|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент     Папин Алексей Владимирович

*фамилия, имя, отчество*

Группа ИУ5Ц-41Б

Тип практики ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА

Название предприятия КАФЕДРА «СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ» ( ИУ5 ) МГТУ им. Н.Э.Баумана

Студент Папин А.В.       

*подпись, дата фамилия, и.о.*

Руководитель практики Правдина А.Д.**\_\_**

*подпись, дата фамилия, и.о.*

Оценка

*2022 г.*

**СОДЕРЖАНИЕ**

[1. Введение 2](#_Toc105799167)

[1.1. Цель лабораторной работы 2](#_Toc105799168)

[1.2. Задачи практики 2](#_Toc105799169)

[1.2.1. Постановка задачи №1 2](#_Toc105799170)

[Пояснения: 2](#_Toc105799171)

[1.2.2. Постановка задачи №2 3](#_Toc105799172)

[Пояснения: 4](#_Toc105799173)

[2. Основная часть 7](#_Toc105799174)

[2.1. Листинги программы: 7](#_Toc105799175)

[MyVector.h 7](#_Toc105799176)

[MyVector.cpp 13](#_Toc105799177)

[MySet.h 17](#_Toc105799178)

[MySet.cpp 20](#_Toc105799179)

[MyStack.h 21](#_Toc105799180)

[MyStack.cpp 26](#_Toc105799181)

[Practium.cpp 27](#_Toc105799182)

[2.2. Результаты работы программы: 34](#_Toc105799183)

[3. Заключение 39](#_Toc105799184)

[4. Список использованных источников 39](#_Toc105799185)

# **Введение**

## **Цель лабораторной работы**

Получение навыков совместного использования знаний по отдельным приемам программирования, изучаемым в курсах «Основы программирования», «Информатика» и «Программирование на основе классов и шаблонов»; знакомство с динамическими структурами данных такими как односвязный список, стек, динамический массив, размер которого может автоматически изменяться (увеличиваться или уменьшаться) в процессе выполнения программы; приобретение практических навыков объектно-ориентированного программирования.

## **Задачи практики**

## **Постановка задачи №1**

Дано описание класса MyStack (Приложение 1, файл MyStack.h), который реализует на основе односвязного списка динамическую структуру данных типа стек.

1. Разработайте реализацию интерфейса класса в виде файла MyStack.cpp.
2. Разработайте функцию (глобальную), которая выполняет разложение на простые множители целого числа N. Для хранения множителей функция должна использовать класс MyStack.

Прототип функции:

void Multipliers(int n, MyStack<int>& stack).

1. В функции main() распечатайте множители, которые функция Multipliers() записывает в стек, сначала по убыванию, а потом по возрастанию.

Например, для N=3960 программа должна вывести:

3960=11 \* 5 \* 3 \* 3 \* 2 \* 2 \* 2

3960=2 \* 2 \* 2 \* 3 \* 3 \* 5 \* 11

Нельзя изменять описание класса, приведенное в файле MyStack.h.

## **Пояснения:**

Стек представляет собой динамическую структуру (то есть структуру, размер которой может изменяться в процессе выполнения программы), предназначенную для временного хранения данных некоторого типа INF, который может быть как базовым, так и определяемым пользователем. Запись данных в стек и выборка данных из стека производятся путем обращения к его вершине.

В данной работе стек реализуется на базе односвязного списка. При записи в стек (операция push) в начало списка добавляется узел, а при выборке из стека (операция pop) узел удаляется из начала списка. Чтобы получить доступ к следующей ячейке стека нужно удалить предыдущую. При выполнении операции pop данные из стека не считываются. Данные могут считываться только из вершины стека. Для чтения данных используется специальная функция, которая читает данные без удаления узла из вершины стека.

Для обеспечения доступа к данным, хранящимся в узлах типа ListNode, класс MyStack сделан дружественным по отношению к классу ListNode. В этом случае все методы класса MyStack получают доступ к скрытым данным класса ListNode.

Чтобы в узлах можно было бы хранить данные различных типов и чтобы узлы класса ListNode можно было бы использовать в различных структурах (например, для реализации списка или очереди), класс ListNode реализован в виде шаблона семейства классов с двумя формальными параметрами: типом хранимых данных (class INF) и дружественным классом, реализующим некоторую структуру данных, например стек (class FRIEND).

Методы шаблонного класса не должны зависеть от значений формальных параметров и должны быть одинаковыми для всех типов хранимых данных и дружественных классов. Для реализации этого требования в лабораторной работе класс ListNode сделан закрытым (то есть в нем нет методов в разделе public, хотя могли бы и быть), а доступ к его элементам осуществляется через интерфейс дружественного класса FRIEND, являющегося одним из двух формальных параметров класса ListNode (в данной работе это MyStack<INF>).

## **Постановка задачи №2**

Разработать класс множество (MySet) на базе класса вектор (MyVector) для выполнения операций над множествами (+, -, \*,+=,-=,\*=,== ) и функцию main() для его тестирования.

Класс вектор должен быть динамическим массивом, размер которого может автоматически изменяться (увеличиваться или уменьшаться) в процессе выполнения программы. Добавление элементов производится в конец вектора.

Для ускорения выполнения операций над множествами вектор, используемый классом множество, должен быть отсортирован (сортировку достаточно делать только при добавлении элемента в множество). Для поиска элементов множества следует использовать метод половинного деления.

Методы add\_element() и delete\_element() производного класса MySet перегружают одноименные методы базового класса MyVector, а остальные элементы класса MyVector наследуются классом MySet.

## **Пояснения:**

Одно из основных применений наследования – использование ранее разработанных и уже проверенных классов для случаев, когда разрабатываемый класс несколько отличается от имеющегося, но есть желание использовать его в качестве базового. Дополнительные трудности на этапе освоения этой технологии возникают при использовании библиотечных классов из-за того, что библиотечные классы разработаны как шаблоны классов и их использование имеет свои особенности.

Множества хранят элементы, значения ключей которых должны быть уникальными. Наиболее часто используемой операцией при работе с множествами является поиск элемента. Поэтому для эффективной работы класс set должен обеспечивать быстрый доступ к данным по ключу. Для реализации этого требования множества, как правило, реализуются на основе сбалансированного дерева или отсортированного массива.

Класс с именем **MySet** включает следующие элементы:

- Все **члены - данные** наследуются из класса ***MyVector*** (элементы множества хранятся в векторе)***.***

**- Конструкторы и деструкторы.** Так как своих данных в ***MySet*** нет, то можно не перегружать имеющиеся (по умолчанию) конструктор копирования, оператор присваивания и деструктор.  
При необходимости будут вызываться соответствующие элементы базового класса.

