**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 ИУ5-24б |  | преподаватель каф. ИУ5 |
| Чернев Н.А. |  |  |
| Подпись и дата: 29.03.2024 |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

## Задание

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

## ПРЕДУПРЕЖДЕНИЕ

## Класс MySet и MyVector должны быть упакованы в отдельную статическую библиотеку. Пример того, как это может быть сделано, представлен в [Приложении 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab5/Instructions/SupplementMaterial1/).

## Необходимо реализовать специализацию шаблонов для работы со скалярными типами (char\*, double, int и др.). В специализациях выполнить корректное определение специальных членов классов (конструкторы и деструкторы).

## 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 переопределяют одноименные методы базового класса MyVector, а остальные элементы класса MyVector наследуются классом MySet.

## При выполнении лабораторной работы использовать файлы с описанием классов MyVector, MySet и функцию main() главного проекта, приведенные в [Приложении 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab5/Instructions/SupplementMaterial1/).

**Разработка алгоритма**

Библиотека VecnSet:

* Класс  MyVector<T>:
  + поля:
    - int max\_size; - вместимость массива
    - int size; - кол-во элементов в массиве
    - T \*pdata; - массив значений
  + методы:
    - void resize() - изменение вместимости массива
    - MyVector() - конструктор без параметров, инициализирует поля нулевыми значениями, выделяет память под 1 элемент массива
    - MyVector(const T &val) - конструктор с параметром типа Т, инициализирует поля и записывает переданное значение в массив по индексу 0
    - MyVector(const MyVector &vec) - конструктор копирования, принимает копию вектора и инициализирует объект данными из переданного
    - ~MyVector() - деструктор, удаляет динамически выделенную память
    - void sort() - сортировка массива
    - void add\_element(const T &val) - принимает элемент типа Т и добавляет его в конец массива
    - bool delete\_element(int ind) - принимает индекс и удаляет элемент по этому индексу, возвращает true если удалось удалить элмент, false в противном случае
    - int find(const T &val) - принимает значение типа Т и возвращает индекс первого вхождения этого элемента в случае, если он есть в массиве, -1 в противном случае
    - int len() - возвращает size(кол-во элементов в массиве)
    - MyVector& operator = (const MyVector &vec) - перегрузка оператора = для MyVector = MyVector
    - T& operator[] (int i) - перегрузка оператора [], возвращает ссылку на объект по индексу i
    - friend std::ostream& operator << (std::ostream &out, MyVector &vec) - перегрузка вывода MyVector
* Класс MySet<T>, наследник MyVector<T> со спец. доступа public
  + Методы:
    - int q\_find(const T &val) - бинарный поиск элемента в множестве, возвращает индекс искомого элемента в случае, если он есть в множестве, -1 в противном случае
    - MySet() - конструктор без параметров, инициализирует поля нулевыми значениями, выделяет память под 1 элемент массива
    - MySet(const T &val) - конструктор с параметром типа Т, инициализирует поля и записывает переданное значение в массив по индексу 0
    - MySet(const MySet &s) - конструктор копирования, принимает копию множества и инициализирует объект данными из переданного
    - bool is\_element(const T &val) - принимает значение типа Т, возвращает true, если такой элемент есть в множестве, false в противном случае
    - void add\_element(const T &val) - принимает элемент типа Т и добавляет его в массив таким образом, чтобы массив оставался отсортированным(если элемент уже есть в множестве, ничего не делает)
    - bool delete\_element(const T &val) - принимает значение типа Т, удаляет его из множества в случае, если он там есть и вовзращает true, в противном случае просто возвращает false
    - bool operator == (const MySet &s) - перегрузка оператора == для MySet == MySet
    - MySet& operator = (const MySet &s) - перегрузка оператора = для MySet = MySet
    - MySet& operator += (MySet &s) - перегрузка оператора += для MySet += MySet
    - MySet& operator \*= (MySet &s) - перегрузка оператора \*= для MySet \*= MySet
    - MySet& operator \*= (MySet &s) - перегрузка оператора \*= для MySet \*= MySet
    - friend MySet operator + (MySet &s1, MySet &s2) - перегрузка оператора + для MySet + MySet
    - friend MySet operator - (MySet &s1, MySet &s2) - перегрузка оператора - для MySet - MySet
    - friend MySet operator \* (MySet &s1, MySet &s2) - перегрузка оператора \* для MySet \* MySet

**Текст программы**

Библиотека VecnSet:

MyVector.h

#pragma once

#include <iostream>

template<class T>

int compare(T val1, T val2) {

return val1 - val2;

