ — недетерминированный), то с помощью соответствующего алгоритма построить эквивалентный детерминированный автомат $\mathcal{A}_D$, иначе — написать для $\mathcal{A}$ программу-анализатор на языке C++.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{c, f\}$: c — переместить перо на одну клетку вверх; f — переместить перо по диагонали на одну клетку вправо-вниз.

Например, по последовательности $ccccffffccccc$ интерпретатор построит букву «N» размера 5×5 , изображенную на рисунке.

Построить грамматику с терминальным алфавитом $\{c, f\}$, описывающую буквы «N» любого размера $n \times n$, $n \geq 1$. В грамматике должно быть не более 7 правил вывода (считая альтернативы).



5. (а) Сформулируйте достаточное условие применимости метода рекурсивного спуска с учетом возможной ϵ -альтернативы («канонический вид» КС-грамматики).

(б) Выполняется ли это условие для грамматики G ? Обоснуйте ответ.

$G: S \rightarrow dDS \mid cBA \mid \epsilon$
 $D \rightarrow dD \mid aAc \mid b$
 $A \rightarrow aAca \mid bB$
 $B \rightarrow cB \mid \epsilon$

6. Описать грамматику, порождающую язык $L_{ax} = \{ c^k b^m \mid m \geq k \geq 0 \}$, и вставить в нее действия вида $\langle cout << 'СИМВОЛ'; \rangle$ так, чтобы для любых целых $m \geq k \geq 0$ в процессе анализа рекурсивным спуском входной цепочки $c^k b^m$ печаталась бы цепочка $ad^k b^{m-k}$.

7. Перечислите этапы технологического цикла создания программного продукта. Кратко охарактеризуйте каждый этап, указав назначение и результат его прохождения.

8. Перечислите основные типы библиотек, используемых в современных системах программирования (не путать со способами подключения!). Дайте их краткую характеристику.

9. Записать результат перевода в постфиксную запись (ПОЛИЗ) фрагмента программы на языке Си:

```
for( i = 0; i < n; i++ ){ a = b - c - d; ++b; }
```

Ответ:

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ																	

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Соглашения: (1) Адреса в постфиксной записи можно обозначать подчеркиванием ($\underline{a} \equiv \&a$), (2) Префиксный $++$ обозначен в постфиксной записи как $\#\#$. (3) Различение префиксных и постфиксных $++$ разрешается на уровне интерпретации. (4) "Мусор" в стеке после выполнения присваивания (т.е. значение присваивания) можно игнорировать или "убрать" с помощью операции $;$; " в ПОЛИЗе).

10. Перечислите основные типы машинно-независимых оптимизирующих преобразований, конкретизируя, что под ними понимается. На каком этапе трансляции производится большая часть этих преобразований?

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика
- $$G: \begin{aligned} S &\rightarrow Bc \mid Sa \mid \varepsilon \\ B &\rightarrow Ad \\ A &\rightarrow d \\ A &\rightarrow bB \end{aligned}$$

(а) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$.

- (б) Каким из перечисленных классов принадлежит язык $L(G)$? Ответ:

Класс $\mathfrak{L}$	$L(G) \in \mathfrak{L}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
регулярные языки	
языки типа 0	

(в) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

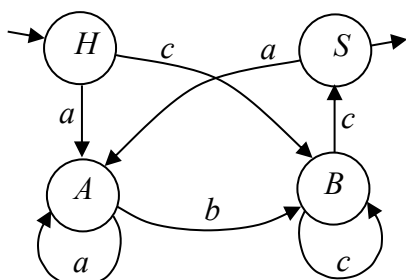
2. (а) Дать определение приведенной КС-грамматики.

$$G_1: \begin{aligned} S &\rightarrow BA \\ A &\rightarrow bAa \mid a \mid \varepsilon \\ B &\rightarrow b \mid \varepsilon \end{aligned}$$

- (б) Эквивалентны ли данные КС-грамматики G_1 и G_2 ? Ответ обосновать.

$$G_2: \begin{aligned} S &\rightarrow E \mid BA \\ D &\rightarrow dDd \mid c \\ E &\rightarrow ED \mid Ed \\ A &\rightarrow bAa \mid ADE \mid a \mid \varepsilon \\ B &\rightarrow DbE \mid b \end{aligned}$$

3. По левосторонней грамматике G была построена диаграмма состояний $\mathcal{A}$:

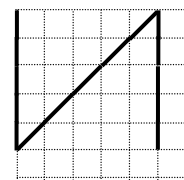


- (а) Восстановить по диаграмме $\mathcal{A}$ грамматику G :

- (б) Детерминирован ли разбор по данной диаграмме? Объясните ответ.

(в) Если разбор по диаграмме состояний $\mathcal{A}$ недетерминированный (т.е. конечный автомат $\mathcal{A}$ — недетерминированный), то с помощью соответствующего алгоритма построить эквивалентный детерминированный автомат $\mathcal{A}_D$, иначе — написать для $\mathcal{A}$ программу-анализатор на языке C++.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вниз. В левом верхнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{e, d\}$: e — переместить перо на одну клетку вниз; d — переместить перо по диагонали на одну клетку вправо-вверх. Например, по последовательности $eeeeedddddeeeee$ интерпретатор построит букву «и» размера 5×5 , изображенную на рисунке. Построить грамматику с терминальным алфавитом $\{e, d\}$, описывающую буквы «и» любого размера $n \times n$, $n \geq 1$. В грамматике должно быть не более 7 правил вывода (считая альтернативы).



5. (а) Сформулируйте достаточное условие применимости метода рекурсивного спуска с учетом возможной ϵ -альтернативы («канонический вид» КС-грамматики).

(б) Выполняется ли это условие для грамматики G ? Обоснуйте ответ.

$G: S \rightarrow aAeS \mid bB \mid \epsilon$
 $A \rightarrow aAd \mid bC \mid aC$
 $C \rightarrow cC \mid \epsilon$
 $B \rightarrow eS \mid d$

6. Описать грамматику, порождающую язык $L_{ex} = \{ c^n d^m \mid m \geq k \geq 0 \}$, и вставить в нее действия вида $\langle cout \ll 'СИМВОЛ'; \rangle$ так, чтобы для любых целых $m \geq n \geq 0$ в процессе анализа курсивным спуском входной цепочки $c^n d^m$ печаталась бы цепочка $b^n a d^{m-n}$.

7. Что такое интегрированная среда разработки? Каковы преимущества (приведите три причины) использования ИСР по сравнению с классической СП?

8. Перечислите основные критерии проектирования стандартных библиотек. Кратко прокомментируйте каждый из критериев.

9. Записать результат перевода в постфиксную запись (ПОЛИЗ) фрагмента программы на языке Си:

```
while (a < b) {S += a < 0 ? -a : b++; b = b - a - 1;}
```

Ответ:

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ																	

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Соглашения: (1) Адреса в постфиксной записи можно обозначать подчеркиванием ($\underline{a} \equiv \&a$), (2) Префиксный ++ обозначен в постфиксной записи как ##. (3) Различение префиксных и постфиксных ++ разрешается на уровне интерпретации. (4) "Мусор" в стеке после выполнения присваивания (т.е. значение присваивания) можно игнорировать или "убрать" с помощью операции "; " в ПОЛИЗе).

10. Какие участки программы называют линейными? Что такое профилировщик? Кратко опишите его назначение.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика

$G: S \rightarrow Bcd$
 $S \rightarrow \varepsilon$
 $B \rightarrow Ca$
 $C \rightarrow a$
 $C \rightarrow Bbc$

(а) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$.

(б) Каким из перечисленных классов принадлежит язык $L(G)$?

Ответ:

Класс $\mathfrak{Z}$	$L(G) \in \mathfrak{Z}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
языки типа 0	
регулярные языки	

(в) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

2. (а) Можно ли любой КС-язык описать неукорачивающей КС-грамматикой? Поясните ответ.

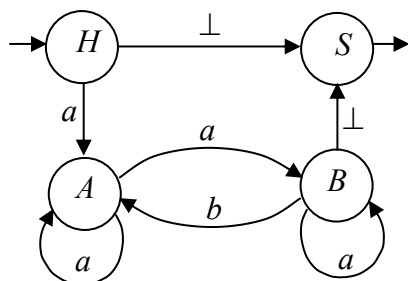
(б) Эквивалентны ли данные КС-грамматики G_1 и G_2 ?

Ответ обосновать.

$G_1: S \rightarrow AS \mid a$
 $A \rightarrow b$

$G_2: S \rightarrow ADS \mid a$
 $A \rightarrow CC \mid bC$
 $D \rightarrow DC \mid \varepsilon$
 $C \rightarrow CDC \mid D$
 $E \rightarrow Eb$

3. По левوليнейной грамматике G была построена диаграмма состояний $\mathcal{A}$:

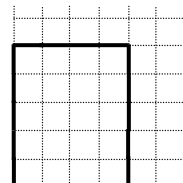


(а) Восстановить по диаграмме $\mathcal{A}$ грамматику G :

(б) Детерминирован ли разбор по данной диаграмме? Объясните ответ.

(в) Если разбор по диаграмме состояний $\mathcal{A}$ недетерминированный (т.е. конечный автомат $\mathcal{A}$ – недетерминированный), то с помощью соответствующего алгоритма построить эквивалентный детерминированный автомат $\mathcal{A}_D$, иначе – написать для $\mathcal{A}$ программу-анализатор на языке C++.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{a, c, e\}$: c – переместить перо на одну клетку вверх; a – переместить перо на одну клетку вправо; e – переместить перо на одну клетку вниз. Например, по последовательности $ssssaaaaae$ интерпретатор построит букву «П» размера 5×4 , изображенную на рисунке. Построить грамматику с терминальным алфавитом $\{a, c, e\}$, описывающую буквы «П» любого размера $n \times m$, $n, m \geq 1$. В грамматике должно быть не более 5 правил вывода (считая альтернативы).



5. (а) По процедурам анализатора, действующего методом рекурсивного спуска, восстановить КС-грамматику.

```
char c; void gc(){cin>>c;}; void S(); void A(); void B();
int main(){ try{ gc(); S(); if(c!='\n') throw; cout<<"OK"<<endl; return 0;}
catch (...) { cout<<"unexpected: "<<c<<endl; return 1;} };
void S(){if(c=='b'){ gc(); S(); A();}else if(c=='a'){gc();} else throw;}
void B(){if(c=='e'){ gc(); B();}
void A(){if(c=='a'){ gc(); B();}
else if(c=='c'){ gc(); A(); if(c=='b')gc();else throw;}else throw;}}
```

(б) Удовлетворяет ли эта грамматика достаточному условию применимости метода рекурсивного спуска с учетом возможной ϵ -альтернативы («канонический вид» КС-грамматики)? Ответ обосновать.

6. Дана грамматика: $G_1: S \rightarrow cAb$

$A \rightarrow dS \mid cd$

Вставить в G_1 действия вида $\langle \text{cout} << \text{'символ'}; \rangle$ по переводу цепочек языка $L_1 = L(G_1)$ в цепочки языка $L_2 = \{b(a)^{k+1} \mid k \geq 1\}$, так, чтобы символов a в цепочке из L_2 было на один больше, чем символов d в соответствующей цепочке из L_1 .

7. Приведите схему работы компилятора, указав входные и выходные данные для каждого этапа.

8. Перечислите основные типы библиотек, что они из себя представляют? Укажите способы их подключения.

9. Записать результат перевода в ПОЛИЗ (постфиксную запись) фрагмента программы на языке Си:

```
if ( a < b ) { S = S - a - c; a++; b = -b; } else b = c = b - a;
```

Ответ:

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Соглашения: (1) Адреса в постфиксной записи можно обозначать подчеркиванием ($\underline{a} \equiv \&a$), (2) Префиксный ++ обозначен в постфиксной записи как ##. (3) Различение префиксных и постфиксных ++ разрешается на уровне интерпретации. (4) "Мусор" в стеке после выполнения присваивания (т.е. значение присваивания) можно игнорировать или "убрать" с помощью операции "; " в ПОЛИЗе).

10. Перечислите основные методы выделения динамической памяти, дав их краткую характеристику и указав основные проблемы, возникающие при их использовании.

Экзамен, 2009 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

- 1. Ответ:** (а) $k = 0$ (есть $S \rightarrow \varepsilon$ и S встречается в правой части правила $S \rightarrow cSad$)
 (б) КС – " ∂a ", КЗ – " ∂a ", тип 0 – " ∂a ", регулярный – "**нет**"
 (в) $L(G) = \{c^{n+k}(ad)^n \mid n \geq 1, k \geq 0\} \cup \{\varepsilon\}$.

Критерии: (а) нет ответа; неверный ответ : **-3** (обоснование не обязательно), (б) за каждый неверный или отсутствующий ответ: **-1**, всего до **-4** за этот пункт (обоснование не обязательно), (в) описание отсутствует или с ошибками: **-3**

- 2. Ответ:** (а) символ $x \in T \cup N$ называется *недостижимым* в грамматике $G = \langle T, N, P, S \rangle$, если он не появляется ни в одной сентенциальной форме этой грамматики, (б) **Да**, эквивалентны. Грамматика G_1 является результатом применения к грамматике G_2 алгоритма приведения КС-грамматики. Другое обоснование: $L = \{ba^n \mid n \geq 1\} = L(G_1) = L(G_2)$.

Критерии: (а) нет ответа; определение с ошибками: **-5**, (б) нет ответа; неверный ответ; нет обоснования; неверное обоснование: **-5**.

- 3. Ответ:** (а) $G: S \rightarrow \perp \mid B\perp$
 $B \rightarrow Bb \mid Aa$
 $A \rightarrow Aa \mid Bb \mid a$

(б) Разбор недетерминирован, т.к. автомат $\mathcal{A}$ не является детерминированным: из состояния A возможен переход в более чем одно состояние (по символу a)

- 4. Ответ:** Описание буквы «Z» размера $n \times n$ — это цепочка $a^n b^n a^n$. Следовательно, требуется построить грамматику, порождающую язык $\{a^n b^n a^n \mid n \geq 1\}$:

$S \rightarrow ABSa \mid Aba$	Замечание. Грамматика $S \rightarrow ABSa \mid ABa$
$BA \rightarrow AB$	$BA \rightarrow AB$
$Bb \rightarrow bb$	$B \rightarrow b$
$Ab \rightarrow ab$	$A \rightarrow a$
$Aa \rightarrow aa$	ошибочная, так как порождает лишние цепочки, напр.: $ababaa$.

- 5. Ответ:** (а) Достаточное условие: каждая группа правил с одинаковой левой частью имеет один из перечисленных ниже видов и выполняются дополнительные условия: (1) либо $X \rightarrow \alpha$, где $\alpha \in (T \cup N)^*$ и это единственное правило вывода для этого нетерминала; (2) либо $X \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n$, где $a_i \in T$ для всех $i = 1, 2, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$, то есть, если для нетерминала X правил вывода несколько, то они должны начинаться с терминалов, причем все эти терминалы должны быть попарно различными; (3) либо $X \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n \mid \varepsilon$, где $a_i \in T$ для всех $i = 1, 2, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$, и $first(X) \cap follow(X) = \emptyset$. **Замечание.** Допускаются обозначения V_T, V_N для обозначения алфавитов терминальных и нетерминальных символов, и обозначение V для их объединения. $V = T \cup N$ ($V = V_T \cup V_N$)

(б) **Да**, т.к. все непустые альтернативы начинаются с попарно различных терминалов, $first(S) \cap follow(S) = \emptyset$, и $first(B) \cap follow(B) = \emptyset$.

В задаче: $first(S) = \{a, b\}$, $follow(S) = \emptyset$; $first(B) = \{c\}$, $follow(B) = \{b\}$.

Критерии: (а) нет ответа, ошибки в формулировке: **-5**, (б) нет ответа; неверный ответ; нет обоснования; ошибочное обоснование: **-5**.

6. Ответ. Соглашение: запись $\langle a \rangle$ является сокращением $\langle cout << 'a'; \rangle$

$L_{ax} = \{ d^n a^{m+n} \mid m, n \geq 0 \} = \{ d^n a^n a^m \mid m, n \geq 0 \}$ (входной язык). Грамматика для L_{ax} , к которой применим РС-метод, и грамматика с действиями выглядят так:

$$\begin{array}{lll} S \rightarrow AB & S \rightarrow AB & \text{или :} & S \rightarrow \langle c \rangle AB \\ A \rightarrow dAa \mid \varepsilon & A \rightarrow dAa \langle b \rangle \mid \varepsilon \langle c \rangle & & A \rightarrow d \langle b \rangle Aa \mid \varepsilon \\ B \rightarrow aB \mid \varepsilon & B \rightarrow a \langle a \rangle B \mid \varepsilon & & B \rightarrow a \langle a \rangle B \mid \varepsilon \\ L_u = \{ cb^n a^m \mid m, n \geq 0 \} & & & \text{(целевой язык).} \end{array}$$

Критерии: Грамматика не порождает L_{ax} , или не генерирует L_u : **0**; Порождает L_{ax} и формально генерирует соответствующие цепочки из L_u , но РС-метод неприменим: **-5**.

7. Ответ: Система программирования — это комплекс программных инструментов и библиотек, который поддерживает **весь** технологический цикл создания программного продукта (ПП). [Вариант: **Система программирования** — это комплекс программных средств (инструментов, библиотек), предназначенных для поддержки программного продукта на протяжении **всего** жизненного цикла этого программного продукта.]

Основные компоненты системы программирования: 1. **Транслятор** (проверяет синтаксическую правильность программ и переводит их с языка программирования на машинный язык, что и позволяет выполнить их на ЭВМ). 2. **Макрогенератор** или макропроцессор (работает непосредственно перед транслятором, используется для получения макrorасширения исходной программы). 3. **Редактор связей** или компоновщик (предназначен для связывания между собой (по внешним данным) объектных файлов, порождаемых компилятором, а также файлов библиотек, входящих в состав СП). 4. **Редактор текстов** (используется для составления и редактирования программ на языке программирования). 5. **Отладчик** (используется для проверочных запусков программ, локализации и исправления ошибок). 6. **Библиотеки** (облегчают работу программиста, предоставляя готовые фрагменты решения тех или иных подзадач программы, используются на этапе трансляции и исполнения). **Замечание.** Макрогенератор может рассматриваться как часть транслятора или пакетного редактора текстов, поэтому допустимо его отсутствие в перечне компонентов. Вместо трансляторов могут упоминаться компиляторы и ассемблеры.

Критерии: неверное определение СП: **-5**; есть только определение СП: **1** балл; отсутствие каждого компонента (кроме макрогенератора): **-3**; за каждое неверное (отсутствующее) описание: **-2**; каждая мелкая погрешность: **-1**.

8. Ответ: Библиотеки функций могут подключаться статически и динамически. ДБ в отличие от СтБ подключаются к программе не во время компиляции программы с помощью редактора связей, а непосредственно в ходе её выполнения. На этапе компоновки программы редактор связей, встречая вызовы функций ДБ, вместо выполнения процедуры связывания формирует таблицу точек вызова функций ДБ для последующей операции динамического связывания (то есть компоновщик не помещает в целевую программу тела функций ДБ). Таким образом, отложенный процесс полной компоновки завершится уже на этапе выполнения целевой программы.

Преимущества ДБ: (1) не требуется включать в программу объектный код часто используемых функций, что существенно сокращает объем кода; (2) различные программы, выполняемые в некоторой ОС, могут пользоваться кодом одной и той же библиотеки, содержащейся в ОП; (3) изменения и улучшения функций библиотек сводится к обновлению только содержимого динамически подключаемых библиотек, а уже существующие тексты программ не требуют перекомпиляции (этот же факт может оказаться недостатком, если при модификации функций каким-то образом меняется логика их работы, поэтому использование динамических библиотек накладывает определенные обязательства как на разработчика программы, так и на создателя библиотеки).

Критерии: есть только ответ на первый вопрос: **1** балл; ответ не на вопрос: **0** баллов; неверный (или отсутствует) ответ на второй вопрос: **-5**; за каждое не упомянутое преимущество: **-2**; отсутствие преимуществ статических библиотек — не снижать (можно отметить относительную простоту реализации; устойчивость к модификациям — все равно все перекомпилируется и какие-то несоответствия со старым вариантом могут обнаружиться на этапе трансляции).

9. Ответ:

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
	&b	&i	0	=	=	&S	a	i	a	b	+	*	<	20	!F	&b	++	
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
	22	!	&i	##	=	i	100	<	30	!F	6	!						

Соглашения: (1) Адреса в постфиксной записи можно обозначать подчеркиванием (a=&a), (2) Префиксный ++ обозначен в постфиксной записи как ##. (3) Различение префиксных и постфиксных ++ разрешается на уровне интерпретации. (4) "Мусор" в стеке после выполнения присваивания игнорируется.

Критерии: пропуск адреса в первом операнде операции(ий) присваивания: **-3**; пропуск адреса в операнде операции(ий) ++: **-3**; неверный ПОЛИЗ первого оператора: **-6**; неверный ПОЛИЗ цикла (любые, не указанные выше ошибки): **0** баллов.

Замечание. Допустимы следующие вариации ПОЛИЗа в ответах: (1) разрешается разворачивать унарные операции ++ и -- : префиксные — ++x ==> &x, x, 1, +, =; постфиксные — x++ ==> &y, x, =, &x, x, 1, +, =, y, где y — новый операнд; (2) может использоваться операция «;» для выбрасывания «мусора» из стека, при трактовке присваивания как операции, оставляющей результат присваивания на вершине стека. Соответственно изменяются адреса переходов в связи с (1) и/или (2).

10. Ответ: При выполнении машинно-зависимой оптимизации компилятор принимает во внимание те или иные составляющие архитектуры вычислительной системы (особенности аппаратных и программных средств), на которой результирующая программа будет выполняться. Основные типы МЗО: **1.** Оптимальное распределение регистров процессора. **2.** Оптимизация передачи параметров в процедуры и функции (через регистры). **3.** Оптимизация кода для процессоров, допускающих распараллеливание вычислений.

Пример: $a+b+c+d+e+f; \Rightarrow$

для одного потока обработки данных: $((((a+b)+c)+d)+e)+f;$

для двух потоков обработки данных: $((a+b)+c)+((d+e)+f);$

для трех потоков обработки данных: $(a+b)+(c+d)+(e+f);$

Машинно-зависимые преобразования производятся в основном над **результирующей объектной программой**.

Критерии: нет ответа на первый вопрос: **-3**; только ответ на 1-й вопрос или только на последний: **1** балл; отсутствие каждого типа: **-2**; неверный (отсутствует) ответ на последний вопрос: **-3**;

Экзамен, 2009 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $k = 1$ (третье и четвертое правила не удовл. типу 2). (б) регулярный – "нет", КЗ – "да", КС – "да", тип 0 – "да". (в) $L(G) = \{(cd)^n c a^{n+k} \mid n \geq 1, k \geq 1\} \cup \{\varepsilon\}$.

Критерии: (а) нет ответа; неверный ответ: -3 (обоснование не обязательно), (б) за каждый неверный или отсутствующий ответ: -1, всего до -4 за этот пункт (обоснование не обязательно), (в) описание отсутствует или с ошибками: -3.

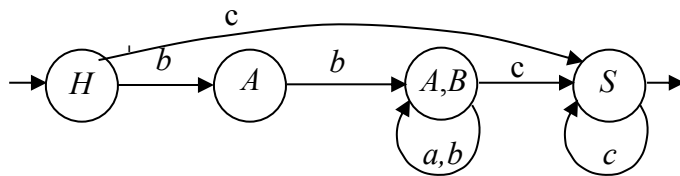
2. Ответ: (а) символ $A \in N$ называется *бесплодным* в грамматике $G = \langle T, N, P, S \rangle$, если множество $\{\alpha \in T^* \mid A \Rightarrow \alpha\}$ пусто. (б) **Нет.** Грамматика G_1 после применения алгоритма приведения не будет содержать ни одного правила (S — бесплодный символ), поэтому $L(G_1) = \emptyset$. Результатом применения алгоритма приведения к грамматике G_2 будет грамматика: $S \rightarrow AA \quad A \rightarrow \varepsilon$. Следовательно, $L(G_2) = \{\varepsilon\}$. Таким образом, $L(G_1) \neq L(G_2)$, так как $\varepsilon \in L(G_2)$, $\varepsilon \notin L(G_1)$.

Критерии: (а) нет ответа; определение с ошибками: -5, (б) нет ответа; неверный ответ; нет обоснования; неверное обоснование: -5. **Замечание.** Для обоснования неэквивалентности достаточно привести одну различающую цепочку. Ответ *почти эквивалентны* считать верным.

3. Ответ: (а) $G: S \rightarrow c \mid Sc \mid Ac \quad A \rightarrow Bb \mid Aa \quad B \rightarrow Aa \mid Bb \mid b$

(б) Разбор недетерминирован, т.к. автомат $\mathcal{A}$ не является детерминированным: из состояния A возможен переход в более чем одно состояние (по символу a). [или: т.к. из состояния B возможен переход в более чем одно состояние (по символу b)]. (в) Таблица переходов для $\mathcal{A}_D$: (начальное состояние: $\{H\}$, заключительное состояние: $\{S\}$).

$\mathcal{A}_D$:



	a	b	c
$\{H\}$	$\emptyset$	$\{B\}$	$\{S\}$
$\{B\}$	$\emptyset$	$\{A, B\}$	$\emptyset$
$\{A, B\}$	$\{A, B\}$	$\{A, B\}$	$\{S\}$
$\{S\}$	$\emptyset$	$\emptyset$	$\{S\}$

Допускаются другие способы построения, дающие в итоге такой же автомат $\mathcal{A}_D$ с точностью до обозначений вершин.

Критерии: (а) нет ответа; грамматика построена, но неэквивалентная или не соответствующая диаграмме: -3. (б) нет ответа; неверный ответ; ошибочное объяснение: -3. (в) нет ответа или программа на C++ приведена; построен неэквивалентный или недетерминированный автомат: -4.

4. Ответ: Описание буквы «Z» размера $n \times n$ — это цепочка $c^n f^n c^n$. Следовательно, требуется построить грамматику, порождающую язык $\{c^n f^n c^n \mid n \geq 1\}$:

$S \rightarrow CFS c \mid Cf c$

$FC \rightarrow CF$

$Ff \rightarrow ff$

$Cf \rightarrow ab$

$Cc \rightarrow cc$

Замечание. Грамматика $S \rightarrow CFS c \mid CF c$

$FC \rightarrow CF$

$F \rightarrow f$

$C \rightarrow c$ ошибочная, так как порождает лишние цепочки, напр.: $cfcfcc$.

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -3 (количество правил в условии дано с запасом).

5. Ответ: (а) Достаточное условие: каждая группа правил с одинаковой левой частью имеет один из перечисленных ниже видов и выполняются дополнительные условия: (1) либо $X \rightarrow \alpha$, где $\alpha \in (T \cup N)^*$ и это единственное правило вывода для этого нетерминала; (2) либо $X \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n$, где $a_i \in T$ для всех $i = 1, 2, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$, т. е. если для нетерминала X правил вывода несколько, то они должны начинаться с терминалов, причем все эти терминалы должны быть попарно различными; (3) либо $X \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n \mid \varepsilon$, где $a_i \in T$ для всех $i = 1, 2, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$, и $first(X) \cap follow(X) = \emptyset$.

Замечание. Допускаются обозначения V_T, V_N для обозначения алфавитов терминальных и нетерминальных символов, и обозначение V для их объединения. $V = T \cup N$ ($V = V_T \cup V_N$)

(б) Да, так как все непустые альтернативы начинаются с попарно различных терминалов, $first(S) \cap follow(S) = \emptyset$, и $first(B) \cap follow(B) = \emptyset$.

В задаче: $first(S) = \{a, b\}$, $follow(S) = \emptyset$; $first(B) = \{c\}$, $follow(B) = \{b\}$.

Критерии: (а) нет ответа, ошибки в формулировке: **-5**, **(б)** нет ответа; неверный ответ; нет обоснования; ошибочное обоснование: **-5**.

6. Ответ. Соглашение : запись $\langle a \rangle$ является сокращением $\langle cout << 'a' ; \rangle$

$L_{ex} = \{ c^k b^m \mid m \geq k \geq 0 \} = \{ c^k b^k b^{m-k} \mid m \geq k \geq 0 \}$ (входной язык). Грамматика для L_{ex} , к которой применим РС-метод, и грамматика с действиями выглядят так:

$S \rightarrow AB$	$S \rightarrow AB$	или :	$S \rightarrow \langle a \rangle AB$
$A \rightarrow cAb \mid \varepsilon$	$A \rightarrow cAb \langle d \rangle \mid \varepsilon \langle a \rangle$		$A \rightarrow c \langle d \rangle Ab \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$	$B \rightarrow b \langle b \rangle B \mid \varepsilon$		$B \rightarrow b \langle b \rangle B \mid \varepsilon$
$L_u = \{ ad^k b^{m-k} \mid m \geq k \geq 0 \}$ (целевой язык).			

Критерии: Грамматика не порождает L_{ex} , или не генерирует L_u : **0**; Порождает L_{ex} и формально генерирует соответствующие цепочки из L_u , но РС-метод неприменим: **-5**.

7. Ответ:

ЭТАП	ХАРАКТЕРИСТИКА
Анализ (определение) требований Постановка задачи	внешние спецификации ПП
Проектирование: - структуры ПП - отдельных модулей	схема иерархии и внешняя спецификация отдельных модулей; выбор алгоритма работы каждого модуля и способ внутреннего представления данных
Написание текста программ (кодирование) Компоновка или интеграция программного комплекса	алгоритм записывается на каком-либо алгоритмическом языке
Верификация, тестирование; отладка	проверка программы на правильность, выявление наличия ошибок; локализация и исправление ошибок
Документирование Внедрение Тиражирование Окончательная сдача ПП	написание текстовой документации, инструкций для пользователей и т.п.
Сопровождение (повторяющее все предыдущие этапы) и эксплуатация ПП	попытка приспособить ПП к измененным целям и исправление не выявленных на более ранних этапах ошибок

Критерии: за каждый пропущенный (крупный) этап, кроме последнего: **-2**; за каждое неверное или отсутствующее пояснение: **-1**; за отсутствие последнего этапа: **-1**.

8. Ответ: 1. Библиотеки функций (или подпрограмм). 2. Библиотеки классов. 3. Библиотеки компонент. **Библиотеки функций** во многом определяют возможности системы программирования в целом. Различают *библиотеки для языков программирования* (например, функции ввода-вывода, работа со строками) и *библиотеки для решения задач в конкретной проблемной области* (например, функции, реализующие алгоритмы линейной алгебры). Библиотеки функций представляют собой откомпилированные объектные модули. **Библиотеки классов** – для СП, базирующихся на ООЯП. Все их классы должны быть написаны на том же ЯП, на котором пишется программа, куда интегрируются библиотечные классы. В библиотеке классов различают *конкретные классы, абстрактные классы, шаблоны классов*. Библиотеки классов включаются в программу на этапе компиляции и компилируются со всей программой вместе. **Библиотеки компонентов** – это библиотеки готовых откомпилированных программных модулей, предназначенных для использования в качестве составной части программы, и которыми можно манипулировать во время разработки программы. Компоненты бывают локальные (находящиеся на той же ЭВМ, где создается ПП) и распределенные (расположенные на сервере и доступные по сети ЭВМ). Примеры технологий, использующих библиотечный компонент: CORBA, COM, DCOM, ActiveX, Java Beans.

Критерии: есть только перечисление типов библиотек: 1 балл; за каждую отсутствующую (неверную) характеристику: -3. **Замечание.** За отсутствие упоминаний о распределенных объектах и компонентах не снижать.

9. Ответ:

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
	&i	0	=	i	n	<	26	!F	15	!	&i	++	4	!	&a	b	c	
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
	-	d	-	=	&b	##	11	!										

Критерии: пропуск адреса в первом операнде операции(ий) присваивания: -3; пропуск адреса в операнде операции(ий) ++ : -3; неверный ПОЛИЗ первого оператора: -6; неверный ПОЛИЗ цикла (любые, не указанные выше ошибки): 0 баллов.

Замечание. Допустимы следующие вариации ПОЛИЗа в ответах: (1) разрешается разворачивать унарные операции ++ и -- : префиксные: ++x ==> &x, x, 1, +, =; постфиксные: x++ ==> &u, x, =, &x, x, 1, +, =, u, где u — новый операнд; (2) может использоваться операция «>» для выбрасывания «мусора» из стека, при трактовке присваивания как операции, оставляющей результат присваивания на вершине стека. Соответственно изменяются адреса переходов в связи с (1) и/или (2).

10. Ответ:

1. Удаление недостижимого кода.
2. Оптимизация линейных участков программы:
 - а) Удаление бесполезных присваиваний.
 - б) Исключение избыточных вычислений.
 - в) Свертка объектного кода (выполнение во время компиляции тех операций исходной программы, для которых значения операндов уже известны).
 - г) Перестановка операций (для дальнейшей свертки или оптимизации вычислений).
 - д) Арифметические преобразования (на основе алгебраических и логических тождеств).
 - е) Оптимизация вычисления логических выражений.

3. Подстановка кода функции вместо ее вызова в объектный код.

4. Оптимизация циклов: а) Вынесение инвариантных вычислений из циклов. б) Замена операций с индуктивными (образующими арифметическую прогрессию) переменными (как правило, умножения на сложение). в) Слияние и развертывание циклов. Машинно-независимые преобразования исходной программы производятся **в форме ее внутреннего представления.**

Критерии: только ответ на последний вопрос: 1 балл; отсутствие каждого типа (арабская цифра): -3; за каждую неточность внутри: -1 (не больше -3 для одной арабской цифры); неверный (отсутствует) ответ на последний вопрос: -3.

Экзамен, 2009 год. Ответы и критерии оценок.

Вариант 3.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $k = 2$ ($C \rightarrow bB$ не удовлетворяет определению левостроительной грамматики, $B \rightarrow Cd$ не удовлетворяет определению правостроительной). (б) КС – " ∂a ", КЗ – " ∂a ", регулярный – "**нет**", тип 0 – " ∂a ". (в) $L(G) = \{ b^n d^{n+2} ca^k \mid n \geq 0, k \geq 0 \} \cup \{ a^k \mid k \geq 0 \}$

Критерии: (а) нет ответа; неверный ответ : **-3** (обоснование не обязательно), (б) за каждый неверный или отсутствующий ответ: **-1**, всего до **-4** за этот пункт (обоснование и формула языка не обязательно), (в) описание отсутствует или с ошибками: **-3**

2. Ответ: (а) КС-грамматика G называется *приведенной*, если в ней нет недостижимых и бесплодных символов. (б) **Нет**. Грамматика G_2 после приведения:

$S \rightarrow BA$
 $A \rightarrow bAa \mid a \mid \varepsilon$
 $B \rightarrow b$

Следовательно, $L(G_1) \neq L(G_2)$, т.к. $\varepsilon \notin L(G_2)$, $\varepsilon \in L(G_1)$.

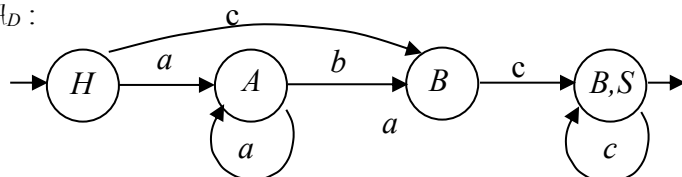
Критерии: (а) нет ответа; неверный ответ : **-5**, (б) нет ответа; неверный ответ; нет обоснования; неверное обоснование: **-5**. **Замечание.** Для обоснования неэквивалентности достаточно привести одну различающую цепочку. Ответ **почти эквивалентны** считать верным.

3. Ответ: (а) $G: S \rightarrow Bc$

$B \rightarrow Bc \mid Ab \mid c$
 $A \rightarrow Aa \mid Sa \mid a$

(б) Разбор недетерминирован, т.к. автомат $\mathcal{A}$ не является детерминированным: из состояния B возможен переход в более чем одно состояние по символу c . (в) Таблица переходов для $\mathcal{A}_D$: (начальное состояние: $\{H\}$, заключительное состояние: $\{S\}$).

$\mathcal{A}_D$:



	a	b	c
$\{H\}$	$\{A\}$	$\emptyset$	$\{B\}$
$\{A\}$	$\{A\}$	$\{B\}$	$\emptyset$
$\{B\}$	$\emptyset$	$\emptyset$	$\{B, S\}$
$\{B, S\}$	$\{A\}$	$\emptyset$	$\{B, S\}$

Допускаются другие способы построения, дающие в итоге такой же автомат $\mathcal{A}_D$ с точностью до обозначений вершин.

Критерии: (а) нет ответа; построена неэквивалентная или не соответствующая диаграмме грамматики: **-3**. (б) нет ответа; неверный ответ; ошибочное объяснение: **-3**. (в) нет ответа или программа на C++ приведена; построен неэквивалентный или недетерминированный автомат: **-4**.

4. Ответ: Описание буквы «И» размера $n \times n$ — это цепочка $e^n d^n e^n$. Следовательно, требуется построить грамматику, порождающую язык $\{ e^n d^n e^n \mid n \geq 1 \}$:

$S \rightarrow EDSe \mid Ede$ **Замечание.** Грамматика $S \rightarrow EDSe \mid Ede$
 $DE \rightarrow ED$ $DE \rightarrow ED$
 $Dd \rightarrow dd$ $D \rightarrow d$
 $Ed \rightarrow ed$ $E \rightarrow e$ ошибочная, так как порождает
 $Ee \rightarrow ee$ лишние цепочки, напр.: $ededee$.

Критерии: порождается пустая цепочка: **-5**, имеется лишняя (непустая) или недостающая цепочка: **-10**, за каждое лишнее правило: **-3** (количество правил в условии дано с запасом).

5. Ответ: (а) Достаточное условие: каждая группа правил с одинаковой левой частью имеет один из перечисленных ниже видов и выполняются дополнительные условия: (1) либо $X \rightarrow \alpha$, где $\alpha \in (T \cup N)^*$ и это единственное правило вывода для этого нетерминала; (2) либо $X \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n$, где $a_i \in T$ для всех $i = 1, 2, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$, т. е. если для нетерминала X правил вывода несколько, то они должны начинаться с терминалов, причем все эти терминалы должны быть попарно различными; (3) либо $X \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n \mid \varepsilon$, где $a_i \in T$ для всех $i = 1, 2, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$, и $\text{first}(X) \cap \text{follow}(X) = \emptyset$. **Замечание.** Допускаются обозначения V_T, V_N для обозначения алфавитов терминальных и нетерминальных символов, и обозначение V для их объединения. $V = T \cup N$ ($V = V_T \cup V_N$)

(б) нет, не выполняется: для правила $A \rightarrow aAd \mid bC \mid aC$, имеющего вид (2), нарушено дополнительное условие, т.к. $\text{first}(aAd) \cap \text{first}(aC) = \{a\} \neq \emptyset$ (две альтернативы начинаются одинаковыми терминалами).

Критерии: (а) нет ответа, ошибки в формулировке: **–5**, **(б)** нет ответа; неверный ответ; нет обоснования; ошибочное обоснование: **–5**.

6. Ответ. Соглашение : запись $\langle a \rangle$ является сокращением $\langle \text{cout} < < 'a ' \rangle$

$L_{ex} = \{ c^n d^m \mid m \geq n \geq 0 \} = \{ c^n d^n d^{m-n} \mid m \geq n \geq 0 \}$ (входной язык). Грамматика для L_{ex} , к которой применим РС-метод, и грамматика с действиями выглядит так:

$S \rightarrow AB$	$S \rightarrow A\langle a \rangle B$	или : $S \rightarrow A\langle a \rangle B$	или : $S \rightarrow AB$
$A \rightarrow cAd \mid \varepsilon$	$A \rightarrow cAd \langle b \rangle \mid \varepsilon$	$A \rightarrow c \langle b \rangle Ad \mid \varepsilon$	$A \rightarrow c \langle b \rangle Ad \mid \varepsilon \langle a \rangle$
$B \rightarrow dB \mid \varepsilon$	$B \rightarrow d \langle d \rangle B \mid \varepsilon$	$B \rightarrow d \langle d \rangle B \mid \varepsilon$	$B \rightarrow d \langle d \rangle B \mid \varepsilon$

$L_{cl} = \{ b^n a d^{m-n} \mid m \geq n \geq 0 \}$ (целевой язык).

Критерии: Грамматика не порождает L_{ex} , или не генерирует L_{cl} : **0**; Порождает L_{ex} и формально генерирует соответствующие цепочки из L_{cl} , но РС-метод неприменим: **–5**.

7. Ответ: Современная ИСР — комплекс интегрированных программных средств, поддерживающих полный жизненный цикл ПП. Основные преимущества: (1) ИСР объединяет в себе возможности текстовых редакторов исходных текстов программ, отладчиков и командный язык компиляции. Программист, не прерывая работу с ИСР, может в удобной форме воспользоваться услугами любого, входящего в ее состав компонента; (2) ИСР обеспечивает многооконный интерфейс с поддержкой режима "буксировки" фрагментов текста мышкой (drag&drop); (3) в текстовом редакторе ИСР благодаря интеграции с компилятором реализованы а) синтаксическая подсветка элементов языка (выделение цветом разных лексем), б) дополнение кода, интерактивная подсказка, в) всплывающие подсказки об атрибутах идентификаторов, если на них установить курсор, г) отображение ошибок, обнаруженных на этапе компиляции, в тексте программы; (4) благодаря интеграции с отладчиком в текстовом редакторе ИСР реализованы, например, а) отображение контрольных точек останова при отладке, б) отображение текущего значения объекта, при наведении курсора на идентификатор, в) выполнение программы до строки, в которой в редакторе стоит курсор, г) выделение выполняемой в данный момент строки; (5) часто ИСР поддерживает репозиторий программного проекта, являющийся основой ИСР, который обеспечивает хранение информации о проекте на всех этапах его жизненного цикла, версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль данных о проекте на полноту и непротиворечивость; (6) как правило, ИСР содержит графические средства анализа и проектирования ПП.

Критерии: Есть только определение ИСР: **1 балл**; безошибочно указано только одно из достоинств ИСР: **–6**; безошибочно указаны только два из достоинств ИСР: **–3**.

Замечание. Допускается «описательное» определение ИСР через свойства, например: «объединяет в себе возможности текстовых редакторов и командного языка компилятора, а также некоторых других компонентов системы программирования, например, редактора связей и отладчика».

8. Ответ: **Общезначимость** содержимого (предоставлять программисту нетривиальные средства, на которые он может рассчитывать, заботясь о переносимости программ, например, средства работы со списками, функции сортировки, потоки ввода/вывода); **Эффективность** (предоставлять функции, которые не могут быть написаны оптимально для всех вычислительных систем на данном языке программирования, например, функции вычисления квадратного корня `sqrt()` или пересылок блоков памяти `memmove()`); **Безопасность** (использование библиотеки не должно провоцировать ошибки, а наоборот, снижать их вероятность); **Завершенность** (библиотека должна обеспечить достаточную функциональность, чтобы ни у кого не возникало желания что-то заменить или доопределить); **Сочетаемость с базовыми типами данных** (предоставлять основу для расширения собственных возможностей, в частности, соглашения и средства поддержки, позволяющие обеспечить операции для данных, имеющих определяемые пользователями типы, в том же стиле, в котором обеспечиваются операции для встроенных типов, например, ввод/вывод); Может служить **фундаментом** для создания других библиотек; **Поддержка свойств языка**, (например, управление памятью и предоставление информации об объектах во время выполнения программ); **Предоставление информации** о зависящих от реализации аспектах языка, например, о максимальных размерах целых значений.

Критерии: указаны и верно прокомментированы не менее 5 критериев: **10** баллов; если менее 5, то отсутствие (ошибочность) любого критерия: **-2** (всего до **-10**); отсутствие каждого соответствующего критерию краткого комментария: **-1**.

9. Ответ:

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ	a	b	<	28	!F	&S	a	0	<	16	!F	a	@	18	!	&b	++

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
+=	&b	b	a	-	1	-	=	1	!								}

Соглашения: (1) Адреса в постфиксной записи можно обозначать подчеркиванием (`a=&a`), (2) Префиксный `++` обозначен в постфиксной записи как `##`. (3) Различение префиксных и постфиксных `++` разрешается на уровне интерпретации. (4) "Мусор" в стеке после выполнения присваивания игнорируется.

Критерии: пропуск адреса в первом операнде операции(ий) присваивания: **-3**; пропуск адреса в операнде операции(ий) `++`: **-3**; неверный ПОЛИЗ первого оператора: **-6**; неверный ПОЛИЗ цикла (любые, не указанные выше ошибки): **0** баллов.

Замечание. Допустимы следующие вариации ПОЛИЗа в ответах: (1) разрешается разворачивать унарные операции `++` и `--`: префиксные: `++x ==> &x, x, 1, +, =`; постфиксные: `x++ ==> &y, x, =, &x, x, 1, +, =, y`, где `y` — новый операнд; (2) может использоваться операция «;» для выбрасывания «мусора» из стека, при трактовке присваивания как операции, оставляющей результат присваивания на вершине стека. Соответственно изменяются адреса переходов в связи с (1) и/или (2).

10. Профилировщик – компонент СП, который строит профиль программы (где выделены линейные участки кода) с указанием того, какой процент времени они выполняются тот или иной фрагмент кода.

Используется для более эффективной оптимизации программы, т.к. обычно 90% времени выполнения программы тратится на 10% ее кода, зная которые можно не оптимизировать оставшиеся 90% и избежать возможных ошибок при оптимизации, а также при отладке программ. (Процентное соотношение указано приблизительно).

Критерии: неверный (отсутствует) ответ на первый вопрос: **-4**; неверный (отсутствует) ответ на второй вопрос: **-4**; неверный (отсутствует) ответ на третий вопрос: **-4**.

Замечание. Все три ответа могут быть сформулированы "своими словами".

Экзамен, 2009 год. Ответы и критерии оценок.

Вариант 4.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $k = 3$ (все правила удовлетворяют определению левосторонней грамматики), (б) КС – " ∂a ", КЗ – " ∂a ", тип 0 – " ∂a ", регулярный – " ∂a "

(в) $L(G) = \{ aa (bca)^n cd \mid n \geq 0 \} \cup \{ \varepsilon \}$

Критерии: (а) нет ответа; неверный ответ : -3 (обоснование не обязательно), (б) за каждый неверный или отсутствующий ответ: -1, всего до -4 за этот пункт (обоснование и формула языка не обязательно), (в) описание отсутствует или с ошибками: -3

Ответ: (а) Да, можно. Существует алгоритм удаления правил с пустой правой частью для КС-грамматик, который по любой КС-грамматике строит эквивалентную неукорачивающую КС-грамматику. (б) Эквивалентны. Грамматика G_1 получается из G_2 применением алгоритма удаления пустых правых частей. **Замечание.** Ссылку на утверждение «для любой КС-грамматики существует эквивалентная неукорачивающая КС-грамматика» тоже можно считать обоснованием.

Критерии: (а) нет ответа; определение с ошибками: -5, (б) нет ответа; неверный ответ; нет обоснования; неверное обоснование: -5.

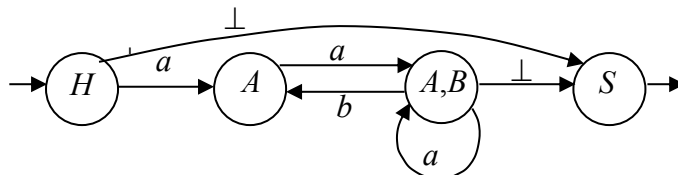
3. Ответ: (а) $G: S \rightarrow \perp \mid B\perp$

$B \rightarrow Ba \mid Aa$

$A \rightarrow Aa \mid Bb \mid a$

(б) Разбор недетерминирован, т.к. автомат $\mathcal{A}$ не является детерминированным: из состояния A возможен переход в более чем одно состояние (по символу a) [или: из состояния B возможен переход в более чем одно состояние (по символу b)]. (в) Таблица переходов для $\mathcal{A}_D$: (начальное состояние: $\{H\}$, заключительное состояние: $\{S\}$).

$\mathcal{A}_D$:



	a	b	$\perp$
$\{H\}$	$\{A\}$	$\emptyset$	$\{S\}$
$\{A\}$	$\{A, B\}$	$\emptyset$	$\emptyset$
$\{A, B\}$	$\{A, B\}$	$\{A\}$	$\{S\}$
$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$

Допускаются другие способы построения, дающие в итоге такой же автомат $\mathcal{A}_D$ с точностью до обозначений вершин.

Критерии: (а) нет ответа; грамматика построена, но неэквивалентная или не соответствующая диаграмме : -3. (б) нет ответа; неверный ответ; ошибочное объяснение: -3. (в) нет ответа или программа на C++ приведена; построен неэквивалентный или недетерминированный автомат: -4.

4. Ответ: Описание буквы «П» размера $n \times m$ — это цепочка $c^n a^m e^n$. Следовательно, требуется построить грамматику, порождающую язык $\{ c^n a^m e^n \mid n, m \geq 1 \}$:

$S \rightarrow cSe \mid cAe$
 $A \rightarrow aA \mid a$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -3 (количество правил в условии дано с запасом).

5. Ответ: (а) $S \rightarrow bSA \mid a$

$B \rightarrow eB \mid \varepsilon$

$A \rightarrow aB \mid cAb$

(б) Да, удовлетворяет: $first(bSA) = \{b\}$, $first(a) = \{a\}$, $first(bSA) \cap first(a) = \emptyset$;
 $first(B) = \{e\}$, $follow(B) = \{a, b, c\}$, $first(B) \cap follow(B) = \emptyset$;
 $first(aB) = \{a\}$, $first(cAb) = \{c\}$, $first(aB) \cap first(cAb) = \emptyset$.

Критерии: (а) нет ответа, неверно восстановленная грамматика: -5, (б) нет ответа; ответ не обоснован: -5, ответ верный, за ошибки в обосновании: до -5.

6. Ответ. Соглашение: запись $\langle a \rangle$ является сокращением $\langle cout \ll 'a'; \rangle$

Грамматика с действиями: $S \rightarrow cAb$ или $S \rightarrow cAb \langle a \rangle$
 $A \rightarrow dS \langle a \rangle \mid c \langle b \rangle d \langle a \rangle \langle a \rangle$ $A \rightarrow dS \mid c \langle b \rangle d \langle a \rangle$

Также возможна перестановка $\langle b \rangle$ непосредственно за символ d в обоих вариантах ответа. Любая другая расстановка действий либо не дает подходящую цепочку из $L_2 = \{ b(a)^{k+1} \mid k \geq 1 \}$, либо не соответствует методу рекурсивного спуска.

Критерии: За каждое неправильно вставленное (или пропущенное) действие вида $\langle cout \ll 'символ'; \rangle$: **-4**. Если, кроме действий по печати символов $\langle cout \ll 'символ'; \rangle$, есть другие вычисления: **0**.

7. Ответ: $исх.пр-ма \Rightarrow$ **Фаза анализа** [$ЛА \Rightarrow$ посл.лексем + таблицы $\Rightarrow$ $СА + КУ \Rightarrow$ внутр.предст.] + **Фаза синтеза** [$(*) \Rightarrow$ ГЕН] $\Rightarrow$ объектная программа, **либо** $исх.пр-ма \Rightarrow$ **Фаза анализа** [$ЛА+СА+КУ \Rightarrow$ внутр.предст. + таблицы] + **Фаза синтеза** [$(*) \Rightarrow$ ГЕН] $\Rightarrow$ объектная программа

$(*)$ – подготовка к генерации объектной программы – распределение памяти, оптимизация.

Критерии: за пропуск любого этапа: **-3**; за пропуск любых входных/выходных данных: **-2**; не указана фаза синтеза (остальное есть): **5** баллов.

8. Ответ: **1.** Библиотеки функций (или подпрограмм). **2.** Библиотеки классов. **3.** Библиотеки компонент. **Библиотеки функций** во многом определяют возможности системы программирования в целом. Различают *библиотеки для языков программирования* (например, функции ввода-вывода, работа со строками) и *библиотеки для решения задач в конкретной проблемной области* (например, функции, реализующие алгоритмы линейной алгебры). Библиотеки функций представляют собой откомпилированные объектные модули. **Библиотеки классов** – для СП, базирующихся на ООЯП. Все их классы должны быть написаны на том же ЯП, на котором пишется программа, куда интегрируются библиотечные классы. В библиотеке классов различают *конкретные классы, абстрактные классы, шаблоны классов*. Библиотеки классов включаются в программу на этапе компиляции и компилируются со всей программой вместе. **Библиотеки компонентов** – это библиотеки готовых откомпилированных программных модулей, предназначенных для использования в качестве составной части программы, и которыми можно манипулировать во время разработки программы. Компоненты бывают локальные (находящиеся на той же ЭВМ, где создается ПП) и распределенные (расположенные на сервере и доступные по сети ЭВМ). Примеры технологий, использующих библиотечный компонент: CORBA, COM, DCOM, ActiveX, Java Beans.

Критерии: есть только перечисление типов библиотек: **1** балл; за отсутствие (неверное представление) какого-либо типа библиотек: **-3**; за каждый не указанный способ подключения: **-2**.

Замечание. За отсутствие упоминаний о распределенных объектах и компонентах не снижать.

9. Ответ:

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
	a	b	<	21	!F	&S	S	a	-	c	-	=	&a	++	&b	b	@	
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
	=	28	!	&b	&c	b	a	-	=	=								

Критерии: пропуск адреса в первом операнде операции(ий) присваивания: **-3**; пропуск адреса в операнде операции(ий) ++ : **-3**; неверный ПОЛИЗ первого оператора: **-6**; неверный ПОЛИЗ цикла (любые, не указанные выше ошибки): **0** баллов.

Замечание. Допустимы следующие вариации ПОЛИЗа в ответах: (1) разрешается разворачивать унарные операции ++ и -- : префиксные: ++x ==> &x, x, 1, +, =; постфиксные: x++ ==> &y, x, =, &x, x, 1, +, =, y, где y — новый операнд; (2) может использоваться операция «;» для выбрасывания «мусора» из стека, при трактовке присваивания как операции, оставляющей результат присваивания на вершине стека. Соответственно изменяются адреса переходов в связи с (1) и/или (2).

10. Ответ: Выделяемые области динамической памяти можно разделить на области памяти, выделяемые пользователем (явно) и непосредственно компилятором (неявно). (1) **Явное выделение блоков фиксированного размера.** При этом блоки памяти связываются в список, над которым легко выполнять операции их выделения и освобождения. Плюсы – если программа полностью использует блок памяти, то никаких дополнительных расходов не нужно. С каждым блоком связан указатель на его начало, и надо хранить только информацию о том, занят блок или нет. **Явное выделение блоков переменного размера.** Основная проблема – фрагментация памяти, т.к. нельзя выделить блок, больший имеющихся свободных (необходимо приостанавливать выполнение программы и начинать реорганизацию кучи). Аналогичная проблема для блоков фиксированного размера не представляет сложности. Один из методов выделения блоков переменного размера называется *методом первого подходящего*. (2) **Неявное выделение/освобождение памяти.** Неявно выделяемые блоки памяти также могут быть фиксированного или переменного размера. При неявном динамическом выделении и освобождении блоков памяти, выделяемые блоки обычно имеют следующую структуру: размер блока (для блоков фиксированного размера эта информация не хранится.); счётчик ссылок, либо пометка: **Счётчик** ссылок подсчитывает количество указателей в программе, которые ссылаются на этот блок (если происходит присваивание указателей, например, $p = q$, то один счётчик (для блока, на который указывал указатель p) уменьшается на единицу, а другой увеличивается), если счётчик равен нулю, то блок не используется и его можно освободить. Существует проблема циклических ссылок, когда счётчики всегда > 0 . **Пометка** фиксирует, задействован данный блок или не задействован. Т.е. у программы есть хотя бы один указатель, ссылающийся на этот блок. В некоторый момент начинает работать «сборщик мусора». Он помечает все блоки как недостижимые, а потом начинает анализ текущих указателей программы, что является очень трудоемким процессом. Блоки, на которые ничего не указывает, считаются свободными и их можно перегруппировать и т.п. по желанию разработчика компилятора.

Допускается более краткий ответ с указанием среди проблем в случае фиксированного размера – неэффективное использование памяти и ее быстрое исчерпание, а случае переменного размера — поиск пригодного блока и реорганизация свободного пространства при отсутствии такого блока. Описывать подробно алгоритмы контроля свободного пространства не обязательно.

Критерии: есть только перечисление основных методов выделения памяти: 1 балл; за каждую неверную (отсутствующую) характеристику: -3; за каждую не указанную проблему использования: -2.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Описать класс A таким образом, чтобы все конструкции функции `main()` были верными, а на экран выдалось **50 300**.

```
int main () {
    A a1 (5),    a2  = 3;
    a1 *= 10;
    a2 *=      a1 *= 2;
    cout << a1.get() << ' ' << a2.get() << endl;
    return 0;
}
```

Изменять функцию `main()`, использовать механизм исключений, а также функции `exit()`, `terminate()` и т.п. для досрочного завершения программы **запрещается**.

2. Если в реализации функций `B::g()` и `main()` есть ошибки, **объясните**, в чем они заключаются. В теле функций `main()` и `B::g()` к именам **каждой** правильной конструкции **этих функций** добавьте префикс, указывающий с помощью операции «`::`», из какой области видимости они выбираются.

```
int x = 0;
void f (int a, int b){ x = a + b;}
class A { int x;
public:    void f () { x = 2; }
          void f (char a1, char b1){ x = a1 - b1; } };
class B: public A {
public:    void f (int a )  { ::x = a; }
          void g ()        { f ();          f (0);
                             f (5.3 , 1); x = 1;          } };
int main () { B b;          f (2);          f (3 , 'a'); return 0; }
```

3. Дан текст фрагмента программы. Есть ли в этом тексте ошибки? Если да, то в чём они заключаются?

```
struct A { int i;
    virtual void f() = 0;
    virtual ~A() {} };
int g(A a) { return a.i * 5; }
```

4. Есть ли ошибки в следующих заголовках шаблонов? Если есть, поясните, в чём они заключаются.

```
template <class Cs> void func(const Cs& ref) { /*...*/ }
template <int n>    void func(int t = n)    { /*...*/ }
template <double f> void func(double d = f) { /*...*/ }
```

5. Что напечатает следующая программа?

```
class A {
    int i;
public:
    A (int x)    { i = x; printf("first\n"); }
    virtual ~A () { printf("second\n"); }
    int f () const { return i + g() + h(); }
    virtual int g () const { return i; }
    int h () const { return 39; } };
class B : public A {
public:
    B () : A(70) { printf("third\n"); }
    ~B ()        { printf("fourth\n"); }
    int f () const { return g() - 2; }
    virtual int g () const { return 4; }
    int h () const { return 6; } };
int main() {
    B b;
    A * p = & b;
    printf ("result = (%d ; %d)\n", p->f(), b.f());
    return 0;
}
```

6. Опишите класс A таким образом, чтобы были верными все конструкции следующего фрагмента программы:

```
int A::x;
int main () { const A a;
               a.x = 1;
               a.get_0();
               return 0; }
```

7. Даны описание класса и функции. Дополните описание класса Cls (не изменяя описание функции f()) так, чтобы только описание f() стало ошибочным.

```
class Cls {      int i;
public:          Cls() { i = 1; }
};
void f(Cls *p, Cls *q) { * p = * q; }
```

8. Что напечатает следующая программа?

```
class I { int i;
public: I() : i(9) { printf ("sun\n"); }
       I(int a) : i(a) { printf ("venus %d\n", i); }
       I(const I& other) : i(other.i) { printf ("earth %d\n", i); }
       ~I() { printf ("moon\n"); }
       int Get() { return i; }
       void operator +=(const I& op) { i+=op.i; }
};
void f(I& x, I y) { y += 1000; x += y; }
int main() { I i1;
             I i2(20);
             i2 += 400;
             f (i1, i2);
             printf("%d %d\n", i1.Get(), i2.Get());
             return 0;
}
```

9. ООП. Дать определение инкапсуляции. Что такое АТД? Каким образом АТД реализуется в C++?

10. Что напечатает следующая программа?

```
class Ex { int code;
public:   Ex (int i) : code(i) {}
         Ex (const Ex& ex) : code(ex.code) {}
         int Get() const { return code; }
};
struct Ex90 : Ex { Ex90() : Ex(90) {} };
void f() { throw Ex90();
           printf("dog\n");
}
void t() { try { f(); }
          catch (Ex90 &x) { printf ("cat\n");
                           throw Ex (x.Get() + 1);
                           printf ("sheep\n");
                           }
          catch (Ex &) { printf ("horse\n"); }
          printf ("cow\n");
}
int main() { try { t(); }
            catch (Ex &x) { printf ("elephant %d\n", x.Get()); }
            catch (...) { printf ("wolf\n"); }
            return 0;
}
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Описать класс В таким образом, чтобы все конструкции функции `main()` были верными, а на экран выдалось **10 20 30.**

```
int main () {
    B b1, b2 = b1, b3 (b2);
    cout << b1.get() << ' ' << b2.get() << ' ' << b3.get() << endl;
    return 0;
}
```

Изменять функцию `main()`, использовать механизм исключений, а также функции `exit()`, `terminate()` и т.п. для досрочного завершения программы **запрещается.**

2. Если в реализации функций `D::h()` и `main()` есть ошибки, объясните, в чем они заключаются. К именам **каждой** правильной конструкции **этих функций** добавьте префикс, указывающий с помощью операции «`::`», из какой области видимости имени выбираются.

```
double a = 0;
void f(double x = 2) { a = x; }
void f() { a = 1; }
struct B { double a;
           void f () { a = 2; } };
class D: B {
public:     void f (int a) { ::a = a; }
           void h ()      { f ('r');      f();
                           a = 2;          } };
int main () { D d;         f();           f(6); return 0; }
```

3. Дан текст описания двух структур. Есть ли в этом тексте ошибки? Если да, то в чём они заключаются?

```
struct S { virtual void f() const = 0;
           virtual ~S() {} };
struct A { S s;
           int i;
};
```

4. Есть ли ошибки в следующих заголовках шаблонов? Если есть, поясните, в чём они заключаются.

```
template <int n>     class A { /*...*/ };
template <float f>   class B { /*...*/ };
template <class Cs> class C { /*...*/ };
```

5. Что напечатает следующая программа?

```
class A {     int i;
public:       A (int x) { i = x;   printf ("mercury\n"); }
             virtual ~A () {      printf ("venus\n"); }
             int f () const { return 96; }
             virtual int g () const { return i; }
             int h () const { return i - f() - g(); }
};
class B : public A {
public:       B (int x) : A(x+20) { printf ("earth\n"); }
             ~B () { printf ("mars\n"); }
             int f () const { return 8; }
             virtual int g () const { return 3; }
             int h () const { return f() + g(); }
};
int main() { B   b(17);
             A * p = & b;
             printf("result = (%d ; %d)\n", p->h(), b.h());
             return 0;
}
```


6. Опишите класс C таким образом, чтобы были верными все конструкции следующего фрагмента программы:

```
const char C::c = '+';
int main () { C ob;
              C::f();
              return 0; }
```

7. Даны описания структуры, переменной и функции. Дополните описание структуры mystr (не изменяя описание функции f()) так, чтобы только описание f() стало ошибочным.

```
struct mystr { int a, b; };
int i = sizeof(mystr);
int f (mystr s) { return 0; }
```

8. Что напечатает следующая программа?

```
class I { int i;
public: I() : i(6) { printf("owl\n"); }
       I(int a) : i(a) { printf("sheep %d\n", i); }
       I(const I& other) : i(other.i) { printf("horse %d\n", i); }
       ~I() { printf("wolf\n"); }
       int Get() { return i; }
       void operator *=(const I& op) { i*=op.i; }
};
void f(I x, I &y) { x *= 1; y *= x; }
int main() { I i1;
            I i2(3);
            i1 *= 7;
            f (i1, i2);
            printf("%d %d\n", i1.Get(), i2.Get());
            return 0;
}
```

9. Пусть производный класс получен из базового способом открытого (public) наследования и имена добавленных в производном классе членов отличаются от имен базового класса. В чем отличие статуса закрытых (private) и открытых (public) членов базового класса в полученном производном классе?

10. Что напечатает следующая программа?

```
class Ex { int code;
public: Ex (int i) : code(i) {}
       Ex (const Ex& ex) : code(ex.code) {}
       int Get() const { return code; }
};
struct Ex60 : Ex { Ex60() : Ex(60) {} };
void f() { throw Ex60();
          printf ("sword\n");
}
void t() { try { f(); }
          catch (Ex60 &x) { printf ("lance\n");
                           throw Ex (x.Get() + 1);
                           printf ("dagger\n");
          }
          catch (Ex &) { printf ("knife\n"); }
          printf ("hammer\n");
}
int main() { try { t(); }
            catch (Ex &x) { printf ("arche %d\n", x.Get()); }
            catch(...) { printf ("pistole\n"); }
            return 0;
}
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Описать класс C таким образом, чтобы все конструкции функции `main ()` были верными, а на экран выдалось **14 10 48.**

```
int main () {
    C c1(7), c2 = 5, c3(c1+c2);
    cout << c1.get() << ' ' << c2.get() << ' ' << c3.get() << endl;
    return 0;
}
```

Изменять функцию `main()`, использовать механизм исключений, а также функции `exit()`, `terminate()` и т.п. для досрочного завершения программы **запрещается.**

2. Если в реализации функций `S::g()` и `main()` есть ошибки, объясните, в чем они заключаются. К именам каждой правильной конструкции этих функций добавьте префикс, указывающий с помощью операции «`::`», из какой области видимости имена выбираются.

```
float y = 0;
void f (float a) { y = a; }
class T { int y;
public: void f() { y = 2; } };
class S: public T {
public: void f (float n, float m) { ::y = n * m; }
       void f (char c1, char c2) { ::y = c1 + c2; }
       void g () { f (); f(1);
                  f (-1 , 1); y = 2; } };
int main () { S b; f (5); f ('+', 6); return 0; }
```

3. Дан текст фрагмента программы. Есть ли в этом тексте ошибки? Если да, то в чём они заключаются?

```
class B {
public: virtual int f () = 0;
        int g () { return f() * 10; }
        virtual ~B () {}
};
int h(B b) { return b.g() + 2; }
```

4. Есть ли ошибки в следующих заголовках шаблонов? Если есть, поясните, в чём они заключаются.

```
struct mystruct { int a, b; };
template <int n> void func(int t = n) { /*...*/ }
template <mystruct a> void func(mystruct *p = &a) { /*...*/ }
template <class Cs> void func(const Cs& ref) { /*...*/ }
```

5. Что напечатает следующая программа?

```
class A {
    int i;
public:
    A (int x) { i = x; printf("dog\n"); }
    virtual ~A () { printf("cat\n"); }
    int f () const { return i + g() + h(); }
    virtual int g () const { return i; }
    int h () const { return 5; }
};
class B : public A {
public:
    B (): A(21) { printf("sheep\n"); }
    ~B () { printf("horse\n"); }
    int f () const { return g() - 3; }
    virtual int g () const { return 7; }
    int h () const { return 9; }
};
int main() {
    B b;
    A * p = &b;
    printf("result = (%d ; %d)\n", p->f(), b.f());
    return 0;
}
```

6. Опишите класс X таким образом, чтобы были верными все конструкции следующего фрагмента программы:

```
int main () { const X x;
               X::g();
               x.h();
               return 0;
            }
```

7. Как нужно описать структуру smartstr, чтобы создать переменную типа smartstr было можно, но при этом было невозможно создать массив элементов типа smartstr в динамической памяти? Приведите пример такого описания.

8. Что напечатает следующая программа?

```
class I { int i;
public: I() : i(5) { printf ("fist\n"); }
       I(int a) : i(a) { printf ("lance %d\n", i); }
       I(const I& other) : i(other.i) { printf ("dagger %d\n", i); }
       ~I() { printf ("pistole\n"); }
       int Get() { return i; }
       void operator +=(const I& op) { i+=op.i; }
};
void f(I& x, I y) { y += 1000; x += y; }
int main() { I i1;
             I i2(30);
             i2 += 700;
             f (i1, i2);
             printf("%d %d\n", i1.Get(), i2.Get());
             return 0;
        }
```

9. ООП. Что такое полиморфизм? Какие виды полиморфизма реализованы в C++, с помощью каких механизмов? Когда эти механизмы работают, во время выполнения программы или во время работы компилятора?

10. Что напечатает следующая программа?

```
class Ex { int code;
public:   Ex (int i) : code(i) {}
         Ex (const Ex& ex) : code(ex.code) {}
         int Get() const { return code; }
};
struct Ex51 : Ex { Ex51() : Ex(51) {} };
void f() { throw Ex51();
           printf("train\n");
        }
void t() { try { f(); }
          catch (Ex51 &x) { printf ("plane\n");
                           throw Ex (x.Get() + 1);
                           printf ("helicopter\n");
                           }
          catch (Ex &) { printf("car\n"); }
          printf("truck\n");
        }
int main() { try { t(); }
            catch (Ex &x) { printf("boat %d\n", x.Get()); }
            catch (...) { printf("rocket\n"); }
            return 0;
        }
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Описать класс D таким образом, чтобы все конструкции функции `main()` были верными, а на экран выдалось **30 10 20**.

```
int main () {
    D d1, d2 (1), d3(2);
    d1 = d2 + d3;
    cout << d1.get() << ' ' << d2.get() << ' ' << d3.get() << endl;
    return 0;
}
```

Изменять функцию `main()`, использовать механизм исключений, а также функции `exit()`, `terminate()` и т.п. для досрочного завершения программы **запрещается**.

2. Если в реализации функций `X::g()` и `main()` есть ошибки, **объясните**, в чем они заключаются. В теле функций `main()` и `B::g()` к именам **каждой** правильной конструкции **этих функций** добавьте префикс, указывающий с помощью операции `<::>`, из какой области видимости они выбираются.

```
void f(){ putchar ('f'); }
class Y {
public:    void f() {} };
class X : public Y { double a;
    X (int k = 0) { a = k; }
public:  X (double r ) { a = r; }
        void f(int x) { a = x; }
        void f(int x, int y = 5) { a = x - y; }
        void g() { f(); f(3); f(1, 2); } };
int main () { Y y; X a; X b(2.5);
                b.f(); return 0; }
```

3. Дан фрагмент программы. Есть ли в этом тексте ошибки? Если да, то в чём они заключаются?

```
struct MM { int i;
    virtual int f () const = 0;
    virtual ~MM () {} };
struct NN : public MM {
    int j;
    int g () const { return j; } };
NN nn1;
```

4. Есть ли ошибки в следующих заголовках шаблонов? Если есть, поясните, в чём они заключаются.

```
class myclass { public: myclass() {} };
template <int n> class A { /*...*/ };
template <class Cs> class C { /*...*/ };
template <myclass c> class B { /*...*/ };
```

5. Что напечатает следующая программа?

```
class A { int i;
public:    A (int x) { i = x; printf("sword\n"); }
        virtual ~A () { printf("lance\n"); }
        int f () const { return 49; }
        virtual int g () const { return i; }
        int h () const { return i - f() - g(); } };
class B : public A {
public:    B (int x): A(x+60) { printf("dagger\n"); }
        ~B () { printf("knife\n"); }
        int f () const { return 5; }
        virtual int g () const { return 2; }
        int h () const { return f() + g(); } };
int main() { B b(13); A * p = &b;
    printf ("result = (%d ; %d)\n", p->h(), b.h());
    return 0;
}
```

6. Опишите класс В таким образом, чтобы были верными все конструкции следующего фрагмента программы:

```
char * B::s = "ABC";
int main () { const B b;
               cout << B::s << b.h() << endl;
               return 0;
}
```

7. Выпишите профили (заголовки) всех методов (включая неявные), определённых для следующей структуры:

```
struct str1 { int a, b;
             str1(int m) : a(m), b(m) {}
};
```

8. Что напечатает следующая программа?

```
class I { int i;
public:
    I() : i(9) { printf("pedestrian\n"); }
    I(int a) : i(a) { printf("helicopter %d\n", i); }
    I(const I& other) : i(other.i) { printf("car %d\n", i); }
    ~I() { printf("rocket\n"); }
    int Get() { return i; }
    void operator *=(const I& op) { i*=op.i; }
};
void f(I x, I &y) { x *= 1; y *= x; }
int main() { I i1;
             I i2(2);
             i1 *= 3;
             f(i1, i2);
             printf("%d %d\n", i1.Get(), i2.Get());
             return 0;
}
```

9. ООП. Перечислите основные постулаты (механизмы) ООП. Каким образом реализация каждого постулата упрощает процесс создания сложного программного обеспечения?

10. Что напечатает следующая программа?

```
class Ex { int code;
public:
    Ex (int i) : code(i) {}
    Ex (const Ex& ex) : code(ex.code) {}
    int Get() const { return code; }
};
struct Ex90 : Ex { Ex90() : Ex(90) {} };
void f() { throw Ex90();
           printf("beta\n");
}
void t() { try { f(); }
          catch (Ex90 &x) { printf ("gamma\n");
                           throw Ex (x.Get() + 1);
                           printf ("delta\n");
                           }
          catch (Ex &) { printf("epsilon\n"); }
          printf ("omicron\n");
}
int main() { try { t(); }
            catch (Ex &x) { printf ("tau %d\n", x.Get()); }
            catch (...) { printf ("omega\n"); }
            return 0;
}
```

Коллоквиум, 2010 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
class A { int x;
public:   A   ( int y )           { x = y;           }
        int get ( )              { return x;         }
        int operator *= ( int y ) { return x = x*y; } };
```

Критерии: За каждый недостающий метод -4, за каждую ошибку реализации метода -2.

2. Ответ:

```
void B::g() { f ();           // ошибка, эта f не видна
            f (0);           // B::f(0);
            f (5.3 , 1);     // ошибка эта f не видна
            x = 1;          } // ошибка, int x private в базовом классе
int main () { B b; f (2);     // ошибка, эта f не видна
            f (3 , 'a'); //::f ( 3 , 'a');   return 0; }
```

Критерии: За каждую не найденную ошибку -2, за каждую лишнюю ошибку или неверно написанное с «::» имя (это строго взаимоисключающие ошибки) -2.

3. Ответ. При передаче переменной типа А в функцию g() по значению должен быть создан объект типа А, а это невозможно, поскольку структура содержит чистую виртуальную функцию.

Критерии: Верный ответ: 10, неверный ответ: 0.

4. Ответ. В качестве параметра шаблона можно использовать либо тип (class, typename), либо параметр целочисленного, перечислимого, указательного, ссылочного типа, либо типа «указатель на функцию-член». Тип double к таковым не относится, то есть третья строчка ошибочна. Больше ошибок нет.

Критерии: Не найдена ошибка: 0, найдена лишняя ошибка -5.

5. Ответ: first third result = (113 ; 2) fourth second

Критерий: За каждый пропущенный, лишний, неверный ответ -2.

6. Ответ:

```
class A { public: static int x; void get_0 () const { } };
```

Критерий: За каждую ошибку в классе -5.

7. Ответ: Необходимо убрать в приватную часть операцию присваивания от аргумента типа <<ссылка на Cls>>, например:

```
class Cls { public: Cls() {} private: void operator = (const Cls&) {} };
```

Критерии: Верный ответ: 10, неверный ответ: 0.

8. Ответ: sun venus 20 venus 400 moon earth 420 venus 1000
moon moon 1429 420 moon moon

Критерий: За каждый пропущенный, лишний, неверный ответ -2.

9. Ответ: Инкапсуляция – механизм, связывающий вместе код и данные, которыми он манипулирует, и одновременно защищающий их от произвольного доступа со стороны другого кода, внешнего по отношению к рассматриваемому. Доступ к коду и данным жестко контролируется интерфейсом. АТД – тип данных с полностью скрытой (инкапсулированной) структурой, а работа с переменными такого типа происходит только через специальные, предназначенные для этого функции. В С++ АТД реализуется с помощью классов (структур), в которых все члены-данные расположены в закрытой (private) области.

Критерии: За неверное понятие инкапсуляции -5, неверное понятие АТД -3, неверный ответ о реализации АТД в С++ -4. В целом до 0.

10. Ответ: cat elephant 91

Критерий: За каждый пропущенный, лишний, неверный ответ -5.

Коллоквиум, 2010 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
class B { int x;
public:   B ()           { x = 10;           }
         B (const B& y) { x = y.x + 10 }
         int get ()     { return x;       } };
```

Критерии: За каждый недостающий метод -4, за каждую ошибку реализации метода -2.

2. Ответ:

```
void D::h() {      f ('r');           // D::f
                 f ();               // ошибка эта f не видна
                 a = 2;             // B::a = 2
int main () { D d; f ();              // ошибка, неоднозначный выбор f
                 f (6);             // ::f
                 return 0;         }
```

Критерии: За каждую не найденную ошибку -2, за каждую лишнюю ошибку или неверно написанное с «::» имя (это строго взаимоисключающие ошибки) -2.

3. Ответ. Полем структуры A является объект типа S, а это невозможно, поскольку структура S содержит чистую виртуальную функцию.

Критерии: Верный ответ: 10, неверный ответ: 0.

4. Ответ. В качестве параметра шаблона можно использовать либо тип (class, typename), либо параметр целочисленного, перечислимого, указательного, ссылочного типа, либо типа «указатель на функцию-член». Тип float к таковым не относится, то есть вторая строчка ошибочна. Больше ошибок нет.

Критерии: Не найдена ошибка: 0, найдена лишняя ошибка -5.

5. Ответ: mercury earth result = (-62 ; 11) mars venus

Критерий: За каждый пропущенный, лишний, неверный ответ -2.

6. Ответ. class C { public: static const char c; static void f () { } };

Критерий: За каждую ошибку в классе -5.

7. Ответ: Необходимо убрать в приватную часть конструктор копирования, например: struct mystr { int a, b; private: mystr(const mystr& s) { } };

Критерии: Верный ответ: 10, неверный ответ: 0.

8. Ответ: owl sheep 3 sheep 7 wolf horse 42 sheep 1
 wolf wolf 42 126 wolf wolf

Критерий: За каждый пропущенный, лишний, неверный ответ -2.

9. Наследование - механизм, с помощью которого один объект (производного класса) приобретает свойства другого объекта (родительского, базового класса). Наследование позволяет какому-либо объекту наследовать от своего родителя общие атрибуты, а для себя определять только те характеристики, которые делают его уникальным внутри класса. В C++ наследование бывает открытым (public), закрытым (private) и защищенным (protected). Закрытые члены базового класса **видны**, но **недоступны** в производном классе, открытые члены и **видны**, и **доступны**.

Критерии: За неверное понятие наследования -5, за каждый недостающий способ наследования -2, неверный ответ о статусе закрытых и открытых членов -3, (если не упомянута видимость – не снижать). В целом до 0.

10. Ответ. Lance arche 61

Критерии: За каждый пропущенный, лишний, неверный ответ -5.

Коллоквиум, 2010 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
class C { int x;  
public: C (int y) { x = 2 * y; }  
      int get () { return x; }  
      C & operator + (C c) { C t (x + c.x); return t; } };
```

Критерии: За каждый недостающий метод -4, за каждую ошибку реализации метода -2.

2. Ответ:

```
void S::g() { f();           // ошибка эта f не видна  
            f(1);          // ошибка эта f не видна  
            f(-1,1);       // ошибка, неоднозначный выбор  
            y = 2;         // ошибка, так как int x private в базовом классе  
int main () { S b; f(5);    // ::f(5)  
            f('+',6);      // ошибка эта f не видна  
            return 0; };
```

Критерии: За каждую не найденную ошибку -2, за каждую лишнюю ошибку или неверно написанное с «::» имя (это строго взаимоисключающие ошибки) -2.

3. Ответ. При передаче переменной типа B в функцию g() по значению должен быть создан объект типа B, а это невозможно, поскольку структура содержит чистую виртуальную функцию.

Критерии: Верный ответ: 10, неверный ответ: 0.

4. Ответ. В качестве параметра шаблона можно использовать либо тип (class, typename), либо параметр целочисленного, перечислимого, указательного, ссылочного типа, либо типа «указатель на функцию-член». Тип struct к таковым не относится, то есть второй шаблон ошибочен. Больше ошибок нет.

Критерии: Не найдена ошибка: 0, найдена лишняя ошибка -5.

5. Ответ: dog sheep result = (33 ; 4) horse cat

Критерий: За каждый пропущенный, лишний, неверный ответ -2.

6. Ответ: class X { public: static void g(){} void h() const {} };

Критерий: За каждую ошибку в классе -5.

7. Ответ. Необходимо убрать в приватную часть конструктор копирования, например:

```
struct smartstr { int a, b;  
private: smartstr(int x) : a(x), b(x) {} };
```

Критерии: Верный ответ: 10, неверный ответ: 0.

**8. Ответ: fist lance 30 lance 700 pistole dagger 730
lance 1000 pistole pistole 1735 730 pistole pistole**

Критерий: За каждый пропущенный, лишний, неверный ответ -2.

9. Ответ. Полиморфизм – механизм, позволяющий использовать один и тот же интерфейс для общего класса действий. Различаются **статический** (на этапе компиляции, реализуется с помощью перегрузки функций/операций), **динамический** (во время выполнения программы, реализуется с помощью виртуальных функций) и **параметрический** (на этапе компиляции, с использованием механизма шаблонов) полиморфизм.

Критерии: За неверное понятие наследования -5, за каждый недостающий способ наследования -2, неверный ответ о статусе закрытых и открытых членов -3 (если не упомянута видимость - не снижать). В целом до 0.

10. Ответ. plane boat 52

Критерии: За каждый пропущенный, лишний, неверный ответ -5.

Коллоквиум, 2010 год. Ответы и критерии оценок.
Вариант 4.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
class D { int x;
public:    D ()      { x = 0; }
          D (int y) { x = y; }
          int get () { return x * 10; }
          D operator + (D d) { D t ( x + d.x ); return t; } };
```

Критерии: За каждый недостающий метод -4, за каждую ошибку реализации метода -2.

2. Ответ:

```
void X::g() {      f ();           // ошибка, эта f не видна
                  f (3);          // ошибка, неоднозначный выбор f
                  f (1 , 2 ); } // X::f (1, 2);
int main () { Y y; X a ;           // ошибка, обращение
                  X b (2.5);       // к закрытому конструктору класса
                  b.f ();          // ошибка, эта f не видна в X
                  return 0; }
```

Критерии: За каждую не найденную ошибку -2, за каждую лишнюю ошибку или неверно написанное с «::» имя (это строго взаимоисключающие ошибки) -2.

3. Ответ. Объект типа NN не может быть создан, поскольку функция f () так и осталась в нем чистой виртуальной функцией.

Критерии: Верный ответ: 10, неверный ответ: 0.

4. Ответ. В качестве параметра шаблона можно использовать либо тип (class, typename), либо параметр целочисленного, перечислимого, указательного, ссылочного типа, либо типа «указатель на функцию-член». Тип myclass к таковым не относится, то есть третья строчка ошибочна. Больше ошибок нет.

Критерии: Не найдена ошибка: 0, найдена лишняя ошибка -5.

5. Ответ: sword dagger result = (22 ; 7) knife lance

Критерий: За каждый пропущенный, лишний, неверный ответ -2.

6. Ответ: class B { public: static char * s; int h() const { return 1; } };

Критерии: За каждую ошибку в классе -5.

7. Ответ. str1::str1 (int m); str1::str1 (const str1& s);
 str1::~str1 (); str1::operator=(const str1& s);

Критерии: За каждый пропущенный, лишний, неверный ответ -3, за отсутствие метода operator=() или указание других операций, определенных для объектов классов (&, ...), не снижать.

**8. Ответ: pedestrian helicopter 2 helicopter 3 rocket car 27
 helicopter 1 rocket rocket 27 54 rocket rocket**

Критерии: За каждый пропущенный, лишний, неверный ответ -2.

9. Постулаты ООП – инкапсуляция, наследование и полиморфизм. Механизм **инкапсуляции** позволяет некоторые детали реализации класса оставлять скрытыми для пользователя (инкапсулировать их в классе), что упрощает работу с объектами класса. Механизм **наследования** позволяет описывать новый класс не обязательно, начиная с нуля, а использовать в создаваемом (производном) классе готовые методы и данные базового класса, что существенно упрощает работу программиста. **Полиморфизм** позволяет программисту помнить и применять один интерфейс для однотипных действий, вместо нескольких, что упрощает работу.

Критерии: За каждый верный ответ +3. Все верные ответы +10.

10. Ответ: gamma tau 91

Критерии: За каждый пропущенный, лишний, неверный ответ -5.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow AB$
 $A \rightarrow a \mid \varepsilon$
 $B \rightarrow CbD \mid C$
 $C \rightarrow E \mid EE$
 $D \rightarrow C \mid \varepsilon$
 $EE \rightarrow EEE$
 $E \rightarrow c$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

(б) Каким из перечисленных классов принадлежит язык $L(G)$?

Класс $\mathfrak{S}$	$L(G) \in \mathfrak{S}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
языки типа 0	
регулярные языки	

(в) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$.

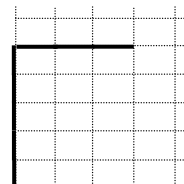
2. Дать определение эквивалентных грамматик. Приведите примеры эквивалентных и почти эквивалентных грамматик. Могут ли неэквивалентные грамматики порождать один и тот же язык?

3. Построить по заданной грамматике G конечный автомат (в виде ДС). Если автомат оказался детерминированным, построить анализатор на Си, иначе – преобразовать автомат в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА, и по полученному ДКА построить левостолбчатую грамматику.

$G:$ $H \rightarrow bA \mid bB \mid aC$
 $A \rightarrow bC$
 $B \rightarrow aC$
 $C \rightarrow aA \mid bB \mid \perp$

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{c, a, e\}$: c – переместить перо на одну клетку вверх; a – переместить перо на одну клетку вправо; e – переместить перо на одну клетку вниз. Например, по последовательности $sssssa$ интерпретатор построит букву «Г» размера 5×3 , изображенную на рисунке.

Построить грамматику с терминальным алфавитом $\{c, a\}$, описывающую буквы «Г» любого размера $n \times m$, $n, m \geq 1$. В грамматике должно быть не более 4 правил вывода (считая альтернативы). В построенную грамматику добавить действия вида $\langle \text{cout} << \text{'символ'}; \rangle$ так, чтобы описание буквы «Г» размера $n \times m$ переводилось в описание буквы «П» того же размера.



5. Восстановить грамматику по заданному анализатору на C++. Сформулировать достаточное условие применимости метода рекурсивного спуска для грамматик без эpsilon-правил. Соответствует ли полученная грамматика данному условию? Обосновать свой ответ. Корректен ли данный анализатор? Обосновать свой ответ (доказать корректность или найти примеры некорректного поведения анализатора).

```
void S () { if (c == 'b') { c = GetS (); A ();
                if (c == 'a') { c = GetS (); C (); D (); }
                else Error ();
                if (c == '\1') return;
            }
            Error ();
        }
void A () { if (c == 'a') { c = GetS (); A (); } else B (); }
void B () { if (c == 'b') { c = GetS (); B (); } else C (); }
void C () { if (c == 'a') { c = GetS (); C (); } else D (); }
void D () { if (c == 'b') { c = GetS (); D (); } else A (); }
```

6. Дать определение системы программирования. Перечислить основные компоненты систем программирования и их основное назначение.

7. Перечислить не менее 5 различных видов внутреннего представления программ, используемых при компиляции.

8. С какими компонентами интегрированных систем программирования взаимодействуют входящие в них символьные отладчики? Какую помощь и для чего оказывают отладчикам эти компоненты?

9. Дать определение приведённых грамматик. Преобразовать контекстно-свободную грамматику G к приведённому виду, выписывая результаты применения каждого шага используемого алгоритма.

$$G: \begin{aligned} S &\rightarrow ACb \mid Aa \\ A &\rightarrow Aa \mid Ca \mid a \\ B &\rightarrow Aa \mid CDb \mid Bb \\ C &\rightarrow Ab \mid b \\ D &\rightarrow Aa \mid Cb \mid a \end{aligned}$$

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов. Постфиксная операция '@' соответствует унарной операции изменения знака.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ЛИЗ	&x	#+	4	+	y	5	—	20	*	<=	30	!F	&a	x	8
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	—	7	*	y	x	@	2	+	/	—	=	;	1	!	&b
	31	32	33	34	35	36									
	&x	&y	+#	=	=	;	}								

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow aA \mid A$
 $A \rightarrow BbB \mid Bb \mid B$
 $B \rightarrow C \mid CC$
 $CC \rightarrow CCC$
 $C \rightarrow c$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

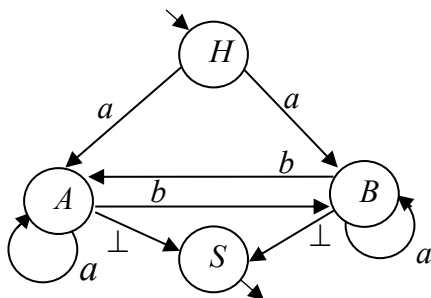
(б) Каким из перечисленных классов грамматик принадлежит G ?

Класс Π	$G \in \Pi?$ (да/нет)
регулярные	
контекстно-зависимые	
контекстно-свободные	
грамматики типа 0	
неукорачивающие	

(в) Тип языка: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком типа $k+1$.

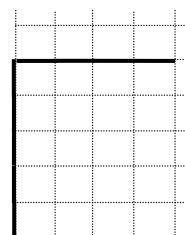
2. Дать определение порождающей грамматики. Какие ограничения на использование терминальных и нетерминальных символов в правилах грамматики наложены этим определением? Предложите какую-либо грамматику для языка $L = \{a^n b^n \mid n \geq 1\}$ и укажите на этом примере все компоненты, упоминаемые в определении грамматики.

3. Дана диаграмма состояний конечного автомата. Построить по этой диаграмме соответствующую левостолбчатую грамматику. Если заданный автомат детерминирован, построить анализатор на Си, иначе – преобразовать автомат в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА, и по полученному ДКА построить правостолбчатую грамматику.



4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{c, a, e\}$: c – переместить перо на одну клетку вверх; a – переместить перо на одну клетку вправо; e – переместить перо на одну клетку вниз. Например, по последовательности $ccccaaaa$ интерпретатор построит букву «Г» размера 5×4 , изображенную на рисунке.

Построить грамматику с терминальным алфавитом $\{c, a\}$, описывающую буквы «Г» любого размера $n \times m$, $n, m \geq 1$. В грамматике должно быть **не более 4** правил вывода (считая альтернативы). В построенную грамматику добавить действия вида $\langle cout \ll 'символ'; \rangle$ так, чтобы описание буквы «Г» размера $n \times m$ переводилось в описание буквы «П» размера $n \times (m+1)$.



5. Восстановить грамматику по заданному анализатору на C++. Сформулировать достаточное условие применимости метода рекурсивного спуска для грамматик с эpsilon-правилами. Соответствует ли полученная грамматика данному условию? Обосновать свой ответ. Корректен ли данный анализатор? Обосновать свой ответ (доказать корректность или найти примеры некорректного поведения анализатора).

```
void S () {      if (c == '1') { c = GetL (); A();
                                if (c != '0') Error ();
                                c = GetL (); C(); D();
                                if (c != '1') Error ();
                                c = GetL (); }
                                else Error ();
void A () {      if (c == '0') { c = GetL (); A(); }
                                else if (c == '1') { c = GetL (); B(); }
                                else Error ();
void B () {      if (c == '0') { c = GetL (); C(); }
                                else if (c == '1') { c = GetL (); B(); }
                                else Error ();
void C () {      if (c == '0') { c = GetL (); C(); }
                                else if (c == '1') { c = GetL (); D(); }
                                else Error ();
void D () {      if (c == '1') { c = GetL (); D(); } }
```

6. Описать основные задачи ассемблеров, компиляторов и интерпретаторов. В чём сходство и различие этих компонентов систем программирования?

7. Нарисовать схему однопроходного компилятора. Каким образом в таких компиляторах взаимодействуют лексический и синтаксический анализаторы? Какой из анализаторов является ведущим в таком взаимодействии?

8. Описать основные подходы, использующие поведенческие и структурные тесты программ. В чем их основные различия?

9. Дать определения бесплодных и недостижимых символов алфавита контекстно-свободной грамматики. Преобразовать контекстно-свободную грамматику G к приведённому виду, выпишывая результаты применения каждого шага используемого алгоритма.

G :
 $S \rightarrow Ab \mid Ba$
 $A \rightarrow Ca$
 $B \rightarrow Bb \mid Da$
 $C \rightarrow Aa \mid Bb \mid a \mid b$
 $D \rightarrow Db \mid Ba$

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов. Постфиксная операция '@' соответствует унарной операции изменения знака.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ЛИЗ	&a	x	@	11	*	y	x	@	/	—	+=	;	&x	#+	5
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
&y	+#	—	x	+	1	—	<=	28	!F	1	!	&y	&x	y	
31	32	33	34	35	36	37									
8	+	=	7	+	=	;	}								

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$$\begin{aligned} S &\rightarrow aA \mid A \\ A &\rightarrow Bb \mid B \mid Ac \\ B &\rightarrow c \mid Bc \end{aligned}$$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

(б) Каким из перечисленных классов принадлежит язык $L(G)$?

Класс $\mathfrak{L}$	$L(G) \in \mathfrak{L}$? (да/нет)
контекстно-свободные	
контекстно-зависимые	
регулярные	
языки типа 0	

(в) Каким из перечисленных классов грамматик принадлежит G ?

Класс $\mathfrak{P}$	$G \in \mathfrak{P}$? (да/нет)
контекстно-свободные	
контекстно-зависимые	
регулярные	
грамматики типа 0	
неукорачивающие	

2. Описать иерархию классов языков по Хомскому. Привести примеры трех языков, относящихся к различным уровням иерархии.

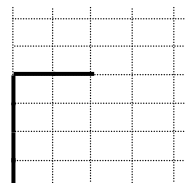
3. Построить по заданной грамматике G конечный автомат (в виде ДС). Если автомат оказался детерминированным, построить анализатор на Си, иначе – преобразовать автомат в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА и по полученному ДКА построить праволинейную грамматику.

$$\begin{aligned} G: \quad S &\rightarrow A\perp \mid B\perp \\ A &\rightarrow Aa \mid Ab \mid Ba \mid Bb \mid b \\ B &\rightarrow Aa \mid Ab \mid Ba \mid Bb \mid a \end{aligned}$$

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{c, a, e\}$: c – переместить перо на одну клетку вверх; a – переместить перо на одну клетку вправо; e – переместить перо на одну клетку вниз. Например, по последовательности $ссссаа$ интерпретатор построит букву «Г» размера 4×2 , изображенную на рисунке.

Построить грамматику с терминальным алфавитом $\{c, a\}$, описывающую буквы «Г» любого размера $n \times m$, $n, m \geq 1$. В грамматике должно быть **не более 4** правил вывода (считая альтернативы).

В построенную грамматику добавить действия вида $\langle \text{cout} \ll \text{'символ'}; \rangle$ так, чтобы описание буквы «Г» размера $n \times m$ переводилось в описание буквы «П» размера $(n+1) \times m$.



5. Восстановить грамматику по заданному анализатору на C++. Сформулировать критерий применимости метода рекурсивного спуска для контекстно-свободных грамматик. Соответствует ли полученная грамматика этому критерию? Обосновать свой ответ. Корректен ли данный анализатор? Обосновать свой ответ (доказать корректность или найти примеры некорректного поведения анализатора).