**- Метод доступа** **IsElement**, который даёт **true**, если строка-параметр есть в множестве, иначе даёт **false**. Для поиска элементов множества следует использовать метод половинного деления. Для его реализации разработать метод **q\_find**, который имеет тип доступа **private**.

**- Методы изменения**:

**AddElement** - добавляет строку в множество, если её там ещё нет. Для ускорения поиска элементов создаваемое множество должно быть отсортировано по возрастанию значения. Для сортировки при добавлении элементов использовать метод **sort()** базового класса.

**DeleteElement** для удаления элемента из множества, если он там есть.

**- Операторные функции =,** *-=, +=, \*=,* где - означает разность, + **-** объединение и **\*** - пересечение множеств. (См. примеры ниже.)

**- Функции – не члены класса (друзья класса):**

Перегруженная операция потокового вывода;

**Операторы +** (объединение), **-** (разность), **\*** (пересечение) и **==** (сравнение: истина, если элементы двух множеств совпадают).

Примеры операций над множествами (приведены для целых чисел):

{1, 4, 5, 6} + {1, 2, 3, 4} => {1, 2, 3, 4, 5, 6}

{1, 4, 5, 6} \* {1, 2, 3, 4} => {1, 4}

{1, 4, 5, 6} - {1, 2, 3, 4} => {5, 6}

Базовый класс **MyVector**, является динамическим ***массивом строк.*** Размер вектора ***maxsize*** должен меняться в процессе выполнения программы следующим образом:

если при добавлении элемента число элементов вектора ***size*** превысит размер вектора, ***maxsize*** увеличивается примерно в 1,5 раза (был 8, станет 12, если ***size***>=8);

если при удалении элемента число элементов вектора ***size*** станет меньше ***maxsize/2,*** ***maxsize*** уменьшается примерно в 1,5 раза, но должен быть не меньше значения по умолчанию (был 12, станет 8, если ***size*** <6). Новый элемент добавляется в конец вектора.

**Члены - данные (protected):**

**maxsize –**размер вектора;

**size** – количество элементов в векторе;

**pdata** – указатель, содержащий адрес динамического массива элементов (строк).

**Класс MyVector должен реализовывать следующие функции:**

**add\_element** – вставка элемента в конец вектора;

**delete\_element** – удаление элемента из произвольного места;

**find(el) –** возвращает индекс элемента или –1, если элемент не найден;

**resize** – изменение размера вектора **maxsize** при его переполнении или освобождении места **(private);**

**Конструкторы и деструктор**

Конструктор с одним параметром (символьная строка) для создания множества размером 1, который имеет значения по умолчанию и поэтому может использоваться для создания пустого множества;

Конструктор копирования;

Деструктор.

**Операторы**

**[ ]** - для возврата элемента вектора (доступ по индексу);

**=** - оператор присваивания.

При выполнении лабораторной работы использовать файлы с описанием классов ***MyVector, MySet*** и тестирующей программой ***lab4***, приведенные в Приложении 2.

Работа состоит из четырех частей. Общее время выполнения работы – 16 часов.

Содержание частей:

– Разработка и тестирование класса ***MyVector*** для хранения символьных строк.

– Разработка и тестирование класса ***MySet*** для выполнения операций над множествами символьных строк.

– Разработка и тестирование шаблонного класса ***MyVector***.

– Разработка и тестирование шаблонного класса ***MySet.***

# **Основная часть**

## **Листинги программы:**

### **MyVector.h**

//файл MyVector.h - описание класса MyVector

#ifndef MyVector\_h

#define MyVector\_h

#include <iostream>

#include <string>

using namespace std;

// Отключение предупреждении о потери данных при преобразовании с double на int (строка 27 и 31)

#pragma warning(disable : 4244)

/\*

Пометка:

Чтобы описать специализированный класс, нужно сначала описать (пример)

// .h - файл

template <typename T> class S {

void test();

};

// Определение "общей" реализации

template <typename T> void S::test() {

...

}

// Явная специализация для 'S<int>' - здесь только объявление

template <> void S<int>::test();

// .cpp - файл

// А здесь - определение

template <> void S<int>::test() {

...

}

\*/

const int MAX\_SIZE = 5;

template<class T = string>

class MyVector

{

protected:

// размер вектора

int maxsize;

// количество элементов в векторе;

int size;

// указатель, содержащий адрес динамического массива элементов (строк).

T\* pdata;

private:

// изменение размера вектора maxsize при его переполнении или освобождении места (private);

void resize();

public:

// Конструктор без параметров

MyVector();

// Конструктор с параметрами

MyVector(T el);

// Конструктор копирования

MyVector(MyVector<T>& v);

// Деструктор

~MyVector();

// Вставка элемента в конец вектора

void add\_element(const T el);

// Удаление элемента из произвольного места;

void delete\_element(int number);

// Для возврата элемента вектора (доступ по индексу);

T operator[](int i);

// Сортировка

void sort();

// Обмен местами

void swap(int i, int j);

// Получение размера элемента вектора

int Size() { return size; }

// Получение размера вектора

int Maxsize() { return maxsize; }

// Возвращает индекс элемента или –1, если элемент не найден

int find(T el);

// Определение длин чисел

int length(T el) {

int count = 0;

while (el) {

el /= 10;

count++;

}

return count;

}

// Перегруженная операция присваивания и конструктор копирования

MyVector<T>& operator=(MyVector<T> v);

//friend ostream& operator<<(ostream& out, MyVector<T>& v);

friend ostream& operator<<(ostream& out, MyVector<T>& v) {

for (int i = 0; i < v.size; i++) {

out << v.pdata[i] << " ";

}

return out;

}

};

//================================================================================================

// ОБЩИЕ (Для любых типов переменных)

//================================================================================================

template<class T>

MyVector<T>::MyVector() {

pdata = new T[MAX\_SIZE]();

maxsize = MAX\_SIZE;

size = 0;

}

template<class T>

MyVector<T>::MyVector(T el) {

pdata = new T[MAX\_SIZE]();

pdata[0] = el;

size = 1;

maxsize = MAX\_SIZE;

}

template<class T>

MyVector<T>::MyVector(MyVector<T>& v) {

if (v.size != 0) {

pdata = new T[v.maxsize];

for (int i = 0; i < v.size; i++) {

pdata[i] = v.pdata[i];

}

}

else pdata = NULL;

size = v.size;

maxsize = v.maxsize;

}

template<class T>

MyVector<T>::~MyVector() {

//cout << "DESTRUCTION! ~MyVector<T>" << endl;

delete[] pdata;

size = NULL;

maxsize = NULL;

}

template<class T>

void MyVector<T>::add\_element(T el) {

while (true)

if (maxsize > size + 1) {

if (size == 0) {

pdata[0] = el;

size++;

return;

}

pdata[size] = el;

size++;

break;

}

else {

resize();

