**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №5

«Наследование. Множества»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-23Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Правдина А.Д. |
| Подпись и дата: 02.03.2022 |  | Подпись и дата: 02.03.2022 |

Москва, 2022 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.

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

**Текст программы**

**Листинг кода программы:**

**MyVector.hpp**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_MYVECTOR\_HPP  
*#define* CODE\_MYVECTOR\_HPP  
*//файл MyVector.h - описание класса MyVector  
  
#include* <iostream>  
  
*using namespace* std;  
  
*const int* MAX\_SIZE = 5;  
  
*template*<*typename* T>  
*class* MyVector {  
*public*:  
 MyVector(); *// Done* MyVector(T, *int* = MAX\_SIZE); *// Done* MyVector(*const* MyVector &v); *// Done* ~MyVector(); *// Done  
  
 void* add\_element(T); *// done  
  
 bool* delete\_element(*int* i); *// done* T *operator*[](*int* i); *// done  
  
 void* sort(); *//done  
  
 int* Size() { *return* size; } *//Done  
  
 int* Maxsize() { *return* maxsize; } *// Done  
  
 int* find(T); *// Done* MyVector<T>& *operator*=(*const* MyVector<T> &v); *// Done  
  
 template*<*typename* T1>  
 *friend* ostream &*operator*<<(ostream &out, MyVector<T1> &v);*// Done  
  
protected*:  
 *int* maxsize;  
 *int* size;  
 T \*pdata;  
*private*:  
 *void* resize(); *// done*};  
  
*#include* "MyVector.inl"  
  
*#endif*

**MySet.hpp**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_MYSET\_HPP  
*#define* CODE\_MYSET\_HPP  
  
*#include* <iostream>  
*#include* "MyVector.hpp"  
  
*using namespace* std;  
  
*template* <*typename* T>  
*class* MySet : *public* MyVector<T> {  
*public*:  
 MySet() : MyVector<T>() {};  
  
 MySet(T el) : MyVector<T>(el, MAX\_SIZE) {};  
  
 *template* <*typename* T1>  
 *friend* ostream &*operator*<<(ostream &out, MySet<T1> &s); *//DONE  
  
 template* <*typename* T1>  
 *friend* MySet<T1> *operator*+(MySet<T1> &s1, MySet<T1> &s2); *// DONE  
  
 template* <*typename* T1>  
 *friend* MySet<T1> *operator*-(MySet<T1> &s1, MySet<T1> &s2);  
  
 *template* <*typename* T1>  
 *friend* MySet<T1> *operator*\*(MySet<T1> &s1, MySet<T1> &s2);  
  
 *bool operator*==(MySet &s);  
  
 MySet &*operator*+=(MySet &s);  
  
 MySet &*operator*-=(MySet &s);  
  
 MySet &*operator*\*=(MySet &s);  
  
 *void* add\_element(T el); *//DONE  
  
 void* delete\_element(T el); *// DONE  
  
 bool* is\_element(T el); *//DONE  
  
private*:  
 *int* q\_find(T el);  
};  
*#include* "MySet.inl"  
  
*#endif //CODE\_MYVECTOR\_HPP*

**MyVector.cpp**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>  
  
  
template*<*typename* T>  
MyVector<T>::MyVector() {  
 size = 0;  
 maxsize = MAX\_SIZE;  
 pdata = *new* T[maxsize];  
}  
  
*template*<>  
MyVector<*char* \*>::MyVector() {  
 size = 0;  
 maxsize = MAX\_SIZE;  
 pdata = *new char* \*[maxsize];  
}  
  
*template*<>  
MyVector<*char* \*>::MyVector(*char* \*el, *int* maxsz) : maxsize(maxsz) {  
 size = 1;  
 pdata = *new char* \*[maxsz];  
 pdata[0] = *new char*[strlen(el) + 1];  
 strcpy(\*pdata, el);  
}  
  
