

**Московский государственный технический
университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

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

Отчет по лабораторной работе № 4

«Множества»

Выполнил:

студент группы ИУ5-23

Терентьев Владислав

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Козлов А. Д.

Подпись и дата:

Москва, 2019 г.

1. Постановка задачи

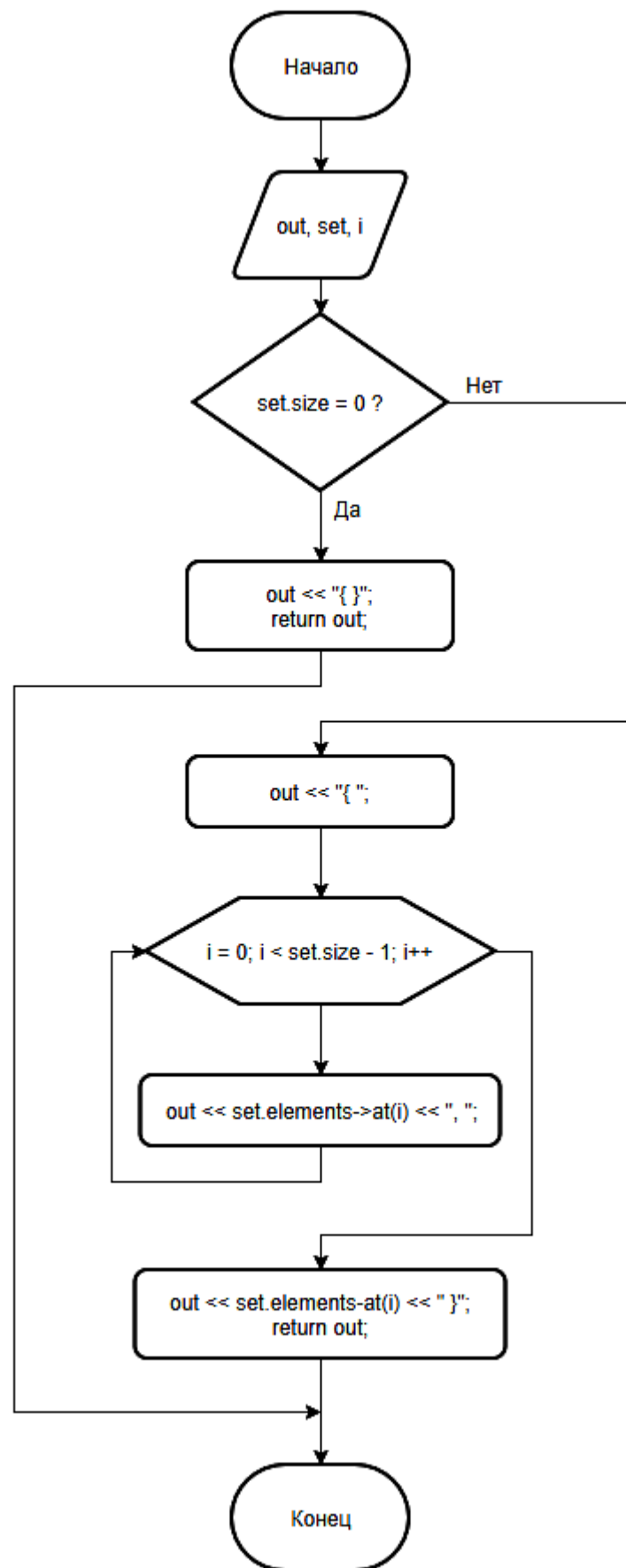
Разработать класс множество **"MySet"** на базе векторов для выполнения операций над множествами (+, -, *, =, +=, -=, *=, ==).