}

}

template<class T>

void MyVector<T>::resize() {

while (true) {

// Увеличиваем в полтора раза

if (maxsize <= size + 1) {

maxsize \*= 1.5;

T\* buff = new T[maxsize]();

for (int i = 0; i < size; i++) {

buff[i] = pdata[i];

}

delete[] pdata;

pdata = buff;

/\*cout << "resize +: " << endl;

for (int i = 0; i < size; i++)

cout << "#: " << i << "\t" << buff[i] << endl;\*/

break;

}

// Уменьшаем в полтора раза

else if (maxsize / 2 >= size) {

maxsize /= 1.5;

T\* buff = new T[maxsize]();

for (int i = 0; i < size; i++) {

buff[i] = pdata[i];

}

delete[] pdata;

pdata = buff;

/\*cout << "resize -: " << endl;

for (int i = 0; i < size; i++)

cout << "#: " << i << "\t" << buff[i] << endl;\*/

break;

}

else if (maxsize > size) {

break;

}

}

};

template<class T>

void MyVector<T>::delete\_element(int number) {

while (true)

if (maxsize / 2 < size) {

// Устанавливаем новый размер буфера

int j = 0;

for (int i = 0; i < size; i++) {

if (i != number) {

pdata[j] = pdata[i];;

j++;

}

}

size--;

break;

}

else

resize();

}

template<class T>

T MyVector<T>::operator[](int i) {

return pdata[i];

};

template<class T>

int MyVector<T>::find(T el) {

for (int i = 0; i < size; i++) {

if (pdata[i] == el)

return i;

}

return -1;

};

template<class T>

MyVector<T>& MyVector<T>::operator=(MyVector<T> v) {

if (this != &v) {

if (v.size != 0) {

pdata = new T [v.size];

for (int i = 0; i < v.size; i++) {

pdata[i] = v.pdata[i];

}

}

size = v.size;

maxsize = v.maxsize;

}

return \*this;

}

template<class T>

void MyVector<T>::sort() {

int size\_word = NULL;

for (int i = 0; i < size; i++)

for (int j = 0; j < size - 1; j++) {

if (length(pdata[j]) > length(pdata[j + 1]) ? size\_word = length(pdata[j + 1]) : size\_word = length(pdata[j]));

for (int k = 0; k < size\_word; k++) {

if (pdata[j] > pdata[j + 1]) {

swap(j, j + 1);

break;

}

else break;

}

}

}

template<class T>

void MyVector<T>::swap(int i, int j) {

T tmp(NULL);

tmp = pdata[i];

pdata[i] = pdata[j];

pdata[j] = tmp;

}

//================================================================================================

// CHAR\*

//================================================================================================

template<>

MyVector<char\*>::MyVector();

template<>

MyVector<char\*>::MyVector(char\* el);

template<>

MyVector<char\*>::~MyVector();

template<>

MyVector<char\*>::MyVector(MyVector<char\*>& v);

template<>

void MyVector<char\*>::add\_element(char\* el);

template<>

void MyVector<char\*>::delete\_element(int number);

template<>

int MyVector<char\*>::find(char\* el);

template<>

void MyVector<char\*>::sort();

template<>

void MyVector<char\*>::resize();

//================================================================================================

// ОСОБЫЕ СЛУЧАИ

//================================================================================================

template<>

void MyVector<string>::sort();

#endif

### **MyVector.cpp**

#include "MyVector.h"

#include <iostream>

#include <string>

//using namespace std;

// Отключение предупреждении о потери данных при преобразовании с double на int (строка 27 и 31)

//#pragma warning(disable : 4244)

// CHAR\*

template<>

MyVector<char\*>::MyVector() {

pdata = new char\* [MAX\_SIZE]();

maxsize = MAX\_SIZE;

size = 0;

}

template<>

MyVector<char\*>::MyVector(char\* el) {

pdata = new char\*[MAX\_SIZE];

pdata[0] = new char[strlen(el) + 1];

strcpy\_s(pdata[0], strlen(el) + 1, el);

size = 1;

maxsize = MAX\_SIZE;

};

template<>

MyVector<char\*>::~MyVector() {

//cout << "DESTRUCTION! ~MyVector<char\*>" << endl;

if (size != 0) {

for (int i = 0; i < size; i++) {

delete[] pdata[i];

}

}

else {

delete pdata;

}

size = NULL;

maxsize = NULL;

}

template<>

MyVector<char\*>::MyVector(MyVector<char\*>& v) {

if (v.size != 0) {

pdata = new char\* [v.size];

for (int i = 0; i < v.size; i++) {

pdata[i] = new char[strlen(v.pdata[i]) + 1];

strcpy\_s(pdata[i], strlen(v.pdata[i]) + 1, v.pdata[i]);

}

}

else pdata = NULL;

size = v.size;

maxsize = v.maxsize;

}

template<>

void MyVector<char\*>::add\_element(char\* el) {

while (true)

if (maxsize > size + 1) {

if (size == 0) {

pdata[0] = new char[strlen(el) + 1];

strcpy\_s(pdata[0], strlen(el) + 1, el);

size++;

return;

}

pdata[size] = new char[strlen(el) + 1];

strcpy\_s(pdata[size], strlen(el) + 1, el);

size++;

break;

}

else {

resize();

}

};

template<>

void MyVector<char\*>::delete\_element(int number) {

while (true)

if (maxsize / 2 < size) {

// Устанавливаем новый размер буфера

char\*\* buff = new char\* [maxsize];

int j = 0;

for (int i = 0; i < size; i++) {

if (i != number) {

buff[j] = new char[strlen(pdata[i]) + 1];

strcpy\_s(buff[j], strlen(pdata[i]) + 1, pdata[i]);

j++;

}

}

// Присвоим все новые значения в текущий массив слов

pdata = buff;

size--;

break;

}

else

resize();

}

template<>

int MyVector<char\*>::find(char\* el) {

for (int i = 0; i < size; i++) {

if (!strcmp(pdata[i], el))

return i;

}

return -1;

};

template<>

void MyVector<char\*>::sort() {

int size\_word = NULL;

for (int i = 0; i < size; i++)

for (int j = 0; j < size - 1; j++) {

if (strlen(pdata[j]) > strlen(pdata[j + 1]) ? size\_word = strlen(pdata[j + 1]) : size\_word = strlen(pdata[j]));

for (int k = 0; k < size\_word; k++) {

if (pdata[j][k] > pdata[j + 1][k]) {

swap(j, j + 1);

break;

}

else break;

}

}

}

template<>

void MyVector<char\*>::resize() {

while (true) {

// Увеличиваем в полтора раза

if (maxsize <= size + 1) {

maxsize \*= 1.5;

char\*\* buff = new char\*[maxsize]();

for (int i = 0; i < size; i++) {

buff[i] = new char[strlen(pdata[i]) + 1];

strcpy\_s(buff[i], strlen(pdata[i]) + 1, pdata[i]);