```
void S () { if (c != 'b') Error ();
           c = GetL (); A (); B (); C (); D ();
           if (c != 'b') Error ();
           c = GetL (); A (); D ();
         }

void A () { C (); E (); }
void B () { if (c == 'b') { c = GetL (); C (); } else D (); }
void C () { if (c == 'a') { c = GetL (); D (); }
void D () { if (c == 'b') { c = GetL (); E (); } else A (); }
void E () { if (c == 'a') { c = GetL (); B (); }
```

6. Описать основные особенности программ, программных продуктов и интегрированных программных продуктов.

7. Перечислить основные виды оптимизации циклов. Привести пример оптимизационного преобразования цикла.

8. Описать деление библиотек систем программирования по функциональному признаку, дать краткие характеристики содержания различных видов библиотек. Признаки каких из перечисленных Вами видов библиотек можно наблюдать в стандартной библиотеке языка C++?

9. Дать определение эквивалентных грамматик. Для контекстно-свободной грамматики G построить эквивалентную ей неукорачивающую грамматику, выписывая результаты применения каждого шага используемого алгоритма.

$$G: \begin{aligned} S &\rightarrow Ab \mid BaCaC \\ A &\rightarrow Ca \\ B &\rightarrow Cb \\ C &\rightarrow Aa \mid Bb \mid a \mid \varepsilon \end{aligned}$$

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов. Постфиксная операция '@' соответствует унарной операции изменения знака.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ЛИЗ	&t	&p	#+	&q	+#	t	@	/	4	*	+	=	;	t	u
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	<	34	!F	26	!	&t	+#	;	14	!	&r	p	7	+	==
	31	32	33	34	35	36	37	38	39						
	;	21	!	&r	r	2	%	=	;	}					

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$$\begin{aligned} S &\rightarrow A \mid Ab \mid Sc \\ A &\rightarrow Bc \mid Ac \\ B &\rightarrow a \mid \varepsilon \end{aligned}$$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

(б) Каким из перечисленных классов принадлежит грамматика G ?

Класс П	$G \in \Pi?$ (да/нет)
контекстно-свободные	
контекстно-зависимые	
неукорачивающие	
грамматики типа 0	
регулярные	

(в) Классификация: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком типа $k+1$.

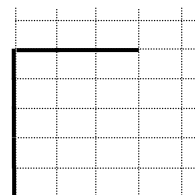
2. Дать определение неоднозначных контекстно-свободных грамматик и языков. Покажите, что язык $L = \{a^n \mid n \leq 2\}$ однозначен. Можно ли описать этот язык неоднозначной грамматикой? Обоснуйте ответ.

3. Дана функция переходов конечного автомата. Построить по этой функции соответствующую праволинейную грамматику. Если заданный автомат детерминирован, построить анализатор на Си, иначе – преобразовать его в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА, и по полученному ДКА построить левوليнейную грамматику.

$$\begin{aligned} \delta(H, a) &= A & \delta(A, a) &= A & \delta(B, b) &= A \\ \delta(H, a) &= B & \delta(A, a) &= B & \delta(B, \perp) &= S \\ \delta(H, b) &= B & \delta(A, \perp) &= S & & \\ \delta(H, \perp) &= S & & & & \end{aligned}$$

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{c, a, e\}$: c – переместить перо на одну клетку вверх; a – переместить перо на одну клетку вправо; e – переместить перо на одну клетку вниз. Например, по последовательности $ssssaaaa$ интерпретатор построит букву «Г» размера 5×3 , изображенную на рисунке.

Построить грамматику с терминальным алфавитом $\{c, a\}$, описывающую буквы «Г» любого размера $n \times m$, $n, m \geq 1$. В грамматике должно быть не более 4 правил вывода (считая альтернативы). В построенную грамматику добавить действия вида $\langle cout \ll \text{'символ'}; \rangle$ так, чтобы описание буквы «Г» размера $n \times m$ переводилось в описание буквы «П» удвоенного размера.



5. Восстановить грамматику по заданному анализатору на C++. Соответствует ли полученная грамматика каноническому виду? Обосновать свой ответ. Корректен ли данный анализатор? Обосновать свой ответ (доказать корректность или найти примеры некорректного поведения анализатора).

```
void S () { if (c == 'q') { c = GetL (); A ();
                    if (c != 'p') Error ();
                    c = GetL (); C ();
                    if (c != 'q') Error ();
                    c = GetL ();
                }
            else Error ();
        }
void A () { if (c == 'p') { c = GetL (); A (); } else B (); }
void B () { if (c == 'q') { c = GetL (); B (); } else C (); }
void C () { if (c == 'q') { c = GetL (); C (); } }
```

6. Описать основные схемы процесса разработки программного продукта, указав их главные особенности.

7. Какие области памяти выделяются компиляторами для хранения локальных данных процедур? Какая дисциплина распределения памяти при этом реализуется? Какой способ использования выбирается для этих областей?

8. Описать основное назначение редакторов связей

9. Выполнить преобразование грамматики G к каноническому виду для метода рекурсивного спуска, выписывая результаты применения каждого последовательного шага преобразования.

G : $S \rightarrow aAb \mid aBc$
 $A \rightarrow Aa$
 $B \rightarrow bC$
 $C \rightarrow aA \mid b$

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ЛИЗ	n	0	<	38	!F	x	y	+	z	—	4	>	23	!F	&x
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
n	2	+	=	;	38	!	z	0	>	34	!F	&x	2	=	
31	32	33	34	35	36	37	38								
;	38	!	&y	z	=	;									

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow A | Ab | Sc$
 $A \rightarrow Bc | Ac$
 $B \rightarrow a | \varepsilon$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

(б) Каким из перечисленных классов принадлежит грамматика G ?

Класс	$G \in \Pi?$ (да/нет)
контекстно-свободные	
контекстно-зависимые	
неукорачивающие	
грамматики типа 0	
регулярные	

(в) Классификация: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком типа $k+1$.

2. Дать определение чистой виртуальной функции. В каких случаях целесообразно их использование? Какие ограничения на объекты соответствующих классов налагаются при программировании на языке Си++? Можно ли из виртуальной функции вызывать другие виртуальные функции?

3. Построить по заданной грамматике G конечный автомат (в виде ДС). Если автомат оказался детерминированным, построить анализатор на Си, иначе – преобразовать его в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА, и по полученному ДКА построить левостороннюю грамматику.

G :
 $S \rightarrow Ab | Ba$
 $A \rightarrow Ca$
 $B \rightarrow Cb$
 $C \rightarrow Aa | Bb | a | b$

4. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
class A;
void g (A& x, int n);
class A
{ public: A()      {try { g (* this, 0);          cout<<'Z'<<endl;      }
                  catch (A&) {                  cout<<'X'<<endl;      }
                  catch (int n) { g(* this, -n); cout<<'T'<<endl;      } }
      A(A&) {      cout<<'B'<<endl;      }
      ~A() {      cout<<'A'<<endl;      } };
void g(A&x, int n) {try { if (n <= 0) throw x;
                        if (n >= 0) throw n-1; cout<<'N'<<endl;      }
                  catch (int) {                  cout<<'P'<<endl; throw; }
                  catch (A&) {                  cout<<'O'<<endl; throw; } }
void main()
{ try { A a; }
  catch (...) {
                                cout<<'V'<<endl;      }
                                cout<<'C'<<endl;      }
```

5. Восстановить грамматику по заданному анализатору на C++. Соответствует ли полученная грамматика каноническому виду? Обосновать свой ответ. Корректен ли данный анализатор? Обосновать свой ответ (доказать корректность или найти примеры некорректного поведения анализатора).

```
void S () { if (c == 'q') { c = GetL (); A ();
                if (c != 'p') Error ();
                c = GetL (); C ();
                if (c != 'q') Error ();
                c = GetL ();
            }
        else Error ();
    }
void A () { if (c == 'p') { c = GetL (); A(); } else B(); }
void B () { if (c == 'q') { c = GetL (); B(); } else C(); }
void C () { if (c == 'q') { c = GetL (); C(); } }
```

6. Какие конструкторы и деструкторы и в каком порядке будут вызываться при работе следующего фрагмента программы

```
class X { int x; };
class Y: public X { int y;
    public: Y sum (Y & p1, Y p2) { return p1.y + p2.y; }
           Y () { y = 0; }
           Y (int p) { y = p; }
};
void main () { Y a; a = a.sum (a, a); }
```

7. Какие виды пользовательских преобразований проводятся при выборе перегруженной функции в программах на языке C++? Привести пример каждого из разрешенных преобразований.

8. Описать основное назначение редакторов связей

9. Написать функцию $g()$ с параметром, представляющим собой контейнер-вектор элементов целого типа. Функция должна, менять местами значения элементов вектора, одинаково удалённых от начала и конца вектора (первого с последним, второго с предпоследним и т. д.). Функция возвращает число сделанных перестановок.

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ЛИЗ	n	0	<	38	!F	x	y	+	z	—	4	>	23	!F	&x
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
	2	+	=	;	38	!	z	0	>	34	!F	&x	2	=	
31	32	33	34	35	36	37	38								
	;	38	!	&y	z	=	;								>

Экзамен, 2010 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{ a^i c^n b^j c^m \mid i, j \in \{0, 1\}, n \geq 1, m \geq 0 \}$

(б) КС – "da" (-1), КЗ – "da" (-1), тип 0 – "da" (-1), регулярный – "da" (-3).

(в) **Ответ:** $k = 0$ (есть укорачивающее правило $A \rightarrow \varepsilon$ и не КС-правило $EE \rightarrow EEE$)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий в ответе (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: (а) Грамматики G_1 и G_2 называются эквивалентными, если их языки совпадают, $L(G_1) = L(G_2)$. (б) Например, из трёх разных грамматик:

$G_1 = (\{0, 1\}, \{A, S\}, P_1, S)$ и $G_2 = (\{0, 1\}, \{S\}, \{S \rightarrow 0S1 \mid 01\}, S)$

$P_1: S \rightarrow 0A1$

$G_3 = (\{0, 1\}, \{S\}, \{S \rightarrow 0S1 \mid \varepsilon\}, S)$

$0A \rightarrow 00A1$

G_1 и G_2 эквивалентны, $L(G_1) = L(G_2) = \{0^n 1^n \mid n > 0\}$.

$A \rightarrow \varepsilon$

G_1 (G_2) и G_3 почти эквивалентны, $L(G_3) = \{0^n 1^n \mid n \geq 0\}$

(в) **Нет**, это невозможно по приведённому определению.

Критерии: (а) нет ответа; определение с ошибками: -5, (б) за каждый из двух отсутствующих примеров: -3, (в) нет ответа; неверный ответ: -5.

3. Ответ: Автомат недетерминирован G' :

$S \rightarrow C \perp$

$A \rightarrow Ca$

$B \rightarrow Cb$

$C \rightarrow Ab \mid Ba \mid Da \mid Db \mid a$

$D \rightarrow b$

Функция переходов исходного автомата

	$\delta(H, a) = C$ $\delta(H, b) = A$ $\delta(H, b) = B$	$\delta(A, b) = C$ $\delta(B, a) = C$	$\delta(C, a) = A$ $\delta(C, b) = B$ $\delta(C, \perp) = S$
	Функция переходов детерминированного автомата ($D \equiv AB$)		
	$\delta'(H, a) = C$ $\delta'(H, b) = D$ $\delta'(A, b) = C$	$\delta'(C, a) = A$ $\delta'(C, b) = B$ $\delta'(C, \perp) = S$	$\delta'(B, a) = C$ $\delta'(D, a) = C$ $\delta'(D, b) = C$

Критерии: (а) построен неэквивалентный или недетерминированный автомат: -5, (б) левостроенная грамматика построена, но неэквивалентная или не соответствующая новой диаграмме: -5, (в) нет ответа или приведена программа анализатора: 0.

4. Ответ: Описание буквы «Г» размера $n \times m$ — это цепочка $c^n a^m$. Следовательно, требуется построить грамматику, порождающую язык $\{c^n a^m \mid n, m \geq 1\}$.

$S \rightarrow cS \mid cA$

$A \rightarrow aA \mid a$

Соглашение запись $\langle a \rangle$ является сокращением $\langle cout \ll "a"; \rangle$

Грамматика с действиями: $S \rightarrow c\langle c \rangle S\langle e \rangle \mid c\langle c \rangle A\langle e \rangle$

$A \rightarrow a\langle a \rangle A \mid a\langle a \rangle$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -3.

5. Ответ: Восстановленная грамматика имеет вид: $G = (\{a, b, \perp\}, \{A, B, C, D, S\}, P, S)$

$$\begin{array}{ll} P: & S \rightarrow bAaCD\perp \\ & A \rightarrow aA \mid B \\ & C \rightarrow aC \mid D \\ & B \rightarrow bB \mid C \\ & D \rightarrow bD \mid A \end{array}$$

Анализатор некорректен: он закичивается, например, на цепочке "b⊥". Достаточное условие применимости метода рекурсивного спуска для грамматик без эpsilon-правил:

(1) либо $A \rightarrow \alpha$, где $\alpha \in (T \cup N)^*$ и это единственное правило вывода для A ;

(2) либо $A \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n$, где $a_i \in T$ для $i = 1, 2, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$.

Грамматика G достаточным условиям не соответствует: не все правила для нетерминальных символов A, B, C и D начинаются с терминальных символов.

Критерии: Неправильно восстановлена грамматика: -5, Неверная формулировка достаточного условия применимости: -4, Неверный ответ на вопрос о соответствии грамматики сформулированному условию: -2, Нет обоснования соответствия/несоответствия грамматики: -1, Неверный ответ на вопрос о корректности анализатора: -2, Нет обоснования корректности (не найден пример "плохой" цепочки): -1.

6. Ответ: Системой программирования называется комплекс программных средств, предназначенных для поддержки программного продукта на протяжении всего жизненного цикла этого продукта. Основные компоненты современных развитых систем программирования: средства интеграции компонентов системы программирования (-1), редакторы текстов (-2), трансляторы: интерпретаторы, компиляторы и ассемблеры, вместе с макрогенераторами (-2), библиотеки (-2), редакторы связей (-2), средства конфигурирования (-1), отладчики (-2), средства тестирования (-1), профилировщики (-1), справочные системы (-1)

Критерии: За пропуск компонентов снижать на 1-2 балла.

7. Ответ: (1) Связные списочные структуры, представляющие синтаксическое дерево, (2) Многоадресный код с явно именуемыми результатами (тетрады), (3) Многоадресный код с неявно именуемыми результатами (триады), (4) Инфиксная запись, (5) Префиксная запись, (6) Постфиксная запись, (7) Язык ассемблера.

Критерии: За каждый пропущенный из необходимых 5 видов: -2.

8. Ответ: С текстовыми редакторами, компиляторами и редакторами связей. Текстовые редакторы обеспечивают доступ к текстовой структуре программе (для расстановки точек останова и пошагового исполнения программы). Компиляторы и редакторы связей обеспечивают доступ к информационным таблицам имен, адресов, к описаниям областей памяти (для выдачи информации в терминах входной программы).

Критерии: За пропуск какого-либо компонента: -3 (за каждый пропуск), за пропуск объяснения какого-либо взаимодействия компонентов: -2 (за каждый).

9. Ответ: Контекстно-свободная грамматика называется приведённой, если в ней нет бесполезных (бесплодных и недостижимых) символов. (1) Удаление бесплодных символов. Строится множество N , из элементов которого за некоторое число шагов выводятся терминальные символы. Бесплодных символов в грамматике G не обнаружено. (2) Удаление недостижимых символов. Строится множество V , в которое помещаются символы, за некоторое число шагов выводимые из символа S . Символы B и D из алфавита грамматики G недостижимы, они и правила их содержащие удаляются из грамматики:

$$G_{\text{прив.}}: \quad S \rightarrow ACb \mid Aa \quad A \rightarrow Aa \mid Ca \mid a \quad C \rightarrow Ab \mid b$$

Критерии: Нет определения или оно неверно: -5, за ошибку при преобразовании: -7.

10. Ответ.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ЛИЗ	&x	#+	4	+	y	5	—	20	*	<=	30	!F	&a	x	8
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	
	—	7	*	y	x	@	2	+	/	—	=	;	!	1	
	30	31	32	33	34	35	36								
	&b	&x	&y	+#	=	=	;	}							

Ответ:

```

М:  if (! (x ++ + 4 <= (y - 5) * 20)) goto L;
    a = (x - 8) * 7 - y / (- x + 2);
    goto M;
L:  b = (x = ++ y);

        while (x ++ + 4 <= (y - 5) * 20)
            a = (x - 8) * 7 - y / (- x + 2);
        b = (x = ++ y);

```

Критерии: за неверный ответ на первый вопрос: **-7**, за неверный ответ на второй вопрос, если первый вариант верен: **-3**, иначе: **-10**.

Экзамен, 2010 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{ a^i c^n b^j c^m \mid i, j \in \{0, 1\}, n \geq 1, m \geq 0 \}$, (б) регулярные – "нет" (-3), КЗ – "да" (-1), КС – "нет" (-3), тип 0 – "да" (-1), неукорачивающие – "да" (-1), (в) $k = 3$ (для заданного языка существует порождающая его регулярная грамматика).

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий в ответе (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: (а) Порождающая грамматика G – это четверка (T, N, P, S) , где T – алфавит терминальных символов (терминалов), N – алфавит нетерминальных символов (нетерминалов), множества N и T не пересекаются друг с другом, то есть $N \cap T = \emptyset$, P – конечное множество правил – подмножество множества $(T \cup N)^+ \times (T \cup N)^*$; S – начальный символ (цель) грамматики, $S \in N$. (б) Этим определением требуется, чтобы в левой части любого правила грамматики присутствовал хотя бы один нетерминальный символ. Других формальных ограничений нет. (в) Например, в грамматике

$G = (\{a, b\}, \{A, S\}, \{S \rightarrow AS \mid A; A \rightarrow a \mid b\}, S)$:

Множество терминальных символов $T = \{a, b\}$, Множество нетерминальных символов $N = \{A, S\}$

Множество правил $P = \{S \rightarrow AS \mid A; A \rightarrow a \mid b\}$, Начальный символ (цель) грамматики S .

Критерии: (а) нет ответа; определение с ошибками: -5, (б) нет ответа; неверный ответ: -3, (в) нет примера грамматики или неправильный пример: -4, за каждый неверный или отсутствующий компонент определения: -1 (до -4).

3. Ответ: Гл: $S \rightarrow A \perp \mid B \perp$

$A \rightarrow Aa \mid Bb \mid a$

$B \rightarrow Ab \mid Ba \mid a$

Автомат недетерминирован

Функция переходов исходного автомата

$\delta(H, a) = A$ $\delta(H, a) = B$	$\delta(A, a) = A$ $\delta(A, b) = B$ $\delta(A, \perp) = S$	$\delta(B, a) = B$ $\delta(B, b) = A$ $\delta(B, \perp) = S$
Функция переходов детерминированного автомата ($C \equiv AB$)		
$\delta'(H, a) = C$	$\delta'(C, a) = C$	$\delta'(C, b) = C$
		$\delta'(C, \perp) = S$

Глев: $S \rightarrow C \perp$

$C \rightarrow Ca \mid Cb \mid a$

Гправ: $H \rightarrow aC$

$C \rightarrow aC \mid bC \mid \perp$

Критерии: (а) нет или неправильная левостроенная грамматика исходного автомата: -5, (б) построен неэквивалентный или недетерминированный автомат: -5, (в) правостроенная грамматика построена, но неэквивалентная или не соответствующая новой диаграмме: -5, (г) нет ответа или приведена программа анализатора: 0.

4. Ответ: Описание буквы «Г» размера $n \times m$ — это цепочка $c^n a^m$. Следовательно, требуется построить грамматику, порождающую язык $\{c^n a^m \mid n, m \geq 1\}$:

$S \rightarrow cS \mid cA$

$A \rightarrow aA \mid a$

Соглашение: запись $\langle a \rangle$ является сокращением $\langle cout \ll "a"; \rangle$

Грамматика с действиями: $S \rightarrow c\langle c \rangle S\langle e \rangle \mid c\langle c \rangle A\langle e \rangle$ $A \rightarrow a\langle a \rangle A \mid a\langle a \rangle \langle a \rangle$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -3.

5. Ответ: Восстановленная грамматика имеет вид: $G = (\{0, 1\}, \{A, B, C, D, S\}, P, S)$

$P: S \rightarrow 1A0CD1 \quad A \rightarrow 0A \mid 1B \quad B \rightarrow 0C \mid 1B \quad C \rightarrow 0C \mid 1D \quad D \rightarrow 1D \mid \varepsilon$

Анализатор некорректен: любая правильная цепочка должна заканчиваться символом '1', а этот символ может быть зря прочитан процедурой "D()", что приведёт к неправильной трактовке всех правильных цепочек, например, анализатор выдаёт ошибку на правильной цепочке "10101011". Достаточное условие применимости метода рекурсивного спуска для грамматик с эpsilon-правилами:

- a) либо $X \rightarrow \alpha$, где $\alpha \in (T \cup N)^*$ и это единственное правило вывода для X;
- b) либо $X \rightarrow a_1\alpha_1 \dots a_n\alpha_n$, где $a_i \in T$ для $i = 1, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$;
- c) либо $X \rightarrow a_1\alpha_1 \dots a_n\alpha_n \varepsilon$, где $a_i \in T$ для $i = 1, \dots, n$; $a_i \neq a_j$ для $i \neq j$; $\alpha_i \in (T \cup N)^*$, и $first(X) \cap follow(X) = \emptyset$.

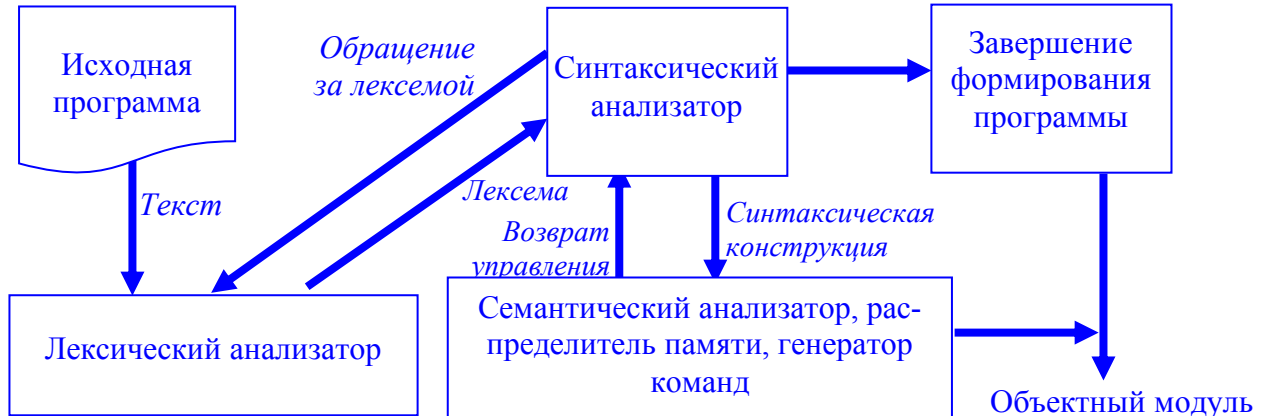
Грамматика G показанным условиям не соответствует: $first(D) \cap follow(D) = \{1\} \neq \emptyset$.

Критерии: Неправильно восстановлена грамматика: **-5**, Неверная формулировка достаточного условия применимости: **-5**, Неверный ответ на вопрос о соответствии грамматики сформулированному условию: **-3**, Нет обоснования соответствия/несоответствия грамматики: **-2**, Неверный ответ на вопрос о корректности анализатора: **-3**, Нет обоснования корректности (нет рассуждений о конечных символах цепочек, не найден пример неправильно трактуемой цепочки и т. д.): **-2**.

6. Ответ: Ассемблер – для трансляции программ, написанных на языках типа 1:1, компилятор – для трансляции программ, написанных на языках высокого уровня, интерпретатор – не для трансляции программ, а для получения результата с помощью алгоритма программы.

Критерии: За отсутствие комментариев к какому-либо виду трансляторов: **-4**.

7. Ответ:



Лексический и синтаксический анализаторы взаимодействуют как сопрограммы: ЛА формирует лексемы по исходной программе по запросам СА. Работа ведется под управлением СА.

Критерии: (а) За отсутствие схемы или ошибки в ней: **-5**, (б) За отсутствие описания способа взаимодействия: **-3**, (в) Нет ответа или ошибка: **-3**.

8. Ответ: Поведенческое тестирование основано на технических требованиях, спецификациях. Структурное тестирование проверяет правильность исполнения каждого структурного фрагмента программы (ветви условного оператора, использование каждого элемента данных и т. д.). Для проведения поведенческого тестирования не обязательно знать внутреннюю структуру исследуемой программы, структурные тесты требуют полного доступа к структуре программы.

Критерии: За отсутствие описания какого-либо из двух видов: **-5** (за каждый пропуск). За пропуск объяснения различия подходов: **-5**.

9. Ответ: Символ A из алфавита контекстно-свободной грамматики называется бесплодным, если множество цепочек терминальных символов им порождаемых пусто ($\{\alpha \mid A \Rightarrow \alpha, \alpha \in T^*\} = \emptyset$). Символ из алфавита контекстно-свободной грамматики называется недостижимым, если он не появляется ни в одной сентенциальной форме этой грамматики.

1. Удаление бесплодных символов. Строится множество N , из элементов которого за некоторое число шагов выводятся терминальные символы. Символы B и D являются бесплодными в грамматике G , их и правила их содержащие можно удалить из грамматики:

$$G_1: \quad S \rightarrow Ab \quad A \rightarrow Ca \quad C \rightarrow Aa \mid a \mid b$$

2. Удаление недостижимых символов. Строится множество V , в которое помещаются символы, за некоторое число шагов выводимые из символа S . Недостижимых символов в грамматике G не обнаружено. Грамматика G_1 – приведенная.

Критерии: За отсутствие каждого определения или ошибки в нем: -3. За ошибку при преобразовании: -7.

10. Ответ:

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
	&a	x	@	11	*	y	x	@	/	—	+=	;	&x	#+	5	&y	+#	
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
	—	x	+	1	—	<=	28	!F	1	!	&y	&x	y	8	+	=	7	+
	36	37																
	=	;																

```

M:  a += - x * 11 - y / (- x);
    if (! (x ++ <= 5 - ++ y + x - 1)) goto L;
    goto M;
L:  y = (x = y + 8) + 7;

```

```

do      a += - x * 11 - y / (- x);
while (x ++ <= 5 - ++ y + x - 1);
y = (x = y + 8) + 7;

```

Критерии: за неверный ответ на первый вопрос: -7, за неверный ответ на второй вопрос, если первый вариант верен: -3, иначе: -10.

Экзамен, 2010 год. Ответы и критерии оценок.

Вариант 3.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{ a^i c^n b^j c^m \mid i, j \in \{0, 1\}, n \geq 1, m \geq 0 \}$

(б) КС – "да" (-1), КЗ – "да" (-1), регулярные – "да" (-3), языки типа 0 – "да" (-1).

(в) КС – "да" (-1), КЗ – "да" (-1), регулярные – "нет" (-3), грамматики типа 0 – "да" (-1), неукорачивающие – "да" (-1).

Критерии: (а) описание отсутствует или с ошибками: -4, (б), (в) снимаемые баллы за каждый неверный или отсутствующий в ответе (обоснование не обязательно).

2. Ответ: (а) $Tun\ 3\ (регулярные) \subset Tun\ 2\ (КС) \subset Tun\ 1\ (КЗ) \subset Tun\ 0$

(б) При обосновании класса языка можно опираться на утверждения

языки вида $L = \{ a^n \mid n \geq 1 \}$ – типа 3, регулярные.

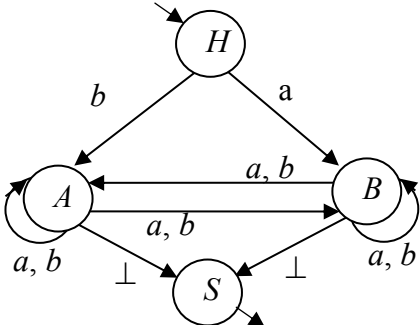
языки вида $L = \{ a^n b^n \mid n \geq 1 \}$ – типа 2, контекстно-свободные.

языки вида $L = \{ a^n b^n c^n \mid n \geq 1 \}$ – типа 1, контекстно-зависимые.

Критерии: (а) нет ответа или иерархия приведена с ошибками: -5, (б) нет ответа; неверный ответ: -2 (в отношении каждого из трех примеров).

3. Ответ: Автомат недетерминирован

G' : $H \rightarrow aB \mid bA$ $A \rightarrow aC \mid bC$ $B \rightarrow aC \mid bC$ $C \rightarrow aC \mid bC \mid \perp$

Функция переходов исходного автомата			
	$\delta(H, a) = B$ $\delta(H, b) = A$	$\delta(A, a) = A$ $\delta(A, b) = A$ $\delta(A, \perp) = S$	$\delta(B, a) = A$ $\delta(B, b) = A$ $\delta(B, \perp) = S$
	Функция переходов детерминированного автомата (C ≡ AB)		
	$\delta(H, a) = B$ $\delta(H, b) = A$	$\delta(A, a) = C$ $\delta(A, b) = C$ $\delta(B, a) = C$ $\delta(B, b) = C$	$\delta(C, a) = C$ $\delta(C, b) = C$ $\delta(C, \perp) = S$

Критерии: (а) нет или неправильная диаграмма состояний исходного автомата: -5, (б) построен неэквивалентный или недетерминированный автомат: -5, (в) праволинейная грамматика построена, но неэквивалентная или не соответствующая новой диаграмме: -5, (г) нет ответа или приведена программа анализатора: 0

4. Ответ: Описание буквы «Г» размера $n \times m$ — это цепочка $c^n a^m$. Следовательно, требуется построить грамматику, порождающую язык $\{ c^n a^m \mid n, m \geq 1 \}$:

$S \rightarrow cS \mid cA$

$A \rightarrow aA \mid a$

Соглашение: запись $\langle a \rangle$ является сокращением $\langle cout \ll "a"; \rangle$

Грамматика с действиями: $S \rightarrow c\langle c \rangle S\langle e \rangle \mid c\langle c \rangle\langle c \rangle A\langle e \rangle\langle e \rangle$ $A \rightarrow a\langle a \rangle A \mid a\langle a \rangle$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -3.

5. Ответ: Восстановленная грамматика имеет вид: $G = (\{a, b\}, \{A, B, C, D, S\}, P, S)$

$P:$ $S \rightarrow bABCD bAD$ $B \rightarrow bC \mid D$ $D \rightarrow bE \mid A$
 $A \rightarrow CE$ $C \rightarrow aD \mid \varepsilon$ $E \rightarrow aB \mid \varepsilon$

Анализатор некорректен: он выдаёт ошибку на правильной цепочке "bb". Критерий применимости метода рекурсивного спуска для грамматик с эпсилон-правилами: Метод рекурсивного спуска применим, если и только если для любых двух правил $X \rightarrow \alpha \mid \beta$ выполняются следующие условия: (1) $first(\alpha) \cap first(\beta) = \emptyset$, (2) справедливо не более чем одно из двух соотношений: $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$; (3) если $\beta \Rightarrow \varepsilon$, то $first(\alpha) \cap follow(X) = \emptyset$.

Грамматика G показанным условиям не соответствует: для двух правил $B \rightarrow bC$ и $B \rightarrow D$ имеем: $first(bC) \cap first(D) = \{b\} \cap \{ab\} \neq \emptyset$.

Критерии: Неправильно восстановлена грамматика: **-5**, Неверная формулировка достаточного условия применимости: **-5**, Неверный ответ на вопрос о соответствии грамматики критерию применимости: **-3**, Нет обоснования соответствия/несоответствия грамматики: **-2**, Неверный ответ на вопрос о корректности анализатора: **-3**, Нет обоснования корректности (нет рассуждений о конечных символах цепочек, не найден пример неправильно трактуемой цепочки и т. д.): **-2**.

6. Ответ: Программа создаётся для решения отдельной задачи автором программы и используется в некоторой конкретной операционной среде. Для программы точно определены и зафиксированы решаемая задача, её автор и операционная среда, для которой предназначена программа. Программа неотделима от автора. Программным продуктом называется программа, работающая без авторского надзора в рамках некоторого набора операционных сред. Программный продукт может исполняться, тестироваться и модифицироваться без участия автора (он отчуждён от автора). Для программного продукта должна быть разработана достаточно полная пользовательская и техническая документация. Программный продукт должен быть настраиваемым. Системный (интегрированный) программный продукт есть комплекс (взаимосогласованных) программных продуктов (пакет).

Критерии: За отсутствие комментариев к какому-либо виду программ: **-4**.

7. Ответ: (1) вынесение инвариантных вычислений из тела цикла, (2) замена операций с переменными цикла, (3) слияние, (4) расщепление и (5) разворачивание циклов. Например, расщепление цикла:

<code>for (i = 0; i < n; i++)</code>		<code>if (x < y)</code>
{	<code>if (x < y)</code> { S ₁ ; }	}
<code>else</code> { S ₂ ; }		
		<code>else for (i = 0; i < n; i++) { S₁; }</code>
		<code>for (i = 0; i < n; i++) { S₂; }</code>

Критерии: (а) За отсутствие какого-либо из видов оптимизации: **-2**, (б) За отсутствие примера: **-4**.

8. Ответ: По функциональному наполнению современные библиотеки делятся на (1) библиотеки функций (процедур) и макроопределений, (2) библиотеки классов и (3) библиотеки компонентов. В свою очередь библиотеки классов могут представлять собой (2.1) совокупности независимых классов, (2.2) иерархии классов, (2.3) иерархии шаблонов классов. Библиотека C++ относится к библиотекам, построенным на основе иерархий шаблонов классов с добавлением отдельных функций и макроопределений.

1. Библиотеки функций (процедур) и макроопределений содержат простейшие (с точки зрения библиотек) составные части, некоторые из которых имеют только неформализованные, семантические связи. 2. Библиотеки классов, в зависимости от степени своей организации, содержат более или менее формально связанные и структурированные описания и реализации классов объектов. 3. Библиотеки компонентов содержат тесно связанные описания и реализации сложных программных объектов, предназначенных для совместного использования.

Критерии: За пропуск какого-либо вида библиотек: **-2** (за каждый пропуск из 6-ти). За пропуск объяснения: **-3** (за каждый из трёх).

9. Ответ: Грамматики G_1 и G_2 называются эквивалентными, если их языки совпадают: $L(G_1) = L(G_2)$.

Строим множество правил X , в которое входят все правила, из которых за некоторое число шагов выводится пустая цепочка:

1. $X_0 = \{C\}$
2. $X_1 = \{C\}, N_1 = N_0$

Строим множество правил P' , в которое входят все правила, кроме правил вида $R \rightarrow \varepsilon$:

P' : $S \rightarrow Ab \mid BaCaC \quad A \rightarrow CaB \rightarrow CbC \rightarrow Aa \mid Bb \mid a$

Дополняем множество правил P' правилами с всевозможными комбинациями вхождений и невхождений символа C , так как в исходной грамматике есть правило $C \rightarrow \varepsilon$:

P'' : $S \rightarrow Ab \mid BaCaC \mid BaaC \mid BaCa \mid Baa$
 $A \rightarrow Ca \mid a \quad B \rightarrow Cb \mid b \quad C \rightarrow Aa \mid Bb \mid a$

$P_{\text{неукор}} = P''$

Критерии: Нет определения или оно неверно: **0**, за каждую существенную ошибку при преобразовании: **-3**.

10. Ответ.

		1	2	3	4	5		6	7	8	9	10	11	12	13	14	15
ПОЛИЗ		&t	&p	#+	&q		+#	t	@	/	4	*	+	=	;	t	u
16	17	18	19	20	21	22	23		24	25	26	27	28	29	30	31	32
<	34	!F	26	!	&t	+#		;	14	!	&r	p	7	+	==	;	21
33	34	35	36	37	38	39											
!	&r	r	2	%	=	;	>										

```

t = (p++) + (++q) / (-t) * 4;
L: if (! (t < u)) goto E;
    goto N;
M: ++t;
    goto L;
N: r -= p + 7;
    goto M;
E: r = r % 2;

for (t = (p++) + (++q) / (-t) * 4; t < u; ++t)
    r -= p + 7; r = r % 2;

```

Критерии: за неверный ответ на первый вопрос: **-7**, за неверный ответ на второй вопрос, если первый вариант верен: **-3**, иначе: **-10**.

Экзамен, 2010 год. Ответы и критерии оценок.

Вариант 4.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{ a^i c^n b^j c^m \mid i, j \in \{0, 1\}, n \geq 1, m \geq 0 \}$

(б) КС – "да" (-1), КЗ – "нет" (-1), неукорачивающие – "нет" (-1), тип 0 – "да" (-1), регулярный – "да" (-3).

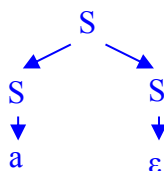
(в) $k = 3$ (поскольку для него существует левостолбчатая, то есть регулярная грамматика)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий в ответе (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: (а) КС-грамматика G называется неоднозначной, если существует (хотя бы одна) цепочка $\alpha \in L(G)$, для которой может быть построено два или более различных деревьев вывода. (б, в) Этот язык однозначен, так как для него можно построить однозначную грамматику, но можно выписать и неоднозначную грамматику. (г) Обоснование неоднозначности: порождаемая грамматикой цепочка "а" имеет два разных дерева вывода.

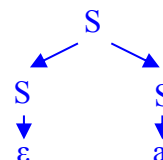
Однозначная

$S \rightarrow \varepsilon$
 $S \rightarrow a$
 $S \rightarrow aa$



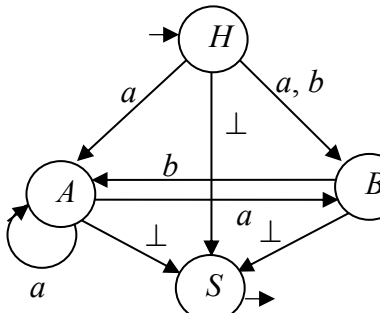
Неоднозначная

$S \rightarrow SS$
 $S \rightarrow \varepsilon$
 $S \rightarrow a$
 $S \rightarrow aa$



Критерии: (а) нет ответа; определение с ошибками: -5, (б) нет ответа; неверный ответ, нет грамматики-примера: -3, (в) нет ответа; неверный ответ: -6, (г) нет обоснования неоднозначности грамматики-примера: -2.

3. Ответ: Автомат недетерминирован

	Правильная грамматика исходного автомата		
	$H \rightarrow aA \mid aB \mid bB \mid \perp$		$A \rightarrow aA \mid aB \mid \perp$ $B \rightarrow bA \mid \perp$
	Функция переходов детерминированного автомата ($C \equiv AB$)		
	$\delta(H, a) = C$ $\delta(H, b) = B$ $\delta(H, \perp) = S$	$\delta(B, b) = A$ $\delta(B, \perp) = S$ $\delta(A, a) = C$ $\delta(A, \perp) = S$	$\delta(C, a) = C$ $\delta(C, b) = A$ $\delta(C, \perp) = S$

Где: $S \rightarrow A\perp \mid B\perp \mid C\perp \mid \perp$ $A \rightarrow Bb \mid Cb$ $B \rightarrow b$ $C \rightarrow Aa \mid Ca \mid a$

Критерии: (а) нет или неправильная правостолбчатая грамматика исходного автомата: -5, (б)

построен неэквивалентный или недетерминированный автомат: -5, (в) левостолбчатая грамматика построена, но неэквивалентная или не соответствующая новой диаграмме: -5, (г) нет ответа или приведена программа анализатора: 0.

4. Ответ: Описание буквы «Г» размера $n \times m$ — это цепочка $c^n a^m$. Следовательно, требуется построить грамматику, порождающую язык $\{c^n a^m \mid n, m \geq 1\}$:

$S \rightarrow cS \mid cA$ $A \rightarrow aA \mid a$

Соглашение: запись $\langle a \rangle$ является сокращением $\langle cout \ll "a"; \rangle$

Грамматика с действиями: $S \rightarrow c\langle c \rangle S\langle e \rangle \langle e \rangle \mid c\langle c \rangle \langle c \rangle A\langle e \rangle \langle e \rangle$ $A \rightarrow a\langle a \rangle \langle a \rangle A \mid a\langle a \rangle \langle a \rangle$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -3.

5. Ответ: Восстановленная грамматика имеет вид: $G = (\{p, q\}, \{A, B, C, D, S\}, P, S)$

$$P: \quad S \rightarrow qApCq \quad A \rightarrow pA \mid B \quad B \rightarrow qB \mid C \quad C \rightarrow qC \mid \varepsilon$$

Анализатор некорректен: он выдаёт ошибку, например, на правильной цепочке "qrrrq". Вообще, любая правильная цепочка должна заканчиваться символом 'q', а этот символ может быть зря прочитан процедурой "C()", что приведёт к неправильной трактовке всех правильных цепочек. Контекстно-свободная грамматика имеет канонический вид для метода рекурсивного спуска, если выполняется критерий применимости этого метода: метод применим, если и только если для любых двух правил $X \rightarrow \alpha \mid \beta$ выполняются следующие условия: (1) $first(\alpha) \cap first(\beta) = \emptyset$; (2) справедливо не более чем одно из двух соотношений: $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$; (3) если $\beta \Rightarrow \varepsilon$, то $first(\alpha) \cap follow(X) = \emptyset$. Грамматика G каноническому виду не соответствует: для двух правил $A \rightarrow pA$ и $A \rightarrow B$ имеем: (1) $first(pA) \cap first(B) = \{p\} \cap \{q\} = \emptyset$; (2) $B \Rightarrow \varepsilon$; из цепочки aA пустую цепочку вывести нельзя, но (3) $first(pA) \cap follow(A) = \{p\} \cap \{pq\} = \{p\} \neq \emptyset$.

Критерии: Неправильно восстановлена грамматика: -5. Неверный ответ на вопрос о соответствии грамматики каноническому виду: -3. Нет обоснования соответствия/несоответствия грамматики: -2. Неверный ответ на вопрос о корректности анализатора: -3. Нет обоснования корректности (не найден пример "плохой" цепочки): -2.

6. Ответ: Основные схемы: нисходящая (идеальный случай, последовательность слабо связанных шагов обработки программ), каскадная (циклическое повторение основных шагов разработки), каскадно-возвратная (оценка ситуации после каждого шага с возможным возвратом на ранее пройденные стадии разработки), спиральная (разработка прототипов и выпуск постепенно развивающихся версий)

Критерии: За пропуск какой-либо из четырех схем: -3. За отсутствие комментариев к какой-либо схеме: -2.

7. Ответ: Для хранения локальных данных процедур используется (1) динамическая память (2) со стековой дисциплиной распределения, (3) относящаяся к локальным областям памяти.

Критерии: Неправильный ответ, нет ответа на любой вопрос: -4. 8. Описать основное назначение редакторов связей.

8. Ответ: (1) Связывание между собой объектных модулей, составляющих единую программу, (2) подготовка таблиц относительных адресов объектов, входящих в общие для них зоны памяти, (3) статическое подключение библиотек.

Критерии: За пропуск какой-либо задачи компонента: -4 (за каждый пропуск).

9. Ответ: 1. Избавляемся от левой рекурсии в правиле для символа A:

$$G: \quad S \rightarrow aAb \mid Abc \quad A \rightarrow aA \mid \varepsilon \quad B \rightarrow Bc \quad C \rightarrow aA \mid b$$

2. В правилах для символа S проводим объединение альтернатив, начинающихся с одинаковых терминальных символов:

$$G: \quad S \rightarrow aS^1 \quad S^1 \rightarrow Ab \mid Bc \quad A \rightarrow aA \mid \varepsilon \quad B \rightarrow bC \quad C \rightarrow aA \mid b$$

3. Подставляем правила для символов A и B в те правила для символов S^1 , которые не начинаются с терминальных символов:

$$G: \quad S \rightarrow aS^1 \quad S^1 \rightarrow aAb \mid b \mid bCc \quad A \rightarrow aA \mid \varepsilon \quad B \rightarrow bC \quad C \rightarrow aA \mid b$$

4. В правилах для символа S^1 проводим объединение альтернатив, начинающихся с одинаковых терминальных символов:

$$G: \quad S \rightarrow aS^1 \quad S^1 \rightarrow aAb \mid bS^2 \quad S^2 \rightarrow Cc \mid \varepsilon \quad A \rightarrow aA \mid \varepsilon \quad B \rightarrow bC \quad C \rightarrow aA \mid b$$

5. Подставляем правила для символа C в правило для символа S^2 , которое не начинается с терминальных символов:

$$G: \quad S \rightarrow aS^1 \quad S^1 \rightarrow aAb \mid bS^2 \quad S^2 \rightarrow aAc \mid bc \mid \varepsilon \\ A \rightarrow aA \mid \varepsilon \quad B \rightarrow bC \quad C \rightarrow aA \mid b$$

6. Считаем пересечения множеств first и follow для символов A и S^2 :

$$first(A) \cap follow(A) = \{a\} \cap \{b, c\} = \emptyset \quad first(S^2) \cap follow(S^2) = \{a, b\} \cap \{\varepsilon\} = \emptyset$$

Критерии: За каждую существенную ошибку при преобразовании: -3.

10. Ответ.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ЛИЗ	n	0	<	38	!F	x	y	+	z	—	4	>	23	!F	&x	n

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
2	+	=	;	38	!	z	0	>	34	!F	&x	2	=	;	38	!

34	35	36	37	38
&y	z	=	;	}

Ответ:

```

if (! (n > 0))          goto E;
if (! (x + y - z > 4))  goto L;
x = n + 2;              goto E;
L: if (! (z > 0))       goto M;
x = 2;                  goto E;
M: y = z;
E:

```

```

if (n < 0) if (x + y - z > 4) x = n + 2; else if (z > 0) x = 2; else y = z;

```

Критерии: за неверный ответ на первый вопрос: **-7**, за неверный ответ на второй вопрос, если первый вариант верен: **-3**, иначе: **-10**.

Экзамен, 2010 год. Ответы и критерии оценок.

Вариант 5.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

Ответ: (а) $L(G) = \{ a^i c^n b^j c^m \mid i, j \in \{0, 1\}, n \geq 1, m \geq 0 \}$

(б) КС – "да" (-1), КЗ – "нет" (-1), неукорачивающие – "нет" (-1), грамматики типа 0 – "да" (-1), регулярные – "да" (-3).

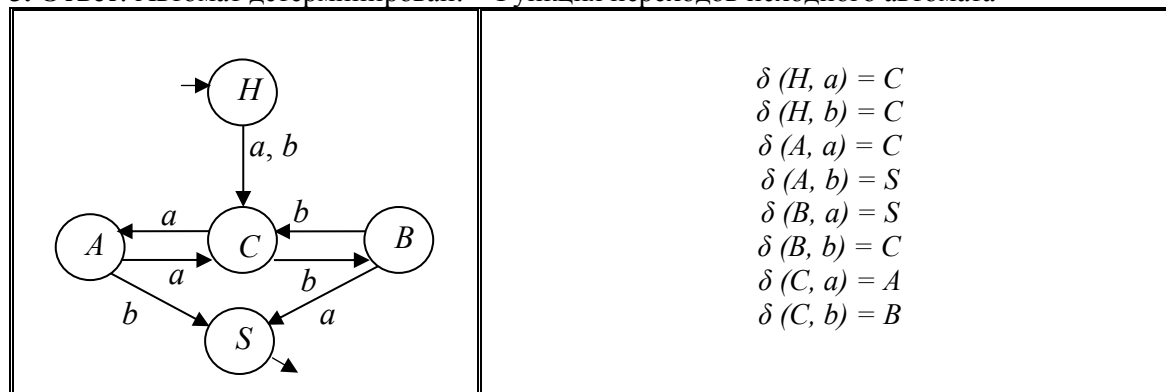
(в) $k = 3$ (поскольку для него существует левостолбчатая, то есть регулярная грамматика)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ см. выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: (а) Чистая виртуальная функция–функция вида: virtual<тип_рез><имя_ф> (<список_ф_п>)=0; (б) Чистые виртуальные функции используются в тех случаях, когда невозможно предложить разумной реализации виртуальной функции в базовом классе, и эти реализации реально нужны только для производных классов. (в) Нельзя создавать объекты классов, имеющих чистые виртуальные функции. (г) Можно.

Критерии: (а) нет ответа или определение дано с ошибками: -5 (правильным считать любой из двух вариантов ответа), (б) нет ответа; неверный ответ: -3, (в) нет ответа; неверный ответ: -3, (г) нет ответа; неверный ответ: -3.

3. Ответ: Автомат детерминирован: Функция переходов исходного автомата



```
enum State { H, A, B, C, S, Error } FA_State = H;
```

```
Analyze () {
    do {switch (FA_State) {case H: if (c == 'a') { c = GetS (); FA_State = C; }
                                else if (c == 'b') { c = GetS (); FA_State = C; }
                                else FA_State = Error; break;
                                case A: if (c == 'a') { c = GetS (); FA_State = C; }
                                else if (c == 'b') { c = GetS (); FA_State = S; }
                                else FA_State = Error; break;
                                case B: if (c == 'a') { c = GetS (); FA_State = S; }
                                else if (c == 'b') { c = GetS (); FA_State = C; }
                                else FA_State = Error;
                                case C: if (c == 'a') { c = GetS (); FA_State = A; }
                                else if (c == 'b') { c = GetS (); FA_State = B; }
                                else FA_State = Error; break;
                                case Error: break;
                                }
        } while (FA_State != S);
}
```

Критерии: (а) нет или неправильная диаграмма состояний исходного автомата: -5, (б) нет ответа на вопрос о детерминированности; неверный ответ; ошибочное объяснение: -5, (в) нет ответа или вместо программы анализатора построен конечный автомат: 0.

4. Ответ: B, O, X, A, A, C

Критерии: за каждую неверную (лишнюю или пропущенную) печать -2 балла.

5. Ответ: Восстановленная грамматика имеет вид: $G = (\{p, q\}, \{A, B, C, D, S\}, P, S)$

$P: S \rightarrow qApCq \quad A \rightarrow pA \mid B \quad B \rightarrow qB \mid C \quad C \rightarrow qC \mid \varepsilon$

Анализатор некорректен: он выдаёт ошибку, например, на правильной цепочке "qrrrq". Вообще, любая правильная цепочка должна заканчиваться символом 'q', а этот символ может быть зря прочитан процедурой "C()", что приведёт к неправильной трактовке всех правильных цепочек. Контекстно-свободная грамматика имеет канонический вид для метода рекурсивного спуска, если выполняется критерий применимости этого метода: метод применим, если и только если для любых двух правил $X \rightarrow \alpha \mid \beta$ выполняются следующие условия: (1) $first(\alpha) \cap first(\beta) = \emptyset$; (2) справедливо не более чем одно из двух соотношений: $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$; (3) если $\beta \Rightarrow \varepsilon$, то $first(\alpha) \cap follow(X) = \emptyset$. Грамматика G каноническому виду не соответствует: для двух правил $A \rightarrow pA$ и $A \rightarrow B$ имеем: (1) $first(pA) \cap first(B) = \{p\} \cap \{q\} = \emptyset$; (2) $B \Rightarrow \varepsilon$; из цепочки aA пустую цепочку вывести нельзя, но (3) $first(pA) \cap follow(A) = \{p\} \cap \{pq\} = \{p\} \neq \emptyset$.

Критерии: Неправильно восстановлена грамматика: -5. Неверный ответ на вопрос о соответствии грамматики каноническому виду: -3. Нет обоснования соответствия/несоответствия грамматики: -2. Неверный ответ на вопрос о корректности анализатора: -3. Нет обоснования корректности (не найден пример "плохой" цепочки): -2.

6. Ответ: X(), Y(), X(), Y(Y & y), X(), Y(int), ~Y(), ~X(), ~Y(), ~X(), ~Y(), ~X().

Критерии: За каждую ошибку: -2.

7. Ответ: (1) Преобразования с помощью конструкторов преобразований, (2) Преобразования с помощью функций преобразования.

Пример: (1) `struct S { S (long); }; // конструктор вида: long --> S`
`void f (long); void f (char*); ... f (a); // f (long)`

Пример: (2) `struct S { operator int (); }; // преобразование вида: S --> int`
`void g (S); void g (char*); ... g (1); // g (S);`

Критерии: Нет ответа на первый вопрос: 0. Отсутствует какое-либо одно из преобразований: -5. Отсутствуют примеры: -3 за каждый отсутствующий пример.

8. Ответ: Связывание между собой объектных модулей, составляющих единую программу, подготовка таблиц относительных адресов объектов, входящих в общие для них зоны памяти, статическое подключение библиотек.

Критерии: За пропуск какой-либо задачи компонента: -4 (за каждый пропуск).

9. Ответ:

```
int g (vector<int> & vect)
{ int cnt = 0; int i = 0, j = vect.size()-1;
  for (; i<j; ++i,--j)
  { int t=vect[i]; vect[i]=vect[j]; vect[j]=t; ++cnt; }
  return cnt;
}
```

Критерии: За каждую существенную ошибку: -3. Если вместо индексации использованы указатели, но программа написана правильно – снижение на 2 балла.

10. Ответ.

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	n	0	<	38	!F	x	y	+	z	—	4	>	23	!F	&x
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	n	2	+	=	;	38	!	z	0	>	34	!F	&x	2	=
32	33	34	35	36	37	38									
	!	&y	z	=	;										

```

if (! (n > 0)) goto E;
if (! (x + y - z > 4)) goto L;
x = n + 2; goto E;
L: if (! (z > 0)) goto M;
x = 2; goto E;
M: y = z;
E:

```

`if (n < 0) if (x + y - z > 4) x = n + 2; else if (z > 0) x = 2; else y = z;`

Критерии: за неверный ответ на первый вопрос: -7. за неверный ответ на второй вопрос, если первый вариант верен: -3, иначе: -10.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Если есть ошибки в операторах функции `main()`, исправьте их с помощью операции указания области видимости `::`. Вычеркните операторы, которые не удалось исправить с помощью `::`. Для оставшихся операторов с помощью операции `::` обозначьте ту область видимости, к которой относится вызываемый метод.

```
int x = 4;
class A { int x; public: A (int n = 1);
        int f (int a = 0, int b = 0); };
class B: public A { int x; public: B (int n = 2);
        int f (int a); };
class C: public B { int x; public: C (int n = 3);
        int f (int a, int b);
        int g (A * p); };
A * p; B b; C c;
int main () { p = & b; x = c.g (& c);
              x = c.f ();
              x = c.f (x);
              x = c.f (x, 1);
              x = p -> f ();
              x = p -> f (x);
              x = p -> f (x, 1); return 1; }
```

2. Если есть ошибки в приведенной программе, то **объясните**, в чем они заключаются. Ошибочные операторы или ключевые слова вычеркните (допускается не более двух вычеркиваний). Что будет выдано в стандартный поток вывода при работе получившейся программы?

```
class A {public: static void f(int x){ h(x);cout<<"A::f,"<<x<<endl; }
        void g() { cout<<"A::g" <<endl; }
        void h(int x){ g(); cout<<"A::h,"<<x<<endl; }};
class B: virtual public A {
        public: static void f(int x){ h(x);cout<<"B::f,"<<x<<endl; }
        void g() { cout<<"B::g" <<endl; }
        void h(int x){ g(); cout<<"B::h,"<<x<<endl; }};
int main() { B::f (0); B b; A * p = & b;
            p -> f (1);
            p -> g ();
            p -> h (2);
            A::f (3); return 0; }
```

3. Для каждого вызова перегруженной функции с одним параметром укажите шаг алгоритма, на котором будет выбрана наиболее подходящая функция или выявлена неоднозначность. Для выбранной функции укажите ее прототип.

```
int f (int a = 0) { return a; }
int f (double a) { return a; }
int main () { short int s; int i; bool b; enum e{A,B,C}; float f = 1.0f;
              f ();
              f (s);
              f (f);
              f (b);
              f (A); }
```

4. Для объектов из задания 1 определить, какие конструкторы и деструкторы и в каком порядке будут выполняться при работе следующего фрагмента программы:

```
int main () { C c; A a = c; struct D { B b; D (): b (5) {} } d; }
```

5. В открытой части класса *cl*, предназначенного для работы со сложными объектами, имеются конструктор и функция преобразования. Для облегчения работы с объектами этого класса создаётся дополнительный класс *ObjPtr*, в закрытую часть которого включается поле указателя на класс *cl*.

```
template <class T> class cl { /* ... */
public: cl (const T s = 0);
       operator T ();
};
template<class T> class ObjPtr { /* ... */ cl<T> *c; public: /* ... */};
```

Для класса "*ObjPtr*" написать реализацию операций сложения "+" таким образом, чтобы правильными оказались выражения (обязательно использовать и методы, и внешние функции):

```
ObjPtr <int> t1, t2; /* ... */ (t + 3) + (5 + t1) + (t1 + t2) /* ... */
```

6. Что такое виртуальный базовый класс и для чего он используется в Си++? Как синтаксически описываются классы, объявляющие и использующие виртуальные базовые классы?

7. Что будет выдано в стандартный поток вывода при работе следующей программы?

```
void f(X & x, int n);
struct X { X () { try { f(*this, -1); cout << "a"; }
               catch (X) { cout << "b"; }
               catch (int){ cout << "c"; } }
        X (X &) { cout << "d"; }
        virtual ~X () { cout << "e"; } };
struct Y: X { Y () { try { f (*this, 0); cout << "f"; }
               catch (Y) { cout << "g"; }
               catch (int){ cout << "h"; }
               cout << "i"; }
        Y(Y&){ cout << "j"; }
        ~Y () { cout << "k"; } };
void f(X & x,int n){ try { if (n < 0) throw -n;
                        else if (n == 0) throw x;
                        else throw n; }
               catch (int) { cout << "l"; } }

int main() { try { Y a; }
               catch (...) { cout << "m"; return 1;}
               cout << "n"; return 0; }
```

8. Если есть ошибки в следующем фрагменте, то в чем они заключаются? Исправьте ошибки, ничего не удаляя, добавив в общей сложности не более 14 символов.

```
class A { public: int * n; int m; };
class B: public A { public: int * p; };
class C: public A { public: int * c; };
class D: public B: public C { public: int * e; };
int main () { D fA, * f = new D; fA.m =0;
               return *((* f).e = & fA.m); }
```

9. Какие виды пользовательских преобразований допустимы в программах на языке Си++? Какие ограничения накладываются на использование таких преобразований при разрешении статического полиморфизма функций? Привести пример фрагмента программы с использованием этих операций.

10. Описать функцию *g()* с двумя параметрами: контейнер-список целых элементов и контейнер-вектор указателей на целые. Функция должна, последовательно проходя по элементам контейнеров от начала к концу вектора и от конца к началу списка, менять местами элементы (целые значения) контейнеров, после чего распечатать в прямом порядке целые значения сначала для списка, затем для вектора. Функция возвращает количество переставленных элементов контейнеров.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Если есть ошибки в операторах функции `main()`, исправьте их с помощью операции указания области видимости `::`. Вычеркните операторы, которые не удалось исправить с помощью `::`. Для оставшихся операторов с помощью операции `::` обозначьте ту область видимости, к которой относится вызываемый метод.

```
int x = 4;
class A { int x; public: A (int n = 3);
        int f (int a = 0, int b = 0); };
class B: public A { int x; public: B (int n = 1);
        int f (int a = 0); };
class C: public B { int x; public: C (int n = 2);
        int f (int a, int b);
        int g (A * p) };
A * p; B b; C c;
int main () { p = & b; x = c.g (& b);
              x = c.f ();
              x = c.f (x);
              x = c.f (x, 1);
              x = p -> f ();
              x = p -> f (x);
              x = p -> f (x, 1);          return 2; }
```

2. Есть ли ошибки в интерфейсах классов `C` и `D` программы на C++? Если есть, то **объясните**, в чем они заключаются и внесите нужные исправления, оставив без изменения реализацию классов и функции `"main"`. Что будет выдано в стандартный поток вывода при работе получившейся программы?

```
class C {public: C (int x = 0) {}
        virtual int f (int x) { cout<<"C::f,"<<x<<endl; return h (x); }
        virtual int g () { cout<<"C::g" <<endl; return 1; }
        virtual int h (int x) { cout<<"C::h,"<<x<<endl; return x; }
        virtual operator int () { return 99; } };
class D: public C { public:
        int f (int x) { cout<<"D::f,"<<x<<endl; return h (x); }
        int g (int x) { cout<<"D::g" <<endl; return 1; }
        int h (int x) { cout<<"D::h,"<<x<<endl; return x; }
        D (int x = 0) {}
        operator int () { return 100; } };
int main() { const D d; C const * const t = & d;
            t -> f (3); t -> f (d);
            t -> g (); t -> h (5);          return 0; }
```

3. Для каждого вызова перегруженной функции с одним параметром укажите шаг алгоритма, на котором будет выбрана наиболее подходящая функция или выявлена неоднозначность. Для выбранной функции укажите ее прототип.

```
int f(double a = 1.0){return a;}
int f(long double a = 5.0){return a;}
int main () { float f = 1.0f; double d = 2.0; long double ld = 3.01;
              f ();
              f (4); f (f);
              f (d); f (ld); }
```

4. Для объектов из задания 1 определить, какие конструкторы и деструкторы и в каком порядке будут выполняться при работе следующего фрагмента программы:

```
int main () { class D { B b; A a; public: D (): a (b) { } } d; C c; }
```

5. В открытой части класса *cl*, предназначенного для работы со сложными объектами, имеются конструктор и функция преобразования. Для облегчения работы с объектами этого класса создаётся дополнительный класс *ObjPtr*, в закрытую часть которого включается поле указателя на класс *cl*.

```
template <class T> class cl { /* ... */
public: cl (const T s = 0);
       operator T ();
};
template<class T> class ObjPtr { /* ... */ cl <T> *c; public: /* ... */};
```

Для класса "*ObjPtr*" написать реализацию операций сравнения на равенство "==" таким образом, чтобы правильными оказались выражения (обязательно использовать и методы, и внешние функции):

```
ObjPtr<int> t1, t2; /* ... */ (t1 == 3) || (5 == t1) && (t1 == t2) /* ... */
```

6. Что такое наследование и для чего оно используется в Си++? Какой длины и какого вида цепочки наследования можно строить в Си++? Как синтаксически описывается наследственная иерархия разных видов в Си++?

7. Что будет выдано в стандартный поток вывода при работе следующей программы?

```
void f(X & x, int n);
struct X { X () { try { f(*this, 0); cout << "a"; }
               catch (X) { cout << "b"; }
               catch (int) { cout << "c"; } }
        X (X &) { cout << "d"; }
        virtual ~X () { cout << "e"; } };
struct Y: X { Y () { try { f(*this, 0); cout << "f"; }
               catch (Y) { cout << "g"; }
               catch (int) { cout << "h"; }
               cout << "i"; }
        Y(Y&) { cout << "j"; }
        ~Y () { cout << "k"; } };
void f(X & x, int n) { try { if (n < 0) throw -n;
                           else if (n == 0) throw x;
                           else throw n; }
               catch (int) { cout << "l"; } }
int main() { try { Y a; }
            catch (...) { cout << "m"; return 1; }
            cout << "n"; return 0; }
```

8. Если есть ошибки в следующем фрагменте, то в чем они заключаются? Исправьте ошибки, ничего не удаляя, добавив в общей сложности не более 12 символов.

```
class S { public int s; void sp (int si) { s = si; } };
class T: S { public int t; void tp (int ti) { t = ti; s = ti; } };
class U: T { public int u; void up (int ui) { u = ui; t = ti; s = ti; } };
void g () { U * pu = new U;
           T * pt = pu;
           S * ps = pu; }
```

9. Что такое "дружественная функция"? Описать основное отличие функций-друзей от методов классов. Привести пример использования дружественной функции.

10. Написать функцию *g()* с параметром, представляющим собой контейнер-список указателей на элементы длинного целого типа. Функция, просматривает контейнер от конца к началу и меняет знак значения, на которое указывает текущий указатель, если следующее за ним значение отрицательно. Например, список (-1,-2,3,4) превратится в (-1,2,-3,4). Затем функция печатает значения, на которые указывают элементы контейнера в прямом порядке. Функция возвращает число измененных элементов.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Если есть ошибки в операторах функции `main()`, исправьте их с помощью операции указания области видимости `::`. Вычеркните операторы, которые не удалось исправить с помощью `::`. Для оставшихся операторов с помощью операции `::` обозначьте ту область видимости, к которой относится вызываемый метод.

```
int x = 4;
class A { int x; public: A (int n = 1);
        virtual int f (int a = 0, int b = 0); };
class B { int x; public: B (int n = 2);
        int f (int a = 0); };
class C: public A, public B { int x; public: C (int n = 3);
        int f (int a, int b = 0);
        int g (A * p); };
A * p; C c;
int main () { p = & c; x = c.g (p);
              x = c.f ();
              x = c.f (x);
              x = c.f (x, 1);
              x = p -> f ();
              x = p -> f (x);
              x = p -> f (x, 1);          return 3; }
```

2. Укажите лишние и ошибочные операции динамического приведения типа, если таковые имеются в функции `main()`. Дайте необходимые пояснения своим исправлениям.

```
class K { public: void g () { cout << "K::g"; } };
class L: public K { public: void f () { cout << "L::f"; } };
class M: public K { public: virtual void h () { cout << "M::h"; } };
class P: public L { public: void f () { cout << "P::f"; } };
class Q: public M { public: virtual void h () { cout << "Q::h"; } };
class R: public P { public: virtual void f () { cout << "R::f"; } };
               virtual void h () { cout << "R::h"; } };
class S: public Q { public: virtual void f () { cout << "S::f"; } };
               virtual void h () { cout << "S::h"; } };

int main() {
    S os, * s = & os; K * k; L * l; M * m; P * p; Q * q; R * r;
    k = dynamic_cast <K *>(s);    s = dynamic_cast <S *>(k);
    l = dynamic_cast <L *>(k);    m = dynamic_cast <M *>(s);
    p = dynamic_cast <P *>(l);    q = dynamic_cast <Q *>(m);
    r = dynamic_cast <R *>(q);    s = dynamic_cast <S *>(p); return 0; }
```

3. Для каждого вызова перегруженной функции с одним параметром укажите шаг алгоритма, на котором будет выбрана наиболее подходящая функция или выявлена неоднозначность. Для выбранной функции укажите ее прототип.

```
int f(int a = 1){return a;}
int f(long double a = 5.0){return a;}
int main () { short int s; int i; bool b; float f=1.0f; double d=2.0;
              f (s);
              f (i);                f (b);
              f (f);                f (d); }
```

4. Для объектов из задания 1 определить, какие конструкторы и деструкторы и в каком порядке будут выполняться при работе следующего фрагмента программы:

```
int main () { C c; class D { C c; B b; public: D () : b (c) { } } d; }
```

5. В открытой части класса *cl*, предназначенного для работы со сложными объектами, имеются конструктор и функция преобразования. Для облегчения работы с объектами этого класса создаётся дополнительный класс *ObjPtr*, в закрытую часть которого включается поле указателя на класс *cl*.

```
template <class T> class cl { /* ... */
public: cl ( const T s = 0);          operator T ();    };
template<class T> class ObjPtr { /* ... */ cl <T> * c; public: /* ...*/};
```

Для класса "*ObjPtr*" написать реализацию операции изменения знака "-" и операций присваивания "=" таким образом, чтобы правильными оказались выражения (обязательно использовать и методы, и внешние функции):

```
ObjPtr <int> t1, t2; /* ... */ t1 = 5; t2 = -t1; /* ... */
```

6. Что такое инкапсуляция и для чего она используется в Си++? Привести синтаксическую запись фрагмента программы, на котором можно проиллюстрировать понятие инкапсуляции.

7. Что будет выдано в стандартный поток вывода при работе следующей программы?

```
void f(X & x, int n);
struct X { X () { try { f(*this, 1); cout << "a"; }
              catch (X) { cout << "b"; }
              catch (int){ cout << "c"; } }
      X (X &) { cout << "d"; }
      virtual ~X () { cout << "e"; }
};
struct Y: X { Y () { try { f (*this, 1); cout << "f"; }
              catch (Y) { cout << "g"; }
              catch (int){ cout << "h"; }
              cout << "i"; }
      Y(Y&){ cout << "j"; }
      ~Y () { cout << "k"; } };
void f(X & x,int n){ try { if (n < 0) throw -n;
                        else if (n == 0) throw x;
                        else throw n; }
              catch (int) { cout << "l"; } }
int main() { try { Y a; }
            catch (...) { cout << "m"; return 1; }
            cout << "n"; return 0; }
```

8. Если есть ошибки в следующем фрагменте, то в чем они заключаются? Исправьте ошибки, ничего не удаляя, добавив в общей сложности не более 29 символов.

```
class W { public:int *w; void wf(int *wp){ w = wp; }};
class X:W { public:int *x; void xf(int *xp){ x = xp; w = xp; }};
class Y:W { public:int *y; void yf(int *yp){ y = yp; w = yp; }};
class Z:X,Y{ public:int *z; void zf(int *zp){ z = zp; x = zp; y = zp;}};
void h () { int hi; W * pw; X * px; Y * py; Z * pz;
            pz = new W; (*pz).w = & hi;
            pz -> xf ((*pz).X::w); }
```

9. Описать условия включения механизма виртуальности для метода класса. Привести пример записи виртуальной функции и обращения к ней.

10. Написать функцию *g()* с параметром, представляющим собой контейнер-вектор указателей на элементы вещественного типа. Функция просматривает контейнер в прямом порядке и обнуляет текущее указываемое значение, если следующее значение отрицательно. Например, [1, -2, 4, -6, -4, 5] => [0, -2, 0, 0, -4, 5]. Далее функция в обратном порядке печатает значения, на которые указывают элементы контейнера. Функция возвращает число измененных значений.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Если есть ошибки в операторах функции `main()`, исправьте их с помощью операции указания области видимости `::`. Вычеркните операторы, которые не удалось исправить с помощью `::`. Для оставшихся операторов с помощью операции `::` обозначьте ту область видимости, к которой относится вызываемый метод.

```
int x = 4;
class A { int x; public: A (int n = 2);      int f (int a, int b); };
class B { int x; public: B (int n = 3);      int f (int a = 0);    };
class C: public B, public A {      int x;
    public: C (int n = 1);
           int f (int a);
           int g (A * p);
};

A * p; B b; C c;
int main () { p = & c;
             x = c.g (& b);           x = c.f ();
             x = c.f (x);             x = c.f (x, 1);
             x = p -> f ();            x = p -> f (x);
             x = p -> f (x, 1);        return 4;
}
```

2. Если есть ошибки в приведенной программе, то **объясните**, в чем они заключаются. Ошибочные операторы или ключевые слова вычеркните (допускается не более двух вычеркиваний). Что будет выдано в стандартный поток вывода при работе получившейся программы?

```
void h (int x);
void f (int x) { h (x); cout << " ::f," <<x<< endl; }
void g ()      { h (0); cout << " ::g"      << endl; }
void h ()      { cout << " ::h"      << endl; }
void h (int x) { cout << " ::h," <<x<< endl; }

class T {
public: virtual void f (int x) { h (); cout << "T::f," <<x<< endl; }
       void g ()      { h (); cout << "T::g"      << endl; }
       virtual void h ()      { cout << "T::h"      << endl; }
       virtual void h (int x) { g (); cout << "T::h," <<x<< endl; } };
class U:virtual public T {
public: void f (int x) { h (x); cout << "U::f," <<x<< endl; }
       virtual void g ()      { cout << "U::g"      << endl; }
       virtual void h ()      { cout << "U::h"      << endl; }
       void h (int x) { g (); cout << "U::h," <<x<< endl; } };

int main() { U u; T * v = &u;      v -> f (1);      f (2);
            v -> g ();      g ();      v -> h ();      h ();
            v -> h (3);      h (4);
            return 0; }
```

3. Для каждого вызова перегруженной функции с одним параметром укажите шаг алгоритма, на котором будет выбрана наиболее подходящая функция или выявлена неоднозначность. Для выбранной функции укажите ее прототип.

```
class X { int x; public: X (const int n = 0) { x = n; }
        operator int () { return x; } };
class Y { X y; public: Y (const X x) { y = x; }
        operator X () { return y; } };
int f (X a = 1) { return a; } X f (Y a) { return a; }
int main () { short int s=1; bool b=true; float f=1; X x=s; Y y=x;
             f (s);           f (x);           f (b);
                                 f (f);           f (y);
}
```

4. Для объектов из задания 1 определить, какие конструкторы и деструкторы и в каком порядке будут выполняться при работе следующего фрагмента программы:

```
int main () { A a; class D { B b; A a; C c; public: D () : a(c) {} }; D d; }
```


5. В открытой части класса *cl*, предназначенного для работы со сложными объектами, имеются конструктор и функция преобразования. Для облегчения работы с объектами этого класса создаётся дополнительный класс *ObjPtr*, в закрытую часть которого включается поле указателя на класс *cl*.

```
template <class T> class cl { /* ... */
public: cl (const T s = 0);
      operator T ();
template<class T> class ObjPtr { /* ... */ cl <T> *c; public: /* ... */};
```

Для класса "*ObjPtr*" написать реализацию префиксных и постфиксных операций уменьшения "--" таким образом, чтобы правильными оказались выражения:

```
ObjPtr <int> t1; /* ... */ t1 --, -- t1 /* ... */
```

Если для какой-либо из префиксных (постфиксных) функций выбирается реализация методом класса "*ObjPtr*", другая префиксная (постфиксная) функция должна реализовываться функцией-другом.

6. Какая функция называется в Си++ виртуальной? Для чего используются виртуальные функции? Как называются классы, содержащие виртуальные функции? Привести пример синтаксической записи такого класса. Можно ли создавать объекты и описывать конструкторы таких классов?

7. Что будет выдано в стандартный поток вывода при работе следующей программы?

```
void f(X & x, int n);
struct X { X () { try { f(*this, 0); cout << "a"; }
              catch (X) { cout << "b"; }
              catch (int) { cout << "c"; } }
      X (X &) { cout << "d"; }
      virtual ~X () { cout << "e"; } };
struct Y: X { Y () { try { f(*this, -1); cout << "f"; }
              catch (Y) { cout << "g"; }
              catch (int) { cout << "h"; }
              Y(Y&) { cout << "i"; }
              ~Y () { cout << "j"; }
              cout << "k"; } };
void f(X & x, int n) { try { if (n < 0) throw -n;
                          else if (n == 0) throw x;
                          else throw n; }
      catch (int) { cout << "l"; } }
int main() { try { Y a; }
      catch (...) { cout << "m"; return 1; }
      cout << "n"; return 0; }
```

8. Если есть ошибки в следующем фрагменте, то в чем они заключаются? Исправьте ошибки, ничего не удаляя, добавив в общей сложности не более 17 символов.

```
class M { public: int t; int mp (int i) { return t + i; } };
class N { public: int t; int np (int i) { return t + i; } };
class O: M { public: int t; int op (int i) { return t + i; } };
class P: N, O { public: int p; int pp (int i) { return t + i; } };
void g () { O * po = new N; N * pn = po; M * pm = po; }
```

9. Перечислить виды полиморфизма, которые можно встретить в программах на языке Си++, дать пояснения. Привести соответствующие примеры фрагментов программы.

10. Написать функцию *g()* с параметром, представляющим собой контейнер-вектор элементов целого типа. Функция просматривает контейнер от конца к началу и меняет текущий элемент с предшествующим в векторе, если текущий меньше предшествующего. Например [1,6,5,3] => [1,3,6,5]: значение 3 поменяется сначала с 5, затем с 6. Далее функция печатает значения элементов контейнера в прямом порядке. Функция возвращает число совершенных обменов значений.

Коллоквиум, 2011 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: `x=c.C::g(& c);` `x=c.A::f();` `x=c.A::f(x);` `// x=c.B::f(x);`
`x=c.C::f(x, 1);` `x=p->A::f();`
`x=p->A::f(x);` `x=p->A::f(x, 1);`

Ошибочны 2 оператора, которые можно исправить (подчеркнуты).

Критерии: За пропуск ошибки (из 2-х): **-4.** За каждое не отмеченное или неверное уточнение области видимости (из 7): **-1.** За каждую лишнюю обнаруженную "ошибку" **-4.**

2. Ответ: Ошибочными являются указания на то, что статические методы `f()` вызывают нестатические методы. Надо исключить в методах `f()` вызовы методов `h()`. Будет выдано в поток:

`B::f, 0` `A::f, 1` `A::g` `A::g` `A::h, 2` `A::f, 3`

Критерии: За каждую не найденную ошибку, лишнюю ошибку или неправильно вызванную функцию **-2.**

3. Ответ: `f();` `// f(int),` (а) на этапе выбора функций,
 `// подходящих по числу параметров; за ответ «точное отождествление» не снижать`
`f(s);` `// f(int),` (б) целочисленное расширение
`f(f);` `// f(double),` (б) вещественное расширение
`f(b);` `// f(int),` (б) целочисленное расширение
`f(A);` `// f(int),` (б) целочисленное расширение

Критерии: За каждую ошибку **-2.**

4. Ответ: `A(), B(), C(), A(const A&), A(), B(), D(), ~D(), ~B(), ~A(), ~A(), ~C(), ~B(), ~A()`.

Критерии: За каждую ошибку **-2.**

5. Ответ:

```
const ObjPtr<T> operator+ (const ObjPtr<T> & s) const
{ ObjPtr<T> v; v.c = new cl<T>; * v.c = * c + * s.c; return v; }
const ObjPtr<T> operator+ (const T & s)
{ ObjPtr<T> v; v.c = new cl<T>; * v.c = * c + s; return v; }
friend const ObjPtr<T> operator+ (const T& s, const ObjPtr<T>& t);
template<class T> const ObjPtr<T> operator+ (const T& s, const ObjPtr<T>& t)
{ ObjPtr<T> v; v.c = new cl<T>; * v.c = * t.c + s; return v; }
```

Критерии: За отсутствие каждого примера (из 3-х) или ошибки в нем: **-4.** За отсутствие реализаций методами класса (из 2-х) при правильной реализации этих же операций функциями-друзьями: **-2.**

6. Ответ: (а). Виртуальный базовый класс – это такой базовый класс, который представляется в своих производных классах одним и тем же совместно используемым объектом. В варианте наследования (множественного) с использованием виртуальных базовых классов одинаковые части нескольких базовых классов включаются в их общий производный класс в единственном экземпляре. Виртуальное наследование позволяет избавиться от неоднозначности, которое при множественном наследовании возникает из-за наследования одного базового класса несколькими классами промежуточных наследственных уровней (как при ромбовидном наследовании).

(б) Пример виртуального наследования (без виртуальности обращения к полю `"n"` из функции `"f()"` создают неоднозначность), где виртуальным базовым классом является класс `"L"`:

```
class L { public int n; ... };
class A: virtual public L { ... };
class B: virtual public L { ... };
class C: public A, protected B { ... void f(); ... };
```

Критерии: За неверный ответ: **-5,** за отсутствие пояснения: **-4,** нет примера: **-4.**

7. Ответ: `l a d e m e`

Критерии: За каждую неверную (лишнюю или пропущенную) печать **-2.**

8. Ответ: Требуется уточнить обращения к полю "m" с помощью операций разрешения области видимости "::" для класса "B" либо для "C", например, так:

```
fA.C::m = 0; return ((* f).e = & fA.C::m);
```

Либо во всех описания производных классов вставить указание виртуального наследования (всего 4 поправки, из которых исправления в классах "B" и "C" при таком подходе – обязательны):

```
class B: virtual public A { public: int * p; };  
class C: virtual public A { public: int * c; };
```

Других ошибок нет.

Критерии: За пропуск любой из трёх ошибок или неправильное исправление: -4. За указание ошибки там, где ее нет: -5. Отсутствие указаний виртуальности наследования в классе "D" при наличии таких исправлений в классах "B" и "C" не снижать.

9. Ответ: К пользовательским преобразованиям относятся преобразования, выполняемые: (а) с помощью **конструкторов преобразования** (конструкторов с одним параметром); (б) с помощью **функций преобразования** – методов класса с профилем "**operator** <mun> ()", которые преобразуют объект класса к типу <mun>.

Ограничения: (а) Пользовательские преобразования применяются неявно только в том случае, если они однозначны. (б) Допускается не более одного пользовательского преобразования для обработки одного вызова для одного параметра. (в) Конструктор должен иметь разрешение быть вызванным неявно (у него должен отсутствовать спецификатор explicit).

```
struct S { S (long); operator int (); };  
void f (long);      void f (char*);      void g (S);      void g (char*);  
void h (const S&);   void h (char*);  
void ex (S &a) { f (a); // f ((long) (a.operator int()));  
              g (1); /* g (S ((long) (1))); */      h (1); /* h (S ((long) (1))); */ }
```

Критерии: За пропуск какого-либо преобразования или ограничения (из 5): -2. За пропуск примера какой-либо операции: -4.

10. Ответ:

```
typedef list<int> L; typedef vector<L::value_type *> V;  
L::size_type g (V & vect, L & lst)  
{ L::size_type count = 0; L::reverse_iterator lp=lst.rbegin();  
  for (V::size_type i=0; i<vect.size() && lp!=lst.rend(); ++i, ++lp)  
    { L::value_type t = *lp; *lp = *vect [i]; * vect [i] = t; ++count; }  
  L::iterator lq = lst.begin ();  
  while (lq != lst.end ()) {cout << * lq; ++ lq; }  
  for (V::size_type i = 0; i < vect.size (); ++ i) cout << * vect [i];  
  return count;  
}
```

Критерии: За каждую существенную ошибку: -3. Отсутствие ссылки в списке формальных параметров -3. Если вместо индексации использованы указатели, но программа правильная – не снижать. Использование вместо типа *size_type* типа *int* – не снижать.

Коллоквиум, 2011 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: `x = c.C::g (& b);` `x = c.A::f ();` `x = c.B::f ();`
`x = c.A::f (x);` `x = c.B::f (x);`
`x = c.C::f (x, 1);` `x = p -> A::f ();`
`x = p -> A::f (x);` `x = p -> A::f (x, 1);`

Ошибочны 2 оператора, которые можно исправить (подчеркнуты).

Критерии: За пропуск ошибки (из 2-х): **-3**. За каждое не отмеченное или неверное уточнение области видимости (из 7): **-1**. За каждую лишнюю обнаруженную "ошибку" **-3**.

2. Ответ: Чтобы функцию `f()` можно было вызвать для констант (`t->f(3)`), надо поставить модификатор `const` после списка формальных параметров этой функции в классе `C`. Чтобы из функции `C::f()` можно было вызвать функцию `C::h()`, надо поставить модификатор `const` после списка формальных параметров функции `C::h()`. Чтобы функцию преобразования `operator int()` можно было вызывать для констант (`t->f(d)`), надо поставить модификатор `const` после списка формальных параметров этой функции в классе `D`. Чтобы функцию `g()` можно было вызывать для констант (`t->g()`), надо поставить модификатор `const` после списка формальных параметров этой функции в классе `C`. При таких исправлениях все функции станут не виртуальными, и будет выдано в поток:

`C::f,3` `C::h,3` `C::f,100` `C::h,100` `C::g` `C::h,5`

Если будут сделаны синхронные исправления описания функций `f()`, `g()` и `h()`, в классе `D`, чтобы функции остались виртуальными, будет выдано в поток:

`D::f,3` `D::h,3` `D::f,100` `D::h,100` `C::g` `D::h,5`

Критерии: За каждое неправильное исправление и неправильно вызванную функцию **-2**.

3. Ответ: `f ();` // (а) неоднозначность определения числа параметров
`f (4);` // (в) неоднозначность при стандартном преобразовании
`f (f);` // `f (double)`, (б) вещественное расширение
`f (d);` // `f (double)`, (а) точное отождествление
`f (ld);` // `f (long double)`, (а) точное отождествление

Критерии: За каждую ошибку **-2**.

4. Ответ: `A()`, `B()`, `A(const A&)`, `D()`, `A()`, `B()`, `C()`, `~C()`, `~B()`, `~A()`, `~D()`, `~A()`, `~B()`, `~A()`.

Критерии: За каждую ошибку **-2**.

5. Ответ:

```
template<class T> class ObjPtr { /* ... */ cl<T> * c; public: /* ... */
    const bool operator==(const ObjPtr<T> & s) const
    { return * c == * s.c; }
    const bool operator==(const T & s)
    { return * c == s; }
    friend const bool operator==(const T & s, const ObjPtr<T> & t);
};
template<class T> const bool operator==(const T & s, const ObjPtr<T> & t)
{ return * t.c == s; }
```

Критерии: За отсутствие каждого примера (из 3-х) или ошибки в нем: **-4**. За отсутствие реализаций методами класса (из 2-х) при правильной реализации этих же операций функциями-друзьями: **-2**.

6. Ответ: (а) Наследование есть способ упорядочения абстракций, путём создания иерархии классов. Наследование позволяет производному объекту наследовать от базового объекта общие атрибуты, определяя собственные дополнительные характеристики.

(б) Наследование может одиночным (единичным) и множественным, когда наследуются атрибуты сразу нескольких классов. Ограничений на длину цепочек в языке нет. Одиночное наследование:

```
class A: { ... }; class B: A { ... }; class C: B { ... };
```

(в) Множественное наследование:

```
class A: { ... }; class B: { ... }; class C: A, B { ... };
```

Критерии: За неверный ответ: **-5**. За отсутствие ответа на вопрос о видах наследования: **-5**. За отсутствие любого из двух примеров: **-2**.

7. Ответ: **d b e d e m e.**

Критерии: За каждую неверную (лишнюю или пропущенную) печать **-2**.

8. Ответ: (а) Необходимо заменить вид наследования класса "S" на открытый, чтобы поле "s" стало доступным в классе "U":

```
class T: public S { public int t; void tp (int ti) { t = ti; s = ti; };
```

(б) Необходимо дополнительно заменить вид наследования класса "T" на открытый, чтобы стали возможными преобразования указателей в процедуре "g()":

```
class U: public T { public int u; void up (int ui) { u=ui; t=ti; s=ti; };
```

(в) Открытости наследования можно добиться, заменяя в наследующем классе слово "class" словом "struct". **(г)** Других ошибок нет.

Критерии: За пропуск любой из двух ошибок или неправильное исправление: **-5**. За указание ошибки там, где ее нет: **-5**.

9. Ответ: Функция-друг, не являясь методом класса, имеет доступ к защищённым и закрытым членам класса. Функция может быть объявлена другом нескольких классов, а метод всегда относится к одному классу.

```
class Matrix;
class Vector { friend Vector operator*(const Matrix&, const Vector&); };
class Matrix { friend Vector operator*(const Matrix&, const Vector&); };
Vector operator*(const Matrix&, const Vector&) { ... }
```

Критерии: За пропуск объяснения понятия "дружбы": **-5**. За пропуск ответа на вопрос об отличиях методов и "друзей": **-3**. За пропуск примера: **-2**.

10. Ответ:

```
typedef list<long int*> L;
L::size_type g (L & lst)
{ L::reverse_iterator lp, lq; L::size_type count = 0; lp = lst.rbegin();
  while (lp != lst.rend())
  { lq = lp; ++ lp;
    if (lp == lst.rend()) break;
    if (** lp < 0) { ** lq = - ** lp; ++ count; }
    ++ lp;
  }
  L::iterator lr = lst.begin ();
  while (lr != lst.end ()) { cout << ** lr; ++ lr; }
  return count;
}
```

Критерии: За каждую существенную ошибку: **-3**. Отсутствие ссылки в списке формальных параметров **-3**. Использование вместо типа `size_type` типа `int` – не снижать.

Коллоквиум, 2011 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

```
1. Ответ:  x = c.g (p);
             x = c.A::f ();           // x = c.B::f ();
             x = c.C::f (x);           x = c.C::f (x, 1);
             x = p -> C::f ();
             x = p -> C::f (x);         x = p -> C::f (x, 1);
```

Ошибочен 1 оператор, который можно исправить (выделен шрифтом).

Критерии: За пропуск ошибки: **-4**. За каждое не отмеченное или неверное уточнение области видимости (из 7): **-1**. За каждую лишнюю обнаруженную "ошибку" **-4**.

```

2. Ответ: k = s; // Приведение производного указателя к базовому
// s = dynamic_cast<S*>(k); // Тип К - не полиморфный
// l = dynamic_cast<L*>(k); // Тип К - не полиморфный
m = s; // Приведение производного указателя к базовому
// p = dynamic_cast<S*>(l); // Тип L - не полиморфный
q = dynamic_cast<Q*>(m);
r = dynamic_cast<R*>(q);
// s = dynamic_cast<S*>(p); // Тип Р - не полиморфный

```

Критерии: За каждое не найденное лишнее преобразование (из двух): **-2**. За каждое не удаленное ошибочное преобразование (из 4-х): **-2**.

3. Ответ: $f(s); // f(\text{int}),$ (б) целочисленное расширение
 $f(i); // f(\text{int}),$ (а) точное отождествление
 $f(b); // f(\text{int}),$ (б) целочисленное расширение
 $f(f); //$ (в) неоднозначность при стандартном преобразовании
 $f(d); //$ (в) неоднозначность при стандартном преобразовании

Критерии: За каждую ошибку -2.

4. Ответ: $A() B() C() A() B() C() B(B\&) D() \sim D() \sim B() \sim C() \sim B() \sim A() \sim C() \sim B() \sim A()$.

Критерии: За каждую ошибку -2.

5. Ответ:

```
template<class T> class ObjPtr { /* ... */ cl<T> * c; public: /* ... */
    ObjectPtr<T> & operator= (const ObjectPtr<T> & s)
    { if (this != & s)
        { delete c; c = new cl<T>; * c = * (s.c); }
        return * this; }
    ObjectPtr<T> & operator= (const int & s)
    { delete c; c = new cl<T>; * c = s;
        return * this; }
    friend const ObjectPtr<T> operator- (const ObjectPtr<T> & s);
};

template<class T>
    const ObjectPtr<T> operator- (const ObjectPtr<T> & s) {
    ObjectPtr<T> v; v.c = new cl<T>; * v.c = (T) 0 - * s.c; return v;
}
```

Критерии: За отсутствие каждого примера (из 3-х) или ошибки в нем: **-4**. За отсутствие реализации унарного минуса функцией-другом при правильной реализации этой операции методом класса: **-2**. За попытку реализации присваивания функцией-другом: **-5**.

6. Ответ: (а) Инкапсуляция обеспечивает сокрытие состояния объекта от остальных объектов, процедур. Инкапсуляция защищает данные от несанкционированного доступа со стороны посторонних алгоритмов. Инкапсуляция изолирует внешнее представление об объекте от его внутренней реализации. Доступ к связанным между собой алгоритмам и данным контролируется интерфейсом (формальным описанием способов доступа к алгоритмам и данным). **(б)** Пример инкапсуляции (доступ к полю "n" возможен только из методов класса и функций-друзей):

```
class A: { int n; public: f () ... };
```

Критерии: За демонстрацию непонимания сущности инкапсуляции: -5. За отсутствие примера: -5.

7. Ответ: l a l f i k e n

Критерии: За каждую неверную (лишнюю или пропущенную) печать -2.

8. Ответ: (а) Чтобы преобразование из класса "W" в функции "h ()" в класс "Z" стало возможным, это преобразование надо исправить: pz = (Z *) (X *) (new W);

(б) Дополнительно необходимо сделать открытым вид наследования класса "W" классом "X" и класса "X" классом "Z":

```
class X: public W { ... }; class Z: public X,Y { ... };
```

(в) Необходимо уточнить область видимости поля "w" в теле функции "h ()": (*pz).X::w=& hi;

(г) Открытости наследования можно добиться, заменяя в наследующем классе слово "class" словом "struct". **(д)** Других ошибок нет.

Критерии: За пропуск любой из пяти ошибок или неправильное исправление: -2. За указание ошибки там, где ее нет: -5.

9. Ответ: (а) Имеется иерархия классов, хотя бы из двух классов – базового и производного **(б)** В базовом классе функция объявлена с ключевым словом **virtual** **(в)** В производном классе есть функция с таким же именем, с таким же списком параметров (количество, типы и порядок параметров совпадают) и с таким же типом возвращаемого значения. **(г)** Вызов функции производного класса осуществляется через указатель на объект базового класса (или с помощью ссылки на объект базового класса) без указания самого объекта и уточнения с помощью операции разрешения области видимости.

```
class B { public: virtual void print (); };
class P: public B { public: void print (); };
void B::print () { ... }
void P::print () { ... } B * ps = new P; ... ps -> print ();
```

Критерии: За пропуск любого условия включения механизма виртуальности: -3. За пропуск примера: -2.

