|  |  |
| --- | --- |
|  | **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 | «Информатика, искусственный интеллект и системы управления» (ИУ) |

|  |  |
| --- | --- |
| КАФЕДРА | «Информационная безопасность» (ИУ8) |

Лабораторная работа № 2

ПО КУРСУ

«Алгоритмические языки»

на тему «Изучение перегрузки стандартных операций в языке Си++»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-24 |  |  |  | Соболева Д.Е. |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | Барыкин Д.В.  Быков А.Ю. |
|  |  |  |  |  | (И.О. Фамилия) |

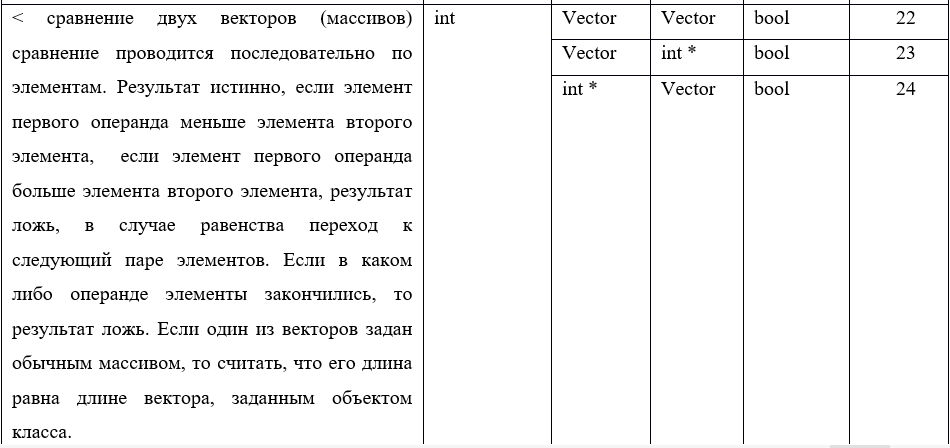
**Цель работы** состоит в овладении навыками разработки программ на языке Си++, использующих перегрузку стандартных операций.

**Вариант 22**

**Задача**

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). Также продемонстрировать разницу между конструктором копирования и конструктором перемещения и между операциями присваивания с копированием и перемещением. Исходные коды класса разместить в двух файлах: в заголовочном файле класса и файле реализации класса. При выполнении работы все входные данные читаются из текстового файла input.txt (создать этот файл любым текстовым редактором), результаты выводятся в файл output.txt. В отчете представить содержимое этих файлов.



**Код программы:**

*Заголовочный файл класса:*

#pragma once

#ifndef VECTOR\_H

#define VECTOR\_H

#include <iostream>

class Vector {

private:

int\* array;

int size;

public:

Vector();

Vector(int n);

Vector(const int\* arr, int n);

Vector(const Vector& other);

Vector(Vector&& other) noexcept;

~Vector();

int& operator[](int index);

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

Vector& operator=(const Vector& other);

Vector& operator=(Vector&& other);

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

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

bool operator<(const Vector& other) const;

};

#endif

*Файл реализации класса:*

#include "Vector.h"

#include <iostream>

#include <fstream>

Vector::Vector() : array(nullptr), size(0) {}

Vector::Vector(int n) : size(n) {

array = new int[n];

}

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

array = new int[n];

std::copy(arr, arr + n, array);

}

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

array = new int[other.size];

std::copy(other.array, other.array + other.size, array);

}

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

other.array = nullptr;

other.size = 0;

}

Vector::~Vector() {

delete[] array;

}

int& Vector::operator[](int index) {

return array[index];

}

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

return array[index];

}

Vector& Vector::operator=(const Vector& other) {

if (this != &other) {

delete[] array;

size = other.size;

array = new int[other.size];

std::copy(other.array, other.array + other.size, array);

}

return \*this;

}

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

if (this != &other) {

delete[] array;

array = other.array;

size = other.size;

other.array = nullptr;

other.size = 0;

}

return \*this;

}

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

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

os << vec.array[i] << " ";

}

return os;

}

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

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

is >> vec.array[i];

}

return is;

}

bool Vector::operator<(const Vector& other) const {

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

if (array[i] < other.array[i]) return true;

else if (array[i] > other.array[i]) return false;

}

return false;

}

void readVectorsFromFile(const std::string& filename, Vector& vec1, Vector& vec2) {

std::ifstream file(filename);

if (file.is\_open()) {

int size1, size2;

file >> size1;

vec1 = Vector(size1);

file >> vec1;

file >> size2;

vec2 = Vector(size2);

file >> vec2;

file.close();

}

else {

std::cerr << "Error opening file for reading." << std::endl;

}

}

int main() {

std::ifstream inputFile("input.txt");

std::ofstream outputFile("output.txt");

Vector vec1, vec2;

readVectorsFromFile("input.txt", vec1, vec2);

std::cout << "Vector 1: " << vec1 << std::endl;

std::cout << "Vector 2: " << vec2 << std::endl;

outputFile << "Vector 1: " << vec1 << std::endl;

outputFile << "Vector 2: " << vec2 << std::endl;

if (vec1 < vec2) {

outputFile << "Vector 1 is less than Vector 2" << std::endl;

}

else {

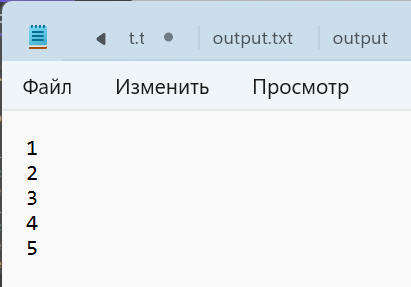
outputFile << "Vector 1 is not less than Vector 2" << std::endl;

}

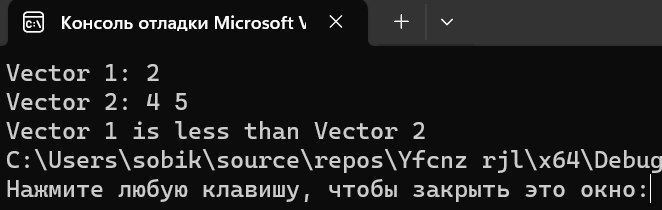
return 0;

}

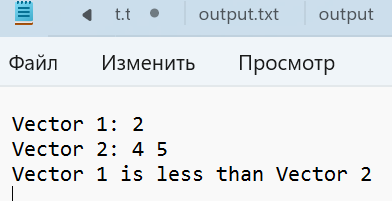
Данные из Блокнота:



Консоль:



Данные сохраненные в output.txt :



**Контрольный расчёт:** Подставив данные параметры в калькулятор и пересчитав руками, я получила точно такие же ответы.

**Вывод:** Выполнив лабораторную работу №2, я изучила тему «Изучение перегрузки стандартных операций в языке Си++» и применила классы, объекты и перегрузки операций на практике.