**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №5

# «Наследование. Множества»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-21Б |  | преподаватель каф. ИУ5 |
| Анцифров Никита |  |  |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

**Постановка задачи**

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.

При разработке воспользоваться методическими указаниями лабораторной работы.

**Описание входных, выходных и вспомогательных данных**

**MyVector** v, v1 – классы векторов

**MySet** s, s1, s2, s3 – классы множеств

**char\*** str – переменная для хранения строки

**char\*** el – переменная для элемента

**int** maxsz – переменная для размера вектора

**MyVector&** v – ссылка на объект класса векторов

**int** i – счетчик цикла или в некоторых случаях номер элемента

**int** k, j – счетчики циклов

**char\*\*** temp\_arr – временный массив для хранения данных

**MySet** s\_temp – временный класс для обработки данных

**int** a, b, c – переменные для метода половинного деления

**Описание прототипов функций**

**ostream&** operator<<(**ostream&** out, **MyVector&** v) – перегрузка оператора <<, функция для вывода объекта класса векторов

**ostream&** operator<<(**ostream&** out, **MySet&** s) – перегрузка оператора << для вывода множества

**MySet** operator+(**MySet&** s1, **MySet&** s2) – перегрузка оператора + для выполнения операции над множеством (принимает ссылки на два множества, возвращает множество)

**MySet** operator-(**MySet&** s1, **MySet&** s2) – перегрузка оператора - для выполнения операции над множеством (принимает ссылки на два множества, возвращает множество)

**MySet** operator\*(**MySet&** s1, **MySet&** s2) – перегрузка оператора \* для выполнения операции над множеством (принимает ссылки на два множества, возвращает множество)

**Разработка интерфейса класса**

Класс:

**class** MyVector – класс для хранения векторов

Поля класса:

**int** maxsize – размер вектора

**int** size – количество элементов в векторе

**char\*\*** pdata – указатель, содержащий адрес динамического массива элементов

Методы класса:

**void** add\_element(**char\*** el) – функция для добавления элемента (принимает элемент)

**bool** delete\_element(**int** i) – функция для удаления элемента (принимает номер этого элемента, возвращает 0, если элемент не удален, и 1, если он удален)

**char\*** operator[](**int** i) – перегрузка оператора [] для вывода элемента

**void** sort() – функция сортировки

**int** Size() – функция для получения количества элементов в векторе (возвращает количество элементов в векторе)

**int** Maxsize() – функция для получения размера вектора (возвращает размер вектора)

**int** find(**char\*** el) – функция поиска элемента (принимает элемент, возвращает номер элемента)

**MyVector&** operator=(**MyVector&** v) – перегрузка оператора = (принимает ссылку на объект класса векторов, возвращает также ссылку на объект класса векторов)

**void** resize() – функция для изменения размера вектора

Дружественные функции:

friend **ostream&** operator<<(**ostream&** out, **MyVector&** v)

Класс:

**class** MySet:public MyVector – класс для множеств

Методы класса:

**bool** operator==(**MySet&** s) – перегрузка оператора == для сравнения классов

**MySet&** operator+=(**MySet&** s) – перегрузка оператора +- для выполнения операций над множествами

**MySet&** operator-=(**MySet&** s) – перегрузка оператора =+ для выполнения операций над множествами

**MySet&** operator\*=(**MySet&** s) – перегрузка оператора \*= для выполнения операций над множествами

**void** add\_element(**char\*** el) – функция для добавления элемента (принимает элемент)

**void** delete\_element(**char\*** el) – функция для удаления элемента (принимает элемент)

**bool** is\_element(**char\*** el) – функция проверки нахождения элемента в множестве

Дружественные функции:

friend **ostream&** operator<<(**ostream&** out, **MySet&** s)

friend **MySet** operator+(**MySet&** s1, **MySet&** s2)

friend **MySet** operator-(**MySet&** s1, **MySet&** s2)

friend **MySet** operator\*(**MySet&** s1, **MySet&** s2)

**Алгоритм**

**main**



**MyVector – add\_element**



**MyVector - delete\_element**

**MyVector – sort**



**MyVector – find**



**MyVector – resize**



**MySet - delete\_element**



**MySet - is\_element**



**Текст программы**

**main.cpp**

#include <iostream>

#include "MyVector.h"

#include "MySet.h"

#include "proto.h"

using namespace std;

void main()

{

setlocale(0,"russian");

MyVector v("Hello!");

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;

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

cout << "Вектор v: " << v << endl;