10. Ответ:

```
typedef vector <float *> V;
V::size_type g (V & vect)
{ V::size_type count = 0; V::size_type vs = vect.size () - 1;
  for (int i = 0; i < vs; i += 2)
    { if (* vect [i + 1] < 0) { * vect [i] = 0; ++ count; } }
  for (int i = vs; i > 0; -- i) cout << * vect [i];
  return count;
}
```

Критерии: За каждую существенную ошибку: -3. Отсутствие ссылки в списке формальных параметров -3. Если вместо индексации использованы указатели, но программа правильная – не снижать. Использование вместо типа **size_type** типа **int** – не снижать.

Коллоквиум, 2011 год. Ответы и критерии оценок.
Вариант 4.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: ~~// x = c.C::g (&b);~~
~~x = c.B::f ();~~ x = c.C::f (x);
~~x = c.A::f (x, 1);~~ // ~~x = p -> f ();~~
~~// x = p -> f (x);~~
~~x = p -> A::f (x, 1);~~

Ошибочны 5 операторов, из которых только 2 можно исправить (подчеркнуты).

Критерии: За пропуск ошибки (из 5): **-3**. За каждое не отмеченное или неверное уточнение области видимости (из 4): **-2**. За каждую лишнюю обнаруженную "ошибку" **-3**.

2. Ответ: Ошибок нет.

Будет выдано в поток: U::g U::h, 1 U::f, 1 ::h, 2 ::f, 2 U::h
T::g ::h, 0 ::g U::h ::h U::g
U::h, 3 ::h, 4

Критерии: За каждую лишнюю ошибку или неправильно вызванную функцию **-2**.

3. Ответ: f (s); // f (X), (г) пользовательское преобразование
f (x); // f (X), (a) точное отождествление
f (b); // f (X), (г) пользовательское преобразование
f (f); // f (X), (г) пользовательское преобразование
f (y); // f (Y), (a) точное отождествление

Критерии: За каждую ошибку **-2**.

4. Ответ: A() B() A(const A&) B() A() C() D() ~D() ~C() ~A() ~B() ~A() ~B() ~A().

Критерии: За каждую ошибку **-2**.

5. Ответ:

```
template<class T> class ObjPtr { /* ... */ cl<T> * c; public: /* ... */
    const ObjPtr<T> & operator-- () { * c = * c - 1; return * this; }
    const ObjPtr<T> operator-- (int empty)
    { ObjPtr<T> v; v.c = new cl<T>; * c = * c - 1; return v; }
    friend const ObjectPtr<T> & operator--(ObjectPtr<T> & s);
    friend const ObjectPtr<T> operator--(ObjectPtr<T> & s, int empty);
template<class T> const ObjectPtr<T>&operator--(ObjectPtr<T> &s)
    { * s.c = * s.c - 1; return s; }
template<class T> const ObjectPtr<T> operator--(ObjectPtr<T> &s, int empty)
    { ObjectPtr<T> v; v.c = new cl<T>; * v.c = * s.c - 1; return v; }
```

Должны быть представлены функции из каждого набора.

Критерии: За отсутствие каждого примера (из 2-х) или ошибки в нем: **-5**. За отсутствие реализаций какого-либо вида при правильной реализации этих операций другим способом: **-4**.

6. Ответ: (а) Виртуальной функцией является функция-член класса, которая описана в этом классе с использованием слова "virtual". (б) Такие функции используются для организации выбора методов классов на этапе выполнения программы по текущему значению указателя (ссылки), используемому для обращения к ним. (в) Классы, содержащие виртуальные функции, называются полиморфными. Создавать объекты и описывать конструкторы полиморфных классов можно. Пример полиморфных классов (пример должен содержать, как минимум, 1 базовый класс и 1 производный):

```
class B{public:virtual void f();};class P:public B{public:void f();};
```

Критерии: За неверный ответ на вопрос о виртуальной функции: **-5**. За неверный ответ на вопрос о полиморфном классе: **-4**. За отсутствие примера: **-5**. За неверный ответ на вопрос об объектах и конструкторах: **-4**.

7. Ответ: d b e l f i k e n

Критерии: За каждую неверную (лишнюю или пропущенную) печать **-2**.

8. Ответ: (а) Необходимо сделать открытым наследование классом "O" членов класса "M", иначе ошибочным оказывается последний оператор функции "g()":

```
class O: public M { public: int t; int op (int i) { return t + i; };
```

(б) Необходимо уточнить область видимости поля "t" в теле метода "pp" класса "P", иначе возникает неоднозначность трактовки ("N::t" или "O::t"):

```
class P: N, O { public: int p; int pp (int i) { return N::t + i; };
```

(в) Необходимо явно указывать способ преобразования указателей с помощью операций преобразования типа, например, так: O * po = (O *) (new N); N * pn = (N *) (po);

(г) Открытости наследования можно добиться, заменяя в классе "O" слово "class" словом "struct". (д) Других ошибок нет.

Критерии: За пропуск любой из пяти ошибок или неправильное исправление: -2. За указание ошибки там, где ее нет: -5.

9. Ответ: (а) Статический полиморфизм (действует только на этапе компиляции и характеризуется использованием одноимённых функций с разными профилями)

Пример:

```
int abs(int j); long int abs(long int j); double abs(double x);  
int i = -4; long int l = -200000; double x = - 18;  
int k = abs (i); long int m = abs (l); double y = abs (x);
```

(б) Динамический полиморфизм (реализуется с помощью виртуальных функций и доступа к функциям по указателям, когда на этапе компиляции не удаётся определить точную вызываемую функцию, так как значение указателя в это время не известно)

Пример:

```
struct TData { int T; virtual void Print () { /* ... */ } } t;  
struct EData: T { int E; void Print () { /* ... */ } } e;  
void Data (TData & d){ /* ... */ d.Print () /* ... */  
Data (t); Data (e);
```

(в) Параметрический или типовый полиморфизм (связан с введением шаблонов как средства параметрического описания классов)

Пример:

```
template<class T> T max (T x, T y) { return x > y ? x : y; }  
max (1, 2); max ('a', 'a'); max<int> ('a', 100);  
max (2.5, 1.0); max<double> (2.5, 1.0);
```

Критерии: За пропуск любого вида полиморфизма: -4. Отсутствие любого примера: -2.

10. Ответ:

```
typedef vector<int> V; V::size_type g (V & vect)  
{ V::size_type count = 0;  
for (int i = vect.size () - 1; i > 0; -- i)  
{ V::value_type e_even, e_odd; e_even = vect [i]; -- i;  
e_odd = vect [i];  
if (e_even >= e_odd) continue;  
vect [i + 1] = e_odd; vect [i] = e_even; ++ count; }  
for (int i = 0; i < vect.size (); ++ i) cout << vect [i];  
return count;  
}
```

Критерии: За каждую существенную ошибку: -3. Отсутствие ссылки в списке формальных параметров -3. Если вместо индексации использованы указатели, но программа правильная – не снижать. Использование вместо типа size_type типа int – не снижать.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :
 $S \rightarrow AX CAB \mid ACAB \mid \varepsilon$
 $X \rightarrow AX CA \mid ACA$
 $AC \rightarrow CA$
 $C \rightarrow a$
 $aA \rightarrow aa$
 $aB \rightarrow ab$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы). **Ответ:**

(б) Каким из перечисленных классов принадлежит язык $L(G)$?

Ответ:

Класс $\mathfrak{L}$	$L(G) \in \mathfrak{L}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
языки типа 0	
регулярные языки	

(в) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$. **Ответ:**

2. Является ли грамматика $G = \langle \{a, d\}, \{A, B, C, S\}, P, S \rangle$, где

$P = \{ S \rightarrow AC \mid AB; A \rightarrow AC \mid \varepsilon; B \rightarrow Bd \mid dBB; C \rightarrow dCB \mid \varepsilon; \}$

(а) приведенной? (б) однозначной? Ответ обосновать.

3. По заданной регулярной грамматике G построить конечный автомат в виде ДС. Детерминирован ли автомат? Ответ обосновать. Если автомат недетерминированный, то с помощью соответствующего алгоритма преобразовать его в эквивалентный ДКА.

G :
 $S \rightarrow B \perp \mid A \perp$
 $B \rightarrow Bb \mid Ab \mid a$
 $A \rightarrow Ab \mid Ba \mid a$

4. Сформулировать критерий применимости метода рекурсивного спуска. Применим ли этот метод к данной КС-грамматике? Ответ обосновать.

$S \rightarrow AC \mid bB$

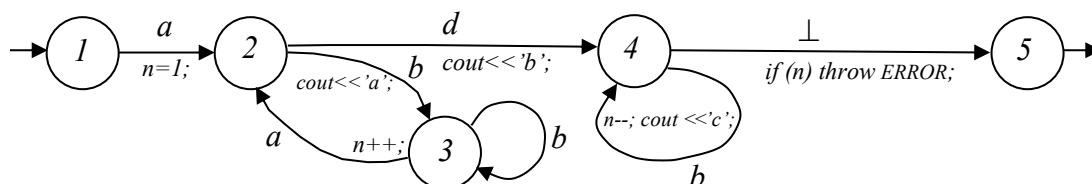
$A \rightarrow aA \mid \varepsilon$

$B \rightarrow b$

$C \rightarrow Bb \mid \varepsilon$

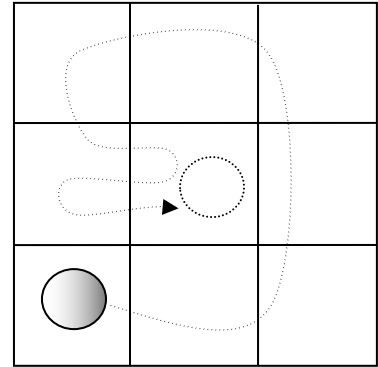
5. Дан автомат $\mathcal{A}$ в виде ДС с действиями. С помощью действий он допускает цепочки языка L_1 и переводит их в цепочки языка L_2 . Определить языки L_1 и L_2 . Построить КС-грамматику, анализируемую методом рекурсивного спуска, с действиями только вида $cout \ll \text{'символ'}$, задающую тот же перевод цепочек L_1 в цепочки L_2 .

$\mathcal{A}$:



6. Привести три примера контекстных условий, выполнение которых контролируется в современных языках программирования. **Ответ:**

7. На клетчатой поверхности размера 3x3 в левой нижней клетке расположен робот-шар. Он умеет выполнять команды перехода в соседнюю клетку: 'a' – переместиться на одну клетку вправо, 'b' – влево, 'c' – вверх, 'd' – вниз. С помощью последовательности команд можно задавать траекторию движения шара. Например, последовательность aaccbbdaba перемещает шар в центральную клетку. Последовательность команд является цепочкой в алфавите {a,b,c,d}. Если цепочка пуста, шар остается на месте. Опишите **регулярную** грамматику с терминальным алфавитом {a,b,c,d}, задающую все возможные траектории шара, удовлетворяющие условиям: шар начинает движение из левой нижней клетки, может двигаться только вокруг центральной клетки против часовой стрелки, не покидая поверхности; остановиться может только в левой нижней клетке. В грамматике должно быть не более 9 правил, считая альтернативы.



Ответ:

8. Является ли верификация частью отладки ? Ответ обосновать.

Ответ:

9. Перечислить этапы создания программного продукта, указав выходные данные каждого из них.

Ответ:

10. Дана постфиксная запись фрагмента программы, в котором нет операторов перехода goto и break, но есть один оператор continue. Вставить пропущенные в постфиксной записи команды перехода ('!', '!F') и недостающие метки; восстановить фрагмент на языке Си.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
x	0	<			y	0	!=			x	#+	;	y	-#	;	x	1	+	0	==
22	23	24	25	26	27	28	29	30	31	32	33	34								
		1		32		x	y	=	;											

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :
 $S \rightarrow AUCAD$
 $AUCA \rightarrow AAUCACA \mid \varepsilon$
 $AC \rightarrow CA$
 $C \rightarrow c$
 $cD \rightarrow cb$
 $cA \rightarrow cc$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы). **Ответ:**

(б) Каким из перечисленных классов принадлежит язык $L(G)$?

Ответ:

Класс $\mathfrak{L}$	$L(G) \in \mathfrak{L}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
языки типа 0	
регулярные языки	

(в) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$. **Ответ:**

2. Является ли грамматика $G = \langle \{a, b\}, \{A, B, D, S\}, P, S \rangle$, где

$$P = \{ S \rightarrow AB \mid Aa \mid aD; \quad A \rightarrow aAB \mid a; \quad B \rightarrow Bb \mid BbB; \quad D \rightarrow BD \mid a; \}$$

(а) приведенной? (б) однозначной? Ответ обосновать.

3. По заданной регулярной грамматике G построить конечный автомат в виде ДС. Детерминирован ли автомат? Ответ обосновать. Если автомат недетерминированный, то с помощью соответствующего алгоритма преобразовать его в эквивалентный ДКА.

G :
 $S \rightarrow C\perp \mid A\perp$
 $C \rightarrow Cd \mid Ad \mid a$
 $A \rightarrow Ad \mid Ca \mid a$

4. Сформулировать критерий применимости метода рекурсивного спуска. Применить ли этот метод к данной КС-грамматике? Ответ обосновать.

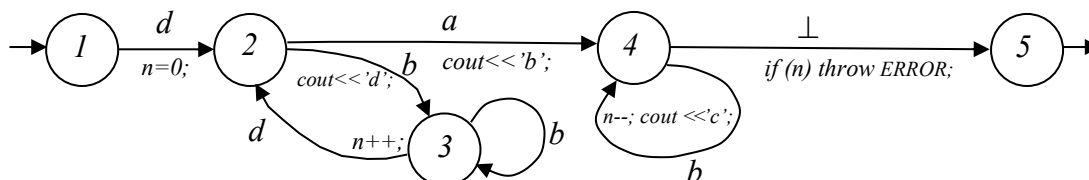
$$S \rightarrow AdS \mid B \mid \varepsilon$$

$$A \rightarrow aB \mid cAb$$

$$B \rightarrow bB \mid \varepsilon$$

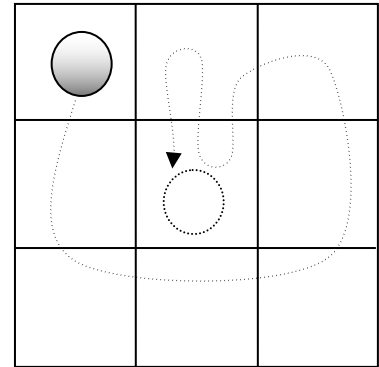
5. Дан автомат $\mathcal{A}$ в виде ДС с действиями. С помощью действий он допускает цепочки языка L_1 и переводит их в цепочки языка L_2 . Определить языки L_1 и L_2 . Построить КС-грамматику, анализируемую методом рекурсивного спуска, с действиями только вида $cout \ll \text{'символ'}$, задающую тот же перевод цепочек L_1 в цепочки L_2 .

$\mathcal{A}$:



6. Привести пример фрагмента программы (на языке Паскаль, Си или Си++), содержащей по одной ошибке каждого вида: лексическую, синтаксическую, семантическую. Пояснить каждую ошибку. **Ответ:**

7. На клетчатой поверхности размера 3x3 в левой верхней клетке расположен робот-шар. Он умеет выполнять команды перехода в соседнюю клетку: 'a' – переместиться на одну клетку вправо, 'b' – влево, 'c' – вверх, 'd' – вниз. С помощью последовательности команд можно задавать траекторию движения шара. Например, последовательность ddaaccbdcdd перемещает шар в центральную клетку. Последовательность команд является цепочкой в алфавите {a,b,c,d}. Если цепочка пуста, шар остается на месте. Опишите **регулярную** грамматику с терминальным алфавитом {a,b,c,d}, задающую все возможные траектории шара, удовлетворяющие условиям: шар начинает движение из левой верхней клетки, может двигаться только вокруг центральной клетки против часовой стрелки, не покидая поверхности; остановиться может только в левой верхней клетке.



В грамматике должно быть не более 9 правил, считая альтернативы. **Ответ:**

8. Является ли верификация видом тестирования? Ответ обосновать.

Ответ:

9. Перечислить методы оптимизации циклов. Привести один пример оптимизации.

Ответ:

10. Дана постфиксная запись фрагмента программы, в котором нет операторов перехода `goto` и `break`, но есть один оператор `continue`. Вставить пропущенные в постфиксной записи команды перехода ('!', 'F') и недостающие метки; восстановить фрагмент на языке Си.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
z	0	!=			i	1	=	;	i	N	<	40				i	#+	;		
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40		
x	i	*	y	>			x	2	==	;	17		z	-#	;					1
41	42																			

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow Ybba \mid \varepsilon$

$Y \rightarrow bZ \mid b$

$Z \rightarrow Ybb$

$Z \rightarrow Z$

$Y \rightarrow Y$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы). **Ответ:**

(б) Каким из перечисленных классов принадлежит язык $L(G)$?

Ответ:

Класс $\mathfrak{L}$	$L(G) \in \mathfrak{L}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
языки типа 0	
регулярные языки	

(в) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$. **Ответ:**

2. Является ли грамматика $G = \langle \{a, b\}, \{A, B, C, S\}, P, S \rangle$, где

$P = \{ S \rightarrow AbB \mid Aa; \quad A \rightarrow aAB \mid a; \quad B \rightarrow Bb \mid BB; \quad C \rightarrow bC \mid b; \}$

(а) приведенной? (б) однозначной? Ответ обосновать.

3. По заданной регулярной грамматике G построить конечный автомат в виде ДС. Детерминирован ли автомат? Ответ обосновать. Если автомат недетерминированный, то с помощью соответствующего алгоритма преобразовать его в эквивалентный ДКА.

G :

$S \rightarrow C\perp \mid B\perp$

$C \rightarrow Cd \mid Bd \mid c$

$B \rightarrow Bd \mid Cc \mid c$

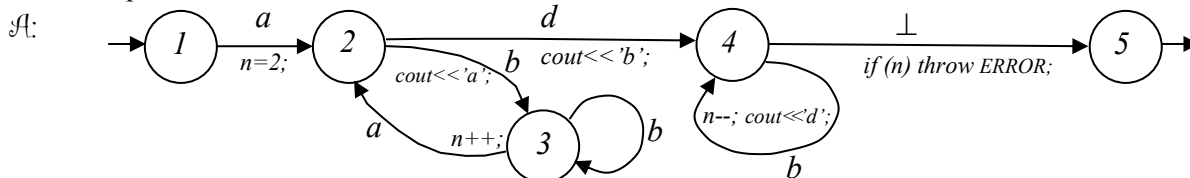
4. Сформулировать критерий применимости метода рекурсивного спуска. Применим ли этот метод к данной КС-грамматике? Ответ обосновать.

$S \rightarrow A \mid cB$

$A \rightarrow aB \mid bBa \mid \varepsilon$

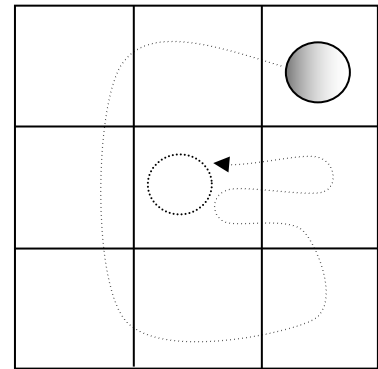
$B \rightarrow aA \mid b$

5. Дан автомат $\mathcal{A}$ в виде ДС с действиями. С помощью действий он допускает цепочки языка L_1 и переводит их в цепочки языка L_2 . Определить языки L_1 и L_2 . Построить КС-грамматику, анализируемую методом рекурсивного спуска, с действиями только вида $cout \ll \text{'символ'}$, задающую тот же перевод цепочек L_1 в цепочки L_2 .



6. Какие задачи решаются на этапе лексического анализа? **Ответ:**

7. На клетчатой поверхности размера 3x3 в правой верхней клетке расположен робот-шар. Он умеет выполнять команды перехода в соседнюю клетку: 'a' – переместиться на одну клетку вправо, 'b' – влево, 'c' – вверх, 'd' – вниз. С помощью последовательности команд можно задавать траекторию движения шара. Например, последовательность bddaacbab перемещает шар в центральную клетку. Последовательность команд является цепочкой в алфавите {a,b,c,d}. Если цепочка пуста, шар остается на месте. Опишите **регулярную** грамматику с терминальным алфавитом {a,b,c,d}, задающую все возможные траектории шара, удовлетворяющие условиям: шар начинает движение из правой верхней клетки, может двигаться только вокруг центральной клетки против часовой стрелки, не покидая поверхности; остановиться может только в правой верхней клетке. В грамматике должно быть не более 9 правил, считая альтернативы.



Ответ:

8. Является ли отладка частью тестирования? Ответ обосновать.

Ответ:

9. Перечислить аспекты машинно-зависимой оптимизации.

Ответ:

10. Дана постфиксная запись фрагмента программы, в котором нет операторов перехода `goto` и `continue`, но есть один оператор `break`. Вставить пропущенные в постфиксной записи команды перехода ('!', 'F') и недостающие метки; восстановить фрагмент на языке Си.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
i	1	=	;	i	N	<			17		i	#+	;			z	0	!=

20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
		i	2	%	1	!=			36	!	z	-#	;			12	}

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow Taad \mid \varepsilon$

$T \rightarrow Ua \mid a$

$U \rightarrow Taa$

$U \rightarrow U$

$T \rightarrow T$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы). **Ответ:**

(б) Каким из перечисленных классов принадлежит язык $L(G)$?

Ответ:

Класс $\mathfrak{L}$	$L(G) \in \mathfrak{L}$? (да/нет)
контекстно-свободные языки	
контекстно-зависимые языки	
языки типа 0	
регулярные языки	

(в) Классификация: найти такое целое k , что G является грамматикой типа k и не является грамматикой типа $k+1$. **Ответ:**

2. Является ли грамматика $G = \langle \{0,1\}, \{A,S\}, P, S \rangle$, где $P = \{ S \rightarrow A; A \rightarrow AA \}$

(а) приведенной? (б) однозначной? Ответ обосновать.

3. По заданной регулярной грамматике G построить конечный автомат в виде ДС. Детерминирован ли автомат? Ответ обосновать. Если автомат недетерминированный, то с помощью соответствующего алгоритма преобразовать его в эквивалентный ДКА.

G :

$S \rightarrow A\perp \mid B\perp$

$A \rightarrow Ad \mid Bd \mid a$

$B \rightarrow Bd \mid Aa \mid a$

4. Сформулировать критерий применимости метода рекурсивного спуска. Применить ли этот метод к данной КС-грамматике? Ответ обосновать.

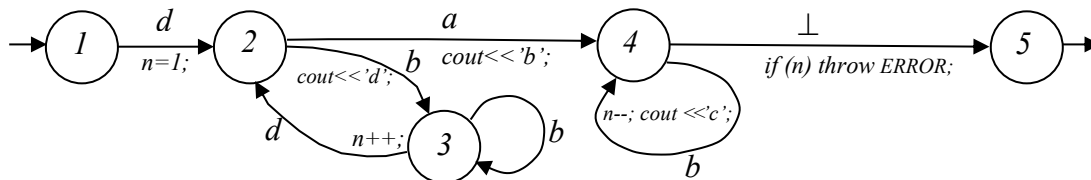
$S \rightarrow A \mid cB$

$A \rightarrow aB \mid bBa \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$

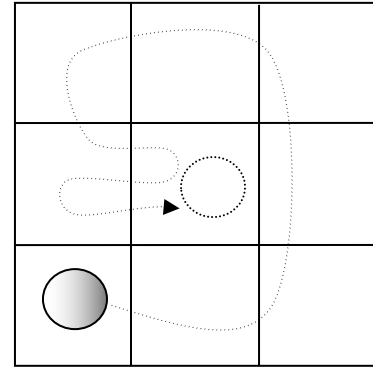
5. Дан автомат $\mathcal{A}$ в виде ДС с действиями. С помощью действий он допускает цепочки языка L_1 и переводит их в цепочки языка L_2 . Определить языки L_1 и L_2 . Построить КС-грамматику, анализируемую методом рекурсивного спуска, с действиями только вида $\text{cout} \ll \text{'символ'}$, задающую тот же перевод цепочек L_1 в цепочки L_2 .

$\mathcal{A}$:



6. Привести три способа внутреннего представления программ и проиллюстрировать каждый способ на примере оператора $a := a + b$. **Ответ:**

7. На клетчатой поверхности размера 3x3 в левой нижней клетке расположен робот-шар. Он умеет выполнять команды перехода в соседнюю клетку: 'a' – переместиться на одну клетку вправо, 'b' – влево, 'c' – вверх, 'd' – вниз. С помощью последовательности команд можно задавать траекторию движения шара. Например, последовательность aaccbbdaba перемещает шар в центральную клетку. Последовательность команд является цепочкой в алфавите {a,b,c,d}. Если цепочка пуста, шар остается на месте. Опишите **регулярную** грамматику с терминальным алфавитом {a,b,c,d}, задающую все возможные траектории шара, удовлетворяющие условиям: шар начинает движение из левой нижней клетки, может двигаться только вокруг центральной клетки против часовой стрелки, не покидая поверхности; остановиться может в любой клетке (кроме центральной). В грамматике должно быть не более 16 правил, считая альтернативы. **Ответ:**



8. Является ли тестирование видом верификации ? Ответ обосновать.

Ответ:

9. Перечислите основные функции отладчика в рамках интегрированной среды разработки программного обеспечения.

Ответ:

10. Дана постфиксная запись фрагмента программы, в котором нет операторов перехода **break** и **continue**, но есть один оператор **goto**. Вставить пропущенные в постфиксной записи команды перехода ('!', 'F') и недостающие метки; восстановить фрагмент на языке Си. ex вас

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
i	1	=	;	i	K	<			17		i	#+	;			j	1	=	;	j
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
L	<			33		j	#+	;			s	j	+=	;	s	N	>			48
43	44	45	46	47	48	49	50	51												
		!	12		s	N	==	;												

Экзамен, 2011 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{a^{3n}b \mid n \geq 1\} \cup \{\varepsilon\}$

(б) КС – "da" (-1), КЗ – "da" (-1), языки типа 0 – "da" (-1), регулярные – "da" (-3).

(в) $k = 1$ (есть правило $AC \rightarrow CA$, которое не является КС-правилом; правило $S \rightarrow \varepsilon$ единственное укорачивающее, и S не встречается в правых частях правил $\Rightarrow$ выполняются условия определения неукорачивающей грамматики)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: (а) **Нет.** Символы $\{a, d, B\}$ бесполезны, т.е. не участвуют в выводах цепочек языка. Для обоснования достаточно указать хотя бы один бесполезный символ. (б) **Нет.** Для единственной выводимой в грамматике G цепочки ε существуют бесконечно много левых выводов (и соответственно деревьев выводов). Для обоснования достаточно привести два различных левых (или правых) вывода (или два дерева вывода) цепочки ε :

$$\begin{aligned} S &\rightarrow AC \rightarrow C \rightarrow \varepsilon; \\ S &\rightarrow AC \rightarrow ACC \rightarrow CC \rightarrow C \rightarrow \varepsilon. \end{aligned}$$

Критерии: (а) ошибочные (отсутствующие) ответ или обоснование: -5, (б) ошибочные (отсутствующие) ответ или обоснование: -5

3. Ответ.

НКА:

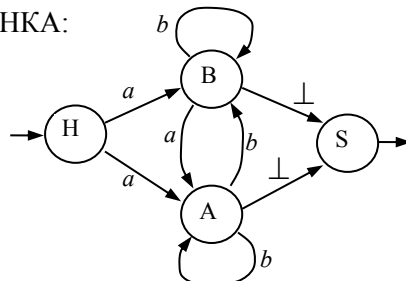
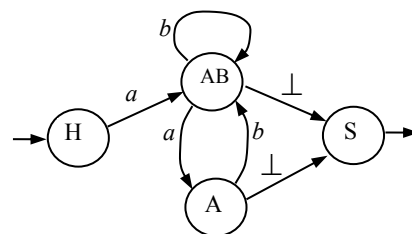


Таблица переходов ДКА:

	a	b	$\perp$
H	AB	$\emptyset$	$\emptyset$
AB	A	AB	S
A	$\emptyset$	AB	S
S	$\emptyset$	$\emptyset$	$\emptyset$

Диаграмма ДКА:



Допускается представить ДКА только в виде таблицы переходов (или набора команд δ) или только в виде диаграммы.

Критерии: Есть ошибки в НКА: -5, Нет обоснования или ошибочное: -3, Есть ошибки в преобразовании в ДКА: -5.

4. Ответ. Критерий применимости метода рекурсивного спуска. Пусть G — КС-грамматика. Метод рекурсивного спуска применим к G , если и только если для любой пары альтернатив $X \rightarrow \alpha \mid \beta$ выполняются следующие условия: (1) $first(\alpha) \cap first(\beta) = \emptyset$; (2) справедливо не более чем одно из двух соотношений: $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$; (3) если $\beta \Rightarrow \varepsilon$, то $first(\alpha) \cap follow(X) = \emptyset$. Для данной грамматики $first(AC) \cap first(bB) = \{b\} \neq \emptyset$. То есть, для пары правил $S \rightarrow AC$ и $S \rightarrow bB$ не выполняется пункт (1) критерия. Других пар, нарушающих Критерии, нет.

Критерии: Есть ошибки в формулировке критерия: -5, Есть ошибки в обосновании: -5.

5. Решение. Автомат $\mathcal{A}$ с действиями допускает цепочки вида $ay_1y_2\dots y_{n-1}db^n\perp$, где $n\geq 1$, $y_i = bx_ia$, $x_i \in \{b\}^*$ для $i = 1, \dots, n-1$, и каждую такую цепочку переводит соответственно в цепочку $a^{n-1}bc^n$. Грамматика с действиями такова (запись вида $\langle a \rangle$ означает $\langle cout \ll 'a' \rangle$):

$S \rightarrow aAb\langle c \rangle \perp$

$A \rightarrow bBa\langle a \rangle Ab\langle c \rangle \mid d\langle b \rangle$ (или $A \rightarrow b\langle a \rangle BaAb\langle c \rangle \mid d\langle b \rangle$)

$B \rightarrow bB \mid \varepsilon$

Возможный вариант ответа:

$S \rightarrow aAb\langle c \rangle \perp$

$A \rightarrow bBaAb\langle c \rangle \mid d\langle b \rangle$

$B \rightarrow bB \mid \varepsilon \langle a \rangle$

Вариант, основанный на рассмотрении цепочек L_1 как $ay_1y_2\dots y_ndbb^n\perp$, где $n\geq 0$, $y_i = bx_ia$, $x_i \in \{b\}^*$ для $i = 1, \dots, n$,

$S \rightarrow aA\perp$

$A \rightarrow bBaAb\langle c \rangle \mid db\langle b \rangle \langle c \rangle$

$B \rightarrow bB \mid \varepsilon \langle a \rangle$

Критерии: есть ошибки в описаниях языков L_1, L_2 : **-3**, грамматика для L_1 ошибочна: **-4**, метод рекурсивного спуска неприменим: **-2**, есть действие не вида $cout \ll 'символ'$: **-10**, конечное число цепочек переводятся неправильно: **-3**, бесконечно много цепочек переводятся неправильно: **-6**.

6. Привести три примера контекстных условий, выполнение которых контролируется в современных языках программирования.

Ответ: (какие-нибудь три подходящих примера:) **(1)** Имя, используемое в программе, должно быть описано. **(2)** Не допускаются два описания одного и того же имени в одной области видимости. **(3)** В операторе присваивания типы переменной и выражения должны быть совместимыми (совпадать). **(4)** В условном операторе и в операторе цикла в качестве условия возможно только логическое выражение (Паскаль, Оберон, Ада). **(5)** Операнды операции отношения должны быть определенного типа (например, одного из перечислимых типов в Паскале). **(6)** Количество фактических параметров в вызове подпрограммы должно соответствовать количеству формальных параметров в описании подпрограммы, а типы фактических параметров должны быть совместимы по присваиванию с типами соответствующих формальных параметров. **(7)** Возможны другие условия.

Критерии: за каждый пропущенный (или ошибочный) пункт: **-4**.

Экзамен, 2011 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{c^{3^n}b \mid n \geq 1\}$

(б) КС – "da" (-1), КЗ – "da" (-1), языки типа 0 – "da" (-1), регулярные – "da" (-3).

(в) $k = 0$ (есть укорачивающее правило $AUCA \rightarrow \varepsilon$, которое не удовлетворяет условиям для грамматики типа 1 (неукорачивающей)).

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. (а) Нет. Символы $\{b, B\}$ бесполезны, т.е. не участвуют в выводах цепочек языка. Для обоснования достаточно указать хотя бы один бесполезный символ. **(б) Нет.** Для единственной выводимой в грамматике G цепочки aa существуют два различных левых (правых) вывода (и соответственно, два различных дерева вывода):

$$S \rightarrow aD \rightarrow aa;$$

$$S \rightarrow Aa \rightarrow aa.$$

Критерии: (а) ошибочные (отсутствующие) ответ или обоснование: -5, (б) ошибочные (отсутствующие) ответ или обоснование: -5.

3. Ответ.



Допускается представить ДКА только в виде таблицы переходов (или набора команд δ) или только в виде диаграммы.

Критерии: Есть ошибки в НКА: -5, Нет обоснования или ошибочное: -3, Есть ошибки в преобразовании в ДКА: -5.

4. Ответ. Критерий применимости метода рекурсивного спуска. Пусть G — КС-грамматика. Метод рекурсивного спуска применим к G , если и только если для любой пары альтернатив $X \rightarrow \alpha \mid \beta$ выполняются следующие условия: **(1)** $first(\alpha) \cap first(\beta) = \emptyset$; **(2)** справедливо не более чем одно из двух соотношений: $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$; **(3)** если $\beta \Rightarrow \varepsilon$, то $first(\alpha) \cap follow(X) = \emptyset$. Для данной грамматики справедливы соотношения $\varepsilon \Rightarrow \varepsilon$ и $B \Rightarrow \varepsilon$, то есть для пары правил $S \rightarrow \varepsilon$ и $S \rightarrow B$ не выполняется пункт (2) критерия. Других пар, нарушающих Критерии, нет.

Критерии: Есть ошибки в формулировке критерия: -5, Есть ошибки в обосновании: -5.

5. Ответ. Автомат $\mathcal{A}$ с действиями допускает цепочки вида $dbx_1dbx_2\dots dx_ndab^n\perp$, где $n \geq 0$, $x_i \in \{b\}^*$ для $i = 1, \dots, n$, $dy_1y_2\dots y_nab^n\perp$, где $n \geq 0$, $y_i = bx_id$, $x_i \in \{b\}^*$ для $i = 1, \dots, n$, и каждую такую цепочку переводит соответственно в цепочку d^nbc^n . Грамматика с действиями такова (запись вида $\langle a \rangle$ означает $\langle cout \ll 'a' \rangle$):

$$S \rightarrow dA\perp$$

$$A \rightarrow bBd\langle d \rangle Ab\langle c \rangle \mid a\langle b \rangle \text{ (или } A \rightarrow b\langle d \rangle BdAb\langle c \rangle \mid a\langle b \rangle)$$

$$B \rightarrow bB \mid \varepsilon$$

$$S \rightarrow dA\perp$$

$$A \rightarrow bBdAb\langle c \rangle \mid a\langle b \rangle$$

$$B \rightarrow bB \mid \varepsilon \langle d \rangle$$

Критерии: есть ошибки в описаниях языков L_1, L_2 : -3, грамматика для L_1 ошибочна: -4, метод рекурсивного спуска неприменим: -2, есть действие не вида $cout \ll 'символ'$: -10, конечное число цепочек переводятся неправильно: -3, бесконечно много цепочек переводятся неправильно: -6.

6. Ответ:

```

program Errors;
var i: integer; t: bool;
begin read(i); t:= (i < >1) and (i#2); {лексическая: не существует лексемы #}
      begin t:=1 {семантическая: тип выражения в присваивании не соответствует типу переменной }
    end. {синтаксическая: не хватает парного end для begin}

```

Критерии: в качестве лексической предложена синтаксическая или семантическая ошибка, например, в описании a: integer, 'integer' является законным именем (лексемой), но именем, не описанным: **-2**, за остальные ошибки в примерах и пояснениях: **-4**.

7. Ответ. Согласно условиям, шар, начав с левой верхней клетки, дальше может двигаться вокруг

1	8	7
2		6
3	4	5

центральной клетки против часовой стрелки и может остановиться в левой верхней клетке. Все клетки (кроме центральной), начиная с левой верхней, нумеруются против часовой стрелки.

Строится конечный автомат с состояниями $S_1, S_2, \dots, S_8$. Состояние S_i соответствует положению шара в i -й клетке. Дуга из S_i в S_{i+1} помечается подходящей буквой из алфавита $\{a, b, c, d\}$. Например,

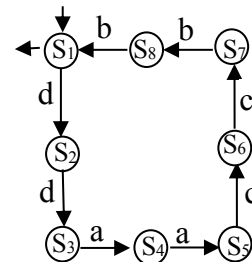
переход из S_1 в S_2 осуществляется по команде 'd', и соответствующая дуга помечается символом 'd'. Начальным является состояние S_1 . Оно же является и заключительным состоянием.

По конечному автомату строится праволинейная автоматная грамматика

с начальным символом S_1 : $S_1 \rightarrow dS_2 \mid \epsilon$ $S_2 \rightarrow dS_3$ $S_3 \rightarrow aS_4$... $S_8 \rightarrow bS_1$

Можно построить эквивалентную регулярную грамматику с начальным символом S , состоящую всего из двух правил: $S \rightarrow ddaaccbbS \mid \epsilon$

Критерии: не допускается ϵ : **-2**, за недопуск любой другой правильной цепочки или за допуск каждой лишней цепочки: **-5**. Нерегулярная грамматика: **-10**.



8. Ответ: Нет. *Верификация* – это процесс проверки программы на правильность. Кроме логических методов доказательства правильности программы существуют эвристические, основанные на тестировании. *Тестирование* – это процесс обнаружения дефектов (несоответствия программы исходным требованиям и спецификациям) путем проверки программы на данных с заранее известными ответами (тестах). То есть верификация более общее понятие, и тестирование является видом верификации.

Критерии: Нет обоснования, но ответ верно угадан: **1** балл, за отсутствие или ошибочность определения верификации: **-5**, за отсутствие или ошибочность определения тестирования: **-5**.

9. Перечислить методы оптимизации циклов. Привести один пример оптимизации.

Ответ: (примеры даются на каждый метод, для ответа достаточно одного примера)

1) вынесение инвариантных вычислений из тела цикла:

for (i=0; i<10; i++) A[i]=B*C*A[i]; $\Rightarrow$ D=B*C; for (i=0; i<10; i++) A[i]=D*A[i];

2) замена операций с переменными цикла:

(a) S = 10; for (i=0; i < N; i++) A[i] = i * S; // i – индуктивная переменная

$\Rightarrow$ S = 10; T = 0; for (i=0; i < 10; i++) T = T + S, A[i] = T;

(б) элиминация индуктивной переменной:

S = 10; for (i=0; i < N; i++) Q = Q + F(S), S = S + 10;

$\Rightarrow$ S = 10; M = S + N * 10; while (S <= M) {Q = Q + F(S); S = S + 10}

3) слияние, расщепление и развертывание циклов.

(a) for (i=0; i < N; i++) { S1 }

for (i=0; i < N; i++) { S2 }

$\Rightarrow$ for (i=0; i < N; i++) { S1; S2 }

(б) for (i=0; i < n; i++) if (x < y) { S1; } else { S2; } $\Rightarrow$

if (x < y) for (i=0; i < n; i++) { S1; } else for (i=0; i < n; i++) { S2; }

(в) for (i = 0; i < n; i++) { A[i]=B*C[i]; } $\Rightarrow$ for (i = 0; i < n; i+=2) { A[i]=B*C[i]; A[i+1]=B*C[i+1]; }

4) вложенные циклы по обработке массивов могут заменяться на одиночный цикл с соответствующим пересчетом индексов.

Критерии: За ошибочность/отсутствие каждого из трех пунктов: **-3**, за отсутствие примера или за ошибку в нем (достаточно одного примера): **-2**.

10. Ответ. Выделены пропущенные клетки.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
z	0	!=	42	!F	i	1	=	;	i	N	<	40	!F	22	!	i	#+	;	10	!
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39			
x	i	*	y	>	35	!F	x	2	==	;	17	!	z	-#	;	17	!			
40	41																			
1	!	}																		

```

while (z != 0)
    for (i = 1; i < N; i++) {
        if (x * i > y) {x -= 2; continue; }
        --z;
    }

```

План решения:

1) Расставляются знаки операций '!' и '!F', исходя из следующих соображений: если перед пропущенным местом стоит логическое выражение и нет операции ';', то переход условный, иначе – безусловный. 2) Расставляются метки, учитывая, что при трансляции управляющих конструкций языка Си: (а) переходы по лжи возможны только вперед; (б) безусловные переходы назад (кроме явных goto) возможны только в циклах.

Критерии: Есть ошибки в постфиксной записи: -5, есть ошибки в восстановленном фрагменте: -5, если без ошибок *только* расставлены операции перехода, оценка за задачу 2 балла.

Вариант ответа

```

while (z!=0) {
    for (i=1; i<N; i++) {
        if (x*i>y)    x-=2;
                    else    --z;
        } continue;
    }

```

формально не является правильным, но семантически эквивалентен. За такое решение снимается 1 балл.

Экзамен, 2011 год. Ответы и критерии оценок.

Вариант 3.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{b^{3n}a \mid n \geq 1\} \cup \{\varepsilon\}$

(б) КС – "da" (-1), КЗ – "da" (-1), языки типа 0 – "da" (-1), регулярные – "da" (-3).

(в) $k=2$ (есть и левостроительное, и правостроительное правила: $Z \rightarrow Ybb$ и $Y \rightarrow bZ$. Все правила удовлетворяют ограничениям грамматики типа 2)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: (а) Нет. Символы $\{b, B, C\}$ бесполезны, т.е. не участвуют в выводах цепочек языка. Для обоснования достаточно указать хотя бы один бесполезный символ. (б) Для единственной выводимой в грамматике G цепочки "aa" существует только один вывод (и, следовательно, единственное дерево вывода): $S \rightarrow Aa \rightarrow aa$.

Критерии: (а) ошибочные (отсутствующие) ответ или обоснование: -5, (б) ошибочные (отсутствующие) ответ или обоснование: -5

3. Ответ.

НКА:

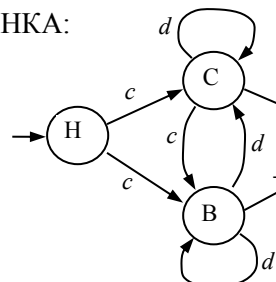
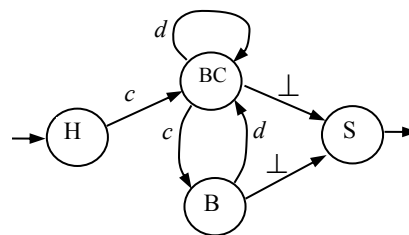


Таблица переходов ДКА:

	c	d	⊥
H	BC	∅	∅
BC	B	BC	S
B	∅	BC	S
S	∅	∅	∅

Диаграмма ДКА:



Допускается представить ДКА только в виде таблицы переходов (или набора команд δ) или только в виде диаграммы.

Критерии: Есть ошибки в НКА: -5, Нет обоснования или ошибочное: -3, Есть ошибки в преобразовании в ДКА: -5.

4. Ответ. Критерий применимости метода рекурсивного спуска. Пусть G — КС-грамматика. Метод рекурсивного спуска применим к G , если и только если для любой пары альтернатив $X \rightarrow \alpha \mid \beta$ выполняются следующие условия: (1) $first(\alpha) \cap first(\beta) = \emptyset$; (2) справедливо не более чем одно из двух соотношений: $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$; (3) если $\beta \Rightarrow \varepsilon$, то $first(\alpha) \cap follow(X) = \emptyset$. Для данной грамматики справедливы соотношения $first(aB) \cap follow(A) = \{a\} \neq \emptyset$, то есть для пары правил $A \rightarrow aB$ и $A \rightarrow \varepsilon$ не выполняется пункт (3) критерия. Других пар, нарушающих Критерии, нет.

Критерии: Есть ошибки в формулировке критерия: -5, Есть ошибки в обосновании: -5.

5. Решение. Автомат $\mathcal{A}$ с действиями допускает цепочки вида $ay_1y_2\dots y_{n-2}db^n\perp$, где $n \geq 2$, $y_i = bx_i a$, $x_i \in \{b\}^*$ для $i = 1, \dots, n-2$, и каждую такую цепочку переводит соответственно в цепочку $a^{n-2}bd^n$. Грамматика с действиями такова (запись вида $\langle a \rangle$ означает $\langle cout << 'a' \rangle$):

$S \rightarrow aAb\langle d \rangle b\langle d \rangle \perp$ $A \rightarrow bBa\langle a \rangle Ab\langle d \rangle \mid d\langle b \rangle$ (или $A \rightarrow b\langle a \rangle BaAb\langle d \rangle \mid d\langle b \rangle$) $B \rightarrow bB \mid \varepsilon$

Возможный вариант ответа: Вариант ответа, основанный на трактовке цепочек L_1 как $ay_1y_2\dots y_ndbbb^n$, $n \geq 0$, $y_i = bx_i a$, $x_i \in \{b\}^*$ для $i = 1, \dots, n$.

$S \rightarrow aAb\langle d \rangle b\langle d \rangle \perp$

$S \rightarrow aA\perp$

$A \rightarrow bBaAb\langle d \rangle \mid d\langle b \rangle$

$A \rightarrow bBaAb\langle d \rangle \mid d\langle b \rangle \langle d \rangle \langle d \rangle$

$B \rightarrow bB \mid \varepsilon \langle a \rangle$

$B \rightarrow bB \mid \varepsilon \langle a \rangle$

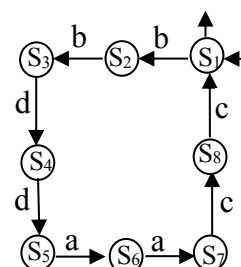
Критерии: есть ошибки в описаниях языков L_1, L_2 : -3, грамматика для L_1 ошибочна: -4, метод рекурсивного спуска неприменим: -2, есть действие не вида $cout << 'символ'$: -10, конечное число цепочек переводятся неправильно: -3, бесконечно много цепочек переводятся неправильно: -6.

6. Ответ: (1) Выделение в тексте цепочек символов, представляющих лексемы, и проверка правильности их записи. (2) Фиксация в специальных таблицах факта появления лексемы в тексте. (3) Преобразование цепочки, представляющей лексему, в пару $\langle \text{тип\_лексемы}, \text{указатель\_на\_информацию\_о\_ней} \rangle$. (4) Удаление пробельных символов и комментариев.

Критерии: за каждый неверный/не указанный пункт: -3.

7. Ответ. Согласно условиям, шар, начав с правой верхней клетки, дальше может двигаться вокруг центральной клетки против часовой стрелки и может остановиться в правой верхней клетке. Все клетки (кроме центральной), начиная с правой верхней, нумеруются против часовой стрелки. Строится конечный автомат с состояниями $S_1, S_2, \dots, S_8$.

Состояние S_i соответствует положению шара в i -й клетке. Дуга из S_i в S_{i+1} помечается подходящей буквой из алфавита $\{a, b, c, d\}$. Например, переход из S_1 в S_2 осуществляется по команде 'b', и соответствующая дуга помечается символом 'b'. Начальным является состояние S_1 . Оно же является и заключительным состоянием.



По конечному автомату строится праволинейная автоматная грамматика с начальным символом S_1 :

$$S_1 \rightarrow bS_2 \mid \varepsilon \quad S_2 \rightarrow bS_3 \quad S_3 \rightarrow dS_4 \quad \dots \quad S_8 \rightarrow cS_1$$

Можно построить эквивалентную регулярную грамматику с начальным символом S , состоящую всего из двух правил: $S \rightarrow b b d d a a c c S \mid \varepsilon$

Критерии: не допускается ε : -2, за недопуск любой другой правильной цепочки или за допуск каждой лишней цепочки: -5. Нерегулярная грамматика: -10.

8. Ответ: Нет. *Тестирование* – это процесс обнаружения дефектов (несоответствия программы исходным требованиям и спецификациям) путем проверки программы на данных с заранее известными ответами (тестах), а *отладка* – это поиск причин и устранение уже обнаруженных дефектов.

Критерии: Нет обоснования, но ответ верно угадан: 1 балл, за отсутствие или ошибочность определения отладки -5, за отсутствие или ошибочность определения тестирования: -5.

9. Ответ: (1) учет регистровой структуры вычислительной аппаратуры (оптимальное распределение регистров, передача параметров в процедуры и функции через регистры); (2) удаление лишних команд; (3) оптимизация потока управления и удаление недостижимых участков программ; (4) снижение "стоимости" программы (есть «дорогие» и «дешевые» команды с точки зрения времени их выполнения процессором); (5) использование «машинных идиом» (например: `хог ах, ах` для обнуления регистра); (6) слияние, дробление и развертывание циклов, иногда требующееся из-за технических особенностей аппаратуры; (7) учет векторных и конвейерных свойств архитектуры.

Критерии: За ошибочность/отсутствие каждого из пунктов 1-3: -3, за отсутствие примера или за ошибку в нем (достаточно одного примера): -2.

10. Ответ. Выделены пропущенные клетки.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
i	1	=	;	i	N	<	38	!F	17	!	i	#+	;	5	!	z	0	!=	36	!F
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37					
i	2	%	1	!=	31	!F	36	!	z	-#	;	17	!	12	!	}				

```

for (i = 1; i < N; i++)
    while (z != 0) {
        if (i % 2 != 1) break;
        --z;
    }

```

Другой правильный ответ:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
i	1	=	;	i	N	<	38	!F	17	!	i	#+	;	5	!	z	0	!=	31	!F
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37					
i	2	%	1	!=	31	!F	36	!	z	-#	;	38	!	12	!	}				

```

for (i = 1; i < N; i++)
    if (z != 0) && (i % 2 != 1);
    else {
        --z; break;
    }

```

План решения:

1) Расставляются знаки операций '!' и '!F', исходя из следующих соображений: если перед пропущенным местом стоит логическое выражение и нет операции ';', то переход условный, иначе – безусловный. 2) Расставляются метки, учитывая, что при трансляции управляющих конструкций языка Си: (а) переходы по лжи возможны только вперед; (б) безусловные переходы назад (кроме явных goto) возможны только в циклах.

Критерии: Есть ошибки в постфиксной записи: -5, есть ошибки в восстановленном фрагменте: -5, если без ошибок *только* расставлены операции перехода, оценка за задачу 2 балла.

Вариант ответа

```

for (i=1; i < N; i++)
    while (z != 0) {
        if (i % 2!=1) else --z;
    }

```

формально не является правильным, но семантически эквивалентен. За такое решение снимается 1 балл.

Экзамен, 2011 год. Ответы и критерии оценок.

Вариант 4.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{a^{3^n}d \mid n \geq 1\} \cup \{\varepsilon\}$

(б) КС – "da" (-1), КЗ – "da" (-1), языки типа 0 – "da" (-1), регулярные – "da" (-3).

(в) $k = 3$ (все правила левостолбчатые)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: (а) Нет. Все символы грамматики бесполезны, т.е. недостижимы или бесплодны. Грамматика порождает пустой язык: $L(G) = \emptyset$. Для обоснования достаточно указать хотя бы один бесполезный символ. (б) Да. Не существует цепочки, выводимой в грамматике G , имеющей два различных дерева вывода, так как язык пуст.

Критерии: (а) ошибочные ответ или обоснование на первый вопрос: -5, (б) ошибочные ответ или обоснование на второй вопрос: -5.

3. Ответ.



Допускается представить ДКА только в виде таблицы переходов (или набора команд δ) или только в виде диаграммы.

Критерии: Есть ошибки в НКА: -5, Нет обоснования или ошибочное: -3, Есть ошибки в преобразовании в ДКА: -5.

4. Ответ. Критерий применимости метода рекурсивного спуска. Пусть G — КС-грамматика. Метод рекурсивного спуска применим к G , если и только если для любой пары альтернатив $X \rightarrow \alpha \mid \beta$ выполняются следующие условия: (1) $first(\alpha) \cap first(\beta) = \emptyset$; (2) справедливо не более чем одно из двух соотношений: $\alpha \Rightarrow \varepsilon, \beta \Rightarrow \varepsilon$; (3) если $\beta \Rightarrow \varepsilon$, то $first(\alpha) \cap follow(X) = \emptyset$. Для данной грамматики вычисляются необходимые для проверки критерия множества:

$$first(A) = \{a, b\}, \quad first(cB) = \{c\}, \quad first(aB) = \{a\}, \quad first(bBa) = \{b\}, \quad first(bB) = \{b\}, \quad first(\varepsilon) = \emptyset, \\ follow(A) = \emptyset, \quad follow(B) = \{a\}.$$

Пересечение множеств $first$ и $follow$ для соответствующих нетерминалов пусто, поэтому выполняются пункты (1) и (3) критерия применимости. Кроме того, у каждого нетерминала не более одной альтернативы, из которой выводится ε , поэтому выполняется пункт (2).

Критерии: Есть ошибки в формулировке критерия: -5, Есть ошибки в обосновании: -5.

5. Автомат $\mathcal{A}$ с действиями допускает цепочки вида $dy_1y_2\dots y_nab^{n+1}\perp$, где $n \geq 0$, $y_i = bx_id$, $x_i \in \{b\}^*$ для $i = 1, \dots, n$, и каждую такую цепочку переводит соответственно в цепочку d^nbc^{n+1} . Запись вида $\langle a \rangle$ означает $\langle cout \ll 'a' \rangle$:

Грамматика с действиями

$$S \rightarrow dAb\langle c \rangle \perp$$

$$A \rightarrow bBd\langle d \rangle Ab\langle c \rangle \mid a\langle b \rangle \quad (\text{или } A \rightarrow b\langle d \rangle BdAb\langle c \rangle \mid a\langle b \rangle)$$

$$B \rightarrow bB \mid \varepsilon$$

Возможный вариант ответа:

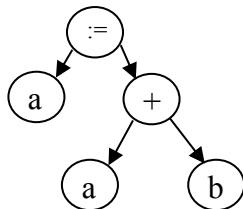
$$S \rightarrow aAb\langle c \rangle \perp$$

$$A \rightarrow bBaAb\langle c \rangle \mid a\langle b \rangle$$

$$B \rightarrow bB \mid \varepsilon \langle d \rangle$$

Критерии: есть ошибки в описаниях языков L_1, L_2 : -3, грамматика для L_1 ошибочна: -4, метод рекурсивного спуска неприменим: -2, есть действие не вида $cout \ll 'символ'$: -10, конечное число цепочек переводятся неправильно: -3, бесконечно много цепочек переводятся неправильно: -6.

6. **Ответ:** (1) Постфиксная запись (ПОЛИЗ): a, a, b +, :=; (2) Префиксная запись: :=, a, +, a, b;
 (3) Многоадресный код с явно именуемым результатом (тетрады):
 1) + a b T1
 2) := T1 ? a (вторая команда в этом примере может быть удалена при оптимизации)
 (4) Многоадресный код с неявно именуемым результатом (триады):
 1) + a b
 2) := a ^1
 (5) Динамические структуры, представляющие синтаксическое дерево
 Пример – дерево операций:



Допустимы и другие внутренние представления, например язык ассемблера, байт-код и т.п.

Критерии: за каждый ошибочный/отсутствующий пункт (из трех): (-4), если отсутствует только пример: -1.

7. **Ответ.** Согласно условиям, шар, начав с левой нижней клетки, дальше может двигаться вокруг центральной клетки против часовой стрелки и может остановиться в любой момент. Все клетки (кроме центральной), начиная с правой верхней нумеруются против часовой стрелки.

7	6	5
8		4
1	2	3

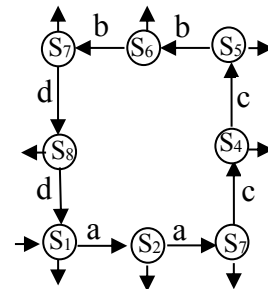
Строится конечный автомат с состояниями $S_1, S_2, \dots, S_8$. Состояние S_i соответствует положению шара в i -й клетке. Дуга из S_i в S_{i+1} помечается подходящей буквой из алфавита $\{a, b, c, d\}$.

Например, переход из S_1 в S_2 осуществляется по команде 'b', и соответствующая дуга помечается символом 'b'. Начальным является состояние S_1 . Оно же является и заключительным состоянием. По конечному автомату строится праволинейная автоматная грамматика с начальным символом

$S_1: \quad S_1 \rightarrow aS_2 \mid \varepsilon \quad S_2 \rightarrow aS_3 \mid \varepsilon \quad S_3 \rightarrow cS_4 \mid \varepsilon \quad \dots \quad S_8 \rightarrow dS_1 \mid \varepsilon$

Можно построить эквивалентную регулярную грамматику с начальным символом S, состоящую всего из 10 правил:

$S \rightarrow aacbbddS \mid \varepsilon \mid a \mid aa \mid aac \mid aacc \mid aacsb \mid aacbb \mid aacbbd \mid aacbbdd$



Критерии: не допускается ε : -2, за недопуск любой другой правильной цепочки или за допуск каждой лишней цепочки: -5. Нерегулярная грамматика: -10.

8. **Ответ:** Да. *Верификация* — это процесс проверки программы на правильность. Кроме логических методов доказательства правильности программы существуют эвристические, основанные на тестировании. *Тестирование* — это процесс обнаружения дефектов (несоответствия программы исходным требованиям и спецификациям) путем проверки программы на данных с заранее известными ответами (тестах)

Критерии: Нет обоснования, но ответ верно угадан: 1 балл, за отсутствие или ошибочность определения верификации: -5, за отсутствие или ошибочность определения тестирования: -5.

9. Ответ:

(1) пошаговое выполнение программы (шаг = строка; с трассировкой внутри вызываемой функции и без нее); (2) выполнение программы до строки, в которой в редакторе стоит курсор; (3) выделение выполняемой в данный момент строки; (4) приостановка выполнения программы: (а) можно запросить значение переменной, (б) можно заказать вычисление некоторого выражения, (в) можно изменить значение переменной и продолжить выполнение программы (!но редко какой отладчик позволяет изменять программный код, т.е. поддерживает частичную перекомпиляцию); (5) расстановка/снятие точек останова, которые визуализируются в текстовом редакторе; (6) выдача всей информации в терминах исходной программы.

Критерии: достаточно привести пять из шести пунктов, за ошибку/отсутствие для каждого из пяти пунктов: -2.

10. Ответ. Выделены пропущенные клетки).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
i	1	=	;	i	K	<	48	!F	17	!	i	#+	;	5	!	j	1	=	;	
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
j	L	<	46	!F	31	!	j	#+	;	17	!	s	j	+=	;	s	N	>	44	!F
42	43	44	45	46	47	48	49	50	51											
44	!	28	!	12	!	s	N	==	;	}										

```
for (i = 1; i < K; i++)
    for (j = 1; j < L; j++) {
        s += j;
        if (s > N) goto cont;
    }
cont: s -= N;
```

План решения:

1) Расставляются знаки операций '!' и '!F', исходя из следующих соображений: если перед пропущенным местом стоит логическое выражение и нет операции ';', то переход условный, иначе – безусловный. 2) Расставляются метки, учитывая, что при трансляции управляющих конструкций языка Си: (а) переходы по лжи возможны только вперед; (б) безусловные переходы назад (кроме явных goto) возможны только в циклах.

Критерии: Есть ошибки в постфиксной записи: -5, есть ошибки в восстановленном фрагменте: -5, если без ошибок *только* расставлены операции перехода, оценка за задачу 2 балла.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Что такое АТД? Каким образом АТД реализуется в C++?
2. Описать класс А так, чтобы: (1) все конструкции функции `main()` были верными, (2) явно в классе А было описано не более одного конструктора, (3) на экран выдалось:

15 60 7.

Нельзя использовать исключения и любые функции досрочного завершения программы.

```
int main () { A a1(5), a2 = 4, a3;
    a2 *= a1 *= 3;
    cout << a1.get() << ' ' << a2.get() << ' ' << a3.get() << endl;
    return 0;
}
```

3. Что будет выдано на печать при работе следующей программы?

```
struct S { int x;
    S (int n) { x = n; printf (" Cons "); }
    S (const S & a) { x = a.x; printf (" Copy "); }
    ~S () { printf ("Des "); }
};
S f (S y) { y = S(3); return y; }
int main () { S s (1);
    f (s);
    printf ("%d ", s.x);
    return 0;
}
```

4. Внести добавления в описания заданных методов (не меняя вывод на экран!) структур В и D так, чтобы все конструкции `main()` были правильными, а на печать выдалось **5535324242.**

```
struct B { float x;
    B (float a) { x = a; cout << 5; }
    B () { cout << 2; }
};
struct D : B {
    D () { cout << 3; }
    ~D () { cout << 4; }
};
int main () {
    B * p1 = new B (1), * p2 = new D[2];
    delete p1;
    delete [ ] p2;
    return 0;
}
```

5. Добавить (если нужно) в класс А сл.слова «`const`», так, чтобы заданный фрагмент программы был верным.

```
class A {
    int i;
public:
    A (int x) { i = x; }
    A (A & y) { i = y.i; }
    const A f (const A & z) { cout << endl; return *this; }
};
const A t1 () {
    const A a = 5;
    return a.f ( a );
}
```

6. Написать: (1) функцию-шаблон (от одного параметра-типа: контейнера STL целых чисел), которая для любого последовательного контейнера STL распечатывает его предпоследний элемент, если таковой имеется, (2) а также функцию `main()`, которая формирует контейнер-список из 5 целых чисел и применяет к нему написанную функцию-шаблон.

7. Что напечатает следующая программа?

```
struct B {      virtual void f(int n)  { cout<<" f(int) from B\n"; }
                static int i;
};
struct D: B {   virtual void f(char n) { cout<<"f(char) from D\n"; }
};
int B::i = 1;
int main ( ) {   D d;      B b1, b2, * pb = & d;
    pb -> f('a');
    b1.i += 2;      b2.i += 3;      d.i += 4;
    cout << b1.i <<' ' << b2.i <<' ' << d.i <<' ' << B::i << endl;
    return 0;
}
```

8. Что напечатает следующая программа?

```
struct S {
    S (int a) { try { if (a > 0) throw *this;
                  else if (a < 0) throw 0;
                }
            catch (S &) { cout << "SCatch_S&\n"; }
            catch (int) { throw; }
            cout << "SConstr\n";
        }
    S (const S & a) { cout << "Copy\n"; }
    ~S () { cout << "Destr\n"; }
};
int main () {      try { S s1(1), s2(-2);
                        cout << "Main\n";
                    }
            catch (S &) { cout << "MainCatch_S&\n"; }
            catch ( ... ) { cout << "MainCatch_...\n"; }
            return 0;
}
```

9. Добавить в функцию f9() использование механизма RTTI так, чтобы ее выполнение всегда завершалось нормально.

```
struct B { virtual void g() {} };
struct D: B { char y [100]; };
void f9 (B & b, D & d, int n) {
    D * pd = ( n > 0 ) ? & d : (D*) & b;
    strcpy (pd -> y, "one_variant\n");
}
```

10. Назвать три разных ситуации, когда автоматически вызывается деструктор.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Какие способы наследования базового класса производным классом приняты в C++? В чем отличие статуса закрытых (`private`) и открытых (`public`) членов базового класса в производном классе при закрытом (`private`) наследовании?
2. Описать класс В так, чтобы:
 - все конструкции функции `main()` были верными,
 - класс В содержал только один явно описанный конструктор,
 - на экран выдалось **17 11 6.**

Нельзя использовать исключения и любые функции досрочного завершения программы.

```
int main () { B b1(1), b2(2, 3), b3(b1);
    b1 += b2 += b3;
    cout << b1.get() << ' ' << b2.get() << ' ' << b3.get() << endl;
    return 0;
}
```

3. Что будет выдано на печать при работе следующей программы?

```
#include <iostream>
struct S { int x;
    S (int n) { x = n; printf (" Cons "); }
    S (const S & a) { x = a.x; printf (" Copy "); }
    ~S () { printf ("Des "); }
};
S f (S & y) { y = S(3); return y; }
int main () { S s (1);
    f (s);
    printf ("%d ", s.x);
    return 0;
}
```

4. Внести добавления в описания заданных методов (не меняя вывод на экран!) структур В и D так, чтобы все конструкции `main()` были правильными, а на печать выдалось **11163343.**

```
struct B { int x;
    B (int a) { x = a; cout << 1; }
    ~B () { cout << 3; }
};
struct D : B {
    D (int d): B (d) { cout << 6; }
    ~D () { cout << 4; }
};
int main () { B * p1 = new B [2], * p2 = new D (1);
    delete [ ] p1;
    delete p2;
    return 0;
}
```

5. Добавить (если нужно) **в класс C1** служебные слова «`const`», так, чтобы заданный фрагмент программы был верным.

```
class C1 { int i;
public: C1 (int x) { i = x; }
    C1 (C1 & y) { i = y.i; }
    const C1 f (C1 &c) const { cout << c.i << endl; return *this; }
};
const C1 t1 (const C1 a) { C1 b = C1 (5); return b.f (a);
}
```

6. Написать:

1) функцию-шаблон (от одного параметра-типа: контейнера STL целых чисел), которая для любого последовательного контейнера STL распечатывает сумму его трех последних элементов, если таковые имеются,

2) а также функцию *main()*, которая формирует контейнер-вектор из 5 целых чисел и применяет к нему написанную функцию-шаблон.

7. Что напечатает следующая программа?

```
struct K { virtual void add_st(K*n){st++;cout<<"add_st (K*) from K\n";}
          static int st;
};
struct L:K{virtual void add_st(L*a){ st++;cout<<"add_st (L*) from L\n";}
};
int K::st = 2;
int main () { L ob, ob2; K k , * pl = & ob;
pl -> add_st (& ob2);
k.st ++; ++ ob.st ;
cout << k.st << ' ' << ob.st << ' ' << K::st << endl; return 0;
}
```

8. Что напечатает следующая программа?

```
struct S {
    S ( int a) { try { if (a > 0) throw *this;
                    else if (a < 0) throw 0;
                }
                catch (S &) { cout << "SCatch_S&\n"; }
                catch (int) { throw; }
                cout << "SConstr\n";
            }
    S (const S & a) { cout << "Copy\n"; }
    ~S () { cout << "Destr\n"; }
};
int main () { try { S s1(0), s2(5);
                  cout << "Main\n";
                }
                catch (S &) { cout << "MainCatch_S&\n"; }
                catch ( ... ) { cout << "MainCatch_...\n"; }
                return 0;
}
```

9. Добавить в функцию *putnull()* использование механизма RTTI так, чтобы ее выполнение всегда завершалось нормально.

```
struct B { virtual void empty () {} };
struct D: B { int mas [30]; };
void putnull (* pb) { D d , *pd = (D*)a pb;
    if (! pb) return;
    for (int i = 0; i < 30; i++) pd -> mas[i] = 0 ;
}
```

10. Назвать три разных ситуации, когда **автоматически** вызывается конструктор копирования.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

- Какие виды полиморфизма реализованы в C++, с помощью каких механизмов? Когда происходит выбор конкретной языковой конструкции для каждого вида полиморфизма?
- Описать класс C так, чтобы:
 - в `main ()` ошибочным было только описание объекта `c2`,
 - класс C содержал только один явно описанный конструктор,
 - после удаления описания `c2` на экран выдалось **14 56**.
 Нельзя использовать исключения и любые функции досрочного завершения программы.

```
int main () { C c1 (7), c2 = 5, c3 (c1 + c1);
    cout << c1.get() << ' ' << c3.get() << endl;
    return 0;
}
```

- Что будет выдано на печать при работе следующей программы?

```
struct S { int x;
    S (int n) { x = n; printf ( " Cons " ); }
    S (const S & a) { x = a.x; printf ( " Copy " ); }
    ~S () { printf ( "Des " ); }
};
S & f (S y, S & z) { y = S (3); return z; }
int main () { S s(1);
    f (s, s);
    printf ("%d ", s.x );
    return 0;
}
```

- Внести добавления в описания заданных методов (не меняя вывод на экран!) структур B и D так, чтобы все конструкции `main ()` были правильными, а на печать выдалось **776898**.

```
struct B { int x;
    B () { x = 7; cout << 7; }
    ~B () { cout << 8; }
};
struct D : B {
    D (int d) { x = d; cout << 6; }
    ~D () { cout << 9; }
};
int main ( ) {
    B * p1 = new B [1], * p2 = new D [1];
    delete [ ] p1;
    delete [ ] p2;
    return 0;
}
```

- Добавить (если нужно) **в класс Cls** служебные слова «`const`», так, чтобы заданный фрагмент программы был верным.

```
class Cls { int i;
public: Cls (int x) { i = x; }
    Cls (Cls & y) { i = y.i; }
    const Cls f (const Cls c){ cout << c. i << endl; return *this; }
};
const Cls t1 (const Cls * a) { Cls b = Cls (3); return a -> f ( b );
}
```

6. Написать:

1) функцию-шаблон (от одного параметра-типа: контейнера STL целых чисел), которая для любого последовательного контейнера STL распечатывает каждый второй элемент, начиная с конца,

2) а также функцию `main()`, которая формирует контейнер-список из 5 целых чисел и применяет к нему написанную функцию-шаблон.

7. Что напечатает следующая программа?

```
struct S { static double d;
          virtual S & g () { cout << "g ( ) from S\n"; }
};
struct T: S { virtual T & g () { cout << "g ( ) from T\n"; }
};
double S::d = 1.5;
int main () { T t; S s , * ps = &t;
             ps -> g ( );
             s.d = 5; t.d = 7;
             cout << s.d << ' ' << t.d << ' ' << S::d << endl; return 0;
}
```

8. Что напечатает следующая программа?

```
struct S {
    S (int a) { try { if (a > 0) throw *this;
                    else if (a < 0) throw 0;
                }
              catch (S &) { cout << "SCatch_S&\n"; throw; }
              catch (int) { cout << "SCatch_int\n"; }
                      cout << "SConstr\n";
    }
    S (const S & a) { cout << "Copy\n"; }
    ~S () { cout << "Destr\n"; }
};
int main () { try { S s1(-3), s2(25);
                  cout << "Main\n";
                }
              catch (S &) { cout << "MainCatch_S&\n"; }
              catch ( ... ) { cout << "MainCatch_...\n"; }
              return 0;
}
```

9. Добавить в функцию `puthi()` использование механизма RTTI, так, чтобы ее выполнение всегда завершалось нормально.

```
struct B { virtual void hi () { cout << "Hi!\n" } };
struct D: B { char txt [10] [4]; };
void puthi (B * pb, D * pd) { D d; int i = 10;
    if (! pb) return;
    pd = (D*) pb;
    while (i) strcpy ((pd -> txt) [-- i], "Hi!");
}
```

10. Назвать три разных ситуации, когда **автоматически** вызывается обычный конструктор (не копирования).

Коллоквиум, 2012 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. АТД – тип данных с полностью скрытой (инкапсулированной) структурой, работа с переменными такого типа происходит только через специальные, предназначенные для этого функции. В C++ АТД реализуются с помощью классов (структур), **все члены-данные** которых расположены в **закрытой** области.

Критерии: Неверное понятие АТД **-5**, неверный ответ о реализации АТД в C++ **-5**, за отсутствие слов о работе с переменными АТД – не снижать.

2. Ответ.

```
class A { int x;
public: A ( int y = 7) { x = y; }
      int get () { return x; }
      int operator *= (int y) { return x *= y; }
};
```

Критерии: За каждый недостающий/неверный метод/лишний конструктор **-4**, за каждую мелкую ошибку реализации **-2**.

3. Ответ. Cons Copy Cons Des Copy Des Des 1 Des

Критерии: За каждую неверную/лишнюю/недостающую печать **-4**, если пропущен конструктор, то за парный ему деструктор не снижать.

4. Ответ. (1) перед деструктором базового класса добавить **virtual**. **2)** добавить либо **B (float a = 0)**, либо **D () : B (0) { ... }**

Критерии: За каждое не сделанное добавление **-5**.

5. Ответ.

```
class A { int i;
public: A ( int x) { i = x; }
      A (const A & y) { i = y.i; }
      const A f (const A & z) const { cout << i << endl; }
};
const A t1 () { const A a = 5; return a.f (a); }
```

Критерии: За каждое не найденную или лишнюю ошибку **-5**.

6. Ответ.

```
template < class T > void f (const T & t) {
    if (t.size () < 2) return;
    typename T::const_reverse_iterator p = t.rbegin ();
    cout << *(++ p) << endl; // можно без скобок
}
int main () { list <int > lst;
    for (int i = 0; i < 5; i++)
        lst.push_back (i);
    f (lst); // или f <list <int> > (lst);
    return 0;
}
```

Критерии: Пропуск **typename** **-4**, если параметр функции **const**, а итератор **const** не имеет, снижение на 4 балла, за каждую другую ошибку – **3**. Отсутствие **&** в заголовке функции **-1**.

7. Ответ.

```
f(int) from B
10 10 10 10
```

Критерии: За неверный вызов функции **-5**, за любую ошибку со статической переменной **-5**.

8. Ответ: Copy → SCatchS& → Destr → SConstr → Destr → MainCatch...

Критерии: За отсутствие Copy (и его Destr) -5, за каждую другую ошибку -2.

9. Ответ:

```
void f9 (B & b, D & d, int n) {  
    D * pd = (n > 0) ? & d : (D*) & b;  
    if (typeid (*pd) == typeid (d)) // возможен вариант с dynamic_cast  
        strcpy (pd -> y, "one_variant\n");  
}
```

Критерии: Правильно добавлены средства RTTI – 10, иначе 0.

10. Ответ:

- (1) при свертке стека – при выходе из блока описания объекта, в частности при обработке исключений, завершении работы функции;
- (2) при уничтожении временных объектов - сразу, как только завершается конструкция, в которой они использовались;
- (3) при выполнении операции delete для указателя на объект (инициализация указателя - с помощью операции new),
- (4) при завершении работы программы при удалении глобальных/статических объектов.

Критерии: За отсутствие каждой из трех необходимых ситуаций -4.

Коллоквиум, 2012 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. В C++ наследование бывает открытым (public), закрытым (private) и защищенным (protected). Закрытые члены базового класса **видны**, но **недоступны** в производном классе, открытые члены – и **видны**, и **доступны**, защищенные члены **видны** и **доступны**, но только внутри производного класса.

Критерии: За каждый недостающий способ наследования **-4**, неверный ответ о статусе открытых и закрытых членов **-4**, если не упомянута видимость – не снижать.

2. Ответ.

```
class B { int x, y;
public: B (int a, int b = 5) { x = a; y = b; }
      int get () { return x + y; }
      B& operator+= (B & b) { x += b.x; y += b.y; return *this; }
};
```

Критерии: За каждый недостающий/неверный метод/лишний конструктор **-4**, за каждую мелкую ошибку реализации **-2**.

3. Ответ. **Cons Cons Des Copy Des 3 Des**

Критерии: За каждую неверную/лишнюю/недостающую печать **-4**, если пропущен конструктор, то за парный ему деструктор не снижать.

4. Ответ. (1) перед деструктором базового класса добавить **virtual**. **(2)** добавить B (int a = 0) { ... }

Критерии: За каждое не сделанное добавление **-5**.

5. Ответ.

```
class Cl { int i;
public: Cl (int x) { i = x; }
      Cl (const Cl & y) { i = y.i; }
      const Cl f (const Cl & c) const { cout << c.i << endl; }
};
const Cl t1 (const Cl a) { Cl b = Cl (5); return b.f (a); }
```

Критерии: За каждое не найденную или лишнюю ошибку **-5**.

6. Ответ.

```
template < class T > void f (const T & t) {
    if (t.size () < 3) return;
    typename T::const_reverse_iterator p = t.begin ();
    cout << *p + *(++p) + *(++p) << endl; // можно без скобок и лучше не
                                           // рассчитывать на порядок вычисления операндов.
}
int main () { vector <int > vec (5);
    for (int i = 0; i < 5; i++)
        vec.push_back (i);
    f (vec); // или f <vector <int > > (vec);
    return 0;
}
```

Критерии: Пропуск **typename** **-4**, если параметр функции **const**, а итератор **const** не имеет, снижение на 4 балла, за каждую другую ошибку – 3. Отсутствие & в заголовке функции **-1**.

7. Ответ.

```
add_st (K*) from K
5 5 5
```

Критерии: За неверный вызов функции **-5**, за любую ошибку со статической переменной **-5**.

8. SConstr → Copy → SCatchS& → Destr → SConstr → Main → Destr → Destr

Критерии: За отсутствие Copy (и его Destr) **-5**, за каждую другую ошибку **-2**.

9. Ответ:

```
void putnull ( B * pb ) {
    try { D d, * pd = (D*) pb;
        if (typeid (*pb) == typeid (d)) // возможен вариант с dynamic_cast
            for (int i = 0; i < 30; i++) pd -> mas [ i ] = 0;
    }
    catch (bad_typeid) { cout << "NULL!\n"; }
}
```

Критерии: Правильно добавлены средства **RTTI** – **10**, иначе **0**. За отсутствие try-catch блока не снижать, если это не может привести к ошибке.

10. Ответ:

1. в случае: Вох а (1, 2, 3); Вох b = a;
2. в случае: Вох с = Вох (3, 4, 5); // оптимизирующий компилятор вызывает только обычный конструктор с указанными параметрами;
3. при передаче параметров функции по значению (при создании локального объекта);
4. при возвращении результата работы функции в виде объекта,
5. при генерации исключения-объекта класса.

Критерии: За отсутствие каждой из трех обязательных ситуаций **-4**.

Коллоквиум, 2012 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. Различаются следующие виды полиморфизма: **статический** (на этапе компиляции, реализуется с помощью перегрузки функций/операций), **динамический** (во время выполнения программы, реализуется с помощью виртуальных функций), **параметрический** (на этапе компиляции, с использованием механизма шаблонов).

Критерии: За каждый недостающий вид полиморфизма **-4**, за каждый неназванный механизм реализации **-2**, за каждый неверный ответ о этапе разрешения полиморфизма **-2**.

2. Ответ.

```
class C { int x;
public: explicit C (int y) { x = 2 * y; }
      int get () { return x; }
      C operator + ( C c ) { C t ( x + c.x ); return t; }
};
```

Критерии: За каждый недостающий/неверный метод/лишний конструктор **-4**, за отсутствие **explicit** **-4**, за каждую ошибку реализации метода (неверный вывод...) **-2**.

3. Ответ. **Cons Copy Cons Des Des 1 Des**

Критерии: За каждую неверную/лишнюю/недостающую печать **-4**, если пропущен конструктор, то за парный ему деструктор не снижать.

4. Ответ. (1) перед деструктором базового класса добавить **virtual**.
(2) добавить D (int d = 0) {...}

Критерии: За каждое не сделанное добавление **-5**.

5. Ответ.

```
class Cls { int i;
public: Cls (int x) { i = x; }
      Cls (const Cls & y) { i = y.i; }
      const Cls f (const Cls c) const { cout << c.i << endl; }
};
const Cls t1 (const Cls * a) { Cls b = Cls (3); return a -> f(b); }
```

Критерии: За каждое не найденную или лишнюю ошибку - **5**

6. Ответ.

```
template < class T > void f (const T & t) {
    if (t.size () < 2) return;
    typename T:: const_reverse_iterator p = t.rbegin ();
    for (int i = 0; i < t.size() / 2; i++, p++)
        cout << * ( ++p ) << endl; // можно без скобок
}
int main () { list <int> lst;
    for (int i = 0; i < 5; i++)
        lst.push_back (i);
    f (lst); // или f <list <int> > (lst);
    return 0;
}
```

Критерии: Пропуск **typename** **-4**, если параметр функции **const**, а итератор **const** не имеет, снижение на 4 балла, за каждую другую ошибку – 3. Отсутствие & в заголовке функции **-1**.

7. Ответ.

```
g () from T  
7 7 7
```

Критерии: За неверный вызов функции -5, за любую ошибку со статической переменной -5.

8. SCatch_int → SConstr → Copy → SCatchS& → Destr → MainCatchS& → Destr

Критерии: За отсутствие Copy (и его Destr) -5, за каждую другую ошибку -2.

9. Ответ:

```
void puthi ( B * pb,    D * pd )    {  
    try { D  d; int i = 10;  
        if (typeid (*pb) == typeid (d)) { // возможен вариант с dynamic_cast  
            pd = (D*) pb;  
            while (i) strcpy ((pd -> txt) [--i], "Hi!"); }  
    }  
    catch (bad_typeid) { cout << "NULL!\n"; }  
}
```

Критерии: Правильно добавлены средства RTTI – 10, иначе 0. За отсутствие try-catch блока не снижать, если это не может привести к ошибке.

10. Ответ:

1. при создании объекта (при обработке описания объекта),
2. при создании объекта в динамической памяти (по new),
3. при композиции объектов наряду с собственным конструктором вызывается конструктор объекта – члена класса,
4. при создании объекта производного класса также вызывается конструктор и базового класса,
5. при автоматическом приведении типов с помощью конструктора преобразования.

Критерии: За отсутствие каждой из трех обязательных ситуаций -4.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow aS | Sb | aAb | ab$
 $aAb \rightarrow aaAbb | aabb$
 $aaAbb \rightarrow aaAbb$
 $aAb \rightarrow aAb$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

(б) Каким из перечисленных классов грамматик принадлежит G ? **Ответ:**

Класс Π	$G \in \Pi?$ (да/нет)
регулярные	
контекстно-зависимые	
контекстно-свободные	
грамматики типа 0	
неукорачивающие	

(в) Тип языка: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком $k+1$. **Ответ:**

2. Среди правил грамматик G_1 и G_2 найти и вычеркнуть одно правило (альтернативу) так, чтобы G_1 и G_2 стали эквивалентными. Ответ обосновать.

$G_1: S \rightarrow aAa | aSaB$
 $A \rightarrow Aa | BA | BS$
 $B \rightarrow bC$
 $C \rightarrow \varepsilon$

$G_2: S \rightarrow aAa | aB | aS$
 $Y \rightarrow Ya | b | YS$
 $B \rightarrow bA | bBC | \varepsilon$
 $C \rightarrow bAY$

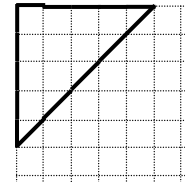
3. (а) Построить по заданной грамматике G конечный автомат (в виде ДС). (б) Если автомат оказался недетерминированным, преобразовать его в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА. (в) По ДКА построить левостолбчатую грамматику. Если автомат недетерминированный, то с помощью соответствующего алгоритма преобразовать его в эквивалентный ДКА.

$G:$
 $H \rightarrow IA | IB$
 $A \rightarrow OB | IC$
 $B \rightarrow IA | \perp$
 $C \rightarrow IA | IC$

4. Имеется клетчатый лист бумаги, бесконечный вправо и вниз. В левом верхнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{a, b, c\}$: a – переместить перо на одну клетку вправо; b – переместить перо по диагонали на одну клетку влево-вниз; c – переместить перо на одну клетку вверх.

Например, по последовательности $aaaaabbbbcccc$ интерпретатор построит прямоугольный равнобедренный треугольник с боковой стороной длины 5 (см. рисунок).

Построить грамматику с терминальным алфавитом $\{a, b, c\}$, описывающую подобные изображенному на рисунке треугольники с боковой стороной длины n , $n \geq 1$. В грамматике должно быть **не более 4** правил вывода (считая альтернативы).



5. Дана однозначная КС-грамматика $G_{balance}$, порождающая язык L , состоящий из цепочек, в которых символов a и b поровну:

$L = \{x \in \{a,b\}^* \mid |x|_a = |x|_b\}$. Вставить в грамматику действия вида $\langle cout \ll 'символ'; \rangle$ так, чтобы в процессе рекурсивного спуска был реализован перевод $\tau = \{(x, (ab)^{f(x)}) \mid x \in L, f(x) = |x|_a + |x|_b\}$.

$G_{balance}:$
 $S \rightarrow aAbS | bBaS | \varepsilon$
 $A \rightarrow aAbA | \varepsilon$
 $B \rightarrow bBaB | \varepsilon$

6. Какие стратегии трансляции могут выбираться в различных системах программирования? Объяснить существенные отличия этих подходов к реализации процесса трансляции с языка программирования. Приведите по одному примеру формального языка, к которому применена каждая из названных вами стратегий трансляции.

7. В чём различие статических и динамических библиотек? Какие компоненты вычислительной системы занимаются подключением к программам каждого типа библиотек?

8. По заданной грамматике $G = \{\{+, -\}, \{R, S\}, P, S\}$ получить эквивалентную неукорачивающую контекстно-свободную грамматику (использовать алгоритм устранения правил с пустой правой частью).

$$P: \quad S \rightarrow +R- \mid \varepsilon \\ R \rightarrow +R \mid -S \mid S$$

9. Дана грамматика G . Применím ли к ней метод рекурсивного спуска? Ответ обосновать.

$$G: \quad S \rightarrow aSb \mid Yb \mid bb \\ Y \rightarrow cSY \mid dd \mid \varepsilon$$

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов. Операция постфиксной записи '@' соответствует унарной операции изменения знака.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ	&a	#+	4	+	b	<=	36	!F	&b	+#	a	b	—	<	34	!F	&a
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
b	8	—	7	*	b	a	@	23	/	/	—	=	;	9	!	39	!
36	37	38	39														
&b	1	=	;														

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow aS \mid aSb \mid aAb \mid ab$
 $aAb \rightarrow aAbb \mid ab$
 $aaAbb \rightarrow aaAbb$
 $aS \rightarrow aaS$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

(б) Каким из перечисленных классов грамматик принадлежит G ? Ответ:

Класс	П	$G \in \Pi$? (да/нет)
регулярные		
контекстно-зависимые		
контекстно-свободные		
грамматики типа 0		
неукорачивающие		

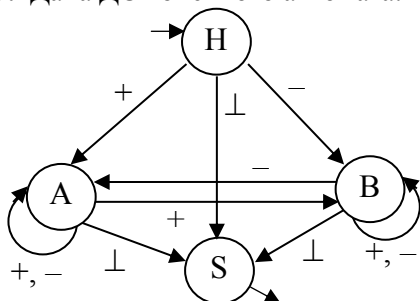
(в) Тип языка: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком $k+1$.

2. Среди правил грамматик G_1 и G_2 найти и вычеркнуть одно правило (альтернативу) так, чтобы G_1 и G_2 стали эквивалентными. Ответ обосновать.

G_1 : $S \rightarrow +S+Q \mid +R+$
 $P \rightarrow \varepsilon$
 $R \rightarrow QR \mid QS \mid R+$
 $Q \rightarrow -P$

G_2 : $S \rightarrow +V+ \mid +W \mid +S$
 $Y \rightarrow Y+ \mid - \mid YS$
 $W \rightarrow -V \mid -W-VY \mid \varepsilon$

3. Дана ДС конечного автомата:



а) Построить по ней соответствующую левостороннюю грамматику.

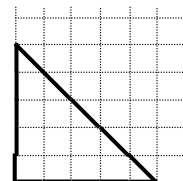
б) Если заданный автомат недетерминирован, преобразовать его в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА.

в) По ДКА построить правостороннюю грамматику.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{a, d, q\}$: a – переместить перо на одну клетку вправо; d – переместить перо по диагонали на одну клетку влево-вверх; q – переместить перо на одну клетку вниз.

Например, по последовательности $aaaaaddddqqqq$ интерпретатор построит прямоугольный равнобедренный треугольник с боковой стороной длины 5 (см. рисунок). Построить грамматику с терминальным алфавитом $\{a, d, q\}$, описывающую подобные изображенному на рисунке треугольники с боковой стороной длины n , $n \geq 1$.

В грамматике должно быть не более 4 правил вывода (считая альтернативы).



5. Дана однозначная КС-грамматика $G_{\text{parenthesis}}$, порождающая язык L сбалансированных скобочных систем.

$$G_{\text{parenthesis}}: \\ S \rightarrow (A)S \mid \varepsilon \\ A \rightarrow (A)A \mid \varepsilon$$

Неделимой называется скобочная система, которую нельзя представить в виде конкатенации двух других скобочных систем. Например, цепочка

$(() ())$ – неделимая скобочная система.

Протяжением скобочной системы называется число неделимых систем, конкатенация которых дает данную систему. Например, цепочка $(() ())$ имеет протяжение 1, а цепочка $(() ()) () ()$ имеет протяжение 2. Пустая цепочка имеет протяжение 0.

Вставить в грамматику действия вида $\langle \text{cout} << \text{'символ'}; \rangle$ так, чтобы в процессе рекурсивного спуска был реализован перевод $\tau = \{ (x, l^{f(x)}) \mid x \in L, f(x) - \text{протяжение цепочки } x \}$.

6. Дать определение и пояснить понятие «прохода компилятора». Как это понятие связано с понятием «фазы компиляции»?

7. Что является объектом оптимизационных преобразований в компиляторах? Укажите не менее трех видов машинно-независимых оптимизационных преобразований и приведите примеры каждого из них.

8. По заданной грамматике $G = \{ \{0, 1\}, \{A, S\}, P, S \}$ получить эквивалентную неукорачивающую контекстно-свободную грамматику (использовать алгоритм устранения правил с пустой правой частью).

$$P: \quad S \rightarrow 0AI \mid A \\ A \rightarrow 0A \mid 1A \mid \varepsilon$$

9. Дана грамматика G . Применим ли к ней метод рекурсивного спуска? Ответ обосновать.

$$G: \quad S \rightarrow dSbS \mid Y \\ Y \rightarrow cSY \mid ad \mid \varepsilon$$

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов. Операция постфиксной записи '@' соответствует унарной операции изменения знака.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ЛИЗ	a	33	!F	&a	&a	b	8	-	7	*	b	a	@	23	/	/	-	=	+=	;
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	
&b	#+	a	b	-	<	31	!F	4	!	40	!	&b	&a	&b	+#	/=	=	;		}

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow Sb|aSb|ab|aAb$
 $aAb \rightarrow aDAb|aab$
 $aD \rightarrow Da$
 $D \rightarrow a$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы или исчерпывающего словесного описания (не более 300 печатных знаков включая пробелы).

(б) Каким из перечисленных классов грамматик принадлежит G ? **Ответ:**

Класс	П	$G \in \Pi?$ (да/нет)
регулярные		
контекстно-зависимые		
контекстно-свободные		
грамматики типа 0		
неукорачивающие		

(в) Тип языка: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком $k+1$.

2. Среди правил грамматик G_1 и G_2 найти и вычеркнуть одно правило (альтернативу) так, чтобы G_1 и G_2 стали эквивалентными. Ответ обосновать.

$G_1: S \rightarrow ITI | ISIU$
 $B \rightarrow 0C | 0$
 $C \rightarrow \varepsilon$
 $T \rightarrow TI | UT | US$

$G_2: S \rightarrow ITI | 10T | 10UC | U | IU | IS$
 $C \rightarrow 0TZ$
 $U \rightarrow 0T | 0UC | \varepsilon$
 $Z \rightarrow ZI | 0 | ZS$

3. (а) Построить по заданной грамматике G конечный автомат (в виде ДС).

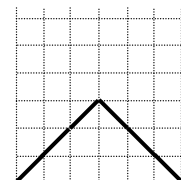
(б) Если автомат оказался недетерминированным, преобразовать его в соответствии с алгоритмом преобразования НКА в эквивалентный ДКА.

(в) По ДКА построить праволинейную грамматику.

$G: S \rightarrow A\perp | B\perp | \perp$
 $A \rightarrow Aa | Ba | Bb | a$
 $B \rightarrow Bb | Aa | Ab | b$

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{a, b, d\}$: a – переместить перо на одну клетку вправо; d – переместить перо по диагонали на одну клетку влево-вверх; b – переместить перо по диагонали на одну клетку влево-вниз.

Например, по последовательности $aaaaaadddbbb$ интерпретатор построит прямоугольный равнобедренный треугольник с основанием длины 6 (см. рисунок). Построить грамматику с терминальным алфавитом $\{a, b, d\}$, описывающую подобные изображенному на рисунке треугольники с основанием длины $2k$, $k \geq 1$. В грамматике должно быть не более 4 правил вывода (считая альтернативы).



5. Дана однозначная КС-грамматика $G_{\text{parenthesis}}$, порождающая язык L сбалансированных скобочных систем. *Неделимой* называется скобочная система, которую нельзя представить в виде конкатенации двух других скобочных систем. Например, цепочка $(())$ – неделимая скобочная система. Вставить в грамматику действия вида $\langle \text{cout} \ll \text{'символ'}; \rangle$ так, чтобы в процессе рекурсивного спуска был реализован перевод $\tau = \{(x, 1^{f(x)}) | x \in L, f(x) \text{ – количество неделимых скобочных подсистем в } x\}$. Например, для $x = (())())()$ должно быть $f(x) = 5$.

$G_{\text{parenthesis}}:$

$S \rightarrow TS | \varepsilon$

$T \rightarrow (S)$

6. От каких компонентов интегрированных систем программирования получают и каким компонентам передают данные для обработки входящие в них редакторы связей? Что подвергается обработке в редакторах связей и с чем связана необходимость этой обработки?

7. Какие области памяти выделяются компиляторами для хранения локальных данных процедур? Какая дисциплина распределения памяти при этом реализуется? Какой способ использования выбирается для этих областей? Какая ещё информация должна размещаться в тех же областях памяти, с той же дисциплиной распределения и с тем же способом использования?

8. По заданной грамматике $G = \{\{a, b, c\}, \{S, T, U\}, P, S\}$ получить эквивалентную неукорачивающую контекстно-свободную грамматику (использовать алгоритм устранения правил с пустой правой частью).

$$P: \begin{aligned} S &\rightarrow aTb \mid UT \mid cS \\ T &\rightarrow aT \mid bU \mid \varepsilon \\ U &\rightarrow bT \mid aU \mid \varepsilon \end{aligned}$$

9. Дана грамматика G . Применим ли к ней метод рекурсивного спуска? Ответ обосновать.

$$G: \begin{aligned} S &\rightarrow aSb \mid Yb \mid bb \\ Y &\rightarrow cSY \mid B \mid \varepsilon \\ B &\rightarrow d \mid \varepsilon \end{aligned}$$

10. По приведенной постфиксной записи фрагмента программы, написанной на языке Си++, восстановите этот фрагмент двумя способами: сначала, пользуясь операторами условного и безусловного перехода на метку, а затем (если это возможно), пользуясь только операторами структурного программирования без переходов. Операция постфиксной записи '@' соответствует унарной операции изменения знака.

