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 выполнила:

Степаненко Александра, ИУ8-24

**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 {

int\* p = nullptr; // Указатель на массив значений вектора

int n = 0; // Размер вектора

public:

Vector(int\* p, int n) { // Конструктор для создания вектора из массива значений

this->n = n;

this->p = new int[n];

for (int i = 0; i < n; i++) this->p[i] = p[i];

}

Vector(int n) : n(n) { // Конструктор для создания пустого вектора заданного размера

p = new int[n]();

}

Vector(const Vector& V) {

n = V.n;

p = new int[n];

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

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

}

Vector(Vector&& V) {

std::swap(p, V.p);

std::swap(n, V.n);

}

void print() const { // Метод для вывода значений вектора

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

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

cout << endl;

}

int& operator[](int index) { // Перегруженный оператор индексации

return p[index];

}

Vector& operator =(const Vector& v2) { // Перегруженный оператор присваивания с копированием

if (this != &v2) { // Проверка, чтобы не было исключений

n = v2.n;

if (p != nullptr) delete[] p;

p = new int[n];

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

p[i] = v2.p[i];

}

return \*this;

}

Vector& operator =(Vector&& v2) { // Перегруженный оператор присваивания с перемещением

if (this != &v2) { // Проверка, чтобы вызвалось без исключений

std::swap(p, v2.p);

std::swap(n, v2.n);

}

return \*this;

}

bool operator<(const Vector& v2) const {

if (n != v2.n) {

return false;

}

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

if (p[i] >= v2.p[i]) {

return false;

}

}

return true;

}

friend ostream& operator<<(ostream& os, const Vector& v) { // Перегруженный оператор вывода в поток

for (int i = 0; i < v.n; i++) // Вывод значений вектора через поток вывода

os << v.p[i] << " ";

return os;

}

friend istream& operator>>(istream& is, Vector& v) { // Перегруженный оператор ввода из потока

for (int i = 0; i < v.n; i++) // Считывание значений вектора из потока ввода

is >> v.p[i];

return is;

}

~Vector() { // Деструктор класса

delete[] p;

}

};

int main() {

ifstream input("input.txt");

int size = 0;

int size1 = 0;

input >> size;

input >> size1;

int\* values = new int[size];

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

input >> values[i];

}

int\* values1 = new int[size1];

for (int i = 0; i < size1; ++i) {

input >> values1[i];

}

Vector v1(values, size);

Vector v2(values1, size1);

bool res = v1 < v2;

cout << res;

delete[] values;

delete[] values1;

}

**ВЫВОД**

В данной работе мы создали класс Vector, который представляет собой вектор и обладает следующими функциональностями:

Возможность создания вектора с помощью конструктора, который принимает массив значений и их количество, а также конструктора, который принимает только количество элементов вектора.

Реализацию конструктора копирования и конструктора перемещения, что позволяет корректно копировать и перемещать объекты класса.

Оператор доступа к элементам вектора [], позволяющий получить доступ к элементам вектора по индексу.

Оператор присваивания для копирования и перемещения, обеспечивающий корректное копирование и перемещение объектов класса.

Оператор сравнения двух векторов

Методы output и input, позволяющие выводить вектор в поток вывода и считывать вектор из потока ввода соответственно.

Деструктор, который освобождает память, выделенную для массива значений вектора.

Перегрузку операторов << и >>, которые позволяют выводить и считывать объекты класса Vector с помощью потоков cout и cin соответственно.

Для демонстрации работы класса был написан простой тестовый код, в котором создаются 2 вектора и сравниваются между собой.

Этот класс может быть полезен при работе с векторами чисел в программировании на C++.