//buff[i] = pdata[i];

}

delete[] pdata;

pdata = buff;

/\*cout << "resize +: " << endl;

for (int i = 0; i < size; i++)

cout << "#: " << i << "\t" << buff[i] << endl;\*/

break;

}

// Уменьшаем в полтора раза

else if (maxsize / 2 >= size) {

maxsize /= 1.5;

char\*\* buff = new char\* [maxsize]();

for (int i = 0; i < size; i++) {

buff[i] = new char[strlen(pdata[i]) + 1];

strcpy\_s(buff[i], strlen(pdata[i]) + 1, pdata[i]);

}

delete[] pdata;

pdata = buff;

/\*cout << "resize -: " << endl;

for (int i = 0; i < size; i++)

cout << "#: " << i << "\t" << buff[i] << endl;\*/

break;

}

else if (maxsize > size) {

break;

}

}

};

//================================================================================================

// ОСОБЫЕ СЛУЧАИ

//================================================================================================

template<>

void MyVector<string>::sort() {

int size\_word = NULL;

for (int i = 0; i < size; i++)

for (int j = 0; j < size - 1; j++) {

if (pdata[j].size() > pdata[j + 1].size() ? size\_word = pdata[j + 1].size() : size\_word = pdata[j].size());

for (int k = 0; k < size\_word; k++) {

if (pdata[j][k] > pdata[j + 1][k]) {

swap(j, j + 1);

break;

}

else break;

}

}

}

### **MySet.h**

//файл MySet.h - описание класса MySet

#ifndef MySet\_h

#define MySet\_h

#include <iostream>

#include <string>

#include "MyVector.h"

using namespace std;

template <typename T = string>

class MySet :public MyVector<T>

{

private:

// Перегрузка оператора присваивания

//using MyVector<T>::operator=;

public:

MySet() : MyVector<T>() {

//cout << "CONSTRUCTION" << endl;

};

MySet(T el) : MyVector<T>(el){

//cout << "CONSTRUCTION" << endl;

};

friend ostream& operator<<(ostream& out, MySet<T>& s) {

if (s.size != 0) {

out << "{ ";

for (int i = 0; i < s.size; i++) {

out << s.pdata[i] << "; ";

}

out << "}";

}

else

out << "Отсутствует" << endl;

return out;

}

// A = {1, 4, 5, 6}; B = {1, 2, 3, 4};

// {1, 4, 5, 6} + {1, 2, 3, 4} => {1, 2, 3, 4, 5, 6}

friend MySet<T> operator+(MySet<T>& s1, MySet<T>& s2) {

MySet<T> buff;

// Объединим все элементы в одно множество

for (int i = 0; i < s1.size; i++) {

buff.add\_element(s1.pdata[i]);

}

for (int i = 0; i < s2.size; i++) {

buff.add\_element(s2.pdata[i]);

}

return buff;

}

// A = {1, 4, 5, 6}; B = {1, 2, 3, 4};

// {1, 4, 5, 6} - {1, 2, 3, 4} => {5, 6} (size = 4; size = 4)

// {1, 4, 5, 6} - {1, 2, 3, 4, 5, 7} => {5, 6} (size = 4; size = 6)

friend MySet<T> operator-(MySet<T>& s1, MySet<T>& s2) {

MySet<T> buff;

for (int i = 0; i < s1.size; i++) {

buff.MyVector<T>::add\_element(s1.pdata[i]);

}

for (int i = 0; i < s1.size; i++) {

// Если совпался элемент с массива s2 с массивом buff, т.е. с массивом s1

if (buff.is\_element(s2.pdata[i])) {

buff.delete\_elemet(s2.pdata[i]);

}

}

return buff;

}

// {1, 4, 5, 6} \* {1, 2, 3, 4} => {1, 4}

friend MySet<T> operator\*(MySet<T>& s1, MySet<T>& s2) {

MySet<T> buff;

for (int i = 0; i < s2.size; i++) {

if (s1.is\_element(s2.pdata[i])) {

buff.MyVector<T>::add\_element(s2.pdata[i]);

}

}

return buff;

}

// Перегрузка оператора присваивания

MySet<T>& operator=(MySet<T> v);

// №2 Конструктор копирования

MySet(const MySet<T>& v);

// Если совпадало, то true

bool operator==(MySet<T>& s) {

if (this->size == s.size && this->maxsize == s.maxsize) {

for (int i = 0; i < this->size; i++)

if (this->pdata[i] != s.pdata[i]) return false;

return true;

}

return false;

}

MySet<T>& operator+=(MySet<T>& s) {

\*this = \*this + s;

return \*this;

}

MySet<T>& operator-=(MySet<T>& s) {

\*this = \*this - s;

return \*this;

}

MySet<T>& operator\*=(MySet<T>& s) {

\*this = \*this \* s;

return \*this;

}

// Добавление строки в множество, если её там ещё нет.

void add\_element(T el) {

if (!is\_element(el))

this->MyVector<T>::add\_element(el);

};

// Удаление элемента из множества, если он там есть.

void delete\_elemet(T el) {

if (MyVector<T>::find(el) != -1) {

this->MyVector<T>::delete\_element(MyVector<T>::find(el));

}

else cout << "Данный элемент: [ " << el << " ] не существует" << endl;

}

// Метод доступа, который даёт true, если строка-параметр есть в множестве, иначе даёт false.

bool is\_element(T el) {

for (int i = 0; i < this->size; i++) {

if (i == MyVector<T>::find(el))

//if (this->pdata[i] == el)

return true;

}

return false;

};

};

template<class T>

MySet<T>& MySet<T>::operator=(MySet<T> v) {

if (this != &v) {

if (v.size != 0) {

this->pdata = new T[v.size];

for (int i = 0; i < v.size; i++) {

this->pdata[i] = v.pdata[i];

}

}

this->size = v.size;

this->maxsize = v.maxsize;

}

return \*this;

}

template<class T>

MySet<T>::MySet(const MySet<T>& v) {

if (v.size != 0) {

this->pdata = new T[v.size];

for (int i = 0; i < v.size; i++) {

this->pdata[i] = v.pdata[i];

}

}

this->size = v.size;

this->maxsize = v.maxsize;

}

template<>

MySet<char\*>& MySet<char\*>::operator=(MySet<char\*> v);

template<>

MySet<char\*>::MySet(const MySet<char\*>& v);

#endif

### **MySet.cpp**

#include "MySet.h"

#include <iostream>

#include <string>

using namespace std;

template<>

MySet<char\*>& MySet<char\*>::operator=(MySet<char\*> v) {

if (this != &v) {

if (v.size != 0) {

pdata = new char\* [v.size];

for (int i = 0; i < v.size; i++) {

pdata[i] = new char[strlen(v.pdata[i]) + 1];

strcpy\_s(pdata[i], strlen(v.pdata[i]) + 1, v.pdata[i]);