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	a	23	!F	&a	&a	b	8	—	7	*	b	a	@	23	/	/	—

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
=	+=	;	28	!	&a	&b	+#	=	;	&b	#+	a	b	—	<	38	!F

36	37	38
1	!	

Экзамен, 2012 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{a^n b^k \mid n, k \geq 1\}$, (б) регулярные – "нет" (-3), КЗ – "да" (-1), КС – "нет" (-3), грамматики типа 0 – "да" (-3), неукорачивающие – "да" (-1).

(в) $k = 3$ (для заданного языка существует порождающая его регулярная грамматика)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: в грамматике G_2 нужно вычеркнуть правило $S \rightarrow aB$ или правило $B \rightarrow \varepsilon$. Тогда $L(G_1) = L(G_2) = \emptyset$.

Критерии: не вычеркнуто ни одного правила, вычеркнуто больше одного, или вычеркнуто не то правило: -10. За отсутствие доказательства эквивалентности полученных грамматик: -5.

3. Ответ: Автомат недетерминирован:

Функция переходов исходного автомата

	$\delta(H, 1) = A$ $\delta(H, 1) = B$		
	$\delta(A, 0) = B$ $\delta(A, 1) = C$ $\delta(B, 1) = A$ $\delta(B, \perp) = S$	$\delta(C, 1) = A$ $\delta(C, 1) = C$	
	Функция переходов детерминированного автомата ($D \equiv AB, E \equiv AC$)		
	$\delta'(H, 1) = D$ $\delta'(D, 0) = B$ $\delta'(D, 1) = E$ $\delta'(D, \perp) = S$	$\delta'(E, 0) = B$ $\delta'(E, 1) = E$ $\delta'(B, 1) = A$ $\delta'(B, \perp) = S$	$\delta'(A, 0) = B$ $\delta'(A, 1) = C$ $\delta'(C, 1) = E$

G' : $S \rightarrow B\perp \mid D\perp$ $A \rightarrow BI$ $B \rightarrow A0 \mid D0 \mid E0$ $C \rightarrow AI$
 $D \rightarrow I$ $E \rightarrow CI \mid DI \mid EI$

Критерии: (а) отсутствует или неэквивалентная ДС: -4, (б) отсутствует или построен неэквивалентный ДКА: -8, (в) отсутствует или неэквивалентная левостолбчатая грамматика: -4.

4. Ответ: Треугольник с боковой стороной длины n описывается цепочкой $a^n b^n c^n$. Следовательно, требуется построить грамматику, порождающую язык $\{a^n b^n c^n \mid n \geq 1\}$:
 $S \rightarrow aSBc \mid abc$ $cB \rightarrow Bc$ $bB \rightarrow bb$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -4. Если верно указан язык, но грамматика составлена неверно или имеет слишком много правил (превышение на три и более правила), за задачу ставить 2 балла.

5. Ответ: Соглашение: запись $\langle a \rangle$ является сокращением $\langle cout \ll "a"; \rangle$

При распознавании каждого символа a или b нужно печатать ab . Другой вариант ответа – печатать $(ab)^2$ при распознавании «ведущего» a или b . Варианты ответов (возможны и другие ответы):

$S \rightarrow a\langle ab \rangle Ab\langle ab \rangle S \mid b\langle ab \rangle Ba\langle ab \rangle S \mid \varepsilon$	$S \rightarrow a\langle abab \rangle AbS \mid b\langle abab \rangle BaS \mid \varepsilon$
$A \rightarrow a\langle ab \rangle Ab\langle ab \rangle A \mid \varepsilon$	$A \rightarrow a\langle abab \rangle AbA \mid \varepsilon$
$B \rightarrow b\langle ab \rangle Ba\langle ab \rangle B \mid \varepsilon$	$B \rightarrow b\langle abab \rangle BaB \mid \varepsilon$

Критерии: Перевод реализован без ошибок: 10 баллов. Есть ошибки: 0 баллов.

6. Ответ: (1) Трансляция может вестись на основе перевода одного языка программирования в другой – компиляция. (2) Трансляция может вестись с применением интерпретатора языка программирования. Такой подход существенно отличается от первого тем, что результирующая программа не создается. В результате обработки программы на исходном языке сразу вычисляется результат, алгоритм получения которого закодирован в исходной программе. (3) Возможен подход к трансляции, сочетающий в себе два предыдущих варианта (смешанная стратегия трансляции), когда сначала создается промежуточное представление исходной программы (подход «компиляция»), а затем это представление интерпретируется (подход «интерпретация»). (4) Паскаль, ассемблеры; ПЛЭНЕР, скриптовые языки; Java, ...

Критерии: За пропуск описания одного из двух основных подходов к трансляции снижать на 4 балла. За пропуск описания смешанной стратегии снижать на 2 балла. За пропуск или неверный ответ на последний вопрос -1.

7. Ответ: Статические и динамические библиотеки различаются по выбору момента времени извлечения их составных частей при формировании полной программы. Статические компоненты извлекаются компоновщиками (редакторами связей) в момент редактирования связей до начала исполнения программы, динамические – при выполнении программы (загрузчиками).

Критерии: За отсутствие или неправильное объяснение разницы концепций -6. За пропуск любого из компонентов: -3 (за каждый пропуск из двух).

8. Ответ: $S' \rightarrow S \mid \varepsilon$ $S \rightarrow +R- \mid +-$ $R \rightarrow +R \mid -S \mid S \mid + \mid -$

Критерии: За любую ошибку в ответе ставить 0 за всю задачу.

9. Ответ: неприменим, так как $first(bb) \cap first(Yb) = \{b\} \neq \emptyset$.

Критерии: ет обоснования или неверный ответ: -10. За каждую ошибку в обосновании: -4.

10. Ответ:

```

if (! (a ++ + 4 <= b)) goto N;
L: if (! (++ b < a - b))      goto M;
    a = (b - 8) * 7 - b / (- a / 23);  goto L;
M: goto E;
N: b = 1;
E:

```

Язык C++: `if (a+++4<=b) while (++b<a-b) a=(b-8)*7-b/(-a/23); else b = 1;`

Критерии: за неверный ответ на первый вопрос: -7, за неверный ответ на второй вопрос, если первый вариант верен: -3, иначе: -10.

Экзамен, 2012 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{a^n b^k \mid n, k \geq 1\}$

(б) регулярные – "нет" (-3), КЗ – "нет" (-1), КС – "нет" (-3), языки типа 0 – "да" (-3), неукорачивающие – "нет" (-1).

(в) $k = 3$ (для заданного языка существует порождающая его регулярная грамматика)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: в грамматике G_2 нужно вычеркнуть правило $S \rightarrow +W$ или $W \rightarrow \varepsilon$. Тогда $L(G_1) = L(G_2) = \emptyset$.

Критерии: не вычеркнуто ни одного правила, вычеркнуто больше одного, или вычеркнуто не то правило: -10. За отсутствие доказательства эквивалентности полученных грамматик: -5.

3. Ответ: Глев:

$$S \rightarrow A\perp \mid B\perp \mid \perp$$

$$A \rightarrow A+ \mid B- \mid A- \mid +$$

$$B \rightarrow B+ \mid B- \mid A+ \mid -$$

Автомат недетерминирован

Функция переходов исходного автомата

$\delta(H, +) = A$	$\delta(A, +) = A$	$\delta(B, +) = B$	
$\delta(H, -) = B$	$\delta(A, +) = B$	$\delta(B, -) = B$	
$\delta(H, \perp) = S$	$\delta(A, -) = A$	$\delta(B, -) = A$	
	$\delta(A, \perp) = S$	$\delta(B, \perp) = S$	
Функция переходов детерминированного автомата ($C \equiv AB$)			
$\delta'(H, +) = A$	$\delta'(A, +) = C$	$\delta'(B, +) = B$	$\delta'(C, +) = C$
$\delta'(H, -) = B$	$\delta'(A, -) = A$	$\delta'(B, -) = C$	$\delta'(C, -) = C$
$\delta'(H, \perp) = S$	$\delta'(A, \perp) = S$	$\delta'(B, \perp) = S$	$\delta'(C, \perp) = S$

Глев: $S \rightarrow A\perp \mid B\perp \mid C\perp \mid \perp$

$$A \rightarrow A- \mid +$$

$$B \rightarrow B+ \mid -$$

$$C \rightarrow A+ \mid B- \mid C+ \mid C-$$

Гправ: $H \rightarrow +A \mid -B \mid \perp$

$$A \rightarrow -A \mid +C \mid \perp$$

$$B \rightarrow +B \mid -C \mid \perp$$

$$C \rightarrow +C \mid -C \mid \perp$$

Критерии: (а) отсутствует или неправильная левосторонняя грамматика: -4, (б) отсутствует или построен неэквивалентный ДКА: -8, (в) отсутствует или неэквивалентная правосторонняя грамматика: -4.

4. Ответ: Треугольник с боковой стороной длины n описывается цепочкой $a^n d^n q^n$. Следовательно, требуется построить грамматику, порождающую язык $\{a^n d^n q^n \mid n \geq 1\}$:
 $S \rightarrow aSDc \mid adq$ $qD \rightarrow Dq$ $dD \rightarrow dd$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -4. Если верно указан язык, но грамматика составлена неверно или имеет слишком много правил (превышение на три и более правил), за задачу ставить 2 балла.

5. Ответ: Соглашение: запись $\langle I \rangle$ является сокращением $\langle cout \ll 'I'; \rangle$

$$S \rightarrow (A) \langle 1 \rangle S \mid \varepsilon$$

$$A \rightarrow (A) A \mid \varepsilon$$

Критерии: Перевод реализован без ошибок: 10 баллов. Есть ошибки: 0 баллов.

6. Ответ: Проход – это процесс последовательного чтения компилятором данных из внешней памяти, их обработки и записи результата во внешнюю память. Фаза компиляции – смысловая часть процесса компиляции, на которой происходит тот или иной процесс обработки: анализа+преобразования или синтеза. На одном проходе может выполняться сразу две фазы компиляции, но каждая фаза компиляции может выполняться за несколько проходов.

Критерии: За отсутствие определения прохода: **-5**. Неправильная трактовка фазы компиляции: **-5**.

7. Ответ: (1) Оптимизация в компиляторах проводится в отношении внутреннего представления исходной программы, (2) при оптимизации вновь формируется внутреннее представление программы, (3) к машинно-независимой оптимизации обычно относят (а) оптимизацию линейных участков, (б) оптимизацию циклов и (в) оптимизацию вызовов функций (процедур). Примеры:

(а) Арифметические преобразования $A=B*C+B*D \Rightarrow A=B*(C+D)$, (б) Расщепление цикла:

for (i = 0; i < n; i ++)	}	if (x < y)		
{ if (x < y)		{ S ₁ ; }	for (i = 0; i < n; i ++)	{ S ₁ ; }
else		{ S ₂ ; }	else	for (i = 0; i < n; i ++)

(в) Оптимизация вызовов: прямая подстановка тела функции, передача параметров на регистрах, девиртуализация вызовов.

Критерии: (а) Неправильный ответ на первые два вопроса: **-3** (за каждый), (б) За отсутствие какого-либо из видов оптимизации: **-2** (за каждый из трех пропусков), (в) За отсутствие примеров: **-1** (за каждый).

8. Ответ: $S' \rightarrow S \mid \varepsilon$ $S \rightarrow 0A1 \mid A \mid 01$ $A \rightarrow 0A \mid 1A \mid 0 \mid 1$

Критерии: За любую ошибку в ответе ставить **0** за всю задачу.

9. Ответ: применим, так как $first(dSbS) \cap first(Y) = \emptyset$, $first(cSY) \cap first(ad) = \emptyset$, $first(Y) \cap follow(Y) = \emptyset$.

Критерии: нет обоснования или неверный ответ: **-10**. За каждую ошибку в обосновании: **-4**.

10. Ответ:

```

if (! a)                goto N;
L: a += a = (b - 8) * 7 - b / (- a / 23);
  if (! (b++ < a - b))    goto M;
  goto L;
M: goto E;
N: b = a /= ++b;
E:

```

Язык C++: `if (a) do a+=a=(b-8)*7-b/(-a/23); while (++b<a-b); else b=a/=++b;`

Критерии: за неверный ответ на первый вопрос: **-7**, за неверный ответ на второй вопрос, если первый вариант верен: **-3**, иначе: **-10**.

Экзамен, 2012 год. Ответы и критерии оценок.

Вариант 3.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{a^n b^k \mid n, k \geq 1\}$

(б) регулярные – "нет" (-3), КЗ – "нет" (-1), КС – "нет" (-3), грамматики типа 0 – "да" (-3), неукорачивающие – "да" (-1).

(в) $k = 3$ (для заданного языка существует порождающая его регулярная грамматика)

Критерии: (а) описание отсутствует или с ошибками: -4, (б) снимаемые баллы за каждый неверный или отсутствующий ответ выше (обоснование не обязательно), (в) нет ответа; неверный ответ: -4 (обоснование не обязательно).

2. Ответ: в грамматике G_2 нужно вычеркнуть правило $U \rightarrow \varepsilon$. Тогда $L(G_1) = L(G_2) = \emptyset$.

Критерии: не вычеркнуто ни одного правила, вычеркнуто больше одного, или вычеркнуто не то правило: -10. За отсутствие доказательства эквивалентности полученных грамматик: -5.

3. Ответ: Автомат недетерминирован.

Функция переходов исходного автомата

	$\delta(H, a) = A$ $\delta(H, b) = B$ $\delta(H, \perp) = S$	$\delta(A, a) = A$ $\delta(A, b) = B$ $\delta(A, \perp) = S$	$\delta(B, a) = A$ $\delta(B, b) = B$ $\delta(B, \perp) = S$
	Функция переходов детерминированного автомата ($C \equiv AB$)		
	$\delta'(H, a) = A$ $\delta'(H, b) = B$ $\delta'(H, \perp) = S$	$\delta'(A, a) = C$ $\delta'(A, b) = B$ $\delta'(A, \perp) = S$ $\delta'(B, a) = A$ $\delta'(B, b) = C$ $\delta'(B, \perp) = S$	$\delta'(C, a) = C$ $\delta'(C, b) = C$ $\delta'(C, \perp) = S$

G' : $H \rightarrow aB \mid bA \mid \perp$ $A \rightarrow aC \mid bB \mid \perp$ $B \rightarrow aA \mid bC \mid \perp$ $C \rightarrow aC \mid bC \mid \perp$

Критерии: (а) отсутствует или неэквивалентная ДС: -4, (б) отсутствует или построен неэквивалентный ДКА: -8, (в) отсутствует или неэквивалентная праволинейная грамматика: -4.

4. Ответ: Треугольник с основанием длины $2n$ описывается цепочкой $a^{2n} d^n b^n$. Следовательно, требуется построить грамматику, порождающую язык

$$\{a^{2n} d^n b^n \mid n \geq 1\}:$$

$$S \rightarrow aaSdb \mid aadb$$

$$bD \rightarrow Db$$

$$dD \rightarrow dd$$

Критерии: порождается пустая цепочка: -5, имеется лишняя (непустая) или недостающая цепочка: -10, за каждое лишнее правило: -4. Если верно указан язык, но грамматика составлена неверно или имеет слишком много правил (превышение на три и более правила), за задачу ставить 2 балла.

5. Ответ: Соглашение: запись $\langle I \rangle$ является сокращением $\langle cout < < 'I'; \rangle$

$$S \rightarrow T \langle 1 \rangle S \mid \varepsilon$$

$$S \rightarrow TS \mid \varepsilon$$

$$S \rightarrow TS \mid \varepsilon$$

$$T \rightarrow (S)$$

$$T \rightarrow (S) \langle 1 \rangle$$

$$T \rightarrow (\langle 1 \rangle S)$$

Критерии: Перевод реализован без ошибок: 10 баллов. Есть ошибки: 0 баллов.

6. Ответ: Редакторы связей получают свои исходные данные от компиляторов и ассемблеров, а также из библиотек. Редакторы связей передают основные результаты своей работы загрузчикам. Таблицы, строящиеся при редактировании связей, передаются отладчикам. Редакторами связей обрабатываются именованные ссылки из одних модулей в другие, например, ссылки на внешние процедуры или глобальные объекты, определенные в других модулях. Компилятор (и ассемблер) не в состоянии обрабатывать такие ссылки, так как обычно используется стратегия раздельной трансляции модулей программы.

Критерии: За пропуск компонентов: **-3** (за компилятор), **-1** (за ассемблер), **-3** (за библиотеки), **-3** (за загрузчик), **-2** (за отладчик). За отсутствие или неправильное описание обрабатываемой информации и/или причины необходимости редактирования связей снижать на **5**.

7. Для хранения локальных данных процедур используется **(1)** динамическая память **(2)** со стековой дисциплиной распределения, **(3)** относящаяся к локальным областям памяти. **(4)** В тех же областях памяти, с той же дисциплиной распределения и с тем же способом использования хранятся записи о текущем состоянии процесса выполнения программ, а также записи о входах в блоки операторов, которые можно рассматривать в качестве процедур без параметров.

Критерии: **(а)** Неправильный ответ на первые три вопроса: **-2** (за каждый), **(б)** За пропуск ответа о записях состояния процесса выполнения: **-3**, **(в)** За пропуск ответа о входах в блоки: **-1**.

8. Ответ:

$$\begin{array}{ll} S' \rightarrow S \mid \varepsilon & S \rightarrow aTb \mid ab \mid UT \mid T \mid U \mid cS \mid c \\ T \rightarrow aT \mid bU \mid a \mid b & U \rightarrow bT \mid aU \mid a \mid b \end{array}$$

Критерии: За любую ошибку в ответе ставить **0** за всю задачу.

9. Ответ: неприменим, так у нетерминала Y есть две альтернативы, из которых выводится ε .

Критерии: нет обоснования или неверный ответ: **-10**. За каждую ошибку в обосновании: **-4**.

10. Ответ:

```

L: if (! a)                                goto M;
    a += a = (b - 8) * 7 - b / (- a / 23);
    goto N;
M: a = ++b;
    goto L;
E:

```

Язык C++: `do if(a) a+=a=(b-8)*7-b/(-a/23); else a=++b; while(++b<a-b);`

Критерии: за неверный ответ на первый вопрос: **-7**, за неверный ответ на второй вопрос, если первый вариант верен: **-3**, иначе: **-10**.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Для **каждого имени** из реализации функций `B::g()` и `main()` укажите (используя операцию "`::`") область видимости имён так, чтобы программа была верной. **Подчеркните** вставленные расширения имён, без которых программа ошибочна. Если вариантов расширения имён несколько, укажите все.

```
static int x;
namespace N1 { int f (int a, int b){ return x = a + b; }
    class A { int x;
        public: A (int n = 13) { x = n; }
        int f(){ return ::x = x; }
    };
    namespace N2 { class B: public A { float x;
        public: int f (int a) { return ::x = x; }
        void g ();
    };
}
void B::g() { x = 20.13; f ();
f (2013); f (13, 20132013);
}
int main () { B b; b. f (); b. f (13);
x = f ('2', 013);
return 0;
}
```

2. Есть ли ошибки в функции `main()` приведённого фрагмента на Си++? Если есть, то **объясните**, в чём они заключаются. Если нужно, удалите ошибочные вызовы из функции `main()`. Что будет выдано в стандартный поток вывода при работе исправленной (если нужно) программы?

```
class A {public:virtual void f() { h (65); cout << "A::f" << endl; }
    virtual void g(int x){ h (x); cout << "A::g," << x << endl; }
    void h(int x){ cout << "A::h," << x << endl; }};
class B: public A { public:
    void g(int x){ h (x); cout << "B::g," << x << endl; }
    virtual void h(int x){ cout << "B::h," << x << endl; }};
class C: public B { float f;
    public: void g() { h (67); cout << "C::g" << endl; }
    virtual void h(int x){ cout << "C::h," << x << endl; }};
int main(){ C c, * pc = & c; B * pb = & c; A * pa = & c;
pa -> f (); pa -> f (2013);
pa -> g (13); pb -> f ();
pb -> g (); pb -> g (13);
pb -> h (0); pc -> f ();
pc -> g (); pc -> h ();
pc -> h (0); return 1;
}
```

3. Что такое функция преобразования в языке Си++? Какие ограничения накладываются на использование функций преобразования при выборе одной из перегруженных функций? Привести пример использования функции преобразования в программах на Си++.

4. Используя описания классов А и В из задания 1, определить, тела каких конструкторов и деструкторов (возможно явно не определённые) и в каком порядке будут исполнены при работе следующего фрагмента программы, расположенного в пространстве имён `N2` из задания 1:

```
void h () { struct C { A a; B b; }; A a(2013); C c; }
```

5. Для класса рациональных дробей

```
template<class T> class fraction { T n; T d; ... };
```

написать два варианта реализации вне класса операции "/", выполняющей деление одной рациональной дроби на другую, а именно – методом класса и функцией-другом этого класса.

6. Описать алгоритм определения возможности использования имён из базовых классов в производном классе в случае множественного наследования. Привести по одному примеру, демонстрирующему случаи обнаружения ошибки на каждом из шагов алгоритма.

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct X; void f(X & x, int n);
struct X { X () { cout << "Xc";
               f (*this, -1); cout << "Xf"; }
          X (X &) { cout << "Xk"; } };
struct Y: X { Y () { try { cout << "Yt"; f (*this, 1);
                           cout << "Yp";
                       }
                       catch (Y) { cout << "Yy"; throw; }
                       catch (int) { cout << "Yi"; throw; }
          }
          Y (Y &) { cout << "Yk"; } };
void f (X & x, int n) { try { cout << "ft";
                           if (n < 0) throw x;
                           cout << "f0";
                           if (n > 0) throw 1;
                           cout << "f1";
                       }
                       catch (int) { cout << "fi"; throw; }
                       catch (X& a) { cout << "fc";
                                       f(x, 1);
                                       cout << "fr";
                                       throw;
                       }
int main() { try { Y a; } catch (...) { cout << "mc"; return 0; }
           cout << "mr"; return 0; }
```

8. В приведённой программе возможно наличие синтаксических ошибок в определении класса А. Если ошибки есть, исправьте их заменой, исключением или добавлением нужных служебных слов языка Си++. Обоснуйте сделанные исправления.

```
class A { int i;
         int f (int & x) { return g (x); }
         int g (int & x) { if (x >= 0) f (-i); return i; } };
int A::i = 2013;
int main () { const A a; A::i = 201; a.f (20); return a.i = 1; }
```

9. Для приведённой ниже программы описать функцию $B^*f(A^*)$, которая, получая в качестве параметра указатель типа A^* , возвращает его значение, наиболее безопасным образом преобразованное к типу B^* , а в случае невозможности преобразования корректно завершает работу программы.

```
struct A { };
struct B:A {int x; B (int y = 5){ x = y; } };
int main () { try { B b, *pb = f(&b); cout << pb -> x << '\n';
                  A a; pb = f(&a); cout << pb -> x << '\n';
              }
              catch (...) { return 0;
              }
```

10. Даны описания: `typedef vector<int> V;`

```
struct Weight_t { V::size_type Index; // индекс элемента вектора
                 float Weight;       // вес элемента вектора
};
typedef list<Weight_t> L;
```

Написать функцию $g()$, которая по заданному вектору типа V и соответствующему ему списку типа L вычисляет средневзвешенное значение элементов вектора (средний результат умножения элементов на их веса), выдавая в выходной поток значения и веса элементов.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Если в реализации функций `g()` и `main()` есть ошибки, исправьте их, используя только операцию расширения области видимости `::`. Для всех обращений к функциям `f()` и `g()` укажите, из какой области видимости они вызываются (если вариантов расширения имён несколько, укажите все). Подчеркните те расширения имён, без которых программа ошибочна.

```
int x;
int f(){ return x = 2; }
struct A { int f() { return x = 0; } };
namespace S1 { class B: public A { int x;
    public: B(int n = 1){ x = n; }
    short int f(){ return x = 3; }
    int f(int a, int b) { return x = a + b; }
};
namespace S2 { class C: public B { int x;
    public: int f(int a) { return a; }
    int f(int a, int b) { return b; }
    int g() { x = f ();
        x = f (2);
        x = f (0, 1); return 3; }
    };
}
}
int main () {
    C c;      x =      c.      f ();
    x =      c.      f (2013);
    return    c.      g ();
}
```

2. Есть ли ошибки в функции `main()` приведённого фрагмента на Си++? Если есть, то **объясните**, в чём они заключаются. Если нужно, удалите ошибочные вызовы из функции `main()`. Что будет выдано в стандартный поток вывода при работе исправленной (если нужно) программы?

```
class T { virtual void p () { z (85); cout << "T::p" << endl; }
    public: void u (int x) { z (x); cout << "T::u," << x << endl; }
    void z (int x) { z (); cout << "T::z," << x << endl; }
    void z () { cout << "T::z" << endl; } };
class U: public T { public:
    virtual void p () { z (86); cout << "U::p" << endl; }
    void u (int x) { z (x); cout << "U::u," << x << endl; }
    virtual void z (int x) { cout << "U::z," << x << endl; } };
class W: public U { public: unsigned u;
    void p (int x) { z (); cout << "W::p," << x << endl; }
    void z () { cout << "W::z" << endl; } };
int main() {
    W w, * pw = &w;      T * pt = &w;      U * pu = &w;
    pt -> p ();           pt -> u (201);      pt -> z ();      pu -> p ();
    pu -> u (201);         pu -> z (20);       pw -> p ();      pw -> p (2013);
    pw -> u (201);         pw -> z ();       pw -> z (20);    return 2; }
}
```

3. Что такое конструктор преобразования? Какие ограничения накладываются на использование конструкторов преобразования при выборе одной из перегруженных функций? Привести пример использования конструктора преобразования в программах на Си++.

4. Используя описания классов A, B и C из задания 1, определить, тела каких конструкторов и деструкторов (возможно явно не определённые) и в каком порядке будут исполнены при работе следующего фрагмента программы, расположенного в пространстве имён `S2` из задания 1:

```
void h () { struct D { A a; C c; D (int x = 0) {} }; D d(2013); }
```

5. Для класса рациональных дробей

```
template <class T> class fraction { T n; T d; ... };
```

написать два варианта реализации вне класса операции "<", выполняющей сравнение двух рациональных дробей, а именно – методом класса и функцией-другом класса.

6. В некоторой программе, написанной на языке Си++, имеется *try-catch* – блок, внутри которого генерируется исключение типа *E*. Каким должен быть тип *H* параметра обработчика исключения, чтобы возникшая исключительная ситуация могла быть передана ему на обработку?

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct A;
void g (A & a, int i);
struct A { A () { cout << "Ak";
              g (*this, -1); cout << "Ag"; }
          A (A &) { cout << "Ap"; }
};
struct B: A { B () { try { cout << "Bt"; g (*this, -1); cout << "Bf"; }
                      catch (B) { cout << "Bb"; throw; }
                      catch (int) { cout << "Bi"; throw; }
                    }
          B (B &) { cout << "Bp"; }
};
void g (A & a, int i) { try { cout << "gt";
                          if (i < 0) throw a;
                          cout << "g0";
                          if (i > 0) throw 1;
                          cout << "g1";
                        }
          catch (int) { cout << "gi"; throw; }
          catch (A& a) { g(a, 0); cout << "gr"; throw; }
}
int main() try { B a; }
          catch (...) { cout << "mc"; return 0; }
                    cout << "mr"; return 0;
}
```

8. В приведённой программе возможно наличие синтаксических ошибок в определении класса *B*. Если ошибки есть, исправьте их заменой, исключением или добавлением нужных служебных слов языка Си++. Обоснуйте сделанные исправления.

```
class B { int x; int y;
        int p () { return y >= 0 ? y = -1 : q (); }
        int q () const { return p (); }
        int r () { return x = y; }
        B (int z) { x = y > 0 ? z % y : -y; }
};
int B::y = 13;
int main () { B::y = 1; B::p();
             const B b1 (2013), b2 (b1);
             b1.q(); return b2.x = 2;
}
```

9. В системе классов, описанной в задаче 1, в пространстве имён *S2* описать функцию *C&(A&)*, которая, получая в качестве параметра ссылку на объект базового класса *A*, возвращает ссылку на объект производного класса *C*, полученную наиболее безопасным образом, а в случае невозможности приведения типов корректно завершает программу.

10. Даны описания: `typedef vector<double> V;`

```
struct Signif_t { V::size_type Index; // индекс элемента вектора
                bool Signif; };      // значимость элемента (true – да, false – нет)
```

```
typedef list<Signif_t> S;
```

Написать функцию *g()*, которая по заданному вектору типа *V* и соответствующему ему списку типа *S* вычисляет сумму значащих элементов вектора, выдавая в выходной поток индексы и значения суммируемых элементов.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Если в реализации функций `g()` и `main()` есть ошибки, исправьте их, используя только операцию расширения области видимости `::`. Для всех обращений к функциям `f()` и `g()` укажите, из какой области видимости они вызываются (если вариантов расширения имён несколько, укажите все). Подчеркните те расширения имён, без которых программа ошибочна.

```
int x = 0;
namespace P1 { int x = 2;      int f (int a) { return x = a; }
               class A { int x;
               public:   A (int n = 0)   { x = n; }
                           int f ()      { return x; }
                           int f (int a) { return x = a; } };
               class B { int x;
               public:   B (int n = 1)   { x = n; }
                           int f (int a) { return x = a; }
                           void g ()    { x = f (3); } };
               namespace P2 { struct C: A, B { int x;
                           int f (int a) { return x = a; } }; } }
int main () {      A a, * pa;      B b, * pb;      C c, * pc;
                x = f (2); pa = &a;      x = pa -> f (0);
pa = &c;      x = pa -> f (3);      x = pa -> f ();
pb = &b;      x = pb -> f (0); pb = &c;      x = pb -> f (1);
pc = &c;      x = pc -> f (3);      x = pc -> f ();
                c.g ();      return 2013; }
```

2. Есть ли ошибки в функции `main()` приведённого фрагмента на Си++? Если есть, то **объясните**, в чём они заключаются. Если нужно, удалите ошибочные вызовы из функции `main()`. Что будет выдано в стандартный поток вывода при работе исправленной (если нужно) программы?

```
class P { public:
    virtual void m () { n (0); cout << "P::m" << endl; }
    virtual void n (int x) { l (x); cout << "P::n," << x << endl; }
    void n () { cout << "P::n" << endl; }
    virtual void l (int x) { cout << "P::l," << x << endl; }
    void l () { cout << "P::l" << endl; } };
class Q: public P { public:
    virtual void m (int x) { n (x); cout << "Q::m," << x << endl; }
    virtual void n (int x) { l (x); cout << "Q::n," << x << endl; }
    virtual void n () { cout << "Q::n" << endl; }
    virtual void l (int x) { cout << "Q::l," << x << endl; }
    virtual void l () { cout << "Q::l" << endl; } };
class R: public Q { public: long l;
    void m (int x) { n (x); cout << "R::m," << x << endl; }
    void n (int x) { cout << "R::n," << x << endl; } };
int main() { R r, * pr = &r; P * pp = &r; Q * pq = &r;
    pp -> m ();      pp -> n ();      pp -> n (2013);
    pp -> l ();      pp -> l (13);      pq -> m (2013);
    pq -> n ();      pq -> l ();      pr -> m (2013);
    pr -> l ();      return 3; }
```

3. Какие виды расширения типов имеются в языке Си++? Приведите пример программы на языке Си++, в котором существенно используется отличие расширения от стандартного преобразования типа.

4. Используя описания классов `A` и `B` из задания 1, определить, тела каких конструкторов и деструкторов (возможно явно не определённые) и в каком порядке будут исполнены при работе следующего фрагмента программы, расположенного в пространстве имён `P2` из задания 1:

```
void h () { A a; class D { B b; A a; public: D (): b(2013), a() {} }; D d; }
```

5. Для класса рациональных дробей

```
template<class T> class fraction { T n; T d; ... };
```

написать два варианта реализации вне класса префиксной операции увеличения "++", выполняющей увеличение на 1 значения рациональной дроби, а именно – методом класса и функцией-другом класса.

6. В некоторой программе на языке Си++ имеются определения классов, каждый из которых содержит $f()$. В функции $main()$ этой программы определен указатель p , который может указывать на объекты, обладающие методом $f()$. Может ли вызов $p \rightarrow f()$, используемый в цикле, давать разные результаты на разных итерациях цикла, если все функции в данной программе работают только с нестатическими локальными переменными и не используют средств получения данных из внешней среды (то есть ввод, обращение к таймеру и тому подобное). Если нет, объяснить почему. Если да, указать условия, при которых это возможно и привести пример такой программы.

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct T;
void f(T & t, int n);
struct T { T () { f(*this, 0); cout << "Te"; }
          T (T &){ cout << "Tk"; } };
struct U: T { U () { try { cout << "Ut"; f (*this, 1); cout << "Uf"; }
                     catch (U) { cout << "Uc"; throw; }
                     catch (int){ cout << "Ui"; throw; } }
          U (U &) { cout << "Uk"; }
};
void f (T & t, int a) {
try { cout << "ft";
      if (a < 0) throw t;
      if (a > 0) throw 1;
      cout << "fr"; }
catch (int) { cout << "fi"; throw; }
catch (T& t) { cout << "fk"; f(t, 0); cout << "ff"; throw; }
}
int main( try { U a; }
catch (...){ cout << "mc"; return 0; }
          cout << "mr"; return 0; }
```

8. В приведённой программе возможно наличие синтаксических ошибок в определении класса M . Если ошибки есть, исправьте их заменой, исключением или добавлением нужных служебных слов языка Си++. Обоснуйте сделанные исправления.

```
class M { int m;
void m1() { if (m < 0) m2(m); }
int m2(int & n) const { return m1(), n; }
void m3(int & n) { m = m2(n); } };
int M::m = 1;
int main() { M::m3 (2013); M mm; return mm.m2 (3); }
```

9. В системе классов, описанной в задаче 1, в пространстве имён $P2$ описать функцию $C * f(B \&)$, которая, получая в качестве параметра ссылку на объект класса B , возвращает указатель на объект класса C , полученный наиболее безопасным образом, а в случае невозможности приведения типов корректно завершает программу.

10. Даны описания:

```
typedef vector<bool> B;
struct Value_t { B::size_type Index; // индекс элемента вектора
               int Value;           }; // значение элемента
typedef list<Value_t> T;
```

Написать функцию $g()$, которая по заданному вектору значимости типа B и соответствующему ему списку типа T в порядке обратном порядку элементов списка выдаёт в выходной поток значения полей элементов списка, сопровождаемое их значимостью, вычисляет сумму значимых целочисленных полей списка и возвращает это значение.

Коллоквиум, 2013 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
void N1::N2::B::g() { N1::N2::B::x = 20.13; N1::A::f();
N1::N2::B::f(2013); N1::f(13, 2013*2013); }
int main() { N1::N2::B b; b.N1::A::f();
b.N1::N2::B::f(13); ::x = N1::f('2', 013); }
```

Отмечены исправляющие (обязательные) изменения.

Критерии: За отсутствие каждого необходимого (из 6) расширения имени (без которого ошибка) -2, за каждое иное не отмеченное уточнение области видимости (из 6) -1, При указании хотя бы одного правильного расширения имени (где это необходимо) за отсутствие вариантов расширений не снижать. Для остальных расширений имен считать достаточным указание ближайшей области видимости, за отсутствие полной цепочки вложенных областей видимости – не снижать.

2. Ответ. Неверные вызовы:

`pc -> f()` (в классе *C* нет функции с именем *f*, поле *f* не доступно)
`pa -> f(2013)` (в классе *A* нет функции *f()* с параметром),
`pb -> g()` (в классе *B* нет функции *g()* без параметров)
`pc -> h()` (в классе *C* нет функции *h()* без параметров)

Результат: **A::h,65 A::f C::h,13 B::g,13 A::h,65 A::f**
C::h,13 B::g,13 C::h,0 C::h,67 C::g C::h,0

Критерии: За каждую не найденную\лишнюю ошибку или неправильный вызов функции -1.

3. Ответ. Функция преобразования – метод класса специального вида (***operator*** <*mun*>()), которые преобразуют объект класса к типу <*mun*>, например, “***operator int***()”. Алгоритм преобразования кодируется в теле функций преобразования.

(1) Пользовательские преобразования автоматически применяются только, если они однозначны.
(2) Допускается не более одного пользовательского преобразования для обработки одного параметра одного вызова.

```
struct S { long s; S(){ s = 0L; } // конструктор умолчания
operator long () { return s; } }; // S --> long
long f(long l) { return l; } // Теперь разрешено: S s; long nl = f(s);
```

Критерии: За неправильное объяснение -5, за каждое из двух неправильно описанных или пропущенных ограничений -3, за отсутствие примера -2.

4. Ответ: `A(2013) A() A() B() C() ~C() ~B() ~A() ~A() ~A()`.

Критерии: За каждую ошибку -3, за указание `A()` вместо `A(2013)` снижать на 1 балл.

5. Ответ.

```
template<class T>fraction<T> fraction<T>::operator/ (const fraction<T>&x)
{ fraction<T> w; w.n = n * x.d; w.d = d * x.n; return w; }
template<class T>fraction<T> operator/ (const fraction<T>&x, const fraction<T>&y)
{ fraction<T> w; w.n = x.n * y.d; w.d = x.d * y.n; return w; }
```

Критерии: За отсутствие\ошибку каждого примера (за однотипные ошибки снижать 1 раз): -5.

6. Ответ. При исследовании возможности использования имени проводится (1) проверка однозначности, (2) выбор перегруженной функции, (3) контроль доступа.

```
class A { int g(int); public: double x;... };
class B { public: double x; void f(short); void f(char);... };
class C: public A, public B { public: ... };
C c; C * p = &c; p -> x = 1.0; // неоднозначность наследования поля A::x
p -> f(1); // неоднозначность выбора функции f()
p -> g(2); // запрет доступа к функции A::g(int)
```

Критерии: За пропуск одного из трёх шагов: -3, за указание их неправильного порядка: -3, за отсутствие каждого из трёх примеров: -1.

7. Ответ: **Xc ft Xk fc ft f0 fi mc**

Критерии: За каждую неверную (лишнюю или пропущенную) печать **-2**.

8. Ответ: (1) Заменить слово `"class"` словом `"struct"`.
(2) Перед определением типа поля `"i"` вставить слово `"static"`.
(3) В определениях методов `"f()"` и `"g()"` вставить после пустого списка формальных параметров слово `"const"`.
(4) В определениях методов `"f()"` и `"g()"` вставить перед типом формального параметра слово `"const"`.

Критерии: За пропуск ошибок доступа к закрытым членам класса: **-5**, за пропуск указания статичности поля `"i"`: **-8**, за пропуск каждой из двух ошибок вызова константных методов: **-4**, за пропуск каждой из двух ошибок в заголовках функций: **-4**, за указание ошибки там, где её нет: **-5**.

9. Ответ:

```
B* f(A* p){
    B* pb = dynamic_cast<A*>(p);
    if(pb) return pb;
    else throw 0; /*exit(0);*/
}
```

Критерии: За утверждение о невозможности безопасного преобразования ставить полный балл: **10**. За безошибочное решение с использованием динамического приведения (как здесь написано, как будто класс полиморфен): **-1**, за любую другую ошибку: **-10** (включая использование статического приведения).

10. Решение:

```
float g (const V & vect, const L & lst)
{
    L::const_iterator lp = lst.begin ();
    V::size_type t, n = 0; float vw;
    float w = 0.0f;
    while (lp != lst.end ())
    {
        t = (* lp).Index;
        vw = (* lp).Weight; ++ lp;
        if (t < 0 || t >= vect.size ()) continue;
        cout << vect [t] << ' ' << vw << endl;
        w += vect [t] * vw;
        ++ n;
    }
    return n ? w / n : 0.0f;
}
```

Критерии: За каждую серьёзную ошибку: **-3**, за использование обычного итератора вместо константного, отсутствие проверок на пустой список и выход за пределы диапазона индексов не снижать.

Коллоквиум, 2013 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
x = :: f (); или A::f (); или S1::B::f ();
x = S1::S2::C::f (2);
x = S1::S2::C::f (0, 1);
int main() { S1::S2::C c; x = c.S1::B::f (); или x = c.S1::A::f ();
x = c.S1::S2::C::f (2013);
return c.S1::S2::C::g (); }
```

Отмечены исправляющие (обязательные) изменения.

Критерии: За каждое не отмеченное уточнение области видимости (из 9) **-1**, за каждый не найденный вариант обязательного исправления (из 5) ещё **-2**.

2. Ошибочными являются вызовы:

pt -> *p*() (в классе *T* нет доступа к имени *p*),
pu -> *p*() (в классе *W* нет функции *p*() без параметров), и
pw -> *u*(201) (в классе *W* имя *u* – это не функция),
pw -> *z*(20) (в классе *W* нет функции *z*() с параметром).

Результат: **T::z T::z,201 T::u,201 T::z U::z,86 U::p**
U::z,201 U::u,201 U::z,20 W::z W::p,2013 W::z

Критерии: За каждую не найденную\лишнюю ошибку или неправильный вызов функции **-1**.

3. Конструктор преобразования – конструктор, имеющий один параметр, но не являющийся конструктором копирования (тип параметра отличается от ссылки на собственный класс). Алгоритм преобразования кодируется в теле конструкторов преобразования.

(1) Пользовательские преобразования автоматически применяются только, если они однозначны.
(2) Допускается не более одного пользовательского преобразования для обработки одного параметра одного вызова. (3). Конструктор не должен иметь спецификатор **“explicit”**.

```
struct S { long s;
S (long nl){ s = nl; } }; // long --> S
S f (S s) { return s; } // Теперь разрешено: S s = f (1L);
```

Критерии: За пропуск упоминания о конструкторе копирования при наличии упоминания об одном параметре **-3**, за совсем неправильное объяснение **-5**, за каждое из трёх неправильно описанных или пропущенных ограничений **-2**, за отсутствие примера **-2**.

4. Ответ: A() A() B() C() D(2013) ~D() ~C() ~B() ~A() ~A().

Критерии: За каждую ошибку **-3**, за указание D() вместо D(2013) снижать на 1 балл.

5. Ответ.

```
template<class T> bool fraction<T>::operator< (const fraction<T>& x)
{ return n * x.d < d * x.n; }
template<class T> bool operator< (const fraction<T>& x, const fraction<T>& y)
{ return x.n * y.d < x.d * y.n; }
```

Критерий: За отсутствие каждого примера или ошибки в нем: **-5**.

6. Ответ. Исключительная ситуация типа *E* обрабатывается обработчиком для ситуации типа *H*: (1) если тип *H* имеет тот же тип, что и *E*, (2) если тип *H* является однозначным доступным базовым типом для типа *E*, (3) если *H* и *E* являются указателями, и условия 1) или 2) выполняются для типов, на которые они ссылаются, (4) если *H* является ссылкой, и условия 1) или 2) выполняются для типа, на который ссылается *H*.

Критерий: За пропуск каждого из условий: **-3**.

7. Ответ: Ak gt Ap gt g0 g1 gr mc

Критерии: За каждую неверную (лишнюю или пропущенную) печать **-2**.

- 8. Ответ:** (1) Заменить слово `"class"` словом `"struct"`.
 (2) Перед определением типа поля `"y"` вставить слово `"static"`.
 (3) В определениях методов `"p ()"` и `"q ()"` перед типом возвращаемого значения вставить слово `"static"`.
 (4) В определении метода `"q ()"` удалить слово `"const"`.

Критерии: За пропуск ошибок доступа к закрытым членам класса: **-5**, за пропуск указания статичности поля `"y"`: **-8**, за пропуск любой ошибки определения статических методов `"p ()"` и `"q ()"`: за пропуск ошибки указания константности статического метода `"q ()"`: **-5**, за указание ошибки там, где её нет (метод `"r ()"` и поле `"x"` не обязаны быть статическими): **-5**.

9. Ответ:

```

C & f (A & ra) {
    try { C & rc = dynamic_cast<C &>(ra);
        return rc; }
    catch (bad_cast) { exit (0); }
}

```

Критерий: За пропуск `try-catch`-блока: **-6**, за любую другую ошибку: **-10**.

10. Решение:

```

double g (const V & vect, const S & lst)
{ S::const_iterator lp = lst.begin (); bool sg;
  V::size_type t;
  double w = 0.0;
  while (lp != lst.end ())
  { t = (* lp).Index;
    sg = (* lp).Signif;
    ++ lp;
    if (t < 0 || t >= vect.size ()) continue;
    if (sg)
    { cout << t << ' ' << vect [t] << endl;
      w += vect [t];
    }
  }
  return w;
}

```

Критерий: За каждую серьёзную ошибку: **-3**, за использование обычного итератора вместо константного, отсутствие проверок на пустой список и выход за пределы диапазона индексов не снижать.

Коллоквиум, 2013 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ:

```
void g () { x = P1::B::f (3); }
int main() { P1::A a, * pa; P1::B b, * pb; P1::P2::C c, * pc;
              x = P1::f (2);
              pa = &a; x = pa -> P1::A:: f (0);
              pa = &c; x = pa -> P1::A:: f (3);
              x = pa -> P1::A:: f ();
              pb = &b; x = pb -> P1::B:: f (0);
              pb = &c; x = pb -> P1::B:: f (1);
              pc = &c; x = pc -> P1::C:: f (3);
              x = pc -> P1::A:: f ();
              c.P1::B::g (); }
```

Отмечены исправляющие (обязательные) изменения.

Критерии: За отсутствие каждого необходимого (из 5) расширения имени (без которого ошибка) **-2**, за каждое иное не отмеченное уточнение области видимости (из 8) **-1**, При указании хотя бы одного правильного расширения имени (где это необходимо) за отсутствие вариантов расширений не снижать. Для остальных расширений имен считать достаточным указание ближайшей области видимости, за отсутствие полной цепочки вложенных областей видимости – не снижать.

2. Ответ. Неверные вызовы:

`pr -> l()` (имя функции скрывает имя поля-данного l)

Результат: R::n,0 P::m P::n R::n, 2013 P::l Q::l, 13
R::n, 2013 R::m, 2013 Q::n Q::l R::n, 2013 R::m, 2013

Критерии: За каждую не найденную ошибку, лишнюю ошибку или неправильно вызванную функцию **-1**.

3. Ответ. Расширение типа – некоторая часть стандартных преобразований (часто – наиболее безопасная часть). Конкретно к расширениям относятся:

- целочисленные расширения: (1) `char --> int (unsigned int)`; (2) `short --> int (unsigned int)`; (3) `enum --> int (unsigned int)`; (4) `bool --> int`; (5) `enum --> long int (unsigned long int`, если тип `int` не подходит); (6) если можно, то битовые поля могут расширяться до `int (unsigned int)`, иначе они не расширяются.
- расширения чисел с плавающей точкой: `float --> double`.

```
void f (double) /* ... */; void f (int) /* ... */;
void g () { short aa = 1; // тип short расширяется до int
           float ff = 1.0;
           f (ff); // f (double)
           f (aa); // f (int)
           } // Здесь расширение избавляет от неоднозначности выбора функции f().
```

Критерии: За пропуск пунктов (5) и (6) не снижать, за пропуск любого другого из 5 оставшихся видов расширения **-2**, за пропуск беззнаковых вариантов – не снижать, за совсем неправильное объяснение **-10**, за отсутствие примера **-2**.

4. Ответ: A () B(2013) A () D () ~D () ~A () ~B () ~A ().

Критерии: За каждую ошибку **-3**, за указание B () вместо B(2013) снижать на 1 балл.

5. `template<class T> fraction<T> & fraction<T>::operator++ ()`
`{ r += d; return * this; }`
`template<class T> fraction<T> & operator++ (fraction<T>& x)`
`{ x.r += x.d; return x; }`

Критерии: За отсутствие\ошибку каждого примера (за однотипные ошибки снижать 1 раз): **-5**.

6. **Ответ.** Да, может. В программе должна быть определена иерархия классов, в которую входят классы с виртуальным методом $f()$. Каждый из методов должен реализовывать разные алгоритмы, возвращающие разные результаты. При обращении к методу $f()$ указатель p на разных итерациях должен содержать значения, относящиеся к разным классам иерархии.

```
struct A { virtual int f(int) { return 0; } ... };
struct B:A { virtual int f(int) { return 1; } ... };
int main () { int i; A * a [5], * p; B b [5]; A a2;
               for (i = 0; i < 5; ++i) { a [i] = & b [i]; }
               a [2] = & a2;
               for (i = 0; i < 5; ++i) { p = a [i]; p -> f (); } }
```

Критерии: За ответ “нет”: **-10**, за неправильное объяснение и пример: **-10**, за отсутствие объяснения при правильном примере: **-2**, за отсутствие примера при правильном подробном объяснении: **-3**.

7. **Ответ:** **ft fr Te Ut ft fi Ui mc**

Критерии : За каждую неверную (лишнюю или пропущенную) печать **-2**.

8. **Ответ:** (1) Заменить слово *"class"* словом *"struct"*.

2. Перед определением типа поля *"m"* вставить слово *"static"*.

3. В определениях методов *"m1 ()"*, *"m2 ()"* и *"m3 ()"* перед типом возвращаемого значения вставить слово *"static"*.

4. В определении метода *"m2 ()"* удалить слово *"const"*.

5. В определениях методов *"m2 ()"* и *"m3 ()"* вставить перед типом формального параметра слово *"const"*.

Критерии: За пропуск ошибок доступа к закрытым членам класса: **-5**, за пропуск указания статичности поля *"m"*: **-8**, за пропуск любой ошибки определения статических методов *"m1 ()"*, *"m2 ()"* и *"m3 ()"*: **-6**, за пропуск ошибки указания константности статического метода *"m2 ()"*: **-5**, за пропуск каждой из двух ошибок в заголовках функций: **-4**, за указание ошибки там, где её нет: **-5**.

9. **Ответ:**

```
C * f (B & rb) {
    try { C & rc = dynamic_cast<C &>(rb); return & rc; }
    catch (bad_cast) { exit (0); }
}
```

Критерии: За утверждение о невозможности безопасного преобразования ставить полный балл: **10**. За безошибочное решение с использованием динамического приведения (как здесь написано, как будто класс полиморфен): **-1**, за любую другую ошибку: **-10** (включая использование статического приведения).

10. **Решение:**

```
int g (const V & vect, const T & lst)
{ T::const_reverse_iterator lp = lst.rbegin ();
  B::size_type t; int vl;
  int w = 0;
  while (lp != lst.rend ())
  { t = (* lp).Index; vl = (* lp ++).Value;
    if (t < 0 || t >= vect.size ()) continue;
    if (vect [t]) w += vl;
    cout << t << ' ' << vl << ' ' << vect [t] << endl;
  }
  return w;
}
```

Критерии: За каждую серьёзную ошибку: **-3**, за использование обычного итератора вместо константного, отсутствие проверок на пустой список и выход за пределы диапазона индексов не снижать.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow aA$

$aA \rightarrow aaB \mid a$

$B \rightarrow abb \mid aCbb \mid b$

$C \rightarrow ab \mid aaD$

$D \rightarrow bb \mid abbb$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы.

Ответ:

$L(G)=$

(б) Каким из перечисленных классов грамматик принадлежит G ? **Ответ:**

Класс П	$G \in \Pi?$ (да/нет)
регулярные	
контекстно-зависимые	
контекстно-свободные	
грамматики типа 0	
неукорачивающие	

(в) Тип языка: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком типа $k+1$.

Ответ: $k =$ _____

2. Дать определение системы программирования.

Из перечисленных компонентов вычеркните те, которые не входят в состав системы программирования: *текстовый редактор,*

макрогенератор,

драйвер клавиатуры,

редактор связей.

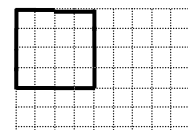
3. 1) Дать определение языка, порождаемого грамматикой $G=(T,N,P,S)$.

2) Привести пример грамматики (задав все элементы порождающей грамматики), которая порождает пустой язык.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вниз. В левом верхнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{a, b, c, d\}$: a – переместить перо на одну клетку вправо; b – вниз; c – влево; d – вверх.

Например, по последовательности $aaaabbbbccccddd$ интерпретатор построит квадрат, изображенный на рисунке, со стороной длины 4.

Построить грамматику с терминальным алфавитом $\{a, b, c, d\}$, описывающую все квадраты со стороной длины n ($n \geq 1$), начерченные по часовой стрелке. В грамматике должно быть **не более 6** правил вывода (считая альтернативы). **Решения с большим числом правил не рассматриваются.** Грамматика не должна описывать другие фигуры.



5. 1) Привести по одному примеру ситуаций, когда под объекты программы выделяется:

а) **статическая,**

б) **динамическая** (произвольного распределения),

в) **стековая память.**

2) В какой памяти располагаются таблицы виртуальных методов полиморфных классов?

6. Дана грамматика G :

$$\begin{aligned} S &\rightarrow aA|b|\varepsilon \\ A &\rightarrow AA|B \\ B &\rightarrow bBB \\ C &\rightarrow d \end{aligned}$$

(а) Является ли G приведенной? **Обоснуйте ответ.** Если нет, примените к ней алгоритм приведения грамматики, и выпишите получившуюся грамматику G' .

(б) Является ли исходная грамматика G однозначной? **Обоснуйте ответ.**

7. Дана автоматная левolineйная грамматика:

$$\begin{aligned} S &\rightarrow C\perp \\ C &\rightarrow AI \\ A &\rightarrow BI|C0|0 \\ B &\rightarrow BI|0 \end{aligned}$$

а) Детерминирован ли разбор по данной грамматике (ответ обосновать)?

б) Построить конечный автомат (КА) по заданной грамматике.

в) Если КА недетерминированный, преобразовать его к ДКА.

г) Построить **праволинейную** грамматику, соответствующую ДКА.

8. Построить неукорачивающую КС-грамматику, эквивалентную заданной КС грамматике, пользуясь алгоритмом преобразования «КС $\rightarrow$ неукорачивающая КС».

$$\begin{aligned} S &\rightarrow ABc|c|A \\ A &\rightarrow aA|\varepsilon \\ B &\rightarrow bB|\varepsilon \end{aligned}$$

9. Применим ли метод рекурсивного спуска к заданной грамматике. **Ответ обосновать.**

$$\begin{aligned} S &\rightarrow ABc|c \\ A &\rightarrow aA|\varepsilon \\ B &\rightarrow bB|\varepsilon \end{aligned}$$

10. Дана постфиксная запись фрагмента программы (на языке Си). Вставьте пропущенные в постфиксной записи команды перехода ('!', '!F') и метки и восстановите фрагмент программы, если известно, что в исходном фрагменте не было операторов перехода.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
x	0	>			y	x	<			&y	y	1	x	+	+	=	;		
21	22	23	24	25	26	27													
27		&	x	=	;														

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow 0A1$
 $0A \rightarrow 0B1 \mid 01$
 $B1 \rightarrow 0C111 \mid 011$
 $C \rightarrow 0D \mid 00D1 \mid 0$
 $D \rightarrow 01$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы.

Ответ:
 $L(G) =$

(б) Каким из перечисленных классов грамматик принадлежит G ? **Ответ:**

Класс Π	$G \in \Pi?$ (да/нет)
регулярные	
контекстно-зависимые	
контекстно-свободные	
грамматики типа 0	
неукорачивающие	

(в) Тип языка: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком типа $k+1$.

Ответ: $k =$ _____

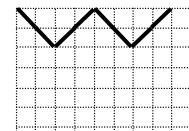
2. Привести общую схему работы компилятора с указанием результатов работы каждого из этапов. Из перечисленных действий вычеркните те, которые не участвуют в компиляции:

лексический анализ,
 контроль контекстных условий,
 верификация,
 генерация машинного кода.

3. 1) Дать определение эквивалентных грамматик. 2) Привести пример эквивалентных грамматик.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вниз. В левом верхнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{a, b\}$: a – переместить перо по диагонали на одну клетку вправо вниз; b – переместить перо по диагонали на одну клетку вправо вверх. Например, по последовательности $aabbaabb$ интерпретатор построит ломаную из четырех равных отрезков длины 2, похожую на букву **W** (см. рисунок).

Построить грамматику с терминальным алфавитом $\{a, b\}$, описывающую все буквы **W** с длиной отрезка n ($n \geq 1$). В грамматике должно быть **не более 6** правил вывода (считая альтернативы). Решения с большим числом правил не рассматриваются. Грамматика не должна описывать другие фигуры.



5. Каковы основные свойства языка внутреннего представления программы? Привести два примера таких языков.

6. Дана грамматика G :

$$\begin{aligned} S &\rightarrow B \mid C \\ C &\rightarrow c \mid cCB \\ D &\rightarrow d \\ B &\rightarrow bcB \mid B \end{aligned}$$

(а) Является ли G приведенной? **Обоснуйте ответ.** Если нет, примените к ней алгоритм приведения грамматики, и выпишите получившуюся грамматику G' .

(б) Является ли исходная грамматика G однозначной? **Обоснуйте ответ.**

7. Дана автоматная
праволинейная грамматика:

$$\begin{aligned} S &\rightarrow 0A \mid 0B \mid 1C \\ A &\rightarrow 0B \mid 1C \\ B &\rightarrow 1B \mid 1A \\ &\rightarrow \perp \end{aligned}$$

а) Детерминирован ли разбор по данной грамматике
(ответ обосновать)?

б) Построить конечный автомат (КА) по заданной грамматике.

в) Если КА недетерминированный, преобразовать его к ДКА.

г) Построить **леволинейную** грамматику, соответствующую S ДКА.

8. Построить неукорачивающую КС-грамматику, эквивалентную заданной КС грамматике, пользуясь **алгоритмом преобразования «КС $\rightarrow$ неукорачивающая КС»**.

$$\begin{aligned} S &\rightarrow ABc \mid A \\ A &\rightarrow aAb \mid \varepsilon \\ B &\rightarrow bBa \mid \varepsilon \end{aligned}$$

9. Применим ли метод рекурсивного спуска к заданной грамматике. **Ответ обосновать.**

$$\begin{aligned} S &\rightarrow AB \mid \varepsilon \\ A &\rightarrow aAb \mid \varepsilon \\ B &\rightarrow bBa \mid \varepsilon \end{aligned}$$

10. Дана постфиксная запись фрагмента программы (на языке Си). Вставьте пропущенные в постфиксной записи команды перехода ('!', '!F') и метки и восстановите фрагмент программы, если известно, что в исходном фрагменте не было операторов перехода.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
y	x	<			x	0	>			&y	y	2	x	+	+	=	;	25	
21	22	23	24	25	26	27													
&y	x	=	;	1															

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow aAb \mid \varepsilon$
 $A \rightarrow aBbb \mid ab \mid \varepsilon$
 $aB \rightarrow aaaCb \mid aa$
 $C \rightarrow D \mid \varepsilon$
 $D \rightarrow ab$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы.

Ответ:
 $L(G)=$

(б) Каким из перечисленных классов грамматик принадлежит G ? **Ответ:**

Класс П	$G \in \Pi?$ (да/нет)
регулярные	
контекстно-зависимые	
контекстно-свободные	
грамматики типа 0	
неукорачивающие	

(в) Тип языка: найти такое целое k , что язык $L(G)$ является языком типа k и не является языком типа $k+1$

Ответ: $k=$ _____

2. В чем отличие компиляторов и интерпретаторов?

Из перечисленных действий вычеркните те, которые не участвуют в интерпретации:

синтаксический анализ,
генерация объектного модуля,
работа с таблицей имен,
ввод входных данных.

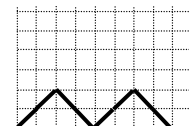
3. 1) Дать определение почти эквивалентных грамматик.

2) Привести пример почти эквивалентных грамматик.

4. Имеется клетчатый лист бумаги, бесконечный вправо и вверх. В левом нижнем углу расположено перо, оставляющее на бумаге след при перемещении. Перо управляется интерпретатором-графопостроителем. На вход интерпретатору подается последовательность символов-команд из алфавита $\{c, d\}$: c – переместить перо по диагонали на одну клетку вправо вверх; d – переместить перо по диагонали на одну клетку вправо вниз.

Например, по последовательности $ccddccdd$ интерпретатор построит ломаную из четырех равных отрезков длины 2, похожую на букву **М** (см. рисунок).

Построить грамматику с терминальным алфавитом $\{c, d\}$, описывающую все буквы **М** с длиной отрезка n ($n \geq 1$). В грамматике должно быть не более 6 правил вывода (считая альтернативы). **Решения с большим числом правил не рассматриваются.** Грамматика не должна описывать другие фигуры.



5. Перечислить основные функции редактора связей. Привести пример ситуации, в которой редактор связей выдаст сообщение об ошибке.

6. Дана грамматика G :

$$S \rightarrow aSB \mid c$$

$$A \rightarrow aA \mid Aa \mid a$$

$$C \rightarrow \varepsilon$$

$$B \rightarrow bB$$

(а) Является ли G приведенной? **Обоснуйте ответ.** Если нет, примените к ней алгоритм приведения грамматики, и выпишите получившуюся грамматику G' .

(б) Является ли исходная грамматика G однозначной? **Обоснуйте ответ.**

7. Дана автоматная
леволинейная грамматика:

$$S \rightarrow C \perp$$

$$C \rightarrow Ab \mid Ba$$

$$A \rightarrow Ab \mid a$$

$$B \rightarrow Bb \mid a$$

а) Детерминирован ли разбор по данной грамматике
(ответ обосновать)?

б) Построить конечный автомат (КА) по заданной грамматике.

в) Если КА недетерминированный, преобразовать его к ДКА.

г) Построить **праволинейную** грамматику, соответствующую ДКА.

8. Построить неукорачивающую КС-грамматику, эквивалентную заданной КС грамматике, пользуясь **алгоритмом преобразования «КС $\rightarrow$ неукорачивающая КС»**.

$$S \rightarrow aSBb \mid bA \mid B$$

$$A \rightarrow aB \mid b$$

$$B \rightarrow bB \mid \varepsilon$$

9. Применим ли метод рекурсивного спуска к заданной грамматике. **Ответ обосновать.**

$$S \rightarrow aSb \mid bA$$

$$A \rightarrow aB \mid b$$

$$B \rightarrow bB \mid \varepsilon$$

10. Дана постфиксная запись фрагмента программы (на языке Си). Вставьте пропущенные в постфиксной записи команды перехода ('!', 'F') и метки и восстановите фрагмент программы, если известно, что в исходном фрагменте не было операторов перехода.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
x	0	>			&y	y	3	x	+	+	=	;	y	x	<			6	

21	22	23	24	25	26	27
27		&y	x	=	;	{

Экзамен, 2013 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Тип грамматики – 0 (другие – “нет”),
 $L(G) = \{ a^n b^{n-1} \mid n=1,2,3,4,5,6 \}$. Тип языка – 3 (регулярный).

Критерии: Неверный язык -4, неверный тип языка -4, неверный основной тип грамматики -3, за каждую неверную принадлежность другим типам -1.

2. Система программирования – это комплекс программных инструментов и библиотек, который поддерживает **весь** технологический (жизненный) цикл создания программного продукта. Не входит – драйвер клавиатуры.

Критерии: Нет\Неверное определение – -6, нет библиотек – не снижать, нет технологического цикла -4, любая ошибка(и) в вычеркивании -5.

3. 1) Язык, порождаемый грамматикой $G=(T,N,P,S)$, это **множество терминальных** цепочек конечной длины, **выводимых из начального символа грамматики**. Или $L(G) = \{ \alpha \in T^* \mid S \Rightarrow \alpha \}$
2) **Пример.** $G = \{ \{a\}, \{S, A\}, \{S \rightarrow Aa\}, S \}$, или $G = \{ \{a\}, \{S\}, \emptyset, S \}$.

Критерии: Неверное определение -5, за отсутствие слов «конечной длины» – не снижать, выделенное шрифтом - **обязательно**. Нет\неверный пример -5.

4. Нужно описать язык $\{a^n b^n c^n d^n \mid n \geq 1\}$. $S \rightarrow aBSCd \mid abcd$
 $Ba \rightarrow aB$
 $dC \rightarrow Cd$
 $Bb \rightarrow bb$
 $cC \rightarrow cc$

Критерии: Выводится лишняя цепочка, не выводится нужная цепочка, превышение количества правил: -10. **Внимание:** в корректном решении не может быть правил вида $B \rightarrow b$, когда не-терминал превращается в терминал независимо от контекста, за подобные правила сразу =0. Если грамматика неверная или отсутствует, но верно указан язык $\{a^n b^n c^n d^n \mid n \geq 1\}$ – всего =2.

5. 1) (а) **Статическая** – при описании, например, глобальных переменных. (б) **Динамическая** – например, при выделении памяти с помощью new, при работе с контейнерами STL. (в) **Стековая** – например, при работе рекурсивных функций, при передаче фактических параметров по значению в функции. 2) В статической.

Критерии: За каждый неверный пример -3, за неверный ответ на вопрос -3.

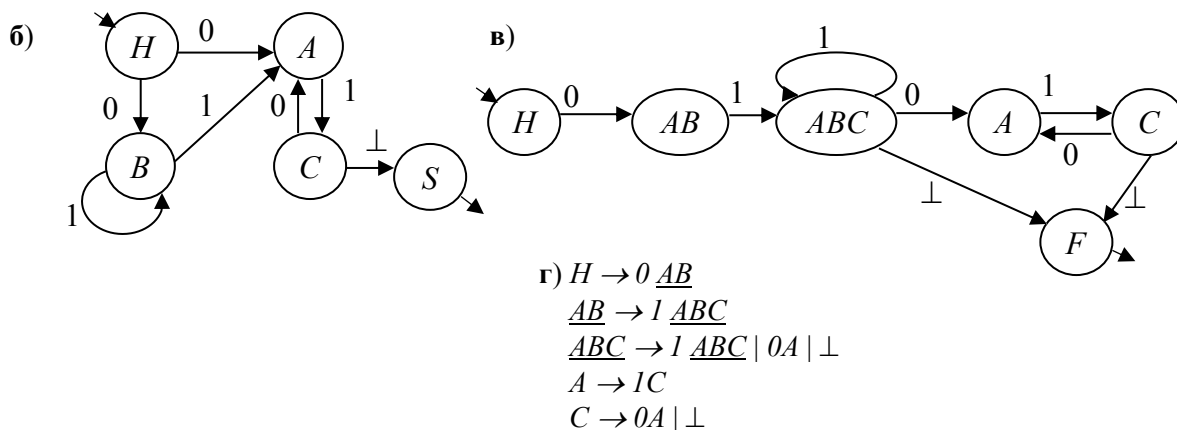
6. (а) Не является приведенной, так как грамматика содержит недостижимый символ C. (Возможно указание других бесполезных символов, достаточно указать хотя бы один).

$G': S \rightarrow b \mid \varepsilon$

(б) Является однозначной, так как в грамматике выводятся всего две цепочки: b и ε, обе выводятся за один шаг вывода, так что дерево вывода каждой цепочки единственно.

Критерии: неверный (необоснованный) ответ или ошибки в пункте (а): -5, неверный (необоснованный) ответ или ошибки в пункте (б): -5.

7. а) **Нет**, так как правила вывода содержат повторяющиеся правые части. Обоснование возможно также через автомат.



Критерии: Неверный ответ на вопрос (а): **-5**, нет обоснования: **-3**, неверный КА: **-4**, неверный ДКА: **-5**, неверная грамматика для ДКА: **-2**.

8. Ответ.

$$\begin{aligned}
 S' &\rightarrow S \mid \varepsilon \\
 S &\rightarrow ABc \mid Ac \mid Bc \mid c \mid A \\
 A &\rightarrow aA \mid a \\
 B &\rightarrow bB \mid b
 \end{aligned}$$

Критерии: Пропуск каждой альтернативы **-5** (Пропуск первой строки **-10**).

9. **Нет**, так как пересечение множеств $first(ABc) = \{a, b, c\}$ и $first(c) = \{c\}$ непусто.

Критерии: Неверный ответ или неверное обоснование **-10**, неточности в множествах $first(A)$ и/или $follow(A)$: **-2**.

10. Ответ.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
x	0	>	23	!F	y	x	<	21	!F	&y	y	1	x	+	+	=	;	6	!
21	22	23	24	25	26	27													
27	!	&y	x	=	;														

`if (x > 0) while (y < x) y = y + (1 + x); else y = x;`

Критерии: Есть ошибки в постфиксной записи: **-5**, есть ошибки в восстановленном фрагменте: **-5**. **Внимание:** скобки в выражении $y+(1+x)$ обязательны, без них постфиксная запись была бы другой. Если без ошибок только расставлены операции перехода, оценка **2 балла** за задачу.

Экзамен, 2013 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Тип грамматики – КЗ, НУ, тип 0 (другие – “нет”),
 $L(G) = \{0^n 1^{n+1} \mid n=1,2,3,4,5\}$. Тип языка 3 (регулярный).

Критерии: Неверный язык -4, неверный тип языка -4, неверный основной тип грамматики -3, за каждую неверную принадлежность другим типам -1.

2. Фаза анализа: (исходная программа на ЯП) $\Rightarrow$ лексический анализ $\Rightarrow$ (последовательность лексем) $\Rightarrow$ синтаксический анализ $\Rightarrow$ (промежуточное представление программы) $\Rightarrow$ семантический анализ//контроль контекстных условий. **Фаза синтеза:** подготовка к генерации объектного модуля $\Rightarrow$ генерация объектного модуля $\Rightarrow$ (объектный модуль).
Не участвует – верификация.

Критерии: Нет\Неверная схема -6, нет фазы синтеза -4, нет промежуточных результатов -2 за каждый. Любая ошибка(и) в вычеркивании -5.

3. 1) Грамматики эквивалентны, если порождают один и тот же язык. 2) Пример:

$$\begin{array}{lcl} S \rightarrow 0S1 \mid 01 & \text{и} & S \rightarrow 0A1 \\ & & A \rightarrow 0A1 \mid \varepsilon \end{array}$$

Критерии: Неверное определение -5, нет\неверный пример -5.

4. Нужно описать язык $\{a^n b^n a^n b^n \mid n \geq 1\}$.
 $S \rightarrow aBSAb \mid abab$
 $Ba \rightarrow aB$
 $bA \rightarrow Ab$
 $aA \rightarrow aa$
 $Bb \rightarrow bb$

Критерии: Выводится лишняя цепочка, не выводится нужная цепочка, превышение количества правил: -10. **Внимание:** в корректном решении не может быть правил вида $B \rightarrow b$, когда нетерминал превращается в терминал независимо от контекста, за подобные правила сразу -10. Если грамматика неверная или отсутствует, но верно указан язык $\{a^n b^n a^n b^n \mid n \geq 1\}$, всего =2.

5. Свойства: (а) позволяет фиксировать синтаксическую структуру исходной программы; (б) текст на нем можно автоматически генерировать во время синтаксического анализа; (в) его конструкции относительно просто транслируются в объектный код либо достаточно эффективно интерпретируются.

Примеры: Р-код, байт-код виртуальной Java-машины, постфиксная запись.

Критерии: За неверный/отсутствующий ответ по пункту (а) -4, за ошибки по пунктам (б) и (с) по -2 за каждый, за каждый неверный/отсутствующий пример -2.

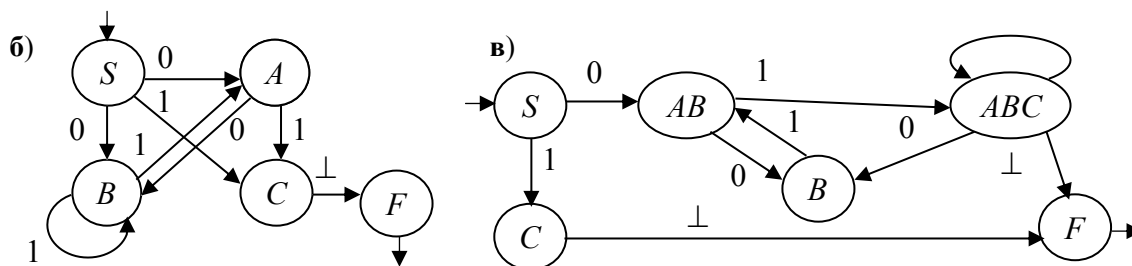
6. (а) Не является приведенной, так как содержит недостижимый символ D . (Возможно указание других бесполезных символов, достаточно указать хотя бы один).

$$G': S \rightarrow C \quad C \rightarrow c$$

(б) Является однозначной, так как в грамматике выводятся всего одна цепочка "с", для которой существует единственный вывод $S \rightarrow C \rightarrow c$, а значит, единственно дерево вывода.

Критерии: неверный (необоснованный) ответ или ошибки в пункте (а): -5, неверный (необоснованный) ответ или ошибки в пункте (б): -5.

7. а) Нет, так как есть альтернативы правил вывода, начинающиеся одинаковыми нетерминалами. Обоснование возможно также через автомат.



г) $S \rightarrow C \perp \mid \underline{ABC} \perp$
 $C \rightarrow \perp$
 $\underline{ABC} \rightarrow \underline{ABC} \perp \mid \underline{AB} \perp$
 $B \rightarrow \underline{ABC} \perp \mid \underline{AB} \perp$
 $\underline{AB} \rightarrow B \perp \mid \perp$

Критерии: Неверный ответ на вопрос (а): **-5**, нет обоснования: **-3**, неверный КА: **-4**, неверный ДКА: **-5**, неверная грамматика для ДКА: **-2**.

8. $S' \rightarrow S \mid \varepsilon$
 $S \rightarrow ABc \mid Ac \mid Bc \mid c \mid A$
 $A \rightarrow aAb \mid ab$
 $B \rightarrow bBa \mid ba$

Критерии: Пропуск каждой альтернативы **-5** (пропуск первой строки **-10**).

9. Нет, так как $AB \Rightarrow \varepsilon$ и $\varepsilon \Rightarrow \varepsilon$.

Критерии: Неверный ответ или неверное обоснование **-10**.

10.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
y	x	<	27	!F	x	0	>	21	!F	&y	y	2	x	+	+	=	;	25	!
21	22	23	24	25	26	27													
&y	x	=	;	1	!	}													

`while (y < x) if (x > 0) y = y + (2 + x); else y = x;`

Критерии: Есть ошибки в постфиксной записи: **-5**, есть ошибки в восстановленном фрагменте: **-5**. **Внимание:** скобки в выражении $y+(2+x)$ обязательны, без них постфиксная запись была бы другой. Если без ошибок только расставлены операции перехода, оценка **2 балла** за задачу.

Экзамен, 2013 год. Ответы и критерии оценок.

Вариант 3.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Тип грамматики – 0 (другие – “нет”), $L(G) = \{ a^n b^n \mid n=0,1,2,3,4,5 \}$.

Тип языка 3 (регулярный).

Критерии: Неверный язык -4, неверный тип языка -4, неверный основной тип грамматики -3, за каждую неверную принадлежность другим типам -1.

2. **Компилятор** преобразует программу на ЯП (немашинном) в эквивалентную программу в машинных кодах, которая затем может использоваться многократно независимо от компилятора, получая во время выполнения входные данные и выдавая результат. **Интерпретатор**, получая на вход исходные данные, читает предложение за предложением исходной программы на ЯП, переводит в машинные коды и непосредственно выполняет его, получая в итоге результат выполнения программы.

Не участвует – генерация объектного модуля.

Критерии: Неверный ответ на первый вопрос -6, неточности (индивидуально) от -1 до -2 за каждую. Любая ошибка(и) в вычеркивании -5.

3. 1) Грамматики почти эквивалентны, если порождаемые ими языки отличаются не более, чем на пустую цепочку (ϵ). Или грамматики G_1 и G_2 *почти эквивалентны*, если $L(G_1) \cup \{\epsilon\} = L(G_2) \cup \{\epsilon\}$.

2) **Пример.** $G = \{\{0, 1\}, \{S\}, \{S \rightarrow 0S1 \mid \epsilon\}, S\}$, или

$G = \{\{0, 1\}, \{S, A\}, \{S \rightarrow 0A1 \mid \epsilon; A \rightarrow 0A1 \mid \epsilon\}, S\}$.

Критерии: Неверное определение -5, за «не более, чем» не снижать. Нет\неверный пример -5.

4. Нужно описать язык $\{c^n d^n c^n d^n \mid n \geq 1\}$.

$S \rightarrow cDSCd \mid cdcd$

$Dc \rightarrow cD$

$dC \rightarrow Cd$

$Dd \rightarrow dd$

$cC \rightarrow cc$

Критерии: Выводится лишняя цепочка, не выводится нужная цепочка, превышение количества правил: -10. **Внимание:** в корректном решении не может быть правил вида $B \rightarrow b$, когда не-терминал превращается в терминал независимо от контекста, за подобные правила сразу =0. Если грамматика неверная или отсутствует, но верно указан язык $\{c^n d^n c^n d^n \mid n \geq 1\}$ – всего =2.

5. (1) связывает между собой по внешним данным объектные модули, порождаемые компилятором и составляющие единую программу, (2) связывает файлы статически подключаемых библиотек с целью получения единого исполняемого модуля, (3) готовит таблицу трансляции относительных адресов для загрузчика, (4) готовит таблицу точек вызова функций динамически подключаемых библиотек.

Пример: отсутствие описания объявленного имени функции при его использовании.

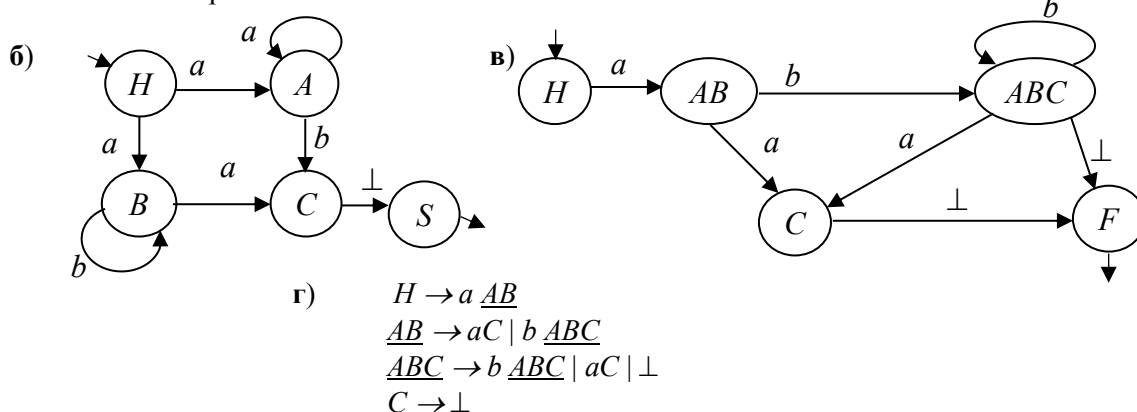
Критерии: За отсутствующий\неверный пункт 1-3 -3, за отсутствующий\неверный пункт 4 -1, нет\неверный пример -3.

6. (а) Не является приведенной, так как содержит недостижимый символ C . (Возможно указание других бесполезных символов, достаточно указать хотя бы один). G' : $S \rightarrow c$

(б) Является однозначной, так как в грамматике выводятся всего одна цепочка: 'c', для которой существует единственный вывод $S \rightarrow c$, а значит, единственное дерево вывода.

Критерии: неверный (необоснованный) ответ или ошибки в пункте (а): -5, неверный (необоснованный) ответ или ошибки в пункте (б): -5.

7. а) **Нет**, так как правила вывода содержат повторяющиеся правые части. Обоснование возможно также через автомат.



Критерии: Неверный ответ на вопрос (а): -5, нет обоснования: -3, неверный КА: -4, неверный ДКА: -5, неверная грамматика для ДКА: -2.

8. Ответ.

$$\begin{aligned} S' &\rightarrow S \mid \varepsilon \\ S &\rightarrow aSBb \mid aBb \mid aSb \mid ab \mid bA \mid B \\ A &\rightarrow aB \mid a \mid b \\ B &\rightarrow bB \mid b \end{aligned}$$

Критерии: Пропуск каждой альтернативы -5 (Пропуск первой строки -10).

9. **Нет**, так как пересечение множеств $first(bB) = \{b\}$ и $follow(B) = \{b\}$ непусто.

Критерии: Неверный ответ или неверное обоснование -10, неточности в множествах $first(A)$ и/или $follow(A)$: -2.

10. Ответ.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
x	0	>	23	!F	&y	y	3	x	+	+	=	;	y	x	<	21	!F	6	!

21	22	23	24	25	26	27
27	!	&y	x	=	;	>

`if (x > 0) do y = y + (3 + x) while (y < x); else y = x;`

Критерии: Есть ошибки в постфиксной записи: -5, есть ошибки в восстановленном фрагменте: -5. **Внимание:** скобки в выражении $y+(3+x)$ обязательны, без них постфиксная запись была бы другой. Если без ошибок только расставлены операции перехода, оценка **2 балла** за задачу.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. При компиляции функции `main()` следующей программы фиксируется ошибка. Объясните, в чем она заключается, и исправьте ее, модифицируя только структуру `B`, ничего в ней не удаляя.

```
struct A { int y;
        A (int n){ y = n;}
};
struct B:A {};
int main () { B b;
            cout << b.y << '\n';
            return 0;
}
```

2. Вычеркните неразрешимые вызовы функций `f()`. Что напечатает получившаяся программа?

```
struct A { int n; A (int m) { n = m; }
};
void f ( int i, char c ) { cout << "f(int, char)\n"; }
void f ( int i, int j ) { cout << "f(int, int)\n"; }
void f ( const char * p, char c ) { cout << "f(const char *, char)\n"; }
void f ( const char * p, A a ) { cout << "f(const char *, A)\n"; }
int main () { A a (1); f (0, a);
                f ('*', a);
                f (0, 'a');
                return 0; }
```

3. C++11. В каких ситуациях использование move-семантики (семантики переноса) позволяет повысить эффективность работы программы?

4. Описать **класс** `A` и необходимые функции так, чтобы при компиляции заданной функции `main()` не были обнаружены ошибки.

```
int main ( ) { A a (1), b (2), c (3);
            cout << (a += b += c) << endl;
            return 0;
}
```

5. Что напечатает программа?

```
struct B { virtual void f (int a) { g (); cout << a << " - B::f (int) \n"; }
        void g () { cout << "B::g () \n"; }
};
class D: public B {
        void f (int b = 7) { g (); cout << b << " - D::f (int) \n"; }
        virtual void g () { cout << "D::g () \n"; }
};
int main () { D d; B b, &rb = d;
            b. f (5);
            rb. f (6); return 0;
}
```

6. Что такое функция преобразования в языке Си++? Привести **пример** использования функции преобразования.

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct A {      A ()          { cout << "A_Constr\n"; }
              A (const A & a) { cout << "A_Copy\n";   }
              ~A ()          { cout << "A_Destr\n";   }
};
struct B : A {  B ()          { cout << "B_Constr\n"; }
              ~B ()          { cout << "B_Destr\n"; }
};
void g() { B bg; throw bg;}
int main () { try { A a; g(); }
             catch ( B &) { cout << "B&_Catch\n"; }
             catch ( A &) { cout << "A&_Catch\n"; }
             return 0;
}
```

8. Исправьте ошибки в нижеследующей программе, ничего в ней не удаляя. Объясните, в чем заключаются исправленные ошибки. Что напечатает программа после внесенных исправлений?

```
struct A { int y;
          static int f () { return (y = 7); }
};
struct B : A { static int f () { return 9; }
};
int main () { B b,d;
             b.y = b.f();
             cout << B::f() << ' ' << B::y << ' ' << d.y << A::f() << endl;
             return 0;
}
```

9. Модифицируйте программу, ничего в ней не удаляя, так, чтобы использование операции `dynamic_cast` было корректно, а программа в целом завершилась нормально (не аварийно).

```
struct B { void f () { cout << "B::f()\n"; }
};
struct D:B {};
int main () { D d, & rd = d;
             B b, & rb = b, & rbd = d;
             rd = dynamic_cast <D&> (rbd);  rd.f();
             rd = dynamic_cast <D&> (rb);   rd.f();
             return 0;
}
```

10. Написать **ОДНУ** функцию `beforelast()`, которая для своего аргумента-контейнера целочисленных элементов возвращает значение предпоследнего элемента, если таковое имеется, или 0. Использование `beforelast()` должно быть корректно в следующей функции:

```
int ss (const vector <int> & V, const list <int> & L){
    return beforelast(V) + beforelast(L);
}
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. При компиляции функции `main()` следующей программы фиксируется ошибка. Объясните, в чем она заключается, и исправьте ее, модифицируя только структуру `B`, ничего в ней не удаляя.

```
struct A { int x, y;
    A (int m, int n) { x = m; y = n; }
};
struct B {A a;};
int main () { B b;
    cout << b.a.y << '\n';
    return 0;
}
```

2. Вычеркните неразрешимые вызовы функций `f()`. Что напечатает получившаяся программа?

```
struct A { operator int () { return 1; }
};
void f (double d, char c) { cout << "f(double, char)\n"; }
void f (double d, int j) { cout << "f(double, int)\n"; }
void f (A a, const char * p) { cout << "f(A, const char *)\n"; }
void f (int i, const char * p) { cout << "f(int, const char *)\n"; }
int main () { A a; f (a, 0);
    f (a, 'a');
    f ('a', 0);
    return 0; }
```

3. C++11. Каково назначение служебного слова ***auto*** в C++11? Привести **пример** его использования.

4. Описать **класс** `B` и необходимые функции так, чтобы при компиляции заданной функции `main()` в ней не были обнаружены ошибки.

```
int main () { B b (1);
    cout << (b ++ ++ << endl;
    return 0;
}
```

5. Что напечатает программа?

```
struct B {
    virtual void f (int a = 0) { cout << a << "B::f (int) \n"; }
    void g () { f (); cout << "B::g () \n"; }
};
class D: public B {
    virtual void f () { cout << "D::f()"; }
    virtual void g () { f (); cout << "D::g()\n"; }
};
int main () { D d; B b, &rb = d;
    b. g ();
    rb. g (); return 0;
}
```

6. Что такое функция-друг в языке Си++? Привести пример описания и использования функции-друга.

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct A { A () { cout << "A_Constr\n"; }
          A (const A & a) { cout << "A_Copy\n"; }
          ~A () { cout << "A_Destr\n"; }
};
struct B : A { B () { cout << "B_Constr\n"; }
              B (const B & a) { cout << "B_Copy\n"; }
              ~B () { cout << "B_Destr\n"; }
};
void g() { B bg; throw bg;}
int main () { try { A a; g(); }
             catch (A &) { cout << "A&_Catch\n"; }
             catch (B &) { cout << "B&_Catch\n"; }
             return 0;
}
```

8. Исправьте ошибки в нижеследующей программе, ничего в ней не удаляя. Объясните, в чем заключаются исправленные ошибки. Что напечатает программа после внесенных исправлений?

```
struct A { int y; int f () { return (y = 2); }
};
struct B : A { static int g () { return 9 + f(); }
};
int A::y;
int main () { B b,d;
              b.y = b.f();
              cout << B::f() << ' ' << B::y << ' ' << d.y << B::g() << endl;
              return 0;
}
```

9. Модифицируйте программу, ничего в ней не удаляя, так, чтобы использование операции `dynamic_cast` было корректно, а программа в целом завершилась нормально (не аварийно).

```
struct B { void g1 () { cout << "B::g1()\n"; }
};
struct D:B {};
int main () { D d, * pd;
              B b, * pb = & b, * pbd = & d;
              pd = dynamic_cast <D*> (pbd); pd -> g1();
              pd = dynamic_cast <D*> (pb); pd -> g1();
              return 0;
}
```

10. Написать ОДНУ функцию `first_last()`, которая для своего аргумента-контейнера целочисленных элементов возвращает сумму первого и последнего элементов, если таковые имеются, или 0. Использование `first_last()` должно быть корректно в следующей функции:

```
int f2 (vector <int> & V, const list <int> & L ) {
    return first_last (V) + first_last (L);
}
```


NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. При компиляции функции `main()` следующей программы фиксируется ошибка. Объясните, в чем она заключается, и исправьте ее, модифицируя только структуру `B`, ничего в ней не удаляя.

```
struct A { int y;
        A (int n) { y = n; }
};
struct B:A { int x;
        B (int m) { x = m; }
};
int main () { B b(10);
             cout << b.y << '\n';
             return 0;
}
```

2. Вычеркните неразрешимые вызовы функций `f()`. Что напечатает получившаяся программа?

```
struct A { int n; A (int i) { n = i; }
};
void f (char a, char b) { cout << "f(char, char)\n"; }
void f (A a, A b) { cout << "f(A, A)\n"; }
void f (const int & r, char c) { cout << "f(const int &, char)\n"; }
void f (char c, const char * p) { cout << "f(char, const char *)\n"; }
int main () { A a (1); f ('a', 0);
                  f (a, 0);
                  f (1, 0);
                  return 0; }
```

3. C++11. Каково специальное назначение идентификатора `final` в C++11? Привести по одному примеру для каждого варианта его специального использования.

4. Описать **класс** `C` и необходимые функции так, чтобы при компиляции заданной функции `main()` в ней не были обнаружены ошибки.

```
int main () { C c (3);
             cout << -- (-- c) << endl;
             return 0;
}
```

5. Что напечатает программа?

```
struct B {
    virtual void f (float a) { g (); cout << a << " - B::f(float)\n"; }
    virtual void g () { cout << "B::g()\n"; }
};
class D: public B {
    virtual void f (double d) { cout << d << " - D::f(double) "; }
    virtual void g () { f (4); cout << "D::g()\n"; }
};
int main () { D d; B b, &rb = d;
             b. f (2);
             rb. f (3); return 0;
}
```

6. Что такое чистая виртуальная функция в языке Си++? Привести пример описания и использования такой функции.

7. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct A { A () { cout << "A_Constr\n"; }
          A (const A & a) { cout << "A_Copy\n"; }
          ~A () { cout << "A_Destr\n"; }
};
struct B : A { B () { cout << "B_Constr\n"; }
              B (const B & a) { cout << "B_Copy\n"; }
              ~B () { cout << "B_Destr\n"; }
};
void g() { B bg; throw &bg;}
void f() { A af; g();}
int main () { try { A a; f(); }
              catch (A *) { cout << "A*_Catch\n"; }
              catch (B *) { cout << "B*_Catch\n"; }
              return 0;
}
```

8. Исправьте ошибки в описаниях классов **A** и **B**, чтобы программа стала правильной. **Объясните**, в чем заключаются исправленные ошибки. **Что напечатает** программа после внесенных исправлений?

```
struct A {
    static int y;
    static int f() const { return y; }
};
struct B : A { virtual static int g() { return 4 * f(); }
};
int A::y = 3;
int main () { B b,d;
              b.y = b.f();
              cout << B::f() << ' ' << B::y << ' ' << d.y << B::g() << endl;
              return 0;
}
```

9. Модифицируйте программу, **ничего в ней не удаляя**, так, чтобы использование операции **dynamic_cast** было корректно, а программа в целом завершилась нормально (не аварийно).

```
struct B { void ttt () { cout << "B::ttt () \n"; }
};
struct D : B { };
int main () { D d, * pd1, * pd2;
              B b, * pb = &b, * pbd = &d;
              pd1 = dynamic_cast <D*> (pbd);
              pd2 = dynamic_cast <D*> (pb);
              if (typeid (*pd1) == typeid (*pd2)) pb -> ttt ();
              return 0;
}
```

10. Написать **ОДНУ функцию** `max_first_last()`, которая для своего аргумента-контейнера целочисленных элементов возвращает наибольшее из значений первого и последнего элементов (если таковые имеются) или 0. Использование `max_first_last()` должно быть **корректно** в следующей функции:

```
int f3 (const vector <int> & V, list <int> & L ) {
    return max_first_last (V) + max_first_last (L);
}
```

Коллоквиум, 2014 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Надо добавить в класс B, например, конструктор умолчания со списком инициализации для вызова конструктора класса A с одним параметром.

Критерии: Ошибка не найдена или неправильно объяснена =0. Неверное или с ошибками только исправление -5.

2. Неразрешимый вызов `f(' ', a)`. Будет напечатано: `f(const char*, A) f(int, char)`.

Критерии: За каждое неверное вычеркивание или неправильно вызванную функцию -5.

3. При копировании временных объектов неплюских классов.

Критерии: За отсутствие уточнения «неплюских» -5. В остальных случаях =0, но! копирование $\equiv$ работа конструктора копирования и\или операции = ...

4.

```
struct A {
    int n;
    A(int i) { n = i; } // не нужно перегружать <<
    A& operator += (A & a) { n += a.n; return * this; }
}
ostream & operator << (ostream & s, A& a) { s << a.n; return s; }
```

Критерии: За каждую неверную\отсутствующую операцию -5, за другие мелкие погрешности -2 за каждую. За `typedef int A` снижать до 0.

5. Ответ.

```
B::g()
5 - B::f(int)
D::g()
6 - D::f(int)
```

Критерии: Ошибки в ответе =0, иначе =10. Ошибки, не связанные с виртуальностью -2.

6. Функция преобразования – метод класса специального вида `operator <mun>()`, который задает правило приведения объектов класса к типу <mun>, например, “`operator int()`”. Алгоритм преобразования кодируется в теле функций преобразования. Пример:

```
struct S { long s; S() { s = 0L; }
    operator long () { return s; } // S --> long
};
long f(long l) { return l; }
S s; long nl = f(s); // Верно!
```

Критерии: За неправильное определение -5. За отсутствие\неверный пример -5.

7.

```
A_Constr
A_Constr
B_Constr
A_Copy
B_Destr
A_Destr
A_Destr
B& Catch
B_Destr
A_Destr
```

Критерии: За одну ошибку – 7, за две ошибки – 3, за большее количество ошибок =0. Если пропущен конструктор, за парный пропущенный деструктор не снижать.

8. Исправления: (1) перед описанием поля "y" вставить "**static**", (2) добавить описание статического поля вне класса: "**int A::y;**". **Объяснение:** статические функции не могут пользоваться нестатическими членами-данными.

Напечатается: 9 7 77.

Критерии: Не найдена ошибка – отсутствие "**static**" – или неверное объяснение =0. Пропущено описание статической переменной вне класса -5. Неверная печать -5. За указание ошибки там, где её нет -5.

9. Модификация: (1) Например, добавить "**virtual**" перед f () в классе B, (2) заключить в **try-catch** блок использование операций **dynamic_cast**, используя **catch**(bad_cast) {}

Критерии: За отсутствие каждого исправления и каждое лишнее исправление -5.