*template*<*typename* T>  
MyVector<T>::MyVector(T el, *int* maxsz) : maxsize(maxsz) {  
 size = 1;  
 pdata = *new* T[maxsz];  
 pdata[0] = el;  
}  
  
*template*<>  
*void* MyVector<*char* \*>::sort() {  
 std::sort(*this*->pdata, *this*->pdata + size, [](*const char* \*first, *const char* \*second) {  
 *return* strcmp(first, second) < 0;  
 });  
}  
  
*template*<*typename* T>  
*void* MyVector<T>::sort() {  
 std::sort(*this*->pdata, *this*->pdata + size);  
}  
  
*template*<>  
*void* MyVector<*char* \*>::resize() {  
 *if* (size >= maxsize)  
 maxsize = (maxsize / 2 + 1) \* 3;  
 *else if* (maxsize != MAX\_SIZE && size <= maxsize / 2)  
 maxsize = (maxsize / 3 + 1) \* 2;  
 *else return*;  
 *char* \*\*newMemory = *new char* \*[maxsize];  
  
 *for* (size\_t i = 0; i < size; ++i) {  
 *if* (pdata[i] == NULL) *break*;  
 newMemory[i] = *new char*[strlen(pdata[i]) + 1];  
 strcpy(newMemory[i], pdata[i]);  
 }  
 *for* (size\_t i = 0; i < size; ++i) {  
 *delete*[] pdata[i];  
  
 }  
 *delete*[] pdata;  
 pdata = newMemory;  
}  
  
*template*<*typename* T>  
*void* MyVector<T>::resize() {  
 *if* (size >= maxsize)  
 maxsize = (maxsize / 2 + 1) \* 3;  
 *else if* (maxsize != MAX\_SIZE && size <= maxsize / 2)  
 maxsize = (maxsize / 3 + 1) \* 2;  
 *else return*;  
 T \*newMemory = *new* T[maxsize];  
  
 *for* (size\_t i = 0; i < size; ++i) {  
 *if* (pdata[i] == NULL) *break*;  
 newMemory[i] = pdata[i];  
 }  
 *delete*[] pdata;  
 pdata = newMemory;  
}  
  
*template*<>  
MyVector<*char* \*>::MyVector(*const* MyVector &v) : maxsize(v.maxsize), size(v.size) {  
 pdata = *new char* \*[v.maxsize];  
 *for* (size\_t i = 0; i < size; ++i) {  
 pdata[i] = *new char*[strlen(v.pdata[i]) + 1];  
 strcpy(pdata[i], v.pdata[i]);  
 }  
}  
  
*template*<*typename* T>  
MyVector<T>::MyVector(*const* MyVector &v) : maxsize(v.maxsize), size(v.size) {  
 pdata = *new* T[v.maxsize];  
 *for* (size\_t i = 0; i < size; ++i) {  
 pdata[i] = v.pdata[i];  
 }  
}  
  
*template*<>  
*void* MyVector<*char* \*>::add\_element(*char* \*el) {  
 size++;  
 resize();  
 pdata[size - 1] = *new char*[strlen(el) + 1];  
 strcpy(pdata[size - 1], el);  
}  
  
*template*<*typename* T>  
*void* MyVector<T>::add\_element(T el) {  
 size++;  
 resize();  
 pdata[size - 1] = el;  
}  
  
*template*<>  
MyVector<*char* \*>::~MyVector() {  
 *for* (size\_t i = 0; i < size; ++i) {  
 *delete*[] pdata[i];  
 }  
 *delete*[] pdata;  
}  
  
*template*<*typename* T>  
MyVector<T>::~MyVector() {  
 *delete*[] pdata;  
}  
  
  
*template*<>  
*bool* MyVector<*char* \*>::delete\_element(*int* i) {  
 *if* (i > -1 && i < size) {  
 *for* (size\_t j = i; j < size - 1; ++j) {  
 pdata[j] = *new char*[strlen(pdata[j + 1]) + 1];  
 strcpy(pdata[j], pdata[j + 1]);  
 }  
 size--;  
 resize();  
 *return true*;  
 } *else  
 return false*;  
}  
  