}

}

size = v.size;

maxsize = v.maxsize;

}

return \*this;

}

template<>

MySet<char\*>::MySet(const MySet<char\*>& v) {

if (v.size != 0) {

pdata = new char\* [v.size];

for (int i = 0; i < v.size; i++) {

pdata[i] = new char[strlen(v.pdata[i]) + 1];

strcpy\_s(pdata[i], strlen(v.pdata[i]) + 1, v.pdata[i]);

}

}

size = v.size;

maxsize = v.maxsize;

}

### **MyStack.h**

#pragma once

#ifndef MyStack\_h

#define MyStack\_h

#include <iostream>

using namespace std;

//Шаблонный класс ListNode (узел односвязного списка)

template <class INF, class FRIEND>

//узел списка

class ListNode

{

private:

//информационная часть узла

INF d;

//указатель на следующий узел списка

ListNode\* next;

//конструктор

ListNode() { next = NULL; }

ListNode(int el = 0) { d = el; next = nullptr; }

~ListNode() {

cout << "ДЕСТРУКТОР! ~ListNode" << endl;

}

INF get\_d() { return d; }

void set\_d(INF D) { d = D; }

int size() {

int size = 1;

ListNode<INF, FRIEND>\* n\_tmp = next;

while (n\_tmp) {

n\_tmp = n\_tmp->get\_next();

size++;

}

return size;

}

ListNode\* get\_next() { return next; }

void set\_next(ListNode\* NEXT) {

this->next = NEXT;

}

friend FRIEND;

};

//Шаблонный класс MyStack на основе односвязного списка.

template <class INF>

class MyStack

{

typedef class ListNode < INF, MyStack <INF> > Node;

Node\* top;

public:

// конструктор

MyStack() { top = nullptr; }

// Конструктор копирования

MyStack(MyStack& obj) {

if (!obj.empty()) {

const int size = obj.size();

ListNode<INF, MyStack<INF>>\* buff = new ListNode<INF, MyStack<INF>>(\*obj.top);

ListNode<INF, MyStack<INF>>\* tmp = new ListNode<INF, MyStack<INF>>(buff->get\_d());

for (int i = 0; i < size; i++) {

// Адрес следующего узла

tmp->set\_next(top);

top = tmp;

// Адрес следующего узла буфера

if (buff->get\_next() == nullptr) break;

buff = new ListNode<INF, MyStack<INF>>(\*buff->get\_next());

tmp = new ListNode<INF, MyStack<INF>>(buff->get\_d());

}

this->sort\_ub();

}

}

// Оператор присваивания

MyStack& operator=(MyStack& stack) {

// Если стек пустой, тогда возвращает нулевое значение

if (stack.empty())

return \*this;

if (this != &stack) {

this->clear();

const int size = stack.size();

ListNode<INF, MyStack<INF>>\* buff = new ListNode<INF, MyStack<INF>>(\*stack.top);

ListNode<INF, MyStack<INF>>\* tmp = new ListNode<INF, MyStack<INF>>(buff->get\_d());

for (int i = 0; i < size; i++) {

// Адрес следующего узла

tmp->set\_next(top);

top = tmp;

// Адрес следующего узла буфера

if (buff->get\_next() == nullptr) break;

buff = new ListNode<INF, MyStack<INF>>(\*buff->get\_next());

tmp = new ListNode<INF, MyStack<INF>>(buff->get\_d());

}

this->sort\_ub();

}

return \*this;

}

// освободить динамическую память

~MyStack() {

cout << "ДЕСТРУКТОР! ~Stack" << endl;

if (this == nullptr) return;

while (this != nullptr) {

// top - адрес на первый узел

ListNode<INF, MyStack<INF>>\* tmp = top;

if (tmp != NULL) {

top = top->get\_next();

// tmp - текущий адрес

delete tmp;

}

else return;

}

}

// стек пустой? Если пустой, то true

bool empty() {

if (top == NULL) return true;

else return false;

}

// добавить узел в вершину стека

void push(INF el) {

//ListNode<INF, MyStack<INF>>\* tmp = new ListNode(el);

ListNode<INF, MyStack<INF>>\* tmp = new ListNode<INF, MyStack<INF>>(el);

if (top == nullptr) {

top = tmp;

}

else {

// Адрес следующего узла

tmp->set\_next(top);

top = tmp;

}

}

// удалить узел из вершины стека

bool pop() {

if (top == nullptr) return false;

else {

ListNode<INF>\* tmp = top;

top = top->get\_next();

delete tmp;

return true;

}

}

//считать информацию из вершины стека

INF top\_inf() { return top->get\_d(); }

Node\* get\_next() { return top->get\_next(); }

void clear() {

if (this == nullptr) return;

while (this != nullptr) {

// top - адрес на первый узел

ListNode<INF, MyStack<INF>>\* tmp = top;

if (tmp != nullptr) {

top = top->get\_next();

// tmp - текущий адрес

delete tmp;

}

else return;

}

}

// Печать

void print() {

ListNode<INF, MyStack<INF>>\* tmp = top;

if (top == nullptr) return;

while (tmp) {

cout << tmp->get\_d();

tmp = tmp->get\_next();

if (tmp != NULL)

cout << " \* ";

}

cout << endl;

}

// Определение размеров

int size() {

// создаем буфер для восстановления

ListNode<INF, MyStack<INF>>\* buff = top;

int size = 1;

if (top == nullptr) return 0;

while (top->get\_next() != nullptr) {

// check - адрес на первый узел

ListNode<INF, MyStack<INF>>\* tmp = top;

top = top->get\_next();

size++;

}

top = buff;

return size;

}

// Сортировка в возрастающем порядке

void sort\_vos() {

if (top == nullptr) return;

// создаем буфер для восстановления

ListNode<INF, MyStack<INF>>\* buff = top;

int size = top->size();

INF\* arr = new INF[size];

ListNode<INF, MyStack<INF>>\* tmp = top;

for (int i = 0; i < size; i++) {

arr[i] = tmp->get\_d();

//cout << "arr [ " << i << " ] = " << arr[i] << endl;

tmp = tmp->get\_next();

}

cout << endl;

// Сортируем по возрастанию

INF arr\_tmp = NULL;

for (int i = 0; i < size; i++)

for (int j = 0; j < size - 1; j++)

if (arr[j] < arr[j + 1]) {

arr\_tmp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr\_tmp;

}

/\*cout << "Проверка после сортировки" << endl;

for (int i = 0; i < size; i++)

cout << "arr [ " << i << " ] = " << arr[i] << endl;

cout << endl;\*/

ListNode<INF, MyStack<INF>>\* null\_stack = nullptr;

for (int i = 0; i < size; i++) {

ListNode<INF, MyStack<INF>>\* new\_stack = new ListNode<INF, MyStack<INF>>(arr[i]);

if (null\_stack == nullptr) {

null\_stack = new\_stack;

