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:

Мирошниченко Максим Алексеевич

ИУ8-24

Вариант 15

**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 <vector>

class Vector {

private:

int\* data;

int size;

public:

// Конструктор на основе массива

Vector(int\* arr, int n) : size(n) {

data = new int[size];

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

data[i] = arr[i];

}

}

// Конструктор с заданной размерностью

Vector(int n) : size(n) {

data = new int[size];

}

// Конструктор копирования

Vector(const Vector& other) : size(other.size) {

data = new int[size];

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

data[i] = other.data[i];

}

}

// Конструктор перемещения

Vector(Vector&& other) noexcept : data(other.data), size(other.size) {

other.data = nullptr;

other.size = 0;

}

// Деструктор

~Vector() {

delete[] data;

}

// Операция [] для неконстантных объектов

int& operator[](int index) {

if (index < 0 || index >= size) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

// Операция [] для константных объектов

const int& operator[](int index) const {

if (index < 0 || index >= size) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

// Операция = (копирование)

Vector& operator=(const Vector& other) {

if (this != &other) {

delete[] data;

size = other.size;

data = new int[size];

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

data[i] = other.data[i];

}

}

return \*this;

}

// Операция = (перемещение)

Vector& operator=(Vector&& other) noexcept {

if (this != &other) {

delete[] data;

data = other.data;

size = other.size;

other.data = nullptr;

other.size = 0;

}

return \*this;

}

// Операция <<

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

// Операция >>

friend std::istream& operator>>(std::istream& in, Vector& vec);

// Операция ^ (побитовое XOR)

friend Vector operator^(const Vector& vec1, const Vector& vec2);

};

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

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

out << vec.data[i] << " ";

}

return out;

}

std::istream& operator>>(std::istream& in, Vector& vec) {

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

in >> vec.data[i];

}

return in;

}

Vector operator^(const Vector& vec1, const Vector& vec2) {

if (vec1.size != vec2.size) {

throw std::invalid\_argument("Vector sizes must be equal");

}

Vector result(vec1);

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

result[i] ^= vec2[i];

}

return result;

}

int main() {

int arr1[] = { 1, 2, 3 };

int arr2[] = { 4, 5, 6 };

Vector v1(arr1, 3);

Vector v2(arr2, 3);

Vector v3 = v1 ^ v2;

std::cout << "v1: " << v1 << std::endl;

std::cout << "v2: " << v2 << std::endl;

std::cout << "v3 = v1 ^ v2: " << v3 << std::endl;

return 0;

}

Данные, поступающие на вход программы:  
1 2 3

4 5 6

Данные, выходящие из программы:

Vector 1: (1, 2, 3)

Vector 2: (4, 5, 6)

Scalar product: 5 7 5

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