**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 ИУ5-21 |  | преподаватель каф. ИУ5 |
| Агличеев Михаил |  | Селиверстова А.В. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

1. 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 .

1. Разработка Интерфейса класса

Что здесь писать???

|  |  |
| --- | --- |
|  |  |

1. Разработка алгоритма

Что писать здесь???

1. Текст программы

Файл vector.h:

#ifndef VECTOR\_H

#define VECTOR\_H

class MyVector {

public:

MyVector(char\* element = (char\*) "a");

MyVector(const MyVector&);

~MyVector();

void add\_element(char\* element);

void delete\_element(char\* element);

int find(char\* element);

void sort();

char\* operator[] (size\_t);

void operator= (MyVector);

size\_t Size();

size\_t Maxsize();

protected:

size\_t maxsize;

size\_t size;

char\*\* pdata;

private:

void resize();

};

void qsort(char\*\*, size\_t, size\_t);

void swap(char\*&, char\*&);

char\*\* copy\_data(char\*\*, size\_t, size\_t);

#endif // VECTOR\_H

Файл vector.cpp:

#include <cstring>

#include <algorithm>

#include <iostream>

#include "vector.h"

using namespace std;

const size\_t WORD\_LEN = 200;

const size\_t DEFAULT\_SIZE = 8;

MyVector::MyVector(char\* element) {

maxsize = DEFAULT\_SIZE;

size = 1;

pdata = new char\*[maxsize];

for (size\_t i = 0; i < maxsize; i++) {

pdata[i] = nullptr;

}

pdata[0] = new char[WORD\_LEN];

strcpy\_s(pdata[0], WORD\_LEN, element);

}

MyVector::MyVector(const MyVector& vector) {

maxsize = vector.maxsize;

size = vector.size;

pdata = new char\*[maxsize];

for (size\_t i = 0; i < size; i++) {

pdata[i] = new char[WORD\_LEN];

strcpy\_s(pdata[i], WORD\_LEN, vector.pdata[i]);

}

}

MyVector::~MyVector() {

size\_t i = 0;

while (i < size && pdata[i] != nullptr) {

delete[] pdata[i];

pdata[i] = nullptr;

i++;

}

pdata = nullptr;

}

void MyVector::resize() {

if (size == maxsize) {

maxsize = maxsize \* 3 / 2;

pdata = copy\_data(pdata, maxsize, size);

}

else if (size < maxsize / 2) {

if (maxsize \* 2 / 3 > DEFAULT\_SIZE) {

maxsize = maxsize \* 2 / 3;

pdata = copy\_data(pdata, maxsize, size);

}

}

}

void MyVector::add\_element(char\* element) {

pdata[size] = new char[WORD\_LEN];

strcpy\_s(pdata[size], WORD\_LEN, element);

size++;

sort();

resize();

}

void MyVector::delete\_element(char\* element) {

size\_t i = find(element);

if (i != -1) {

delete[] pdata[i];

for (i; i < size - 1; i++) {

pdata[i] = pdata[i + 1];

}

pdata[size - 1] = nullptr;

size--;

resize();

cout << "Successfully deleted an element" << endl;

}

cout << endl;

}

int MyVector::find(char\* element) {

for(size\_t i = 0; i < size; i++) {

if (strcmp(pdata[i], element) == 0) {

return i;

}

}

cout << "That element was not found" << endl;

return -1;

}

void MyVector::sort() {

qsort(pdata, 0, size - 1);

}

char\* MyVector::operator[] (size\_t i) {

if (i < size) {

return pdata[i];

}

cout << "Index Error" << endl << endl;

return nullptr;

}

void MyVector::operator= (MyVector source) {

maxsize = source.maxsize;

size = source.size;

pdata = new char\*[maxsize];

for (size\_t i = 0; i < size; i++) {

pdata[i] = new char[WORD\_LEN];

strcpy\_s(pdata[i], size, source.pdata[i]);

}

}

size\_t MyVector::Size() {

return size;

}

size\_t MyVector::Maxsize() {

return maxsize;

}

void qsort(char\*\* arr, size\_t start, size\_t end) { // Quick sort from start to end including both.

char\*\* pivot = arr + end; // Element to be placed at the right position

size\_t i = start, j;

for (j = start; j < end; j++) {

if (strcmp(arr[j], \*pivot) < 0) {

swap(arr[i], arr[j]);

++i;

}

}

swap(arr[i], arr[end]);

if (i - 1 > start) {

qsort(arr, start, i - 1); // Sorting "less than pivot" and "more than pivot" arrays.

}

if (end > i + 1) {

qsort(arr, i + 1, end);

}

}

void swap(char\*& first, char\*& second) {

char\* temp = first;

first = second;

second = first;

}

char\*\* copy\_data(char\*\* source, size\_t length, size\_t source\_length) {

char\*\* result = new char\*[length];

for (size\_t i = 0; i < length; i++) {

result[i] = nullptr;

}

size\_t i = 0;

while (i < min(length, source\_length)) {

result[i] = new char[WORD\_LEN];

strcpy\_s(result[i], WORD\_LEN, source[i]);

i++;

}

return result;

}

Файл check.h:

#ifndef CHECK\_H

#define CHECK\_H

int main();

#endif // CHECK\_H

Файл check.cpp:

#include <iostream>

#include <Windows.h>

#include "..\Vector\vector.h"

#include "check.h"

using namespace std;

const size\_t WORD\_LEN = 200;

int main() {

MyVector vector = MyVector("first");

vector.add\_element("second");

cout << "Initial vector:" << endl;

for (size\_t i = 0; i < vector.Size(); ++i) {

cout << vector[i] << endl;

}

cout << "Enter the third string that you want to add:" << endl;

char word[WORD\_LEN];

cin.getline(word, WORD\_LEN);

vector.add\_element(word);

cout << "Resulting vector:" << endl;

for (size\_t i = 0; i < vector.Size(); ++i) {

cout << vector[i] << endl;

}

cout << "And this is what we get by deleting second string:" << endl;

vector.delete\_element("second");

for (size\_t i = 0; i < vector.Size(); ++i) {

cout << vector[i] << endl;

}

system("pause");

return 1;

}

5. Анализ результатов