```
10. template <class T>
    int beforelast (T & t) { // const не мешает
        if (t.size() < 2)
            return 0;
        typename T::const_reverse_iterator ri = t.rbegin ();
        ri ++;
        return * ri;
    }
```

Критерии: За пропуск "**typename**" -5. За использование неконстантного итератора -5. За ошибки в алгоритме -5. За другие мелкие ошибки -2 за каждую.

Коллоквиум, 2014 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Надо добавить в класс B, например, конструктор умолчания со списком инициализации для вызова конструктора класса A с двумя параметрами.

Критерии: Ошибка не найдена или неправильно объяснена =0. Только исправление неверное или с ошибками -5.

2. Неразрешимые вызовы `f(a, 0)` и `f('a', 0)`. Будет напечатано: `f(double, char)`.

Критерии: За каждое неверное вычеркивание или неправильно вызванную функцию -5.

3. Описание явно инициализируемой переменной может содержать ключевое слово **auto**: при этом тип созданной переменной будет тип инициализирующего выражения. Например, `auto i = 7;`

Критерии: По сути неверный ответ =0. Отсутствие\неверный пример -5.

4. Ответ.

```
struct B { int n;
    B (int i) {n = i;}
    B& operator++(int) { B b = *this; n++; return b; } // Можно также вернуть int & на n!
} // Тогда не обязательно перегружать <<
ostream& operator<<(ostream&s, const B& a) { s << a.n; return s; }
```

Критерии: За каждую неверную\отсутствующую операцию -5, за другие мелкие погрешности -2 за каждую. За `typedef int A` снижать до 0.

5. Ответ. 0 - B::f(int)
 B::g()
 0 - B::f(int)
 B::g()

Критерии: Ошибки в ответе =0, иначе =10. Ошибки, не связанные с виртуальностью -2.

6. Функция-друг класса – внешняя по отношению к классу функция, которой разрешен доступ к закрытым и защищенным членам класса.

Пример: `class I { char * c; friend ostream& operator<<(ostream&s, I*i); };
ostream& operator<<(ostream &s, I*i) { s << i->c << endl; return s; }`

Критерии: Неверное определение =0. Неверный\отсутствует пример -5.

7. A_Constr
 A_Constr
 B_Constr
 A_Constr
 B_Copy
 B_Destr
 A_Destr
 A_Destr
 A& Catch
 B_Destr
 A_Destr

Критерии: За каждую неверную (лишнюю или пропущенную) печать -2.

8. Исправления: (1) перед описанием поля "y" вставить "**static**", (2) перед описанием функции $f()$ вставить "**static**". **Объяснение:** статические функции не могут пользоваться нестатическими членами класса.

Напечатается: 2 2 211.

Критерии: Не найдена ошибка – отсутствие "**static**" – за каждую -5. Неверная печать -5. За указание ошибки там, где её нет -5.

9. Модификация: (1) Например, добавить "**virtual**" перед `g1()` в классе B, (2) Вставить проверку на NULL возвращаемого результата операций (и) **dynamic_cast**.

Критерии: За отсутствие каждого исправления и каждое лишнее исправление -5.

10.

```
template <class T>
int first_last ( T & t){ // const не мешает
    if (t.size() == 0)
        return 0;
    typename T::const_iterator bi = t.begin(), ei = t.end();
    ei--;
    return *bi + *ei;
}
```

Критерии: За пропуск "**typename**" -5. За использование неконстантного итератора -5. За ошибки в алгоритме -5. За другие мелкие ошибки -2 за каждую.

Коллоквиум, 2014 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Надо добавить список инициализации в конструктор класса В для вызова конструктора класса А с одним параметром.

Критерии: Ошибка не найдена или неправильно объяснена =0. Неверное\с ошибками только исправление -5.

2. Неразрешимый вызов `f('a', 0)`. Будет напечатано: `f(A, A) f(const int&, char)`.

Критерии: За каждое неверное вычеркивание или неправильно вызванную функцию -5.

3. Идентификатор `final` обозначает следующее:

- **в описании классов** – то, что они не могут быть базовыми для новых классов наследственной иерархии, например: `struct F final { };`
- **в описании виртуальных функций** – то, что возможные производные классы (от рассматриваемого) не могут иметь виртуальные функции, которые бы замещали финальные функции, например: `virtual void g() final;`
- **в других позициях** это обычный идентификатор, подчиняющийся общим для них правилам: `static int final;`

Критерии: За **каждое** неверное объяснение использования -5. За **каждый** неверный\отсутствующий пример использования -3.

4.

```
struct C { int n; C (int i) { n = i; }  
      C& operator -- () { --n; return *this; } // можно также вернуть int & на n!, тогда  
      }                                     // тогда не обязательно перегружать <<  
ostream & operator << (ostream & s, C& a) { s << a.n; return s; }
```

Критерии: За каждую неверную\отсутствующую операцию -5, за другие мелкие погрешности -2 за каждую. За `typedef int A` снижать до 0.

5. Ответ.

```
B::g()  
2 - B::f(float)  
4 - D::f(double)  
D::g()  
3 - B::f(float)
```

Критерии: Ошибки в ответе =0, иначе =10. Ошибки, не связанные с виртуальностью -2.

6. Чистая виртуальная функция – виртуальный метод класса вида: `virtual <min><имя> () = 0`. В примере чистая виртуальная функция должна быть переопределена в производном классе.

Критерии: За неправильное определение =0. За отсутствие\неверный пример -5.

7.

```
A_Constr  
A_Constr  
A_Constr  
B_Constr  
B_Destr  
A_Destr  
A_Destr  
A_Destr  
A*_Catch
```

Критерии: За одну ошибку – 7, за две ошибки – 3, за большее количество ошибок =0. Если пропущен конструктор, за парный пропущенный деструктор не снижать.

8. Исправления: (1) в описании функции `f()` "убрать **const**", (2) в описании функции `g()` "убрать **virtual**". **Объяснение:** статические функции не могут быть виртуальными и константными.

Напечатается: 3 3 312.

Критерии: Не найдены ошибки – за каждую **-5**. Неверная печать **-5**. За указание ошибки там, где её нет **-5**.

9. Модификация: (1) Например, добавить **virtual** перед `ttt()` в классе B, (2) Заключить в **try-catch** блок условный оператор в функции `main()`, используя `catch (bad_cast) {}`

Критерии: За отсутствие каждого исправления и каждое лишнее исправление **-5**.

10.

```
template <class T>
int max_first_last (T & t){ // const не мешает
    if (t.size() == 0)
        return 0;
    typename T::const_iterator bi = t.begin(), ei = t.end();
    ei--;
    return * bi > * ei ? * bi : * ei;
}
```

Критерии: За пропуск **typename** **-5**. За использование неконстантного итератора **-5**. За ошибки в алгоритме **-5**. За другие мелкие ошибки **-2** за каждую.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow aS \mid Sb \mid aAb \mid \varepsilon$
 $A \rightarrow aaAbb \mid aabb$
 $A \rightarrow A$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы:

Ответ: $L(G) =$

(б) Каким из перечисленных классов грамматик принадлежит G ? **Ответ:**

Класс	П	$G \in \Pi?$ (да/нет)
регулярные		
контекстно-зависимые		
контекстно-свободные		
грамматики типа 0		
неукорачивающие		

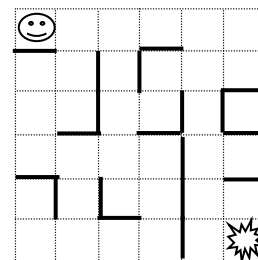
(в) Тип языка: найти такое целое k , что $L(G)$ является языком типа k , но не языком типа $k+1$.

Ответ: $k =$

2. Является ли однозначной данная грамматика G , порождающая язык цепочек в алфавите $\{0,1\}$, в которых символов 0 и 1 поровну? Ответ обосновать.

G : $S \rightarrow 0S1S \mid 1S0S \mid 01BB \mid \varepsilon$
 $B \rightarrow BB \mid B01$

3. Тесей должен пройти из левого верхнего угла лабиринта в правый нижний, чтобы убить Минотавра. Нить Ариадны оставляет след перемещений Тесея. Он умеет за один шаг перемещаться на клетку вниз (след обозначается символом a), вверх (след b) или вправо (след c). Влево не умеет. Каждая клетка посещается не более одного раза. Постройте грамматику, порождающую язык в алфавите $\{a, b, c\}$ всевозможных путей Тесея, приводящих к цели. Например, цепочка $ssssaaaaas$ приведет Тесея к Минотавру. В грамматике должно быть не более 4-х правил вывода, считая альтернативы.



4. Дан список слов: *в, и, исправление, обнаружение, ошибки, программа*. Составьте из него, употребив слова в нужном порядке и форме, определение термина (понятия) из раздела СП и назовите сам термин (он не входит в заданный список).

5. Можно ли считать утилиту *make* системы *Unix* компилятором? Обоснуйте ответ.

6. По заданной грамматике $G = \{\{a, b\}, \{B, S\}, P, S\}$ получить эквивалентную неукорачивающую контекстно-свободную грамматику (использовать алгоритм устранения правил с пустой правой частью).

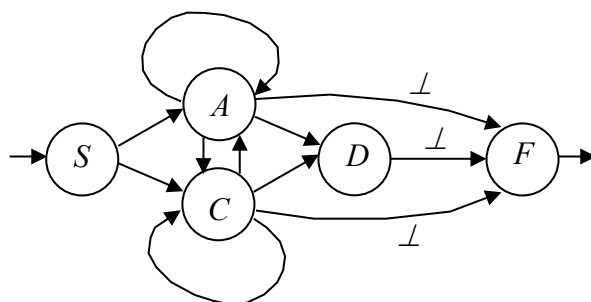
P : $S \rightarrow aBb \mid \varepsilon$
 $B \rightarrow aB \mid bS \mid SS$

Ответ:

7. Даны 1) функция-анализатор на языке Си++ для левостолбчатой грамматики G_L :

```
bool scan_G ()
{   enum state { H, B, S, ER }; // множество состояний
    state CS = H;                // CS — текущее состояние
    do { gc ();                  // считывает символ в глобальный объект
        switch (CS) { case H:    if (c == 'a') CS = B;
                                else CS = ER; break;
                      case B:    if (c == 'b');
                                else if (c == '⊥') CS = S;
                                else CS = ER; break;
                      }
        } while (CS != S && CS != ER);
    return CS == S;              // true, если CS != ER, иначе false
}
```

2) заготовка диаграммы состояний праволинейной грамматики G_R :



(а) Расставить в заготовке диаграммы для G_R терминальные символы так, чтобы грамматики G_R и G_L были эквивалентны.

(б) Восстановить грамматику G_R . **Ответ:**

(в) Сколько состояний будет иметь конечный автомат, полученный алгоритмом детерминизации диаграммы для G_R ?

Ответ: _____

8. Дана КС-грамматика G , порождающая язык L . Вставить в грамматику действия вида $\langle cout << "символы"; \rangle$ так, чтобы в процессе рекурсивного спуска был реализован перевод $\tau = \{(\omega, a^k b^{k+n}) \mid \omega \in L, k = |\omega|_a, n = |\omega|_b\}$.

$G: S \rightarrow aA \mid bB \mid \varepsilon$

$A \rightarrow cA \mid S$

$B \rightarrow cB \mid S$

9. Дана грамматика G . Докажите, что метод рекурсивного спуска неприменим к ней. Можно ли вычеркнуть одно вхождение терминального символа в правила грамматики так, чтобы к получившейся грамматике метод был бы применим?

$G: S \rightarrow aSb \mid Yb \mid bb$

$Y \rightarrow cSY \mid bd \mid \varepsilon$

10. Постройте постфиксную запись для фрагмента программы на Си. Префиксный ++ в постфиксной записи обозначается как +#, постфиксный как #+.

		<code>for (i = 0, j = 10; i == j ? 0 : 1; --j) a += 2 < i++ > j;</code>																
ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$S \rightarrow cA \mid cAb \mid cb \mid \varepsilon$
 $cAb \rightarrow cAbb \mid cb \mid ccAb$
 $ccAbb \rightarrow ccAbb$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы:

Ответ: $L(G) =$

(б) Каким из перечисленных классов грамматик принадлежит G ? Ответ:

Класс	П	$G \in \text{П?}$ (да/нет)
регулярные		
контекстно-зависимые		
контекстно-свободные		
грамматики типа 0		
неукорачивающие		

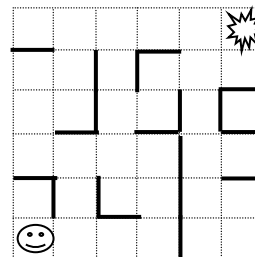
(в) Тип языка: найти такое целое k , что $L(G)$ является языком типа k , но не языком типа $k+1$.

Ответ: $k =$

2. Является ли однозначной данная грамматика G , порождающая язык цепочек в алфавите $\{a, b\}$? Ответ обосновать.

G : $S \rightarrow aSaS \mid aBa \mid aSa \mid A$
 $B \rightarrow BB \mid Ba$
 $A \rightarrow b \mid \varepsilon$

3. Баба-Яга дала Иванушке клубочек, который через лесной лабиринт проведет его к сундуку, где находится смерть Кощеева. Клубочек (а за ним и Иванушка) умеет за один шаг перемещаться на клетку вниз (оставляемый след обозначается символом a), вверх (след b) или вправо (след c). Влево не умеет. Каждая клетка посещается не более одного раза. Постройте грамматику, порождающую язык в алфавите $\{a, b, c\}$ всевозможных путей Иванушки, приводящих к цели. Иванушка начинает путь из левого нижнего угла. Цель – в правом верхнем. Например, цепочка $cbbcbbbbcc$ приведет Иванушку к цели. В грамматике должно быть не более 4-х правил вывода, считая альтернативы.



4. Дан список слов: *входные, заранее, данные, для, известный, программа, работа, результат, с*. Составьте из него, употребив слова в нужном порядке и форме, определение термина (понятия) из раздела СП и назовите сам термин (он не входит в заданный список).

5. Можно ли считать утилиту *make* системы *Unix* интерпретатором? Обоснуйте ответ.

6. По заданной грамматике $G = \{\{a, c\}, \{B, S\}, P, S\}$ получить эквивалентную неукорачивающую контекстно-свободную грамматику (использовать алгоритм устранения правил с пустой правой частью).

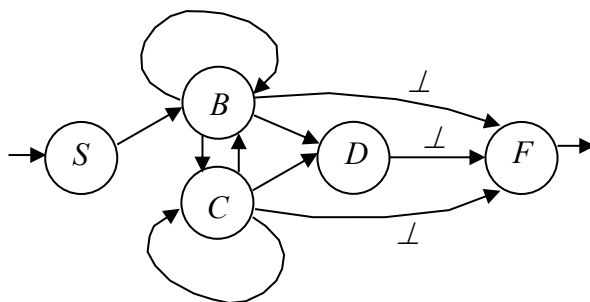
P : $S \rightarrow aBc \mid \varepsilon$
 $B \rightarrow aB \mid cS \mid SaS$

Ответ:

7. Даны 1) функция-анализатор на языке Си++ для левостолбчатой грамматики G_L :

```
bool scan_G ()
{
    enum state { H, A, S, ER }; // множество состояний
    state CS = H;               // CS — текущее состояние
    do { gc ();                  // считывает символ в глобальный объект 'c'
        switch (CS) { case H:    if ( c == 'a' || c == 'b' ) CS = A;
                                else CS = ER; break;
                      case A:    if ( c == 'a' );
                                else if ( c == '1' ) CS = S;
                                else CS = ER; break;
                      }
    } while (CS != S && CS != ER);
    return CS == S;              // true, если CS != ER, иначе false
}
```

2) заготовка диаграммы состояний праволинейной грамматики G_R :



(а) Расставить в заготовке диаграммы для G_R терминальные символы так, чтобы грамматики G_R и G_L были эквивалентны.

(б) Восстановить грамматику G_R .
Ответ:

(в) Сколько состояний будет иметь конечный автомат, полученный алгоритмом детерминизации диаграммы для G_R ? **Ответ:** _____

8. Дана КС-грамматика G , порождающая язык L . Вставить в грамматику действия вида $\langle \text{cout} << \text{"символы"}; \rangle$ так, чтобы в процессе рекурсивного спуска был реализован перевод $\tau = \{(\omega, a^k d^{k-n}) \mid \omega \in L, k = |\omega|, n = |\omega|_d\}$.

G :

$S \rightarrow aA \mid dD \mid \varepsilon$

$A \rightarrow cA \mid S$

$D \rightarrow cD \mid S$

9. Дана грамматика G . Докажите, что метод рекурсивного спуска неприменим к ней. Можно ли вычеркнуть одно вхождение терминального символа в правила грамматики так, чтобы к получившейся грамматике метод был бы применим?

$G: S \rightarrow aSb \mid Xa \mid bb$

$X \rightarrow cSX \mid bd \mid \varepsilon$

10. Постройте постфиксную запись для фрагмента программы на Си. Префиксный ++ в постфиксной записи обозначается как +#, постфиксный как #+.

```
i = 0, j = 10; do a += 2 < i++ > --j; while (i == j ? 0 : 1);
```

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17				
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Дана грамматика G :

$$\begin{aligned} S &\rightarrow Sc \mid aSc \mid \varepsilon \mid aAc \\ aAc &\rightarrow aEAc \mid aac \\ aE &\rightarrow Ea \\ E &\rightarrow a \end{aligned}$$

(а) Описать язык $L(G)$ в виде теоретико-множественной формулы:

Ответ: $L(G) =$

(б) Каким из перечисленных классов грамматик принадлежит G ? Ответ:

Класс П	$G \in \Pi?$ (да/нет)
регулярные	
контекстно-зависимые	
контекстно-свободные	
грамматики типа 0	
неукорачивающие	

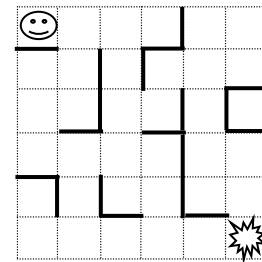
(в) Тип языка: найти такое целое k , что $L(G)$ является языком типа k , но не языком типа $k+1$.

Ответ: $k =$

2. Является ли однозначной данная грамматика G , порождающая язык цепочек в алфавите $\{b, c\}$? Ответ обосновать.

G :
$$\begin{aligned} S &\rightarrow ASS \mid SSA \mid B \mid A \\ B &\rightarrow bB \mid cB \mid \varepsilon \end{aligned}$$

3. Колобок должен пройти через лесной лабиринт из левого верхнего угла в правый нижний, чтобы спеть песенку лисе. Он умеет за один шаг перемещаться на клетку вниз (след обозначается символом a), вверх (след b) или вправо (след c). Влево не умеет. Каждая клетка посещается не более одного раза. Постройте грамматику, порождающую язык в алфавите $\{a, b, c\}$ всевозможных путей колобка, приводящих к лисе. Например, цепочка $ssaaaaass$ приведет колобка к цели. В грамматике должно быть не более 4-х правил вывода, считая альтернативы.



4. Дан список слов: *на, правильность, проверка, программа, помощь, с, тесты*. Составьте из него, употребив слова в нужном порядке и форме, определение термина (понятия) из раздела СП и назовите сам термин (он не входит в заданный список).

5. Можно ли язык, на котором записывается задание для утилиты *make* системы *Unix*, считать интерпретируемым языком? Обоснуйте ответ.

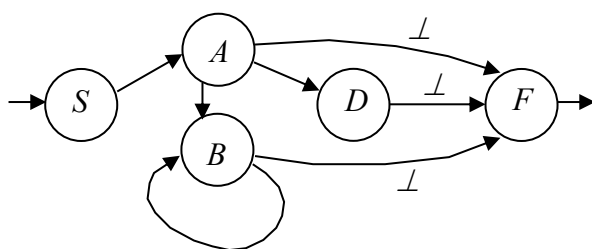
6. По заданной грамматике $G = \{\{d, b\}, \{B, S\}, P, S\}$ получить эквивалентную неукорачивающую контекстно-свободную грамматику (использовать алгоритм устранения правил с пустой правой частью).

P :
$$\begin{aligned} S &\rightarrow dBb \mid S \mid \varepsilon \\ B &\rightarrow dB \mid bS \mid SSd \end{aligned}$$
 Ответ:

7. Даны 1) функция-анализатор на языке Си++ для левوليной грамматики G_L :

```
bool scan_G ()
{
    enum state { H, C, S, ER }; // множество состояний
    state CS = H;               // CS — текущее состояние
    do { gc ();                  // считывает символ в глобальный объект c
        switch (CS) { case H:    if ( c == 'a' ) CS = C;
                               else CS = ER; break;
                      case C:    if ( c == 'c' );
                               else if ( c == 'l' ) CS = S;
                               else CS = ER; break;
                      }
    } while (CS != S && CS != ER);
    return CS == S;              // true, если CS != ER, иначе false
}
```

2) заготовка диаграммы состояний праволинейной грамматики G_R :



(а) Расставить в заготовке диаграммы для G_R терминальные символы так, чтобы грамматики G_R и G_L были эквивалентны.

(б) Восстановить грамматику G_R .

Ответ:

(в) Сколько состояний будет иметь конечный автомат, полученный алгоритмом детерминизации диаграммы для G_R ? **Ответ:** _____

8. Дана КС-грамматика G , порождающая язык L . Вставить в грамматику действия вида $\langle cout << "символы"; \rangle$ так, чтобы в процессе рекурсивного спуска был реализован перевод $\tau = \{(\omega, a^{2k}b^{k-n}) \mid \omega \in L, k = |\omega|, n = |\omega|_a\}$.

G :

$S \rightarrow aA \mid bB \mid \varepsilon$

$A \rightarrow cA \mid S$

$B \rightarrow cB \mid S$

9. Дана грамматика G . Докажите, что метод рекурсивного спуска неприменим к ней. Можно ли вычеркнуть одно вхождение терминального символа в правила грамматики так, чтобы к получившейся грамматике метод был бы применим?

G : $S \rightarrow aSb \mid YD \mid db$

$Y \rightarrow cSY \mid bd \mid \varepsilon$

$D \rightarrow dd$

10. Постройте постфиксную запись для фрагмента программы на Си. Префиксный ++ в постфиксной записи обозначается как #+, постфиксный как #+.

`i = 0, j = 10; while (i == j ? 0 : 1) a += 2 < ++i > j--;`

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17				
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

Экзамен, 2014 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{a^n b^m \mid m, n \geq 0\}$

(б) регулярные – "нет" (-2), КЗ – "нет" (-2), КС – "да" (-4), грамматики типа 0 – "да" (-4), неукорачивающие – "да" (-2).

(в) $k = 3$ (для заданного языка существует порождающая его регулярная грамматика)

Критерии: ошибка в формуле для (а): -4, за каждую неверную строчку в (б): см. ответ, ошибка в (в): -3.

2. Ответ: Грамматика не приведенная. Символ B и правила его содержащие – бесполезны и не участвуют в выводах терминальных цепочек из S , а значит не влияют на однозначность. Их можно удалить: $S \rightarrow 0S1S \mid 1S0S \mid \varepsilon$.

Имеются два различных левых вывода для цепочки '0101':

$$\begin{aligned} S &\rightarrow 0S1S \rightarrow 01S \rightarrow 010S1S \rightarrow 0101S \rightarrow 0101 & \text{и} \\ S &\rightarrow 0S1S \rightarrow 01S0S1S \rightarrow 010S1S \rightarrow 0101S \rightarrow 0101 \end{aligned}$$

Следовательно, грамматика неоднозначна.

Можно также привести два различных дерева вывода или два различных правых вывода для цепочки '0101'. Можно найти и другие цепочки, демонстрирующие нарушение определения однозначности.

Критерии: неверный ответ или отсутствует верное обоснование: =0.

3. Ответ: Существуют всего четыре возможных пути при заданных ограничениях: $ssssaaaaas$, $ssssaaaasa$, $ssaaacbsaaaaas$, $ssaaacbsaaaasa$. Поэтому возможна следующая регулярная грамматика:

$$S \rightarrow sssssaaaaas \mid sssssaaaasa \mid ssaaacbsaaaaas \mid ssaaacbsaaaasa$$

Возможны и другие варианты, например: $S \rightarrow sssssaaaaA \mid ssaaacbsaaaA \quad A \rightarrow as \mid sa$

Критерии: за каждый неверный (отсутствующий) путь в порождаемом языке: -5.

4. Ответ: Отладка – обнаружение и исправление ошибок в программе.

Критерии: неверный ответ: =0.

5. Ответ: *нет*. Утилита *make* не переводит текст из *Makefile* в исполняемый файл, а интерпретирует его.

Критерии: неверный (не обоснованный) ответ: =0.

6. Ответ: $S' \rightarrow S \mid \varepsilon \quad S \rightarrow aBb \mid ab \quad B \rightarrow a \mid aB \mid bS \mid b \mid SS \mid S$

Критерии: за каждое лишнее (отсутствующее) правило: -5.

7. Ответ: Восстановленные грамматики

$$\begin{aligned} (1) G_L: \quad S &\rightarrow B \perp & (26) G_R: \quad S &\rightarrow aA \mid aC \\ B &\rightarrow Bb \mid a & A &\rightarrow bA \mid bD \mid bC \mid \perp \\ & & C &\rightarrow bC \mid bD \mid bA \mid \perp \\ & & D &\rightarrow \perp \end{aligned}$$

Соответствующий язык: $L(G_L) = \{a b^m \perp \mid m \geq 0\}$

(2а) Переход из S в A или C возможен только по a согласно формуле языка. Далее все незаполненные промежуточные дуги помечаются символом b , так как a стоит только в начале цепочки, а $\perp$ – только в конце.

(2в) Число состояний автомата, полученного детерминизацией диаграммы для $G_R = 4$.

Состояния ДКА: $\{S\}, \{A, C\}, \{A, C, D\}, \{F\}$

Критерии: (а) нет эквивалентности: -3 (пустых переходов не должно быть, в определении КА их нет), (б) восстановленная G_R не соответствует диаграмме студента (сама диаграмма может быть ошибочной, этот пункт оценивается независимо): -4, (в) неверное число: -3 (обоснование не обязательно).

8. Ответ: $S \rightarrow a\langle a \rangle A\langle b \rangle \mid bB\langle b \rangle \mid \varepsilon$ $S \rightarrow aA\langle b \rangle \mid bB\langle b \rangle \mid \varepsilon$ $S \rightarrow aA \mid bB \mid \varepsilon$ Есть еще варианты.

$$A \rightarrow cA \mid S$$

$$A \rightarrow cA \mid \langle a \rangle S$$

$$A \rightarrow cA \mid \langle a \rangle S\langle b \rangle$$

$$B \rightarrow cB \mid S$$

$$B \rightarrow cB \mid S$$

$$B \rightarrow cB \mid S\langle b \rangle$$

Запись $\langle cout < < "a"; \rangle$ сокращается до $\langle a \rangle$

Критерии: за любую ошибку: =0.

9. Ответ: $first(Yb) \cap first(bb) = \{b\} \neq \emptyset$. Нет, не применим: какой бы символ ни вычеркнуть, метод будет неприменим. Обоснование не обязательно. Замечание: Пустая цепочка, обозначенная ε , не является терминальным символом – ее вычеркивать нельзя.

Критерии: ошибочное или отсутствующее доказательство: -7, ошибочный или отсутствующий ответ на второй вопрос: -3.

10. Ответ.

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	&i	0	=	;	&j	10	=	;	i	j	==	17	!F	0	18	!	1

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
36	!F	26	!	&j	-#	9	!	&a	2	&i	#+	<	j	>	+=	22	!

Критерии: за каждую ошибку: -4.

Экзамен, 2014 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{c^n b^m \mid m, n \geq 2\} \cup \{c^k b^k \mid 0 \leq k \leq 1\}$ или $\{c^n b^m \mid m, n \geq 2\} \cup \{\varepsilon\} \cup \{cb\}$.

(б) регулярные – "нет" (-2), КЗ – "нет" (-2), КС – "нет" (-2),

грамматики типа 0 – "да" (-4), неукорачивающие – "нет" (-2).

(в) $k = 3$ (для заданного языка существует порождающая его регулярная грамматика)

Критерии: ошибка в формуле для (а): -4, за каждую неверную строчку в (б): см. ответ, ошибка в (в): -3.

2. Грамматика не приведенная. Символ B и правила его содержащие - бесполезны и не участвуют в выводах терминальных цепочек из S , а значит не влияют на однозначность. Их можно удалить:

$S \rightarrow aSaS \mid aSa \mid A \quad A \rightarrow b \mid \varepsilon$

Имеются два различных левых вывода для aa :

$S \rightarrow aSaS \rightarrow aAaS \rightarrow aaS \rightarrow aaA \rightarrow aa$ и $S \rightarrow aSa \rightarrow aAa \rightarrow aa$

Следовательно, грамматика неоднозначна.

Можно также привести два различных дерева вывода или два различных правых вывода для цепочки aa . Можно найти и другие цепочки, демонстрирующие нарушение определения однозначности.

Критерии: неверный ответ или отсутствует верное обоснование: =0.

3. Ответ: Существуют всего четыре возможных пути при заданных ограничениях: $cbbcbbccccc$, $cbbcbbccacab$, $cbbcbbcccb$, $cbbcbbccac$. Поэтому возможна следующая регулярная грамматика:

$S \rightarrow cbbcbccccc \mid cbbcbccacab \mid cbbcbcccb \mid cbbcbccac$

Критерии: за каждый неверный (отсутствующий) путь в порождаемом языке: -5.

4. Ответ: Тест – входные данные для программы с заранее известным результатом работы.

Критерии: неверный ответ: =0.

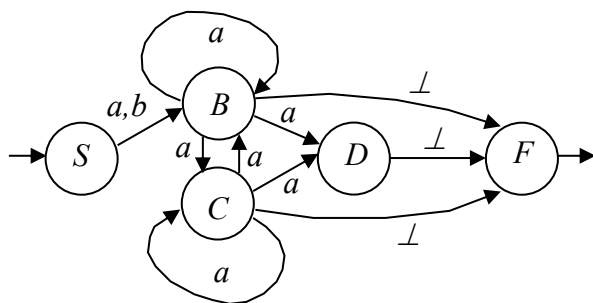
5. Ответ: да. Утилита *make* интерпретирует текст из *Makefile*, запуская необходимые процессы аналогично командному интерпретатору *shell*.

Критерии: неверный (не обоснованный) ответ: =0.

6. Ответ: $S' \rightarrow S \mid \varepsilon \quad S \rightarrow aBc \quad B \rightarrow aB \mid cS \mid c \mid aS \mid Sa \mid SaS \mid a$

Критерии: за каждое лишнее (отсутствующее) правило: -5.

7. Ответ: (2) Диаграмма состояний праволинейной грамматики G_R с (а) расставленными терминальными символами. Грамматики G_R и G_L эквивалентны.



(б) Ответ:

$S \rightarrow aB \mid bB$

$B \rightarrow aB \mid aC \mid aD \mid \perp$

$C \rightarrow aC \mid aD \mid aB \mid \perp$

$D \rightarrow \perp$

(в) Ответ: 4

Состояния ДКА: $\{S\}, \{B\}, \{B,C,D\}, \{F\}$

Критерии: (а) нет эквивалентности: -3 (пустых переходов не должно быть, в определении КА их нет), (б) восстановленная G_R не соответствует диаграмме студента (сама диаграмма может быть ошибочной, этот пункт оценивается независимо): -4, (в) неверное число: -3 (обоснование не обязательно).

8. Ответ: $S \rightarrow a\langle a \rangle A\langle d \rangle \mid d\langle a \rangle D \mid \varepsilon$ $S \rightarrow aA\langle d \rangle \mid d\langle a \rangle D \mid \varepsilon$ Есть еще варианты.
 $A \rightarrow c\langle a \rangle A\langle d \rangle \mid S$ $A \rightarrow c\langle a \rangle A \langle d \rangle \mid S$
 $D \rightarrow c\langle a \rangle D\langle d \rangle \mid S$ $D \rightarrow c\langle a \rangle D\langle d \rangle \mid \langle a \rangle S$

Запись `<cout << "a";>` сокращается до `<a>`

Критерии: за любую ошибку: **=0**.

9. Ответ: $first(Xa) \cap first(aSb) = \{a\} \neq \emptyset$. **Нет, не применим:** какой бы символ ни вычеркнуть, метод будет неприменим. Обоснование не обязательно. **Замечание:** Пустая цепочка, обозначаемая ε , не является терминальным символом – ее нельзя вычеркивать.

Критерии: ошибочное или отсутствующее доказательство: **-7**, ошибочный или отсутствующий ответ на второй вопрос: **-3**.

10. Постройте постфиксную запись для фрагмента программы на Си. Префиксный `++` в постфиксной записи обозначается как `+#`, постфиксный как `#+`.

`i = 0, j = 10; do a+=2 < i++ >--j; while (i == j ? 0:1);`

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ	&i	0	=	;	&j	10	=	;	&a	2	&i	#+	<	&j	-#	>	+=

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
;	i	j	==	27	!F	0	28	!	1	32	!F	9	!	}

Критерии: за каждую ошибку: **-4**.

Экзамен, 2014 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) $L(G) = \{ a^n c^m \mid m, n \geq 1 \} \cup \{ c^k \mid k \geq 0 \}$

(б) регулярные – "нет" (-2), КЗ – "нет" (-2), КС – "нет" (-2), грамматики типа 0 – "да" (-4), неукорачивающие – "нет" (-2).

(в) $k = 3$ (для заданного языка существует порождающая его регулярная грамматика)

Критерии: ошибка в формуле для (а): -4, за каждую неверную строчку в (б): см. ответ, ошибка в (в): -3.

2. Ответ: Грамматика не приведенная. Символ A и правила его содержащие – бесполезны и не участвуют в выводах терминальных цепочек из S , а значит не влияют на однозначность. Их можно удалить:

$$S \rightarrow B \quad B \rightarrow bB \mid cB \mid \varepsilon$$

Правило $S \rightarrow B$ применяется в любом выводе на первом шаге, а на следующих шагах работают правила из подграмматики $B \rightarrow bB \mid cB \mid \varepsilon$. Данная подграмматика автоматная праволинейная и ей соответствует детерминированная диаграмма состояний, которая «строит» левый вывод единственным способом. Следовательно, подграмматика $B \rightarrow bB \mid cB \mid \varepsilon$ однозначна, и исходная грамматика тоже однозначна. Другое обоснование: к преобразованной приведенной грамматике применим метод рекурсивного спуска, следовательно, она однозначна (РС-метод неприменим к неоднозначным грамматикам).

Возможны и другие неформальные рассуждения, подтверждающие единственность вывода в данной грамматике для любой цепочки, а следовательно, единственность левого, правого выводов и дерева вывода.

Критерии: неверный ответ или отсутствует верное обоснование: =0.

3. Ответ: Существуют всего четыре возможных пути при заданных ограничениях: $ssaaasacass$, $ssaaaacass$, $ssaacbsaaasa$, $ssaacbsaasaa$. Поэтому возможна следующая регулярная грамматика:
 $S \rightarrow ssaasacass \mid ssaasacass \mid ssaacbsaaasa \mid ssaacbsaasaa$

Критерии: за каждый неверный (отсутствующий) путь в порождаемом языке: -5.

4. Ответ: Тестирование – проверка программы на правильность с помощью тестов.

Критерии: неверный ответ: =0.

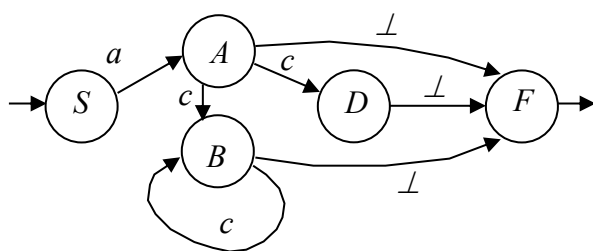
5. Ответ: да. Утилита *make* интерпретирует текст на языке для *Makefile*, запуская необходимые процессы аналогично командному интерпретатору *shell*.

Критерии: неверный (не обоснованный) ответ: =0.

6. Ответ: $S' \rightarrow S \mid \varepsilon \quad S \rightarrow dBb \mid S \quad B \rightarrow dB \mid bS \mid b \mid d \mid Sd \mid SSd$

Критерии: за каждое лишнее (отсутствующее) правило: -5.

7. (2) Диаграмма состояний праволинейной грамматики G_R с (а) расставленными терминальными символами. Грамматика G_R и G_L эквивалентны.



(б) Ответ:

$S \rightarrow aA$
 $A \rightarrow cB \mid cD \mid \varepsilon$
 $B \rightarrow cB \mid \varepsilon$
 $D \rightarrow \varepsilon$

(в) Ответ: 5.

Состояния ДКА: $\{S\}, \{A\}, \{B,D\}, \{B\}, \{F\}$

Критерии: (а) нет эквивалентности: -3 (пустых переходов не должно быть, в определении КА их нет), (б) восстановленная G_R не соответствует диаграмме студента (сама диаграмма может быть ошибочной, этот пункт оценивается независимо): -4, (в) неверное число: -3 (обоснование не обязательно).

8. Ответ: $S \rightarrow a\langle aa \rangle A \mid b\langle aa \rangle B\langle b \rangle \mid \varepsilon \quad S \rightarrow aA \mid b\langle aa \rangle B\langle b \rangle \mid \varepsilon$ Есть еще варианты.

$$\begin{array}{l|l} A \rightarrow c\langle aa \rangle A\langle b \rangle \mid S & A \rightarrow c\langle aa \rangle A\langle b \rangle \mid \langle aa \rangle S \\ B \rightarrow c\langle aa \rangle B\langle b \rangle \mid S & B \rightarrow c\langle aa \rangle B\langle b \rangle \mid S \end{array}$$

Запись $\langle cout < < "a"; \rangle$ сокращается до $\langle a \rangle$

Критерии: за любую ошибку: =0.

9. Дана грамматика G . Докажите, что метод рекурсивного спуска неприменим к ней. Можно ли вычеркнуть один терминальный символ в правилах грамматики так, чтобы к получившейся грамматике метод был бы применим?

$$\begin{array}{l} G: \quad S \rightarrow aSb \mid YD \mid db \\ \quad Y \rightarrow cSY \mid bd \mid \varepsilon \\ \quad D \rightarrow dd \end{array}$$

Ответ: $first(YD) \cap first(db) = \{d\} \neq \emptyset$. Нет: какой бы символ ни вычеркнуть, метод будет неприменим. Обоснование не обязательно. **Замечание:** Пустая цепочка, обозначаемая ε , не является терминальным символом – ее нельзя вычеркивать.

Критерии: Ошибочное или отсутствующее доказательство: -7, ошибочный или отсутствующий ответ на второй вопрос: -3.

10. Ответ.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ	&i	0	=	;	&j	10	=	;	i	j	==	17	!F	0	18	!	1

18	19	20	21	22	23	24	25	26	27	28	29	30	31	
32	!F	&a	2	&i	+#	<	&j	#-	>	+=	;	9	!	}

Критерии: за каждую ошибку: -4.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Приведите фрагмент программы с примером *ошибочного* обращения к видимой, но недоступной функции.

2. Что будет напечатано данной программой? Можно ли удалить хотя бы одну операцию разрешения области видимости так, чтобы печать не изменилась?

```

namespace N { int f= 2;
              int g=-2;
}
namespace M { int f= 3;
              int g=-3;
              int h=0;
}
int f = -5;

int main() { int f = 1;
            cout << ::f << N::f << M::f << M::h << endl;
            using namespace N;
            cout << f << g << M::h << endl;
            using namespace M;
            cout << f << M::g << h << endl;
            return 0;
}
    
```

3. Можно ли продемонстрировать параметрический полиморфизм с помощью оператора-выражения `a+(b+a)` ;? Обоснуйте ответ. Перечислите другие виды полиморфизма.

4. Данная программа выводит на печать несколько строк. Вычеркните *фрагменты* из функции `main()` так, чтобы напечаталось верное утверждение об ООП и C++.

```

class A {
public:
    A() { cout<<"абстрактный "; }
    virtual ~A() {cout<< endl;}
    void t(int n){cout<<"конструктор ";}
    virtual void t() {cout<<"метод ";}
};
class B: public A {
public:
    B() {}
    B(int x) { cout << "класс ";}
    void t() { cout << "тип "; }
    void t(int n) { cout<<"интерфейс ";}
    ~B() {cout<< "данных";}
};

int main() {
    cout<<"класс, не содержащий \
открытых членов-данных, - это ";
    A * qa= new A;
    qa -> t(5);
    delete qa;
    qa = new B;
    qa -> A::t();
    delete qa;
    qa = new B;
    { A & ra= * qa;
      ra.t();
    }
    delete qa;
    B * qb = new B;
    qb -> t(5);
    delete qb;
    B b(5);
}
    
```

5. Может ли одна и та же функция одновременно участвовать в перегрузке (*overloading*), скрытии (*hiding*) и замещении (*overriding*) ? Обоснуйте ответ.

Примечание: "замещение" проявляется при работе механизма виртуальных функций.

6. Добавьте реализацию метода `sum` (из класса `C`), который вычисляет сумму всех числовых полей объекта класса `C` (каждое поле учитывается в сумме один раз). Операцию `::` можно использовать не более одного раза в каждом первичном выражении (то есть `Y::x` возможно, `Z::Y::x` – нет).

Примечание: у операции `->` приоритет выше, чем у операции приведения типа `(тип) x`.

<pre>struct Base { int a; }; struct B1: virtual Base { int a; };</pre>	<pre>struct A1:Base { int a; }; struct B2: virtual Base { int a; };</pre>	<pre>struct A2:Base { int a; }; struct C: A1, A2, B1, B2 { int sum(); int a; };</pre>

7. Что будет напечатано в результате выполнения программы? Компилятор **не** оптимизирующий.

<pre>class A { static int c; int n; public: A():n(++c){ cout<<" A()="<< n; } A(const A&a):n(++c){ cout<<" A(A&)="<< n; } ~A(){ cout<<" ~A()="<< n; } };</pre>	<pre>void f(){ try { throw A(); } throw 1; } catch (int) { cout << "int"; } catch (A a) { throw; } cout << " f() "; } int A::c = 0; int main() { try { try { f(); } cout << " after_f "; } catch (A & a) { throw a; } catch (...) { cout << "inner ..."; } } catch (...) { cout << " ... "; } }</pre>

8. Опишите прототипы двух перегруженных функций `f()` из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
f (-6);
f ('+', 6);
f ();
f ("abc");
```

9. Объясните, какой смысл имеет выражение `typeid(*p)`.

10. Перепишите данный фрагмент, не используя средства из C++11:

```
long m;
decltype(m) n = 50000;
vector<long> v;
...
for (auto p=v.begin(); p!=v.end(); ++p) { n += * p;}
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Приведите пример *ошибочного* обращения к доступному, но не видимому (без операции `": : "`) методу класса.

2. Что будет напечатано данной программой? Можно ли удалить хотя бы одну операцию разрешения области видимости так, чтобы печать не изменилась?

```

namespace M {
    int f() { return 2; }
    int g = -2;
}
namespace N {
    int f() { return 3; }
    int g = -3;
    int h = 0;
}

int f() { return -7; }
int main() { int g = -1;
    cout << f() << N::f() << M::f() << N::h << endl;
    using namespace N;
    cout << N::f() << g << h << endl;
    using namespace M;
    cout << M::f() << M::g << h << endl;
    return 0;
}
    
```

3. Можно ли продемонстрировать параметрический полиморфизм с помощью оператора-выражения `(b+a)+b`? Обоснуйте ответ. Перечислите другие виды полиморфизма.

4. Данная программа выводит на печать несколько строк. Вычеркните *фрагменты* из функции `main()` так, чтобы напечаталось верное утверждение об ООП и C++.

```

class A {
public:
    A() { cout<<"абстрактный "; }
    virtual ~A() {cout<< endl;}
    void g(int n) {cout<<"класс ";}
    virtual void g() {cout<<"метод ";}
};

class B : public A {
public: B(){}
    B(int x) {cout<<"деструктор ";}
    void g() { cout<<"конструктор ";}
    void g(int n) {cout<<"интерфейс ";}
    ~B() {cout<< "данных";}
};

int main() {
    cout<<" класс, содержащий чистую \
    виртуальную функцию, - это ";
    A * qa = new B;
    qa -> A::g();
    delete qa;
    qa = new B;
    { A & ra = *qa;
      ra.g();
    }
    delete qa;
    B * qb = new B;
    qb -> g(7);
    delete qb;
    qa = new A;
    qa -> g(7);
    delete qa;
    B b(7);
}
    
```

5. Может ли функция одновременно участвовать в скрытии (*hiding*) и замещении (*overriding*) и при этом не участвовать в перегрузке (*overloading*)? Обоснуйте ответ.

Примечание: "замещение" проявляется при работе механизма виртуальных функций.

6. Добавьте реализацию метода `prod` (из класса `C`), который вычисляет произведение всех числовых полей объекта класса `C` (каждое поле учитывается в произведении один раз). Операцию `::` можно использовать не более одного раза в каждом первичном выражении (то есть `Y::X` — возможно, `Z::Y::X` — нет).

Примечание: у операции `"->"` приоритет выше, чем у операции приведения типа `"(тип) x"`.

<pre>class Base { public: int a; }; class B1: virtual public Base { public: int a; }; </pre>	<pre>class A1: public Base { public: int a; }; class B2: virtual public Base { public: int a; }; </pre>	<pre>class A2 : public Base { public: int a; }; class C: public A1, public A2, public B1, public B2 { public: int prod(); int a; }; </pre>

7. Что будет напечатано в результате выполнения программы? Компилятор *не* оптимизирующий.

<pre>class A { static int c; int n; public: A():n(++c){ cout<<" A()="<< n; } A(const A&a):n(++c){ cout<<" A(A&)="<< n; } ~A(){ cout<<" ~A()="<< n; } }; </pre>	<pre>void f(){ try { throw A(); } throw 1; } catch (int) { cout << "int"; } catch (A &) { throw; } cout<< " f() "; } int A::c = 0; int main(){ try { try { f(); } cout << " after_f "; } catch (A a) { throw a; } catch (...) { cout << "inner ..."; } } catch (...) {cout << " ... ";} return 0; } </pre>

8. Опишите прототипы двух перегруженных функций `h()` из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
h (-6, "bcd");
h ('-', 6);
h ();
h (3.5);
```

9. Объясните, какой смысл имеет выражение `typeid (ИМЯ_типа)`.

10. Перепишите данный фрагмент, не используя средства из C++11:

```
double t;
decltype (t) x = 50.000;
list<double> l;

auto p = l.rbegin();
while (p != l.rend()) { x -= *p; p++; }
```


NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Что будет напечатано данной программой? Можно ли удалить хотя бы одну операцию разрешения области видимости так, чтобы печать не изменилась?

```
namespace D { int f= 2;
              int g=-2;
}
namespace C { int f= 3;
              int g=-3;
              int h=0;
}
int f=-5;

int main() { int f=1;
            cout<< ::f << D::f << C::f << C::h << endl;
            using namespace D;
            cout << f << g << C::h << endl;
            using namespace C;
            cout << f << C::g << h << endl;
            return 0;
}
```

2. Опишите прототипы двух перегруженных функций `g()` из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
g (-6);
g ('+', 6);
g ();
g ("abc");
```

3. Перепишите данный фрагмент, не используя средства из C++11:

```
long w;
decltype(w) n = 70000;
vector<long> v;
...
for (auto p = v.begin(); p != v.end(); ++ p) {n += * p;}
```

4. Объясните, какой смысл имеет выражение `typeid(*p)`.

5. Что будет напечатано в результате выполнения программы? Компилятор **не** оптимизирующий.

```
class Y {
    static int c;
    int n;
public:
    Y():n(++c) {
        cout<<" Y()="<< n;
    }
    Y(const Y&a):n(++c) {
        cout<<" Y(Y&)="<< n;
    }
    ~Y() {
        cout<<" ~Y()="<< n;
    }
};

void h() { try { throw Y(); } throw 1; }
          catch (int) { cout << "int"; }
          catch (Y a) { throw; }
          cout << " h() ";
}

int Y::c=0;
int main() {
    try { try { h(); } cout << " after_h "; }
        catch (Y & a) { throw a; }
        catch (...) { cout << "inner ..."; }
    }
    catch (...) { cout << " ... "; }
}
```

6. Можно ли продемонстрировать параметрический полиморфизм с помощью оператора-выражения `a*(b*a)`? Обоснуйте ответ. Перечислите другие виды полиморфизма.

7. Данная программа выводит на печать несколько строк. Вычеркните *фрагменты* из функции `main()` так, чтобы напечаталось верное утверждение об ООП и C++.

<pre>class X { public: X() { cout<<"абстрактный "; } virtual ~X() {cout<< endl;} void t(int n) {cout<<"конструктор ";} virtual void t() {cout<<"метод ";} }; class Y : public X { public: Y() {} Y(int x) {cout<<"класс ";} void t() { cout<<"тип ";} void t(int n) { cout<<"интерфейс ";} ~Y() {cout<< "данных";} };</pre>	<pre>int main() { cout<<"класс, не содержащий \ открытых членов-данных, - это "; X * qx = new X; qx -> t(5); delete qx; qx = new Y; qx -> X::t(); delete qx; qx = new Y; { X & rx = * qx; rx.t(); } delete qx; Y * qy = new Y; qy -> t(5); delete qy; Y y(5); }</pre>

8. Может ли одна и та же функция одновременно участвовать в перегрузке (*overloading*), скрытии (*hiding*) и замещении (*overriding*)? Обоснуйте ответ.

Примечание: замещение проявляется при работе механизма виртуальных функций.

9. Добавьте реализацию метода `sum` (из класса `C`), который вычисляет сумму всех числовых полей объекта класса `C` (каждое поле учитывается в сумме один раз). Операцию `::` можно использовать не более одного раза в каждом первичном выражении (то есть `Y::x` возможно, `Z::Y::x` – нет).

Примечание: у операции `->` приоритет выше, чем у операции приведения типа `(тип) x`.

<pre>struct Base { int b; }; struct B1: virtual Base { int b; };</pre>	<pre>struct A1:Base { int b; }; struct B2: virtual Base { int b; };</pre>	<pre>struct A2:Base { int b; }; struct C: A1, A2, B1, B2 { int sum(); int b; };</pre>

10. Приведите фрагмент программы с примером *ошибочного* обращения к видимой, но недоступной функции.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Что будет напечатано данной программой? Можно ли удалить хотя бы одну операцию разрешения области видимости так, чтобы печать не изменилась?

```

namespace L {
    int f(){ return 2; }
    int g = -2;
}
namespace K {
    int f(){ return 3; }
    int g = -3;
    int h = 0;
}

int f() { return -7; }
int main() { int g = -1;
    cout << f() << K::f() << L::f() << K::h << endl;
    using namespace K;
    cout << K::f() << g << h << endl;
    using namespace L;
    cout << L::f() << L::g << h << endl;
    return 0;
}
    
```

2. Опишите прототипы двух перегруженных функций `p` из некоторой области видимости, для которых будут верны следующие обращения к ним:

```

p ();
p (-6, "q×q");
p (-0.5);
p ('*', 9);
    
```

3. Перепишите данный фрагмент, не используя средства из C++11:

```

double r;
decltype (r) x = 70.000;
list<double> l;
auto p = l.rbegin();
while (p != l.rend()) { x -= * p; p ++; }
    
```

4. Объясните, какой смысл имеет выражение `typeid(имя_типа)`.

5. Что будет напечатано в результате выполнения программы? Компилятор *не* оптимизирующий.

```

class B {
    static int c;
    int n;
public:
    B():n(++c) {
        cout<<" B()="<< n;
    }
    B(const B&a):n(++c) {
        cout<<" B(B&)="<< n;
    }
    ~B() {
        cout<<" ~B()="<< n;
    }
};

void s(){ try {throw B(); throw 1;}
    catch (int) {cout << "int";}
    catch (B &) {throw;}
    cout<< " s() ";
}

int B::c=0;
int main(){
    try { try { s(); cout<< " after_s ";}
        catch(B a) {throw a;}
        catch(...) { cout << "inner ..."; }
    }
    catch (...) {cout << " ... ";}
    return 0;
}
    
```

6. Можно ли продемонстрировать параметрический полиморфизм с помощью оператора-выражения $(b*a)*b$? Обоснуйте ответ. Перечислите другие виды полиморфизма.

7. Данная программа выводит на печать несколько строк. Вычеркните *фрагменты* из функции `main()` так, чтобы напечаталось верное утверждение об ООП и C++.

<pre>class Y { public: Y() { cout<<"абстрактный "; } virtual ~Y() {cout<< endl;} void g(int n){cout<<"класс ";} virtual void g() {cout<<"метод ";} }; class Z : public Y { public: Z(){} Z(int x) {cout<<"деструктор ";} void g() { cout<<"конструктор ";} void g(int n) { cout<<"интерфейс ";} ~Z() {cout<< "данных";} };</pre>	<pre>int main() { cout<<" класс, содержащий чистую \ виртуальную функцию, - это "; Y * qy = new Z; qy -> Y::g(); delete qy; qy = new Z; { Y & ry = * qy; ry.g(); } delete qy; Z * qz = new Z; qz -> g(7); delete qz; qy = new Y; qy -> g(7); delete qy; Z z(7); }</pre>

8. Может ли функция одновременно участвовать в скрытии (*hiding*) и замещении (*overriding*) и при этом не участвовать в перегрузке (*overloading*)? Обоснуйте ответ.

Примечание: замещение проявляется при работе механизма виртуальных функций.

9. Добавьте реализацию метода `prod()` (из класса `C`), который вычисляет произведение всех числовых полей объекта класса `C` (каждое поле учитывается в произведении один раз). Операцию `::` можно использовать не более одного раза в каждом первичном выражении (то есть `Y::x` возможно, `Z::Y::x` – нет).

Примечание: у операции `->` приоритет выше, чем у операции приведения типа `"(тип) x"`.

<pre>class Base { public: int b; }; class B1: virtual public Base { public: int b; };</pre>	<pre>class A1: public Base { public: int b; }; class B2 : virtual public Base { public: int b; };</pre>	<pre>class A2: public Base { public: int b; }; class C: public A1, public A2, public B1, public B2 { public: int prod(); int b;};</pre>

10. Приведите пример *ошибочного* обращения к доступному, но не видимому (без операции `::`) методу класса.

Коллоквиум, 2015 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ.

```
class A { void f (){}; }; class B: public A { public: void g(){ f(); } };  
// метод f() в частной части A недоступен для наследника
```

Критерии: пример не удовлетворяет условию или отсутствует: **-10**.

2. Ответ.

-5230

1-20

1-30

Ничего удалить нельзя.

Критерии: за каждую неверно напечатанную строчку: **-3**, за неправильный ответ на второй вопрос: **-4**.

3. Ответ: другие виды полиморфизма – статический, динамический.

Параметрический продемонстрировать можно: нужно привести такое описание шаблонной операции, чтобы разные вхождения операции '+' в выражение $a + (b + a)$; настраивались на разные типы операндов (одна и та же операция будет означать разное в зависимости от контекста – это и есть проявление полиморфизма). Однако, если приведен пример с настройкой на один и тот же тип, то такой ответ тоже засчитывается без снятия баллов. Обоснование может быть схематичным, например, без реализации тел функций.

```
#include <iostream>  
using namespace std;  
template <typename T1, typename T2>  
T1 operator+ (T1 a, T2 b) { cout << a << "+" << b << " | "; return a; }  
class A { char c; public: A(): c ('A') {}  
    friend ostream & operator << (ostream & out, const A & a); };  
ostream & operator << (ostream & out, const A & a) { out << a.c; return out; }  
class B { char c; public: B(): c ('B') {}  
    friend ostream & operator << (ostream & out, const B & b); };  
ostream & operator << (ostream & out, const B & b) { out << b.c; return out; }  
int main() { A a; B b; a+(b+a); // operator+ (a, operator+(b,a));  
    return 0; } // Напечатается: B+A | A+B |
```

Критерии: за каждый не названный (или не существующий) вид полиморфизма: **-2**, за неверный ответ или отсутствующее обоснование про параметрический полиморфизм: **-6**.

4. Ответ: напечатаются строки:

класс, не содержащий открытых членов-данных, – это абстрактный конструктор
абстрактный метод данных
абстрактный тип данных
абстрактный интерфейс данных
абстрактный класс данных

Единственный правильный вариант:

класс, в котором нет открытых членов, – это абстрактный тип данных

Поэтому в теле функции main() нужно оставить только:

```
cout << "класс, не содержащий открытых членов-данных, – это ";  
A * qa; qa = new B; { A & ra = *qa; ra.t(); } delete qa;
```

Критерии: за неверный ответ или вычеркивание за пределами функции main(): **-10**.

5. Ответ: да, в задаче 4 таковой является функция void t() из класса B.

Критерии: **-10** за неправильный или не обоснованный ответ.

6. Решение. Для начала выясняется, сколько всего полей в объекте класса *C*.

1) *a* из *C* (объявлено непосредственно в *C*), 2) *a* из *A1* (объявлено в *A1*), 3) *a* из *A2*, 4) *a* из *B1*, 5) *a* из *B2*, 6) *a* из *Base*, который является частью *A1*, 7) *a* из *Base*, который является частью *A2*, 8) *a* из *Base*, который является общей частью *B1* и *B2* («виртуальное» наследование).

Видимость всех 8 полей "*a*" из метода `sum()` определяется индивидуально. Первые пять из перечисленных в списке полей видны без проблем. Например, "*A1* : *a*" означает поле "*a*", объявленное в классе *A1*. Выражение "*Base*::*a*" в классе *C* неоднозначно, так как *Base* присутствует в классе *C* в трех экземплярах – непонятно, какое из трех "*a*" имеется в виду.

[Компилятор Visual C++ умеет различать их таким способом: *A1::Base::a*, *A2::Base::a*, *B1::Base::a*. Вместо последнего можно было также написать *B2::Base::a* (оба способа именуют одно и то же поле). Компилятор g++ считает имена *A1::Base::a*, *A2::Base::a*, *B1::Base::a* неоднозначными. Это, по-видимому, более правильно, поскольку грамматика языка C++ так устроена, что операция "::" является правоассоциативной. Чтобы не сталкиваться с разным поведением компиляторов, лучше запрещать в условии задачи дважды использовать "::" в одном первичном выражении].

Следует использовать указатель на объект `this`. Если привести его явно к указателю на *A2*, из исходного объекта возникнет указатель на «подобъект» типа *A2*, далее полученный указатель можно привести к указателю на тип *Base* и построить указатель на «подобъект» типа *Base*, причем именно тот *Base*, который «находится» в *A2*. То же выполняется и для оставшихся двух полей типа *Base*. Другой допустимый вариант: привести `this` сначала к указателю на *A1* (соответственно *A2*, *B1*) и использовать доступ к полю *Base*::*a* через указатель, например: `((A1*)this)->Base::a`.

Ответ:

```
int C::sum() { return a + A1::a + A2::a + B1::a + B2::a +
              ((Base*)(A1*)this)->a + // другой вариант: ((A1*)this)->Base::a
              ((Base*)(A2*)this)->a + // другой вариант: ((A2*)this)->Base::a
              // или ((B2*)this)->Base::a
              ((Base*)(B1*)this)->a; } // другой вариант: ((B1*)this)->Base::a
```

Чтобы проверить, можно воспользоваться следующей функцией:

```
int main() { C c, * p = &c; c.A1::a = 1; c.A2::a = 1; c.a = 1; c.B1::a = 1;
            c.B2::a = 1; ((A1*)p)->Base::a = 1;
            ((B1*)p)->Base::a = 1; ((A2*)p)->Base::a = 1; // ((B2*)p)->Base::a = 1;
            cout << c.sum(); return 0; }
```

Критерии: за каждое неверное слагаемое : -3.

7. Конструктор `A()` в операторе `throw A()`; будет вызываться для создания временного объекта, с которого будет «сниматься копия» конструктором копирования для создания объекта-исключения, после чего исключение будет передано перехватчику, а временный объект, созданный `A()`, исчезнет. Компиляторы обычно оптимизируют эту ситуацию, не делая лишнего копирования. Однако в условии явно сказано, что оптимизации нет. Поэтому ответ не совсем похож на то, что выдаст скомпилированная программа. В случае `throw a`; временный объект не создается, копия при создании объекта-исключения «снимается» с уже существующего объекта *a*.

Ответ: `A()=1 A(A&)=2 ~A()=1 A(A&)=3 ~A()=3 A(A&)=4 ~A()=2 ... ~A()=4.`

Критерии: за каждую ошибку -3.

8. Ответ: `f(int i=5,int j=9); f(const char * s);` //Возможны другие решения

Критерии: -3 за каждый неподходящий вызов из четырех, но не меньше 0. Если описано больше двух функций: =0.

9. Ответ : динамическое определение типа объекта, на который указывает указатель *p*.

Критерии: за отсутствие или неправильный ответ =0.

10. Ответ:

```
long m, n = 50000;
vector<long> v;
for(vector<long>::iterator p=v.begin(); p!=v.end(); ++p) {n += *p;}
```

Критерии: неправильно заменен `decltype`: -5, неправильно заменен `auto`: -5.

Коллоквиум, 2015 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: `class A{public: void f() {} }; class B: public A{public: void g(int f) {f();};`
// параметр f скрыл метод f() из A

Или `class A{public: void f() {} }; class B: public A{int f; public: void g() {f();};`
// поле f скрыло метод f() из A.

Критерии: пример не удовлетворяет условию или отсутствует: **-10.**

2. Ответ: **-7320** **3-10** **2-20**

Ничего удалить нельзя.

Критерии: за каждую неверно напечатанную строчку: **-3**, за неправильный ответ на второй вопрос: **-4.**

3. Ответ: другие виды полиморфизма – статический, динамический.

Параметрический продемонстрировать можно: нужно привести такое описание шаблонной операции, чтобы разные вхождения операции '+' в выражение (b+a)+b; настраивались на разные типы операндов (одна и та же операция будет означать разное в зависимости от контекста – это и есть проявление полиморфизма). Однако, если приведен пример с настройкой на один и тот же тип, то такой ответ тоже засчитывается без снятия баллов. Обоснование может быть схематичным, например, без реализации тел функций.

```
#include <iostream>
using namespace std;
template <typename T1, typename T2>
T2 operator+ (T1 a, T2 b) { cout << a << "+" << b << " | "; return b; }
class A { char c; public: A(): c ('A') {}
    friend ostream & operator << (ostream & out, const A & a); };
ostream & operator << (ostream & out, const A & a) { out << a.c; return out; }
class B { char c; public: B(): c ('B') {}
    friend ostream & operator << (ostream & out, const B & b); };
ostream & operator << (ostream & out, const B & b) { out << b.c; return out; }
int main() { A a; B b; (b+a)+b; // operator+ ( operator+(b,a),b);
    return 0; } // Напечатается: B+A | A+B |
```

Критерии: за каждый не названный (или не существующий) вид полиморфизма: **-2**, за неверный ответ или отсутствующее обоснование про параметрический полиморфизм: **-6.**

4. Ответ: напечатаются строки:

класс, содержащий чистую виртуальную функцию, – это абстрактный метод данных
абстрактный конструктор данных
абстрактный интерфейс данных
абстрактный класс
абстрактный деструктор данных

Единственный правильный вариант:

класс, содержащий чистую виртуальную функцию, – это абстрактный класс

Поэтому в теле функции main() нужно оставить только:

```
cout << "класс, содержащий чистую виртуальную функцию, – это " ;
A * qa; qa = new A; qa -> g(7); delete qa;
```

Критерии: за неверный ответ или вычеркивание за пределами функции main(): **-10.**

5. Ответ: да, если в задаче 4 в классе B вычеркнуть g(int); то оставшаяся функция void g() из класса B будет удовлетворять условию задачи.

Критерии: **-10** за неправильный или не обоснованный ответ.

6. Ответ:

```
int C::prod() { return a * A1::a * A2::a * B1::a * B2::a *
                ((Base*)(A1*)this)->a * // другой вариант: ((A1*)this)->Base::a
                ((Base*)(A2*)this)->a *
                ((Base*)(B1*)this)->a; }
```

Критерии: за каждое неверное слагаемое : **-3**.

7. Ответ: A()=1 A(A&)=2 ~A()=1 A(A&)=3 A(A&)=4 ~A()=3 ~A()=2 ... ~A()=4.

Критерии: за каждую ошибку **-3**.

8. Ответ: h(double d=5,int j=9);h(int i,const char*s); // Есть другие решения

Критерии: **-3** за каждый неподходящий вызов из четырех, но не меньше **0**. Если описано больше двух функций: **=0**.

9. Ответ : динамическое определение типа объекта, на который указывает указатель p.

Критерии: за отсутствие или неправильный ответ **=0**.

10. Ответ:

```
double t, x = 50.000;
list <double> l;
list <double>::reverse_iterator p = l.rbegin();
while (p != l.rend()) {x-= * p; p ++;}
```

Критерии: неправильно заменен decltype: **-5**, неправильно заменен auto: **-5**.

Коллоквиум, 2015 год. Ответы и критерии оценок.
Вариант 3.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. -5230 1-20 1-30

Ничего удалить нельзя.

Критерии: за каждую неверно напечатанную строчку: -3, за неправильный ответ на второй вопрос: -4.

2. Ответ: `g(int i=5,int j=9); g(const char * s);` //Возможны другие решения

Критерии: -3 за каждый неподходящий вызов из четырех, но не меньше 0. Если описано больше двух функций: =0.

3. Ответ:

```
long w, n = 70000;
vector<long> v;
for(vector<long>::iterator p=v.begin(); p!=v.end(); ++p) {n+= *p;}
```

Критерии: неправильно заменен decltype: -5, неправильно заменен auto: -5.

4. Ответ : динамическое определение типа объекта, на который указывает указатель p.

Критерии: за отсутствие или неправильный ответ =0.

5. Ответ: $Y()=1$ $Y(Y\&)=2$ $\sim Y()=1$ $Y(Y\&)=3$ $\sim Y()=3$ $Y(Y\&)=4$ $\sim Y()=2\dots$ $\sim Y()=4$.

Критерии: за каждую ошибку -3.

6. Ответ: другие виды полиморфизма – статический, динамический.

Параметрический продемонстрировать можно: нужно привести такое описание шаблонной операции, чтобы разные вхождения '*' в выражение $a*(b*a)$; настраивались на разные типы операндов (одна и та же операция будет означать разное в зависимости от контекста – это и есть проявление полиморфизма). Однако, если приведен пример с настройкой на один и тот же тип, то такой ответ тоже засчитывается без снятия баллов. Обоснование может быть схематичным, например, без реализации тел функций.

```
#include <iostream>
using namespace std;
template <typename T1, typename T2>
T1 operator* (T1 a, T2 b) { cout << a << "*" << b << " | "; return a; }
class A { char c; public: A(): c ('A') {}
    friend ostream & operator<<(ostream & out, const A & a); };
ostream & operator << (ostream & out, const A & a) { out << a.c; return out; }
class B { char c; public: B(): c ('B') {}
    friend ostream & operator<<(ostream & out, const B & b); };
ostream & operator << (ostream & out, const B & b) { out << b.c; return out; }
int main() { A a; B b; a*(b*a); // operator* (a, operator*(b,a));
    return 0; } // Напечатается : B*A | A*B |
```

Критерии: за каждый не названный (или не существующий) вид полиморфизма: -2, за неверный ответ или отсутствующее обоснование про параметрический полиморфизм: -6.

7. Ответ: напечатаются строки

класс, не содержащий открытых членов-данных, – это абстрактный конструктор
 абстрактный метод данных
 абстрактный тип данных
 абстрактный интерфейс данных
 абстрактный класс данных

Единственный правильный вариант:

класс, в котором нет открытых членов, – это абстрактный тип данных

Поэтому в теле `main()` нужно оставить только:

```
cout<< "класс, не содержащий открытых членов-данных, – это ";
    X * qx;      qx = new Y;      { X & rx = *qx;      rx.t(); }      delete qx;
```

Критерии: за неверный ответ или вычеркивание за пределами функции `main()`: -10.

8. Ответ: да, в задаче 7 таковой является функция `void t()` из класса `Y`.

Критерии: -10 за неправильный или не обоснованный ответ.

9. Ответ:

```
int C::sum() { return b + A1::b+ A2::b+ B1::b + B2::b +  
                ((Base*)(A1*))this->b + // другой вариант: ((A1*)this)->Base::b  
                ((Base*)(A2*))this->b +  
                ((Base*)(B1*))this->b; }
```

Критерии: за каждое неверное слагаемое : -3.

10. Ответ.

```
class A {void f (){}; }; class B: public A { public: void g(){ f(); } };  
// метод f() в приватной части A недоступен для наследника
```

Критерии: пример не удовлетворяет условию или отсутствует: -10.

Коллоквиум, 2015 год. Ответы и критерии оценок.
Вариант 4.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. -7320 3-10 2-20

Ничего удалить нельзя.

Критерии: за каждую неверно напечатанную строчку: -3, за неправильный ответ на второй вопрос: -4.

2. Ответ: `p(double d=5,int j=9);p(int i,const char*s);` // Есть другие решения

Критерии: -3 за каждый неподходящий вызов из четырех, но не меньше 0. Если описано больше двух функций: =0.

3. Ответ:

```
double r, x = 70.000;
list <double> l;
list <double>::reverse_iterator p = l.rbegin();
while (p != l.rend()) {x-= *p; p++;}
```

Критерии: неправильно заменен `decltype`: -5, неправильно заменен `auto`: -5.

4. Ответ: в этом случае операция `typeid` возвращает ссылку на объект типа `type_info`, представляющий тип именно с этим именем.

Критерии: за отсутствие или неправильный ответ =0.

5. Ответ. `B()=1 B(B&)=2 ~B()=1 B(B&)=3 B(B&)=4 ~B()=3 ~B()=2... ~B()=4.`

Критерии: за каждую ошибку -3.

6. Ответ: другие виды полиморфизма – статический, динамический.

Параметрический продемонстрировать можно: нужно привести такое описание шаблонной операции, чтобы разные вхождения ' * ' в выражение `(b*a)*b`; настраивались на разные типы операндов (одна и та же операция означает разное в зависимости от контекста – это и есть проявление полиморфизма). Однако, если приведен пример с настройкой на один и тот же тип, то такой ответ тоже засчитывается без снятия баллов. Обоснование может быть схематичным, например, без реализации тел функций.

```
#include <iostream>
using namespace std;
template <typename T1, typename T2>
T2 operator* (T1 a, T2 b) {cout << a << "*" << b << " | "; return b; }
class A { char c; public: A(): c ('A') {}
    friend ostream & operator << (ostream & out, const A & a); };
ostream & operator << (ostream & out, const A & a){ out << a.c; return out; }
class B { char c; public: B(): c ('B') {}
    friend ostream & operator << (ostream & out, const B & b); };
ostream & operator << (ostream & out, const B & b){ out << b.c; return out; }
int main() { A a; B b; (b*a)*b; // operator* ( operator*(b,a),b);
    return 0; } // Напечатается : B*A | A*B |
```

Критерии: за каждый не названный (или не существующий) вид полиморфизма: -2, за неверный ответ или отсутствующее обоснование про параметрический полиморфизм: -6.

7. Ответ: напечатаются строки:

класс, содержащий чистую виртуальную функцию, – это абстрактный метод данных
абстрактный конструктор данных
абстрактный интерфейс данных
абстрактный класс
абстрактный деструктор данных

Единственный правильный вариант:

класс, содержащий чистую виртуальную функцию, – это абстрактный класс

Поэтому в теле main нужно оставить только:

```
cout << "класс, содержащий чистую виртуальную функцию, – это " ;  
Y * qy; qy = new Y; qy -> g(7); delete qy;
```

Критерии: за неверный ответ или вычеркивание за пределами функции main(): **-10**.

8. Ответ: *да*, если в задаче 7 в классе Z вычеркнуть g(int); то оставшаяся функция void g() из класса Z будет удовлетворять условию задачи.

Критерии: **-10** за неправильный или не обоснованный ответ.

9. Ответ:

```
int C::prod() { return b * A1::b * A2::b * B1::b * B2::b *  
                ((Base*) (A1*)this)->b * // другой вариант: ((A1*)this)->Base:: b  
                ((Base*) (A2*)this)->b *  
                ((Base*) (B1*)this)->b; }
```

Критерии: за каждое неверное слагаемое : **-3**.

10. Ответ: `class A{public: void f(){};}; class B:public A{public: void g(int f){f();};`
// параметр f скрыл метод f() из A

Или `class A{public: void f(){};}; class B:public A{int f; public: void g(){f();}`
// поле f скрыло метод f() из A

Критерии: пример не удовлетворяет условию или отсутствует: **-10**.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. При компиляции функции `main()` следующей программы будет зафиксирована ошибка: "неразрешимый вызов функции `f(a, 0)`". Объяснить причину такой ошибки, пользуясь алгоритмом выбора наиболее подходящей (best-matching) функции.

```

struct A {
    operator int () { return 1; }
};
void f (double d, char c) { cout<<"f(double,char)\n"; }
void f (double d, int j) { cout<<"f(double,int)\n"; }
void f (A a, const char*p) { cout<<"f( A,const char*)\n"; }
void f (int i, const char*p) { cout<<"f(int,const char*)\n"; }

int main () {
    A a ;
    f (a, 0);
    return 0;
}

```

2. Привести два примера классов, для которых нельзя создать объект. Причины, по которым нельзя создать объект, должны быть для этих классов **разными**.

3. При компиляции приведенной программы на C++ фиксируются ошибки. Объяснить, в чем они заключаются. Исправить все ошибочные конструкции **только** путем **вставки** или **замены** служебных слов C++. Что будет напечатано в результате работы получившейся правильной программы?

```

struct B {
    virtual char f () {
        cout << "f_from_B\n"; return 0; }
    virtual void g () {
        cout << "g_from_B\n"; }
};
struct T : B { int f () {
    cout << "f_from_T\n"; return 0; }
    void g () { cout << "g_from_T\n "; }
};

int main () { int n;
    const T ct;
    T t;
    B * pb = &t;
    n = pb -> f ();
    pb -> g ();
    ct.g();
    return 0;
}

```

4. Что будет напечатано в результате выполнения программы? Компилятор оптимизирующий.

```

class A { public:
    static A sa;
    A(){ cout << "A()\n"; }
    A(const A & a){
        cout << "A(const A&)\n"; }
    ~A(){ cout << "~A()\n"; }
};

A A::sa;
int main() {
    try { A a1 = A::sa;
        try { throw A(); }
        catch (int) {cout<<"111<<endl; }
    }
    catch (...) {cout<<" ... \n"; }
    cout<<"222<<endl;
    return 0;
}

```

5. Написать на C++ **алгоритм в стиле STL**, который применим к контейнерам, имеющим итератор произвольного доступа (`Random_accessIterator`). Алгоритм должен распечатывать элементы с 10-го по 5-й из заданного диапазона контейнера (элементы внутри диапазона нумеруются с единицы). Если количество заданных элементов меньше 10, напечатать последний элемент диапазона или 0 в случае пустого диапазона.

6. а) Построить приведенную грамматику, эквивалентную заданной грамматике G:

$$\begin{aligned}S &\rightarrow aABb \mid C \\ B &\rightarrow bB \mid \varepsilon \\ A &\rightarrow aA \mid D \\ C &\rightarrow cCc \mid B \\ D &\rightarrow dD\end{aligned}$$

- б) Каков тип получившейся грамматики?
в) Какой язык порождает грамматика (привести формулу языка)?
г) Каков тип этого языка?

Для типов грамматики и языка указать максимально возможный номер по Хомскому.

7. Дана автоматная праволинейная грамматика G: $S \rightarrow 1B$
 $B \rightarrow 1B \mid 1A$
 $A \rightarrow 1C$
 $C \rightarrow 0A \mid \perp$

- а) Детерминирован ли разбор по этой грамматике? **Обосновать ответ.**
б) Построить по грамматике конечный автомат и преобразовать его к детерминированному виду по алгоритму НКА $\rightarrow$ ДКА.
в) Построить по получившемуся ДКА левوليнейную автоматную грамматику.
г) Эквивалентны ли исходная и получившаяся автоматные грамматики? Обосновать ответ.

8. Дана КС-грамматика G: $S \rightarrow aABc \mid C$
 $A \rightarrow aB \mid bA$
 $B \rightarrow bBC \mid \varepsilon$
 $C \rightarrow cC \mid \varepsilon$

- а) Преобразовать грамматику G к неукорачивающей КС-грамматике с помощью алгоритма устранения пустых правых частей в КС грамматике (КС $\rightarrow$ НКС).
б) Применим ли метод рекурсивного спуска к исходной грамматике? **Ответ обосновать.**

9. Построить грамматику выражений, содержащих

- левоассоциативные целочисленные бинарные операции ' * ' и ' / ';
- односимвольные целые числа 0, 1, 2, 3, 4, 5;
- скобок нет.

Добавить в грамматику действия (только вида cout <<"символ") по переводу заданных выражений в постфиксную запись во время анализа методом рекурсивного спуска.

Приоритет операций ' * ' и ' / ' должен быть **задан построенной грамматикой** таким образом, чтобы постфиксная запись выражения:

$$5 * 3 / 2 / 1 * 2 * 1$$

совпадала с постфиксной записью выражения со скобками:

$$((5 * ((3 / 2) / 1) * 2) * 1).$$

10. а) Перечислить этапы **создания** программного продукта. б) Что называют программным продуктом?

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. При компиляции функции `main()` следующей программы будет зафиксирована ошибка: "неразрешимый вызов функции `f('a', 0)`". Объяснить причину такой ошибки, пользуясь алгоритмом выбора наиболее подходящей (best-matching) функции.

```

struct A {
    operator int () { return 1; }
};
void f (double d, char c) { cout<<"f (double,char)\n";}
void f (double d, int j) { cout<<"f (double,int)\n"; }
void f (A a, const char*p) { cout<<"f( A,const char *)\n";}
void f (int i,const char*p){ cout<<"f(int,const char *)\n";}

int main () {
    A a ;
    f ('a', 0);
    return 0;
}

```

2. Привести пример программы, в которой в результате вызова из функции `main()` метода `f()` некоторого класса через указатель `p` (`p -> f()`) проработает метод `f()`, описанный в закрытой (private) области другого класса.

3. При компиляции приведенной программы на C++ фиксируются ошибки. Объяснить, в чем они заключаются. Исправить все ошибочные конструкции **только** путем **вставки** или **замены** служебных слов C++. Что будет напечатано в результате работы получившейся правильной программы?

```

struct A {
    virtual void h (int & a) { a++;
        cout << "h_from_A\n"; }
    virtual int t () {
        cout << "t_from_A\n"; return 0; }
};
struct S : A {
    int h (long & a) { a++;
        cout << "h_from_S\n"; return 0;}
    int t () {
        cout << "t_from_S\n"; return 0;}
};

int main() { int n = 5;
    const A a1;
    S s1;
    A * p = &s1;
    a1.t ();
    p -> h (n);
    p -> t ();
    cout << "n = " << n << endl;
    return 0;
}

```

4. Что будет напечатано в результате выполнения программы? Компилятор оптимизирующий.

```

struct B { B(){ cout << "B()\n"; }
    ~B(){ cout << "~B()\n"; }
};
class A { static B sa;
public:
    A(){ cout << "A()\n"; }
    A(const A & a) {
        cout << "A(const A&)\n"; }
    ~A(){ cout << "~A()\n"; }
};

B A::sa;
int main() { try { A a1;
    try { throw B(); }
    catch (A&){cout<<333<<endl;}
}
    catch (...) {cout<<" ... \n";}
    cout << " End\n ";
    return 0;
}

```

5. Написать алгоритм в стиле STL, который применим к контейнерам, имеющим двунаправленный итератор (`BidirectionalIterator`). Алгоритм должен распечатывать 3-й с конца элемент заданного диапазона контейнера. Если количество заданных элементов меньше 3, напечатать первый заданный элемент или строку `"Empty\n"` в случае пустого диапазона.

6. а) Построить приведенную грамматику, эквивалентную заданной грамматике G:

$$\begin{aligned}S &\rightarrow aAb \mid CB \\ B &\rightarrow bB \\ A &\rightarrow aA \mid D \\ C &\rightarrow cCc \mid B \mid c \\ D &\rightarrow d\end{aligned}$$

- б) Каков тип получившейся грамматики?
в) Какой язык порождает грамматика (формула)?
г) Каков тип этого языка?

Для типов грамматики и языка указать максимально возможный номер по Хомскому.

7. Дана автоматная левостроительная грамматика G: $S \rightarrow C\perp$
 $A \rightarrow Aa \mid a$
 $B \rightarrow Aa \mid b$
 $C \rightarrow Ba$

- а) Детерминирован ли разбор по этой грамматике? **Обосновать ответ.**
б) Построить по грамматике конечный автомат и преобразовать его к детерминированному виду по алгоритму НКА $\rightarrow$ ДКА.
в) Построить по получившемуся ДКА праволинейную автоматную грамматику.
г) Эквивалентны ли исходная и получившаяся автоматные грамматики? **Обосновать ответ.**

8. Дана КС-грамматика G: $S \rightarrow A \mid B$
 $A \rightarrow aA \mid \varepsilon$
 $B \rightarrow bB \mid cC \mid \varepsilon$
 $C \rightarrow aABC \mid c$

- а) Преобразовать грамматику G к неукорачивающей КС-грамматике с помощью алгоритма устранения пустых правых частей в КС грамматике (КС $\rightarrow$ НКС).
б) Применим ли метод рекурсивного спуска к исходной грамматике? **Ответ обосновать.**

9. Построить грамматику выражений, содержащих

- левоассоциативные целочисленные бинарные операции '+' и '-' ;
- односимвольные целые числа 0, 2, 4, 6 ;
- скобок нет.

Добавить в грамматику действия (только вида cout <<"символ") по переводу заданных выражений в постфиксную запись во время анализа методом рекурсивного спуска.

Приоритет операций '+' и '-' должен быть **задан построенной грамматикой** таким образом, чтобы постфиксная запись выражения:

$$2 + 4 - 6 - 0 + 4 + 2$$

совпадала с постфиксной записью выражения со скобками:

$$((2+((4-6)-0)+4)+2).$$

10. а) Назвать основные задачи редактора связей. б) Привести пример ситуации, в которой редактор связей выдаст сообщение об ошибке.

7. Ответ. (а) *Нет*, так как есть альтернативы вывода из В, начинающиеся одинаковыми символами.

$$(б) \delta(S, 1) = B$$

$$\delta(B, 1) = \underline{AB}$$

$$\delta(\underline{AB}, 1) = \underline{ABC}$$

$$\delta(\underline{ABC}, 1) = \underline{ABC}$$

$$\delta(\underline{ABC}, 0) = A$$

$$\delta(\underline{ABC}, \perp) = F$$

$$\delta(A, 1) = C$$

$$\delta(C, 0) = A$$

$$\delta(C, \perp) = F$$

$$(в) F = S, S = H$$

$$S \rightarrow C \perp \mid \underline{ABC} \perp$$

$$C \rightarrow A \ 1$$

$$A \rightarrow C \ 0 \mid \underline{ABC} \ 0$$

$$\underline{ABC} \rightarrow \underline{ABC} \ 1 \mid \underline{AB} \ 1$$

$$\underline{AB} \rightarrow B \ 1$$

$$B \rightarrow 1$$

(г) *Да*, по построению (порождают один и тот же язык), $L = \{I^n(01)^m \perp \mid n \geq 3, m \geq 0\}$.

Критерии: неверный \ без обоснования ответ на пункт (а) или (б) -4, за каждый, неверный\без обоснования ответ на пункт (в) или (г) -3 за каждый.

8. Ответ.

$$(а) S \mid \varepsilon$$

$$S \rightarrow aABc \mid aAc \mid C$$

$$A \rightarrow aB \mid bA \mid a$$

$$B \rightarrow bBC \mid bB \mid bC \mid b$$

$$C \rightarrow cC \mid c$$

(б) *Нет*, так как, например, $first(C) \cap follow(C) = \{c\} \neq \emptyset$.

Критерии: За любую ошибку в пункте (а) -5. За неверный ответ\обоснование в пункте (б) -5.

9. Ответ.

$$E \rightarrow T \{ * T < cout << '*' > \}$$

$$T \rightarrow F \{ / F < cout << '/' > \}$$

$$F \rightarrow 1 < cout << '1' > \mid 2 < cout << '2' > \mid 3 < cout << '3' > \mid 4 < cout << '4' > \mid 5 < cout << '5' >$$

Критерии: Неверная грамматика -5. Неверно вставленные действия -5.

10. Ответ.

1) Постановка задачи. Анализ (определение) требований.

2) Проектирование.

3) Написание текста программ (программирование, “кодирование”)

4) Компоновка или интеграция программного комплекса.

5) Верификация, тестирование и отладка

6) Документирование

Программный продукт – программа, оформленная, документированная и специфицированная таким образом, что ее можно использовать независимо, отчужденно от автора программы.

Критерии: За каждый пропущенный этап -2. Неверное понимание программного продукта -3.

Экзамен, 2015 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. На этапе 2

- по первому параметру наилучшим образом подходит последняя функция (на шаге б) – целочисленное расширение,
 - по второму параметру – вторая функция (на шаге а) – точное отождествление.
 Пересечение пусто, следовательно, вызов неразрешим.

Критерии: Неверное объяснение =0. Достаточно указать выбранную функцию для каждого параметра.

2. Ответ. В базовом классе надо описать виртуальную функцию в открытой (public) области, а в производном классе – функцию с таким же прототипом в закрытой (private) области. В функции main() надо описать объект производного класса и указатель на базовый класс, инициализированный адресом объекта производного класса. Далее через указатель можно вызвать соответствующий метод.

Критерии: Любая ошибка =0.

3. Ошибка: неконстантный метод t() вызывается для константного объекта a1.

Исправление: сделать метод t() класса A константным (добавить const).

Будет напечатано:

t_from_A h_from_A t_from_A n = 6

Критерии: За неверно найденную или неверно исправленную ошибку -5. За каждую неверную печать: -3.

4. Ответ. B() A() B() ~A() ... ~B() End ~B()

Критерии: За одну ошибку -7. За две ошибки -3, за большее количество ошибок =0.

5. Ответ.

```
template <class BidirectionalIterator>
void p3 (BidirectionalIterator first, BidirectionalIterator last) {
    BidirectionalIterator tt = last;
    i = 0;
    while (tt != first) { i++; tt--; }
    if (i == 0) cout << " Empty\n";
    else if (i < 3) cout << *first << endl;
    else { last--; last--; last--; cout << *last << endl; }
}
```

Критерии: (а) Отсутствие указания типа допустимого итератора в заголовке шаблона -5, (б) ошибки в описании шаблонной функции -5 каждая, (в) мелкие ошибки -2 каждая.

6. Ответ. (а) $G_{\text{привед.}}: S \rightarrow aAb$ (б) грамматика типа 2 (КС)
 $A \rightarrow aA \mid D$ (в) язык $L = \{ a^n d b \mid n \geq 1, m \geq 0 \}$
 $D \rightarrow d$ (г) тип языка 3 (Рег.)

Критерии: За неправильную приведенную грамматику -4. За неверный язык, неверный тип грамматики, неверный тип языка -3 за каждый.

Замечание: Если для неверной грамматики и/или неверного языка указан верный тип, считать ответ по пунктам (б) и/или (г) верным.

7. **Ответ. (а) Нет**, так как есть одинаковые правые части в правилах вывода грамматики

(б) $\delta(H, a) = A$	(в) $H = S, S = F$	$S \rightarrow aA \mid bB$
$\delta(H, b) = B$		$B \rightarrow aC$
$\delta(A, a) = \underline{AB}$		$A \rightarrow a \underline{AB}$
$\delta(B, a) = C$		$AB \rightarrow a \underline{ABC}$
$\delta(\underline{AB}, a) = \underline{ABC}$		$\underline{ABC} \rightarrow a \underline{ABC} \mid \perp$
$\delta(\underline{ABC}, a) = \underline{ABC}$		$C \rightarrow \perp$
$\delta(\underline{ABC}, \perp) = S$		
$\delta(C, \perp) = S$		

(г) **Да**, по построению (порождают один и тот же язык), $L = \{ba\} \cup \{a^n \perp \mid n \geq 3\}$.

Критерии: неверный \ без обоснования ответ на пункт (а) или (б) -4, за каждый, неверный\без обоснования ответ на пункт (в) или (г) -3 за каждый.

8. **Ответ.**

(а) $SI \rightarrow S \mid \varepsilon$
 $S \rightarrow A \mid B$
 $A \rightarrow aA \mid a$
 $B \rightarrow bB \mid b \mid cC$
 $C \rightarrow aABc \mid aAC \mid aBC \mid aC \mid c$

(б) **Нет**, так как из альтернатив нетерминала S выводится ε (или $first(A) \cap follow(A) = \{a\} \neq \emptyset$).

Критерии: За любую ошибку в пункте (а) -5. За неверный ответ\обоснование в пункте (б) -5.

9. **Ответ.**

$$E \rightarrow T \{ + \mid T < \text{cout} << '+' > \}$$

$$T \rightarrow F \{ - \mid F < \text{cout} << '-' > \}$$

$$F \rightarrow 0 < \text{cout} << '0' > \mid 2 < \text{cout} << '2' > \mid 4 < \text{cout} << '4' > \mid 6 < \text{cout} << '6' >$$

Критерии: Неверная грамматика -5. Неверно вставленные действия -5.

10. **Ответ.**

- 1) Связывает между собой по внешним данным объектные модули, порождаемые компилятором и составляющие единую программу.
- 2) Связывает файлы статически подключаемых библиотек с целью получения единого исполняемого модуля.
- 3) Готовит таблицу трансляции относительных адресов для загрузчика.
- 4) Готовит таблицу точек вызова функций динамически подключаемых библиотек.

Пример: отсутствие описания объявленного имени функции при его использовании.

Критерии: За отсутствующий или неверный пункт (1)-(3) -3, за отсутствующий или неверный пункт (4) -1, нет ответа\неверный пример -3.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Вычеркните только те строки функции `main()`, которые приводят к синтаксической ошибке (если таковые имеются). Обязательно **объясните** причины ошибок. Что будет напечатано в результате работы получившейся программы.

<pre> struct A { int n; A(){ n = 9; } A operator+(const A & a){ A t; t.n = n + a.n; return t; } }; struct B:A { B operator+(const B & a){ B t; t.n = n + a.n + 10; return t; } }; </pre>	<pre> int main () { A a; B b; a = a + b; cout << a.n << '\n'; b = b + a; cout << b.n << '\n'; return 0; } </pre>

2. Исправьте **только описание класса A** так, чтобы в приведенной ниже программе не было ошибок, а на экран напечаталось `f(int,int)`.

```

struct A { int n;
  A (int m) { n = m; }
  operator int () { return 1; }
};
void f (int i, int j ) { cout << "f(int,int)\n"; }
void f (A b, A a) { cout << "f(A, A)\n"; }
int main () { A a (1);
  f(a, 1);
  return 0;
}

```

3. Опишите необходимые классы так, чтобы в заданной функции `main()` не было ошибок, а на экран напечаталось **310**.

```

int main () { T t (3);
  P <T> p1(&t), p2(0);
  cout << p1 -> n << p2 -> n << endl;
  return 0;
}

```

4. Добавьте в описание класса A из задачи 1 метод

```
virtual A& operator-() { n = -(n+1); cout << n << "A::operator-\n"; return *this; }
```

добавьте в описание (производного от класса A) класса B из задачи 1 метод

```
B& operator-() { cout << n << " B::operator-\n"; return *this; }
```

Что напечатает программа с нижеприведенной функцией `main()` и получившимися классами A и B?

```

int main () {
  B b;
  A a, & ra = b;
  a = -a;
  ra = -ra;
  ra = a;
  ra = -ra; return 0;
}

```

5. Что такое абстрактный класс в C++? Обязательно приведите **пример** описания и использования абстрактного класса.

6. Укажите все прототипы конструкторов и деструкторов в порядке их выполнения в следующей программе:

```
class A {};  
class B: public A{};  
struct D { A a; };  
  
int main () { D d;  
    { d.a = B();  
      B b; }  
    A a1, a2 = a1;  
    return 0;  
}
```

7. Добавьте в следующую программу необходимые описания так, чтобы в ней не было ошибок.

```
int main () { const A a;  
    a.x = 3;  
    cout << a.y << a.x << a.f(1) << endl;  
    return 0; }
```

8. Модифицируйте функцию *main()*, ничего в ней не удаляя, не используя комментарии, *goto* и прочие переходы так, чтобы программа завершилась нормально (не аварийно).

```
struct B { virtual void f() { cout << "B::f()\n"; } };  
struct D: B { void f() { cout << "D::f()\n"; } };  
int main () {  
    D d, & rd = d;  
    B b, & rb = b, & rbd = d;  
    rd = dynamic_cast<D&> (rbd); rd.f();  
    rd = dynamic_cast<D&> (rb); rd.f();  
    return 0;  
}
```

9. C++11. Вычеркните неверные конструкции в следующей программе на C++11. Обязательно объясните причины ошибок в вычеркнутых конструкциях.

```
struct A { int i = 10; };  
void f (A && x) {}  
int main () { A a;  
    int && n, m;  
    auto b = A();  
    decltype (a) c;  
    f (b);  
    f ( A() );  
    m = nullptr;  
    return 0;  
}
```

10. STL. Напишите шаблонную функцию, подсчитывающую сумму пяти последних элементов заданного контейнера (списка или вектора, константного или неконстантного). Тип элементов контейнера является числовым типом. Если в заданном контейнере меньше 5 элементов, функция должна вернуть 0 соответствующего типа.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Вычеркните только те строки функции `main()`, которые приводят к синтаксической ошибке (если таковые имеются). Обязательно **объясните** причины ошибок. Что будет напечатано в результате работы получившейся программы.

<pre> struct T { int i; T(){ i = 3; } T(int t){ i = t; } T operator*(const T & a){ T t; t.i = i * a.i; return t; } }; struct P:T { P operator*(const P & a){ P t; t.i = i * a.i * 2; return t; } }; </pre>	<pre> int main () { T a(5); P b; a = a * b; cout << a.i << '\n'; b = b * a; cout << b.i << '\n'; return 0; } </pre>

2. Исправьте **только описание класса А** так, чтобы в приведенной ниже программе не было ошибок, а на экран напечаталось `f(A, A)`.