MyVector v1=v;

cout << "Вектор v1: " << v1 << endl;

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

v1.delete\_element(0);

cout << "Вектор v1: " << v1 << endl;

MySet s("Yes"), s1, s2;

s.add\_element("Привет!");

s.add\_element("No");

char\* str="Hello!";

s.add\_element(str);

cout << "Множество s: " << s << endl;

s1.add\_element("Cat");

s1.add\_element("No");

s1.add\_element("Привет!");

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

s2=s1-s;

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

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

cout << "Множество s: " << s << endl;

s2=s-s1;

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

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

cout << "Множество s: " << s << endl;

s2=s1+s;

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

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

cout << "Множество s: " << s << endl;

s2=s1\*s;

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

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

cout << "Множество s: " << s << endl;

MySet s3=s2;

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

if (s3==s2)

cout << "Множество s3=s2" << endl;

else

cout << "Множество s3!=s2" << endl;

if(s3==s1)

cout << "Множество s3=s1" << endl;

else

cout << "Множество s3!=s1" << endl;

if(s1==s3)

cout << "Множество s1=s3" << endl;

else

cout << "Множество s1!=s3" << endl;

system("pause");

}

**proto.h**

#include "MySet.h"

ostream& operator<<(ostream& out, MySet& s);

MySet operator+(MySet& s1, MySet& s2);

MySet operator-(MySet& s1, MySet& s2);

MySet operator\*(MySet& s1, MySet& s2);

ostream& operator<<(ostream& out, MyVector& v);

**MyVector.h**

#ifndef MYVECTOR\_H

#define MYVECTOR\_H

#include <iostream>

using namespace std;

const int MAX\_SIZE=5;

class MyVector

{

public:

MyVector(char \*el=NULL, int maxsz=MAX\_SIZE);

MyVector(MyVector& v);

~MyVector();

void add\_element(char\* el);

bool delete\_element(int i);

char\* operator[](int i);

void sort();

int Size();

int Maxsize();

int find(char\* el);

MyVector& operator=(MyVector& v);

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

protected:

int maxsize;

int size;

char\*\* pdata;

private:

void resize();

};

#endif

**MyVector.cpp**

#include "MyVector.h"

MyVector::MyVector(char\* el, int maxsz)

{

if (!el)

{

maxsize=MAX\_SIZE;

size=0;

pdata=new char\*;

}

else

{

maxsize=maxsz;

size=1;

pdata=new char\*[maxsize];

pdata[0]=el;

}

}

MyVector::MyVector(MyVector& v)

{

maxsize=v.maxsize;

size=v.size;

pdata=new char\*[maxsize];

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

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

}

MyVector::~MyVector()

{

delete pdata;

}

void MyVector::add\_element(char\* el)

{

if (size>maxsize)

{

maxsize\*=1.5;

resize();

}

pdata[size]=el;

size++;

}

bool MyVector::delete\_element(int i)

{

if (size<1)

return false;

else

{

for (int k=i; k<maxsize; k++)

pdata[k]=pdata[k+1];

size--;

if (size<maxsize/2)

{

maxsize/=1.5;

resize();

}

return true;

}

}

char\* MyVector::operator[](int i)

{

return pdata[i];

}

void MyVector::sort()

{

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

{

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

{

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

swap(pdata[j], pdata[j+1]);

}

}

}

int MyVector::Size()

{

return size;

}

int MyVector::Maxsize()

{

return maxsize;

}

int MyVector::find(char\* el)

{

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

{

if (pdata[i]==el)

return i;

}

return -1;

}

MyVector& MyVector::operator=(MyVector& v)

{

maxsize=v.maxsize;

size=v.size;

pdata=new char\*[maxsize];

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

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

return \*this;

}

ostream& operator <<(ostream& out, MyVector& v)

{

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

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

return out;

}

void MyVector::resize()

{

char\*\* temp\_arr=new char\*[maxsize];

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

temp\_arr[i]=pdata[i];

pdata=temp\_arr;

}

**MySet.h**

#ifndef MYSET\_H

#define MYSET\_H

#include <iostream>

#include "MyVector.h"

using namespace std;

class MySet:public MyVector

{

public:

MySet(char\* el=NULL):MyVector(el){};

friend ostream& operator<<(ostream& out, MySet& s);

friend MySet operator+(MySet& s1, MySet& s2);

friend MySet operator-(MySet& s1, MySet& s2);

friend MySet operator\*(MySet& s1, MySet& s2);

bool operator==(MySet& s);