}

template<>

int compare<char \*>(char \*val1, char \*val2) {

if (!val1 || !val2) return -1;

return strcmp(val1, val2);

}

template<>

int compare<std::string>(std::string s1, std::string s2) {

return s1.compare(s2);

}

template <class T>

class MyVector {

protected:

int max\_size;

int size;

T \*pdata;

void resize() {

if (size >= max\_size - 1) {

max\_size \*= 2;

}

else if (size < max\_size / 4) {

max\_size /= 2;

}

else {

return;

}

T \*tmp = new T[max\_size];

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

tmp[i] = pdata[i];

}

*//delete[] pdata;*

pdata = tmp;

}

public:

MyVector() {

pdata = new T[1];

size = 0;

max\_size = 1;

}

MyVector(const T &val) {

pdata = new T[1];

pdata[0] = val;

size = 1;

max\_size = 1;

}

MyVector(const MyVector &vec) {

pdata = new T[vec.max\_size];

size = vec.size;

max\_size = vec.max\_size;

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

pdata[i] = vec.pdata[i];

}

}

~MyVector() {

delete[] pdata;

}

void sort() {

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

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

if (compare(pdata[i], pdata[j]) > 0) {

std::swap(pdata[i], pdata[j]);

}

}

}

}

void add\_element(const T &val) {

resize();

pdata[size++] = val;

}

bool delete\_element(int ind) {

if (ind < 0 || ind >= size) return false;

for (int i = ind + 1; i < size; i++) {

pdata[i - 1] = pdata[i];

}

size--;

resize();

return true;

}

int find(const T &val) {

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

if (compare(pdata[i], val) == 0) {

return i;

}

}

return -1;

}

int len() {

return size;

}

MyVector& operator = (const MyVector &vec) {

delete[] pdata;

pdata = new T[vec.max\_size];

size = 0;

max\_size = vec.max\_size;

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

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

}

return \*this;

}

T& operator[] (int i) {

if (i < 0 || i >= size) {

std::cout << "Index Error\n";

exit(1);

}

return pdata[i];

}

friend std::ostream& operator << (std::ostream &out, MyVector &vec) {

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

out << vec[i] << ' ';

}

out << '\n';

return out;

}

};

template<>

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

pdata = new char \*[1];

pdata[0] = new char[1];

pdata[0][0] = '\0';

size = 0;

max\_size = 1;

}

template<>

MyVector<char \*>::MyVector(char \*const &str) {

pdata = new char \*[1];

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

strcpy(pdata[0], str);

size = 1;

max\_size = 1;

}

template<>

MyVector<char \*>::MyVector(const MyVector &vec) {

pdata = new char \*[vec.max\_size];

size = vec.size;

max\_size = vec.max\_size;

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

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

strcpy(pdata[i], vec.pdata[i]);

}

}

template<>

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

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

delete[] pdata[i];

}

delete[] pdata;

}

template<>

void MyVector<char \*>::add\_element(char \*const &str) {

resize();

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

strcpy(pdata[size], str);

size++;

}

template<>

bool MyVector<char \*>::delete\_element(int ind) {

if (ind < 0 || ind >= size) return false;

if (pdata[ind]) delete[] pdata[ind];

for (int i = ind + 1; i < size; i++) {

pdata[i - 1] = pdata[i];

}

size--;

resize();

return true;

}

template<>

MyVector<char \*>& MyVector<char \*>::operator = (const MyVector& vec) {

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

delete[] pdata[i];

}

delete[] pdata;

pdata = new char \*[vec.max\_size];

size = 0;

max\_size = vec.max\_size;

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

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

}

return \*this;

}

MySet.h

#include "MyVector.h"

template<class T>

class MySet : public MyVector<T> {

int q\_find(const T &val) {

int l = 0, r = this->size - 1;

int mid = (l + r) / 2;

while ((compare(this->pdata[mid], val) != 0) && (l < r)){

if (compare(this->pdata[mid], val) > 0){

r = mid;

}

else{

l = mid + 1;

}

mid = (l + r) / 2;

}

if (compare(this->pdata[mid], val) == 0) {

return mid;

}

return -1;

}

public:

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

MySet(const T &val) : MyVector<T>(val) {}

MySet(const MySet &s) : MyVector<T>(s) {}

bool is\_element(const T &val) { return q\_find(val) != -1; }

void add\_element(const T &val) {

if (is\_element(val)) return;

this->size++;

this->resize();

int ind = 0;

while ((this->pdata[ind] < val) && (ind < this->size - 1)) {

ind++;

}

for (int i = this->size - 1; i > ind; i--) {

this->pdata[i] = this->pdata[i - 1];

