Лабораторная работа 8

Файловые потоки и контейнеры STL

Цель этой лабораторной работы — познакомиться с некоторыми объектами стандартной библиотеки C++.

Некоторые задания в этой работе требуют ввода матриц. В ходе написания решения вы будете несколько раз запускать программу для проверки. Если одно число еще можно повводить несколько раз, то повторно вводить 3×4 матрицу утомительно. В таких случаях логично один раз записать данные в текстовый файл, а потом заставить программу читать из файла. Как организовать ввод из файла и вывод в файл?

В стандартной библиотеке Си++ предусмотрены объекты, представляющие текстовые потоки для ввода `ifstream` и вывода `ofstream`. Они объявляются в заголовочном файле `fstream`. Они должны быть связаны с файлом на диске. Это можно сделать при создании объекта при помощи конструктора. Конструктор — это специальная функция для создания объекта и настройки его внутреннего состояния, возможно из некоторых начальных данных, передаваемых как параметры. При этом имя функции совпадает с именем класса создаваемого объекта. Такое определение имеет вид

а) `класс имяобъекта = класс(параметрыконструктора);`

или

б) `класс имяобъекта(параметрыконструктора);`

Создадим объекты `in` и `out` согласно варианту б), представляющие файлы `input.txt` и `output.txt` для чтения и записи, соответственно. Введем число из первого файла и запишем строку, содержащую запись этого числа, в выходной файл. Как видно, работа с файловыми потоками не отличается от работы с `cin`/`cout`. Хорошая практика — закрывать файлы, как только они перестали быть вам нужны.

```
#include <fstream>
using namespace std;
int main() {
  ifstream in("input.txt");
  ofstream out("output.txt");
  in >> n;
  in.close();
  out << "Hi " << n " times!\n";
  out.close();
```

```
  return 0;
}
```

При работе с файлами иногда бывает нужно считывать заранее неизвестное количество данных, пока файл не закончится. Потоковый объект «не узнает», что файл закончился, пока не попробует прочитать дальше конца файла. Ваша программа может проверить это, приводя переменную типа `ifstream` к логическому типу: `false`означает, что файл не в порядке, в частности, закончился. Таким образом, если известно, что в файле `input.txt` записаны целые числа, но неизвестно заранее, сколько их, то узнать их количество можно так:

```
#include <fstream>
#include <iostream>
int main() {
  std::ifstream in;
  int a, count = 0;
  in.open("input.txt");
  in >> a;
  while(in) {
    count++;
    in >> a;
  }
  in.close();
  std::cout << count << "\n";
  return 0;
}
```

ОБЩИЕ ЗАДАНИЯ

1. Дан текстовый файл `input.txt`, выведите строку наибольшей длины в этом файле в выходной файл `longest.txt`.

2. Дан файл `input.txt`, содержащий целые числа по одному в строке. Выберите наибольшее из них, принадлежащее интервалу $[a, b]$. Целочисленные концы интервала a, b вводятся со стандартного устройства ввода.

3. Дан файл `input.txt`, содержащий целые числа по одному в строке. Выведите только те из них, что не меньше последнего. Использовать векторы.

4. Дан текстовый файл `input.txt`, содержащий целочисленную матрицу размера $n \times m$, записанную в следующем виде. В первой строке файла через пробел записаны два целых числа n и m, число строк и столбцов в матрице. В каждой из последующих n строк записаны по m целых чисел, элементы матрицы. Выведите на стандартное устройство вывода максимальные элементы каждой строки поочередно, каждый в отдельной строке.

5. Даны текстовые файлы `a.txt` и `b.txt`, содержащие две матрицы A и B, соответственно, записанные как в предыдущем пункте, причем число столбцов в A совпадает с числом строк в B. Выведите в файл `ab.txt` матрицу их произведения AB, в том же формате. Использовать векторы.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

1.1. Продолжение следует.