2. Разработка алгоритма

В программе описан шаблонный класс **MySet** (параметры шаблона: class INF) с псевдонимом **vest** для шаблонного класса **vector** <INF> и с переменными-членами (спецификатор доступа **private**): size типа **int** для хранения размера множества; elements указатель на объект класса **vest** для хранения и использования адреса, где расположены данные (элементы) множества; и метод Search типа **int** и параметром типа **INF** (элемент) для поиска нужного элемента в векторе. Так же в классе описаны (спецификатор доступа **public**): **конструкторы** (**default**; копирующий; с параметром типа **INF** для создания множества из одного элемента; с параметром типа **const vest &** для создания множества из вектора); деструктор для удаления данных множества и освобождения памяти. **Методы**: IsElement типа **bool** и параметром типа **INF** для проверки наличия во множестве данного элемента; AddElement типа **void** и параметром типа **INF** для добавления элемента в множество; DeleteElement типа **void** и параметром типа **INF** для удаления элемента из множества. И перегруженные **операторы** =, +, -, *, ==, +=, -=, *=, <<.

Описана глобальная шаблонная функция compare (параметры шаблона: class NFF) типа **int** и параметрами **const void***, **const void*** которая сравнивает два элемента для использования этой функции при выполнении функции **qsort** (быстрой сортировки).

Схема алгоритма оператора вывода (<<):



3. Текст программы

```
#include <iostream>
#include <vector>

using namespace std;
template <class NFF>
int compare(const void* x1, const void* x2) {
    if (*(NFF*)x1 < *(NFF*)x2) {
        return -1;
    }
    else {
        if (*(NFF*)x1 > *(NFF*)x2) {
            return 1;
        }
        else {
            return 0;
        }
    }
}

template <class INF>
class MySet {
    typedef class vector <INF> vect;
private:
    int size;

    vect* elements;

    int Search(INF a) {
        if (size == 0) {
            return -1;
        }
        int lf = 0, rg = size, temp2 = -1;
        int temp1 = (lf + rg) / 2;
        while (temp1 != temp2) {
            if (elements->at(temp1) < a) {
                lf = temp1;
            }
            else {
                if (elements->at(temp1) > a) {
                    rg = temp1;
                }
            }
            temp2 = temp1;
            temp1 = (lf + rg) / 2;
        }
        if (elements->at(temp1) == a) {
            return temp1 + 1;
        }
        else {
            return -1;
        }
    }
}

public:
    MySet() :size(0), elements(NULL) {}

    MySet(INF a) :size(1), elements(new vect(1)) {
        elements->at(0) = a;
    }

    MySet(const MySet & a) {
        if (a.size == 0) {
            size = 0;
            elements = NULL;
        }
    }
}
```

```

        else {
            size = a.size();
            elements = new vect{ a.elements->at(0) };
            for (int i = 1; i < a.elements->size(); i++) {
                elements->push_back(a.elements->at(i));
            }
        }
    }

MySet(const vect & vec) {
    if (vec.empty()) {
        size = 0;
        elements = NULL;
    }
    else {
        vect temp = vec;
        qsort(&temp.at(0), temp.size(), sizeof(INF), compare<INF>);
        elements = new vect{ temp.at(0) };
        for (int i = 1; i < temp.size(); i++) {
            if (temp.at(i - 1) != temp.at(i)) {
                elements->push_back(temp.at(i));
            }
        }
        size = elements->size();
    }
}

~MySet(){
    if (size != 0) {
        elements->~vect();
        elements = NULL;
    }
}

bool IsElement(INF a) {
    if (Search(a) < 1) {
        return false;
    }
    else {
        return true;
    }
}

void AddElement(INF a) {
    if (size == 0) {
        MySet(a);
    }
    else {
        if (!IsElement(a)) {
            int i;
            for (i = 0; i < size; i++) {
                if (a < elements->at(i)) {
                    break;
                }
            }
            elements->insert(elements->begin() + i, a);
        }
    }
}

void DeleteElement(INF a) {
    int i = Search(a);
    if (i > 0) {
        elements->erase(elements->begin() + i - 1);
        size--;
    }
}

MySet& operator= (const MySet & a) {