}

this->pdata[ind] = val;

}

bool delete\_element(const T &val) {

int ind = q\_find(val);

if (ind == -1) return false;

return MyVector<T>::delete\_element(ind);

}

bool operator == (const MySet &s) {

if (this->size != s.size) return false;

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

if (compare(this->pdata[i], s.pdata[i]) != 0) return false;

}

return true;

}

MySet& operator += (MySet &s) {

\*this = \*this + s;

return \*this;

}

MySet& operator -= (MySet &s) {

\*this = \*this - s;

return \*this;

}

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

\*this = \*this \* s;

return \*this;

}

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

MySet tmp = s1;

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

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

}

return tmp;

}

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

MySet tmp = s1;

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

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

}

return tmp;

}

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

MySet tmp;

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

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

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

}

}

return tmp;

}

MySet& operator = (const MySet &vec) {

MyVector<T>::operator=(vec);

return \*this;

}

};

template<>

void MySet<char \*>::add\_element(char \*const &el) {

if (is\_element(el)) return;

resize();

size++;

int ind = 0;

while ((ind < size - 1) && (strcmp(pdata[ind], el) < 0)) {

ind++;

}

for (int i = size - 1; i > ind; i--) {

pdata[i] = pdata[i - 1];

}

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

strcpy(pdata[ind], el);

}

MyVector.cpp

#include "MyVector.h"

MySet.cpp

#include "MySet.h"

CMakeLists.txt библиотеки VecnSet

cmake\_minimum\_required(VERSION 3.23)

set(project "VecnSet")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

MyVector.cpp

MySet.cpp)

set(${project}\_HEADERS

MyVector.h

MySet.h)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES}

${${project}\_HEADERS})

add\_library(${project}

STATIC

${${project}\_SOURCE\_LIST})

main.cpp

#include "VecnSet/MySet.h"

#define MAX\_SIZE 3

int main() {

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

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

std::cout << "Вектор v: " << v << std::endl;

v.add\_element("Привет!");

v.add\_element("Привет!");

v.add\_element("Привет!");

std::cout << "Вектор v: " << v << std::endl;

MyVector v1 = v;

std::cout << "Вектор v1: " << v1 << std::endl;

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

v1.delete\_element(0);

}

std::cout << "Вектор v1: " << v1 << std::endl;

MySet<std::string> s("Yes"), s1, s2;

s.add\_element("Привет!");

s.add\_element("No");

char \*str = "Hello!";

s.add\_element(str);

std::cout << "Множество s: " << s << std::endl;

s1.add\_element("Cat");

s1.add\_element("No");

s1.add\_element("Привет!");

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

s2 = s1 - s;

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

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

std::cout << "Множество s: " << s << std::endl;

s2 = s - s1;

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

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

std::cout << "Множество s: " << s << std::endl;

s2 = s1 + s;

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

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

std::cout << "Множество s: " << s << std::endl;

s2 = s1 \* s;

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

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

std::cout << "Множество s: " << s << std::endl;

MySet s3 = s2;

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

if (s3 == s2)

std::cout << "Множество s3=s2\n";

else

std::cout << "Множество s3!=s2\n";

if (s3 == s1)

std::cout << "Множество s3=s1\n";

else

std::cout << "Множество s3!=s1\n";

if (s1 == s3)

std::cout << "Множество s1=s3\n";

else

std::cout << "Множество s1!=s3\n";

return 0;

}

CMakeLists.txt проекта

cmake\_minimum\_required(VERSION 3.23)

set(project "lab05\_t")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

main.cpp)

add\_subdirectory(VecnSet)