```

struct A { int n;
    A (int m) { n = m; }
    operator int () { return 1; }
};
void f (int i, int j ) { cout << "f(int,int)\n"; }
void f (A b, A a ) { cout << "f(A,A)\n"; }
int main () { A a (1);
    f(a, 1);
    return 0;
}
        
```

3. Опишите необходимые классы так, чтобы в заданной функции `main()` не было ошибок, а на экран печаталось **520**.

```

int main () { C c (5);
    B <C> b1(&c), b2(0);
    cout << (*b1).n << (*b2).n << endl;
    return 0;
}
        
```

4. Добавьте в описание класса **Т** из задачи 1 метод

```
virtual T& operator--(){ i = i-1; cout << i << "T::operator--()\n"; return *this;}
```

добавьте в описание (производного от класса **Т**) класса **Р** из задачи 1 метод

```
P& operator-- () { cout << i << " P::operator--()\n"; return *this; }
```

Что напечатает программа с нижеприведенной функцией `main()` и получившимися классами **Т** и **Р**?

```

int main () { P b;
    T a(5), &rt = b;
    --a;
    --rt;
    rt = a;
    --rt;    return 0;
}
        
```

5. Что такое функция-друг в языке Си++? Приведите **пример** описания и использования функции-друга.

6. Укажите все прототипы конструкторов и деструкторов в порядке их выполнения в следующей программе:

```
class A {};  
class B: public A{};  
struct T { B * pb; };  
  
int main () { A a;  
    { a = B();  
        T t; }  
    B b1, b2 = b1;  
    return 0;  
}
```

7. Добавьте в следующую программу необходимые описания так, чтобы в ней не было ошибок.

```
int main () { B::y = 3;  
    const B b;  
    cout << b.x - B::h(2) << b.h(0) << endl;  
    return 0; }
```

8. Модифицируйте функцию *main()*, ничего в ней не удаляя, не используя комментарии, *goto* и прочие переходы так, чтобы программа завершалась нормально (не аварийно).

```
struct B { virtual void g() { cout << "B::g()\n"; } };  
struct D: B {};  
int main () {  
    D d, * pd1, *pd2;  
    B b, * pb = & b, * pbd = & d;  
    pd1 = dynamic_cast <D*> (pbd);  
    pd2 = dynamic_cast <D*> (pb);  
    if (typeid (*pd1) == typeid (*pd2) ) pb -> g ();  
    return 0;  
}
```

9. C++11. Вычеркните неверные конструкции в следующей программе на C++11.

Обязательно **объясните** причины ошибок в вычеркнутых конструкциях.

```
struct S { int x = 1; };  
void g (S && s) { cout << s.x << endl; }  
int main () { int && i;  
    auto k = nullptr;  
    decltype (S()) a;  
    char * s = k;  
    k = s;  
    g (a);  
    g (S());  
    return 0;  
}
```

10. STL. Напишите шаблонную функцию, подсчитывающую сумму первых семи элементов заданного контейнера (списка или вектора, константного или неконстантного). Тип элементов контейнера является числовым типом. Если в заданном контейнере меньше 7 элементов, функция должна вернуть 0 соответствующего типа.

Коллоквиум, 2016 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. **18 9**

Вычеркнуть: строку "b = b + a;" поскольку фактическим параметром при вызове метода operator+(const B& b) из класса B является объект класса A. Приведение типа A к производному от типа A типу B приводит к синтаксической ошибке.

Критерии: Ошибка не найдена или неправильно объяснена =0. Лишнее вычеркивание, неверный ответ -5 за каждую ненаведенную ошибку.

2. Ответ. В конструктор преобразования надо добавить спецификатор explicit.

Критерии: Правильный ответ =10, иначе =0.

3. Ответ.

<pre>template <class TT> class P { public: TT * p, * p_null; P (TT * t) { p = t; } TT * operator -> () { if (p) return p; else return p_null = new TT; } ~P() { if (! p) delete p_null; } };</pre>	<pre>struct T { int n; T (int k = 10) {n = k;} };</pre>

Критерии: За неверную\отсутствующую операцию '->' -7. За ошибки в шаблоне и других местах -2 за каждую погрешность. За отсутствие деструктора и отсутствие освобождения динамической памяти не снижать.

4. Ответ: -10 A::operator-() 9 B::operator-() -10 B::operator-()

Критерии: Ошибки в ответе =0, иначе =10. Ошибки, не связанные с виртуальностью -2.

5. Ответ. Абстрактный класс – класс, содержащий хотя бы одну чистую виртуальную функцию.

Критерии: За неправильное определение =0. За отсутствие\неверный пример -5.

6. Ответ. A(), D(), A(), B(), ~B(), ~A(), A(), B(), ~B(), ~A(), A(),
 A(const A&), ~A(), ~A(), ~D(), ~A()

Критерии: За каждый лишний/пропущенный/переставленный конструктор/деструктор -4.

7. Например,

```
struct A { int y; static int x;  
          static int f (int k) { return x + k; }  
          // или          int f (int k) const { return x + k; }  
          A () { y = 5; }  
};  
int A::x = 0; // !!!
```

Критерии: Нет static при описании поля x **-5**. Нет static или const при описании метода f(int) **-5**. Не описано статическое поле x вне класса **-5**. Если поле y нестатическое и нет явного конструктора **-1**. Другие ошибки **-2** за каждую.

8. Модификация: Заключить в try-catch блок использование операций (и) операцию динамического приведения типа dynamic_cast, используя catch (bad_cast) {....}

Критерии: Правильный ответ **=10**, иначе **=0**.

9. Ответ. Неверные конструкции:

int && n; - ссылка на правое значение (r-value) должна быть инициализирована;
f (b); - нельзя передавать левое значение (l-value) выражения по правой ссылке;
n = nullptr; - тип nullptr_t не приводится к целому типу.

Критерии: За каждое неверное\отсутствующее\необъясненное вычеркивание **-4**.

10. Ответ.

```
template <class T>  
typename T::value_type sum5(const T& t){  
    typename T::value_type s = 0;  
    typename T::const_iterator p = t.end();  
    for (int i = 0; i < 5; i++) {  
        if (p == t.begin())  
            return 0;  
        s += *--p;  
    }  
    return s;  
}
```

Критерии: За пропуск typename **-5** (за все вместе). За использование неконстантного итератора **-5**. За ошибки в алгоритме **-5**. За другие ошибки **-2** за каждую.

Коллоквиум, 2016 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. 15 3

Вычеркнуть: строку "b = b * a;" поскольку фактическим параметром при вызове метода `operator*(const P& b)` из класса P является объект класса T. Приведение типа T к производному от T типу P приводит к синтаксической ошибке.

Критерии: Ошибка не найдена или неправильно объяснена =0. Лишнее вычеркивание, неверный ответ -5 за каждую ненаведенную ошибку.

2. Ответ. Надо удалить функцию преобразования `operator int()`.

Критерии: Правильный ответ =10, иначе =0.

3. Ответ.

<pre>template <class T> class B { public: T * p, * p_null; B (T * t) { p = t; } T& operator *() { if (p) return *p; else return *(p_null = new T); } ~B() { if (!p) delete p_null; } };</pre>	<pre>struct C { int n; C (int k = 20) { n = k; } };</pre>

Критерии: За неверную\отсутствующую операцию '*' -7. За ошибки в шаблоне и других местах -2 за каждую погрешность. За отсутствие деструктора и отсутствие освобождения динамической памяти не снижать.

4. Ответ: 4 T::operator--() 3 P::operator--() 4 P::operator--()

Критерии: Ошибки в ответе =0, иначе =10. Ошибки, не связанные с виртуальностью -2.

5. Ответ. Функция-друг класса – внешняя по отношению к классу функция, которой разрешен доступ к закрытым и защищенным членам класса.

Критерии: За неправильное определение =0. За отсутствие\неверный пример -5.

6. Ответ. A(), A(), B(), ~B(), ~A(), T(), ~T(), A(), B(), A(const A&), B(const B&); ~B(), ~A(), ~B(), ~A(), ~A()

Критерии: За каждый лишний/пропущенный/переставленный конструктор/деструктор -4.

7. Например,

```
struct B { int x;
          static int y;
          static int h (int k) { return y * k; }
          B () { x = 10; }
};
int B::y = 1; // !!!
```

Критерии: Нет static при описании поля y -5. Нет static при описании метода h (int) -5. Не описано статическое поле y вне класса -5. Если поле x нестатическое и нет явного конструктора -1. Другие ошибки -2 за каждую.

8. Модификация: Заключить в try-catch блок условный оператор в функции main (), используя catch (bad_typeid) {....}

Критерии: Правильный ответ =10, иначе =0.

9. Ответ. Неверные конструкции:

```
int && i; - ссылка на правое значение (r-value) ссылка должна быть инициализирована;
f (a);    - нельзя передавать левое значение (l-value) выражения по правой ссылке;
k = s;    - тип char * не приводится к типу nullptr_t.
```

Критерии: За каждое неверное\отсутствующее\необъясненное вычеркивание -4.

10. Ответ.

```
template <class T>
typename T::value_type sum7(const T& t){
    typename T::value_type s = 0;
    typename T::const_iterator p = t.begin();
    for (int i = 0; i < 7; i++) {
        if (p == t.end())
            return 0;
        s += *p++;
    }
    return s;
}
```

Критерии: За пропуск typename -5 (за все вместе). За использование неконстантного итератора -5. За ошибки в алгоритме -5. За другие ошибки -2 за каждую.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Ответьте на следующие вопросы. Обоснуйте ответ на каждый вопрос. (1) Верно ли, что каждая регулярная грамматика является неукорачивающей грамматикой? *Ответ:*

(2) Верно ли, что каждый регулярный язык является контекстно-зависимым языком? *Ответ:*

2. Дана регулярная грамматика G :

$S \rightarrow Bc \mid Ac$

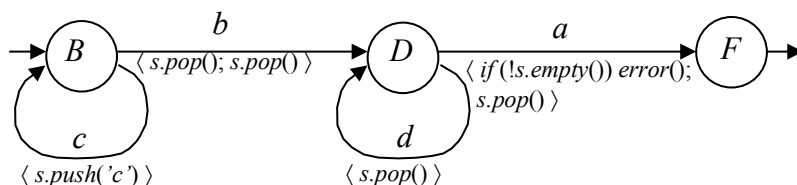
$B \rightarrow Ba \mid Bb \mid Aa \mid Ab$

$A \rightarrow b \mid Ba$

(1) Сколько состояний будет иметь ДКА A_1 , построенный по регулярной грамматике G (с проведением детерминизации, если нужно)? *Ответ:* _____. (2) Построить ДКА A_2 с тремя состояниями, такой что $L(A_1) = L(A_2)$. *Ответ:*

3. Дан перечень понятий: *отладчик, редактор связей, транслятор, драйвер, макрогенератор, редактор текстов, библиотеки стандартных подпрограмм*. Вычеркнуть из перечня одно лишнее понятие и назвать термин (его нет в перечне), в который входят все оставшиеся понятия. *Ответ:*

4. Дан автомат A с действиями над стеком s типа `stack<char>` (стек полагается потенциально бесконечным, вначале пуст, извлечение элемента из пустого стека операцией `pop()` – ошибка):



(1) Привести формулу языка $L(A)$
Ответ:

(2) С помощью КС-грамматики, анализируемой методом рекурсивного спуска, дополненной действиями вида `<cout<<' символ';>` реализовать формальный перевод τ :

$$\tau = \{ (x, y) \mid x \in L(A), y = x^R \}.$$

Ответ:

5. Дана постфиксная запись фрагмента программы на языке Си, в которой пропущены некоторые метки и операции перехода ('!', '!F'). Исходный фрагмент содержал одну логическую операцию ('&&' или '||') и не содержал ни одного оператора перехода (*goto*, *break* и т.п.). Вставьте пропущенные элементы в постфиксную запись и восстановите фрагмент на языке Си.

ПО-ЛИБ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	&x	x	1	+	=	;	a	b	==	17	!F	b	x	<	18	!	0
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	
		&y	#+	;	y	p	q	p	-	+	<		!F		!		

Ответ:

6. Объясните все роли ключевого слова **throw** языка C++ в различных смысловых контекстах. Например, знак '-' в зависимости от контекста играет роли унарного или бинарного минуса.
Ответ:

7. Вычеркните, если есть, ошибочные вызовы функций. Что напечатает получившаяся программа?

<pre>void g (const int & i) { cout << "g(const int&)\n"; } void g (int i) { cout << "g(int)\n"; } void f (int x) { cout << "f(int)\n"; } void f (short x) { cout << "f(short)\n"; }</pre>	<pre>int main () { int a = 48; const int b = 10; f((char) a); g((char) a); f(short(b)); g(b); }</pre>

Ответ:

8. Можно ли продемонстрировать динамический полиморфизм с помощью оператора-выражения вида $(b+a) + b$; ? Если да – приведите соответствующий пример, если нет – объясните почему. Перечислите другие виды полиморфизма в C++.

Ответ:

9. Опишите один класс A с не более чем тремя явно описанными методами (включая специальные) так, чтобы программа с данной функцией `main()` (и **без** каких-либо директив) успешно выполнялась.

<pre>int main () { A a; A(a, a); A((a, a)); }</pre>	<i>Ответ:</i>

10. Объясните, что делает данная программа:

```
vector<bool> v(10);

int main() { int k;
    for (int i = 0; i < v.size(); i++) { cin >> k; v[i] = k % 2; }
    for (vector<bool>::const_iterator p = v.begin(); p != v.end(); ++p)
        cout << *p;
}
```

Ответ:

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Ответьте на следующие вопросы. Обоснуйте ответ на каждый вопрос. (а) Верно ли, что каждая КС-грамматика является неукорачивающей грамматикой? *Ответ:*

(б) Верно ли, что каждый КС-язык является контекстно-зависимым языком? *Ответ:*

2. Дана регулярная автоматная грамматика G :

$$S \rightarrow bC$$

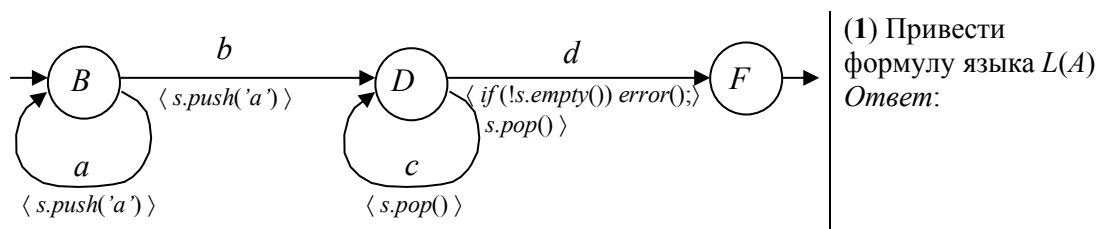
$$C \rightarrow aD \mid bD \mid c$$

$$D \rightarrow c \mid aD \mid bD \mid aC$$

(1) Сколько состояний будет иметь ДКА A_1 , построенный по грамматике G (с проведением детерминизации, если нужно)? *Ответ:* _____. (2) Построить ДКА A_2 с тремя состояниями, такой что $L(A_1) = L(A_2)$. *Ответ:*

3. Дан перечень понятий: *семантический анализ, подготовка к генерации объектного модуля, верификация, лексический анализ, генерация объектного модуля, синтаксический анализ*. Вычеркнуть из перечня один лишний термин и назвать понятие (его нет в перечне), в которое входят только все оставшиеся термины. *Ответ:*

4. Дан автомат A с действиями над стеком s типа $stack<char>$ (стек полагается потенциально бесконечным, вначале пуст, извлечение элемента из пустого стека операцией $pop()$ – ошибка):



(2) С помощью КС-грамматики, анализируемой методом рекурсивного спуска, дополненной действиями вида $\langle cout \ll ' \text{символ} ' ; \rangle$ реализовать формальный перевод τ :

$$\tau = \{ (x, y) \mid x \in L(A), x = y^R \}$$

Ответ:

5. Дана постфиксная запись фрагмента программы на Си, в которой пропущены некоторые метки и операции перехода ('!', '!F'). Исходный фрагмент содержал одну логическую операцию ('&&' или '||') и не содержал ни одного оператора перехода (*goto*, *break* и т.п.). Вставьте пропущенные в постфиксной записи элементы и восстановите фрагмент на Си.

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	&y	y	1	-	=	;	d	x	<	15	!F	1	18	!	b	x	<
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	
		&x	-#	;	y	q	p	q	+	+	>		!F		!		

Ответ:

6. Объясните все роли ключевого слова `static` языка C++ в различных смысловых контекстах. Например, знак '-' в зависимости от контекста играет роли унарного или бинарного минуса.
Ответ:

7. Вычеркните, если есть, ошибочные вызовы функций. Что напечатает получившаяся программа?

<pre>void h (const char &c) { cout << "h(const char&)\n"; } void h (char c) { cout << "h(char)\n"; } void f (const char x) { cout << "f(const char)\n"; } void f (int x) { cout << "f(int)\n"; }</pre>	<pre>int main () { char a = 57; const int b = 20; f(a); h((const char)a); f(short(b)); h((char)b); }</pre>

Ответ:

8. Можно ли продемонстрировать динамический полиморфизм с помощью оператора-выражения вида `a+(b+a)` ;? Если да – приведите соответствующий пример, если нет – объясните почему. Перечислите другие виды полиморфизма в C++.

Ответ:

9. Опишите один класс A с не более чем двумя явно описанными методами (включая специальные) так, чтобы программа с данной функцией `main()` (и без каких-либо директив) успешно выполнялась.

```
int main () { A a;
              A(a -> a);
              A(a -> a -> a);
}
```

Ответ:

10. Объясните, что делает данная программа:

```
vector<bool> v(10);

int main() { int k;
    for (int i=0; i < v.size(); i++) { cin >> k; v[i] = k % 2; }
    for (vector<bool>::const_reverse_iterator p = v.rbegin(); p != v.rend(); ++p)
        cout << *p;
}
```

Ответ:

Экзамен, 2016 год. Ответы и критерии оценок.

Вариант 1.

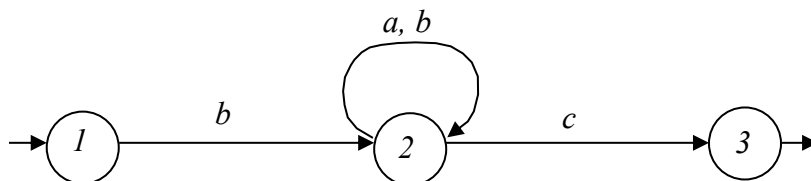
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. (1) Не верно. Регулярная грамматика $S \rightarrow aS \mid \varepsilon$ не является неукорачивающей (в соответствии с определением неукорачивающей грамматики).

(2) Верно. Если язык регулярный, то он задается регулярной грамматикой. Алгоритм устранения пустых правых частей дает эквивалентную регулярную грамматику, которая удовлетворяет определению КЗ-грамматики. Следовательно, регулярный язык – контекстно-зависимый.

Критерии: неверный ответ или не обоснованный, ошибка в обосновании: **-5** за каждый пункт.

2. Ответ. (1) У автомата A_1 будут пять состояний. **(2).** Диаграмма состояний автомата A_2 :



Критерии: за каждый неверно отвеченный пункт **-5**.

3. Ответ: Система программирования.

Критерии: за неправильное вычеркивание **-5**, за неправильный термин **-8**.

4. Ответ: (1) $L(A) = \{c^{n+2}bd^na \mid n \geq 0\}$. **(2)** $S \rightarrow cc \langle a \rangle Ba \langle cc \rangle$
 $B \rightarrow c \langle d \rangle Bd \langle c \rangle \mid b \langle b \rangle$

Критерии: за каждый неверно отвеченный пункт **-5**.

5. Ответ.

ПО- ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	&x	x	1	+	=	;	a	b	==	17	!F	b	x	<	18	!	0
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	
23	!F	&y	#+	;	y	p	q	p	-	+	<	34	!F	1	!		

```
do { x=x+1; if (a==b && b < x) y++; } while (y < p +( q - p ));
```

Критерии: за неверно заполненные элементы **-8**, за неверно восстановленный фрагмент **-5**, за отсутствие скобок в выражении "p+(q-p)" **-2**.

6. Ответ: (1) задает список исключений в заголовке функции, **(2)** в операции с операндом – генерирует исключение, **(3)** в операции без операнда – «пробрасывает» текущее обрабатываемое исключение на объемлющий уровень обработки; если текущего исключения нет, то приводит к ненормальному завершению программы.

Критерии: за отсутствие или ошибку в пункте **(1)** **-2**, за отсутствие или ошибку в пункте **(2)** **-2**, за отсутствие (или ошибку в объяснении) операции без операнда в пункте **(3)** **-4**, за отсутствие (или ошибку) описания особенностей операции без операнда в пункте **(3)** **-4**.

7. Ответ: Печать: `f(int)                      f(short)`
Вычеркивание: `g((char)a); g(b);`

Критерии: за каждый ошибочно вычеркнутый или оставленный вызов и за каждую неправильную печать -4.

8. Ответ: Другие виды полиморфизма – статический, параметрический.

Динамический продемонстрировать можно. Компилятор трактует выражение "(a+b)+a" как "(a.operator+(b)).operator+(a)". Полиморфизм для операции означает, что в зависимости от типа операндов вызываются разные методы с одинаковым обозначением операции '+'. Чтобы сработал именно динамический полиморфизм, методы должны вызываться через таблицу виртуальных методов (ТВМ), в данном выражении это возможно, если 'a' и 'b' являются ссылками на базовый класс для разных полиморфных объектов из иерархии. Возможный пример:

```
class A { public:
    virtual A& operator+(A& a) { cout << "+ from A | "; return a; } };
class B: public A { public:
    A& operator+(A& a) { cout << "+ from B | "; return *this; } };
int main() { A a1; B b1; A &a = a1, &b = b1;

    (a+b)+a ; // (a.operator+(b)).operator+(a);
              // Вызов виртуального метода через ссылку на базовый класс, используется ТВМ
              // Напечатается: + from A | + from B |

    return 0;
}
```

Критерии: за каждый не названный (или не существующий) вид полиморфизма: -2, за неверный ответ или отсутствующее обоснование про параметрический полиморфизм: -6.

Примечание: Если приведен пример, где в двух различных контекстах за счет динамического полиморфизма выражение "(a+b)+a" имеет разный смысл (a не в одном выражении операция '+' имеет разный смысл), то оценка не снижается – такой ответ тоже считается правильным, так как ограничений на единственность вхождений в пример выражения "a+(b+a)" в условии нет.

9. Ответ:

<pre>int main () { A a; A (a, a); A ((a, a)); }</pre>	<pre>struct A { A (int a = 0, int b = 0) { } operator int() { return 1; } A operator, (A a) { return a; } };</pre>

Критерии: за превышение числа методов (> 3) и/или использование неразрешенных возможностей языка -2, за каждое неправильное определение метода класса А -4, за каждую неработающую (при компиляции или выполнении) строчку в функции main() -4.

10. Ответ: Показанная программа вводит 10 целых чисел из входного потока, определяя их четность и создавая контейнер логических элементов (четным числам соответствуют элементы *false*, нечетным – элементы *true*). Затем значения элементов созданного контейнера выдаются в выходной поток.

Критерии: за неправильный ответ 0.

Экзамен, 2016 год. Ответы и критерии оценок.

Вариант 2.

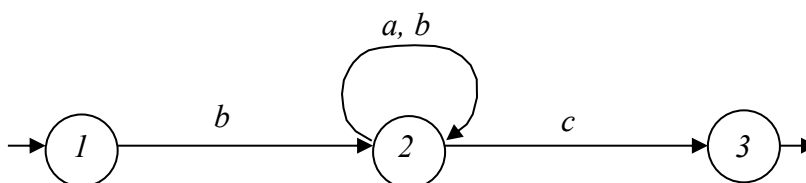
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. (1) Не верно. КС-грамматика $S \rightarrow aSb \mid \varepsilon$ не является неукорачивающей (в соответствии с определением неукорачивающей грамматики).

(2) Верно. Если язык контекстно-свободный, то он задается КС-грамматикой. Алгоритм устранения пустых правых частей дает эквивалентную КС-грамматику, которая удовлетворяет определению КЗ-грамматики. Следовательно, КС-язык КЗ.

Критерии: неверный ответ, ответ без обоснования, ошибка в обосновании: **-5** за каждый пункт.

2. Ответ. (1) У автомата A_1 будут пять состояний. **(2).** Диаграмма состояний автомата A_2 :



Критерии: за каждый неверно отвеченный пункт **-5**.

3. Вычеркнуть слово '~~верификация~~'. Описывается процесс *компиляции* (этапы компиляции).

Критерии: за неправильное вычеркивание **-5**, за неправильный термин **-8**.

4. Ответ: (1) $L(A) = \{ a^n b c^{n+1} d \mid n \geq 0 \}$ **(2)** $S \rightarrow \langle dc \rangle A c d$
 $A \rightarrow a \langle c \rangle A c \langle a \rangle \mid b \langle b \rangle$

Критерии: за каждый неверно отвеченный пункт **-5**.

5. Ответ.

ПО-ЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	&y	y	1	-	=	;	d	x	<	15	!F	1	18	!	b	x	<
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
	23	!F	&x	-#	;	y	q	p	q	+	+	>	34	!F	1	!	

`do { y=y-1; if (d<x || b<x) --x; } while (y > q + ( p + q ));`

Критерии: за неверно заполненные элементы **-8**, за неверно восстановленный фрагмент **-5**, за отсутствие скобок в выражении " $q + (p + q)$ " **-2**.

6. Ответ: (1) для глобальных объектов или функций делает их невидимыми в других модулях, **(2)** внутри функции делает локальную переменную статической, она сохраняет значения между вызовами функции, **(3)** в классе помечает статические члены-данные (принадлежащие классу, а не отдельным объектам) и статические методы (для работы со статическими данными).

Критерии: за отсутствие или ошибку в пункте **(1)** **-2**, за отсутствие или ошибку в пункте **(2)** **-2**, за отсутствие (или ошибку в объяснении) членов-данных в пункте **(3)** **-4**, за отсутствие (или ошибку в объяснении) членов-функций в пункте **(3)** **-4**. Если в пункте **(3)** говорится просто про статические члены (без подразделения на члены-данные и члены-методы) – не снижать.

7. Ответ: Печать: `f(const char); f(int)`
Вычеркивание: `h((const char)a); h((char)b);`

Критерии: за каждый ошибочно вычеркнутый или оставленный вызов и за каждую неправильную печать -4.

8. Другие виды полиморфизма – статический, параметрический.

Динамический продемонстрировать можно. Компилятор трактует выражение "a+(b+a)" как "a.operator+(b.operator+(a))". Полиморфизм для операции означает, что в зависимости от типа операндов вызываются разные методы с одинаковым обозначением операции '+'. Чтобы сработал именно динамический полиморфизм, методы должны вызываться через ТВМ, – в данном выражении это возможно, если 'a' и 'b' являются ссылками на базовый класс для разных полиморфных объектов из иерархии. Приведем возможный пример:

```
class A { public:
    virtual A& operator+(A& a) { cout << "+ from A | "; return *this; } };
class B: public A { public:
    A& operator+(A& a) { cout << "+ from B | "; return *this; } };
int main() { B b1; A a1, &a=a1, &b=b1;

    a+(b+a); // a.operator+(b.operator+(a));
              // Вызов виртуального метода через ссылку на базовый класс, используется ТВМ
              // Напечатается: + from B | + from A |
    return 0;
}
```

Критерии: за каждый не названный (или не существующий) вид полиморфизма: -2, за неверный ответ или отсутствующее обоснование про параметрический полиморфизм: -6.

Примечание: Если приведен пример, где в двух различных контекстах за счет динамического полиморфизма выражение "a+(b+a)" имеет разный смысл (а не в одном выражении операция '+' имеет разный смысл), то оценка не снижается – такой ответ тоже считается правильным, так как ограничений на единственность вхождений в пример выражения "a+(b+a)" в условии нет.

9. Ответ:

```
struct A { A * a;
    A (A * b = 0) { a = this; }
    A * operator-> () { return this; }
};
```

Критерии: за превышение числа методов (> 2) и/или использование неразрешенных возможностей языка -2, за каждое неправильное определение метода класса А -4, за каждую неработающую (при компиляции или выполнении) строчку в функции main() -4.

10. Программа вводит десять чисел, превращая их в ноль или единицу и заполняя ими булевский вектор, после чего распечатывает его обратном порядке.

Критерии: за неправильный ответ 0.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Вычеркните из функции `main()` блоки, содержащие ошибки компиляции (предупреждения ошибками не считаются). Что будет напечатано получившейся после вычеркивания программой?
Соглашение: указатель можно без потерь преобразовать к типу `long`, а форма с многоточием поддерживает передачу любых объектов. (Стандарт C++14, компилятор ничего не оптимизирует).

<pre>struct A { A(int a=0) {cout<<1;} A(const A&a) {cout<<2;} ~A() {cout<<endl;} };</pre>	<pre>struct B: A { B(B& b) {cout<<3;} B(const B& b) {cout<<4;} }; struct C: B { C(...) :B(*this) {cout<<5;} };</pre>	<pre>int main() { {A a((long)&a);} {A d=d;d;} {B b;} {const B b=b;} {C c(1);} {return 0;} }</pre>	<u>Ответ:</u>

2. Измените данную программу, добавляя только операции разрешения области видимости '::' так, чтобы было напечатано число **39**.

<pre>int x=0; namespace X { // int x = 39; namespace Y { int x = 17; } const int x = 9; }</pre>	<pre>struct A { int x; struct B { int x; B(){ x = 39;}} x1; A(){ x = 22; } public: int h(){ { int x = 39; } return x + x + x; } }; int main(){ A a; cout << a.h(); return 0; }</pre>

3. Не изменяя оператор `return` и не используя символы `'/'`, `'\"'`, `'\"'`, `'%'`, `'#'`, добавьте всё необходимое в описание класса `A` так, чтобы компиляция прошла без ошибок.

```
class A { public: int f(){ return (i,j)[i,j](i,j);} };
```

Ответ:

4. Перечислите имена классов, описания которых содержат ошибки.

<pre>struct A { int register f() { return 1; } int const f(int x) { return 2; } };</pre>	<pre>struct B { static int f() { return 1; } int f() const { return 2; } };</pre>	<pre>struct C { static int f(const int& x) { return 1; } int f(int& x) { return 2; } };</pre>	<pre>struct D { static virtual int f() { return 0; } };</pre>

Ответ:

5. Что напечатает данная программа?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class L { public: virtual void f (char x = 'L') {cout << x << "0" << endl; }};
class P : public L { public: void f (char x = 'P') {cout << x << "1" << endl; }};
int main() { P p; L * pl = &p; pl->f(); dynamic_cast<P*>(pl)->f(); return 0; }
```

Ответ:

6. Что будет напечатано в результате выполнения следующей программы?

<pre>#include <iostream> using namespace std; int f (int n){ try { if (n<=0) throw n; f(--n); } catch (int& k){cout << k++; throw; throw "b"; } catch (const char *s) { cout << s; } throw "d"; }</pre>	<pre>int main() { try { cout << 'a'; f(2); cout << 'c'; } catch (int k) { cout << k; } catch (const char *s){ cout << s; } cout << 'e'; return 0; }</pre>

Ответ:

7. Объясните ошибки компиляции для данной программы. Не исправляя имеющиеся строки, вставьте новую строку кода, в которой нет символов '/', '\', '"', '%', '#', '(', ')', так, чтобы программа скомпилировалась. **Примечание:** угловые скобки использовать можно.

<pre>typedef int T; T p(T a){ return a; } float p(float a){ return a; }</pre>	<pre>int main() { return int (p('a') + p(3.14)); }</pre>

8. Может ли абстрактный класс в языке C++ являться абстрактным типом данных? Если да, то приведите пример, если нет – объясните почему. Ответ:

9. Добавьте в функцию main() один оператор так, чтобы при выполнении программы произошла ошибка обращения по ссылке к уже не существующему объекту. Объясните причину ошибки. **Примечание:** повторный вызов деструктора и обращение к членам объекта после его работы сами по себе не являются ошибками.

```
class List { char elem; List & next;
public:
    List(): elem ('\n'), next (* this){}
    List(const char*s): elem(*s? *s: '\n'), next(*s? *new List(s + 1) : *this){}
    List(const List& l):elem(l.elem), next(&l!=&l.next? *new List(l.next): *this){}
    void print() const { cout << elem; if(& next != this) next.print();}
    ~List() { if (& next != this) { delete & next; } }
};
int main() { List b("abc");

    b.print();

}
```

10. Приведите функцию, эквивалентную данной, не содержащую новых конструкций из C++11.

<pre>#include <list> int f(int x, const std::list<int>& l){ auto p=l.begin(); decltype (x) s = 0; while (p != l.end()) s += * p++ == x; return s; }</pre>	<u>Ответ:</u>

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Вычеркните из функции `main()` блоки, содержащие ошибки компиляции (предупреждения ошибками не считаются). Что будет напечатано получившейся после вычеркивания программой?
Соглашение: указатель можно без потерь преобразовать к типу `long`, а форма с многоточием поддерживает передачу любых объектов. (Стандарт C++14, компилятор ничего не оптимизирует).

<pre>struct X { X(int x=0) {cout<<1;} X(const X&x) {cout<<2;} ~X(){cout<<endl;} };</pre>	<pre>struct Y: X { Y(){cout<<3;} Y(Y& y):X(*this) {cout<<4;} }; struct Z: Y { Z(...){cout<<5;} };</pre>	<pre>int main() { {X x((long)&x);} {Y c(c=c);} {const X b=b;} {X p; Z q=p;} {Z z;} {return 0;} }</pre>	<u>Ответ:</u>

2. Измените данную программу, добавляя только операции разрешения области видимости `::` так, чтобы было напечатано число 27.

<pre>int t=0; namespace A { // int t = 27; namespace B { int t = 14; } const int t = 11; }</pre>	<pre>struct X { int t; struct Y { int t; Y(){ t = 27;}} y1; X(){ t = 13; } public: int h(){ { int t = 27; } return t + t + t; } }; int main(){ X b; cout << b.h(); return 0; }</pre>

3. Не изменяя оператор `return` и не используя символы `'/'`, `'\"'`, `'\"'`, `'%'`, `'#'`, добавьте всё необходимое в описание класса `B` так, чтобы компиляция прошла без ошибок.

```
class B { public: int f(){ return b--->a<---b; } // пробелов в выражении нет
};
```

Ответ:

4. Перечислите имена классов, описания которых содержат ошибки.

<pre>struct W { int static g(const int& x) { return 1;} int const g(int&x) { return 2; } };</pre>	<pre>struct X { static int g() const { return 0; } };</pre>	<pre>srtuct Y { static int g() { return 1;} virtual int g() const { return 2; } };</pre>	<pre>struct Z { int register&g() { return 1; } int const g (int x) { return 2; } };</pre>

Ответ:

5. Что напечатает данная программа?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class C { public: virtual void f (char x = 'C') {cout << x << "1" << endl; }};
class D : public C { public: void f (char x = 'D') {cout << x << "2" << endl; }};
int main(){ D d; C & rc = d; rc.f(); dynamic_cast<D&>(rc).f(); return 0; }
```

Ответ:

6. Что будет напечатано в результате выполнения следующей программы?

<pre>int f (int n){ try { if (n>0) throw n; throw "b"; } catch (int k){ cout << k--; f(k); } catch (const char *s) { cout << s; throw; } }</pre>	<pre>int main() { try { cout << 'a'; f(2); cout << 'c'; } catch (int k) { cout << k; } catch(const char *s){ cout << s; } cout << 'e'; return 0; }</pre>

Ответ:

7. Объясните ошибки компиляции для данной программы. Не исправляя имеющиеся строки, вставьте новую строку кода, в которой нет символов '/', '\', '\'', '%', '#', '(', ')', так, чтобы программа скомпилировалась. **Примечание:** угловые скобки использовать можно.

<pre>typedef int T; short g(short b){ return b; } T g(T b){ return b; }</pre>	<pre>int main() { return int (g('b') + g(9.8)); }</pre>

8. Может ли некоторый класс в языке C++ являться абстрактным классом и при этом не быть абстрактным типом данных? Если да, то приведите пример, если нет – объясните почему.

Ответ:

9. В процессе выполнения данной программы происходит ошибка обращения к уже не существующему объекту ("по висячему указателю"). Объясните причину ошибки. Не изменяя функции main() и представление данных, исправьте класс List так, чтобы этой ошибки не возникало.

Примечание: повторный вызов деструктора и обращение к членам объекта после его работы сами по себе не являются ошибками.

```
class List { char elem; List * next;
public:
    List(): elem ('\n'), next (this){}
    List(const char*s): elem(*s? *s:'\n'), next(*s ? new List (s + 1): this){}
    List(const List& l):elem(l.elem), next(&l!=l.next ? new List(*l.next): this){}
    void print() const { cout << elem; if (next != this) next -> print();}
    ~List() { if (next != this) delete next; }
};
int main() { List c("cde"); c.~List(); c.print(); }
```

Ответ:

10. Приведите функцию, эквивалентную данной, не содержащую новых конструкций из C++11.

```
#include <vector>
int f(int y, const std::vector<int>& v){
    auto q=v.rbegin();
    decltype(y) s=0;
    while (q != v.rend()) s += * q++ == y;
    return s;
}
```

Ответ:

Коллоквиум, 2017 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: 1 → 2 → 14 → 135 →. **Решение.** Область действия описываемого имени объекта начинается сразу после вхождения его описателя (описатели бывают вида `p`, `*p`, `p()` – функция, `p[]` – массив) в описание, перед инициализатором, если он есть (пункт 3.3.2 Стандарта 2011). В языках Си и C++ (в отличие от стандартного Паскаля) не считается ошибкой использование неинициализированного объекта (обращение к нему). То есть в выражении инициализации можно использовать описываемое имя объекта, например, для взятия адреса объекта.

[Обращение к полям (или методам) объекта или к его базовой части до начала выполнения конструктора или после окончания выполнения деструктора приводит (согласно пункту 12.7 Стандарта) к неопределенному поведению (*undefined behavior*) программы: например, в одной реализации такая программа может корректно выполняться (пункт 1.4.2 и сноска 2 в нем), в другой – приводить к ошибке компиляции или выполнения (см. 1.3.25)]. [Передача потенциально-вычисляемого аргумента классического типа (9), имеющего нетривиальный конструктор копирования, нетривиальный конструктор перемещения или нетривиальный деструктор, в отсутствие соответствующего параметра, поддерживается условно; семантика такой передачи определяется реализацией (см. 5.2.2.7)]. [Компилятор Visual Studio не поддерживает форму инициализации `B b(b);`]. В первом блоке действует конструктор с целым параметром, напечатается "1". Во втором блоке описание `A d=d=d` трактуется как `A d(d=d)`, то есть сначала выполняется операция присваивания, сгенерированная автоматически (которая в данном примере по сути ничего не делает), затем конструктор копирования (который тоже по сути ничего не копирует), но напечатает "2". В третьем блоке ошибка, так как в классе B нет конструктора умолчания, это вычеркивается. В четвертом блоке описание `const B b=b` трактуется как `const B b(b)`, то есть срабатывает константный конструктор копирования (который по сути ничего не копирует). Для базового класса вызывается конструктор умолчания, напечатается "1", затем конструктор производного класса напечатает число "4". В пятом блоке для объекта `c` вызывается единственный конструктор ("на все случаи жизни"), который явно вызывает конструктор копирования (неконстантный) класса B, для которого вызывается конструктор умолчания класса A. В результате напечатается "135". (Конструктор может быть с многоточием, поскольку он является хотя и специальной, но функцией, а функция в C++ может быть описана с многоточием). В шестом блоке есть лишь правильный оператор `return`, ошибки нет. При выходе из каждого блока прорабатывается деструктор базового класса, осуществляющий перевод строки ("→"). Для проверки компилятором: `$> g++ -std=c++14 -fno-elide-constructors var1_1.cpp`

Критерии: за каждое неверное вычеркивание (или невычеркивание) для всех блоков, кроме второго и четвертого -3, за каждую ошибочную напечатанную строку (в соответствии с невычеркнутыми блоками) -1, за неверное разбиение на строки -1, за вычеркивание второго и/или четвертого блока (всего) -1.

2. Ответ: изменить выражение в операторе `return`: `x + ::x + X::Y::x;`

Критерии: за каждое неверное слагаемое (порядок слагаемых может быть любым, вместо `x` может быть `A::x`) -4. Комментарий «`// x=39;`» ни на что не влияет, это проверка внимательности.

3. Ответ:

```
class A { int a; public: A(int a=0) { this->a = a; }
    A operator[] (int i) { return A(i); }
    A operator , (A a) { return a; }
    int operator () (A a1, A a2) { return a2.a + a1.a; }
    operator int () { return a; }
    int f() { A i,j; return (i,j)[i,j](i,j); }
};
```

Возможны вариации решения. Запрет использования `'/'` и `'''` не позволяет закомментировать фрагмент или сделать его строкой, запрет `'#'` не позволяет макросы, запрет `'\'` – эскейп-последовательности и переносы в макросах, запрет `'%'` не позволяет использовать диграф `':'` как альтернативу для `'#'`. [Триграфы исключены из Стандарта 2017].

Критерии: за неверную трактовку выражения -5, за каждую ошибку при описании класса -2.

4. Ответ: A, B, D. В классе A ошибочно использован спецификатор `register` – он применяется только к данным. В классе B статическая функция не может быть перегружена нестатической функцией с такими же параметрами. В классе C ошибок нет, нужная функция выбирается в зависимости от константности фактического параметра. В классе D ошибка в том, что статическая функция не получает скрытый параметр `this` на объект, от имени которого она вызвана, и поэтому не может непосредственно обращаться к нестатическим полям своего объекта (в том числе к скрытому полю `pvtbl`, содержащему адрес таблицы виртуальных методов – ТВМ) и, следовательно, не может быть виртуальной.

Критерии: за каждый пропущенный или лишний класс -3.

5. Ответ: L1 P1

Параметр по умолчанию для функции, вызванной через указатель или ссылку, определяется статически (в момент компиляции) по типу указателя или ссылки.

Критерии: за каждую неверную строку -5.

6. Ответ: a0123e (исключение перебрасывается в предыдущий рекурсивный вызов с увеличением на 1).

Критерии: за каждый неверно напечатанный (или пропущенный) символ -3.

7. Ответ: ошибка в неоднозначности для вызова `p(3.14)`: на шаге (в) алгоритма поиска наиболее подходящей функции (best-matching) подходят обе функции `p()`. Нужно вставить строку `template<class T> непосредственно перед T p(T a) {return a;}`. Запрет круглых скобок не позволяет добавить перегруженную без шаблона функцию.

Критерии: если неверно объяснена ошибка (не найдена, найдены лишние) -5, если неверно вставлена строка (или не вставлена) -5.

8. Ответ: В C++ АД реализуется с помощью классов (структур), в которых нет открытых членов-данных. Абстрактный класс содержит чистую виртуальную функцию. Следовательно, ответ положительный, вот пример абстрактного класса, являющегося АД:

```
class A { public: virtual void f() = 0; }
```

Критерии: за неверный (отсутствующий) ответ, ошибочные примеры или объяснения =0.

9. Ответ: `int main() { List b("abc"); b.~List(); b.print(); }`

При явном вызове деструктора освобождается память, занимаемая звеньями списка, причем значение поля `next` объекта `b` становится неопределенным, но при следующем вызове `b.print()` делается попытка его использования. [Повторный вызов деструктора относится к неопределенному поведению (12.4.15). После окончания работы деструктора, обращение к членам объекта также относится к неопределенному поведению (12.7.1). Это означает широкий диапазон поведения в разных реализациях: в одной реализации такая программа считается корректной (что предполагалось в условии задачи), в другой может фиксироваться ошибка, так как объект «перестает существовать» с момента первого вызова деструктора].

Критерии: если не предложено верное добавление -5, если нет правильного объяснения -5.

10. Ответ:

```
int f (int x, const std::list<int>& l){
    std::list<int>::const_iterator p = l.begin();
    int s = 0;
    while(p != l.end()) s += * p++ == x;
    return s;
}
```

Критерии: за каждую неверно замененную конструкцию (`auto`, `decltype`) -5.

Коллоквиум, 2017 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: 1 → 24 → 2 → 135 →. **Решение.** В первом блоке действует конструктор с целым параметром, напечатается "1". Согласно пункту 5.2.10.4 Стандарта 2011, указатель может быть преобразован к целому типу, достаточному для его представления. Во втором блоке сначала выполняется операция присваивания (для вычисления выражения в скобках), сгенерированная автоматически (которая в данном примере по сути ничего не делает), затем конструктор копирования (который тоже по сути ничего не копирует), сначала он явно вызывает конструктор базового класса, который напечатает "2", после чего конструктор производного класса напечатает "4". В третьем блоке описание `const X b=b;` трактуется как `const X b(b)`, то есть срабатывает конструктор копирования (который по сути ничего не копирует). Напечатается "2". В четвертом блоке ошибка, так как делается попытка инициализировать объект производного класса объектом базового класса, это вычеркивается. В пятом блоке для объекта `z` вызывается единственный конструктор ("на все случаи жизни"), вызывающий конструктор умолчания класса `B`, для которого вызывается конструктор умолчания класса `A`. В результате напечатается "135". (Конструктор может быть с многоточием, поскольку он является хотя и специальной, но функцией, а функция в C++ может быть описана с многоточием). В шестом блоке есть лишь правильный оператор `return`, ошибки нет. При выходе из каждого блока прорабатывается деструктор базового класса, осуществляющий перевод строки ("→"). Для проверки компилятором: `$> g++ -std=c++14 -fno-elide-constructors var2_1.cpp`

Критерии: за каждое неверное вычеркивание (или невычеркивание) для всех блоков, кроме второго и третьего -3, за каждую ошибочную напечатанную строку (в соответствии с невычеркнутыми блоками) -1, за неверное разбиение на строки -1, за вычеркивание второго и/или третьего блока (всего) -1.

2. Ответ: нужно изменить выражение в операторе `return t + ::t + A::B::x;`

Критерии: за каждое неверное слагаемое (порядок слагаемых может быть любым, вместо `t` может быть `X::t`) -4. Комментарий «`// x=27;`» ни на что не влияет, это проверка внимательности.

3. Ответ: Если `b` описать как объект класса `B`, содержащего поле `a`, то выражение можно трактовать следующим образом:

```
b.operator--(int).operator->()<b.operator-().operator--().operator int().
```

Для этого необходимо определить в классе необходимые операции и, возможно, конструкторы.

```
class B { int a; public: B(int a=0) { this->a=a; }
        B & operator--() { a--; return *this; }
        B operator--(int) { B temp(*this); a--; return temp; }
        B * operator->() { return this; }
        B operator-() { B temp(-a); return temp; }
        operator int() { return a; }
        int f() { B b; return b---->a<---b; }
};
```

Возможны вариации решения. Запрет использования `/'` и `'''` не позволяет закомментировать фрагмент или сделать его строкой, запрет `#` не позволяет макросы, запрет `\` – эскейп-последовательности и переносы в макросах, запрет `%` не позволяет использовать диграф `%'` как альтернативу для `#`.

Критерии: за неверную трактовку выражения -5, за каждую ошибку при описании класса -2.

4. Ответ: X, Y, Z. В классе `W` ошибок нет, нужная функция выбирается в зависимости от константности фактического параметра. В классе `X` ошибка в том, что статическая функция по определению не пользуется нестатическими полями (в том числе обозначаемым полем `this`) и, следовательно, не может быть константной. В классе `Y` статическая функция не может быть перегружена нестатической функцией с такими же параметрами. В классе `Z` ошибочно использован спецификатор `register` – он применяется только к данным в блоке или параметрам функции (7.1.1.2). [В Стандарте 2017 слово `'register'` всё ещё остаётся зарезервированным, но уже ничего не означает].

Критерии: за каждый пропущенный или лишний класс -3.

5. Ответ: C2 D2

Параметр по умолчанию для функции, вызванной через указатель или ссылку, определяется статически (в момент компиляции) по типу указателя или ссылки.

Критерии: за каждую неверную строку -5.

6. Ответ: a21bbe (функция рекурсивно вызывается из обработчика).

Критерии: за каждый неверно напечатанный (или пропущенный) символ -3.

7. Ответ: ошибка в неоднозначности для вызова `g(9.8)`: на шаге (в) алгоритма поиска наиболее подходящей функции (best-matching) подходят обе функции `g()`. Нужно вставить строку `template<class T>` непосредственно перед `T g(T b) { return b; }`. Запрет круглых скобок не позволяет добавить перегруженную без шаблона функцию.

Критерии: если неверно объяснена ошибка (не найдена, найдены лишние) -5, если неверно вставлена строка (или не вставлена) -5.

8. Ответ: В C++ АДД реализуется с помощью классов (структур), в которых нет открытых членов-данных. Абстрактный класс содержит чистую виртуальную функцию. Следовательно, ответ положительный, вот пример абстрактного класса, не являющегося АДД:

```
class A { public: int a; virtual void f() = 0; }
```

Критерии: за неверный (отсутствующий) ответ, ошибочные примеры или объяснения =0.

9. Ответ: `~List() { if(next!=this) { delete next; next=this; elem='\n'; } }`

При явном вызове деструктора освобождается память, занимаемая звеньями списка, причем значение поля `next` объекта `s` становится неопределенным, но при следующем вызове `s.print()` делается попытка его использования.

Критерии: если не предложено верное добавление -5, если нет правильного объяснения -5.

10. Ответ:

```
#include <vector>
int f (int y, const std::vector<int>& v) {
    std::vector<int>::const_reverse_iterator q = v.rbegin();
    int s=0;
    while (q != v.rend()) s += *q++ == y;
    return s;
}
```

Критерии: за каждую неверно замененную конструкцию (auto, decltype) -5.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Описать класс **A** и одну функцию **sum(x,y)**, вычисляющую сумму своих двух аргументов, так, чтобы в приведенной функции `main()` не было ошибок.

```
int main () {
    A a, b(5);
    return sum (a,b) + sum <int> (17, 18);
}
```

2. Что напечатает данная программа? Укажите прототипы конструкторов и деструкторов классов **A** и **B** в порядке их выполнения.

<pre>struct A { virtual void f () { cout << "A_f\n"; } virtual void g () { cout << "A_g\n"; } }; struct B: A { void f () { cout << "B_f\n"; } };</pre>	<pre>int main () { try { A a; B b; throw b; } catch (A & a) { a.f(); a.g(); } catch (...) { cout << "catch(...)\n"; } return 0; }</pre>

3. Ничего не удаляя и не исправляя структуру **D** и функцию **main()**, добавить к нижеприведенному фрагменту необходимые описания так, чтобы в получившейся программе не было ошибок, а на экран выдалось **6**.

```
struct D : B { int func1 (int a) { return a; } };
int main () {
    D d;
    cout << d.func1(B::x) + d.func2(2) + d.B::func1(1) << endl;
    return 0;
}
```

4. Если при компиляции следующей программы будут обнаружены ошибки, найдите их и объясните причины, иначе укажите, что будет выдано на печать.

```
Char fff (char c) { return c + 1; }
double fff (double d) { return d + 2; }
namespace N {
    int fff (int i) { return i + 3; }
    float fff (float f) { return f + 4; }
}
int main () { cout << fff(3) << fff(5.0f) << endl; return 0; }
```

5. STL. Что такое итераторы? Какие категории итераторов реализованы в STL? Чем разные категории итераторов отличаются друг от друга?

6. а) В чем различие статических и динамических библиотек?

б) Какие компоненты вычислительной системы и на каком этапе работы занимаются подключением к программам каждого из этих видов библиотек?

7. а) Эквивалентны ли следующие грамматики? **Ответ обосновать.**

б) Каков **тип** каждой из заданных **грамматик**?

в) Каков **тип языка**, порождаемого каждой грамматикой?

$$G_1: S \rightarrow S I \mid S 2 \mid I \mid 2$$

$$G_2: S \rightarrow IA \mid 2A$$

$$G_3: S \rightarrow ABS \mid I \mid 2$$

$$A \rightarrow IA \mid 2A \mid \varepsilon$$

$$AB \rightarrow BA$$

$$A \rightarrow IA \mid \varepsilon$$

$$B \rightarrow 2B \mid \varepsilon$$

8. а) **Вставить** в одну из грамматик предыдущего задания № 7 **действия** вида $\langle \text{cout} << \text{"символы"}; \rangle$ так, чтобы во время анализа **методом рекурсивного спуска** цепочек ω исходного языка L (порождаемого грамматикой) был реализован перевод

$$\tau = \{(\omega, I^k 2^m) \mid \omega \in L, k = |\omega|_1, m = |\omega|_2\}$$

б) Возможно ли вставить подобные действия в другую грамматику из задания № 7? **Ответ обосновать.**

9. а) Для заданной грамматики построить эквивалентную неукорачивающую грамматику, пользуясь **алгоритмом удаления пустых правых частей в КС-грамматике**.

б) Построить **конечный автомат** по получившейся грамматике. Если получившийся автомат недетерминирован, преобразовать его к **ДКА**, иначе построить по нему **анализатор**.

$$S \rightarrow Aa \mid Bb$$

$$A \rightarrow Bb \mid b \mid \varepsilon$$

$$B \rightarrow Aa \mid \varepsilon$$

10. Записать постфиксной записи фрагмент программы на языке Си:

`a = x < --y + 7 ? -x * (a = b = 5) : b - a - 1;`

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	3 1

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Описать класс **B** и одну функцию **neg(x)**, меняющую знак своего аргумента на противоположный, так, чтобы в приведенной функции `main()` не было ошибок.

```
int main () {
    B a, b(7);
    return neg (a) * neg<B> (b) * neg<int> (-5);
}
```

2. Что напечатает данная программа? Укажите прототипы конструкторов и деструкторов классов **B** и **D** в порядке их выполнения.

<pre>struct B { virtual void f () { cout << "B_f\n"; } virtual void g () { cout << "B_g\n"; } }; struct D: B { void f () { cout << "D_f\n"; } };</pre>	<pre>int main () { try { D d; throw d; } catch (B a) { a.f(); a.g(); } catch (...) {cout<<"catch(...)\n";} return 0; }</pre>

3. Ничего не удаляя и не исправляя структуру **B** и функцию `main()`, добавить к нижеприведенному фрагменту необходимые описания так, чтобы в получившейся программе не было ошибок, а на экран выдалось **10**.

```
struct B : A { int g2 (int a) { return a; } };
int main () {
    B b;
    cout << b.g1 (6) + b.g2 (4) + b.A::g1 (2) + A::x << endl;
    return 0;
}
```

4. Если при компиляции следующей программы будут обнаружены ошибки, найдите их и объясните причины, иначе укажите, что будет выдано на печать.

```
int f2 (int i) { return i + 1; }
long double f2 (long double ld) { return ld+ 2; }
namespace M {
    char f2 (char c) { return c + 3; }
    double f2 (double d) { return d + 4; }
}
int main () { cout << f2('3') << f2(5.0) << endl; return 0; }
```

5. Какие типы стандартных исключений автоматически генерируются компилятором при динамической идентификации типа (механизм **RTTI**)? Объясните, в каких ситуациях они генерируются.

6. а) Какие стратегии трансляции могут выбираться в различных системах программирования?

б) В чем отличия этих подходов к реализации процесса трансляции?

в) Приведите по одному примеру формального языка, к которому применяется каждая из названных вами стратегий.

7. а) Эквивалентны ли следующие грамматики? **Ответ обосновать.**

б) Каков **тип** каждой из заданных **грамматик**?

в) Каков **тип языка**, порождаемого каждой грамматикой?

$$G_1: \begin{aligned} S &\rightarrow bA \\ A &\rightarrow aC \\ B &\rightarrow aBb \mid \varepsilon \end{aligned}$$

$$G_2: \begin{aligned} S &\rightarrow bAa \\ A &\rightarrow aAb \mid \varepsilon \\ C &\rightarrow aCb \mid \varepsilon \end{aligned}$$

$$G_3: \begin{aligned} S &\rightarrow bAa \\ bA &\rightarrow bC \\ C &\rightarrow Bba \mid \varepsilon \end{aligned}$$

8. а) **Вставить** в одну из грамматик предыдущего задания № 7 **действия** вида $\langle \text{cout} << \text{"символы"}; \rangle$ так, чтобы во время анализа **методом рекурсивного спуска** цепочек ω исходного языка L (порождаемого грамматикой) был реализован перевод

$$\tau = \{(\omega, b^k a^m) \mid \omega \in L, k = |\omega|_b, m = |\omega|_a\}$$

б) Возможно ли вставить подобные действия в другую грамматику из задания № 7? **Ответ обосновать.**

9. а) Для заданной грамматики построить эквивалентную неукорачивающую грамматику, пользуясь **алгоритмом удаления пустых правых частей в КС-грамматике**.

б) Построить **конечный автомат** по получившейся грамматике. Если получившийся автомат недетерминирован, преобразовать его к **ДКА**, иначе построить по нему **анализатор**.

$$\begin{aligned} S &\rightarrow 0A \mid 1B \\ A &\rightarrow 0A \mid 0B \\ B &\rightarrow 1A \mid 1B \mid \varepsilon \end{aligned}$$

10. Записать постфиксной записи фрагмент программы на языке Си:

do { $x=z=(x + y + 5)*b-3/f$; $b++$; } **while** ($-x+y>0$)

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33		

Экзамен, 2017 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. `class A { // Возможны варианты решения !!!
 int x;
 public: A (int a = 0) { x = a; }
 int operator + (const A & b) { return x + b.x; }
};
template <class T>
int sum (const T & x, const T & y) {return x + y; }`

Критерии: нет\неверное описание шаблона функции -5, неверное описание класса (нет нужных конструкторов, не перегружена '+') -5, если параметры в sum() и operator+() передаются по ссылке, то за неконстантную ссылку -2, если решение не требует использования операций (например, в классе реализована в функция преобразования объекта в int), то за отсутствие перегрузки операций не снижать.

2. Ответ. `B_f` `A_g`

`A(), A(), B(), A(const A&), B(const B&), ~B(), ~A(), ~A(), ~B(), ~A()`

Критерии: неверная печать -5, каждое нарушение порядка, лишней\отсутствующий конструктор\деструктор (за парный к отсутствующему конструктору деструктор не снижать) -3.

3. Ответ. `struct B { // Возможны варианты решения !!!
 static int x;
 int func1 (int a) { return a; }
 int func2 (int a) { return a; }
};
int B::x = 3;`

Критерии: за отсутствие описания статической переменной x -5, за отсутствие описания x вне класса, нет необходимых методов в B, другие ошибки -3 за каждую.

4. Ответ. Ошибка одна – неоднозначный выбор перегруженной функции в вызове fff (3) – обнаруживается на шаге (в) работы соответствующего алгоритма.

Критерии: нет объяснения ошибки, ошибка не найдена =0, наряду с правильным ответом найдены другие ошибки -5 за каждую.

5. Ответ. Итератор – тип данных, объекты которого играют роль указателей при работе с контейнерами STL и используются для доступа к элементам контейнеров. Категории итераторов – (1) ввода, (2) вывода, (3) однонаправленный, (4) двунаправленный, (5) произвольного доступа. Категории итераторов отличаются определенным для них набором операций.

Критерии: нет\неверное понятие итератора =0, пропущены категории итераторов -2 за каждую, не указаны отличия категорий итераторов -5.

6. Ответ. Статические и динамические библиотеки различаются способом подключения (моментом времени подключения) их фрагментов к формируемой полной программе. Статические компоненты извлекаются и подключаются к программе редактором связей на этапе редактирования связей и формирования исполняемого модуля. Динамические – при выполнении программы в соответствии с точками динамического вызова, формируемыми также редактором связей. Загружаются динамические компоненты в память загрузчиком.

Критерии: неверно указанное отличие -5, неверное указание компонентов -3 за каждый, неверное указание этапа подключения -3 за каждый.

7. Ответ: да, эквивалентны, поскольку порождают один и тот же язык: $L = \{\alpha \mid \alpha \in \{1, 2\}^+\}$
 G_1 – тип 3, регулярная левостолбчатая (автоматная), G_2 – тип 3, регулярная правостолбчатая,
 G_3 – тип 0. Тип языка – тип 3, регулярный.

Критерии: неверно ответ на 1 вопрос -5, неверное указание типа грамматики -3 за каждый, неверное указание типа языка -3 за каждый.

8. Ответ. (а) G_2 : $S \rightarrow 1 \text{ <cout << '1';> } A \mid 2A \text{ <cout << '2';> }$
 $A \rightarrow 1 \text{ <cout << '1';> } A \mid 2A \text{ <cout << '2';> } \mid \varepsilon$

(б) **Нет, нельзя:** G_1 – не контекстно-свободная, метод рекурсивного спуска неприменим, G_3 – метод рекурсивного спуска неприменим из-за левой рекурсии в правилах вывода

Критерии: неверно выбрана грамматика и\или вставлены действия -5, неверное обоснование выбора -5.

9. Ответ. Неукорачивающая $S \rightarrow Aa \mid Bb \mid a \mid b$ ДКА: $q(H, a) = \underline{BS}$
грамматика: $A \rightarrow Bb \mid b$ $q(H, b) = \underline{AS}$
 $B \rightarrow Aa \mid a$ $q(BS, b) = \underline{AS}$
 $q(AS, a) = \underline{BS}$

Критерии: неверно построена неукорачивающая грамматика -5, нет\неверно построен ДКА -5.

10. Ответ. $a = x < -y + 7 \text{ ? } -x * (a = b = 5) : b - a - 1;$

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	&a	x	&y	-#	7	+	<	20	!F	x	@	&a	&b	5	=
16	17	18	19	20	21	22	23	24	25	26	20	21	22	23	
=	*	25	!	b	a	-	1	-	=	;	b	a	-	1	
24	25	26	27												
-	=	;													

Критерии: пропуск & (адреса), каждой операции -2, пропуск ';' -1, каждая неверная позиция операции -4.

Экзамен, 2017 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. `class B { // Возможны варианты решения !!!
 int x;
 public: B (int a = 0) { x = a; }
 int operator - () { return -x; }
};
template < class T >
int neg (const T & t) {return -t; }`

Критерии: нет\неверное описание шаблона функции -5, неверное описание класса (нет нужных конструкторов, не перегружена '+') -5, если параметры в `neg()` и `operator-()` передаются по ссылке, то за неконстантную ссылку -2, если решение не требует использования операций (например, в классе реализована в функция преобразования объекта в `int`), то за отсутствие перегрузки операций не снижать.

2. Ответ. `B_f`  `B_g` 
`B(), D(), B(const B&), D(const D&), ~D(), ~B(), B(const B&), ~B(), ~D(), ~B()`

Критерии: неверная печать -5, каждое нарушение порядка, лишней\отсутствующий конструктор\деструктор (за парный к отсутствующему конструктору деструктор не снижать) -3.

3. Ответ. `struct A { // Возможны варианты решения !!!
 static int x;
 int g1 (int a) { return a; }
 int g2 (int a) { return a; }
};
int A::x = -2;`

Критерии: за отсутствие описания статической переменной `x` -5, за отсутствие описания `x` вне класса, нет необходимых методов в `B`, другие ошибки -3 за каждую.

4. Ответ. Ошибка одна – неоднозначный выбор перегруженной функции в вызове `f2(5.0)` – обнаруживается на шаге (в) работы соответствующего алгоритма.

Критерии: нет объяснения ошибки, ошибка не найдена =0, наряду с правильным ответом найдены другие ошибки -5 за каждую.

5. Ответ. (1) *bad_cast* – при невозможности приведения типов полиморфных классов с помощью операции *dynamic_cast*. (2) *bad_typeid* – при попытке раскрытия нулевого указателя в аргументе операции *typeid*.

Критерии: нет\неверное объяснение *bad_cast* -5, нет\неверное объяснение *bad_typeid* -5

6. Ответ. Стратегии трансляции: (1) **компиляция** (вся программа с ЯП переводится в машинные коды - Паскаль, Си, ...), (2) **интерпретация** (в машинные коды переводится конструкция за конструкцией ЯП и сразу выполняется - любой скриптовый язык,...), (3) **смешанная стратегия** (транслятор переводит программу на ЯП в программу на промежуточном языке, которая затем интерпретируется, JavaScript ... в принципе почти любой интерпретируемый язык переводится в некоторое внутреннее представление).

Критерии: пропуск каждой стратегии -4, неверное объяснение сути каждой стратегии -3, нет\неверный пример -2 за каждый.

7. Ответ. Да, эквивалентны, поскольку порождают один и тот же язык: $L = \{b a^n b^n a \mid n \geq 0\}$

G_1 – тип 2 (контекстно-свободная), G_2 – тип 2 (контекстно-свободная), G_3 – тип 0. Тип языка = 2 (контекстно-свободный).

Критерии: неверный ответ на 1 вопрос -5, неверное указание типа грамматики -3 за каждый, неверное указание типа языка -3 за каждый.

8. Ответ. $G_1: S \rightarrow b \text{cout} << 'b'; > A$

$A \rightarrow aC \text{cout} << 'a'; >$

$B \rightarrow a \text{cout} << 'b'; > B \text{cout} << 'a'; > b \mid \varepsilon$

$C \rightarrow \text{cout} << 'b'; > B \text{cout} << 'a'; > ba \mid \varepsilon$

(б) Нет, нельзя: G_3 - не контекстно-свободная грамматика, метод рекурсивного спуска неприменим, G_2 - этот метод неприменим из-за нарушения критерия: $first(A) \cap follow(A) = \{a\}$.

Критерии: неверно выбрана грамматика и/или вставлены действия -5, неверное обоснование выбора -5.

9. Ответ. Неукорачивающая $S \rightarrow 0A \mid 1B \mid 1$ ДКА: $q(S, 0) = A$
 грамматика: $A \rightarrow 0A \mid 0B \mid 0$ $q(S, 1) = \underline{BF}$
 $B \rightarrow 1B \mid 1A \mid 1$ $q(A, 0) = \underline{ABF}$
 $q(\underline{BF}, 1) = \underline{ABF}$
 $q(\underline{ABF}, 0) = \underline{ABF}$
 $q(\underline{ABF}, 1) = \underline{ABF}$

Критерии: неверно построена неукорачивающая грамматика -5, нет\неверно построен ДКА -5.

10. Ответ. `do {x = z = (x + y + 5) * b - 3/f; b++; } while (-x + y > 0)`

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	&x	&z	x	y	+	5	+	b	*	3	f	/	-	=	=	;	&b
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	
	#+	;	x	@	y	+	0	>	30	!F	1	!					

Критерии: пропуск & (адреса), каждой операции -2, пропуск ';' -1, каждая неверная позиция операции -4.

1	2	3

1. (50 баллов) Дана программа на языке программирования C++.

```

1. #include <iostream>
2. #include <typeinfo>
3. using namespace std;
4. class A {
5.     int x, y;
6.     A(const A& b) {}
7. public:
8.     A (int a = 0, int b = 0) { x = a; y = b;}
9.     int g() noexcept { return 555;}
10. };
11.
12. class B : public A {
13.     int z;
14.     B (const B& b) {}
15. public:
16.     int w;
17.     B (int b) { z = b; }
18. };
19.
20. B& operator- (B& b) {
21.     b.z = -b.z;
22.     return b;
23. }
24.
25. int x = 1;
26.
27. void f(float y) {
28.     cout << "::f \n";
29. }
30. template <class T> void f(T t) noexcept {
31.     cout << "f<> \n";
32. }
33.
34. namespace nnn {
35.     int x = 2;
36.     void f(int c) {
37.         cout << "nnn::f \n";
38.     }
39. }
40. using namespace nnn;
41.
42. int main () {
43.     try {
44.         B b1, b2(6), *pb;
45.         const B b;
46.         A a1, a2(3), *pa = &b2, *pa1 = &a1;
47.         pb = dynamic_cast <B*>(pa1);
48.         if (pb) cout << typeid (*pb).name() << endl;
49.         f(1);
50.         cout << ::x << endl;
51.         cout << b.w++ << endl << b.g() << endl;
52.         throw *pa;
53.     }
54.     catch (B&) { cout << "error BBB " << endl; }
55.     catch (A&) { cout << "error AAA " << endl; }
56.     catch( ... ) { cout << "error ... " << endl; }
57.     return 0;
58. }

```

Для этой программы: (а) **Найти и обязательно объяснить** все ошибки, обнаруженные при компиляции приведенной программы на C++ (с элементами новых стандартов). Если в какой-то конструкции программы несколько ошибок, укажите и объясните их все.

(б) **Исправить найденные ошибки, не меняя текст ниже 19-й строки и внося только необходимые** для компиляции программы исправления. **Нельзя** закрывать члены-данные заданных классов делать открытыми. Правильные конструкции заменять, комментировать или удалять нельзя.

(в) **Что будет выдано на печать** при работе получившейся правильной программы?

2. (40 баллов) Дана программа на языке программирования C++.

```
#include <iostream>
using namespace std;

struct A {
    int a;
    A (int n):a(n){}
    virtual void f() {cout << "f()_A\n";}
    virtual int g()=0;
};

struct T {
    int a,b;
    T(int x = 0, int y = 0) {
        a = x; b = y;
    }
};

int main () {
    B b;
    A* pa = &b;
    pa -> f();
    cout << pa->a << endl;
    T t;
    P p(&t);
    p->a = p->b = 3;
    cout << t.a << ' ' << t.b << endl;
    P p1(NULL);
    cout << p1->a << ' ' << p1->b << endl;
    return 0;
}
```

Для этой программы требуется добавить необходимые описания, никоим образом не изменяя и не удаляя приведенный текст, так, чтобы в получившейся программе не было ошибок, а на экран выдалось:

```
f()_B
5
3 3
0 0
```

3. (10 баллов) STL. (а) Что такое алгоритм в STL? (б) Каким образом алгоритмы STL получают доступ к тому или иному контейнеру? (в) Какое принято соглашение с целью определения того, к каким контейнерам применим тот или иной алгоритм STL?

1	2	3

1. (50 баллов) Дана программа на языке программирования C++.

```

1. #include <iostream>
2. #include <typeinfo>
3. using namespace std;
4. class B {
5.     int x;
6.     public:
7.         B (int a = 0) { x = a;}
8.         B (B & b) {x = b.x;}
9.         int g() { return 222;}
10. };
11.
12. class D : public B {
13.     int z;
14.     public:
15.         int y;
16.         D (int b) { z = b; }
17. };
18.
19. B operator+ (B& b1, B& b2) {
20.     return b1.x + b2.x;
21. }
22.
23. int x = 1;
24.
25. void f (float y) {
26.     cout << "::f(float) \n";
27. }
28. void f (char y) {
29.     cout << "::f(char) \n";
30. }
31. template <class T> void f (T t){
32.     cout << "f<> \n";
33. }
34.
35. namespace kkk {
36.     int x = 2;
37.     void f (int c) {
38.         cout << c << "kkk::f(int) \n";
39.     }
40. }
41. using namespace kkk;
42.
43. int main () {
44.     try {
45.         const D d;
46.         D && rrr = D();
47.         const B & rb = d;
48.         const D & rd = dynamic_cast <const D &>(rb);
49.         f(::x);
50.         f(10.5);
51.         cout << --d.y << endl << d.g() << endl;
52.         throw rrr;
53.     }
54.     catch( B& ) { cout << "error BBB " << endl; }
55.     catch( ... ) { cout << "error ... " << endl; }
56.     return 0;
57. }

```

Для этой программы: (а) Найти и обязательно объяснить все ошибки, обнаруженные при компиляции приведенной программы на C++ (с элементами новых стандартов). Если в какой-то конструкции программы несколько ошибок, укажите и объясните их все.

(б) Исправить найденные ошибки, не меняя текст ниже 18-й строки и внося только необходимые для компиляции программы исправления. Нельзя закрытые члены-данные заданных классов делать открытыми. Правильные конструкции заменять, комментировать или удалять нельзя.

(в) Что будет выдано на печать при работе получившейся правильной программы?

2. (40 баллов) Дана программа на языке программирования C++.

```
#include <iostream>
using namespace std;

struct B {
    int b;
    B (int x){ b = x;}
    virtual void func () { cout << "f()_B\n"; }
    virtual void h (double)=0;
};

struct T {
    int a;
    T (int x = 1) { a = x; }
};

int main () {
    D d;
    C c;
    B *bp = &d, *bp1 = &c;
    bp -> func();
    bp1-> func();
    T t;
    P p(&t);
    (*p).a = 28;
    cout << t.a << endl;
    P p1(NULL);
    cout << (*p1).a << endl;
    return 0;
}
```

Для этой программы требуется добавить необходимые описания, никоим образом не изменяя и не удаляя приведенный текст, так, чтобы в получившейся программе не было ошибок, а на экран выдалось:

```
f()_D
f()_B
28
1
```

3. (10 баллов) STL. (а) Что такое итератор STL? (б) Перечислите категории итераторов STL.

**Коллоквиум, 2018 год. Ответы и критерии оценок.
Вариант 1.**

1. Максимум 50 баллов. Ошибки:

- 1) с. 21 – обращение к закрытому члену класса (сделать эту функцию другом В).
- 2) с. 44 – при создании объекта b1 - нет конструктора умолчания (например, внести в конструктор класса В значение параметра по умолчанию).
- 3) с. 47 – операция `dynamic_cast` применена к указателю непалиморфного класса (любым способом сделать класс В полиморфным).
- 4) с. 51 – а) попытка изменить поле константного объекта (сделать его статическим: добавить `static` перед `int w;` и вне класса описание статического поля: `int B::w=0;` с некоторой инициализацией для дальнейшей печати).
- 5) с. 51 – б) применение неконстантного метода к константному объекту (сделать метод `g()` константным: `const` перед `noexcept`, если метод `g()` виртуальный, статическим он быть не может).
- 6) с. 52 – при генерации объекта исключения вызывается конструктор копирования класса А, наположенный в закрытой области (перенести его в открытую область или перегрузить в открытой области без признака константности `const`, конструктор копирования класса В никак не задействован, его изменять не нужно).

На печать будет выдано:

```
nnn : f
1
0      (или другое инициализирующее значение)
555
error AAA
```

Критерии: Ошибка не найдена или неправильно объяснена, лишняя ошибка -5 за каждую. Неверное исправление -5 за каждое. Нет описания статического поля вне класса -2. Неверная печать – 5 за каждый неверный элемент печати.

2. Максимум 40 баллов. //Возможны варианты решения!!!

```
struct B : A {
    B() : A(5) {}
    virtual void f() { cout << "f()_B\n"; }
    virtual int g() { return 555; }
};

class P {
    T* pt;
    bool del;
public:
    P(T* t) {
        del = t ? false : true;
        pt = t ? t : new T();
    }
    T* operator -> () { return pt; }
    ~P() { if (del) delete pt; }
};
```

Критерии: Нет отношения наследования между А и В -20. Неверная\отсутствующая операция '`->`' -10. Нет списка инициализации в конструкторе В(), нет конструктора -10. Не переопределена функция `g()` -10. Не освобождается используемая динамическая память -2. Неверная печать -5 за каждый неверный элемент печати.

3. Максимум 10 баллов. Ответ. (а) Алгоритм STL – шаблонная функция, выполняющая некоторое действие над некоторым контейнером. **(б)** Доступ к контейнеру осуществляется посредством итераторов контейнера, передаваемых в качестве параметров функции-шаблона. **(в)** Алгоритм применим к контейнерам, тип итераторов которых "не ниже" обозначенного именем формального параметра шаблона.

Критерии: Неверный ответ на пункт (а) -4. Неверный ответ на пункт (б) -3. Неверный ответ на пункт (в) -3.

Коллоквиум, 2018 год. Ответы и критерии оценок.
Вариант 2.
Максимум за все три задачи в сумме 100 баллов

1. Максимум 50 баллов. Ошибки:

- 1) с. **20** – обращение к закрытому члену класса (сделать эту функцию другом В)
- 2) с. **20** – при формировании возвращаемого по значению объекта-результата вызывается конструктор копирования. Ему по неконстантной ссылке передаётся временный объект, что запрещено (сделать параметр конструктора копирования класса В константным).
- 3) с. **45** – при создании объекта d – нет конструктора умолчания (например, внести в конструктор класса D значение параметра по умолчанию).
- 4) с. **48** – операция `dynamic_cast` применена к ссылке непалиморфного класса (любым способом сделать класс В полиморфным).
- 5) с. **51** – а) попытка изменить поле константного объекта (сделать его статическим: добавить `static` перед `int y`; и вне класса описание статического поля: `int D::y=0`; (какая-то инициализация для дальнейшей печати).
- 6) с. **51** – б) применение неконстантного метода к константному объекту (сделать метод `g()` константным: если метод `g()` сделан виртуальным, статическим он быть не может).

На печать будет выдано: **kkk::f(int)**

f <>

-1 (если инициализирующее значение 0, или, соответственно, иное)

222

error BBB

Критерии: Ошибка не найдена или неправильно объяснена, лишняя ошибка **-5 за каждую**. Неверное исправление **-5 за каждое**. Нет описания статического поля вне класса **-2**. Неверная печать **-5 за каждый неверный элемент печати**.

2. Максимум 40 баллов. Возможны варианты решения!!!

```
struct C : B {
    C(): B(12) { }
    void h(double d) { cout << d << endl;}
};
struct D : C {
    void func () { cout << "f()_D\n"; }
};
class P {
    T* pt;
    bool bb;
public:
    P(T* t){
        bb = t ? true: false;
        pt = bb ? t: new T();
    }
    T& operator* () { return *pt; }
    ~P(){ if (!bb) delete pt; }
};
```

Критерии: Нет отношения наследования между В, С и\или D **-20**. Неверная\отсутствующая операция `*` **-10**. Нет списка инициализации в конструкторе `C()`, нет конструктора `C()` **-10**. Не переопределена функция `h(double)` **-10**. В методе `operator*` результат возвращается НЕ по ссылке **-5**. Не освобождается используемая динамическая память **-2**. Неверная печать **-5 за каждый неверный элемент печати**.

3. Ответ.

а) Итератор **STL** - шаблонный класс, объекты которого выполняют роль указателя по отношению к некоторым контейнерам.

б) Выделяют 5 категорий итераторов:

1. **Вывода** (output) (запись в контейнер – '*p=', '++').
2. **Ввода** (input) (считывание из контейнера – '*p', '->', '++', '==', '!=') .
3. **Однонаправленный** (forward) (считывание и запись в одном направлении - '*p=', '->', '*p', '++', '==', '!=') .
4. **Двунаправленный** (bidirectional) ('*p=', '*p', '->', '++', '--', '==', '!=') – list, map, set.
5. **С произвольным доступом** (random_access) (операции всех 4-х категорий плюс '[]', '+', '-', '+=', '-=', '<', '>', '<=', '>=') – vector, deque.

в) Типы итераторы различаются набором применимых к ним операций.

Критерии: Неверный ответ на пункт а) -4. Неверный ответ на пункт б) -1 за каждый пропущенный тип итераторов. Неверный ответ на пункт в) -3.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Ответьте на следующие вопросы. Обоснуйте ответ на каждый вопрос.

(а) Может ли грамматика одновременно быть регулярной и не быть контекстно-зависимой?

(б) Может ли конечный язык быть неоднозначным?

(в) Существует ли регулярный язык, который не порождается неукорачивающей грамматикой?

2. Постройте грамматику выражений над переменными a и b с операциями сложения и умножения и унарным минусом (наивысший приоритет), к которой применим метод рекурсивного спуска. Добавьте в грамматику действия только вида $\langle \text{cout} \ll \text{'СИМВОЛ'}; \rangle$, реализующие формальный перевод $\tau = \{ (x, y) \mid x - \text{арифметическое выражение, } y - \text{эквивалентное арифметическое выражение, не содержащее двух рядом стоящих унарных минусов} \}$.

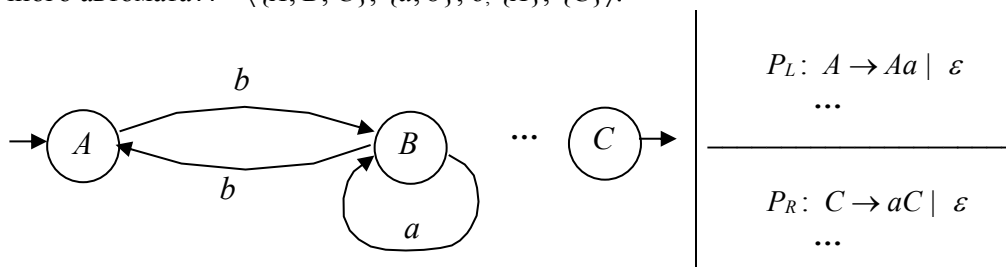
Пример: $-(a + - - b * - - - (a)) \Rightarrow - (a + b * - (a))$.

3. Даны названия программных систем: *CVS*, *SVN*, *GIT*, *Make*. Вычеркнуть среди них одно лишнее и дать определение понятию, примерами которого являются все оставшиеся системы.

Ответ:

4. Даны: **1)** часть множества P_L левостолбчатой грамматики $G_L = \langle \{A, B, C\}, \{a, b\}, P_L, C \rangle$; **2)** часть множества P_R правостолбчатой грамматики $G_R = \langle \{A, B, C\}, \{a, b\}, P_R, A \rangle$; **3)** часть множества дуг δ конечного автомата $A = \langle \{A, B, C\}, \{a, b\}, \delta, \{A\}, \{C\} \rangle$.

δ :



Пополнить δ , P_L , P_R (дописать недостающие правила, дорисовать дуги) так, чтобы одновременно выполнялись условия: **(а)** грамматики G_L и G_R , приведённые и эквивалентные; **(б)** A получается из G_L алгоритмом построения конечного автомата по левостолбчатой грамматике; **(в)** A получается из G_R алгоритмом построения конечного автомата по правостолбчатой грамматике; **(г)** A не является детерминированным; **(д)** количество дуг в A не превосходит 6.

5. Дана постфиксная запись фрагмента программы на Си. Восстановите этот фрагмент на языке Си без операторов перехода. Приведите еще один (отличающийся, но также без операторов перехода) текстовый фрагмент, имеющий такую же постфиксную запись. **Примечание:** пустой оператор не занимает место в постфиксной записи.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПО- ЛИЗ	&b	+#	7	!F	7	!	&a	-#	;	a	b	+	17	!F	1	!	

Ответ:

6. Если в функции `main()` есть блоки, содержащие ошибки компиляции, вычеркните их. Что будет напечатано получившейся программой? Считать, что компилятор ничего не оптимизирует.

<pre>struct A { A(int a){cout<<1;} A(const A&a = A(1)) {cout<<2;} A(A& a) {cout<<3;} ~A() {cout<<'a';} };</pre>	<pre>struct B: A { B(const B& b) {cout<<4;} B(int i=0):A(*this) {cout<<5;} ~B cout<<'b';} };</pre>	<pre>int main(){ {cout<< "start";} {cout<<endl; A a(1);} {cout<<endl; A a;} {cout<<endl; B b;} return 0; }</pre>	Ответ:

7. (а) Привести описания не более двух функций, чтобы были верны следующие обращения:

`g(50L);` `g((const double) 1);` `g();` `g("str");` `g(0,0);` `g(3,"str");`

(б) Может ли константный член класса быть статическим? Обоснуйте ответ.

8. Может ли в выражении вида $a+b$ проявиться одновременно и статический, и динамический полиморфизм? Обоснуйте ответ. Какой еще вид полиморфизма есть в C++?

9. Что будет напечатано в результате работы данной программы? В курсе рассматривались стандартные исключения `bad_alloc`, `bad_cast`, `bad_typeid`.

<pre>#include <iostream> #include <exception> #include <typeinfo> using namespace std; class List { char elem; List * next; int ballast[30000000]; public: List(const char*s): elem(*s? *s:'\n'), next(*s? new List(s+1):NULL){} List & operator*= (List & l) { List *q=&l, *p=this; while (p->next!=NULL) p=p->next; while (q->next!=NULL){ p->elem = q -> elem; p->next = new List(""); q = q -> next; p = p -> next; } return *this; }</pre>	<pre>~List(){ if(next!=NULL) {delete next;} } }; int main() { List w("xyz"); try { try { throw '5'; throw 5; } catch (int) { cout<< "int ";} catch (char){ cout<< "char "; w *= w; throw; } } catch (char){cout<< "char"<< endl;} catch (exception & e){ cout<<typeid(e).name()<<endl; } return 0; }</pre>

Ответ:

10. Опишите шаблонную функцию `Reverse(a)`, которая по контейнеру `a` (вектор, список, очередь и т.п.) строит в качестве значения новый контейнер, в котором элементы `a` расположены в обратном порядке. Например, для списка $\langle 1, 2, 3 \rangle$ функция построит список $\langle 3, 2, 1 \rangle$.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Ответьте на следующие вопросы. Обоснуйте ответ на каждый вопрос.

(а) Может ли грамматика одновременно быть контекстно-свободной и не быть контекстно-зависимой?

(б) Может ли регулярный язык быть неоднозначным?

(в) Существует ли контекстно-свободный язык, который не порождается неукорачивающей грамматикой?

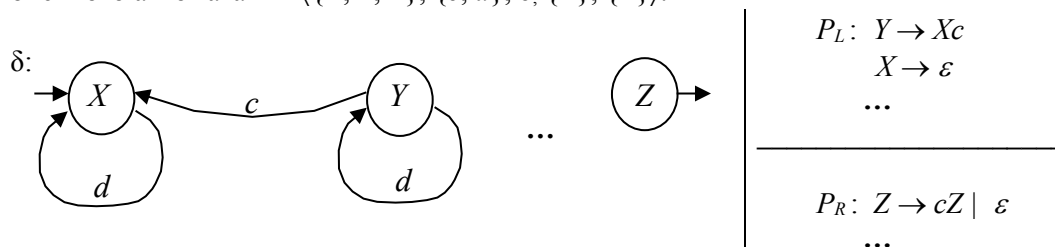
2. Постройте грамматику логических выражений над переменными a и b , к которой применим метод рекурсивного спуска. Добавьте в грамматику действия только вида $\langle cout \ll \text{'СИМВОЛ'}; \rangle$, реализующие формальный перевод $\tau = \{ (x, y) \mid x - \text{логическое выражение, } y - \text{эквивалентное логическое выражение, не содержащее двух рядом стоящих операций } not \}$.

Пример: $not(a \text{ or } not \text{ not } b \text{ and } not \text{ not } not(a)) \Rightarrow not(a \text{ or } b \text{ and } not(a))$.

3. Даны названия программных средств: *CORBA*, *COM*, *Java Beans*, *Valgrind*. Вычеркнуть среди них одно лишнее и дать определение понятию, примерами которого являются все оставшиеся средства.

Ответ:

4. Даны: **1)** часть множества P_L левостолбчатой грамматики $G_L = \langle \{X, Y, Z\}, \{c, d\}, P_L, Z \rangle$; **2)** часть множества P_R правостолбчатой грамматики $G_R = \langle \{X, Y, Z\}, \{c, d\}, P_R, X \rangle$; **3)** часть множества дуг δ конечного автомата $A = \langle \{X, Y, Z\}, \{c, d\}, \delta, \{X\}, \{Z\} \rangle$.



Пополнить δ, P_L, P_R (дописать недостающие правила, дорисовать дуги) так, чтобы одновременно выполнялись условия: (а) грамматики G_L и G_R , приведённые и эквивалентные; (б) A получается из G_L алгоритмом построения конечного автомата по левостолбчатой грамматике; (в) A получается из G_R алгоритмом построения конечного автомата по правостолбчатой грамматике; (г) A не является детерминированным; (д) количество дуг в A не превосходит 6.

5. Дана постфиксная запись фрагмента программы на Си. Восстановите этот фрагмент на языке Си (без операторов перехода). Приведите еще один (отличающийся, но также без операторов перехода) текстовый фрагмент, имеющий такую же постфиксную запись. **Примечание:** пустой оператор не занимает место в постфиксной записи.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПО- ЛИЗ	a	b	+	17	!F	&b	#-	12	!F	12	!	&a	#+	;	1	!	

Ответ:

6. Если в функции `main()` есть блоки, содержащие ошибки компиляции, вычеркните их. Что будет напечатано получившейся программой? Считать, что компилятор ничего не оптимизирует.

<pre>struct C { C(int c) {cout<<1;} C(const C& c=C(1)) {cout<<2;} C(C & c) {cout<<3;} ~C() {cout<<'c';} };</pre>	<pre>struct D: C { D(const D&d) {cout<<4;} D(int i=0) {cout<<5;} ~D() {cout<<'d';} };</pre>	<pre>int main(){ {cout<< "start";} {cout<<endl;C c=C(1);} {cout<<endl;C c(1); } {cout<<endl;D d; } return 0; }</pre>	Ответ:

7. (а) Привести описания не более двух функций, чтобы были верны следующие обращения:

```
h("str");    h();    h((const int)2.0);    h(0,0);    h("b",10);    h(10,"0.0");
```

(б) Может ли статический член класса быть константным? Обоснуйте ответ.

8. Может ли в выражении вида `a[b]` проявиться одновременно и статический, и динамический полиморфизм? Обоснуйте ответ. Какой еще вид полиморфизма есть в C++?

9. Что будет напечатано в результате работы данной программы? В курсе рассматривались стандартные исключения `bad_alloc`, `bad_cast`, `bad_typeid`.

<pre>#include <iostream> #include <exception> #include <typeinfo> using namespace std; class List { char elem; List * next; int ballast[20000000]; public: List(const char*s): elem(*s? *s:'\n'), next(*s? new List(s+1):NULL){} List & operator+= (List & l) { List *p=&l, *q=this; while (q->next!=NULL) q=q->next; while (p->next!=NULL){ q->elem=p->elem; q->next = new List(""); p=p->next; q=q->next; } return *this; } };</pre>	<pre>~List(){ if(next!=NULL) {delete next;} } }; int main() { List b("abc"); try{ try { throw 5; throw '5'; } catch (char) { cout<< "char ";} catch (int) { cout<< "int "; b+=b; throw; } } catch (int) {cout<<"int"<< endl;} catch(exception & e){ cout<<typeid(e).name()<<endl; } return 0; }</pre>

Ответ:

10. Опишите шаблонную функцию *Concat* (*a*, *b*), которая по двум однотипным контейнерам *a* и *b* (векторы, списки, очереди и т.п.) строит в качестве значения новый контейнер, содержащий конкатенацию *a* и *b*. Например, для списков $\langle 1, 2, 3 \rangle$ и $\langle 4, 5 \rangle$ конкатенацией будет $\langle 1, 2, 3, 4, 5 \rangle$.

Экзамен, 2018 год. Ответы и критерии оценок.

Варианты 1 и 3.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (а) Да, может. Например, регулярная грамматика $S \rightarrow aS \mid \varepsilon$ не удовлетворяет ограничениям КЗ-грамматики: если для начального символа S есть правило $S \rightarrow \varepsilon$, то S не должен встречаться в правых частях других правил, но есть правило $S \rightarrow aS$.

(б) Нет, не может. Для конечного языка с n цепочками можно построить грамматику с n альтернативами для начального символа, где в правых частях – цепочки языка. Каждая цепочка в такой грамматике выводится за один шаг единственным образом.

(в) Нет, не существует. Любую регулярную грамматику можно преобразовать в эквивалентную регулярную неукорачивающую с помощью алгоритма устранения правил с пустой правой частью.

Критерии: за каждый пункт: если неверный ответ: **-4**, если ответ верный, но не обоснован: **-3**.

2. Ответ. Запись действия $\langle \text{cout} \ll \text{'СИМВОЛ'}; \rangle$ сокращена до $\langle \text{СИМВОЛ} \rangle$. Фигурные скобки означают итерацию, как в БНФ. Грамматика может быть и без них, с использованием только рекурсии:

$$\begin{array}{llll} S \rightarrow A \{ + \langle + \rangle A \} & A \rightarrow B \{ * \langle * \rangle B \} & B \rightarrow N F & N \rightarrow - M \mid \varepsilon \\ M \rightarrow - N \mid \varepsilon \langle - \rangle & F \rightarrow a \langle a \rangle \mid b \langle b \rangle \mid (\langle () S \rangle) \end{array}$$

Критерии: построенная грамматика описывает не все выражения или есть лишние цепочки: **-5**, метод рекурсивного спуска не применим к построенной грамматике: **-3**, за каждую неверную или отсутствующую печать в грамматике с действиями: **-2**.

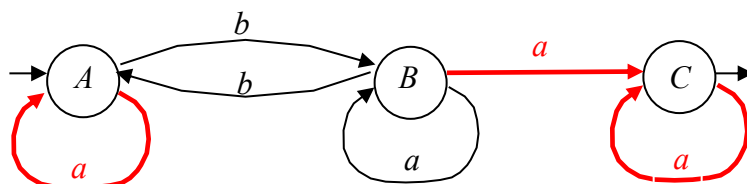
Замечание. За формально правильную схему перевода даже при наличии дефектов в исходной грамматике оценка не снижается, так как за дефекты грамматики (например, рекурсивный спуск неприменим) оценка уже снижена.

3. Ответ. Лишнее – *Make*. Система контроля версий – в самом общем понимании осуществляет отслеживание версий (ревизий) некоторого набора объектов (например, для файлов можно при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и т.п.).

Критерии: за неправильное вычеркивание: **-5**, за ошибочное или отсутствующее определение: **-5**. Определение может быть дано другими словами.

4. Ответ.

δ :



$$P_L: \begin{array}{l} C \rightarrow Ca \mid Ba \\ B \rightarrow Ba \mid Ab \\ A \rightarrow Aa \mid \varepsilon \mid Bb \end{array}$$

$$P_R: \begin{array}{l} A \rightarrow aA \mid bB \\ B \rightarrow aB \mid aC \mid bA \\ C \rightarrow aC \mid \varepsilon \end{array}$$

Элементы, добавленные в δ , P_L , P_R , на рисунке показаны утолщенными линиями красного цвета. Есть другой вариант ответа, где из B в C идет дуга, помеченная символом b – такой автомат тоже будет недетерминированным.

Критерии. Есть ошибка (лишняя дуга или нет нужной) в дополнении δ : **-4**, есть ошибка в дополнении P_L : **-3**, есть ошибка в дополнении P_R : **-3**.

5. Ответ: `do { if (++b); else; --a; } while (a+b);`

Чтобы получить еще один пример, можно добавить круглые скобки вокруг арифметического подвыражения или пустой оператор изобразить как {}, или непустой оператор заключить в блок.

Критерии: неверно восстановлен фрагмент: -5, не приведен еще один эквивалентный фрагмент: -5.

6. Ответ. Ошибок в программе нет. Будет выдано: **start 1a 12aa 35ba**

Команда компиляции: `$> g++ -std=c++14 -fno-elide-constructors v1_6.cpp`

Критерии: за каждую ошибочную строку (из четырех) -3.

7. Ответ. (a)

```
void g (double n = 0, const char * t = 0) {}
void g (const char * t) {}
```

Возможны другие решения. Вместо `'const char*'` допускается `'char*'`.

(б) Да, может. Пример:

```
struct A{ static const int zero;}; const int A::zero=0;
```

Замечание. Функция-член класса не может быть одновременно константной и статической, так как `const` относится к “скрытому параметру” `this`, а статическая функция такого параметра “не получает”.

Критерии: **(a)** за каждую ошибочную, недостающую или лишнюю функцию -4, **(б)** за неверный или не обоснованный ответ -3.

8. Ответ: еще есть параметрический полиморфизм.

Одновременно статический, и динамический полиморфизм проявиться может. Пусть есть базовый класс с двумя перегруженными виртуальными операциями '+' (например, одна с параметром типа `int`, другая – типа `char`). Пусть имя `a` описано как ссылка на базовый класс, инициализированная объектом производного класса, а имя `b` описано с типом `int`. Тогда в выражении `a+b` статически выберется операция '+' с параметром `int`, и динамически через ТВМ вызовется метод '+' для производного класса.

В качестве обоснования может быть просто дано описание классов и имен `a` и `b`, без подробных словесных пояснений.

Критерии: за не названный, или не существующий вид полиморфизма : -2, за правильный, но не обоснованный ответ на первый вопрос: -7, за неправильный ответ на первый вопрос: -8.

9. Ответ: char bad_alloc

Критерии: за каждую неверную печать -5.

10. Ответ:

```
template <class T>
T Reverse (const T& a) {
    T res;
    typename T::const_reverse_iterator it = a.rbegin();
    while (it != a.rend()) res.push_back (* it ++);
    return res;
}
```

Замечание. Метод `push_front()` для векторов не определен.

Критерии: за неконстантный итератор (при передаче параметра по ссылке): -2, за другие ошибки: -3 за каждую.

Экзамен, 2018 год. Ответы и критерии оценок.

Вариант 2 и 4.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. (а) Да, может. Например, КС-грамматика $S \rightarrow aS \mid \varepsilon$ не удовлетворяет ограничениям КЗ-грамматики: если для начального символа S есть правило $S \rightarrow \varepsilon$, то S не должен встречаться в правых частях других правил, но есть правило $S \rightarrow aS$.

(б) Нет, не может. Регулярный язык всегда можно задать детерминированным конечным автоматом, и построить по нему праволинейную грамматику, например. Она будет однозначной вследствие детерминированности автомата.

(в) Нет, не существует. Любую КС-грамматику можно преобразовать в эквивалентную неукорачивающую КС-грамматику с помощью алгоритма устранения правил с пустой правой частью.

Критерии: за каждый пункт: если неверный ответ: **-4**, если ответ верный, но не обоснован: **-3**.

2. Ответ. Запись действия $\langle cout << \text{"СИМВОЛЫ"}; \rangle$ сокращена до $\langle \text{СИМВОЛЫ} \rangle$. Фигурные скобки означают итерацию, как в БНФ. Грамматика может быть и без них, с использованием только рекурсии:

$$\begin{array}{lll} S \rightarrow A \{ \text{or} \langle \text{or} \rangle A \} & A \rightarrow B \{ \text{and} \langle \text{and} \rangle B \} & B \rightarrow NF \\ N \rightarrow \text{not } M \mid \varepsilon & M \rightarrow \text{not } N \mid \varepsilon \langle \text{not} \rangle & F \rightarrow a \langle a \rangle \mid b \langle b \rangle \mid (\langle (\rangle S \rangle) \end{array}$$

Критерии: построенная грамматика описывает не все выражения или есть лишние цепочки: **-5**, метод рекурсивного спуска не применим к построенной грамматике: **-3**, за каждую неверную или отсутствующую печать в грамматике с действиями: **-2**.

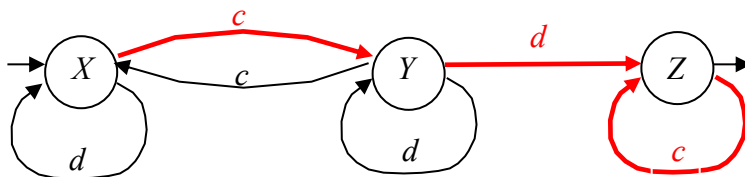
Замечание. За формально правильную схему перевода даже при наличии дефектов в исходной грамматике оценка не снижается, так как за дефекты грамматики (например, рекурсивный спуск неприменим) оценка уже снижена.

3. Ответ. Лишнее – *Valgrind. Библиотека компонентов* – это библиотека готовых откомпилированных программных модулей, предназначенных для использования в качестве составных частей программ, и которыми можно манипулировать во время разработки программ.

Критерии: за неправильное вычеркивание: **-5**, за ошибочное или отсутствующее определение: **-5**. Определение может быть дано другими словами.

4. Ответ.

δ :



$$\begin{array}{l} P_L: \quad \textcolor{red}{Z} \rightarrow \textcolor{red}{Zc} \mid \textcolor{red}{Yd} \\ \quad \quad Y \rightarrow Xc \mid \textcolor{red}{Yd} \\ \quad \quad X \rightarrow \varepsilon \mid \textcolor{red}{Xd} \mid \textcolor{red}{Yc} \end{array}$$

$$\begin{array}{l} P_R: \quad \textcolor{red}{X} \rightarrow \textcolor{red}{dX} \mid \textcolor{red}{cY} \\ \quad \quad \textcolor{red}{Y} \rightarrow \textcolor{red}{dZ} \mid \textcolor{red}{dY} \mid \textcolor{red}{cX} \\ \quad \quad Z \rightarrow \textcolor{red}{cZ} \mid \varepsilon \end{array}$$

Элементы, добавленные в δ , P_L , P_R , на рисунке показаны утолщенными линиями красного цвета. Есть другой вариант ответа, где из Y в Z идет дуга, помеченная символом c – такой автомат тоже будет недетерминированным.

Критерии. Есть ошибка (лишняя дуга или нет нужной) в дополнении δ : **-4**, есть ошибка в дополнении P_L : **-3**, есть ошибка в дополнении P_R : **-3**.

5. Ответ. `while (a+b){ if (b--); else; a++; }`

Чтобы получить еще один пример, можно добавить круглые скобки вокруг арифметического подвыражения или пустой оператор изобразить как `{}`, или непустой оператор заключить в блок.

Критерии: неверно восстановлен фрагмент: -5, не приведен еще один эквивалентный фрагмент: -5.

6. Ответ. Ошибок в программе нет. Будет выдано: `start 12cc 1c 12c5dc`

Команда компиляции: `$> g++ -std=c++14 -fno-elide-constructors v2_6.cpp`

Критерии: за каждую ошибочную строку (из четырех) -3.

7. Ответ. (a) `void h (double r = 0, const char * s = 0) {}`
`void h (const char *s, int n = 0) {}`

Возможны другие решения. Вместо `'const char*'` допускается `'char*'`.

(б) Да, может. Пример: `struct A{ static const int zero;}; const int A::zero=0;`

Замечание. Функция-член класса не может быть одновременно константной и статической, так как `const` относится к “скрытому параметру” `this`, а статическая функция такого параметра “не получает”.

Критерии: (а) за каждую ошибочную, недостающую или лишнюю функцию -4, (б) за неверный или не обоснованный ответ -3.

8. Ответ: еще есть параметрический полиморфизм.

Одновременно статический, и динамический полиморфизм проявиться может. Пусть есть описания классов:

```
class A{ public: virtual void operator[] (int) {cout<<"A[int]"; }  
          virtual void operator[] (char) {cout<<"A[char]"; } };  
class B: public A{ public:  
          virtual void operator[] (int) {cout<<"B[int]"; }  
          virtual void operator[] (char) {cout<<"B[char]"; } };
```

Пусть имя `a` описано как ссылка на базовый класс `A`, инициализированная объектом производного класса, а имя `b` описано с типом `int`. Тогда в выражении `a[b]` статически выберется операция `'[]'` с параметром `int`, и динамически через ТВМ вызовется метод `'[]'` для производного класса. Напечатается `B[int]`. Обоснование может быть и словесным, без конкретного примера класса.

Критерии: за неназванный, или несуществующий вид полиморфизма : -2, за правильный, но не обоснованный ответ на первый вопрос: -7, за неправильный ответ на первый вопрос: -8.

9. Что будет напечатано в результате работы данной программы? В курсе рассматривались стандартные исключения `bad_alloc`, `bad_cast`, `bad_typeid`.

Ответ: `int bad_alloc`

Критерии: за каждую неверную печать -5.

10. Ответ:

```
template <class T>  
T Concat (const T & a, const T & b){  
    T res;  
    typename T::const_iterator it=a.begin();  
    while (it != a.end()) res.push_back (* it ++);  
    it=b.begin();  
    while (it != b.end()) res.push_back (* it ++);  
    return res;  
}
```

Возможны другие решения, например, без циклов через `insert()`:

```
res.insert(res.end(), a.begin(), a.end());  
res.insert(res.end(), b.begin(), b.end());
```

Критерии: за неконстантный итератор (при передаче параметра по ссылке): -2, за другие ошибки: -3 за каждую.

1	2

1. (50 баллов) Дана заготовка класса *Scanner*. Метод *analyze()* данного класса осуществляет разбор по некоторой детерминированной левосторонней грамматике G_{left} с нетерминальным алфавитом $\{A, B\}$ и множеством правил $\{A \rightarrow Aa \mid Ba \mid \varepsilon, B \rightarrow Bb \mid Ab\}$. Полагаем, что задаваемая во входном потоке *cin* цепочка для анализа всегда завершается символом '#', который не входит в алфавит терминальных символов грамматики G_{left} . Если анализатор попадает в состояние ошибки *ERR*, то он прекращает работу, оставаясь в этом состоянии.