}

else {

// Адрес следующего узла

new\_stack->set\_next(null\_stack);

null\_stack = new\_stack;

}

}

// Присвоим новый отсортированный стек в текущий стек

top = null\_stack;

}

// Сортировка в убывающем порядке

void sort\_ub() {

if (top == nullptr) return;

// создаем буфер для восстановления

ListNode<INF, MyStack<INF>>\* buff = top;

int size = top->size();

INF\* arr = new INF[size];

ListNode<INF, MyStack<INF>>\* tmp = top;

for (int i = 0; i < size; i++) {

arr[i] = tmp->get\_d();

//cout << "arr [ " << i << " ] = " << arr[i] << endl;

tmp = tmp->get\_next();

}

cout << endl;

// Сортируем по возрастанию

INF arr\_tmp = NULL;

for (int i = 0; i < size; i++)

for (int j = 0; j < size - 1; j++)

if (arr[j] > arr[j + 1]) {

arr\_tmp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr\_tmp;

}

/\*cout << "Проверка после сортировки" << endl;

for (int i = 0; i < size; i++)

cout << "arr [ " << i << " ] = " << arr[i] << endl;

cout << endl;\*/

ListNode<INF, MyStack<INF>>\* null\_stack = nullptr;

for (int i = 0; i < size; i++) {

ListNode<INF, MyStack<INF>>\* new\_stack = new ListNode<INF, MyStack<INF>>(arr[i]);

if (null\_stack == nullptr) {

null\_stack = new\_stack;

}

else {

// Адрес следующего узла

new\_stack->set\_next(null\_stack);

null\_stack = new\_stack;

}

}

// Присвоим новый отсортированный стек в текущий стек

top = null\_stack;

}

};

void Multipliers(int n, MyStack<int>& stack);

#endif

### **MyStack.cpp**

#include "MyStack.h"

#include <iostream>

using namespace std;

void Multipliers(int n, MyStack<int>& stack) {

int div = 2;

const int value = n;

if (n > 1) {

while (n > 1) {

while (n % div == 0) {

n = n / div;

stack.push(div);

}

if (div == 2) div++;

else div = div + 2;

}

}

else {

stack.push(n);

}

cout << value << " = ";

stack.print();

// Сортировка в возрастающем порядке

stack.sort\_vos();

cout << value << " = ";

stack.print();

cout << endl;

}

### **Practium.cpp**

// Practium.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <string>

#include "MyStack.h"

#include "MyVector.h"

#include "MySet.h"

#pragma warning(disable : 4996)

using namespace std;

int main()

{

system("chcp 1251 >> null");

while (true) {

system("cls");

cout << "Введите номер пункта для выполнения заданий:" << endl;

cout << "№1 Разложение на простые множители целого числа N с применением stack" << endl;

cout << "№2 Множество, выполняющее несколько операций над множествами" << endl;

cout << "№0 Выход с программы." << endl;

int sw;

cout << "Номер пункта: ";

cin >> sw;

switch (sw)

{

case 1:

{

system("cls");

cout << "№1 Разложение на простые множители целого числа N с применением stack" << endl;

cout << "Введите любое целое число для разложения" << endl;

cout << "Пример, пусть дан число N = 3960" << endl

<< "Разложили его так (в убывающем порядке):\n\t3960 = 11 \* 5 \* 3 \* 3 \* 2 \* 2 \* 2" << endl

<< "Разложили его так (в возрастающем порядке):\n\t3960 = 2 \* 2 \* 2 \* 3 \* 3 \* 5 \* 11" << endl;

cout << endl;

int value = NULL;

MyStack<int> obj;

cout << "Введите любое число для разложения" << endl;

cout << "Число: ";

cin >> value;

cout << endl;

Multipliers(value, obj);

MyStack<int> obj1(obj), obj2;

obj2 = obj;

cout << "Проверка конструктора копирования и перегрузка присваивания" << endl;

cout << "obj1(obj): ";

obj1.print();

cout << "obj2 = obj: ";

obj2.print();

system("pause");

break;

} case 2:

{

system("cls");

cout << "№2 Множество, выполняющее несколько операций над множествами" << endl;

MyVector<string> v("Hello!");

cout << v << endl;

cout << endl;

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

cout << "Вектор v: " << v << endl;

cout << endl;

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

cout << "Вектор v: " << v << endl;

cout << endl;

cout << "find: " << v.find("Привет!") << endl;

cout << endl;

MyVector<> v1 = v;

MyVector<> v2(v);

cout << "Вектор v1: " << v1 << endl;

cout << "Вектор v2(v): " << v2 << endl;

for (int i = 0; i < MAX\_SIZE; i++)

v1.delete\_element(0);

cout << "Вектор v1: " << v1 << endl;

cout << endl;

cout << "==============================================================" << endl;

cout << "Множество типа string" << endl;

cout << "==============================================================" << endl;

cout << endl;

system("pause");

{

MySet<> s("Yes");

MySet<> s1, s2;

s.add\_element("Привет!");

s.add\_element("No");

char\* str = (char\*)"Hello!";

s.add\_element(str);

cout << s << endl;

cout << "Множество s: " << s << endl;

//cout << endl;

s1.add\_element("Cat");

s1.add\_element("No");

s1.add\_element("Привет!");

cout << "Множество s1: " << s1 << endl;

//cout << endl;

s2 = s1 - s;

cout << "Множество s2 = s1 - s: " << s2 << endl;

cout << endl;

cout << "Множество s1: " << s1 << endl;

cout << "Множество s: " << s << endl;

s2 = s - s1;

cout << "Множество s2 = s - s1: " << s2 << endl;

cout << endl;

cout << "Множество s1: " << s1 << endl;

cout << "Множество s: " << s << endl;

s2 = s1 + s;

cout << "Множество s2 = s1 + s: " << s2 << endl;

cout << endl;

cout << "Множество s1: " << s1 << endl;

cout << "Множество s: " << s << endl;

s2 = s1 \* s;

cout << "Множество s2 = s1 \* s: " << s2 << endl;

cout << endl;

cout << "Множество s1: " << s1 << endl;

cout << "Множество s: " << s << endl;

MySet<> s3 = s2;