*template*<*typename* T>  
*bool* MyVector<T>::delete\_element(*int* i) {  
 *if* (i > -1 && i < size) {  
 *for* (size\_t j = i; j < size - 1; ++j) {  
 pdata[j] = pdata[j + 1];  
 }  
 size--;  
 resize();  
 *return true*;  
 } *else  
 return false*;  
}  
  
*template*<>  
*char* \*MyVector<*char* \*>::*operator*[](*int* i) {  
 *return* pdata[i];  
}  
  
*template*<*typename* T>  
T MyVector<T>::*operator*[](*int* i) {  
 *return* pdata[i];  
}  
  
*template*<*typename* T>  
*int* MyVector<T>::find(T el) {  
 *int* q = std::find\_if(pdata, pdata + size, [el](T q) { *return* q == el; }) - pdata;  
 *if* (q == size) q = -1;  
 *return* q;  
}  
  
*template*<>  
*int* MyVector<*char* \*>::find(*char* \*el) {  
 *int* q = std::find\_if(pdata, pdata + size, [el](*char* \*q) { *return* strcmp(q, el); }) - pdata;  
 *if* (q == size) q = -1;  
 *return* q;  
}  
  
*template*<*typename* T>  
MyVector<T> &MyVector<T>::*operator*=(*const* MyVector<T> &v) {  
 *if* (*this* != &v) {  
 maxsize = v.maxsize;  
 size = v.size;  
 pdata = *new* T \*[v.maxsize];  
 *for* (size\_t i = 0; i < size; ++i) {  
 pdata[i] = *new* T;  
 pdata[i] = v.pdata[i];  
 }  
 }  
 *return* \**this*;  
}  
  
*template*<>  
MyVector<*char* \*> &MyVector<*char* \*>::*operator*=(*const* MyVector<*char* \*> &v) {  
 *if* (*this* != &v) {  
 maxsize = v.maxsize;  
 size = v.size;  
 pdata = *new char* \*[v.maxsize];  
 *for* (size\_t i = 0; i < size; ++i) {  
 pdata[i] = *new char*[strlen(v.pdata[i]) + 1];  
 strcpy(pdata[i], v.pdata[i]);  
 }  
 }  
 *return* \**this*;  
}  
  
*template*<*typename* T>  
ostream &*operator*<<(ostream &out, MyVector<T> &v) {  
 *for* (size\_t i = 0; i < v.size; ++i) {  
 out << '[' << i << ']' << " : " << "{\"" << v.pdata[i] << "\"}" << endl;  
 }  
 *return* out;  
}  
  
*template*<>  
ostream &*operator*<<(ostream &out, MyVector<*char* \*> &v) {  
 *for* (size\_t i = 0; i < v.size; ++i) {  
 out << '[' << i << ']' << " : " << "{\"" << v.pdata[i] << "\"}" << endl;  
 }  
 *return* out;  
}