```
#include<iostream>
#include<typeinfo>
using namespace std;
class Scanner {
    class State {
        State * operator() (char c) const = ;
    };

    class A: public State { public:
        State * operator() (char c) const {
            if (c=='b') return new B;
            else if (c=='a') return new A;
            else return new ERR;
        }
    };
    class B: {

    };

    class ERR: public State {
        State * operator() (char c) const {
            return new ERR; }
        // из состояния ошибки нет переходов
        // в другие состояния
    };

    char c; // текущий символ
    State * temp, *next; // текущее и
    //следующее состояния автомата (ДС)
public:
    void analyze(){ State::st_count = 0;
        for (temp = new A, cin >> c;
            c != '#'&&
            !(dynamic_cast<ERR*>(temp));
            cin >> c) {
            next = (* temp)(c);
            delete temp;
            temp = next;
        }
        if (c != '#' ||
            typeid(*temp)!=typeid(B)
            )
            cout << "ERROR" << endl;
        else
            cout << "OK, длина пути в \
            диаграмме состояний ="
            << State::st_count << endl;
            delete temp;
        }
    };
    int Scanner::State::st_count;
```

(а) Определите начальный символ грамматики G_{left} по описанию класса *Scanner*. (Обратите внимание прежде всего на метод *analyze()*). Ответ:

(б) Определите язык, порождаемый грамматикой G_{left} (формула или словесное описание).
Ответ:

(в) Постройте диаграмму состояний для G_{left} — пусть это будет конечный автомат $\mathcal{U}$.
Ответ:

(г) Постройте эквивалентный автомат $\mathcal{U}'$ путем добавления в $\mathcal{U}$ только одной дуги так, чтобы $\mathcal{U}'$ получился бы недетерминированным. (Новые состояния не добавлять.)
Ответ:

(д) Добавьте недостающие фрагменты описаний в классы *State* и *B* (и только в них!) так, чтобы:

- метод *analyze()* считал правильными только все цепочки языка $L(G_{left})$,
- никаким способом невозможно было бы создать отдельный объект класса *State*,
- в случае успеха *analyze()* печатал бы количество вершин на пути по диаграмме состояний (ДС для G_{left}) из начальной вершины в заключительную.

2. (50 баллов) Дана заготовка класса *Parser*. Метод *analyze()* класса *Parser* осуществляет разбор по некоторой контекстно-свободной грамматике $G = \langle \{a, b\}, \{A, B, S\}, P, S \rangle$, используя принцип рекурсивного спуска. Полагаем, что задаваемая во входном потоке *cin* цепочка для анализа всегда завершается символом '#'.

```
#include<iostream>
#include<list>
using namespace std;

class Parser {
    static list<char> result;
    // результат перевода исходной цепочки
    static char c; // текущий символ(лексема)

    static void gc() {cin >> c;}
    // получить следующий символ-лексему

    class S { public:
        S() { if (c=='a') { gc(); A();
                if (c=='b') { gc(); B();
                }
                else throw c;
            }
            else throw c;
        }
        ~S() { result.push_front('S'); }
    };
    class A { public:
        A() { if (c=='a') { gc(); A(); }
        }
        ~A() { result.push_front('A'); }
    };
};

class B {
public:
    B() { if (c=='b') gc();
          else throw c;
    }
    ~B() { result.push_front('B'); }
};

public:
    void analyze() {
        try { result.clear();
              gc();
              S();
              if (c!='#') throw c;
              cout << "Результат перевода \
для входной цепочки: ";

              cout << endl;
          }
          catch( ) {
          }
    }; //end of Parser
char Parser::c;
list<char> Parser::result;
```

(а) Восстановите по анализатору множество правил P грамматики G .

Ответ:

(б) Докажите, что метод рекурсивного спуска применим для G .

Ответ:

(в) Однозначна ли грамматика G ? Обоснуйте ответ.

Ответ:

(г) Постройте эквивалентную неукорачивающую грамматику алгоритмом устранения эпсилон-правил. Ответ:

(д) Добавьте в метод *analyze()* (и только в него!) все необходимое так, чтобы в результате работы анализатора

1) для **корректных** входных цепочек на экран печатался результат перевода входной цепочки в цепочку в алфавите $\{S, A, B\}$, в которой нетерминальные символы грамматики располагаются в порядке их замены при построении **правостороннего** вывода,

2) в случае **ошибки** печаталось сообщение:

Ошибочный символ: <здесь печатать символ, ставший причиной ошибки>

1	2

1. (50 баллов) Дана заготовка класса *Parser*. Метод *analyze()* класса *Parser* осуществляет разбор по некоторой контекстно-свободной грамматике $G = \langle \{b, d\}, \{B, D, S\}, P, S \rangle$, используя принцип рекурсивного спуска. Полагаем, что задаваемая во входном потоке *cin* цепочка для анализа всегда завершается символом '#'.

```
#include<iostream>
#include<list>
using namespace std;
class Parser {
    static list<char> trans;
    // результат перевода исходной цепочки
    static char c; // текущий символ(лексема)

    static void gc() {cin >> c;}
    // получить следующий символ-лексема

    struct S {
        S() { if (c=='b') { gc(); B(); D();
                if (c=='d') { gc();
                    }
                else throw c;
            }
            else if (c=='d') gc();
            else throw c;
        }
        ~S() { trans.push_front('S'); }
    };
    struct D {
        D() { if (c=='d') gc();
                else throw c;
            }
        ~D() { trans.push_front('D'); }
    };

    struct B {
        B() { if (c=='b') { gc(); B(); }
            }
        ~B() { trans.push_front('B'); }
    };

public:
    void analyze() {
        try { trans.clear();
            gc();
            S();
            if (c!='#') throw c;
            cout << "Результат перевода \
для входной цепочки: ";

            cout << endl;
        }
        catch( ) {
        }
    }
}; //end of Parser

char Parser::c;
list<char> Parser::trans;
```

(а) Восстановите по анализатору множество правил *P* грамматики *G*.

Ответ:

(б) Докажите, что метод рекурсивного спуска применим для *G*.

Ответ:

(в) Однозначна ли грамматика *G*? Обоснуйте ответ.

Ответ:

(г) Постройте эквивалентную неукорачивающую грамматику алгоритмом устранения эpsilon-правил. Ответ:

(д) Добавьте в метод *analyze()* (и только в него!) все необходимое так, чтобы в результате работы анализатора

- 1) для **корректных** входных цепочек на экран печатался результат перевода входной цепочки в цепочку в алфавите $\{S, B, D\}$, в которой нетерминальные символы грамматики располагаются в порядке их замены при построении **правостороннего** вывода,
- 2) в случае **ошибки** печаталось сообщение:

Ошибочный символ: <здесь печатать символ, ставший причиной ошибки>

2. (50 баллов) Дана заготовка класса *Scanner*. Метод *analyze()* данного класса осуществляет разбор по некоторой детерминированной левосторонней грамматике G_{left} с нетерминальным алфавитом $\{C, D\}$ и множеством правил $\{D \rightarrow Da \mid Ca; C \rightarrow Cb \mid Db \mid \varepsilon\}$. Полагаем, что задаваемая во входном потоке *cin* цепочка для анализа всегда завершается символом '#', который не входит в алфавит терминальных символов грамматики G_{left} . Если анализатор попадает в состояние ошибки *ERR*, то он прекращает работу, оставаясь в этом состоянии.

```
#include<iostream>
#include<typeinfo>
using namespace std;

class Scanner {
    class State {
        State * operator() (char c) const = ;
    };

    class C: public State { public:
        State * operator() (char c) const {
            if (c=='b') return new C;
            else if (c=='a') return new D;
            else return new ERR;
        }
    };

    class D: {

    };

    class ERR: public State {
        State * operator() (char c) const {
            return new ERR;
        }
        // из состояния ошибки нет переходов
        // в другие состояния
    };

    char c; // текущий символ
    State * temp, *next; // текущее и
    //следующее состояния автомата (ДС)
public:
    void analyze(){ State::st_number = 0;
        for (temp = new C, cin >> c;
            c != '#'&&
                !(dynamic_cast<ERR*>(temp));
            cin >> c) {
            next = (* temp) (c);
            delete temp;
            temp = next;
        }
        if (c != '#' ||
            typeid(*temp) != typeid(D)
        )
            cout << "ERROR" << endl;
        else
            cout<< "OK, количество \
состояний в пройденном пути ="
                << State::st_number <<endl;
            delete temp;
        }
    };

    int Scanner::State::st_number;
```

(а) Определите начальный символ грамматики G_{left} по описанию класса *Scanner*. (Обратите внимание прежде всего на метод *analyze()*). Ответ:

(б) Определите язык, порождаемый грамматикой G_{left} (формула или словесное описание).
 Ответ:

(в) Постройте диаграмму состояний для G_{left} – пусть это будет конечный автомат $\mathcal{U}$.
 Ответ:

(г) Постройте эквивалентный автомат $\mathcal{U}'$ путем добавления в $\mathcal{U}$ только одной дуги так, чтобы $\mathcal{U}'$ получился бы недетерминированным. (Новые состояния не добавлять).
 Ответ:

(д) Добавьте недостающие фрагменты описаний в классы *State*, *D* (и только в них!) так, чтобы:

- метод *analyze()* считал правильными только все цепочки языка $L(G_{left})$,
- никаким способом невозможно было бы создать отдельный объект класса *State*,
- в случае успеха *analyze()* печатал бы количество состояний на пути по диаграмме состояний (ДС для G_{left}) из начального состояния в заключительное.

Коллоквиум, 2019 год. Ответы и критерии оценок.

Вариант 1.

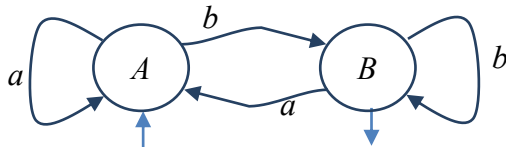
Максимум за обе задачи в сумме 100 баллов

1. Ответ.

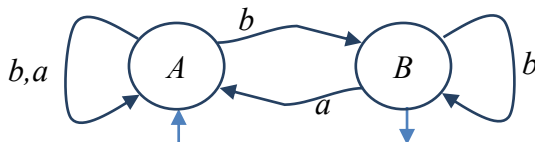
(а) B – начальный символ в G_{left} .

(б) $L(G_{left}) = \{\omega b \mid \omega \in \{a, b\}^*\}$. Непустая цепочка в алфавите $\{a, b\}$, заканчивающаяся на b .

(в) $\mathcal{Y}$:



(г) $\mathcal{Y}'$:



Ответом может быть добавление через запятую другой буквы на любую из четырех дуг.

(д)

```
class State { public: static int st_count;
              State() { st_count++; }
              virtual State * operator () (char c) const = 0;
              virtual ~State() {}
};
class B: public State { public:
    State * operator () (char c) const {
        if (c == 'b') return new B;
        else if (c == 'a') return new A;
        else return new ERR;
    }
};
```

Критерии: Пункты (а, б, в): по -5 каждый нерешенный или неправильный пункт. За ошибку в пункте (г): -10. Пункт (д): по -5 за каждую ошибку – нет слова 'virtual', нет '=0', 'public' и т. п., неправильно запрограммирован переход в новое состояние и т.д., но в целом снижение не более 25 баллов за этот пункт. За отсутствие деструктора оценка не снижается.

2. Ответ.

(а) $P = \{ S \rightarrow aAbB; A \rightarrow aA \mid \varepsilon; B \rightarrow b \}$

(б) $first(aA) \cap first(\varepsilon) = \emptyset, first(A) \cap follow(A) = \emptyset \Rightarrow$ выполняется критерий применимости.

(в) Для неоднозначных грамматик нарушается хотя бы один пункт критерия применимости рекурсивного спуска. Из (б) следует, что грамматика G однозначна.

(г) $S \rightarrow aAbB \mid abB; A \rightarrow aA \mid a; B \rightarrow b$

```
(д) void analyze() {
    try { result.clear();
        gc();
        S();
        if (c != '#') throw c;
        cout << "Результат перевода для входной цепочки: ";
        for (list<char>::iterator p = result.begin();
            p != result.end(); p++) cout << *p;
        cout << endl;
    }
    catch(char c) { cout << "Ошибочный символ " << c << endl;
    }
```

Критерии: Пункты (а, б, в): по -5 каждый нерешенный или неправильный пункт. За ошибку в пункте (г): -10. Пункт (д): по -7 за каждую ошибку – неправильно описан итератор (auto допускается), печать в обратном порядке, нет печати требуемого сообщения и символа и т.д., но в целом снижение за этот пункт не более, чем на 25 баллов.

Коллоквиум, 2019 год. Ответы и критерии оценок.

Вариант 2.

Максимум за обе задачи в сумме 100 баллов

1. Ответ.

(а) $P = \{ S \rightarrow bBDd \mid d; B \rightarrow bB \mid \varepsilon; D \rightarrow d \}$

(б) $\text{first}(bBDd) \cap \text{first}(d) = \emptyset, \text{first}(bB) \cap \text{first}(\varepsilon) = \emptyset, \text{first}(B) \cap \text{follow}(B) = \emptyset \Rightarrow$ выполняется критерий применимости.

(в) Для неоднозначных грамматик нарушается хотя бы один пункт критерия применимости рекурсивного спуска. Из (б) следует, что грамматика G однозначна.

(г) $S \rightarrow bBDd \mid bDd \mid d; B \rightarrow bB \mid b; D \rightarrow d$

```
(д) void analyze() {
    try { trans.clear();
        gc();
        S();
        if (c!='#') throw c;
        cout << "Результат перевода для входной цепочки: ";
        for (list<char>::iterator p = trans.begin();
            p != trans.end(); p++) cout << *p;
        cout<<endl;
    }
    catch(char c){cout<< "Ошибочный символ " << c<<endl;
    }
```

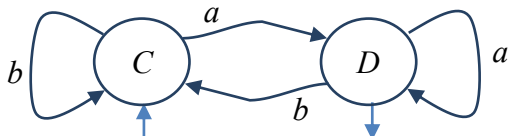
Критерии: Пункты (а, б, в): по -5 каждый нерешенный или неправильный пункт. За ошибку в пункте (г): -10. Пункт (д): по -7 за каждую ошибку – неправильно описан итератор (auto допускается), печать в обратном порядке, нет печати требуемого сообщения и символа и т.д., но в целом снижение за этот пункт не более, чем на 25 баллов.

2. Ответ.

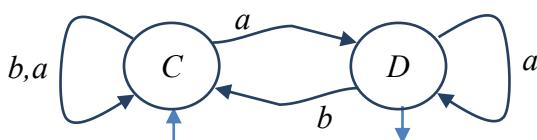
(а) D – начальный символ в G_{left} .

(б) $L(G_{\text{left}}) = \{\omega a \mid \omega \in \{a, b\}^*\}$. Непустая цепочка в алфавите $\{a, b\}$, заканчивающаяся на a .

(в) U :



(г) U' :



Ответом может быть добавление через запятую другой буквы на любую из четырех дуг.

(д)

```
class State { public: static int st_number;
    State(){st_number++;}
    virtual State * operator () (char c) const = 0;
    virtual ~State(){};
};
class D :public State { public:
    State * operator () (char c) const{
        if (c == 'a') return new D;
        else if (c == 'b') return new C;
        else return new ERR;
    }
};
```

Критерии: Пункты (а, б, в): по -5 каждый нерешенный или неправильный пункт. За ошибку в пункте (г): -10. Пункт (д): по -5 за каждую ошибку – нет слова 'virtual', нет '=0', 'public' и т. п., неправильно запрограммирован переход в новое состояние и т.д., но в целом снижение не более 25 баллов за этот пункт. За отсутствие деструктора оценка не снижается.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Добавить в приведенную программу необходимые конструкции C++ (не заменяя и не удаляя имеющиеся) так, чтобы при ее компиляции не было ошибок, а на печать выдалось:

```
Constr_B
Constr_A
Destr_A
Destr_B
```

```
struct B {
    virtual void f() = 0;
    B() { cout << "Constr_B\n"; }
    ~B() { cout << "Destr_B\n"; }
};
int main () {
    NN::A a;
    B *pb = &a;
    return 0; }
```

2. Добавить только в класс A необходимые элементы, не вводя никаких дополнительных членов класса A, ничего не удаляя и не заменяя, так, чтобы в получившейся программе не было ошибок, а на печать выдалось:

```
class A {
    int n;
public:
    int f(int n = 0) { return n; }
    A(int a) { n = a; }
};
```

```
7 3.
int main () {
    A a, b (3);
    cout<<a.f()<<" "<<b.f(1)<<endl;
    return 0;
}
```

3. Найти ошибки в приведенной программе и объяснить их. Исправить найденные ошибки.

Что напечатает получившаяся программа?

```
template <class T>
T::value_type f (const T& t) {
    if (t.size())
        return t[t.size()-1];
    else
        return (typename T::value_type)0;
}
int main () {
    vector <int> v;
    list <int> l(2,3);
    cout << f<vector<int>>(v) << f<list<int>>(l) << endl;
    return 0; }
```

4. Удалить некоторые элементы в теле функции main(), ничего не добавляя и не заменяя, так, чтобы в получившейся программе не было ошибок, а на экран выдалось:

A() A() B() ~B() ~A() ~A() ~A() ~B() ~A()

```
struct A {
    A(){ cout << "A() "; }
    ~A(){ cout << "~A() "; }
};
struct B:A {
    B(){ cout << "B() "; }
    ~B(){ cout << "~B() "; }
};
```

```
int main () {
    try { int n = -1; A a; B b;
        if(n > 0) throw a;
        else throw b;
    }
    catch (B & x) {return 1;}
    catch (A & y) {return 2;}
}
```

5. Внести необходимые изменения в программу, не вводя дополнительных членов класса C, не меняя функцию main() и ничего не удаляя, так, чтобы в ней не было ошибок, а на экран выдалось:

```
class C {
    int i;
public:
    explicit C (int k = 0) { i = k; }
    void i_out() { cout << i << " "; }
};
```

```
int main () {
    C a(3), b(2);
    (a * b).i_out();
    (a * b * a).i_out();
    return 0;
}
```

6. Даны грамматики G_1 и G_2 .

$G_1: S \rightarrow ASB \mid \varepsilon$
 $AB \rightarrow BA$
 $A \rightarrow a$
 $B \rightarrow b$

(а) Эквивалентны ли G_1 и G_2 ? Ответ обосновать. *Ответ:* _____

(б) Определить тип грамматики G_1 : _____

(в) Определить тип грамматики G_2 : _____

(г) Определить тип языка $L(G_1)$: _____

(д) Определить тип языка $L(G_2)$: _____

7. Дана грамматика многочленов от одной переменной x с односимвольными коэффициентами и степенями, к которой применим метод рекурсивного спуска. Фигурные скобки в грамматике означают итерацию. Добавьте в грамматику действия только вида $\langle \text{cout} \ll \text{"символы"}; \rangle$, реализующие формальный перевод $\tau = \{ (p, r) \mid p - \text{многочлен}, r - \text{первообразная многочлена } p \}$.

Пример: $2x^2 - 3x^0 + 5x^1 \Rightarrow 2x^3/3 - 3x^1 + 5x^2/2 + C$

$S \rightarrow A \{ Z A \}$ *Ответ:*

$Z \rightarrow + \mid -$

$A \rightarrow Kx^N$

$N \rightarrow 0 \mid 1 \mid 2$

$K \rightarrow 0 \mid 1 \mid 2$

8. Дана левوليнейная грамматика G_{left} :

$S \rightarrow Sa \mid Aa$
 $A \rightarrow Aa \mid b \mid a$

(а) Применим ли метод рекурсивного спуска к данной грамматике? Обоснуйте ответ.

Ответ:

(б) Постройте эквивалентную детерминированную праволинейную грамматику G_{right}

Ответ:

9. Дана заготовка КС-грамматики $G = \langle \{A, B, S\}, \{a, b\}, P, S \rangle$, где $P = \{ A \rightarrow AA \mid a; B \rightarrow b; X \rightarrow a \}$.

Найти такие $X \in \{A, B, S\}$ и $\alpha \in \{A, B, S, a, b\}^*$ для G , чтобы одновременно выполнялись условия:

(а) G не является приведённой грамматикой;

(б) $L(G) \neq \emptyset$;

(в) грамматика G однозначна.

Ответ:

10. Что такое оптимизация программы? Ниже дана постфиксная запись фрагмента программы на языке Си. Оптимизируйте длину кода (постройте более короткую эквивалентную постфиксную запись), используя только те виды операций, которые есть в исходном варианте.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПОЛИЗ	a	b	>	12	!F	&a	c	=	;	16	!	&a	d	=	;		

Ответ:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПОЛИЗ																	

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std;`

1	2	3	4	5	6	7	8	9	10

1. Добавить в приведенную программу необходимые конструкции C++ (не заменяя и не удаляя имеющиеся) так, чтобы при ее компиляции не было ошибок, а на печать выдалось:

```
Constr_B
Constr_D
Destr_D
Destr_B
```

```
namespace SS { struct B {
    virtual void g() = 0;
    B() { cout << "Constr_B\n"; }
    ~B() { cout << "Destr_B\n"; }
};
}
int main () { D d;
    B *pb = &d;
    return 0; }
```

2. Добавить только в класс C необходимые элементы, не вводя никаких дополнительных членов класса C, ничего не удаляя и не заменяя, так, чтобы в получившейся программе не было ошибок, а на печать выдалось: 5 3.

```
class C {
    int i;
public:
    C(int a) { i = a; }
    int f(int i = 0) { return i; }
};

int main () {
    C c1(5), c2;
    cout << c1.f() << " " << c2.f() << endl;
    return 0;
}
```

3. Найти ошибки в приведенной программе и объяснить их. Исправить найденные ошибки.

Что напечатает получившаяся программа?

```
template <class T>
typename T::value_type f (T& t) {
    if (t.size()) { t.push_front(t.back());
        return t.front(); }
    else
        return ( T::value_type) 0;
}

int main () {
    list <int> l(2, 5);
    vector <int> v;
    cout << f<vector<int>>(l) << f<list<int>>(v) << endl;
    return 0; }
```

4. Добавить некоторые элементы в тело функции main(), ничего не удаляя и не заменяя, так, чтобы в получившейся программе не было ошибок, а на экран выдалось:

```
A() B() A(const A&) ~B() ~A() ~A()
struct A {
    A() { cout << "A() "; }
    A(const A& a)
    { cout << "A(const A&) "; }
    ~A() { cout << "~A() "; }
};
struct B:A {
    B() { cout << "B() "; }
    ~B() { cout << "~B() "; }
};

int main () {
    try { B b; A *pa = &b;
        throw *pa;
    }
    catch (B x) { return 1; }
}
```

5. Внести необходимые изменения в программу, не вводя дополнительных членов класса C, не меняя функцию main() и ничего не удаляя, так, чтобы в ней не было ошибок, а на экран выдалось: 2.5 0.5.

```
class P {
    double d;
public:
    explicit P (double k = 0) { d = k; }

int main ()
    P a(5), b(2);
    (a / b).out_d();
    (a / b / a).out_d();
}
```

```

        void out_d() { cout << d << " "; }
    };
    return 0;
}

```

6. Даны грамматики G_1 и G_2 .

$G_1: S \rightarrow PRS \mid \varepsilon$
 $PR \rightarrow RP$
 $RP \rightarrow PR$
 $R \rightarrow 0$
 $P \rightarrow 1$

$G_2: S \rightarrow 1S0 \mid 0S1 \mid SS \mid \varepsilon$

(а) Эквивалентны ли G_1 и G_2 ? Ответ обосновать. *Ответ:* _____

(б) Определить тип грамматики G_1 : _____

(в) Определить тип грамматики G_2 : _____

(г) Определить тип языка $L(G_1)$: _____

(д) Определить тип языка $L(G_2)$: _____

7. Дана грамматика многочленов от одной переменной x с односимвольными коэффициентами и степенями в троичной системе счисления, к которой применим метод рекурсивного спуска. Фигурные скобки в грамматике означают итерацию. Добавьте в грамматику действия только вида $\langle \text{cout} << \text{"символы"}; \rangle$, реализующие формальный перевод $\tau = \{ (t, w) \mid t - \text{многочлен}, w - \text{производная многочлена } t \}$.

Пример: $1*x^2 - 1 + 2*x^1 \Rightarrow 1*x^1*2 - 1*0 + 2*x^0$

$S \rightarrow A \{ Z A \}$ *Ответ:*

$Z \rightarrow + \mid -$

$A \rightarrow KL$

$L \rightarrow *x^N \mid \varepsilon$

$N \rightarrow 1 \mid 2$

$K \rightarrow 0 \mid 1 \mid 2$

8. Дана левострогая грамматика G_{left} :

$S \rightarrow Sa \mid Ba \mid a$

$B \rightarrow Ba \mid b$

(а) Применим ли метод рекурсивного спуска к данной грамматике? Обоснуйте ответ.

Ответ:

(б) Постройте эквивалентную детерминированную правострогую грамматику G_{right}

Ответ:

9. Дана заготовка КС-грамматики $G = \langle \{A, B, S\}, \{a, b\}, P, S \rangle$, где $P = \{ A \rightarrow AA \mid \varepsilon, B \rightarrow b; X \rightarrow \alpha \}$. Найти такие $X \in \{A, B, S\}$ и $\alpha \in \{A, B, S, a, b\}^*$ для G , чтобы одновременно выполнялись условия:

(а) G не является приведенной грамматикой;

(б) $L(G) = \{\varepsilon\}$;

(в) грамматика G неоднозначна.

Ответ:

10. Что такое машинно-независимые преобразования при оптимизации программы? Ниже дана постфиксная запись фрагмента программы на языке Си. Оптимизируйте длину кода (постройте более короткую эквивалентную постфиксную запись), используя только те виды операций, которые есть в исходном варианте.

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	a	b	>	16	!F	&c	d	=	;	&a	b	=	;	20	!	&c	d	=	;

Ответ:

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Экзамен, 2019 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Следует добавить: `namespace NN { struct A:B {
void f() {}
A(){ cout << "Consrt_A\n"; }
~A(){ cout << "Destr_A\n"; }
};`

Критерии: (а) Нет `namespace`: -5, (б) нет метода `f()` в классе `A`: -5, (в) другие ошибки: по -2 за каждую.

2. Следует добавить: `int f(int n = 0) { return this -> n; }
A(int a = 7) { n = a; }`

Критерии: (а) Нет `this`: -5, (б) Нет конструктора умолчания: -5, (в) другие ошибки: по -2 за каждую.

3. Ошибки: `template <class T>
typename T::value_type f(const T& t){
if(t.size())
return t[t.size()-1]; // ошибка для T=list<int>
можно заменить на if(t.size()) return t.back()`

Будет выдано: 03.

Критерии: Нет `typename`: -5, неверно исправили ситуацию с `[]`: -5; другие ошибки: -2 каждая.

4. Следует удалить первый перехватчик `catch` (B & b) и ссылку во втором перехватчике `catch (A a)`.

Критерии: За каждое неверное или невыполненное удаление по -5.

5. Следует описать и объявить в классе `C` дружественную функцию (или, если без такой функции, добавить еще один спецификатор открытой зоны `public` перед описанием `int i`):

`... friend C operator*(C, C); ... C operator*(C a, C b){ C r(a.i * b.i); return r; }`

Критерии: Нет дружественной функции или спецификатора `public`: -5, нет/неверная функция `operator*()`: -5, другие ошибки: по -2 за каждую.

6. (а) Да, обе грамматики порождают язык цепочек в алфавите $\{a, b\}$, в которых символов a и b поровну, (б) тип 0, то есть грамматика без ограничений, (в) тип 2, контекстно-свободная грамматика, (г) тип 2, контекстно-свободный язык, (д) тип 2, контекстно-свободный язык.

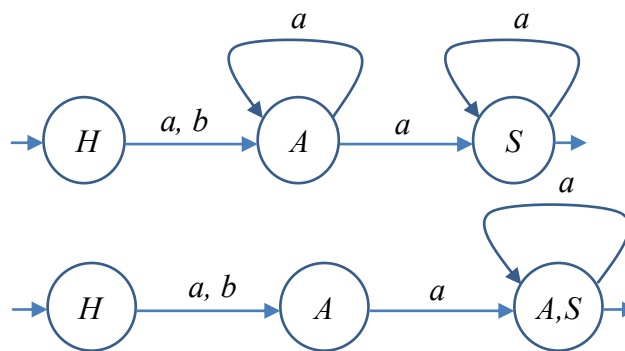
Критерии: (а) за неверный ответ -4, какое-то обоснование обязательно, остальные пункты по -2 за каждый неверный ответ.

7. $S \rightarrow A \{ Z A \} \langle \text{cout} \ll "+C "; \rangle$
 $Z \rightarrow + \langle \text{cout} \ll "+"; \rangle \mid - \langle \text{cout} \ll "-"; \rangle$
 $A \rightarrow K x^{\wedge} \langle \text{cout} \ll "x^{\wedge}"; \rangle N$
 $N \rightarrow 0 \langle \text{cout} \ll "1"; \rangle \mid 1 \langle \text{cout} \ll "2/2"; \rangle \mid 2 \langle \text{cout} \ll "3/3"; \rangle$
 $K \rightarrow 0 \langle \text{cout} \ll "0"; \rangle \mid 1 \langle \text{cout} \ll "1"; \rangle \mid 2 \langle \text{cout} \ll "2"; \rangle$

Критерии: за каждую неверную или отсутствующую печать в грамматике с действиями по -2. За альтернативный функционально эквивалентный ответ оценку не снижать.

8. (а) **Нет**, так как $first(Sa) \cap first(Aa) \supset \{a\} \neq \emptyset \Rightarrow$ нарушается критерий применимости;

(б)
$$\begin{aligned} S &\rightarrow Sa \mid Aa \\ A &\rightarrow Aa \mid b \mid a \end{aligned} \Rightarrow$$



$\Rightarrow \{ \text{детерминизация} \} \Rightarrow$

$$\begin{aligned} H &\rightarrow bA \mid aA \\ \Rightarrow A &\rightarrow aX, \text{ где } X = \{A, S\} \\ X &\rightarrow \varepsilon \mid aX \end{aligned}$$

Критерии: за каждый неправильный или отсутствующий пункт: **-5**, возможна другая эквивалентная грамматика в качестве ответа в пункте (б). Промежуточные преобразования необязательны. Допускается введение нового заключительного состояния и добавление перехода в него по маркеру конца цепочки $\perp$.

9. Подойдет любое правило вида $S \rightarrow \alpha$, где α – любая цепочка (в том числе пустая) в алфавите $\{a, b, B\}$.

Критерии: за неправильный или отсутствующий ответ **-10**.

10. Оптимизация программы – это изменение транслируемой программы (в основном переупорядочивание и замена операций) с целью получения более эффективного объектного кода.

Исходный ПОЛИЗ соответствует фрагменту на Си: **if** ($a > b$) $a = c$; **else** $a = d$;

Он эквивалентен фрагменту $a = a > b ? c : d$; , которому соответствует более короткая постфиксная запись:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПО- ЛИЗ	&a	a	b	>	10	!F	c	11	!	d	=	;					

Критерии: неверно приведено понятие оптимизации программы: **-5**, не приведена эквивалентная более короткая постфиксная запись: **-5**.

Экзамен, 2019 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Следует добавить: `using namespace SS;` и

```
struct D:B { // или, если using ниже D:SS::B
    void g() {}
    D() { cout << "Consrt_D\n"; }
    ~D() { cout << "Destr_D\n"; }
};
```

Критерии: Нет `using namespace SS`, -5, нет метода `g()` в классе `A`: -5, другие ошибки: по -2 за каждую.

2. Следует добавить: `...int f(int i = 0) { return this -> i; }`
`C(int a = 3) { i = a; } ...`

Критерии: Нет `this`: -5, нет конструктора умолчания: -5, другие ошибки: по -2 за каждую.

3. Ошибки: `... template <class T> ...`

```
t.push_front(t.back()); // заменить на t.insert(t.begin(), t.back())
return (typename T::value_type)0; ...
```

Будет выдано: 50.

Критерии: Нет `typename`: -5, неверно исправлено обращение к `push_front`: -5; другие ошибки: -2 каждая.

4. Следует добавить второй перехватчик `catch (A & a)`.

Критерии: Нет второго перехватчика: -10, перехватываются только ссылки: -5.

5. Следует описать и объявить в классе `P` дружественную функцию (или, если без такой функции, добавить еще один спецификатор открытой зоны `public` перед описанием `double d`):

```
... friend P operator/(P, P); ... P operator/(P a, P b) { P r(a.d / b.d); return r; }
```

Критерии: Нет дружественной функции или спецификатора `public`: -5, нет/неверная функция `operator/()`: -5, другие ошибки: по -2 за каждую.

6. (а) Да, обе грамматики порождают язык цепочек в алфавите $\{0, 1\}$, в которых символов 0 и 1 поровну, (б) тип 0, то есть грамматика без ограничений, (в) тип 2, контекстно-свободная грамматика, (г) тип 2, контекстно-свободный язык, (д) тип 2, контекстно-свободный язык.

Критерии: (а) за неверный ответ -4, какое-то обоснование обязательно, остальные пункты по -2 за каждый неверный ответ.

7. $S \rightarrow A \{ Z A \}$

$Z \rightarrow + \langle \text{cout} << "+" ; \rangle \mid - \langle \text{cout} << "-" ; \rangle$

$A \rightarrow K L$

$L \rightarrow *x^{\wedge} \langle \text{cout} << "*"x^{\wedge} " ; \rangle N \mid \varepsilon \langle \text{cout} << "*"0" ; \rangle$

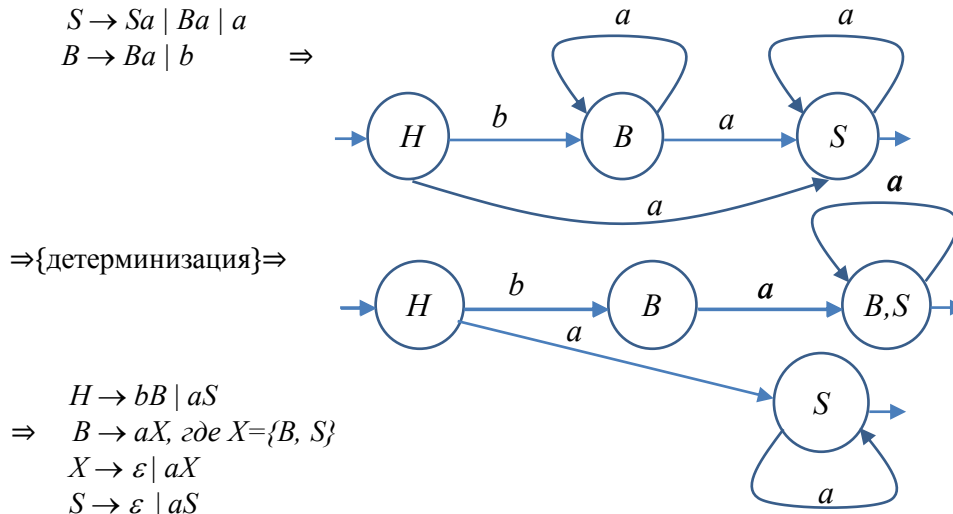
$N \rightarrow 1 \langle \text{cout} << "0" ; \rangle \mid 2 \langle \text{cout} << "1*2" ; \rangle$

$K \rightarrow 0 \langle \text{cout} << "0" ; \rangle \mid 1 \langle \text{cout} << "1" ; \rangle \mid 2 \langle \text{cout} << "2" ; \rangle$

Критерии: за каждую неверную или отсутствующую печать в грамматике с действиями по -2. За альтернативный функционально эквивалентный ответ оценку не снижать.

8. (а) **Нем**, так как $first(Sa) \cap first(a) \supset \{a\} \neq \emptyset \Rightarrow$ нарушается критерий применимости;

(б)
$$\begin{aligned} S &\rightarrow Sa \mid Ba \mid a \\ B &\rightarrow Ba \mid b \end{aligned} \Rightarrow$$



Критерии: за каждый неправильный или отсутствующий пункт: **-5**, возможна другая эквивалентная грамматика в качестве ответа в пункте (б). Промежуточные преобразования необязательны. Допускается введение нового заключительного состояния и добавление перехода в него по маркеру конца цепочки $\perp$.

9. Подойдет любое правило вида $S \rightarrow \alpha$, где α – любая непустая цепочка в алфавите $\{A\}$. Правило $S \rightarrow \varepsilon$ не является правильным ответом, так как не выполняется пункт (в).

Критерии: за неправильный или отсутствующий ответ **-10**, **-5** за каждое лишнее правило (альтернативу).

10. Машинно-независимая оптимизация программы – машинно-независимые преобразования исходной программы производятся в основном над ее внутренним представлением и основаны на известных математических и логических преобразованиях. Исходный ПОЛИЗ соответствует фрагменту на Си: **if** ($a > b$) { $c=d$; $a=b$; } **else** $c=d$; Этот фрагмент эквивалентен фрагменту $c = d$; **if** ($a > b$) $a = b$; , которому соответствует более короткая постфиксная запись:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
&c	d	=	;	a	b	>	14	!F	&a	b	=	;				

Критерии: неверно приведено понятие машинно-независимой оптимизации программы: **-5**, не приведена эквивалентная более короткая постфиксная запись: **-5**.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1

Проанализировать следующие грамматики:

1. $\begin{cases} S \rightarrow aSAc \mid \varepsilon \\ A \rightarrow bA \mid \varepsilon \end{cases}$
2. $\begin{cases} S \rightarrow aSBc \mid \varepsilon \\ B \rightarrow aB \mid b \end{cases}$
3. $\begin{cases} S \rightarrow Ac \mid Sb \\ A \rightarrow Ac \mid a \end{cases}$
4. $\begin{cases} S \rightarrow Aa \mid Sb \\ A \rightarrow Ac \mid b \end{cases}$

Никак **не преобразовывая** грамматики, выбрать одну из них для построения по грамматике корректного анализатора **методом рекурсивного спуска**, а другую – для анализатора по **конечному автомату**.

Выбор **ОБОСНОВАТЬ** (по 10 баллов за каждый правильный и правильно обоснованный ответ).

Какие **языки** порождают выбранные грамматики? (по 5 баллов за каждый правильный ответ).

Каков **тип** этих языков по Хомскому? (по 5 баллов за каждый правильный ответ).

Описать два класса **LexAn** и **SyntAn**, каждый из которых – **производный** от заданного класса **PPP** (по 25 баллов за каждый правильно в соответствии условию задачи описанный класс):

```
struct PPP {
    static int x;
    PPP (int t) { x += t; }
    virtual void gc ( ) = 0; // получает очередной символ анализируемой цепочки
    virtual const char*operator+(const char*str)=0; // выполняет анализ
    // цепочки str по выбранной грамматике, возвращает строку-сообщение
    // о принадлежности (или нет) цепочки языку, порождаемому грамматикой.
    virtual ~PPP( ) { cout << x << endl; }
};
int PPP::x = 0;
```

а также **одну** функцию `run_an(const char*str)`, запускающую анализатор по входной цепочке, (10 баллов за правильно описанную функцию) так, что в следующей функции `main()` не будет ошибок:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string str;
    cin >> str; // ввод цепочки для анализа по конечному автомату
    const char*s = str.c_str(); // возвращает указатель на строку(цепочку) в смысле языка Си

    run_an <LexAn> (s);

    cin >> str; // ввод цепочки для анализа методом рекурсивного спуска
    s = str.c_str();

    run_an <SyntAn> (s);

    return 0;
}
```

При выполнении программы на экран должно выдаваться:

```
Yes!!! // или No! – в случае непринадлежности цепочки языку
Magic number – 3
Success!!! // или Bad string! – в случае непринадлежности цепочки языку
Magic number – 8
```

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1

Проанализировать следующие грамматики:

1.
$$\begin{cases} S \rightarrow 0A \mid IS \\ A \rightarrow 1A \mid 1 \end{cases}$$
2.
$$\begin{cases} S \rightarrow aSBc \mid \varepsilon \\ B \rightarrow bB \mid c \end{cases}$$
3.
$$\begin{cases} S \rightarrow A0 \mid SI \\ A \rightarrow AI \mid 0 \end{cases}$$
4.
$$\begin{cases} S \rightarrow cASa \mid \varepsilon \\ A \rightarrow aA \mid \varepsilon \end{cases}$$

Никак **не преобразовывая** грамматики, выбрать одну из них для построения по грамматике корректного анализатора **методом рекурсивного спуска**, а другую – для анализатора по **конечному автомату**.

Выбор **ОБОСНОВАТЬ** (по 10 баллов за каждый правильный и правильно обоснованный ответ).

Какие **языки** порождают выбранные грамматики? (по 5 баллов за каждый правильный ответ).

Каков **тип** этих языков по Хомскому? (по 5 баллов за каждый правильный ответ).

Описать **два** класса **LexAn** и **SyntAn**, каждый из которых – **производный** от заданного класса **Base**: (по 25 баллов за каждый правильно в соответствии условию задачи описанный класс):

```
struct Base {
    static int n;
    Base (int t) { n *= t; }
    virtual void gc ( ) = 0;    // получает очередной символ анализируемой цепочки
    virtual const char*operator*(const char*str)=0; // выполняет анализ
    // цепочки str по выбранной грамматике, возвращает строку-сообщение
    // о принадлежности (или нет) цепочки языку, порождаемому грамматикой.
    virtual ~Base( ) { cout << n << endl; }
};
int Base::n = 1;
```

а также **одну** функцию `analyze (const char*str)`, запускающую анализатор по входной цепочке, (10 баллов за правильно описанную функцию) так, что в следующей функции `main ( )` не будет ошибок:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string str;
    cin >> str;                                // ввод цепочки для анализа по конечному автомату
    const char*s = str.c_str(); // возвращает указатель на строку(цепочку) в смысле языка Си

    analyze <LexAn> (s);

    cin >> str;                                // ввод цепочки для анализа методом рекурсивного спуска
    s = str.c_str();

    analyze <SyntAn> (s);

    return 0;
}
```

При выполнении программы на экран должно выдаваться:

```
Yes!!!    // или  No! – в случае непринадлежности цепочки языку
Lucky number – 7
Success!!! // или  Bad string! – в случае непринадлежности цепочки языку
Lucky number – 777
```

Экзамен, 2021 год. Ответы и критерии оценок.

Вариант 1.

Исходный балл – 100

1. Для анализа по КА выбирается регулярная грамматика, для которой автомат детерминирован – это № 4 (другие или не регулярные, или анализ по ним недетерминированный).

Критерий: сделан неверный выбор (нет обоснования) **-10.**

2. Для анализа РС-методом выбирается КС-грамматика, к которой метод применим, это № 1 (к другим метод неприменим: левая рекурсия или плохая ε-альтернатива)

Критерий: сделан неверный выбор (нет обоснования) **-10.**

3. Язык грамматики № 1: $L(1) = \{ a^n (b^{m_n} c)^n \mid n \geq 0, m_n \geq 0 \}$ // ошибка: **-5**, тип = 2, иначе **-5.**

4. Язык грамматики № 4: $L(4) = \{ b c^n a b^m \mid n, m \geq 0 \}$ // ошибка: **-5**, тип = 3, иначе **-5.**

```
class LexAn : public PPP {
    const char * p; char c;
    void gc() { c = *p++; } // нет функции - -10, неверная - -5
public: LexAn():PPP(3){ } // нет конструктора или списка инициализации - -10
    const char * operator+ (const char * str) { // неверный алгоритм - -15
        try { p = str; // за каждую мелкую ошибку - -2
            enum state {H, A, S} CS;
            CS = H;
            do { gc();
                switch(CS){
                    case H: if (c == 'b') CS = A; else throw c;
                           break;
                    case A: if (c == 'a') CS = S; else if (c == 'c');
                           else throw c;
                           break;
                    case S: if (c == 'b'); else if (c == '\0') return "Yes!";
                           else throw c;
                }
            } while(true);
        }
        catch(char){ return "NO!"; }
    }
    ~LexAn() { cout << "Magic number - "; }
};

class SyntAn: public PPP{
    const char * p; char c;
    void gc() { c = *p++; } // нет функции - -10, неверная - -5
    void S () { if(c=='a'){ gc(); S(); A();
                  if(c=='c') gc(); else throw c;
            }
    }
    void A () { if(c=='b'){ gc(); A(); }
    }
public: SyntAn():PPP(5) { } // за каждую ошибку в РС-методе - -5
    ~SyntAn () {cout << "Magic number - ";}
    const char * operator+ (const char * str) { // неверный алгоритм - -15
        try { p = str; // за каждую мелкую ошибку - -2
            gc(); S();
            if (c == '\0') return "Success!!!"; else throw c;
        }
        catch(char) { return "Bad string!"; }
    }
};

template <class T> // неверная функция - -10
void run_an (const char * str) { T t;
    cout << t + str << endl;
}
```

Критерии: Неверный вывод на экран: **-5** за каждую неверную строку (за 1-ю и/или 3-ю, 2-ю и/или 4-ю неверные строки вывода снижение делается по одному разу).

Экзамен, 2021 год. Ответы и критерии оценок.

Вариант 2.

Исходный балл – 100

1. Для анализа по КА выбирается регулярная грамматика, для которой автомат детерминирован – это № 3 (другие или не регулярные, или анализ по ним недетерминированный).

Критерий: сделан неверный выбор (нет обоснования) **-10**.

2. Для анализа РС-методом выбирается КС-грамматика, к которой метод применим, это № 2 (к другим метод неприменим: левая рекурсия, плохая ε -альтернатива, пересечение множеств *first (...)* в альтернативах).

Критерий: сделан неверный выбор (нет обоснования) **-10**.

3. Язык грамматики № 2: $L(2) = \{ a^n (b^m cc)^n \mid n \geq 0, m \geq 0 \}$ // ошибка: **-5**, тип = 2, иначе **-5**.

4. Язык грамматики № 3: $L(3) = \{ 0 I^n 0 I^m \mid n, m \geq 0 \}$ // ошибка: **-5**, тип = 3, иначе **-5**.

```
class LexAn : public Base {
    const char * p; char c;
    void gc() { c = *p++; } // нет функции - -10, неверная - -5
public: LexAn():Base(7){} // нет конструктора или списка инициализации - -10
    const char * operator* (const char * str) { // неверный алгоритм - -15
        try { p = str; // за каждую мелкую ошибку - -2
            enum state {H, A, S} CS;
            CS = H;
            do { gc();
                switch(CS){
                    case H: if (c == '0') CS = A; else throw c;
                           break;
                    case A: if (c == '0') CS = S; else if (c == '1'); else throw c;
                           break;
                    case S: if (c == '1'); else if (c == '\0') return "Yes!";
                           else throw c;
                }
            } while(true);
        } catch(char) {return "NO!";}
    }
    ~LexAn() { cout << "Magic number - "; }
};

class SyntAn: public Base {
    const char * p; char c;
    void gc() { c = *p++; } // нет функции - -10, неверная - -5
    void S () { if(c=='a'){ gc(); S(); B();
                  if(c=='c') gc(); else throw c;
            }
    void B() { if(c=='b'){ gc(); B(); } else if (c == 'c') gc(); else throw c;
    } // за каждую ошибку в РС-методе - -5
public: SyntAn():Base(111){ } // ошибки, как в LexAn - караются 1 раз
    ~SyntAn() {cout << "Lucky number - ";}
    const char * operator * (const char * str) { // неверный алгоритм - -15
        try { p = str; // за каждую мелкую ошибку - -2
            gc(); S();
            if (c == '\0') return "Success!!!"; else throw c;
        }
        catch(char) { return "Bad string!"; }
    }
};

template <class T> // неверная функция - -10
void analyze (const char * str) { T t;
    cout << t * str << endl;
}
```

Критерии: Неверный вывод на экран: **-5** за каждую неверную строку (за 1-ю и/или 3-ю, 2-ю и/или 4-ю неверные строки вывода снижение делается по одному разу).

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Что будет выдано на экран в результате работы следующей программы? Перечислить все конструкторы и деструкторы строго в порядке завершения их выполнения (компилятор ничего **не оптимизирует**).

```
class X { public:
    virtual X& g (int a) {
        cout << 4 << endl; return *this; }
    virtual void f (X& ob) { ob.g(1);
        cout << 3 << endl; }
};
struct Y: X { Y() { }
    Y (const Y& ob) { }
    Y& g (int a) {
        cout << 9 << endl; return *this; }
    virtual void f (Y& ob) { ob.g(7);
        cout << 8 << endl; }
};

int main() { X *p = new Y();
    try { Y x, *r = &x;
        X *p = new Y();
        X().f(x); p->f(x); r->f(x);
        try { throw Y(); cout << 10 << endl; }
        catch (X&) { cout << 11 << endl; }
            cout << 12 << endl; }
        catch (Y&) { cout << 13 << endl; }
            cout << 14 << endl;
        delete p; return 0;
    }
```

2. Внести необходимые изменения в программу, **не вводя дополнительных членов** класса A, **не меняя функцию main ()** и ничего **не удаляя**, так, чтобы в ней не было ошибок, а на экран выдалось **5 53**.

```
struct A { int n;
    int k;
    A (A& a) { n = a.n; }
    int f() { return k; }
    A (int a) { n = a; }
};

int main () { A a = A(1), b;
    cout << A::f() << " " << a.k << b.n << endl;
    return 0;
}
```

3. Вставить в пропущенные места (в `cout`) правильные вызовы функций и/или обращения к переменным (минимальной длины), чтобы на печать было выдано: **7 0 8 4 3 9 5**

```
int G() {return 1;}
int n;
namespace X { int a = 3;
    int G (int y) { return y; }
    int f () { return 4; }
    class A { int a;
        public: A (int n) { a = n; }
        int G () { return a; } };
    namespace Y { int a;
        void G () { a = 8; } }
namespace Z { int a = 5; }

int main () {
    X::A ob(7), *p = &ob;
    int a, G;
    X::Y::G();
    cout << _____ << ' ' << _____ << ' ' << _____ << ' ' << ' ';
    using namespace X;
    cout << _____ << ' ' << _____ << ' ' << _____ << ' ' << ' ';
    using namespace Y;
    cout << _____ << endl;
    return 0;
}
```

4. Что будет напечатано в результате работы следующей программы?

```
struct A {
    A () { cout << 1; }
    A (const A &) { cout << 2; }
    virtual ~A() { cout << 3; }
};
struct B : A {
    B() { cout << 4; }
    B(const B&) { cout << 5; }
    ~B() { cout << 6; }
};

void f(A & a, B & b) {
    try {
        if (typeid(a) == typeid(b)) throw a;
        else throw b;
    }
    catch (B &) { cout << "catch(B&)\n";
        b = dynamic_cast<B&>(a); }
    catch (A &) { cout << "catch(A&)\n"; }
}

int main () { try { A a; B b; f(a,b); }
    catch (...) { cout << "catch(...)\n"; }
    cout << "End\n"; return 0; }
```

5. Опишите шаблонную функцию $f(K1, K2)$ от двух параметров – непустых контейнеров STL, хранящих числовые значения, которая заменяет каждый нечетный элемент первого контейнера на первый элемент второго контейнера (нумерация элементов контейнера с 1).

6. Выполните все возможные машинно-независимые оптимизирующие преобразования, которые можно применить к данному фрагменту программы. Выпишите оптимизированный фрагмент программы.

```
#define e 2.7182818284
#define Pi 3.1415926535
float k, SUM;
int i, m, n, * p; /* ... */

for (i = 0; i < m; ++ i) C += A [i] * B [i];
for (i = 0; i < m; ++ i) p += e * i / Pi;
/* ... */
```

7. Дана КС-грамматика G :

$$\begin{aligned} S &\rightarrow Sa \mid Bb \mid Ca \mid \varepsilon \\ B &\rightarrow Ba \mid Sa \\ C &\rightarrow Bb \end{aligned}$$

- Доказать, что грамматика G неоднозначна.
- Построить эквивалентную однозначную грамматику G' , используя подходящие алгоритмы из лекций.

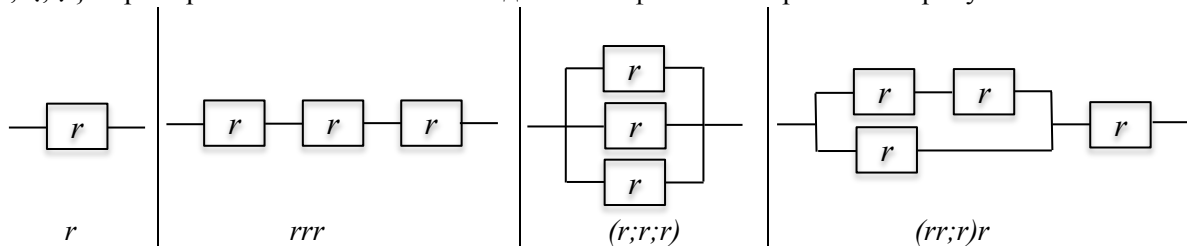
8. Дана КС-грамматика G :

$$\begin{aligned} S &\rightarrow dSd \mid aY \mid W \\ Y &\rightarrow Y \mid YYY \mid \varepsilon \\ X &\rightarrow Xa \mid a \end{aligned}$$

- Доказать, что грамматика G не является приведённой.
- Используя подходящие алгоритмы из лекций, построить эквивалентную неукорачивающую грамматику G'' , для которой применим метод рекурсивного спуска.

9. В префиксной записи выражения сначала идет знак операции, затем операнды. Например, префиксная запись арифметического выражения $a*(a+a)$ такова: $*a+aa$. Постфиксная запись (ПОЛИЗ) выражения $a*(a+a)$ выглядит как $aaa+*$. (а) Построить КС-грамматику G префиксных выражений в алфавите $\{a, +, *\}$. (б) Построить простую схему синтаксически управляемого перевода префиксных выражений в постфиксные путем добавления в грамматику G выходных символов вида $\langle \text{символ} \rangle$. (При реализации схемы перевода методом рекурсивного спуска выходные символы печатаются).

10. Электрическая цепь состоит из одинаковых резисторов r , соединяемых в разных комбинациях последовательно и параллельно, а схему такой цепи можно описать словом в алфавите $\{r, ;, (,)\}$. Примеры схем и их описаний в данном алфавите изображены на рисунке:



Общее сопротивление последовательно соединенных участков цепи с сопротивлениями $R_1, R_2, \dots, R_N$ рассчитывается по формуле $R_{\text{общ}} = R_1 + R_2 + \dots + R_N$. Общее сопротивление параллельно соединенных участков цепи $R_1, R_2, \dots, R_N$ рассчитывается по формуле $1/R_{\text{общ}} = 1/R_1 + 1/R_2 + \dots + 1/R_N$ (то есть общее сопротивление будет меньше минимального из сопротивлений параллельных участков). Дана КС-грамматика G с итерациями, описывающая электрические схемы:

$$G: \quad S \rightarrow P \quad P \rightarrow A \{A\} \quad A \rightarrow r \mid (P;B) \quad B \rightarrow P \{;P\}$$

Добавьте в грамматику семантические действия так, чтобы рекурсивным спуском допускались только электрические схемы с общим сопротивлением большим единицы, считая что сопротивление каждого резистора r равно единице.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Что будет выдано на экран в результате работы следующей программы (компилятор ничего не оптимизирует)?

```
class X { public:
    X() { cout << 0 << endl; }
    ~X() { cout << 1 << endl; }
    X(const X&) { cout << 2 << endl; }
    virtual X& g(int a) { cout << 4 << endl;
        return * this; }
    virtual void f (X & ob) { ob.g(1);
        cout << 3 << endl; }
};

struct Y: X {
    Y () { cout << 5 << endl; }
    ~Y () { cout << 6 << endl; }
    Y(const Y&ob) { cout << 7 << endl; }
    Y& g(int a) { cout << 9 << endl;
        return * this; }
    virtual void f (Y & ob) { ob.g(7);
        cout << 8 << endl; }
};

int main () { X *p = new Y();
    try { Y x, * r = & x;
        X *p = new Y();
        X().f(x);
        p -> f(x); r -> f(x);
        try { throw Y();
            cout << 10 << endl; }
        catch (X &)
            { cout << 11 << endl; }
            cout << 12 << endl; }
        catch (Y &) {cout << 13 << endl; }
            cout << 14 << endl; }
    delete p; return 0;
}
```

2. Внести необходимые изменения в программу, не вводя дополнительных членов класса A, не меняя функцию `main()` и ничего не удаляя, так, чтобы в ней не было ошибок, а на экран выдалось **0 10**.

```
struct A { int n;
    int k;
    A (A& a) { k = a.k; }
    int f() { return n; }
    A (int a) { k = a; }
};

int main () { A a = A(1), b;
    cout << A::f() << " " << a.k << b.n << endl;
    return 0;
}
```

3. Вставить в пропущенные места (в `cout`) правильные вызовы функций и/или обращения к переменным (минимальной длины), чтобы на печать было выдано: **9 1 5 0 2 8 4**

```
int G() {return 6;}
int a = 1;
namespace X {int a = 2;
    int G (int y) { return y; }
    int f () { return 0; }
    class A { int a;
        public: A (int n) { a = n; }
        int G () { return a; } };
    namespace Y {int a;
        void G () {a = 5; } } }
namespace Z { int a = 4; }

int main () {
    X::A ob(9), *p = & ob;
    int a = 7, G;
    X::Y::G();
    cout << _____ << ' ' << _____ << ' ' << _____ << ' ' << ' ';
    using namespace X;
    cout << _____ << ' ' << _____ << ' ' << _____ << ' ' << ' ';
    using namespace Z;
    cout << _____ << endl;
    return 0;
}
```

4. Что будет напечатано в результате работы следующей программы?

```
struct A {
    A () { cout << 1; }
    A (const A&) { cout << 2; }
    virtual ~A() { cout << 3; }
};

struct B : A {
    B() { cout << 4; }
    B(const B&) { cout << 5; }
    ~B() { cout << 6; }
};

int main () { try { A a; B b; f(b,b); }
    catch (...) {cout << "catch(...)\n"; }
    cout << "End\n"; return 0; }

void f(A & a, B & b) {
    try {
        if (typeid(a) == typeid(b)) throw a;
        else throw b;
    }
    catch (B &) { cout << "catch(B&)\n";
        b = dynamic_cast<B>(a); }
    catch (A &) { cout << "catch(A&)\n"; }
```


5. Опишите шаблонную функцию $f(K1, K2)$ от двух параметров – непустых контейнеров STL, хранящих числовые значения, которая заменяет каждый четный элемент первого контейнера на последний элемент второго контейнера (нумерация элементов контейнера с 1).

6. Выполните все возможные машинно-независимые оптимизирующие преобразования, которые можно применить к данному фрагменту программы. Выпишите оптимизированный фрагмент программы.

```
#define e 2.7182818284
#define Pi 3.1415926535
float k, SUM;
int i, m, n, * p; /* ... */
for (i = k = SUM; i > m * 2 - 1; i -= 4, n = i)
{ if (k < e * SUM / Pi) *(p += 4) = 0; } /* ... */
```

7. Дана КС-грамматика G :

$$\begin{aligned} S &\rightarrow Sb \mid Xa \mid Yb \mid \varepsilon \\ X &\rightarrow Xb \mid Sb \\ Y &\rightarrow Xa \end{aligned}$$

- Доказать, что грамматика G неоднозначна.
- Построить эквивалентную однозначную грамматику G' , используя подходящие алгоритмы из лекций.

8. Дана КС-грамматика G :

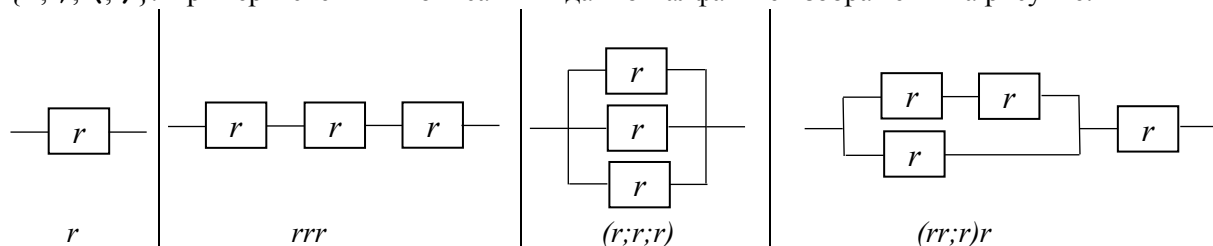
$$\begin{aligned} S &\rightarrow dB \mid aS \mid C \\ B &\rightarrow BB \mid \varepsilon \\ A &\rightarrow aAa \mid a \end{aligned}$$

- Доказать, что грамматика G не является приведенной.
- Используя подходящие алгоритмы из лекций, построить эквивалентную неукорачивающую грамматику G'' , для которой применим метод рекурсивного спуска.

9. В префиксной записи выражения сначала идет знак операции, затем операнды. Например, префиксная запись логического выражения $a \wedge (a \vee a)$ такова: $\wedge a \vee aa$. Постфиксная запись (ПОЛИЗ) выражения $a \wedge (a \vee a)$ выглядит как $aa \vee \wedge$. Знак $\wedge$ означает конъюнкцию, $\vee$ – дизъюнкцию.

- Построить КС-грамматику G префиксных выражений в алфавите $\{a, \vee, \wedge\}$.
- Построить простую схему синтаксически управляемого перевода префиксных выражений в постфиксные путем добавления в грамматику G выходных символов вида $\langle \text{символ} \rangle$. (При реализации схемы перевода методом рекурсивного спуска выходные символы печатаются).

10. Электрическая цепь состоит из одинаковых резисторов r , соединяемых в разных комбинациях последовательно и параллельно, а схему такой цепи можно описать словом в алфавите $\{r, ;, (,)\}$. Примеры схем и их описаний в данном алфавите изображены на рисунке:



Общее сопротивление последовательно соединенных участков цепи с сопротивлениями $R_1, R_2, \dots, R_N$ рассчитывается по формуле $R_{\text{общ}} = R_1 + R_2 + \dots + R_N$. Общее сопротивление параллельно соединенных участков цепи $R_1, R_2, \dots, R_N$ рассчитывается по формуле $1/R_{\text{общ}} = 1/R_1 + 1/R_2 + \dots + 1/R_N$ (то есть общее сопротивление будет меньше минимального из сопротивлений параллельных участков). Дана КС-грамматика G с итерациями, описывающая электрические схемы:

$$G: \quad S \rightarrow P \quad P \rightarrow C \{C\} \quad C \rightarrow r \mid (P;D) \quad D \rightarrow P \{;P\}$$

Добавьте в грамматику семантические действия так, чтобы рекурсивным спуском допускались только электрические схемы с общим сопротивлением меньшим единицы, считая что сопротивление каждого резистора r равно единице.

Экзамен, 2022 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: 9 - 3 - 9 - 3 - 9 - 8 - 11 - 12 - 14

Последовательность конструкторов и деструкторов:

X() Y() X() Y() X() Y() X() ~X() X() Y() X() Y(const Y&)
~Y() ~X() ~Y() ~X() ~Y() ~X() ~X()

Команда компиляции: \$> g++ -std=c++14 -fno-elide-constructors v1_1.cpp

Критерии: (1) За каждую неверную печать: **-1**, (2) за каждый пропуск/неуместный/неверный конструктор/деструктор: **-1**, (3) за неверные конструкторы/деструкторы в неоптимизируемом фрагменте (подчеркнуты): **-3**.

2. Ответ: `struct A { int n; static int k;
 A (int a = 3) { n = a; } static int f () {return k;}
 A (const A& a) { n = a.n; } };
int A::k = 5;`

Критерии: (1) Нет слова 'static' в описании функции: **-5**, при этом отсутствие 'static' и описания статического члена вне класса: **-1**, а их наличие: **-5**. (2) Есть 'static' в описании функции f(), при этом отсутствие 'static' перед описанием k: **-3**, а отсутствие описания статического члена вне класса: **-2**. (3) Нет конструктора умолчания: **-3**. (4) Нет слова 'const' в конструкторе копирования: **-3**. (5) Неверная печать: **-2**.

3. Ответ: `cout << ob.G() << ' ' << n << ' ' << X::Y::a;
 cout << f() << ' ' << X::a << ' ' << X::G(9);
 cout << Z::a << endl;`

Критерии: За каждое неверное обращение: **-2**.

4. Ответ: 11415catch(B&)- 63633catch(...)

Критерии: (1) За каждый пропуск/неуместный/неверный конструктор/деструктор: **-2**.
 (2) За каждую неверную/пропущенную печать в обработчиках исключений: **-5**.

5. Ответ: `template <class T, class S>
 void f(T x, S y) { typename T::iterator px = x.begin();
 typename S::iterator py = y.begin();
 int i = 1;
 while (px != x.end()) {
 if (i) { *px = *py; }
 px++;
 i = !i;
 }
 }`

Критерии: (1) Нет слов 'typename': **-5**. (2) Другие ошибки: **-3** за каждую.

6. Ответ: 1. Слияние циклов.
 2. Перестановка и вычисление значений операций с известными операндами (e/π).
 (Либо 1-2: второй цикл можно вообще удалить, заменив суммой.)

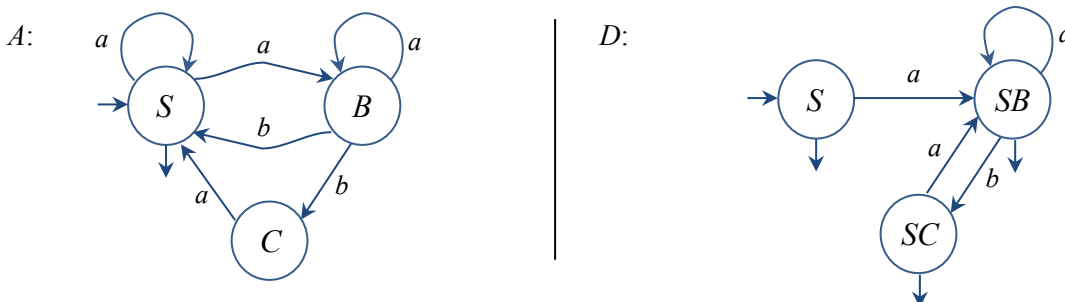
Критерии: За отсутствие пункта (1): **-5**, за отсутствие пункта (2): **-5**.

7. Ответ: (а) Существуют два различных правых вывода для цепочки *aab* (возможны и другие цепочки с различными правыми – они же левые – выводами):

$S \rightarrow Bb \rightarrow Sab \rightarrow Sabb \rightarrow abb$ и $S \rightarrow Bb \rightarrow Bab \rightarrow Sabb \rightarrow abb$.

Этим выводам соответствуют различные деревья вывода, значит, грамматика *G* неоднозначна.

(б) Алгоритма, который по произвольной КС-грамматике строит эквивалентную однозначную, не существует. Однако грамматика G левوليнейна, – по ней можно построить эквивалентный конечный автомат A : После детерминизации A получается эквивалентный ДКА D :



По автомату D строится эквивалентная праволинейная грамматика G' :

$$S \rightarrow aS_{SB} \mid \varepsilon \quad S_{SB} \rightarrow aS_{SB} \mid bS_{SC} \mid \varepsilon \quad S_{SC} \rightarrow aS_{SB} \mid \varepsilon$$

Полученная по ДКА грамматика однозначна, так как для любой допускаемой цепочки в ДКА существует единственный путь и ему соответствует единственный вывод в построенной праволинейной грамматике. **Замечание.** Левوليнейную грамматiku по ДКА D построить невозможно из-за наличия более, чем одного заключительного состояния. Если свести их в одно новое состояние через маркер конца цепочки, то теряется эквивалентность – в исходном языке не было маркера конца.

Критерии: За ошибку в пункте (а): -5, за ошибки в пункте (б): -5. Обоснование в пункте (б) не требуется, достаточно построенной через ДКА праволинейной грамматики.

8. Ответ: (а) Грамматика G не является приведённой, так как символ W бесплоден (или потому что X недостижим). (б) Алгоритм приведения грамматики строит эквивалентную грамматику:

$$S \rightarrow dSd \mid aY \quad Y \rightarrow Y \mid YYY \mid \varepsilon$$

Алгоритм устранения пустых правых частей в КС-грамматике строит промежуточную грамматику:

$$S \rightarrow dSd \mid aY \mid a \quad Y \rightarrow Y \mid YY \mid YYY$$

С учётом последнего шага по приведению после устранения эpsilon-правил, строится грамматика:

$$S \rightarrow dSd \mid a$$

Эта грамматика неукорачивающая, она удовлетворяет критерию применимости метода рекурсивного спуска, поэтому больше никаких преобразований не требуется – это и есть искомая грамматика G'' .

Критерии: За ошибку в пункте (а): -5, за наличие ошибок в пункте (б): -5 (подробное обоснование не требуется, достаточно построений по соответствующим алгоритмам).

9. Ответ: (а) Грамматика G для префиксных выражений выглядит так:

$$S \rightarrow +SS \mid *SS \mid a$$

(б) Схема синтаксически управляемого перевода префиксных выражений в постфиксные такова:

$$S \rightarrow +SS \langle + \rangle \mid *SS \langle * \rangle \mid a \langle a \rangle$$

Критерии: За ошибку в пункте (а): -5, за наличие ошибок в пункте (б): -5.

10. Ответ: Строится глобальный стек вещественных значений $stack<double>s$ и локальные переменные $double R$ в каждой из процедур P, A, B .

```
S → {while (!s.empty()) s.pop();} P {if (s.top() > 1) s.pop(); else error();}
P → A {R = s.top(); s.pop();} { A {R += s.top(); s.pop();} } {s.push(R);}
A → r {s.push(1);} | (P; {R = 1/s.top(); s.pop();} B {R += s.top(); s.pop();} ) {s.push(1/R);}
B → P {R = 1/s.top(); s.pop();} {; P {R += 1/s.top(); s.pop();} } {s.push(R);}
```

Критерии: за каждое ошибочное/отсутствующее действие -2, за отсутствие очистки стека не снижать.

Экзамен, 2022 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ: (в скобках указано количество снижаемых баллов за ошибку в очередном фрагменте; все числа печатаются в столбик, но здесь для экономии места – через пробел):

0 5 (-1) 0 5 (-1) 0 5 (-1) 0 9 3 1 (-2) 9 3 (-2) 9 8 (-2)
0 5 0 7 6 1 (-3) 11 6 1 (-2) 12 6 1 (-2) 14 (-1) 1 (-2)

Команда компиляции: `$> g++ -std=c++14 -fno-elide-constructors v2_1.`

Критерии: За каждый фрагмент с ошибкой – снижение на указанное количество баллов.

2. Ответ: `struct A { int k; static int n;
A (int a = 1) { k = a; }
A (const A& a) { k = a.k; } static int f () {return n; } };
int A::n;`

Критерии: (1) Нет слова 'static' в описании функции: -5, при этом отсутствие 'static' и описания статического члена вне класса: -1, а их наличие: -5. (2) Есть 'static' в описании функции f(), при этом отсутствие 'static' перед описанием n: -3, а отсутствие описания статического члена вне класса: -2. (3) Нет конструктора умолчания: -3. (4) Нет слова 'const' в конструкторе копирования: -3. (5) Неверная печать: -2.

3. Ответ: `cout << ob.G() << ' ' << ::a << ' ' << X::Y::a;
cout << f() << ' ' << X::a << ' ' << X::G(8);
cout << Z::a << endl;`

Критерии: За каждое неверное обращение: -2.

4. Ответ: 1142catch(A&) 3633End

Критерии: (1) За каждый пропуск/неуместный/неверный конструктор/деструктор: -2.
(2) За каждую неверную/пропущенную печать в обработчиках исключений: -5.

5. Ответ: `template <class T, class S>
void f(T x, S y) { typename T::iterator px = x.begin();
typename S::iterator py = y.rbegin();
int i = 0;
while (px != x.end()) {
if (i) { *px = *py; }
px++;
i = !i;
}
}`

Критерии: (1) Нет слов 'typename': -5. (2) Другие ошибки: -3 за каждую.

6. Ответ: 1. Исключение избыточных вычислений ($n=i$) и лишних переменных (n).
2. Вынесение условного оператора из цикла (for-if => if-for).

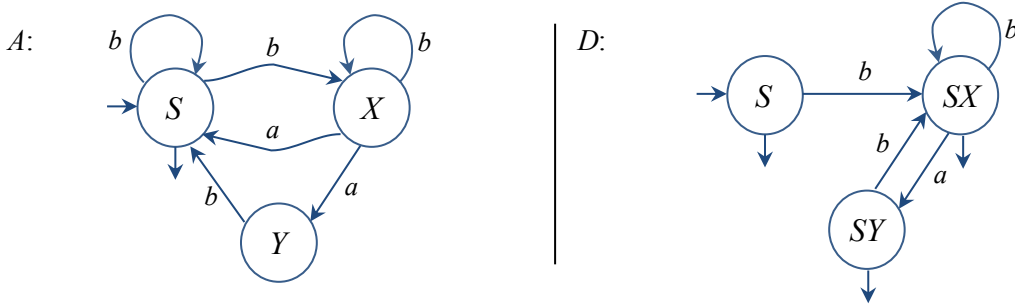
Критерии: За отсутствие пункта (1): -5, за отсутствие пункта (2): -5.

7. Ответ: (а) Существуют два различных правых вывода для цепочки bba (возможны и другие цепочки с различными правыми – они же левые – выводами):

$$S \rightarrow Xa \rightarrow Sba \rightarrow Sbba \rightarrow bba \text{ и } S \rightarrow Xa \rightarrow Xba \rightarrow Sbba \rightarrow bba.$$

Этим выводам соответствуют различные деревья вывода, значит, грамматика G неоднозначна.

(б) Алгоритма, который по произвольной КС-грамматике строит эквивалентную однозначную, не существует. Однако грамматика G левوليнейна, – по ней можно построить эквивалентный конечный автомат A : После детерминизации A строится эквивалентный ДКА D :



По автомату D строится эквивалентная праволинейная грамматика G' :

$$S \rightarrow bS_{SX} \mid \varepsilon \quad S_{SX} \rightarrow bS_{SX} \mid aS_{SY} \mid \varepsilon \quad S_{SY} \rightarrow bS_{SX} \mid \varepsilon$$

Полученная по ДКА грамматика однозначна, так как для любой допускаемой цепочки в ДКА существует единственный путь и ему соответствует единственный вывод в построенной праволинейной грамматике. **Замечание.** Левوليнейную грамматiku по ДКА D построить невозможно из-за наличия более чем одного заключительного состояния. Если свести их в одно новое состояние через маркер конца цепочки, то теряется эквивалентность – в исходном языке не было маркера конца.

Критерии: За ошибку в пункте (а): -5, за ошибки в пункте (б): -5. Обоснование в пункте (б) не требуется, достаточно построенной через ДКА праволинейной грамматики.

8. Ответ: (а) Грамматика G не является приведённой, так как символ C бесплоден (или потому что A недостижим). (б) Алгоритм приведения грамматики строит эквивалентную грамматику:

$$S \rightarrow dB \mid aS \quad B \rightarrow BB \mid \varepsilon$$

Алгоритм устранения пустых правых частей в КС-грамматике, строит промежуточную грамматику:

$$S \rightarrow dB \mid aS \mid a \quad B \rightarrow BB \mid B$$

С учётом последнего шага по приведению после устранения эpsilon-правил, строится грамматика:

$$S \rightarrow aS \mid d$$

Эта грамматика неукорачивающая, она удовлетворяет критерию применимости метода рекурсивного спуска, поэтому больше никаких преобразований не требуется – это и есть искомая грамматика G'' .

Критерии: За ошибку в пункте (а): -5, за наличие ошибок в пункте (б): -5 (подробное обоснование не требуется, достаточно построений по соответствующим алгоритмам).

9. Ответ. (а) Грамматика G для префиксных выражений выглядит так:

$$S \rightarrow \vee SS \mid \wedge SS \mid a$$

(б) Схема синтаксически управляемого перевода префиксных выражений в постфиксные такова:

$$S \rightarrow \vee SS \langle \vee \rangle \mid \wedge SS \langle \wedge \rangle \mid a \langle a \rangle$$

Критерии: За ошибку в пункте (а): -5, за наличие ошибок в пункте (б): -5.

10. Ответ: Строится глобальный стек вещественных значений $stack<double>s$ и локальные переменные $double R$ в каждой из процедур P, C, D .

```
S → {while (!s.empty()) s.pop();} P {if (s.top() < 1) s.pop(); else error();}
P → C {R=s.top(); s.pop();} { C {R+=s.top(); s.pop();} } {s.push(R);}
C → r {s.push(1);} | ( P; {R=1/s.top(); s.pop();} D {R+=s.top(); s.pop();} ) {s.push(1/R);}
D → P {R=1/s.top(); s.pop();} {; P {R+=1/s.top(); s.pop();} } {s.push(R);}
```

Критерии: за каждое ошибочное/отсутствующее действие -2, за отсутствие очистки стека не снижать.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8

1. (15 баллов) Что будет выдано на печать в результате работы следующей программы?

```
struct A {
    A() { cout << 1; }
    A(const A&) { cout << 2; }
    virtual ~A() { cout << 3; }
};

struct B : A {
    B() { cout << 4; }
    B(const B&) { cout << 5; }
    ~B() { cout << 6; }
};

void f(A &a, B &b) {
    try {
        if (typeid(a) == typeid(b)) throw a;
        else throw b;
    }
    catch (B&) { cout << "catch (B&) \n"; b = dynamic_cast<B&>(a); }
    catch (A&) { cout << "catch (A&) \n"; b = dynamic_cast<B&>(a); }
}

int main() {
    try { A a; B b; f(a, b); }
    catch (...) { cout << "catch(...)"; }
    cout << "End\n"; return 0; }
}
```

2. (20 баллов) Опишите необходимые класс *A* и булевскую функцию *fun()* от двух параметров (контейнера-списка и целого), определяющую, есть ли в списке элемент, соответствующий заданному целому, так, чтобы в приведенной ниже функции *main()* не было ошибок.

```
int main() {
    const list<A> L { /* инициализация списка */ };
    cout << A::n << fun<A>(L, 2) << endl;
}
```

3. (10 баллов) Модифицируйте только описание классов *A* и *B*, ничего в них не удаляя каким-либо образом, так, чтобы в приведенной ниже функции *main()* не было ошибок и ситуаций *undefined behavior* и напечаталось: *ConstrA ConstrB B::f() DestrB DestrA*.

```
struct A {
    A(int a) { cout << "ConstrA "; }
    ~A() { cout << "DestrA "; }
    virtual void g() = 0;
    void f() { cout << "A::f() "; }
};

struct B : A {
    B() { cout << "ConstrB "; }
    ~B() { cout << "DestrB "; }
    void f() { cout << "B::f() "; }
};

int main() { A &p = *new B(); p.f(); delete &p; }
```

4. (5 баллов) Приведенная ниже программа не компилируется. Укажите причины ошибок и модифицируйте двумя способами только описание класса *A* так, чтобы ошибок не было. **Что будет напечатано** в результате работы получившейся программы?

```
class A { int x, y;
public:
    A(int a = 5) { x = y = a; }
};

namespace N1 {
    void f(int) { cout << "N1::f(int)\n"; }
}

void f(const A &a) { cout << a.x << a.y << "f(A)\n"; }

int main() { A a; f(a); f(2); }
```

5. (10 баллов) Существует ли приведённая КС-грамматика с единственным правилом вывода (без альтернатив), к которой метод рекурсивного спуска неприменим? Обоснуйте ответ.

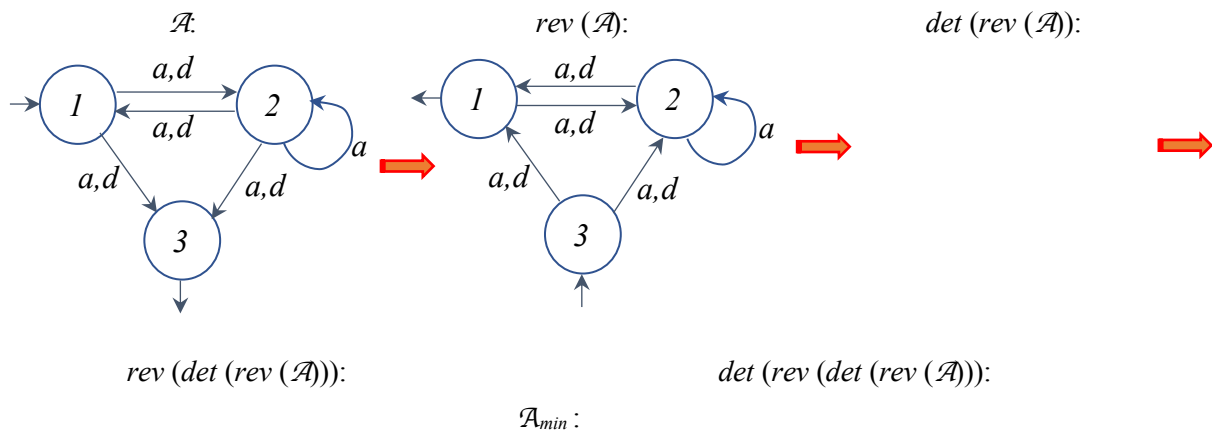
6. (5 баллов) Дайте определение понятию *система программирования*.

7. (20 баллов) В префиксной записи выражения сначала идет знак операции, затем операнды. Например, префиксная запись выражения $a*(a+a)$ такова: $*a+aa$. Дана заготовка грамматики с действиями C++. Добавьте в нее недостающие действия так, чтобы анализатор, построенный методом рекурсивного спуска, переводил инфиксную (обычную) запись в префиксную, например: $a*a*(a+a) \Rightarrow **aa+aa$. Анализатору доступен глобальный стек `stack<string>st` и своя локальная переменная `string s` в каждой рекурсивной функции S, A, M, F . В качестве добавляемых действий допускаются только операторы-выражения, содержащие операции со стеком и/или арифметические операции с присваиванием.

```
S → <while(!st.empty()) st.pop();> A <s = st.top(); st.pop(); cout << s;>
A → M { + M }
M → F { * F }
F → (A) | a <s="a"; st.push(s);>
```

8. (15 баллов) Обращением $rev(A)$ конечного автомата A будет конечный автомат, у которого все начальные (входные) вершины A становятся заключительными (выходными), а выходные – входными, и направления дуг заменяются на противоположные (см. рисунок). Через $det(A)$ обозначим результат детерминизации автомата A . По заданному автомату A постройте эквивалентный и минимальный по числу состояний ДКА A_{min} , используя алгоритм Бржозовского:

$$A_{min} := det (rev (det (rev (A)))) .$$



NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8

1. (15 баллов) Что будет выдано на печать в результате работы следующей программы?

```
struct A {      A()      { cout << 1; }
               A(const A&) { cout << 2; }
               virtual ~A() { cout << 3; }
};

void f (A & a, B & b) { try { if (typeid(a) == typeid(b)) throw a;
                           else                               throw b;
                           }
  catch (B&) { cout << "catch(B&)\n"; b = dynamic_cast <B&>(a); }
  catch (A&) { cout << "catch(A&)\n"; b = dynamic_cast <B&>(a); }
}

int main() { try { A a;   B b;   f (b, b);
                }
  catch (...) { cout << "catch(...)";
               cout << "End\n";      return 0;  }
}
```

2. (20 баллов) Опишите необходимые класс `A` и функцию `count()` от двух параметров (контейнера-списка и целого), подсчитывающую количество элементов списка, соответствующих заданному целому, так, чтобы в приведенной ниже функции `main()` не было ошибок.

```
int main() { const list<A> L { /* инициализация списка */;
  cout << A::n << count<A> (L, 2) << endl;
}
```

3. (10 баллов) Модифицируйте **только** описание классов `A` и `B`, **ничего в них не удаляя каким-либо образом**, так, чтобы в приведенной ниже функции `main()` не было ошибок и ситуаций *undefined behavior* и напечаталось: `ConstrA ConstrB B::g() DestrB DestrA`.