MySet<> s4(s2);

cout << "Множество s3 = s2: " << s3 << endl;

cout << "Множество s4(s2): " << s4 << endl;

cout << endl;

if (s3 == s2)

cout << "Множество s3 = s2\n";

else

cout << "Множество s3 != s2\n";

if (s3 == s1)

cout << "Множество s3 = s1\n";

else

cout << "Множество s3 != s1\n";

if (s1 == s3)

cout << "Множество s1 = s3\n";

else

cout << "Множество s1 != s3\n";

cout << endl;

cout << "Множество s2: " << s2 << endl;

cout << "Множество s: " << s << endl;

s2 += s;

cout << "Множество s2 += s (s2 = s2 + s): " << s2 << endl;

cout << endl;

cout << "Множество s1: " << s1 << endl;

cout << "Множество s: " << s << endl;

s1 -= s;

cout << "Множество s1 -= s (s1 = s1 - s): " << s1 << endl;

cout << endl;

cout << "Множество s2: " << s2 << endl;

cout << "Множество s1: " << s1 << endl;

s2 \*= s1;

cout << "Множество s2 \*= s1 (s2 = s2 \* s1): " << s2 << endl;

cout << endl;

}

cout << endl;

cout << "==============================================================" << endl;

cout << "Множество типа int" << endl;

cout << "==============================================================" << endl;

cout << endl;

system("pause");

{

MySet<int> i(0);

MySet<int> i1, i2;

i.add\_element(1);

i.add\_element(3);

i.add\_element(5);

i.add\_element(7);

cout << "Множество i: " << i << endl;

//cout << endl;

i1.add\_element(0);

i1.add\_element(2);

i1.add\_element(4);

i1.add\_element(5);

i1.add\_element(6);

i1.add\_element(8);

cout << "Множество i1: " << i1 << endl;

//cout << endl;

i2 = i1 - i;

cout << "Множество i2 = i1 - i: " << i2 << endl;

cout << endl;

cout << "Множество i1: " << i1 << endl;

cout << "Множество i: " << i << endl;

i2 = i - i1;

cout << "Множество i2 = i - i1: " << i2 << endl;

cout << endl;

cout << "Множество i1: " << i1 << endl;

cout << "Множество i: " << i << endl;

i2 = i1 + i;

cout << "Множество i2 = i1 + i: " << i2 << endl;

i2.sort();

cout << "Множество i2 после сортировки: " << i2 << endl;

cout << endl;

cout << "Множество i1: " << i1 << endl;

cout << "Множество i: " << i << endl;

i2 = i1 \* i;

cout << "Множество i2 = i1 \* i: " << i2 << endl;

cout << endl;

cout << "Множество i1: " << i1 << endl;

cout << "Множество i: " << i << endl;

MySet<int> i3 = i2;

MySet<int> i4(i2);

cout << "Множество i3 = s2: " << i3 << endl;

cout << "Множество i4(i2): " << i4 << endl;

cout << endl;

if (i3 == i2)

cout << "Множество i3 = i2\n";

else

cout << "Множество i3 != i2\n";

if (i3 == i1)

cout << "Множество i3 = i1\n";

else

cout << "Множество i3 != i1\n";

if (i1 == i3)

cout << "Множество i1 = i3\n";

else

cout << "Множество i1 != i3\n";

cout << endl;

cout << "Множество i2: " << i2 << endl;

cout << "Множество i: " << i << endl;

i2 += i;

cout << "Множество i2 += i (i2 = i2 + i): " << i2 << endl;

cout << endl;

cout << "Множество i1: " << i1 << endl;

cout << "Множество i: " << i << endl;

i1 -= i;

cout << "Множество i1 -= i (i1 = i1 - i): " << i1 << endl;

cout << endl;

cout << "Множество i2: " << i2 << endl;

cout << "Множество i1: " << i1 << endl;

i2 \*= i1;

cout << "Множество i2 \*= i1 (i2 = i2 \* i1): " << i2 << endl;

cout << endl;

}

cout << endl;

cout << "==============================================================" << endl;

cout << "Множество типа char\*" << endl;

cout << "==============================================================" << endl;

cout << endl;

system("pause");

{

MySet<char\*> c((char\*)"First");

MySet<char\*> c1, c2;

c.add\_element((char\*)"second");

c.add\_element((char\*)"third");

c.add\_element((char\*)"four");

c.add\_element((char\*)"five");

cout << "Множество c: " << c << endl;

//cout << endl;

c1.add\_element((char\*)"six");

c1.add\_element((char\*)"eleven");

c1.add\_element((char\*)"five");

c1.add\_element((char\*)"four");

c1.add\_element((char\*)"ten");

c1.add\_element((char\*)"eleven");

cout << "Множество c1: " << c1 << endl;

//cout << endl;

c2 = c1 - c;

cout << "Множество c2 = c1 - c: " << c2 << endl;

cout << endl;

cout << "Множество c1: " << c1 << endl;

cout << "Множество c: " << c << endl;

c2 = c - c1;

cout << "Множество c2 = c - c1: " << c2 << endl;

cout << endl;

cout << "Множество c1: " << c1 << endl;

cout << "Множество c: " << c << endl;

c2 = c1 + c;

cout << "Множество c2 = c1 + c: " << c2 << endl;

cout << endl;

cout << "Множество c1: " << c1 << endl;

cout << "Множество c: " << c << endl;

c2 = c1 \* c;

cout << "Множество c2 = c1 \* c: " << c2 << endl;

cout << endl;

cout << "Множество c1: " << c1 << endl;

cout << "Множество c: " << c << endl;

MySet<char\*> c3 = c2;

MySet<char\*> c4(c2);

cout << "Множество c3 = s2: " << c3 << endl;

cout << "Множество c4(c2): " << c4 << endl;

cout << endl;

if (c3 == c2)

cout << "Множество c3 = c2\n";

else

cout << "Множество c3 != c2\n";

if (c3 == c1)

cout << "Множество c3 = c1\n";

else

cout << "Множество c3 != c1\n";

if (c1 == c3)

cout << "Множество c1 = c3\n";

else

cout << "Множество c1 != c3\n";

cout << endl;

cout << "Множество c2: " << c2 << endl;

cout << "Множество c: " << c << endl;

c2 += c;

cout << "Множество c2 += c (c2 = c2 + c): " << c2 << endl;

cout << endl;

cout << "Множество c1: " << c1 << endl;

cout << "Множество c: " << c << endl;

c1 -= c;

cout << "Множество c1 -= c (c1 = c1 - c): " << c1 << endl;

cout << endl;

cout << "Множество c2: " << c2 << endl;

cout << "Множество c1: " << c1 << endl;

c2 \*= c1;

cout << "Множество c2 \*= c1 (c2 = c2 \* c1): " << c2 << endl;

cout << endl;

}

system("pause");

break;

}

case 0:

{

exit(1);

}

default:

{

cout << "Введенный Вами пункт не существует. Возврат в меню!";

break;

}

}

}

}

## **Результаты работы программы:**

№1 Разложение на простые множители целого числа N с применением stack

Введите любое целое число для разложения

Пример, пусть дан число N = 3960

Разложили его так (в убывающем порядке):

3960 = 11 \* 5 \* 3 \* 3 \* 2 \* 2 \* 2

Разложили его так (в возрастающем порядке):

3960 = 2 \* 2 \* 2 \* 3 \* 3 \* 5 \* 11

Введите любое число для разложения

Число: 3960

3960 = 11 \* 5 \* 3 \* 3 \* 2 \* 2 \* 2

3960 = 2 \* 2 \* 2 \* 3 \* 3 \* 5 \* 11

Проверка конструктора копирования и перегрузка присваивания

obj1(obj): 11 \* 5 \* 3 \* 3 \* 2 \* 2 \* 2

obj2 = obj: 11 \* 5 \* 3 \* 3 \* 2 \* 2 \* 2

Для продолжения нажмите любую клавишу . . .