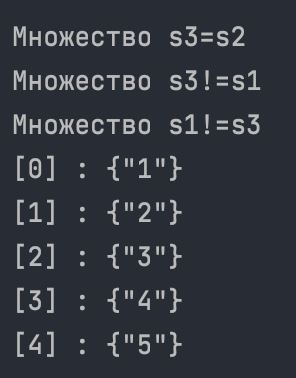
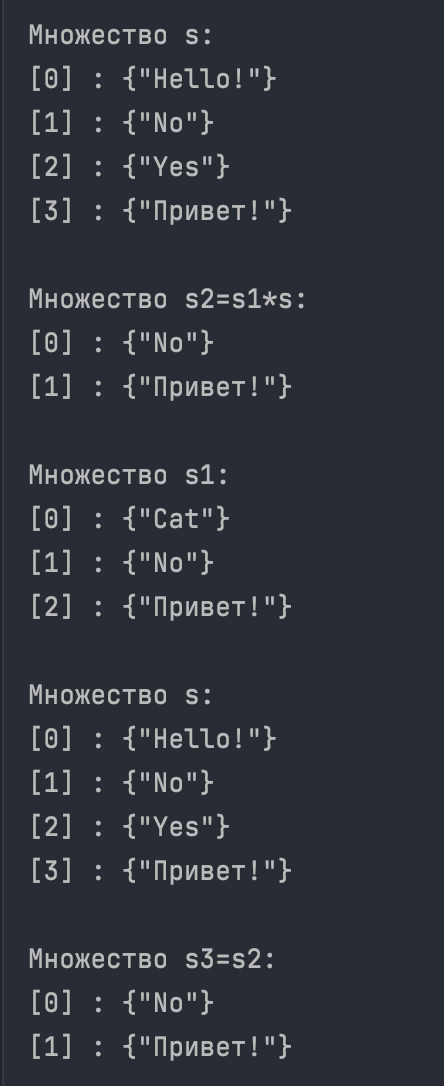
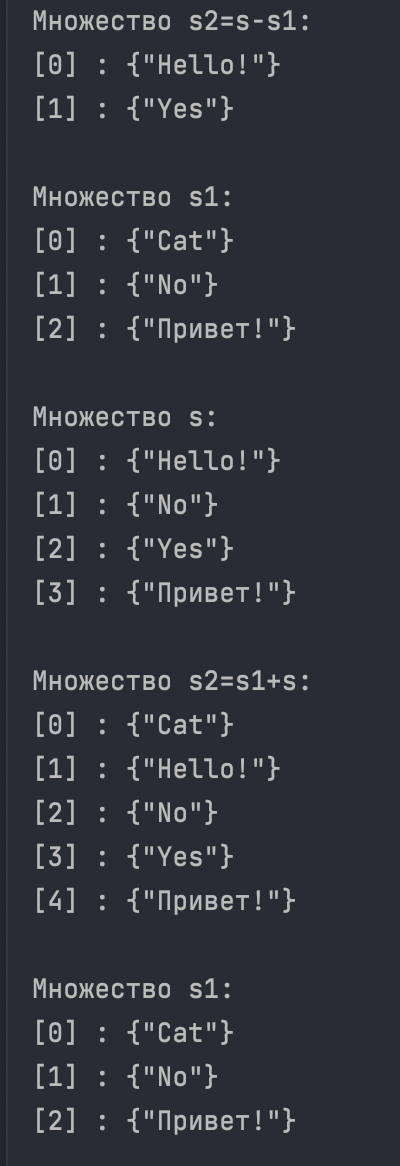
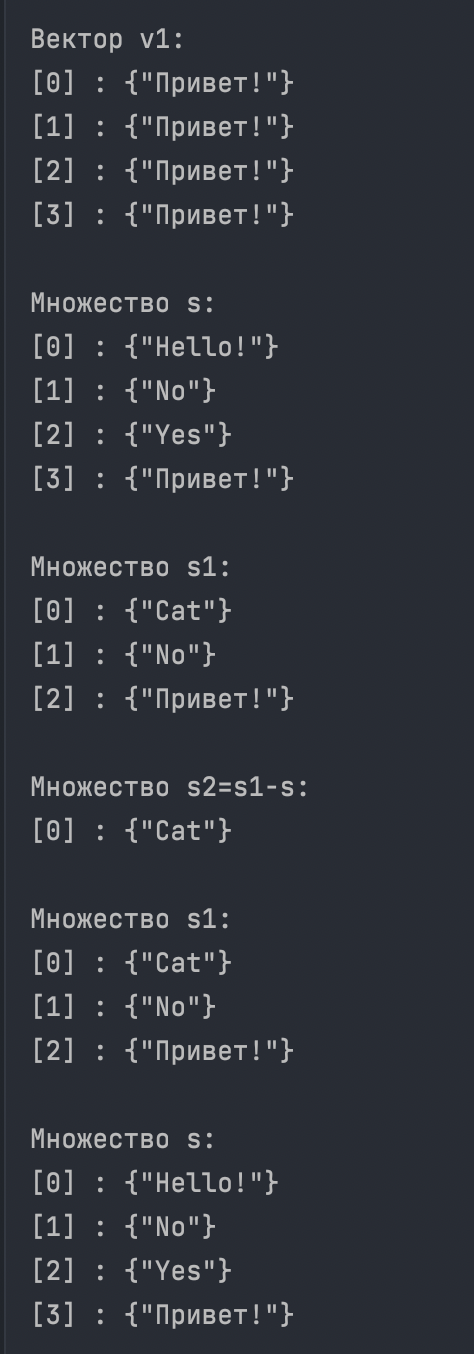
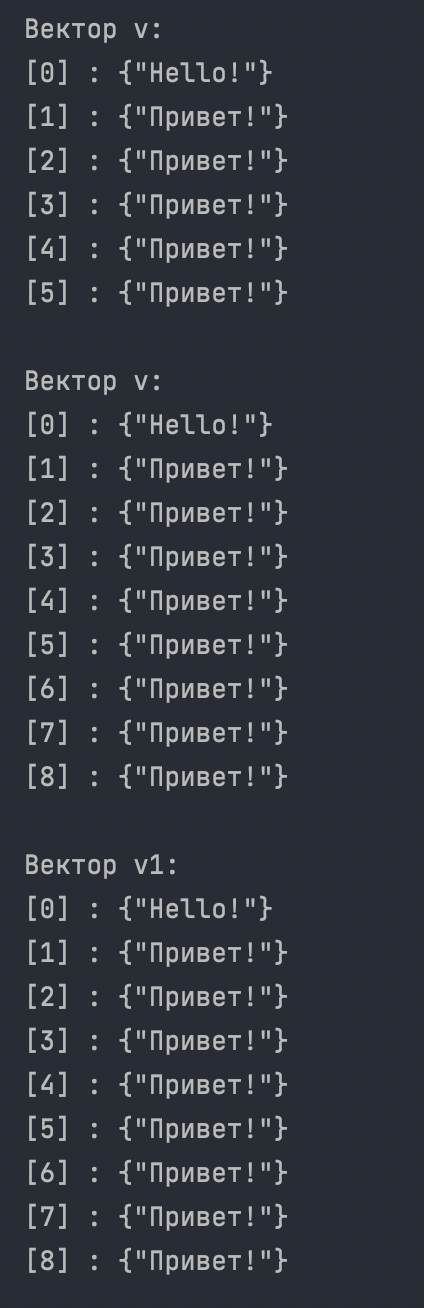
**MySet.inl**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>  
  