```
struct A {      A(int a) { cout << "ConstrA "; }
               ~A()     { cout << "DestrA "; }
               virtual void f() = 0;
               void g()   { cout << "A::g() "; }
};

struct B : A {   B()      { cout << "ConstrB "; }
               ~B()     { cout << "DestrB "; }
               void g()   { cout << "B::g() "; }
};

int main() { A * p = new B(); p -> f(); delete p; }
```

4. (5 баллов) Приведенная ниже программа не компилируется. Укажите **причины** ошибок и **модифицируйте двумя способами только** описание класса `A` так, чтобы ошибок не было. Что будет напечатано в результате работы получившейся программы?

```
class A { int x, y;
public:
    A(int a = 5) { x = y = a; }
};

namespace N1 {
    void f (int) { cout << "N1::f(int)\n"; }
}

void f(const A & a) { cout << a.x << a.y << "f(A)\n"; }
using namespace N1;
int main() { A a; f(a); f(2); }
```

5. (10 баллов) Существует ли приведённая КС-грамматика, которая порождает единственную цепочку, имеет два правила вывода (считая альтернативы), и к которой метод рекурсивного спуска неприменим? Обоснуйте ответ.

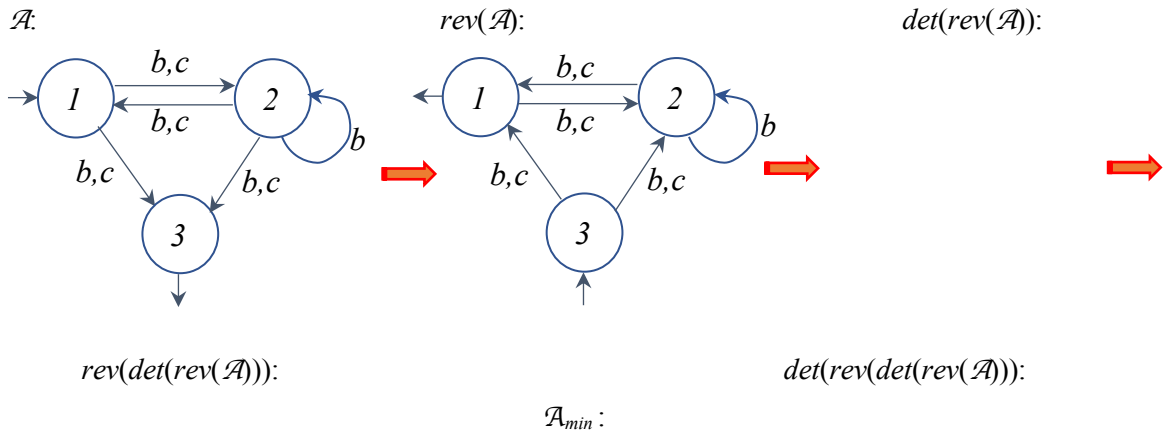
6. (5 баллов) Дайте определение понятию *программный продукт*.

7. (20 баллов) В префиксной записи выражения сначала идет знак операции, затем операнды. Например, префиксная запись арифметического выражения $a*(a+a)$ такова: $*a+aa$. Постфиксная запись (ПОЛИЗ) выражения $a*(a+a)$ выглядит как $aaa+*$, а обращение постфиксной записи $aaa+*$ таково: $*+aaa$. Дана заготовка грамматики с действиями C++. Грамматика порождает обращения постфиксных записей. Добавьте в нее недостающие действия так, чтобы анализатор, построенный методом рекурсивного спуска, переводил обращение постфиксной записи в префиксную, например: $**+aaaa \Rightarrow *a*a+aa$. Анализатору доступен глобальный стек `stack<string>st` и своя локальная переменная `string s` в каждой рекурсивной функции S и A . В качестве добавляемых действий допускаются только операторы-выражения, содержащие операции со стеком и/или арифметические операции с присваиванием.

```
S → {while(!st.empty()) st.pop();} A {s = st.top(); st.pop(); cout << s;}
A → + A A | * A A | a {s = "a"; st.push(s);}
```

8. (15 баллов) Обращением $rev(A)$ конечного автомата A будет конечный автомат, у которого все начальные (входные) вершины A становятся заключительными (выходными), а выходные – входными, и направления дуг заменяются на противоположные (см. рисунок). Через $det(A)$ обозначим результат детерминизации автомата A . По заданному автомату A постройте эквивалентный и минимальный по числу состояний ДКА A_{min} , используя алгоритм Бржозовского:

$$A_{min} := det (rev (det (rev (A)))).$$



Коллоквиум, 2023 год. Ответы и критерии оценок.
Вариант1 (в задании 2 – функция fun).
Максимальная оценка всех восьми задач в совокупности 100 баллов

1. (15 баллов) Будет выдано в выходной поток: **11415catch (B&) 63633catch (...) End**

Критерии: за каждую отсутствующую/неверную печать -3.

2. (20 баллов)

```
struct A { static int n; int t;
    A (int m =3) {t = m;}
    bool operator== (int a) const { return t == a; }
};
int A::n = 3;
template <class T> bool fun (const list<T> & v, int a) {
typename list<T>::const_iterator p = v.begin();
    while (p != v.end()) {
        if (*p == a) return true;
        p++;
    }
    return false;
}
```

Критерии: за отсутствие (неверная вставка) каждого выделенного элемента -5.

3. (10 баллов)

```
struct A { A (int a){ cout << "ConstrA ";}
    virtual ~A() { cout << "DestrA "; }
    virtual void g() = 0;
    virtual void f(){ cout << "A::f() ";}
};
struct B : A { B() : A(1) { cout << "ConstrB ";} // или A(int a = 1)
    ~B() { cout << "DestrB "; }
    void g() {}
    void f(){ cout << "B::f() ";}
};
```

Критерии: за отсутствие (неверная вставка) каждого выделенного элемента -4.

4. (5 баллов). Добавить в класс 'friend void f(const A&);' или поместить x и y в открытой области класса (public). В поток будет выдано: **55f (A) 22f (A)**

Критерии: за отсутствие каждого варианта исправления -2, за каждую неверную печать -2.

5. (10 баллов). *Нет.* Единственное правило вывода не содержит альтернатив – следовательно, все три пункта критерия применимости заведомо выполняются.

Критерии: за неверный или необоснованный(неверно обоснованный) ответ -10.

6. (5 баллов). Система программирования – набор инструментальных средств для поддержки программного продукта в течение всего его жизненного цикла.

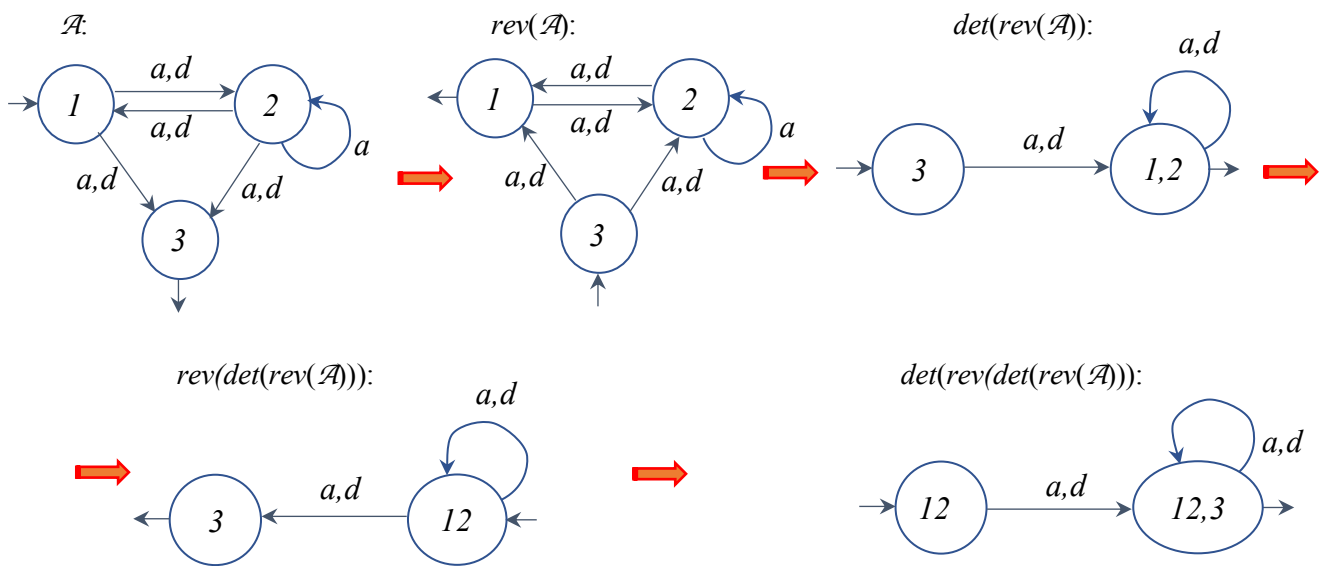
Критерии: неверный ответ -5.

7. (20 баллов).

```
S → <while(!st.empty()) st.pop();> A {s = st.top(); st.pop(); cout << s;}
A → M { + M {s = st.top(); st.pop(); s = "+" + st.top() + s; st.pop(); st.push(s);} }
M → F { * F {s = st.top(); st.pop(); s = "*" + st.top() + s; st.pop(); st.push(s);} }
F → (A) | a {s = "a"; st.push(s);}
```

Критерии: за ошибочные действия для каждого из нетерминалов A и M -10.

8. (15 баллов).



Критерии: За ошибку (отсутствие) на шаге1 ($det(rev(\mathcal{A}))$) **-6**, за ошибку (отсутствие) на шаге2 ($rev(det(rev(\mathcal{A})))$) **-3**, за ошибку (отсутствие) на шаге3 ($det(rev(det(rev(\mathcal{A}))))$) **-6**.

Коллоквиум, 2023 год. Ответы и критерии оценок.
Вариант1 (в задании 2 – функция count).
Максимальная оценка всех восьми задач в совокупности 100 баллов

1. (15 баллов). Будет выдано в выходной поток: **1142catch(A&) 3633End**

Критерии: за каждую отсутствующую/неверную печать **-3**.

2. (20 баллов)

```
struct A { // возможны варианты
    static int n; int t;
    A(int m = 3) {t = m;}
    bool operator== (int a) const { return t == a; }
};
int A::n = 3;
template <class T> int count (const list <T> & v, int a) {
    int k = 0;
    typename list <T>::const_iterator p = v.begin();
    while (p != v.end()) {
        k += *p == a;
        p++;
    }
    return k;
}
```

Критерии: за отсутствие (неверная вставка) каждого выделенного элемента - **5**.

3. (10 баллов)

```
struct A {
    A (int a) { cout << "ConstrA ";}
    virtual ~A() { cout << "DestrA "; }
    virtual void f() = 0;
    void g() { cout << "A::g() ";}
};
struct B : A {
    B() : A(1) { cout << "ConstrB ";} // или A(int a = 1)
    ~B() { cout << "DestrB "; }
    void f() {g();}
    void g() { cout << "B::g() ";}
};
```

Критерии: за отсутствие (неверная вставка) каждого выделенного элемента - **4**.

4. (5 баллов). Добавить в класс 'friend void f(const A&);' или поместить x и y в открытой области класса (public). В поток будет выдано: **55f(A) N1::f(int)**

Критерии: за отсутствие каждого варианта исправления **-2**, за каждую неверную печать **-2**.

5. (10 баллов) . Да. Например, $S \rightarrow S \mid \varepsilon$. В данной грамматике выводится только ε и нарушается пункт 2 критерия применимости рекурсивного спуска: из двух альтернатив выводится ε . Возможный вариант ответа: $S \rightarrow S \mid a$. В этой грамматике выводится только a , и нарушается пункт 1 критерия применимости рекурсивного спуска: множества $first()$ у альтернатив пересекаются из-за левой рекурсии.

Критерии: за неверный или необоснованный(неверно обоснованный) ответ **-10**.

6. (5 баллов). Программный продукт – программа, которую можно использовать отчужденно от ее автора, в разных операционных средах.

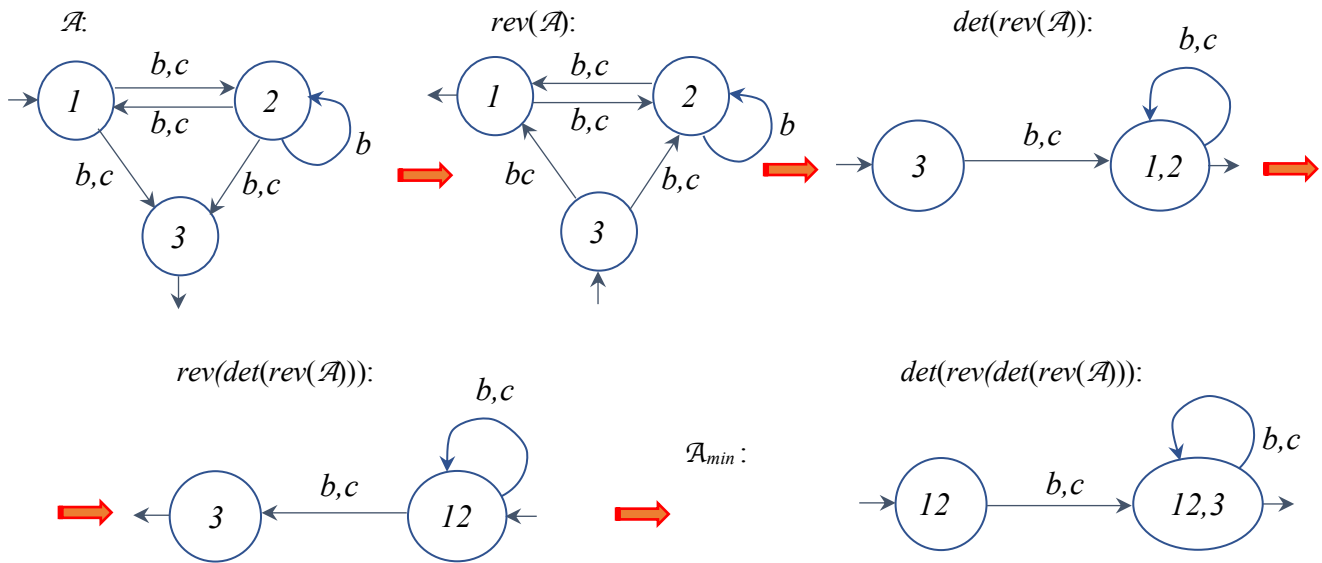
Критерии: неверный ответ **-5**.

7. (20 баллов).

```
S → <while(!st.empty()) push st.pop();> A < s = st.top(); st.pop(); cout << s;>
A → + A A <s = st.top(); st.pop(); s = "+" + s + st.top(); st.pop(); st.push(s);> |
    * A A <s = st.top(); st.pop(); s = "*" + s + st.top(); st.pop(); st.push(s);> |
    a < s = "a"; st.push(s);>
```

Критерии: за ошибочные действия для каждой альтернативы **A -10**.

8. (15 баллов).



Критерии: За ошибку (отсутствие) на шаге1 ($det(rev(\mathcal{A}))$) **-6**, за ошибку (отсутствие) на шаге2 ($rev(det(rev(\mathcal{A})))$) **-3**, за ошибку (отсутствие) на шаге3 ($det(rev(det(rev(\mathcal{A}))))$) **-6**.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std.`

1	2	3	4	5	6	7	8	9	10

1. Вычеркните неразрешимые вызовы функции `g()`. Что напечатает получившаяся программа?

```

struct B { operator int () {return 3;} };
void g(B& b, double d){cout << "g(B&, double)\n";}
void g(char c, const char * b){cout << "g(char, const char*)\n";}
void g(char c, int d){cout << "g(char, int)\n";}

int main() {
    B b;
    g (1.5, 0);
    g (b, 2);
    g (2, b);
}

```

2. Что такое **абстрактный тип данных** (АТД, не путать с абстрактным классом) в языке Си++? Привести пример описания и использования такого класса.

3. Если при компиляции следующей программы будут ошибки, **объясните их**. Исправьте ошибки, модифицируя **только функцию `main()`**, ничего в ней не удаляя каким-либо образом.

```

namespace N1 { int x = 1, y = 3;
               void f() {cout << "f()\n";}
               void g() {cout << "g()\n";}
}
namespace N2 { int x = 5, y = 7;
               int f() { return 1; }
               int g (int a){ return x = a; }
}

using namespace N1;
using namespace N2;

int main () { g();
              y = x + g(1) + f();
}

```

4. Опишите одну функцию $F()$ так, чтобы вычисление выражения $c = F(a, b)$ с переменными любого одинакового типа (имеющего соответствующую операцию присваивания) приводило к тому, что в переменной a установится исходное значение b , в переменной b – значение, получаемое выражением $b=1$, а в переменной c – первоначальное значение a . Приведите пример описания подходящего класса для переменных a, b, c .

5. Следующая программа построена по конечному автомату $\mathcal{A}$ с состояниями A и B и допускает те и только те входные цепочки, которые допускает автомат $\mathcal{A}$. (а) Опишите конечный автомат $\mathcal{A}$ (в виде диаграммы состояний). (б) Какой язык задает автомат $\mathcal{A}$? (охарактеризовать множество цепочек).

```

class Scanner { static char c;
               class State { public:
                   State * next = this;
                   virtual State* operator -> () const = 0;
                   virtual ~State() {}
               };
               class stA: public State { public:
                   State * operator ->() const {
                       if (c == 'b') return &B;
                       else if (c == 'a') return &A;
                       else throw c;
                   }
               };
               class stB: public State { public:
                   State * operator ->() const {
                       if (c=='b') return &A;
                       else if (c=='a') return &B;
                       else throw c;
                   }
               };
               static stB B; // вершина B
               static stA A; // вершина A
               State * vert; // указатель текущей
               // вершины на диаграмме состояний
               // автомата

public:
    void gc(){ cin.get(c); } // считать
                               // очередной символ
    bool accept(){
        try { vert=&A;
            while (gc(), !cin.eof()) {
                vert = (* vert) -> next;
            }
            return typeid (*vert) ==
                   typeid (stA);
        }
        catch (...) { return false; }
    }; // конец класса Scanner

char Scanner::c;
Scanner::stA Scanner::A;
Scanner::stB Scanner::B;

int main() { Scanner G1;
    cout << (G1.accept() ? "\nYes": "\nNo")
         << endl;
}

```

6. Дана программа синтаксического анализа по принципу рекурсивного спуска для грамматики G с нетерминалами A и S , печатающая анализируемую входную цепочку, выводимую в грамматике.

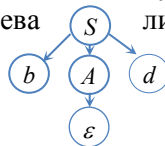
```
class Parser {
    static char c; // текущий символ
                    // (лексема)
    static void gc() {cin>>c;}
    // получить следующий символ (лексему)
    class S { public:
        void parse() {
            if (c=='b') { cout<<'b'; gc();
                A().parse();
            }
            if (c=='d') { cout<<'d'; gc();
            } else throw c;
            else if (c=='d') {cout<<'d'; gc();
            }
            else throw c;
        }
    };
};

class A { public:
    void parse() {
        if (c=='a') {
            cout<<'a'; gc(); A().parse();
        }
    };
};

public: void analyze() {
    try { gc(); S().parse();
        if (!cin.eof()) throw -1;
    }
    catch(...) { cout << "\nERROR";
        cout << endl;
    }
};

char Parser::c;
int main() { Parser().analyze(); }
```

(а) Восстановите грамматику G по программе. (б) Не изменяя методы $parse()$, добавьте недостающие описания в классах A и S так, чтобы правильные цепочки переводились данной программой в линейную запись соответствующего дерева вывода. Символу S в линейной записи соответствует пара скобок '(' и ')', обрамляющих выводимую из S цепочку, символу A соответствует пара скобок '<' и '>'. Например, для дерева



7. Дана грамматика G : $S \rightarrow Sa | Ab$
 $A \rightarrow Aa | Ab | c$

(а) Обоснуйте следующее утверждение: любую автоматную грамматику через детерминизацию диаграммы состояний можно преобразовать в эквивалентную праволинейную, к которой применим метод рекурсивного спуска. (б) Преобразуйте заданную грамматику G в соответствии с пунктом (а).

8. Дана КС-грамматика G_a с действиями над глобальными переменными $depth$ и $level$.

(а) Какой язык задает данная грамматика (какое множество цепочек)? (б) Построить КС-грамматику G без действий для языка $L(G_a)$, содержащую не более 4-х правил вывода, считая альтернативы: G_a : $S \rightarrow \langle depth=0; level=0; \rangle A \langle \text{if}(depth>2) \text{error}(); \rangle$
 $A \rightarrow (\langle level++; depth = level > depth ? level : depth; \rangle A) \langle level--; \rangle A | \varepsilon$

9. Может ли быть неоднозначной грамматика, порождающая пустой язык? Обоснуйте ответ.

10. Дана постфиксная запись (ПОЛИЗ) программы на модельном М-языке. (а) Восстановить по ней тело программы на М-языке. (б) Оптимизируйте эту запись по длине, то есть постройте эквивалентную (выдающую тот же результат) запись минимальной длины, считая, что вводимое значение b всегда положительное. **Замечание.** Операции mod (остаток от деления) в М-языке нет, но есть деление с отбрасыванием остатка (/), сложение, умножение, вычитание и сравнение. Сокращенный условный оператор if-then без else есть.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ	&a	5	:=	&b	read	a	b	<	30	!F	a	b	>	23	!F	&a	a
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32			
b	-	:=	28	!	&b	b	a	-	:=	6	!	a	write				
ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(оптим.)																	
18	19	20	21	22	23	24	25	26	27	28	29	30	31				

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8	9	10

1. Вычеркните неразрешимые вызовы функции `g()`. Что напечатает получившаяся программа?

```

struct B { B(int n = 0){} };
void g(B& b, double d){cout << "g(B&, double)\n";}
void g(char c, const char * b){cout << "g(char, const char*)\n";}
void g(char c, int d){cout << "g(char, int)\n";}

int main() {
    B b;
    g(1.5, 0);
    g(b, 2);
    g(2, b);
}

```

2. Что такое **функция-друг** в языке Си++? Привести пример описания и использования функции-друга.

3. Если при компиляции следующей программы будут ошибки, **объясните их**. Исправьте ошибки, модифицируя **только функцию `main()`**, **ничего в ней не удаляя** каким-либо образом.

```

namespace N1 { int x = 1, y = 3;
                void f() {cout << "f()\n";}
                void g() {cout << "g()\n";}
}
namespace N2 { int x = 5, y = 7;
                int f() { return 1; }
                int g(int a){ return x = a; }
}

using namespace N1;
using namespace N2;

int main() { int y;
              y = x + f() + g(3);
}

```

4. Опишите не более двух перегруженных функций `G()` так, чтобы вычисление выражения `c=G(a,b)`, с переменными любого одинакового типа `a` и `b` (имеющего операцию сравнения '>') и целой переменной `c` приводило к тому, что в переменной `c` будет значение `1`, если `a>b`, и значение `0` – иначе. Сравнение Си-строк `a` и `b` должно происходить лексикографически с использованием `strcmp()`.

5. Следующая программа построена по конечному автомату $\mathcal{A}$ с состояниями A и B и допускает те и только те входные цепочки, которые допускает автомат $\mathcal{A}$. (а) Опишите конечный автомат $\mathcal{A}$ (в виде диаграммы состояний). (б) Какой язык задает автомат $\mathcal{A}$? (охарактеризовать множество цепочек).

```

class Scanner { static char c;
                class State { public:
                    State * next = this;
                    virtual State * operator->() const = 0;
                    virtual ~State(){}
                };
                class stB: public State { public:
                    State * operator->() const {
                        if (c == 'a') return &A;
                        else if (c == 'b') return &B;
                        else throw c;
                    }
                };
                class stA: public State { public:
                    State * operator->() const {
                        if (c=='a') return &B;
                        else if (c=='b') return &A;
                        else throw c;
                    }
                };
                static stA A; // вершина A
                static stB B; // вершина B
                State *vert; // указатель текущей
                // вершины на диаграмме состояний
                // автомата

public:
    void gc(){ cin.get(c); } // считать
                               // очередной символ
    bool accept(){
        try{ vert=&B;
            while (gc(),!cin.eof()){
                vert=(*vert)->next;
            }
            return typeid (*vert) ==
                   typeid (stB);
        }
        catch(...) {return false;}
    }
}; // конец класса Scanner

char Scanner::c;
Scanner::stB Scanner::B;
Scanner::stA Scanner::A;

int main() {
    Scanner G1;
    cout << (G1.accept() ? "\nYes": "\nNo")
         << endl;
}

```


6. Дана программа синтаксического анализа по принципу рекурсивного спуска для грамматики G с нетерминалами A и S , печатающая анализируемую входную цепочку, выводимую в грамматике.

```
class Parser {
    static char c; // текущий символ
                    // (лексема)
    static void gc() {cin>>c;}
    // получить следующий символ (лексему)
    class A { public:
        void parse() {
            if (c=='b') { cout<<'b'; gc();
                A().parse();
            }
            if (c=='d') { cout<<'d'; gc();
            } else throw c;
        }
        else if (c=='c') { cout<<'c'; gc();
        }
        else throw c;
    }
};
```

```
class S { public:
    void parse() {
        if (c=='a') {
            cout<<'a'; gc(); S().parse();
        }
        else A().parse();
    }
};

public: void analyze() {
    try { gc(); S().parse();
        if (!cin.eof()) throw -1;
    }
    catch (...) { cout<< "\nERROR";
        cout<<endl;
    }
};

char Parser::c;
int main() { Parser().analyze(); }
```

(а) Восстановите грамматику G по программе. (б) Не изменяя методы $parse()$, добавьте недостающие описания в классах A и S так, чтобы правильные цепочки переводились данной программой в линейную запись соответствующего дерева вывода. Символу S в линейной записи соответствует пара скобок ' $<$ ' и ' $>$ ', обрамляющих выводимую из S цепочку, символу A соответствует пара скобок '(' и ')'. Например, для дерева



7. Дана грамматика G : $S \rightarrow Ac \mid Ab$
 $A \rightarrow Ac \mid b \mid Sa$

(а) Обоснуйте следующее утверждение: любую автоматную грамматику через детерминизацию диаграммы состояний можно преобразовать в эквивалентную праволинейную, к которой применим метод рекурсивного спуска. (б) Преобразуйте заданную грамматику G в соответствии с пунктом (а).

8. Дана КС-грамматика G_a с действиями над глобальными переменными $depth$ и $level$.

(а) Какой язык задает данная грамматика (какое множество цепочек)? (б) Построить КС-грамматику G без действий для языка $L(G_a)$, содержащую не более 4-х правил вывода, считая альтернативы: G_a : $S \rightarrow \langle depth=0; level=0; \rangle A \langle \text{if}(depth<2) \text{error}(); \rangle$
 $A \rightarrow \langle \langle level++; depth = level > depth ? level : depth; \rangle A \rangle \langle level--; \rangle A \mid \varepsilon$

9. Может ли быть неоднозначной грамматика, порождающая язык $\{\varepsilon\}$? Обоснуйте ответ.

10. Дана постфиксная запись (ПОЛИЗ) программы на модельном М-языке. (а) Восстановить по ней тело программы на М-языке. (б) Оптимизируйте эту запись по длине, то есть постройте эквивалентную (выдающую тот же результат) запись минимальной длины, считая, что вводимое значение b всегда положительное. **Замечание.** Операции mod (остаток от деления) в М-языке нет, но есть деление с отбрасыванием остатка (/), сложение, умножение, вычитание и сравнения. Сокращенный условный оператор if-then без else есть.

ПО-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ЛИЗ	&a	7	:=	&b	read	a	b	<>	32	!F	a	b	<	25	!F	&c	b

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
:=	&b	a	:=	&a	c	:=	&a	a	b	-	:=	6	!	a	write	

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(оптим.)																	

18	19	20	21	22	23	24	25	26	27	28	29	30	31

Экзамен, 2023 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. Неразрешимый вызов – 'g (b, 2) ;'. Будет выдано: **g(char, int) -> g(char, int).**

Критерии. За каждое неверное вычеркивание или неправильно вызванную функцию: **-5.**

2. Ответ. АТД называют тип данных с полностью скрытой (инкапсулированной) структурой, работа с переменными такого типа происходит только через специальные, предназначенные для этого функции. В C++ АТД реализуется с помощью классов (структур), в которых нет открытых членов-данных. **Пример** абстрактного типа данных, представленного классом A:

```
class A { int a; public: void set_a (int n); int get_a() const { return a; } };
void A::set_a (int n) { a = n; }
int main () { A obj1, obj1.set_a(5); cout<< obj1.get_a() << endl; return 0; }
```

Критерии. Неверное определение: **=0.** За отсутствие/неверный пример: **-5.**

3. Ответ. Ошибки – неоднозначный выбор имен x и y, неразрешимый вызов функции f().
Исправления – N1::y=N1::x+N2::f()+g(1); // возможны варианты.

Критерии. За каждую неверно найденную или неисправленную ошибку: **-5.**

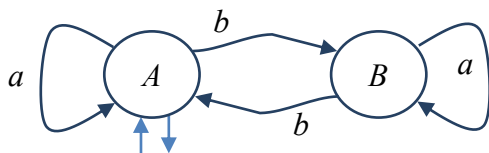
4. Пример решения:

```
template<class T> T F(T& ob1, T& ob2) { T k(ob1); ob1=ob2; ob2=1; return k; }
class M {public: int r; M () {r=0;} M (int a) {r=a;} };
```

Критерии. Если написано больше одной функции: **-10**, если не использована шаблонная функция: **-10**, за ошибки в описании шаблона (нет параметров, слово 'template' не там, не использованы слова 'class' или 'typename', не правильно использованы параметры шаблона, ...): по **-5** за каждую ошибку, за отсутствие конструктора преобразования в классе **-5**, за остальные программистские ошибки: по **-2.**

5. Ответ.

(а) $\mathcal{A}$:



(б) $L(\mathcal{A}) = \{x \in \{a,b\}^* \mid |x|_b \bmod 2 = 0\}$, то есть множество всех цепочек в алфавите $\{a,b\}$ с четным количеством символов 'b'.

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (а), (б): по **-5.**

6. Ответ. (а) $S \rightarrow bAd \mid d \quad A \rightarrow aA \mid \varepsilon$

(б) Добавить в открытую часть класса S: $S() \{ \text{cout} << '('; \sim S() \{ \text{cout} << ')'; \}$, добавить в открытую часть класса A: $A() \{ \text{cout} << '<'; \sim A() \{ \text{cout} << '>'; \}$.

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (а), (б): по **-5.**

7. Ответ. (а) По любой автоматной грамматике можно построить эквивалентный конечный автомат. По любому конечному автомату можно построить детерминированный конечный автомат. По детерминированному конечному автомату всегда можно построить праволинейную автоматную грамматику (возможно с пустыми правыми частями). В такой грамматике пункт (1) критерия применимости выполняется в силу детерминированности: непустые альтернативы начинаются с различных терминалов, то есть множества $first()$ у альтернатив не пересекаются. Пункт (2) выполняется, поскольку непустые альтернативы обязательно содержат один терминальный символ и из них нельзя вывести еще одну пустую цепочку. Пункт (3) выполняется, поскольку в любой сентенциальной форме имеется не более одного нетерминала в крайней правой позиции, а значит множество $follow()$ для любого нетерминала пусто, и его пересечение с любым множеством тоже пусто. Более лаконичные, но по сути верные ответы тоже принимаются.

(б) $H \rightarrow cA \quad (H - \text{начальный символ}) \quad A \rightarrow aA \mid bS_{\{A,S\}} \quad S_{\{A,S\}} \rightarrow aS_{\{A,S\}} \mid bS_{\{A,S\}} \mid \varepsilon$

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (а), (б): до **-5**, за каждое неправильное преобразование в цепочке $G_L \Rightarrow KA \Rightarrow ДКА \Rightarrow G_R$ по **-2**. За добавление вместо пустой альтернативы маркера конца $\perp$ оценка не снижается, но внимание студента обращается на тот факт, что грамматика становится не эквивалентной исходной.

8. Ответ. (а) Язык скобочных систем с глубиной вложенности не более 2.

(б) $S \rightarrow (B)S \mid B$
 $B \rightarrow () B \mid \varepsilon$

Критерии. За неправильный/отсутствующий ответ на пункт (а): **-5**, за каждое лишнее правило (альтернативу): **-2**.

9. Ответ. Нет, так как не существует цепочки в двумя различными правыми выводами (вообще ни одного вывода не существует). Или так: не существует цепочки с двумя различными деревьями выводов (вообще ни одного дерева не существует).

Критерии. За неверный ответ: **-10**, за отсутствие обоснования: **-5**.

10. Ответ. (а) $a:=5$; read(b); while $a < b$ do if $a > b$ then $a:=a-b$ else $b:=b-a$; write (a)

Программа выдает значение НОД(5, b), то есть 5, если b делится на 5 и 1 – иначе (так как 5 – простое число).

(б) Этого же результата можно достичь фрагментом if $b/5*5 < b$ then write(1) else write(5). Его постфиксная запись:

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(оптим.)	b	5	/	5	*	b	<	14	!F	1	write	16	!	5	write		
	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
																	}

Критерии: За неправильный/отсутствующий ответ на каждый из пунктов (а), (б) снижение на **-5**.

Экзамен, 2023 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач – 10 баллов (всего 100 баллов)

1. Ответ. Неразрешимый вызов – 'g(2,b);'. Будет выдано: **g(char,int)→g(B&,double)**.

Критерии. За каждое неверное вычеркивание или неправильно вызванную функцию: **-5**.

2. Ответ. Функция-друг класса – внешняя по отношению к классу функция, которой разрешен доступ к закрытым и защищенным членам класса. **Пример:**

```
class A { int a; friend void f(A& r); void f(A& r){r.a=1;} int main(){A a; f(a);}
```

Критерии. Неверное определение: **=0**, за отсутствие/неверный пример: **-5**.

3. Ответ. Ошибки – неоднозначный выбор имени x и неразрешимый вызов функции f().

Исправления – `y = N1::x + N2::f() + g(3); // возможны варианты.`

Критерии. За каждую неверно найденную или неисправленную ошибку: **-5**.

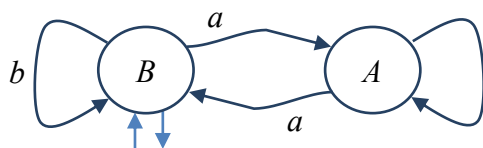
4. Ответ. **Пример решения:**

```
template <class T> int g(T ob1, T ob2) { if (ob1>ob2) return 1; else return 0;}
int g(const char*ob1,const char*ob2){if(strcmp(ob1,ob2)>=1) return 1;else return 0;}
```

Критерии. Если написано больше двух функций: **-10**, если не использована шаблонная функция: **-10**, за ошибки в описании шаблона (нет параметров, слово 'template' не там, не использованы слова 'class' или 'typename', не правильно использованы параметры шаблона, ...): по **-5** за каждую ошибку, за отсутствие конструктора преобразования в классе **-5**, за остальные программистские ошибки: по **-2**.

5. Ответ.

(a) $\mathcal{A}$:



(б) $L(\mathcal{A}) = \{x \in \{a,b\}^* \mid |x|_a \bmod 2 = 0\}$, то есть множество всех цепочек в алфавите $\{a,b\}$ с четным количеством символов 'a'.

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (a), (б): по **-5**.

6. Ответ. (a) $S \rightarrow aS \mid A \quad A \rightarrow bAd \mid c$

(б) Добавить в открытую часть класса S : $\underline{S() \{cout << '<';\} \sim S() \{cout << '>';\}}}$, добавить в открытую часть класса A : $\underline{A() \{cout << '(<';\} \sim A() \{cout << '>';\}}}$.

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (a), (б): по **-5**.

7. Ответ. (a) По любой автоматной грамматике можно построить эквивалентный конечный автомат. По любому конечному автомату можно построить детерминированный конечный автомат. По детерминированному конечному автомату всегда можно построить праволинейную автоматную грамматику (возможно с пустыми правыми частями). В такой грамматике пункт (1) критерия применимости выполняется в силу детерминированности: непустые альтернативы начинаются с различных терминалов, то есть множества $first()$ у альтернатив не пересекаются. Пункт (2) выполняется, поскольку непустые альтернативы обязательно содержат один терминальный символ и из них нельзя вывести еще одну пустую цепочку. Пункт (3) выполняется, поскольку в любой сентенциальной форме имеется не более одного нетерминала в крайней правой позиции, а значит множество $follow()$ для любого нетерминала пусто, и его пересечение с любым множеством тоже пусто. Более лаконичные, но по сути верные ответы тоже принимаются.

(б) $H \rightarrow bA$ (H – начальный символ) $A \rightarrow bS \mid cS_{\{A,S\}} \quad S_{\{A,S\}} \rightarrow cS_{\{A,S\}} \mid aA \mid bS \mid \varepsilon \quad S \rightarrow aA \mid \varepsilon$

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (a), (б): до **-5**, за каждое неправильное преобразование в цепочке $G_L \Rightarrow KA \Rightarrow ДКА \Rightarrow G_R$ по **-2**. За добавление вместо пустой альтернативы маркера конца $\perp$ оценка не снижается, но внимание студента обращается на тот факт, что грамматика становится не эквивалентной исходной.

8. Ответ. (а) Язык скобочных систем с глубиной вложенности не менее 2.

$$\begin{aligned} (б) S &\rightarrow ()S \mid ((B))B \\ B &\rightarrow (B)B \mid \varepsilon \end{aligned}$$

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (а), (б): по -5, за каждое лишнее правило (альтернативу): -2.

9. Ответ. Да: $S \rightarrow SS \mid \varepsilon$. Для цепочки ε существует бесконечно много различных деревьев вывода.

Критерии. За неверный ответ: -10. За отсутствие обоснования: -5.

10. Ответ. (а) `a:=7 ; read(b); while a<>b do
begin if a<b then begin c:=b; b:=a; a:=c end; a:=a-b end; write(a)`

Программа выдает значение НОД(7, b), то есть 7, если b делится на 7 и 1 – иначе (так как 7 – простое число).

(б) Этого же результата можно достичь фрагментом `if b/7*7 < b then write(1) else write(7)`. Его постфиксная запись:

ПОЛИЗ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(оптим.)	b	7	/	7	*	b	<	14	!F	1	write	16	!	7	write		
18	19	20	21	22	23	24	25	26	27	28	29	30	31				

Критерии. За неправильный/отсутствующий ответ на каждый из пунктов (а), (б): снижение на -5.

1	2

1. Дан класс *Scanner*, моделирующий автомат $\mathcal{U}$ с множеством состояний $\{A, B\}$. Метод *accept()* допускает цепочки языка $L(\mathcal{U})$.

```
#include<iostream>
#include<typeinfo>

using namespace std;

class Scanner {
    class State { public:
        virtual void operator()(char c) const =0;
    };

    class A: public State { public:
        void operator()(char c) const {
            if (cin.eof()) throw true; //допуск
            if (c=='b') throw stB;
            else if (c=='a') throw stA;
            else throw false;
        }
    };

    class B: public State { public:
        void operator()(char c) const {
            if (cin.eof()) throw false;
            if (c=='a') throw stA;
            else throw false;
        }
    };

    static A stA; // состояние A
    static B stB; // состояние B

    void next(State&s){//текущее состояние в s
        try { char c=cin.get();
            // текущий символ в c

            s(c); // переход из s по c
        }
        catch (A&) { next(stA); }
        catch (B&) { next(stB); }
    }

public:
    bool accept(){ try{ next(stA);}
        catch(bool b) { return b; }
        return false;
    }
}; //end of Scanner

Scanner::A Scanner::stA;
Scanner::B Scanner::stB;
```

(а) Опишите функцию *main()*, анализирующую входную цепочку с помощью средств класса *Scanner*. *Ответ:*

(б) Постройте конечный автомат $\mathcal{U}$ в виде ДС, указав начальное и заключительные состояния. *Ответ:*

(в) Определите язык $L(\mathcal{U})$ формулой или словесно. *Ответ:*

(г) Постройте эквивалентную автомату $\mathcal{U}$ левостолбчатую грамматику G_{left} . *Ответ:*

(д) Каким недостатком обладает реализация автомата с помощью класса *Scanner* с точки зрения прагматики использования механизма исключений? *Ответ:*

(е) Добавьте в класс *Scanner* действия по печати названий состояний автомата, проходимых последовательно при допуске цепочки (полагая что метод *name()* из *typeid* возвращает имя соответствующего класса в виде строки).

2. По КС-грамматике с **одним** нетерминальным символом S построен анализатор, действующий по принципу рекурсивного спуска. Ниже дана его заготовка с одной пропущенной функцией.

```
#include <iostream>

void S();

struct alt1{ std::string s = "aSb"; };

namespace NS { struct alt2 {
    std::string s = "cc";
};

char c; // текущий символ-лексема
void gc() { std::cin>>c; }

//функция alt_analyze:

void S() {
    alt1 a1; NS::alt2 a2;
    if (c == 'a')
        alt_analyze (a1);
    else if (c == 'c')
        alt_analyze <NS::alt2>(a2);
}

int main() {
    try { gc(); S();
        if (std::cin.eof())
            std::cout << "\nOK!\n";
        else throw c;
    }
    catch (char) { std::cout<<"\nNO!\n"; }
}
```

(а) Восстановить КС-грамматику G , по которой был построен анализатор. *Ответ:*

(б) Какой язык порождает грамматика G ?
Ответ: $L(G) =$

(в) Описать недостающую функцию `alt_analyze()` в соответствии с принципом рекурсивного спуска. *Ответ:*

(г) Корректен ли получившийся анализатор? Обоснуйте ответ. *Ответ:*

(д) Если грамматика G не является неукорачивающей, то с помощью алгоритма устранения пустых правых частей привести ее к неукорачивающему виду. *Ответ:*

1	2

1. Дан класс *Scanner*, моделирующий автомат $\mathcal{U}$ с множеством состояний $\{B, C\}$. Метод *accept()* допускает цепочки языка $L(\mathcal{U})$.

```
#include<iostream>
#include<typeinfo>

using namespace std;

class Scanner {
    class State { public:
        virtual void operator()(char c) const =0;
    };

    class B: public State { public:
        void operator()(char c) const {
            if (cin.eof()) throw true; //допуск
            if (c=='c') throw &stC;
            else if (c=='b') throw &stB;
            else throw false;
        }
    };

    class C: public State { public:
        void operator()(char c) const {
            if (cin.eof()) throw false;
            if (c=='b') throw &stB;
            else throw false;
        }
    };

    static B stB; // состояние B
    static C stC; // состояние C

    void next(State *s){
        // s указывает текущее состояние
        try{ char c=cin.get();
            // текущий символ в c

            (*s)(c); // переход из s по c
        }
        catch (B* b) { s=b; }
        catch (C* c) { s=c; }
        next(s);
    }

public:
    bool accept(){ try{ next(&stB);}
        catch(bool b) {return b;}
        return false;
    }
}; //конец класса Scanner

Scanner::B Scanner::stB;
Scanner::C Scanner::stC;
```

(а) Опишите функцию *main()*, анализирующую входную цепочку с помощью средств класса *Scanner*. *Ответ:*

(б) Постройте конечный автомат $\mathcal{U}$ в виде ДС, указав начальное и заключительные состояния. *Ответ:*

(в) Определите язык $L(\mathcal{U})$ формулой или словесно. *Ответ:*

(г) Постройте эквивалентную автомату $\mathcal{U}$ праволинейную грамматику G_{right} . *Ответ:*

(д) Каким недостатком обладает реализация автомата с помощью класса *Scanner* с точки зрения прагматики использования механизма исключений? *Ответ:*

(е) Добавьте в класс *Scanner* действия по печати названий состояний автомата, проходимых последовательно при допуске цепочки (полагая что метод *name()* из *typeid* возвращает имя соответствующего класса в виде строки).

2. По КС-грамматике с **одним** нетерминальным символом S построен анализатор, действующий по принципу рекурсивного спуска. Ниже дана его заготовка с одной пропущенной функцией.

```
#include <iostream>

void S();

struct alt1{ std::string s = "aSc"; };

namespace NS { struct alt2 {
    std::string s = "cc";
};

char c; // текущий символ-лексема
void gc() { std::cin>>c; }

//функция alt_analyze:
```

(а) Восстановить КС-грамматику G , по которой был построен анализатор. *Ответ:*

(б) Какой язык порождает грамматика G ?
Ответ: $L(G) =$

(в) Описать недостающую функцию `alt_analyze()` в соответствии с принципом рекурсивного спуска. *Ответ:*

```
void S() {
    alt1 a1; NS::alt2 a2;
    if (c == 'a')
        alt_analyze (a1);
    else if (c == 'c')
        alt_analyze <NS::alt2>(a2);
}

int main() {
    try { gc(); S();
        if (std::cin.eof())
            std::cout << "\nOK!\n";
        else throw c;
    }
    catch (char) { std::cout<<"\nNO!\n"; }
}
```

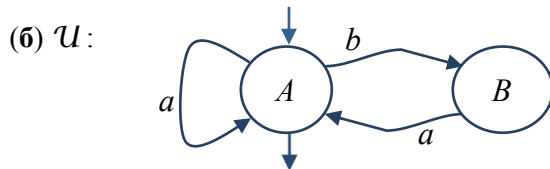
(г) Корректен ли получившийся анализатор? Обоснуйте ответ. *Ответ:*

(д) Если грамматика G не является неукорачивающей, то с помощью алгоритма устранения пустых правых частей привести ее к неукорачивающему виду. *Ответ:*

Коллоквиум, 2024 год. Ответы и критерии оценок.
Вариант 1.
Максимальная оценка каждой из задач – 50 баллов (всего 100 баллов)

1. (а) `int main(){ cout<< (Scanner().accept() ? "\nYes\n" : "\nNo\n"); }`

Критерии: 10 баллов за подходящую функцию `main()`, возможны вариации правильного ответа.



Критерии: 10 баллов за правильный автомат. Если не указаны (неверно указаны) начальное/заключительное состояния, то за этот пункт дается **5 баллов**.

(в) $L(\mathcal{U}) = \{a^i x_l \dots x_n \mid i \geq 0, n \geq 0, x_l \in \{ba^k \mid k > 0\} \text{ для } 1 \leq l \leq n\}$. Все цепочки в алфавите $\{a, b\}$, в которых за каждым символом b непосредственно следует a . Можно регулярным выражением: $(a+ba)^*$. Или так: $L(\mathcal{U}) = \{a^{i_0} ba^{i_1} \dots ba^{i_k} \mid k \geq 0, i_0 \geq 0, i_j > 0 \text{ для } j > 0\}$.

Критерии: 5 баллов за правильное описание языка (любым способом) построенного автомата.

(г) $G_{left} = \langle \{a, b\}, \{A, B\}, \{A \rightarrow Aa \mid Ba \mid \varepsilon; B \rightarrow Ab\}, A \rangle$.

Критерии: 10 баллов за верную грамматику (верное множество правил вывода), эквивалентную автомату, построенному в качестве ответа в пункте (б).

(д) Механизм исключений в классе `Scanner` встроен в основной алгоритм обхода состояний автомата, в то время как прагматика исключений предполагает их использование в ошибочных ситуациях для досрочного завершения основного алгоритма.

Критерии: 5 баллов за правильный ответ.

(е) Модифицировать метод `next()` следующим образом:

```

void next(State &s){ // текущее состояние в s
    try{ char c = cin.get(); // очередной символ в c
        cout << typeid(s).name() << ' ';
        s(c); // переход из s по c
    }
    catch (A&) { next(stA); }
    catch (B&) { next(stB); }
}
  
```

Критерии: Возможны альтернативные решения, например, с помощью динамического приведения типа `dynamic_cast`. При верном решении (любым способом) за этот пункт ставится **10 баллов**. Если вместо `'typeid'` написано `'typeidinfo'` – **7 баллов**.

2. (а) $G = \langle \{a, b, c\}, \{S\}, \{S \rightarrow aSb \mid cc \mid \varepsilon\}, S \rangle$.

Критерии: 10 баллов за правильную грамматику (множество правил грамматики).

(б) $L(G) = \{a^n (cc)^i b^n \mid n \geq 0, i \in \{0, 1\}\}$. Возможны другие вариации записи формулы.

Критерии: 10 баллов за правильное описание языка.

(в) **Ответ:**

```
template <class T>
void alt_analyze(const T & t){
    for (size_t i = 0; i < t.s.size(); i++){
        if (t.s[i] == 'S') S();
        else if (t.s[i] == c) gc();
        else throw c;
    }
}
```

Критерии: 10 баллов за правильный ответ. Возможны вариации ответа. За каждую существенную ошибку в функции: **-3**.

(г) **Да:** 1) $first(aSb) \cap first(\varepsilon) = \emptyset$, $first(cc) \cap first(\varepsilon) = \emptyset$, $first(aSb) \cap first(cc) = \emptyset$; 2) ни из 'cc', ни из 'aSb' не выводится ' ε '; 3) $first(S) \cap follow(S) = \emptyset \Rightarrow$ выполняется критерий применимости.

Критерии: 10 баллов за правильный и обоснованный ответ.

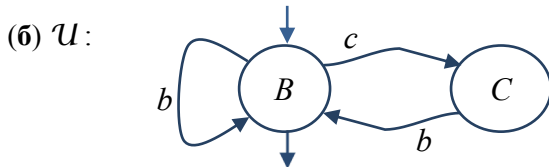
(д) $S' \rightarrow S \mid \varepsilon$ $S \rightarrow aSb \mid ab \mid cc$

Критерии: 10 баллов за правильный ответ.

Коллоквиум, 2024 год. Ответы и критерии оценок.
Вариант 2.
Максимальная оценка каждой из задач – 50 баллов (всего 100 баллов)

1. (а) `int main(){ cout<< (Scanner().accept() ? "\nYes\n" : "\nNo\n"); }`

Критерии: 10 баллов за подходящую функцию `main()`, возможны вариации правильного ответа.



Критерии: 10 баллов за правильный автомат. Если не указаны (неверно указаны) начальное/заключительное состояния, то за этот пункт дается **5 баллов**.

(в) $L(\mathcal{U}) = \{b^i x_l \dots x_n \mid i \geq 0, n \geq 0, x_l \in \{cb^k \mid k > 0\} \text{ для } 1 \leq l \leq n\}$. Все цепочки в алфавите $\{b, c\}$, в которых за каждым символом c непосредственно следует b . Можно регулярным выражением: $(b + cb)^*$. Или так: $L(\mathcal{U}) = \{b^{i_0} cb^{i_1} \dots cb^{i_k} \mid k \geq 0, i_0 \geq 0, i_j > 0 \text{ для } j > 0\}$.

Критерии: 5 баллов за правильное описание языка (любым способом) построенного автомата.

(г) $G_{right} = \langle \{b, c\}, \{B, C\}, \{B \rightarrow bB \mid cC \mid \varepsilon; C \rightarrow bB\}, B \rangle$.

Критерии: 10 баллов за верную грамматику (верное множество правил вывода), эквивалентную автомату, построенному в качестве ответа в пункте (б).

(д) Механизм исключений в классе `Scanner` встроен в основной алгоритм обхода состояний автомата, в то время как прагматика исключений предполагает их использование в ошибочных ситуациях для досрочного завершения основного алгоритма.

Критерии: 5 баллов за правильный ответ.

(е) Модифицировать метод `next()` следующим образом:

```

void next(State *s){ // s указывает текущее состояние
    try{ char c = cin.get(); // текущий символ в c
        cout << typeid(*s).name() << ' ';
        (*s)(c); // переход из s по c
    }
    catch (B* b) { s = b; }
    catch (C* c) { s = c; }
    next(s);
}
  
```

Критерии: Возможны альтернативные решения, например, с помощью динамического приведения типа `dynamic_cast`. При верном решении (любым способом) за этот пункт ставится **10 баллов**. Если вместо `'typeid'` написано `'typeidinfo'` – **7 баллов**.

2. (а) $G = \langle \{a, c\}, \{S\}, \{S \rightarrow aSc \mid cc \mid \varepsilon\}, S \rangle$.

Критерии: 10 баллов за правильную грамматику (множество правил грамматики).

(б) $L(G) = \{a^n (cc)^i c^n \mid n \geq 0, i \in \{0, 1\}\}$. Возможны другие вариации записи формулы.

Критерии: 10 баллов за правильное описание языка.

(в) **Ответ.**

```
template <class T>
void alt_analyze(const T & t){
    for (size_t i = 0; i < t.s.size(); i++){
        if (t.s[i] == 'S') S();
        else if (t.s[i] == c) gc();
        else throw c;
    }
}
```

Критерии: 10 баллов за правильный ответ. Возможны вариации ответа. За каждую существенную ошибку в функции: **-3**.

(г) *Нем*: $first(S) \cap follow(S) \neq \emptyset \Rightarrow$ нарушается пункт (3) критерия применимости.

Критерии: 10 баллов за правильный и обоснованный ответ.

(г) $S' \rightarrow S \mid \varepsilon \qquad S \rightarrow aSc \mid ac \mid cc$

Критерии: 10 баллов за правильный ответ.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8

1. (10 баллов) Что напечатается в результате работы следующей программы?

```

struct A { int n;
    A(int k = 0) { n = k; cout << "1 "; }
    A(const A&) { cout << "2 "; }
    virtual ~A() { cout << "3 "; }
};

void func (int x, char*y) { cout << "::func(int, char*)\n"; }
void func (const char*a, double b) { cout << "::func(const char*, double)\n"; }

namespace N {
    void func (const A& a, int i) { cout << "N::func(A, int)\n "; }
}

void func (A & a, B & b) {
    try { if ( typeid (a) == typeid (b)) throw a;
        else throw b; }
    catch (B&) { cout << "catch(B&)\n"; }
    catch (A&) { cout << "catch(A&)\n"; }
}

int main() { A a; B b;
    func(a, b);
    func (0, 1);
    using namespace N;
    func (b, 5);
}

```

```

struct B : A {
    B() { cout << "4 "; }
    B(const B&) { cout << "5 "; }
    ~B() { cout << "6 "; }
};

```

2. (10 баллов) (а) Приведенная ниже программа не компилируется. Объясните **причину** ошибки и исправьте **только** структуру `D`, ничего **не удаляя** как-либо. (б) Что будет напечатано в результате работы получившейся программы?

```

class B { public:
    virtual void f11() {
        cout << "B::f11\n"; }
    void f22() {
        f11(); cout << "B::f22\n"; }
    virtual void f33 () = 0;
};

void f (B & b) { b.f11(); }

int main () { D d; B * pb = &d;
    pb -> f22();
    f(d);
}

```

```

struct D : B {
    void f11() {
        cout << "D::f11\n"; }
    virtual void f22() {
        f11(); cout << "D::f22\n"; }
};

```

3. (20 баллов) (а) Приведенная ниже программа не компилируется. Объясните **причины** ошибок и модифицируйте **только** функцию `MAX`, ничего **не удаляя** как-либо. (б) Что будет напечатано в результате работы получившейся программы? (в) Приведите одну альтернативную реализацию функции `MAX`, не нарушающую правильность программы в целом, выполняющую те же вычисления, но **не** использующую служебное слово `auto`.

```

auto MAX (std::vector <auto> & arr) { // arr – непустой массив элементов некоторого типа
    auto iter = arr.begin(), iter_of_max = iter;
    for (iter++; iter != arr.end(); iter++)
        if (*iter_of_max < *iter)
            iter_of_max = iter;
    return *iter_of_max;
}

int main() {
    std::vector <int> vi {1, 2, 3};
    std::vector <double> vd = {4, 10, 1};
    MAX(vd) = MAX (vi);
    std::cout << MAX (vd) << std::endl;
}

```

4. (10 баллов) (а) Приведенная ниже программа не компилируется. Устраните все ошибки **двумя** разными способами, исправляя **только** класс *A* и ничего **не удаляя** как-либо.
 (б) Что будет напечатано в результате работы **каждого** варианта получившейся программы?

```
struct A { double d;
          int n;
          A (double a = 8.5) { d = n = a ; }
          A ( A && a) { d = a.d; }
          int f () { return n; }
};
int main() {
    const A a = A(1.1), b;
    cout << a.f() << " " << a.d << " " << b.n << endl;
}
```

5. (10 баллов) Существует ли диаграмма состояний с единственной вершиной (состоянием) для конечного автомата, не являющегося детерминированным? Ответ обосновать.

6. (10 баллов) Дана КС-грамматика G :
- $$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow Aa \mid a \\ B &\rightarrow Ba \mid b \end{aligned}$$

- (а) Обоснуйте неприменимость метода рекурсивного спуска для G . (б) Опишите язык $L(G)$ формулой. (в) Постройте эквивалентную праволинейную грамматику, для которой рекурсивный спуск применим (если это возможно).

7. (20 баллов) Даны правила грамматики: $S \rightarrow ASB \mid AXB$

$$\begin{aligned} A &\rightarrow \text{система} \\ B &\rightarrow \text{программирования} \end{aligned}$$

- (а) Вставьте одно пропущенное правило так, чтобы из S выводилась ровно одна терминальная цепочка из строчных русских букв с символом пробела внутри. (б) Определите тип (по Хомскому) получившейся грамматики. (в) Дайте определение термина из лекционного курса, задаваемого цепочкой из пункта (а).

8. (10 баллов) (а) Построить КС-грамматику скобочных систем в алфавите прямоугольных скобок $\{ [,] \}$, содержащую два правила (считая альтернативы). (б) Добавить в построенную грамматику действия, чтобы по произвольной скобочной системе с количеством скобок $2n$ печаталась система вида $[^n]^n$, $n \geq 0$.

Примеры: $\varepsilon \Rightarrow \varepsilon$; $[] \Rightarrow []$; $[][] \Rightarrow [[]]$; $[][][] \Rightarrow [[[]]]$; $[][][] \Rightarrow [[[]]]$.

NB! Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы `using namespace std`.

1	2	3	4	5	6	7	8

1. (10 баллов) Что напечатается в результате работы следующей программы?

```

struct A { int n;
  A(int k = 0) {n = k; cout << "1 "; }
  A(const A&) { cout << "2 "; }
  virtual ~A() { cout << "3 "; }
};

void func (int x, char*y) { cout << "::func(int, char*)\n"; }
void func (const char*a, double b) { cout << "::func(const char*, double)\n"; }
namespace N {
  void func (const A & a, int i) { cout << "N::func(A, int)\n "; }
}

void func (A & a, B & b) {
  try { if ( typeid (a) == typeid (b)) throw a;
        else throw b; }
  catch (B&) { cout << "catch(B&)\n"; }
  catch (A&) { cout << "catch(A&)\n"; }
}

int main() { A a; B b;
            func (b, b);
            func (1, 0);
            using namespace N;
            func (1, 2);
            }

```

```

struct B : A {
  B() { cout << "4 "; }
  B(const B&) { cout << "5 "; }
  ~B() { cout << "6 "; }
};

```

2. (10 баллов) (а) Приведенная ниже программа не компилируется. Объясните **причину** ошибки и исправьте **только** класс *B*, ничего **не удаляя** как-либо. (б) Что будет напечатано в результате работы получившейся программы?

```

struct A {
  virtual void g1() {
    cout << "A::g1\n";
  }
  void g2() {
    g1(); cout << "A::g2\n";
  }
  virtual void g3() = 0;
};

void f (A & a) { a.g1(); }
int main () { B b; A * pa = &b;
             pa -> g2();
             f (b);
             }

```

```

class B : public A {
  void g1() {
    cout << "B::g1\n";
  }
  virtual void g2() {
    g1(); cout << "B::g2\n";
  }
};

```

3. (20 баллов) (а) Приведенная ниже программа не компилируется. Объясните **причины** ошибок и модифицируйте **только** функцию *MIN*, ничего **не удаляя** как-либо. (б) Что будет напечатано в результате работы получившейся программы? (в) Приведите одну альтернативную реализацию функции *MIN*, не нарушающую правильность программы в целом, выполняющую те же вычисления, но не использующую служебное слово *auto*.

```

auto MIN (std::list<auto> & lst) { // lst – непустой список элементов некоторого типа
  auto iter = lst.begin(), iter_of_min = iter;
  for (iter++; iter != lst.end(); iter++)
    if (*iter_of_min > *iter)
      iter_of_min = iter;
  return *iter_of_min;
}

int main() {
  std::list<int> Li{1,2,3};
  std::list<double> Ld = {-4, -10, -5};
  MIN(Ld) = MIN (Li);
  std::cout << MIN (Ld) << std::endl;
}

```


4. (10 баллов) (а) Приведенная ниже программа не компилируется. Устраните все ошибки **двумя** разными способами, исправляя **только** класс S и ничего **не удаляя** как-либо.

(б) Что будет напечатано в результате работы **каждого** варианта получившейся программы?

```
struct S { double d;
          int n;
          S (double a = 3.5) { d = n = a; }
          S (S && a) { d = a.d; }
          int f() { return n; }
};
int main() {
    const S s = S(7.3), t;
    cout << s.f() << " " << s.d << " " << t.n << endl; }
```

5. (10 баллов) Существует ли конечный автомат с двумя состояниями, детерминизация которого дает автомат с пятью состояниями? Ответ обосновать.

6. (10 баллов) Дана КС-грамматика G :

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow aA \mid a \\ B &\rightarrow aB \mid b \end{aligned}$$

(а) Обоснуйте неприменимость метода рекурсивного спуска для G . (б) Опишите язык $L(G)$ формулой. (в) Постройте эквивалентную праволинейную грамматику, для которой рекурсивный спуск применим (если это возможно).

7. (20 баллов) Даны правила грамматики:

$$S \rightarrow CSD \mid CYD$$

$$\begin{aligned} C &\rightarrow \text{программный} \\ D &\rightarrow \text{продукт} \end{aligned}$$

(а) Вставьте одно пропущенное правило так, чтобы из S выводилась ровно одна терминальная цепочка из строчных русских букв с символом пробела внутри. (б) Определите тип (по Хомскому) получившейся грамматики. (в) Дайте определение термина из лекционного курса, задаваемого цепочкой из пункта (а).

8. (10 баллов) (а) Построить КС-грамматику скобочных систем в алфавите круглых скобок $\{ (,) \}$, содержащую два правила (считая альтернативы). (б) Добавить в построенную грамматику действия, чтобы по произвольной скобочной системе с количеством скобок $2n$ печаталась система вида $(^n)^n$, $n \geq 0$.

Примеры: $\varepsilon \Rightarrow \varepsilon$; $() \Rightarrow ()$; $()() \Rightarrow (())$; $((())) \Rightarrow (((())))$; $((())()) \Rightarrow (((())))$.

Экзамен, 2024 год. Ответы и критерии оценок.

Вариант 1.

Максимальная оценка всех задач в совокупности 100 баллов

1. (10 баллов) Будет выдано:

1 1 4 1 5 catch(B&) -6 3 ::func(const char*,double) -N::func (A,int) -6 3 3

Критерии: Неверная печать/порядок печати -2 за каждую ошибку, если нет конструктора, за парный деструктор не снижать.

2. (10 баллов) (а) класс D абстрактный, нельзя создавать объекты. Исправление: описать в классе D , например, такой метод `void f33() {}`, (б) будет выдано:

$D::f11()$ - $B::f22()$ - $D::f11$ (в столбик)

Критерии: Не объяснена, не найдена, неверно найдена ошибка -5, неверная выдача -3 за каждую, найдены несуществующие ошибки -3 за каждую.

3. (20 баллов)

(а) Результат работы функции не *l-value*, она должна возвращать ссылку (*auto & ...*), (б) 4,

(в) `template <class T>`

```
T & MAX (std::vector<T> & arr){
    typename std::vector<T> :: iterator iter = arr.begin(), iter_of_max = iter;
    for (iter++; iter != arr.end(); iter++)
        if (*iter_of_max < *iter)
            iter_of_max = iter;
    return *iter_of_max;
}
```

Критерии: Не найдена, неверно объяснена ошибка -8, Отсутствует, неверный выписан шаблон функции -12, Нет слова 'typename' при верном в остальном шаблоне -4, лишние найденные ошибки, другие ошибки, неверная выдача -3 за каждую.

4. (10 баллов)

(а) 1 вариант:

```
struct A {
    double d;
    static int n;
    A(double a = 8.5){ d = n = a; }
    A (A && a) { d = a.d; }
    static int f() { return n; }
};
int A::n;
```

2 вариант:

```
struct A {
    double d;
    int n;
    A(double a = 8.5){ d = n = a; }
    A (A && a) { d = a.d; }
    int f() const { return n; }
};
```

(б) 1 вариант: - 8 1 8

2 вариант: - 1 1 8

Критерии: Не написан/неверно написан вариант исправления -5 за каждый, только неверная выдача -2 за каждую, в варианте со статической переменной не отводится память вне класса -3.

5. (10 баллов) *Нет*, так как: (1) не может быть двух начальных состояний и (2) по любому символу можно перейти только в то же состояние $\Rightarrow$ нет различных дуг с одинаковой пометкой.

Критерии: за отсутствие каждого из пунктов (1, 2) в обосновании по -5.

6. (10 баллов)

(а) $first(Aa) \cap first(a) = \{a\} \neq \emptyset$ или $first(Ba) \cap first(b) = \{b\} \neq \emptyset \Rightarrow$ в любом случае нарушается критерий применимости. (б) $L(G) = \{a^n \mid n \geq 1\} \cup \{ba^k \mid k \geq 0\}$ (возможны вариации ответа). Соответствующее регулярное выражение: $(a + b) a^*$. (в) Для регулярного языка можно построить ДКА, а по нему – праволинейную автоматную грамматику. К такой грамматике рекурсивный спуск всегда применим (см. задачу 7а из экзамена 2023 года): $S \rightarrow aA \mid bA \quad A \rightarrow aA \mid \varepsilon$

Критерии: за отсутствие/ошибку в обосновании неприменимости: -3, за отсутствие/ошибку в формуле: -3, за отсутствие/ошибку в грамматике: -4.

7. (20 баллов)

(а) Вставить правило $X \rightarrow$ (или: $S \rightarrow$ система программирования).

(б) Тип 1 (контекстно-зависимая).

(в) Система программирования – набор инструментальных средств для поддержки программного продукта в течение всего его жизненного цикла.

Критерии: за неверно вставленное правило (удалять правила нельзя): -5, за неверный тип: -5, за отсутствие/ошибку в определении: -10.

8. (10 баллов)

(а) $S \rightarrow [S]S \mid \varepsilon$

(б) решение в виде простой схемы ' $S \rightarrow [\langle [\rangle S] S \langle \rangle] \mid \varepsilon$ ' не всегда дает правильный перевод, например система '[[[]]]' перейдет в '[[[]]]', а не в '[[[]]]'.

Пусть есть изначально пустой глобальный стек: `stack<char> st`. Тогда нужный перевод (в процессе рекурсивного спуска) можно задать так:

```
S → [ { cout << '['; st.push(''); } S ] S | ε { if (cin.eof())
                                         while (!st.empty())
                                           { cout << st.top(); st.pop();
                                           }
                                         }
```

Критерии: за отсутствие/ошибку в грамматике: -5, за отсутствие/ошибку в действиях: -5.

Экзамен, 2024 год. Ответы и критерии оценок.

Вариант 2.

Максимальная оценка всех задач в совокупности 100 баллов

1. (10 баллов) Будет выдано:

1 1 4 2 catch(A&) -3 ::func (int, char*) -1 N::func (A, int) -3 6 3 3

Критерии: Неверная печать/порядок печати -2 за каждую ошибку, если нет конструктора, за парный деструктор не снижать.

2. (10 баллов) (а) класс *B* абстрактный, нельзя создавать объекты. Исправление: описать в классе *B*, например, такой метод `void g3() {}`, (б) будет выдано:

B::g1() - *A::g2()* - *B::g1* (в столбик)

Критерии: Не объяснена, не найдена, неверно найдена ошибка -5, неверная выдача -3 за каждую, найдены несуществующие ошибки -3 за каждую.

3. (20 баллов)

(а) Результат работы функции не *l-value*, она должна возвращать ссылку (*auto & ...*), (б) -5,

(в) `template <class T>`

```
T & MIN (std::list <T> & lst){
    typename std::list <T> :: iterator iter = lst.begin(), iter_of_min = iter;
    for (iter++; iter != lst.end(); iter++)
        if (*iter_of_min > *iter)
            iter_of_min = iter;
    return *iter_of_min;
}
```

Критерии: Не найдена, неверно объяснена ошибка -8, Отсутствует, неверный выписан шаблон функции -12, Нет слова 'typename' при верном в остальном шаблоне -4, лишние найденные ошибки, другие ошибки, неверная выдача -3 за каждую.

4. (10 баллов)

(а) 1 вариант:

```
struct S {
    double d;
    static int n;
    S (double a = 3.5) { d = n = a; }
    S (S && a) { d = a.d; }
    static int f() { return n; }
};
```

`int S::n;`

2 вариант:

```
struct S {
    double d;
    int n;
    S (double a = 3.5) { d = n = a; }
    S (S && a) { d = a.d; }
    int f() const { return n; }
};
```

(б) 1 вариант: - 3 7 3

2 вариант: - 7 7 3

Критерии: Не написан/неверно написан вариант исправления -5 за каждый, только неверная выдача -2 за каждую, в варианте со статической переменной не отводится память вне класса -3.

5. (10 баллов) *Нет*, так как состояниями ДКА являются подмножества исходного КА. Для множества из двух состояний максимальное число подмножеств равно четырем.

Критерии: за отсутствие/ неверное обоснование: -10.

6. (10 баллов)

(а) $first(A) \cap first(B) = \{a\} \neq \emptyset$ или $first(aA) \cap first(a) = \{a\} \neq \emptyset \Rightarrow$ нарушается критерий применимости; (б) $L(G) = \{a^n \mid n \geq 1\} \cup \{a^k b \mid k \geq 0\}$ (возможны вариации ответа). Соответствующее регулярное выражение: $a^* (a + b)$. (в) Для регулярного языка можно построить ДКА, а по нему – праволинейную автоматную грамматику. К такой грамматике рекурсивный спуск всегда применим (см. задачу 7а из экзамена 2023 года): $S \rightarrow aA \mid b$ $A \rightarrow aA \mid b \mid \varepsilon$

Критерии: за отсутствие/ошибку в обосновании неприменимости: -3, за отсутствие/ошибку в формуле: -3, за отсутствие/ошибку в грамматике: -4.

7. (20 баллов)

(а) Вставить правило $Y \rightarrow$ (или: $S \rightarrow$ *программный продукт*).

(б) Тип 1 (контекстно-зависимая).

(в) Программный продукт – программа, которую можно использовать отчужденно от ее автора.

Критерии: за неверно вставленное правило (удалять правила нельзя): -5, за неверный тип: -5, за отсутствие/ошибку в определении: -10.

8. (10 баллов)

(а) $S \rightarrow (S)S \mid \varepsilon$

(б) решение в виде простой схемы ' $S \rightarrow (\langle \rangle S) S \langle \rangle \mid \varepsilon$ ' не всегда дает правильный перевод, например система '()'() перейдет в '()()', а не в '((()))'.

Пусть есть изначально пустой глобальный стек: `stack<char> st`. Тогда нужный перевод (в процессе рекурсивного спуска) можно задать так:

```
S → ( { cout<<'('; st.push(''); } S ) S | ε { if (cin.eof())
                                     while (!st.empty())
                                     { cout<<st.top(); st.pop();
                                     }
                                     }
```

Критерии: за отсутствие/ошибку в грамматике: -5, за отсутствие/ошибку в действиях: -5.

Тематический указатель

Настоящий раздел предназначен для преподавателей, ведущих текущие занятия со студентами, изучающими основы объектно-ориентированного программирования, язык программирования Си++, методы формального описания языков, а также основы теории трансляции. Этот раздел также может использоваться студентами, ведущими самостоятельную подготовку по указанным темам.

Каждый подраздел, относящийся к той или иной теме, включает себя таблицу, пользуясь данными из которой в основных разделах пособия несложно найти текст задачи.

Каждая запись в таблице имеет следующую структуру:

год, когда задача давалась студентам;
вид испытания, на котором задача давалась студентам:
К – коллоквиум;
Э – экзамен;
номер варианта;
номер задачи.

Парадигма объектно-ориентированного программирования

2009-К-2-8	2010-К-1-9	2010-К-3-9	2010-К-4-9
2011-К-3-6	2011-К-4-9	2012-К-3-1	2015-К-1-3
2015-К-1-4	2015-К-2-4	2015-К-3-6	2015-К-4-6
2015-К-4-7	2016-Э-1-8	2016-Э-2-8	

Новые стандарты языка Си++

2014-К-1-3	2014-К-2-3	2014-К-3-3	2015-К-1-10
2015-К-2-10	2015-К-3-3	2015-К-4-3	2016-К-1-9
2016-К-2-9	2017-К-1-10	2017-К-2-10	

Абстрактные типы данных (АТД). Классы. Конструкторы и деструкторы

2008-К-1-3	2008-К-1-4	2008-К-1-9	2008-К-2-3
2008-К-2-4	2008-Э-1-2	2009-К-1-4	2009-К-1-6
2009-К-2-6	2009-К-2-8	2009-К-3-7	2009-К-4-6
2009-К-4-7	2010-К-1-1	2010-К-1-8	2010-К-1-9
2010-К-2-1	2010-К-2-6	2010-К-2-7	2010-К-2-8
2010-К-3-1	2010-К-3-6	2010-К-3-7	2010-К-3-8
2010-К-4-1	2010-К-4-7	2010-К-4-8	2010-Э-2-1
2010-Э-5-6	2011-К-1-4	2011-К-2-4	2011-К-2-9
2011-К-3-4	2011-К-4-4	2012-К-1-1	2012-К-1-2
2012-К-1-3	2012-К-1-4	2012-К-1-10	2012-К-2-2
2012-К-2-3	2012-К-2-4	2012-К-2-10	2012-К-3-2
2012-К-3-3	2012-К-3-4	2012-К-3-10	2013-К-1-3
2013-К-1-4	2013-К-2-3	2013-К-2-4	2013-К-3-4
2014-К-1-1	2014-К-1-4	2014-К-1-6	2014-К-2-1
2014-К-2-4	2014-К-2-5	2014-К-3-1	2015-К-1-2
2015-Э-1-2	2015-Э-2-4	2016-К-1-2	2016-К-1-6
2016-К-2-2	2016-К-2-6	2016-Э-1-9	2016-Э-2-9
2017-К-1-1	2017-К-1-8	2017-К-1-9	2017-К-2-1
2017-К-2-8	2017-К-2-9	2017-Э-1-2	2017-Э-2-2
2018-Э-1-6	2018-Э-2-6	2019-Э-1-1	2019-Э-1-2
2019-Э-1-4	2019-Э-2-1	2019-Э-2-2	2019-Э-2-4
2022-Э-1-1	2022-Э-2-1		

Функции-друзья и классы-друзья

2008-К-1-9	2008-Э-1-2	2009-К-1-5	2009-К-2-1
2009-К-3-3	2009-К-3-4	2009-К-4-5	2011-К-3-5
2011-К-4-5	2011-К-4-9	2013-К-1-5	2013-К-2-5
2013-К-3-5	2014-К-2-6	2016-К-2-5	2019-Э-1-5
2019-Э-2-5	2023-К-1-4	2023-К-2-4	2023-Э-1-2
2023-Э-2-2			

Константные и статические члены класса

2008-К-1-8	2008-К-2-8	2008-Э-1-5	2009-К-1-8
2009-К-2-9	2009-К-3-9	2009-К-4-1	2010-К-1-6
2010-К-2-6	2010-К-3-6	2010-К-4-6	2011-К-1-2
2012-К-1-5	2012-К-2-5	2012-К-2-7	2012-К-3-5
2013-К-1-8	2013-К-2-8	2013-К-3-8	2014-К-1-8
2014-К-2-8	2014-К-3-8	2015-Э-1-3	2015-Э-2-3
2016-К-1-7	2016-К-2-7	2016-Э-2-6	2017-К-1-4
2017-К-2-4	2017-Э-1-3	2017-Э-2-3	2022-Э-1-2
2022-Э-2-2	2023-К-1-2	2023-К-2-2	2024-Э-1-4
2024-Э-2-4			

Области видимости и пространства именования

2008-К-1-1	2008-К-2-1	2009-К-1-1	2009-К-2-3
2009-К-3-2	2009-К-4-3	2010-К-1-2	2010-К-1-7
2010-К-2-2	2010-К-2-7	2010-К-3-2	2010-К-4-2
2011-К-1-1	2011-К-1-8	2011-К-2-1	2011-К-3-1
2011-К-4-1	2013-К-1-1	2013-К-2-1	2013-К-3-1
2015-К-1-1	2015-К-1-6	2015-К-2-2	2015-К-2-6
2015-К-3-1	2015-К-3-9	2015-К-4-1	2015-К-4-9
2017-К-1-2	2017-К-2-2	2019-Э-1-1	2019-Э-1-5
2019-Э-2-1	2019-Э-2-5	2023-Э-1-3	2023-Э-2-3
2024-Э-1-1	2024-Э-2-1		

Статический полиморфизм. Перегрузка операций. Перегрузка функций

2008-К-1-5	2008-К-2-5	2009-К-1-3	2009-К-1-5
2009-К-1-9	2009-К-2-1	2009-К-2-2	2009-К-2-5
2009-К-3-3	2009-К-3-4	2009-К-3-8	2009-К-4-2
2009-К-4-5	2010-К-1-1	2010-К-1-7	2010-Э-5-7
2011-К-1-3	2011-К-1-5	2011-К-1-9	2011-К-2-3
2011-К-2-5	2011-К-3-3	2011-К-3-5	2011-К-4-3
2011-К-4-5	2012-К-1-2	2013-К-1-5	2013-К-2-5
2013-К-3-3	2013-К-3-5	2014-К-1-2	2014-К-2-2
2014-К-3-2	2014-К-3-4	2015-К-1-3	2015-К-1-5
2015-К-1-8	2015-К-2-8	2015-К-3-2	2015-К-3-8
2015-К-4-2	2015-К-4-8	2015-Э-1-1	2015-Э-2-1
2016-К-1-2	2016-К-1-3	2016-К-2-3	2016-Э-1-7
2016-Э-1-8	2016-Э-2-7	2016-Э-2-8	2016-Э-2-9
2017-К-1-3	2017-К-1-7	2017-К-2-3	2017-К-2-7
2017-Э-1-4	2017-Э-2-1	2017-Э-2-4	2018-Э-1-7
2018-Э-1-8	2018-Э-2-7	2018-Э-2-9	2019-Э-1-5
2019-Э-2-5	2022-Э-1-3	2022-Э-2-3	2023-К-1-2
2023-К-2-2	2023-Э-1-1	2023-Э-2-1	2023-Э-1-4
2023-Э-2-4	2024-Э-1-1	2024-Э-2-1	

Отношения между классами. Наследование. Видимость и доступность имен

2008-Э-1-10	2010-К-2-9	2011-К-1-2	2011-К-1-6
2011-К-2-6	2011-К-2-8	2011-К-3-8	2011-К-4-6
2011-К-4-8	2012-К-2-1	2013-К-1-6	2015-К-1-1
2015-К-2-1	2015-К-3-10	2015-К-4-8	2015-К-4-10
2016-К-1-1	2016-К-2-1		

Динамический полиморфизм. Виртуальные функции. Абстрактные классы

2008-К-1-2	2008-К-2-2	2008-К-2-6	2008-Э-1-4
2009-К-1-2	2009-К-1-6	2009-К-2-7	2009-К-3-1
2009-К-4-8	2010-К-1-3	2010-К-1-5	2010-К-2-3
2010-К-2-5	2010-К-3-3	2010-К-3-5	2010-К-4-3
2010-К-4-5	2010-Э-5-2	2011-К-2-2	2011-К-3-9
2011-К-4-2	2012-К-1-7	2012-К-3-7	2013-К-1-2
2013-К-2-2	2013-К-3-2	2013-К-3-6	2014-К-1-5
2014-К-2-5	2014-К-3-5	2014-К-3-6	2015-К-3-7
2015-К-4-7	2015-Э-1-2	2015-Э-2-2	2016-К-1-4
2016-К-1-5	2016-К-2-4	2017-К-2-8	2017-Э-1-2
2018-Э-1-8	2018-Э-2-8	2019-Э-1-1	2019-Э-2-1
2022-Э-1-1	2022-Э-2-1	2023-К-1-3	2023-К-2-3
2024-Э-1-2	2024-Э-2-2		

Аппарат исключений

2008-К-1-7	2008-К-2-7	2008-Э-1-1	2009-К-1-7
2009-К-2-9	2009-К-3-10	2009-К-4-4	2009-К-4-10
2010-К-1-10	2010-К-2-10	2010-К-3-10	2010-К-4-10
2010-Э-5-4	2011-К-1-7	2011-К-2-7	2011-К-3-7
2011-К-4-7	2012-К-1-8	2012-К-2-8	2012-К-3-8
2013-К-1-7	2013-К-1-9	2013-К-2-6	2013-К-2-7
2013-К-3-7	2014-К-1-7	2014-К-2-7	2014-К-3-7
2015-К-1-7	2015-К-2-7	2015-К-3-5	2015-К-4-5
2015-Э-1-4	2016-К-1-8	2016-Э-1-6	2017-К-1-6
2017-К-2-6	2017-Э-1-2	2018-Э-1-9	2018-Э-2-9
2019-Э-1-4	2019-Э-2-4	2022-Э-1-1	2022-Э-2-1
2023-К-1-1	2023-К-2-1	2024-К-1-1	2024-К-2-1

Динамическая идентификация и приведение типов

2008-К-2-9	2009-К-3-5	2011-К-3-2	2012-К-1-9
2012-К-2-9	2012-К-3-9	2013-К-1-9	2013-К-2-9
2013-К-3-9	2014-К-1-6	2014-К-1-9	2014-К-2-9
2014-К-3-9	2015-К-1-9	2015-К-2-9	2015-К-3-4
2015-К-4-4	2016-К-2-8	2017-К-1-5	2017-К-2-5
2017-Э-2-5	2018-Э-1-9	2018-Э-2-9	2022-Э-1-4
2022-Э-2-4	2024-К-1-1	2024-К-2-1	2024-Э-1-1
2024-Э-2-1			

Параметрический полиморфизм. Шаблоны

2008-К-1-6	2009-К-1-5	2009-К-1-6	2009-К-2-1
2009-К-3-4	2009-К-4-5	2010-К-1-4	2010-К-2-4
2010-К-3-4	2010-К-4-4	2011-К-1-5	2011-К-2-5
2011-К-3-5	2011-К-4-5	2012-К-2-6	2012-К-3-6
2013-К-1-5	2013-К-2-5	2013-К-3-5	2014-К-1-10
2015-К-1-3	2015-К-3-6	2015-К-4-6	2016-К-1-3
2016-К-2-3	2016-Э-1-8	2017-К-1-7	2017-К-2-7
2017-Э-1-1	2017-Э-2-1	2018-Э-1-8	2018-Э-2-8
2023-Э-1-4	2023-Э-2-4		

STL

2008-К-1-10	2008-Э-1-3	2009-К-1-5	2009-К-1-10
2009-К-2-4	2009-К-3-6	2009-К-4-9	2010-Э-5-9
2011-К-1-10	2011-К-2-10	2011-К-3-10	2011-К-4-10
2012-К-1-6	2012-К-2-6	2012-К-3-6	2013-К-1-10
2013-К-2-10	2013-К-3-10	2014-К-1-10	2014-К-2-10
2014-К-3-10	2015-Э-1-5	2015-Э-2-5	2016-К-1-10
2016-К-2-10	2016-Э-1-10	2016-Э-2-10	2017-Э-1-5
2018-К-1-3	2018-К-2-3	2018-Э-1-10	2018-Э-2-10
2019-Э-1-3	2019-Э-2-3	2022-Э-1-5	2022-Э-2-5
2023-К-1-2	2023-К-2-2	2024-Э-1-3	2024-Э-2-3

Формальные грамматики и формальные языки

2008-Э-1-6	2009-Э-1-1	2009-Э-2-1	2009-Э-3-1
2009-Э-4-1	2010-Э-1-1	2010-Э-1-2	2010-Э-2-2
2010-Э-3-1	2010-Э-3-2	2010-Э-3-9	2010-Э-4-1
2010-Э-5-1	2011-Э-1-1	2011-Э-2-1	2011-Э-3-1
2011-Э-4-1	2012-Э-1-1	2012-Э-1-2	2012-Э-2-1
2012-Э-2-2	2012-Э-3-1	2012-Э-3-2	2013-Э-1-1
2013-Э-1-3	2013-Э-2-1	2013-Э-2-3	2013-Э-3-1
2013-Э-3-3	2014-Э-1-1	2014-Э-1-2	2014-Э-2-1
2014-Э-2-2	2014-Э-3-1	2014-Э-3-2	2015-Э-1-6
2016-Э-1-1	2016-Э-2-1	2017-Э-1-7	2017-Э-2-7
2018-Э-1-1	2018-Э-2-1	2019-Э-1-6	2019-Э-2-6
2022-Э-1-7	2022-Э-2-7	2023-Э-1-9	2023-Э-2-9
2024-К-1-1	2024-К-1-2	2024-К-2-1	2024-К-2-2
2024-Э-1-7	2024-Э-2-7		

Алгоритмы контекстно-свободных грамматик

2009-Э-1-2	2009-Э-2-2	2009-Э-3-2	2009-Э-4-2
2010-Э-1-9	2010-Э-2-9	2010-Э-4-2	2011-Э-1-2
2011-Э-2-2	2011-Э-3-2	2011-Э-4-2	2012-Э-1-8
2012-Э-2-8	2012-Э-3-8	2013-Э-1-6	2013-Э-1-8
2013-Э-2-6	2013-Э-2-8	2013-Э-3-6	2013-Э-3-8
2014-Э-1-6	2014-Э-2-6	2014-Э-3-6	2014-Э-3-9
2015-Э-1-8	2015-Э-2-6	2015-Э-2-8	2019-Э-1-9
2019-Э-2-9	2022-Э-1-8	2022-Э-2-8	2024-К-1-2
2024-К-2-2			

Конечные автоматы, регулярные грамматики

2008-Э-1-7	2009-Э-1-3	2009-Э-1-4	2009-Э-2-3
2009-Э-2-4	2009-Э-3-3	2009-Э-3-4	2009-Э-4-3
2009-Э-4-4	2010-Э-1-3	2010-Э-1-4	2010-Э-2-3
2010-Э-2-4	2010-Э-3-3	2010-Э-3-4	2010-Э-4-3
2010-Э-4-4	2010-Э-5-3	2011-Э-1-3	2011-Э-1-7
2011-Э-2-3	2011-Э-2-5	2011-Э-2-7	2011-Э-3-3
2011-Э-3-5	2011-Э-3-7	2011-Э-4-3	2011-Э-4-5
2011-Э-4-7	2012-Э-1-3	2012-Э-1-4	2012-Э-2-3
2012-Э-2-4	2012-Э-3-3	2012-Э-3-4	2013-Э-1-4
2013-Э-1-7	2013-Э-2-4	2013-Э-2-7	2013-Э-3-4
2013-Э-3-7	2014-Э-1-3	2014-Э-1-7	2014-Э-2-7
2014-Э-3-3	2014-Э-3-7	2015-Э-1-3	2015-Э-2-7
2016-Э-1-2	2016-Э-2-2	2017-Э-1-9	2017-Э-2-9
2018-Э-1-4	2018-Э-2-4	2019-Э-1-8	2019-Э-2-8
2023-К-1-8	2023-К-2-8	2023-Э-1-5	2023-Э-2-5
2023-Э-1-7	2023-Э-2-7	2024-К-1-1	2024-К-2-1
2024-Э-1-5	2024-Э-2-5	2024-Э-1-6	2024-Э-2-6

Метод рекурсивного спуска

2009-Э-1-5	2009-Э-2-5	2009-Э-3-5	2009-Э-4-5
2010-Э-1-5	2010-Э-2-5	2010-Э-3-5	2010-Э-4-5
2010-Э-4-9	2010-Э-5-5	2011-Э-1-4	2011-Э-1-5
2011-Э-2-4	2011-Э-3-4	2011-Э-4-4	2012-Э-1-9
2012-Э-2-9	2012-Э-3-9	2013-Э-1-9	2013-Э-2-9
2013-Э-3-9	2015-Э-2-8	2019-Э-1-8	2019-Э-2-8
2023-К-1-5	2023-К-2-5	2023-Э-1-6	2023-Э-2-6
2024-К-1-2	2024-К-2-2	2024-Э-1-6	2024-Э-2-6

Граматики с действиями, синтаксически управляемый перевод

2008-Э-1-8	2009-Э-1-6	2009-Э-2-6	2009-Э-3-6
2009-Э-4-6	2011-Э-1-5	2011-Э-2-5	2011-Э-3-5
2011-Э-4-5	2012-Э-1-5	2012-Э-2-	2012-Э-2-5
2012-Э-3-5	2014-Э-1-9	2014-Э-2-8	2014-Э-3-8
2015-Э-1-9	2015-Э-2-9	2016-Э-1-4	2016-Э-2-4
2017-Э-1-8	2017-Э-2-8	2018-Э-1-2	2018-Э-2-2
2019-Э-1-7	2019-Э-2-7	2022-Э-1-9	2022-Э-1-10
2022-Э-2-9	2022-Э-2-10	2023-К-1-7	2023-К-2-7
2023-Э-1-8	2023-Э-2-8	2024-Э-1-8	2024-Э-2-8

Методы трансляции и внутреннее представление программ

2009-Э-4-7	2010-Э-1-7	2010-Э-2-7	2011-Э-1-6
2011-Э-2-6	2011-Э-3-6	2011-Э-4-6	2012-Э-1-6
2012-Э-2-6	2013-Э-2-5	2013-Э-3-2	2016-Э-2-3
2017-Э-2-6	2024-К-1-1	2024-К-1-2	2024-К-2-1
2024-К-2-2			

Постфиксная запись операторов программ

2008-Э-1-9	2009-Э-1-9	2009-Э-2-9	2009-Э-3-9
2009-Э-4-9	2010-Э-1-10	2010-Э-2-10	2010-Э-3-10
2010-Э-4-10	2010-Э-5-10	2011-Э-1-10	2011-Э-2-10
2011-Э-3-10	2011-Э-4-10	2012-Э-1-10	2012-Э-2-10
2012-Э-3-10	2013-Э-1-10	2013-Э-2-10	2013-Э-3-10
2014-Э-1-10	2014-Э-2-10	2014-Э-3-10	2016-Э-1-5
2016-Э-2-5	2017-Э-1-10	2017-Э-2-10	2018-Э-1-5
2018-Э-2-5	2019-Э-1-10	2019-Э-2-10	2023-Э-1-10
2023-Э-2-10			

Оптимизирующие преобразования и распределение памяти

2009-Э-1-10	2009-Э-2-10	2009-Э-3-10	2009-Э-4-10
2010-Э-3-7	2010-Э-4-7	2011-Э-2-9	2011-Э-3-9
2012-Э-2-7	2012-Э-3-7	2013-Э-1-5	2019-Э-1-10
2022-Э-1-6	2022-Э-2-6		

Компоненты систем программирования, элементы программной инженерии

2009-Э-1-7	2009-Э-1-8	2009-Э-2-7	2009-Э-2-8
2009-Э-3-7	2009-Э-3-8	2009-Э-4-8	2010-Э-1-6
2010-Э-1-8	2010-Э-2-6	2010-Э-2-8	2010-Э-3-6
2010-Э-3-8	2010-Э-4-6	2010-Э-4-8	2010-Э-5-8
2011-Э-1-8	2011-Э-1-9	2011-Э-2-8	2011-Э-3-8
2011-Э-4-8	2011-Э-4-9	2012-Э-1-7	2012-Э-3-6
2013-Э-1-2	2013-Э-2-2	2013-Э-3-2	2013-Э-3-5
2014-Э-1-4	2014-Э-1-5	2014-Э-2-4	2014-Э-2-5
2014-Э-3-4	2014-Э-3-5	2015-Э-1-10	2015-Э-2-10
2016-Э-1-3	2017-Э-1-6	2018-Э-1-3	2018-Э-2-3
2023-К-1-6	2023-К-2-6	2024-Э-1-7	2024-Э-2-7

Задачи на разные темы

2015-К-1-3	2015-К-1-4	2015-К-2-3	2015-К-2-4
2017-Э-1-2	2018-К-1-1	2018-К-1-2	2018-К-2-1
2018-К-2-2	2019-К-1-1	2019-К-1-2	2021-Э-1-1
2021-Э-2-1	2023-К-1-2	2023-К-2-2	2024-К-1-1
2024-К-2-1	2024-Э-1-1	2024-Э-2-1	

Учебное издание

ВОЛКОВА Ирина Анатольевна
ВЫЛИТОК Алексей Александрович
КАРПОВ Леонид Евгеньевич

СБОРНИК ЭКЗАМЕНАЦИОННЫХ ЗАДАЧ ПО КУРСУ
«СИСТЕМЫ ПРОГРАММИРОВАНИЯ»

Объектно-ориентированное программирование на Си++ и методы трансляции
на основе формальных грамматик и автоматов

Учебное пособие для студентов II курса

Издательский отдел
Факультета вычислительной математики и кибернетики
МГУ имени М. В. Ломоносова
Лицензия ИД N 05899 от 24.09.01 г.
119992, ГСП-2, Москва, Ленинские горы,
МГУ имени М.В. Ломоносова, 2-й учебный корпус

Издательство «МАКС Пресс»
Главный редактор: *Е.М. Бугачева.*
Обложка: *А.В. Кононова*

Напечатано с готового оригинал-макета
Подписано в печать 17.07.2026 г.
Формат 70х100 1/16. Усл.печ.л. 27,3. Тираж 50 экз. Заказ 105.

Издательство ООО «МАКС Пресс».
Лицензия ИД N 00510 от 01.12.99 г.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус, 527 к.
Тел. 8(495)939-3890/91. Тел./Факс 8(495)939-3891

Отпечатано в полном соответствии с качеством
предоставленных материалов в ООО «Фотоэксперт»
109316, г. Москва, Волгоградский проспект, д. 42,
корп. 5, эт. 1, пом. I, ком. 6.3-23Н