Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

Факультет (ИУ) «Информатика, искусственный интеллект и системы управления»

Кафедра Информационная безопасность (ИУ8)

Лабораторная работа № 2

ПО КУРСУ

«Алгоритмические языки»

на тему «Изучение перегрузки стандартных операций в языке C++

Студент ИУ8-24 (Группа) А. А. Урнышева (И. О. Фамилия)

Преподаватель: Д. В. Барыкин (И.О. Фамилия)

Цель работы: овладение навыками разработки программ на языке C++, использующих перегрузку стандартных операций.

Условие задачи: Дан класс (например, с именем *Vector*), задающий вектор размерности *n*. Поля класса: указатель на массив, задающий вектор (тип элемента *double*), массив должен создаваться динамически; число элементов (размерность) вектора (тип *int*). Класс включает: конструктор без параметров, задающий пустой вектор (число элементов равно 0); конструктор, создающий объект вектор на основе обычного одномерного массива размерности *n*; конструктор копирования, конструктор перемещения, деструктор.

Необходимо перегрузить операции и продемонстрировать их работу. Перегрузить операцию *[]* (обращение к элементу вектора по индексу), операцию = (копирование вектора или создание копии вектора), операцию * (умножение числа на вектора), на выходе вектор такой же размерности, каждый элемент которого равен произведению соответствующего элемента исходного вектора на число.

Вариант 24: < сравнение двух векторов (массивов) сравнение проводится последовательно по элементам. Результат истинно, если элемент первого операнда меньше элемента второго элемента, если элемент первого операнда больше элемента второго элемента, результат ложь, в случае равенства переход к следующий паре элементов. Если в каком либо операнде элементы закончились, то результат ложь.

Текст программы с комментариями:

Файл Vector.h:

```
#pragma once
#include <iostream>
#include <fstream>

using namespace std;

class Vector { //класс вектор

   double* p = nullptr; //указатель на массив
   int n = 0; //размерность вектора
```

```
public:
    Vector(double* p, int n) {//коструктор, создающий одномерный массив размерности n
        this->n = n; // Задаем число элементов
        this->p = new double[n]; // Выделяем память
        for (int i = 0; i < n; i++) this->p[i] = p[i]; // Копируем один массив в другой
        // cout << "Vector(double *p, int n)" << endl;
    }

    Vector(int n) : n(n) { // Конструктор - выделяем память без инициализации
        p = new double[n];
        //cout << "Vector(int n)" << endl;
    }

    Vector(const Vector& V) { //конструктор копирования
        n = V.n;
        p = new double[n];
        for (int i = 0; i < n; i++)
            p[i] = V.p[i];
    }

    Vector(Vector&& V) { //конструктор перемещения
        swap(p, V.p);
        swap(n, V.n);
    }
    void print() const {  //печать вектора (массива), далее заменено на перегрузку <<
        cout << "n = " << n << endl;
        for (int i = 0; i < n; i++)
            cout << p[i] << " ";
        cout << std::endl;
    }
    Vector() { //конструктор без параметров, задает пустой объект
        p = nullptr; n = 0;
    }

    double& operator[](int index) { //перегрузка оператора обращения по индексу
        return p[index];
    }
    Vector& operator =(const Vector& v2) { //перегрузка оператора копирования объекта
        if (this != &v2) { //запрет копирования вектора самого в себя
            n = v2.n;
            if (p != nullptr) delete[] p; //освобождаем память старого вектора
            p = new double[n]; //выделяем память для нового вектора
            for (int i = 0; i < n; i++) {
                p[i] = v2.p[i]; //копируем каждый элемент
            }
        }
        return *this; //возвращаем ссылку на текущий объект
    }

    Vector& operator =(Vector&& v2) { //перегрузка операции перемещения объекта
        if (this != &v2) { //запрет перемещения вектора самого в себя
            std::swap(p, v2.p);
            std::swap(n, v2.n);
        }
        return *this; //возвращаем ссылку на текущий объект
    }
    ~Vector() { //деструктор
        if (p != nullptr) delete[] p; //освобождаем память
    }

    //дружественные функции, определенные вне класса, для перегрузки математических операций

    friend Vector operator *(double x, Vector& v2);
    friend Vector operator +(const Vector& v1, const Vector& v2);
    friend bool operator <(const Vector& v1, const Vector& v2);
```

```
    //дружественные функции, определенные вне класса, для перегрузки операций ввода и вывода
    friend ostream& operator <<(ostream& os, const Vector& v1);
    friend istream& operator >>(istream& is, const Vector& v1);
};

Vector operator *(double x, Vector& v2) //перегрузка оператора умножения вектора на число
{
    Vector V(v2.n); //создаем новый объекта заданной размерности
    for (int i = 0; i < v2.n; i++) {
        V.p[i] = x * v2.p[i]; //заполняем массив
    }
    return V; // Возвращаем объект
}

Vector operator +(const Vector& v1, const Vector& v2) { //перегрузка оператора сложения двух векторов
    Vector V(v1.n + v2.n); //создаем новый объект заданного размера
    for (int i = 0; i < v1.n; i++) {
        V.p[i] = v1.p[i]; //заполняем массив
    }
    for (int i = 0; i < v2.n; i++) {
        V.p[i + v1.n] = v2.p[i]; //заполняем массив
    }
    return V; //возвращаем объект
}

bool operator <(const Vector& v1, const Vector& v2) { //перегрузка оператора сравнения двух векторов
    bool f = true;
    if (v1.n != v2.n) { //если их длины разные, то выводим false
        f = false;
    }
    for (int i = 0; i < v1.n; ++i) {
        if (v1.p[i] >= v2.p[i]) { //если элемент v1 >= элемент v2, то выводим false
            f = false;
            break;
        }
    }
    return f;
}

ostream& operator <<(ostream& os, const Vector& v1) { //перегрузка операции вывода
    for (int i = 0; i < v1.n; ++i) {
        os << v1.p[i] << ' ';
    }
    return os;
}

istream& operator >>(istream& is, const Vector& v1) { //перегрузка операции ввода
    for (int i = 0; i < v1.n; ++i) {
        is >> v1.p[i];
    }
    return is;
            }
```

Файл main.cpp:

```
#include "Vector.h"

void ReadFromFile(const string& filename, Vector& v1, Vector& v2) { //функция для чтения элементов класса из
файла
    ifstream file(filename);
    if (file.is_open()) { //проверяем, открыт ли файл
        int n1, n2;
        file >> n1;
        v1 = Vector(n1);
```

```
        file >> v1;
        file >> n2;
        v2 = Vector(n2);
        file >> v2;
        file.close();
    }
    else {
        cerr << "Can't open this file :(" << endl;
    }
}

void SaveToFile(const string& filename, const Vector& v1) {
    ofstream file(filename);
    if (file.is_open()) {
        file << v1;
        file.close();
    }
    else {
        cerr << "Can't open this file :(" << endl;
    }
}

int main()
{
    Vector V1, V2, V3;
    ReadFromFile("input.txt", V1, V2);
    cout << "Vector 1 " << V1 << endl;
    cout << "Vector 2 " << V2 << endl;
    V3 = V1 + V2;
    bool b = V1 < V2;
    cout << "b = " << b << endl;
    cout << "Sum of V1 and V2 = " << V3 << endl;
    SaveToFile("output.txt", V3);
    return 0;
}
```

Файл input.txt

4

1 2 3 4

4

5 6 7 8

Файл output.txt

1 2 3 4 5 6 7 8

Вывод: овладели навыками разработки программ на языке C++, использующих перегрузку стандартных операций.