template*<>  
*void* MySet<*char* \*>::add\_element(*char* \*el) {  
 MyVector<*char* \*>::add\_element(el);  
 MyVector<*char* \*>::sort();  
}  
  
*template*<*typename* T>  
*void* MySet<T>::add\_element(T el) {  
 MyVector<T>::add\_element(el);  
 MyVector<T>::sort();  
}  
  
*template*<>  
*int* MySet<*char* \*>::q\_find(*char* \*el) {  
 *int* left = 0;  
 *int* right = size - 1;  
 *int* mid = 0;  
 *while* (*true*) {  
 mid = (left + right) / 2;  
 *if* (strcmp(el, pdata[mid]) < 0)  
 right = mid - 1;  
 *else if* (strcmp(el, pdata[mid]) > 0)  
 left = mid + 1;  
 *else  
 return* mid;  
 *if* (left > right)  
 *return* -1;  
 }  
}  
  
*template*<*typename* T>  
*int* MySet<T>::q\_find(T el) {  
 *int* left = 0;  
 *int* right = *this*->size - 1;  
 *int* mid = 0;  
 *while* (*true*) {  
 mid = (left + right) / 2;  
 *if* (el < *this*->pdata[mid])  
 right = mid - 1;  
 *else if* (el > *this*->pdata[mid])  
 left = mid + 1;  
 *else  
 return* mid;  
 *if* (left > right)  
 *return* -1;  
 }  
}  
  
