Ответы на вопросы экзамена по курсу «Языки программирования» 09.01.2017

В ответах курсивом выделены необязательные пояснения, которые можно опустить (особенно на экзамене)

Вариант 2

Задача 2-1

Есть ли ошибка в тексте приведенной ниже функции `r_vowels(s)` на языке C# (функция подсчитывает число гласных букв русского алфавита в строке-аргументе s)? Если есть, то объясните, в чем она состоит и как изменить текст функции, чтобы она работала правильно?

```
public static int r_vowels(string s) {
  int i,cnt = 0; const string vowels = "АЕЁИОУЫЭЮЯаеёиоуыэюя";
  for (i=0; i<s.Length; ++i) if (vowels.IndexOf(s[i])>=0) ++cnt;
  return cnt;
}
```

Ответ

Текст этой задачи с небольшими изменениями, вызванными синтаксическими различиями C# и JavaScript, скопирован из задачи 1 первого варианта (для JavaScript), однако ответы в задачах различны в силу различий в представлении символов в двух языках.

В языке C# представление символов в оттранслированной программе (и, следовательно, строк) зафиксировано — это кодировка Юникода UTF16 (в .NET – UTF16 Little-Endian). Поэтому операция индексирования строк вполне корректна для выделения символа из строк на русском языке, и функция будет работать правильно.

Задача 2-2

Что будет выдано в стандартный канал вывода после выполнения следующей программы на C++? Считать, что все нужные директивы `include` и `using` добавлены в текст.

```
class TT {
   int _p;
public:
    TT(const TT& t) : _p(t._p){ cout << "TT& " << _p<< endl;}
    explicit TT(int p = 0) : _p(p){cout << "Hello, "<< _p << endl; }
    ~TT() {cout << "goodbye, " << _p << endl;}
};
int main() { list<TT> v; v.push_back(TT(1)); v.push_back(TT(2)); return 0; }
```

Добавьте в класс TT РОВНО ОДИН метод так, чтобы результатом стали следующие строки (<CR> означает переход на новую строку):

Hello, 1<**CR**> TT&& 1<**CR**> goodbye, -1<**CR**> Hello, 2<**CR**> TT&& 2<**CR**> goodbye, -2<**CR**> goodbye, 1<**CR**> goodbye, 2<**CR**>

Ответ

Будет выдано:

Hello, 1

TT& 1

goodbye, 1

Hello, 2

TT& 2

goodbye, 2

goodbye, 1

goodbye, 2

Надо вставить конструктор перемещения (move-конструктор):

TT(TT&& t): _p(t._p) { t._p = -t._p; cout << "TT&& " << _p << endl;}

Задача 2-3

Дайте определение функции-генератора языка Python. Напишите пример функции-генератора, включая ее использование. Как связаны понятия функции-генератора и сопрограммы в Python?

Ответ

Функция-генератор отличается от обычной тем, что содержит конструкцию yield. Вызов ФГ возвращает объект-итератор it. Последовательный вызов фунции next(it) передает управление ФГ до очередного yield. Данные, которые указаны в yield, возвращаются как результат вызова next(it). При выходе из ФГ генерируется исключение StopIteration

Пример;

```
def NN(n):
     for in range(1,n+1):
          yield i*i
Использование:
for x in NN(100):
  print(x)
Печатает первые 100 квадратов целых чисел (1,4,9,16,..…)
```

Связь сопрограмм и ФГ — см ответ на задачу 1-3 первого варианта

Задача 2-4

На языке Java написан класс `CSV` (не компилируется из-за отсутствия объявления имени `Printer` и знаков вопроса в вызовах метода *`print`*):

```
public class CSV {
  public static void main(String[] args) {
    for (int i = 0; i < 10; ++i) print(???, i); // подставить правильное значение
    print(???, 10); // подставить правильное значение
  }
  static void print(Printer p, Object x) {
    p.print(x);
  }
  static void printInner(Object x) {
    System.out.print('\"');System.out.print(x); System.out.print("\", ");
  }
  static void printLast(Object x) {
       System.out.print('\"');System.out.print(x); System.out.println("\"");
  }
}
```

Объявить тип данных `Printer` и заменить вопросы в вызовах метода *`print`* на правильные значения (НЕ ИСПОЛЬЗУЯ лямбда-выражения) так, чтобы в канал вывода было выдано:

"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10"

Ответ

См. ответ и замечания по поводу задачи 1-4 из первого варианта

Во-первых, надо определить функциональный интерфейс `Printer` с единственным методом `print(Object x)`:

```
public interface Printer {
	void print(Object x);
}
```

Во-вторых, вставить вместо вопросов ссылки на методы класса `CSV`:

```
for (int i = 0; i < 10; ++i) print(CSV::printInner, i);
print(CSV::printLast, 10);
```

Замечание: если бы не ограничение в условии, то можно было бы вместо вопросов подставить лямбда-выражения вида: x⟶{ тело соответствующего метода}