```

```

    if (this == &a) {
        return *this;
    }
    if (size != 0) {
        elements->~vect();
    }
    if (a.size != 0) {
        size = a.size;
        elements = new vect(*a.elements);
        return *this;
    }
    else {
        size = 0;
        elements = NULL;
        return *this;
    }
}

MySet& operator+ (MySet & a) {
    if (a.size == 0) {
        return *this;
    }
    if (size == 0) {
        MySet* rs = new MySet(a);
        return *rs;
    }
    vect* temp1 = new vect;
    int i = 0, j = 0;
    while ((i < size) && (j < a.size)) {
        if (elements->at(i) == a.elements->at(j)) {
            temp1->push_back(elements->at(i));
            i++;
            j++;
        }
        else {
            if (elements->at(i) < a.elements->at(j)) {
                temp1->push_back(elements->at(i));
                i++;
            }
            else {
                temp1->push_back(a.elements->at(j));
                j++;
            }
        }
    }
    if (i == size) {
        for (j; j < a.size; j++) {
            temp1->push_back(a.elements->at(j));
        }
    }
    else {
        for (i; i < size; i++) {
            temp1->push_back(elements->at(i));
        }
    }
    MySet* temp2 = new MySet;
    temp2->size = temp1->size();
    temp2->elements = temp1;
    return *temp2;
}

MySet& operator- (MySet & a) {
    if (a.size == 0 || size == 0) {
        return *this;
    }
    vect* temp1 = new vect;
    for (int i = 0; i < size; i++) {
        if (a.IsElement(elements->at(i)) == false) {
            temp1->push_back(elements->at(i));
        }
    }
}

```

```

        }
    }
    MySet* temp2 = new MySet;
    temp2->size = temp1->size();
    temp2->elements = temp1;
    return *temp2;
}

MySet& operator* (MySet & a) {
    if (a.size == 0) {
        MySet* rs = new MySet(a);
        return *rs;
    }
    if (size == 0) {
        return *this;
    }
    vect* temp1 = new vect;
    for (int i = 0; (i < size) && (i < a.size); i++) {
        if ((IsElement(elements->at(i)) == true) && (a.IsElement(elements->at(i)) ==
true)) {
            temp1->push_back(elements->at(i));
        }
    }
    MySet* temp2 = new MySet;
    temp2->size = temp1->size();
    temp2->elements = temp1;
    return *temp2;
}

bool operator== (MySet & a) {
    if (size != a.size) {
        return false;
    }
    else {
        for (int i = 0; i < size; i++) {
            if (elements->at(i) != a.elements->at(i)) {
                return false;
            }
        }
        return true;
    }
}

MySet operator+= (MySet & a) {
    *this = *this + a;
    return *this;
}

MySet operator-= (MySet & a) {
    *this = *this - a;
    return *this;
}

MySet operator*= (MySet & a) {
    *this = *this * a;
    return *this;
}

friend std::ostream& operator<< (std::ostream & out, MySet set) {
    if (set.size == 0) {
        out << "{ }";
        return out;
    }
    int i;
    out << "{ ";
    for (i = 0; i < set.size - 1; i++) {
        out << set.elements->at(i) << ", ";
    }
    out << set.elements->at(i) << " }";
}

```

```

        return out;
    }

};

int main()
{
    setlocale(LC_ALL, "Russian");


    MySet<string> a("1"), b, c(a), d("8"), e(a + d), f(d - a), j(e * b);
    cout << a << endl << b << endl << c << endl << d << endl << e << endl << f << endl << j <<
endl << endl;

    MySet<string> aa({ "6", "3", "1", "8" }), bb({"4", "2", "6", "8"});
    cout << "A = " << aa << endl << "B = " << bb << endl << "A + B = " << aa + bb << endl << "A *
B = " << aa * bb << endl << "A - B = " << aa - bb << endl;

    system("pause");
    return 0;
}

```

4. Анализ результатов

 C:\Users\webma\source\repos\lab4\Debug\lab4.exe

```

{ 1 }
{ }
{ 1 }
{ 8 }
{ 1, 8 }
{ 8 }
{ }

```

```

A = { 1, 3, 6, 8 }
B = { 2, 4, 6, 8 }
A + B = { 1, 2, 3, 4, 6, 8 }
A * B = { 6, 8 }
A - B = { 1, 3 }

```

Press any key to continue . . . █

Программа работает исправно.