*template*<>  
*void* MySet<*char* \*>::delete\_element(*char* \*el) {  
 *int* q = q\_find(el);  
 *if* (q != -1) {  
 MyVector<*char* \*>::delete\_element(q);  
 MyVector<*char* \*>::sort();  
 }  
}  
  
*template*<*typename* T>  
*void* MySet<T>::delete\_element(T el) {  
 *int* q = q\_find(el);  
 *if* (q != -1) {  
 MyVector<T>::delete\_element(q);  
 MyVector<T>::sort();  
 }  
}  
  
  
*template*<*typename* T>  
*bool* MySet<T>::is\_element(T el) {  
 *if* (q\_find(el) != -1) *return true*;  
 *return false*;  
}  
  
*template*<*typename* T>  
ostream &*operator*<<(ostream &out, MySet<T> &s) {  
 *for* (size\_t i = 0; i < s.size; ++i) {  
 out << '[' << i << ']' << " : " << "{\"" << s.pdata[i] << "\"}" << '\n';  
 }  
 *return* out;  
}  
  
*template*<>  
ostream &*operator*<<(ostream &out, MySet<*char* \*> &s) {  
 *for* (size\_t i = 0; i < s.size; ++i) {  
 out << '[' << i << ']' << " : " << "{\"" << s.pdata[i] << "\"}" << '\n';  
 }  
 *return* out;  
}  
  
  
*template*<*typename* T>  
MySet<T> *operator*+(MySet<T> &s1, MySet<T> &s2) {  
 MySet<T> newSet = s1;  
 *for* (size\_t i = 0; i < s2.size; ++i) {  
 *if* (newSet.q\_find(s2[i]) == -1) {  
 newSet.add\_element(s2[i]);  
 }  
 }  
 *return* newSet;  
}  
  
*template*<>  
MySet<*char*\*> *operator*+(MySet<*char*\*> &s1, MySet<*char*\*> &s2) {  
 MySet<*char*\*> newSet = s1;  
 *for* (size\_t i = 0; i < s2.size; ++i) {  
 *if* (newSet.q\_find(s2[i]) == -1) {  
 newSet.add\_element(s2[i]);  
 }  
 }  
 *return* newSet;  
}  
  
*template*<*typename* T>  
MySet<T> *operator*-(MySet<T> &s1, MySet<T> &s2) {  
 MySet<T> newSet;  
 *for* (size\_t i = 0; i < s1.size; ++i) {  
 *if* (s2.q\_find(s1[i]) == -1) {  
 newSet.add\_element(s1[i]);  
 }  
 }  
 *return* newSet;  
}  
  
*template*<>  
MySet<*char*\*> *operator*-(MySet<*char*\*> &s1, MySet<*char*\*> &s2) {  
 MySet<*char*\*> newSet;  
 *for* (size\_t i = 0; i < s1.size; ++i) {  
 *if* (s2.q\_find(s1[i]) == -1) {  
 newSet.add\_element(s1[i]);  
 }  
 }  
 *return* newSet;  
}  
  
*template*<*typename* T>  
MySet<T> *operator*\*(MySet<T> &s1, MySet<T> &s2) {  
 MySet<T> newSet;  
 *for* (size\_t i = 0; i < s1.size; ++i) {  
 *if* (s2.q\_find(s1[i]) != -1) {  
 newSet.add\_element(s1[i]);  
 }  
 }  
 *return* newSet;  
}  
  
*template*<>  
MySet<*char*\*> *operator*\*(MySet<*char*\*> &s1, MySet<*char*\*> &s2) {  
 MySet<*char*\*> newSet;  
 *for* (size\_t i = 0; i < s1.size; ++i) {  
 *if* (s2.q\_find(s1[i]) != -1) {  
 newSet.add\_element(s1[i]);  
 }  
 }  
 *return* newSet;  
}  
  