Задача 2-5

Переписать программу из задачи №4 на языке C# так, чтобы она выдавала в стандартный канал вывода те же самые значения (набор и сигнатуры методов класса `CSV` должны остаться без изменения). Для вывода использовать **`System.Console.Write(…) и WriteLine(…)`**

Ответ

См. ответ и замечания по поводу задачи 1-5 из первого варианта

```
public class CSV
{
    delegate void Printer(Object x);
    static void Main(String[] args) {
        for (int i = 0; i < 10; ++i) print(CSV.printInner, i);
        print(CSV.printLast, 10);
    }
    static void print(Printer p, Object x)
    {
        p(x);
    }
    static void printInner(Object x) {
          System.Console.Write('\"');
          System.Console.Write(x);
          System.Console.Write("\", ");
    }
    static void printLast(Object x) {
        System.Console.Write('\"');
          System.Console.Write(x);
          System.Console.WriteLine("\"");
    }
}
```

Задача 2-6

Объявление шаблонного класса **`CalcSort`** на языке C++ предполагает, что над параметром шаблона T выполняются 4 арифметические операции (+,-,*,/), операции сравнения:

```
template <typename T> class CalcSort { … };
```

Напишите эквивалентное объявление обобщенного класса **`CalcSort`** на языке C# (опустив, как и выше, все объявления членов класса), добавив при необходимости нужные объявления типов вне этого класса.

Ответ

В силу того, что реальное инстанцирование обобщений и генерация кода в C# происходят

в момент выполнения (runtime), то вся информация о свойствах типа (то есть об операциях над типом) должна быть доступна компилятору в точке определения обобщения (а не в точке конкретизации, как в C++), когда ничего не известно о конкретных фактических аргументах. Ее задают посредством where-клаузы в определении обобщения. При этом удобнее всего «запаковать» эти свойства в интерфейс (а не конкретный класс, хотя и так можно) — и указать, что тип-фактический параметр должен реализовывать этот интерфейс. Тут возникает две проблемы: во-первых, только классы или структуры могут быть фактическими параметрами этого обобщения, во-вторых, интерфейсы не могут содержать перегрузки стандартных операций (в C# стандартные операции перегружаются статическими методами, которых у интерфейсов быть не может). Поэтому надо указать для сравнений — стандартный интерфейс IComparable, а арифметические операции выразить с помощью своего интерфейса, который по условию и требуется определить.

```
interface Arithmetic<T> {
        T Add(T x);
        T Mult(T x);
        T Sub(T x);
        T Div(T x);
}
class CalcSort<T> where T : IComparable<T>, Arithmetic<T> { … }
```

Ошибкой является указание нескольких параметров методов арифметического интерфейса, неупотребление обобщенных интерфейсов, другие ограничения (зачем лишние?) типа class, struct и т. д., перегрузки стандартных операций в интерфейсах. Грубейшей ошибкой является зыбыть про ограничения вообще

Задача 2-7

Дайте определение и пример внутреннего класса в языке Java. Есть ли аналогичное понятие в языках C++ или C#?

Ответ

Коротко: детали ответа можно найти в книге Гослинга, Арнольда и Холмса «Язык программирования Java» 3-е издание, п.5.2. Внутренние классы, стр.142-144.

В C# и C++ такого понятия нет.

Замечание. Постоянная (и грубая) ошибка: путать внутренние и вложенные классы. Это - «две большие разницы». Подробно и обстоятельно написано там же в 5 главе «Вложенные классы и интерфейсы».

Задача 2-8

Пусть программа на языке Java содержит следующий ошибочный фрагмент:

```
class D { }
interface IFace <T, R>
{
   R Generate();
   void Process(T x);
}
IFace<Object, D> i1;
i1 = new ObjectFace(); //error!
i1 = new DFace(); //error!
```

```
class ObjectFace implements IFace<Object,
Object> {
  public Object Generate() { return new
Object();}
  public void Process(Object x) {}
}
class DFace implements IFace<D, D> {
  public D Generate() { return new D(); }
  public void Process(D x) { }
}
```

Объясните, в чем состоит ошибка, и исправьте РОВНО ОДНУ строчку во фрагменте так, чтобы ошибка исчезла.

Ответ

См. ответ и замечания по поводу задачи 1-8 из первого варианта

В языке Java указание вариантности делается не при определении, а при ИСПОЛЬЗОВАНИИ обобщенного интерфейса. Это делается с помощью понятия метасимвольной маски типа-параметра обобщения. Без метасимвольной маски конкретизация обобщенного интерфейса инвариантна. Но мы можем объявить переменную с типом — конкретизацией обобщенного интерфейса, где в качестве параметра типа стоит символьная маска. Тогда к этой переменной может быть ковариантно (? super T) или контравариантно (? extends T) преобразовано значение конкретизаций обобщенного интерфейса без масок.

Вместо
```
IFace<Object, D> i1;
```
поставить `IFace<? extends Object, ? super D> i1;`