Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

**Лабораторная работа № 2. Изучение перегрузки стандартных операций в языке Си++**

Работу выполнил:

Кулиев Орхан Микаел оглы

ИУ8-24

Вариант 13

**Цель**

Цель работы состоит в овладении навыками разработки программ на языке Си++, 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 *int* или *double* в зависимости от варианта), 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 = (присваивание с копированием), операцию = (присваивание с перемещением), а также операцию вставки (<<) объекта в поток cout или в файл (объект класса ostream) и операцию извлечения (>>) объекта из потока cin или из файла (объект класса istream).

**ТЕКСТ ПРОГРАММЫ**

#include <iostream>

#include <fstream>

using namespace std;

class Vector {

double\* p = nullptr; // Указатель на массив, задающий вектор

int n = 0; //Размерность вектора(число элементов)

public:

// Конструктор, создающий объект вектор на основе обычного одномерного массива размерности n

Vector(double\* p, int n) {

this->n = n;

this->p = new double[n]; // Выделение памяти под вектор

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

this->p[i] = p[i];// Копирование значений из переданного массива

}

}

// Конструктор, создающий пустой объект вектор заданной размерности

Vector(int n) :n(n) {

p = new double[n];// Выделение памяти под вектор

}

// Конструктор копирования

Vector(const Vector& V) {

n = V.n;

p = new double[n];

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

p[i] = V.p[i];

}

}

// Конструктор перемещения

Vector(Vector&& V) {

swap(p, V.p);// Обмен указателями на массивы

swap(n, V.n);// Обмен значениями размерностей

}

// Функция печати вектора

void print() const {

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

cout << p[i] << " ";

}

cout << endl;

}

// Перегрузка оператора доступа к элементу вектора по индексу

double& operator[](int index) {

return p[index];

}

// Перегрузка оператора присваивания с копированием

Vector& operator =(const Vector& v2) {

if (this != &v2) {

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 значения из другого вектора

}

}

return \*this; // Возвращаем ссылку на текущий объект

}

// Перегрузка оператора присваивания с перемещением

Vector& operator=(Vector&& v2) {

if (this != &v2) {

swap(p, v2.p); // Обмен указателями на массивы

swap(n, v2.n); // Обмен значениями размерностей

}

return \*this;// Возвращаем ссылку на текущий объект

}

// Перегрузка оператора вставки в поток вывода

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

out << "(";

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

out << vec.p[i]; // Вывод элементов вектора через запятую

if (i < vec.n - 1) {

out << ", ";

}

}

out << ")";

return out;

}

// Перегрузка оператора извлечения из потока ввода

friend istream& operator>>(istream& in, Vector& ve) {

for (int i = 0; i < ve.n; ++i) {

in >> ve.p[i];// Ввод элементов вектора из потока

}

return in;

}

// Скалярное произведение векторов

friend double operator\*(const Vector& ve1, const Vector& ve2) {

if (ve1.n != ve2.n) {

throw std::invalid\_argument("Vectors must be of the same size"); // Проверка размеров векторов

}

double r = 0;

for (int i = 0; i < ve1.n; ++i) {

r += ve1.p[i] \* ve2.p[i]; // Вычисление скалярного произведения

}

return r;

}

// Деструктор

~Vector() {

if (p != nullptr) delete[] p; // Освобождаем память, если она была выделена

}

};

int main()

{

ifstream input("input.txt"); // Открыть файл для чтения

ofstream output("output.txt"); // Открыть файл для записи

double x[6], y[6];

for (int i = 0;i < 6;i++) {

input >> x[i];// Чтение данных для первого вектора из файла

}

for (int i = 0;i < 6;i++) {

input >> y[i];// Чтение данных для второго вектора из файла

}

Vector ve1(x, 6);// Создание первого вектора

Vector ve2(y, 6);// Создание второго вектора

output << "Vector 1:" << ve1 << endl;// Вывод первого вектора в файл

output << "Vector 2:" << ve2 << endl;// Вывод второго вектора в файл

output << "scalar product:" << ve1 \* ve2;// Вывод скалярного произведения векторов

}

Данные, поступающие на вход программы:  
1 2 3 4 5 6

7 8 9 10 11 12

Данные, выходящие из программы:

Vector 1: (1, 2, 3,4, 5, 6)

Vector 2: (7,8,9,10,11,12)

Scalar product: 217

Вывод: овладел навыками разработки программ на языке Си++, использующих перегрузку стандартных операций.