*template*<>  
*bool* MySet<*char* \*>::*operator*==(MySet<*char* \*> &s) {  
 *if* (size == s.size) {  
 *for* (size\_t i = 0; i < size; ++i) {  
 *if* (strcmp(pdata[i], s[i]) != 0)  
 *return false*;  
 }  
 *return true*;  
 }  
 *return false*;  
}  
  
*template*<*typename* T>  
*bool* MySet<T>::*operator*==(MySet<T> &s) {  
 *if* (*this*->size == s.size) {  
 *for* (size\_t i = 0; i < *this*->size; ++i) {  
 *if* (*this*->pdata[i] != s[i])  
 *return false*;  
 }  
 *return true*;  
 }  
 *return false*;  
}  
  
*template*<*typename* T>  
MySet<T> &MySet<T>::*operator*+=(MySet<T> &s) {  
 \**this* = \**this* + s;  
 *return* \**this*;  
}  
  
*template*<*typename* T>  
MySet<T> &MySet<T>::*operator*-=(MySet<T> &s) {  
 \**this* = \**this* - s;  
 *return* \**this*;  
}  
  
*template*<*typename* T>  
MySet<T> &MySet<T>::*operator*\*=(MySet<T> &s) {  
 \**this* = \**this* \* s;  
 *return* \**this*;  
}  
  
*//{1, 4, 5, 6} \* {1, 2, 3, 4} => {1, 4}*

**main.cpp**

*#include* <iostream>  
  
*#include* "MyVector.hpp"  
*#include* "MySet.hpp"  
  
*using namespace* std;  
  
*int* main ()  
{  
 MyVector<*char*\*> 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:\n"<<v<<endl;  
 v.add\_element("Привет!");  
 v.add\_element("Привет!");  
 v.add\_element("Привет!");  
 cout<<"Вектор v:\n"<<v<<endl;  
 MyVector<*char*\*> v1=v;  
 cout<<"Вектор v1:\n"<<v1<<endl;  
 *for*(*int* i=0;i<MAX\_SIZE;i++)  
 v1.delete\_element(0);  
 cout<<"Вектор v1:\n"<<v1<<endl;  
 MySet<*char*\*> s1,s2;  
 MySet<*char*\*> s("Yes");  
 s.add\_element("Привет!");  
 s.add\_element("No");  
 *char*\* str="Hello!";  
 s.add\_element(str);  
 cout<<"Множество s: \n"<<s<<endl;  
 s1.add\_element("Cat");  
 s1.add\_element("No");  
 s1.add\_element("Привет!");  
 cout<<"Множество s1: \n"<<s1<<endl;  
 s2=s1-s;  
 cout<<"Множество s2=s1-s:\n"<<s2<<endl;  
 cout<<"Множество s1: \n"<<s1<<endl;  
 cout<<"Множество s: \n"<<s<<endl;  
 s2=s-s1;  
 cout<<"Множество s2=s-s1: \n"<<s2<<endl;  
 cout<<"Множество s1: \n"<<s1<<endl;  
 cout<<"Множество s: \n"<<s<<endl;  
 s2=s1+s;  
 cout<<"Множество s2=s1+s: \n"<<s2<<endl;  
 cout<<"Множество s1: \n"<<s1<<endl;  
 cout<<"Множество s: \n"<<s<<endl;  
 s2=s1\*s;  
 cout<<"Множество s2=s1\*s: \n"<<s2<<endl;  
 cout<<"Множество s1: \n"<<s1<<endl;  
 cout<<"Множество s: \n"<<s<<endl;  
 MySet<*char*\*> s3=s2;  
 cout<<"Множество s3=s2: \n"<<s3<<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\n  
  
 MyVector<*int*> mv;  
 mv.add\_element(1);  
 mv.add\_element(2);  
 mv.add\_element(3);  
 mv.add\_element(4);  
 mv.add\_element(5);  
 std::cout << mv << std::endl;  
  
 *return* 0;  
}

**Анализ результатов**

****