№2 Множество, выполняющее несколько операций над множествами

Hello!

Вектор v: Hello! Привет! Привет! Привет! Привет! Привет!

Вектор v: Hello! Привет! Привет! Привет! Привет! Привет! Привет! Привет! Привет!

find: 1

Вектор v1: Hello! Привет! Привет! Привет! Привет! Привет! Привет! Привет! Привет!

Вектор v2(v): Hello! Привет! Привет! Привет! Привет! Привет! Привет! Привет! Привет!

Вектор v1: Привет! Привет! Привет! Привет!

==============================================================

Множество типа string

==============================================================

Для продолжения нажмите любую клавишу . . .

{ Yes; Привет!; No; Hello!; }

Множество s: { Yes; Привет!; No; Hello!; }

Множество s1: { Cat; No; Привет!; }

Множество s2 = s1 - s: { Cat; }

Множество s1: { Cat; No; Привет!; }

Множество s: { Yes; Привет!; No; Hello!; }

Множество s2 = s - s1: { Yes; Hello!; }

Множество s1: { Cat; No; Привет!; }

Множество s: { Yes; Привет!; No; Hello!; }

Множество s2 = s1 + s: { Cat; No; Привет!; Yes; Hello!; }

Множество s1: { Cat; No; Привет!; }

Множество s: { Yes; Привет!; No; Hello!; }

Множество s2 = s1 \* s: { Привет!; No; }

Множество s1: { Cat; No; Привет!; }

Множество s: { Yes; Привет!; No; Hello!; }

Множество s3 = s2: { Привет!; No; }

Множество s4(s2): { Привет!; No; }

Множество s3 = s2

Множество s3 != s1

Множество s1 != s3

Множество s2: { Привет!; No; }

Множество s: { Yes; Привет!; No; Hello!; }

Множество s2 += s (s2 = s2 + s): { Привет!; No; Yes; Hello!; }

Множество s1: { Cat; No; Привет!; }

Множество s: { Yes; Привет!; No; Hello!; }

Множество s1 -= s (s1 = s1 - s): { Cat; }

Множество s2: { Привет!; No; Yes; Hello!; }

Множество s1: { Cat; }

Множество s2 \*= s1 (s2 = s2 \* s1): Отсутствует

==============================================================

Множество типа int

Для продолжения нажмите любую клавишу . . .

Множество i: { 0; 1; 3; 5; 7; }

Множество i1: { 0; 2; 4; 5; 6; 8; }

Множество i2 = i1 - i: { 2; 4; 6; 8; }

Множество i1: { 0; 2; 4; 5; 6; 8; }

Множество i: { 0; 1; 3; 5; 7; }

Множество i2 = i - i1: { 1; 3; 7; }

Множество i1: { 0; 2; 4; 5; 6; 8; }

Множество i: { 0; 1; 3; 5; 7; }

Множество i2 = i1 + i: { 0; 2; 4; 5; 6; 8; 1; 3; 7; }

Множество i2 после сортировки: { 0; 1; 2; 3; 4; 5; 6; 7; 8; }

Множество i1: { 0; 2; 4; 5; 6; 8; }

Множество i: { 0; 1; 3; 5; 7; }

Множество i2 = i1 \* i: { 0; 5; }

Множество i1: { 0; 2; 4; 5; 6; 8; }

Множество i: { 0; 1; 3; 5; 7; }

Множество i3 = s2: { 0; 5; }

Множество i4(i2): { 0; 5; }

Множество i3 = i2

Множество i3 != i1

Множество i1 != i3

Множество i2: { 0; 5; }

Множество i: { 0; 1; 3; 5; 7; }

Множество i2 += i (i2 = i2 + i): { 0; 5; 1; 3; 7; }

Множество i1: { 0; 2; 4; 5; 6; 8; }

Множество i: { 0; 1; 3; 5; 7; }

Множество i1 -= i (i1 = i1 - i): { 2; 4; 6; 8; }

Множество i2: { 0; 5; 1; 3; 7; }

Множество i1: { 2; 4; 6; 8; }

Множество i2 \*= i1 (i2 = i2 \* i1): Отсутствует

==============================================================

Множество типа char\*

=============================================================

Для продолжения нажмите любую клавишу . . .

Множество c: { First; second; third; four; five; }

Множество c1: { six; eleven; five; four; ten; }

Множество c2 = c1 - c: { six; eleven; ten; }

Множество c1: { six; eleven; five; four; ten; }

Множество c: { First; second; third; four; five; }

Множество c2 = c - c1: { First; second; third; }

Множество c1: { six; eleven; five; four; ten; }

Множество c: { First; second; third; four; five; }

Множество c2 = c1 + c: { six; eleven; five; four; ten; First; second; third; }

Множество c1: { six; eleven; five; four; ten; }

Множество c: { First; second; third; four; five; }

Множество c2 = c1 \* c: { four; five; }

Множество c1: { six; eleven; five; four; ten; }

Множество c: { First; second; third; four; five; }

Множество c3 = s2: { four; five; }

Множество c4(c2): { four; five; }

Множество c3 != c2

Множество c3 != c1

Множество c1 != c3

Множество c2: { four; five; }

Множество c: { First; second; third; four; five; }

Множество c2 += c (c2 = c2 + c): { four; five; First; second; third; }

Множество c1: { six; eleven; five; four; ten; }

Множество c: { First; second; third; four; five; }

Множество c1 -= c (c1 = c1 - c): { six; eleven; ten; }

Множество c2: { four; five; First; second; third; }

Множество c1: { six; eleven; ten; }

Множество c2 \*= c1 (c2 = c2 \* c1): Отсутствует

Для продолжения нажмите любую клавишу . . .

Введите номер пункта для выполнения заданий:

№1 Разложение на простые множители целого числа N с применением stack

№2 Множество, выполняющее несколько операций над множествами

№0 Выход с программы.

Номер пункта: 0

# **Заключение**

В рамках данной практики был разработан программный продукт, в парадигме объектно-ориентированного программирования, и получены навыки по его созданию.

# **Список использованных источников**

1. <http://cppstudio.com/>