МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
факультет вычислительной математики и кибернетики

Кузина Л.Н.

Сборник практических заданий по языку Си++. Учебно-методическое пособие для студентов бакалавриата 2 курса, обучающихся по направлению «Прикладная математика».
Москва, 2016.

Данное методическое пособие содержит задачи, предлагаемые для практического выполнения на ЭВМ студентам бакалавриата 2 курса специализации ПМ в рамках семинаров по практикуму.
Пособие может быть использовано студентами для самостоятельной подготовки, а также может быть полезно преподавателям и аспирантам для проведения занятий по программированию на языке Си++.

Содержание

Задание 1. Введение. Понятие класса. Конструкторы.

1. Написать класс с данными разных типов, среди которых обязательно должны быть указатели.
Для этого класса написать конструкторы (конструктор умолчания, конструкторы с параметрами, конструктор копирования, конструктор перемещения), деструктор, оператор присваивания, оператор перемещения и произвольную функцию-член класса.
В каждую из специальных функций класса включить отладочный вывод на экран.

Функция main должна демонстрировать работу с объектами данного класса.
Использование STL запрещено.

Задание 2. Абстрактные типы данных (АТД).

2.1. АТД. Список данных.

Класс — **Список**, элемент данных — **int** (либо произвольного типа).
Члены-данные класса должны находиться в закрытой области класса.
Определить необходимые конструкторы.
Обеспечить корректное уничтожение объектов.

В классе должны быть функции:
добавления элемента в начало (**push_front**) и в конец (**push_back**),
чтение первого элемента списка (**front**),
чтение последнего элемента списка (**back**),
удаление первого элемента списка (**pop_front**),
удаление последнего элемента списка (**pop_back**),
добавление элемента **x** перед позицией **p** (**insert(p,x)**),
удаление элемента из позиции **p** (**erase(p)**)
проверка списка на пустоту (**empty**),
текущее число элементов (**size**),
вывод информации об элементах списка (**print**).

Использование STL запрещено.

Пример работы: List l1; l1.push_front(1); l1.print(); и т.д.

2.2. АТД. Очередь для хранения данных на основе списка.

На основе класса **Список** (см. п.2.1) определить класс **Очередь**, который должен быть производным от класса Список.
Элемент данных — см. вариант задания (п.2-5).
Максимальный допустимый размер очереди определен по умолчанию, а также может явно указываться при создании объекта-Очереди.

В классе должны быть функции:

добавления элемента в конец очереди (**back**),
чтение первого элемента из очереди без его удаления (**front**) ,
удаление первого элемента очереди (**pop**),
проверка очереди на пустоту (**empty**),
текущее число элементов (**size**),
проверка, что очередь целиком заполнена (**full**).
вывод информации об элементах очереди без ее изменения (**print**).

Обеспечить корректное уничтожение объектов.
Использование STL запрещено.

Пример работы: Queue q1(5), q2; q1.back(el); cout<<q2.size(); и т.д.

2.3. АТД. Очередь для хранения данных на основе массива.

Определить класс **Очередь**, для хранения данных использовать массив.
Элемент данных — см. вариант задания.
Максимальный допустимый размер очереди указывается при создании каждого объекта-Очереди.
В классе должны быть функции:
добавления элемента в конец очереди с контролем выхода за границу (**back)**,
чтение первого элемента из очереди без его удаления (**front**) ,
удаление первого элемента очереди (**pop**),
проверка очереди на пустоту (**empty**),
текущее число элементов (**size**),
проверка, что очередь целиком заполнена (**full**).
вывод информации об элементах очереди без ее изменения (**print**).

Обеспечить корректное уничтожение объектов.
Использование STL запрещено.

2.4. АТД. Стек для хранения данных.

Определить класс **Стек**, для хранения данных использовать массив.
Элемент данных — см. вариант задания.
Максимальный допустимый размер стека указывается при создании каждого объекта-Стек.
В классе должны быть функции:
добавление элемента на вершину стека (**push**),
удаление верхнего элемента стека без его просмотра (**pop**) ,
получение элемента с вершины стека (**top**),
проверка стека на пустоту (**empty**),
текущее число элементов (**size**),
проверка, что стек полон (**full**),
вывод информации об элементах стека без его изменения (**print**).

Обеспечить корректное уничтожение объектов.
Использование STL запрещено.

2.5. Варианты данных, которые помещаются в очередь./стек.

1. **Элемент данных -** объект, содержащий информацию о клиенте: фамилия, время добавления в очередь (целое или строка). Хранится также информация о текущем количестве клиентов в очереди.

2. **Элемент данных** – объект, содержащий информацию о заказе: название фирмы, номер телефона (целое или строка), номер заказа. Нумерация заказов единая для всех списков или очередей.

3. **Элемент данных** – объект, содержащий информацию о книге: автор, название, год издания, код - номер книги в порядке поступления (нумерация единая для всех очередей/списков).

4. **Элемент данных** – объект, содержащий информацию о работе с файлом: имя файла, вид операции (чтение, запись, чтение+запись). В классе должна храниться информация о количестве файлов в очереди/списке.