set(${project}\_SOURCE\_LIST

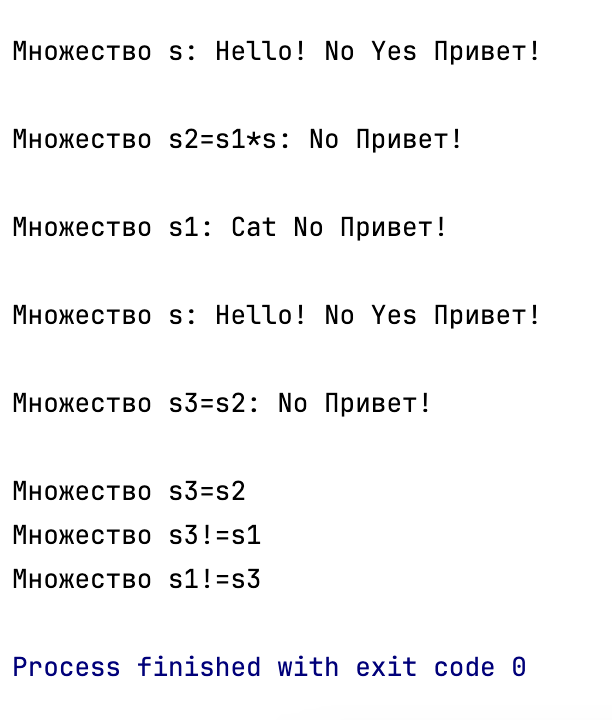
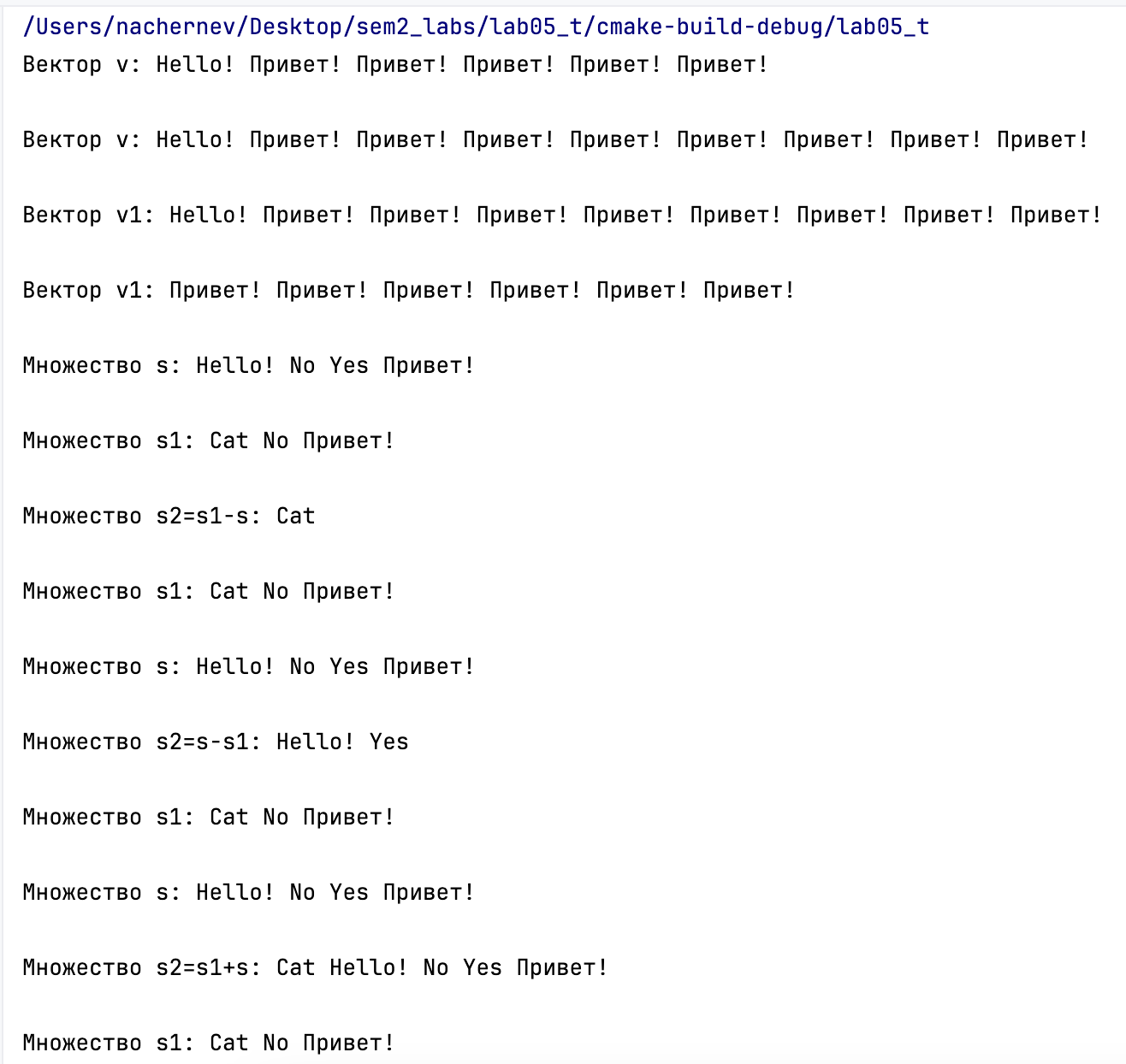
${${project}\_SOURCES})

add\_executable(${project}

${${project}\_SOURCE\_LIST})

target\_link\_libraries(${project} VecnSet)

**Анализ результатов**

****