MySet& operator+=(MySet& s);

MySet& operator-=(MySet& s);

MySet& operator\*=(MySet& s);

void add\_element(char\* el);

void delete\_element(char\* el);

bool is\_element(char\* el);

};

#endif

**MySet.cpp**

#include "MySet.h"

ostream& operator<<(ostream& out, MySet& s)

{

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

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

return out;

}

MySet operator+(MySet& s1, MySet& s2)

{

MySet s\_temp(s1);

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

{

if (s\_temp.is\_element(s2.pdata[i])==0)

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

}

return s\_temp;

}

MySet operator-(MySet& s1, MySet& s2)

{

MySet s\_temp(s1);

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

{

if (s\_temp.is\_element(s2.pdata[i])==1)

s\_temp.delete\_element(s2.pdata[i]);

}

return s\_temp;

}

MySet operator\*(MySet& s1, MySet& s2)

{

MySet s\_temp(s1);

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

{

if (s\_temp.is\_element(s2.pdata[i])==0)

s\_temp.delete\_element(s2.pdata[i]);

}

return s\_temp;

}

bool MySet::operator==(MySet& s)

{

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

{

if (pdata[i]>s.pdata[i])

return false;

}

return true;

}

MySet& MySet::operator+=(MySet& s)

{

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

{

if (is\_element(s.pdata[i])==0)

add\_element(s.pdata[i]);

}

return \*this;

}

MySet& MySet::operator-=(MySet& s)

{

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

{

if (is\_element(s.pdata[i])==1)

delete\_element(s.pdata[i]);

}

return \*this;

}

MySet& MySet::operator\*=(MySet& s)

{

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

{

if (is\_element(s.pdata[i])==0)

delete\_element(pdata[i]);

}

return \*this;

}

void MySet::add\_element(char\* el)

{

MyVector::add\_element(el);

MyVector::sort();

}

void MySet::delete\_element(char\* el)

{

int k=-1;

int a=0;

int b=size;

int c;

while (a-b>1)

{

c=(a+b)/2;

if (el>pdata[c])

a=c;

else

b=c;

}

if (pdata[a]==el)

k=a;

if (k!=-1)

MyVector::delete\_element(k);

}

bool MySet::is\_element(char\* el)

{

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

{

if (pdata[i]==el)

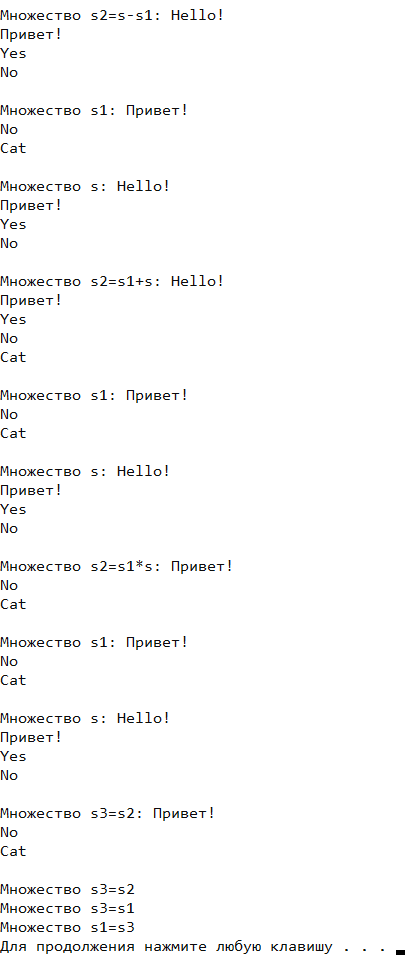
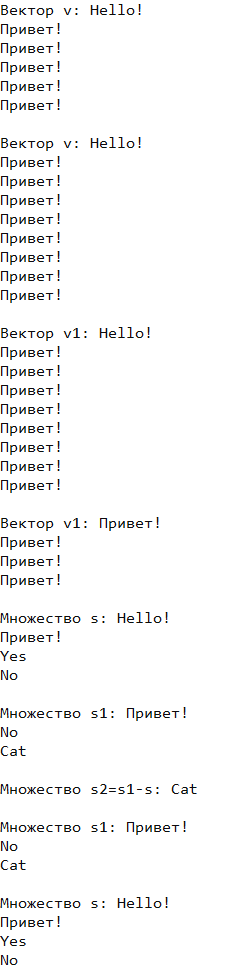
return true;

}

return false;

}

**Анализ результатов**

****