5. **Элемент данных –** объект «банковский счет». Необходимые члены-данные: номер счета, владелец счета, дата создания счета (число или строка), сумма денег, которая на нем хранится. Нумерация счетов единая для всех очередей (списков).

Задание 3. Абстрактные классы.

Определить абстрактный класс (предметная область и содержание произвольное, члены-данные должны быть объявлены в закрытой части класса) и несколько (>=2) производных от него классов. Определить независимый от этой иерархии класс, который работает с массивом указателей/ссылок на объекты типа абстрактного класса.
Функция main должна демонстрировать работу с объектами указанных классов.

Задание 4. Перегрузка операций.

Определить (либо дополнить, если был ранее написан) класс Матрица таким образом, чтобы
1) к объектам этого типа была применима двойная индексация,
2) к объектам этого типа была применима операция сложения,
3) к объектам этого типа была применима операция вывода на экран.

Должен быть верным, например, следующий фрагмент программы:

```
Matrix m1; … m1[1][2] = 5;  int x = m1[2][3];
Matrix m2,  m3 (m1+ m2); …
cout<< m1 << m2<< m3;
```

Задание 5. Исключения.

Модифицировать ранее написанную программу (например, программу работы с Очередью):

Определить класс Исключение (возможно иерархию классов), описывающий исключительные ситуации. В классе должно содержаться по крайней мере 2 поля: № ошибки, текстовое описание ошибки.

Обеспечить обработку особых ситуаций, которые могут возникать при работе данной программы, таких как ввод неверных данных, переполнение очереди и т.п.

Задание 6. STL. Работа с контейнерами list и vector.

При решении задач данного раздела запрещено использовать ключевое слово ***auto*** для объявления типа. Обратите внимание на использование квалификатора ***const.***
Для демонстрации работы функций в ***main*** создайте необходимые контейнеры, среди них должны быть и константные.

6.1. Написать **шаблонную** функцию, которая выводит на экран содержимое контейнера сначала в прямом порядке, потом в обратном порядке (с применением обратного итератора).

6.2. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый вектор, полученный из исходного заменой четных элементов вектора на нечетные элементы списка.
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные и результат работы функции (см.6.1.).

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Vnew : l1-v1-l3-v3.....

6.3. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый вектор, полученный из исходного заменой нечетных элементов вектора на четные элементы списка.
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Vnew : v0-l2-v2-l4-v4.....

6.4. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый список, полученный из исходного добавлением после четных элементов списка нечетных элементов вектора.
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Lnew : l1-l2-v1-l3-l4-v3.....

6.5. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый список, полученный из исходного добавлением перед нечетными элементами списка нечетных элементов вектора.
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Lnew : v1-l1-l2-v3-l3-l4.....

6.6. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый список, полученный из исходного чередованием пар элементов списка и вектора.
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Lnew : l1-l2-v0-v1-l3-l4-v2-v3.....

6.7. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый вектор, полученный из исходного заменой четных элементов вектора на нечетные элементы списка (считать элементы списка с конца).
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Vnew : l_m-v1-l_{m-2}-v3.....

6.8. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый вектор, полученный из исходного заменой нечетных элементов вектора на четные элементы списка (считать элементы списка с его конца).
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Vnew : v0-l_{m-1}-v2-l_{m-3}-v4.....

6.9. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый список, полученный из исходного добавлением после четных элементов списка нечетных элементов вектора (считать элементы вектора с его конца).
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные (в прямом порядке и в обратном – с использованием обратных итераторов) и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Lnew : l1-l2-v_n-l3-l4-v_{n-2}.....

6.10. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый список, полученный из исходного добавлением перед нечетными элементами списка нечетных элементов вектора (считать элементы вектора с его конца).
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные (в прямом порядке и в обратном – с использованием обратных итераторов) и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Lnew : v_n-l1-l2-v_{n-2}-l3-l4.....

6.11. Написать **шаблонную** функцию от 2ух параметров (вектор и список). Функция должна построить и вернуть новый список, полученный из исходного чередованием пар элементов списка и вектора. Просмотр списка ведется с его начала, а вектора – с конца.
Длины вектора и списка могут быть различны.
Вывести на экран исходные данные (в прямом порядке и в обратном – с использованием обратных итераторов) и результат работы функции.

Пример работы: V: v0-v1-v2-…vn L: l1-l2-…lm
Lnew : l1-l2-v_n-v_{n-1}-l3-l4-v_{n-2}-v_{n-3}.....

Список литературы.

1. Волкова И. А., Иванов А. В., Карпов Л. Е. Основы объектно-ориентированного программирования. Язык программирования C++. Учебное пособие для студентов 2 курса. – М.: Издательский отдел факультета ВМК МГУ, 2011 – 112 с.
2. Standard for the C++ Programming Language ISO/IEC 14882, 2011.
3. Страуструп Б. Язык программирования C++. Специальное изд./Пер. с англ. - М.: "Бином", 2015.