|  |  |
| --- | --- |
|  | **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 и системы управления» (ИУ) |

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

Лабораторная работа № 2

ПО КУРСУ

«Алгоритмические языки»

на тему «Перегрузка операций»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-23 |  |  |  | В. С. Ажгирей |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | М. В. Малахов |
|  |  |  |  |  | (И.О. Фамилия) |

2024

Введение

Цели и задачи работы

Цель работы состоит в овладении навыками разработки программ на языке Си++, использующих перегрузку стандартных операций. Для достижения цели необходимо выполнить следующие задачи:

- изучить необходимые учебные материалы, посвященные перегрузке стандартных операций в языке Си++ ;

- разработать программу на языке Си++ для решения заданного варианта задания;

- отладить программы;

- выполнить решение контрольного примера с помощью программы и ручной расчет контрольного примера;

- подготовить отчет по лабораторной работе.

Общие условия

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.*

Условие для 1 варианта

Описание операции перегруженной операции: + сложение векторов одинаковой размерности, на выходе вектор такой же размерности элемент которого равен сумме соответствующих элементов двух векторов

Тип элемента вектора (массива): double

Типы операндов и результата для перегруженной операции:

Первый операнд: Vector

Второй операнд: Vector

Результат: Vector

Основная часть

Исходный текст программы:

Файл заголовка sources.hpp:

#pragma once

#include <fstream>

#include <iostream>

#include <typeinfo>

class Vector

{

double\* data\_;

size\_t size\_ = 0;

public:

Vector();

Vector(const double\*, size\_t);

Vector(const Vector&);

Vector(Vector&&);

double& operator[](size\_t);

Vector& operator=(const Vector&);

Vector& operator=(Vector&&);

Vector operator+(const Vector&) const;

friend std::ostream& operator<<(std::ostream& output\_stream, const Vector& vector);

friend std::istream& operator>>(std::istream& input\_stream, Vector& vector);

Vector& operator++();

Vector operator++(int);

~Vector();

};

Файл описания sources.cpp:

#include "sources.hpp"

Vector::Vector() : data\_(nullptr), size\_(0) {}

Vector::Vector(const double\* array, size\_t size) : size\_(size)

{

data\_ = new double[size\_];

for (size\_t i = 0; i < size\_; ++i)

{

data\_[i] = array[i];

}

}

Vector::Vector(const Vector& other)

{

if (this != &other)

{

size\_ = other.size\_;

data\_ = new double[size\_];

for (size\_t i = 0; i < size\_; ++i)

{

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

}

std::cout << "The copy constructor is called" << std::endl;

}

}

Vector::Vector(Vector&& other)

{

if (this != &other)

{

data\_ = other.data\_;

size\_ = other.size\_;

other.data\_ = nullptr;

other.size\_ = 0;

std::cout << "The move constructor is called" << std::endl;

}

}

double& Vector::operator[](size\_t index)

{

return data\_[index];

}

Vector Vector::operator+(const Vector& other) const

{

if (size\_ != other.size\_)

{

return Vector();

}

Vector result(other);

for (size\_t i = 0; i < size\_; ++i)

{

result.data\_[i] += data\_[i];

}

return result;

}

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

{

if (this != &other)

{

delete[] data\_;

size\_ = other.size\_;

data\_ = new double[size\_];

for (size\_t i = 0; i < size\_; ++i)

{

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

}

}

return \*this;

}

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

{

if (this != &other)

{

delete[] data\_;

data\_ = other.data\_;

size\_ = other.size\_;

other.data\_ = nullptr;

other.size\_ = 0;

}

return \*this;

}

Vector::~Vector()

{

delete[] data\_;

data\_ = nullptr;

}

std::ostream& operator<<(std::ostream& output\_stream, const Vector& vector)

{

for (size\_t i = 0; i < vector.size\_; ++i)

{

output\_stream << vector.data\_[i] << " ";

}

output\_stream << std::endl;

return output\_stream;

}

std::istream& operator>>(std::istream& input\_stream, Vector& vector)

{

if (typeid(std::cin) == typeid(input\_stream))

std::cout << "Enter lenght of vector: ";

size\_t size\_;

input\_stream >> size\_;

double\* array = new double[size\_];

for (size\_t i = 0; i < size\_; ++i)

{

input\_stream >> array[i];

}

vector = Vector(array, size\_);

delete[] array;

return input\_stream;

}

Vector& Vector::operator++()

{

for (size\_t i = 0; i < size\_; ++i)

{

data\_[i] += 1;

}

return \*this;

}

Vector Vector::operator++(int a)

{

Vector temp(\*this);

for (size\_t i = 0; i < size\_; ++i)

{

data\_[i] += 1;

}

return temp;

}

Исполняемый файл main.cpp:

#include "sources.hpp"

int main()

{

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

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

Vector vector1, vector2;

inputFile >> vector1; // Чтение объекта типа Vector из потока ввода

outputFile << vector1; // Вывод объекта типа Vector в поток вывода

inputFile >> vector2; // Чтение объекта типа Vector из потока ввода

outputFile << vector2; // Вывод объекта типа Vector в поток вывода

Vector vector = vector1 + vector2; // Перегруженная операция сложения

outputFile << vector;

vector2 = vector1; // Конструктор копирования

Vector vector3(std::move(vector1)); // Конструктор перемещения

vector2[0] = 10; // Обращение к элементу вектора по индексу

vector3 = vector2; // Операция присваивания с копированием

Vector vector4;

vector4 = std::move(vector3); // Операция присваивания с перемещением

outputFile << vector2; // Вставка объекта в поток вывода

outputFile << vector4; // Вставка объекта в поток вывода

inputFile.close();

outputFile.close();

return 0;

}

Снимки выполнения работы программы

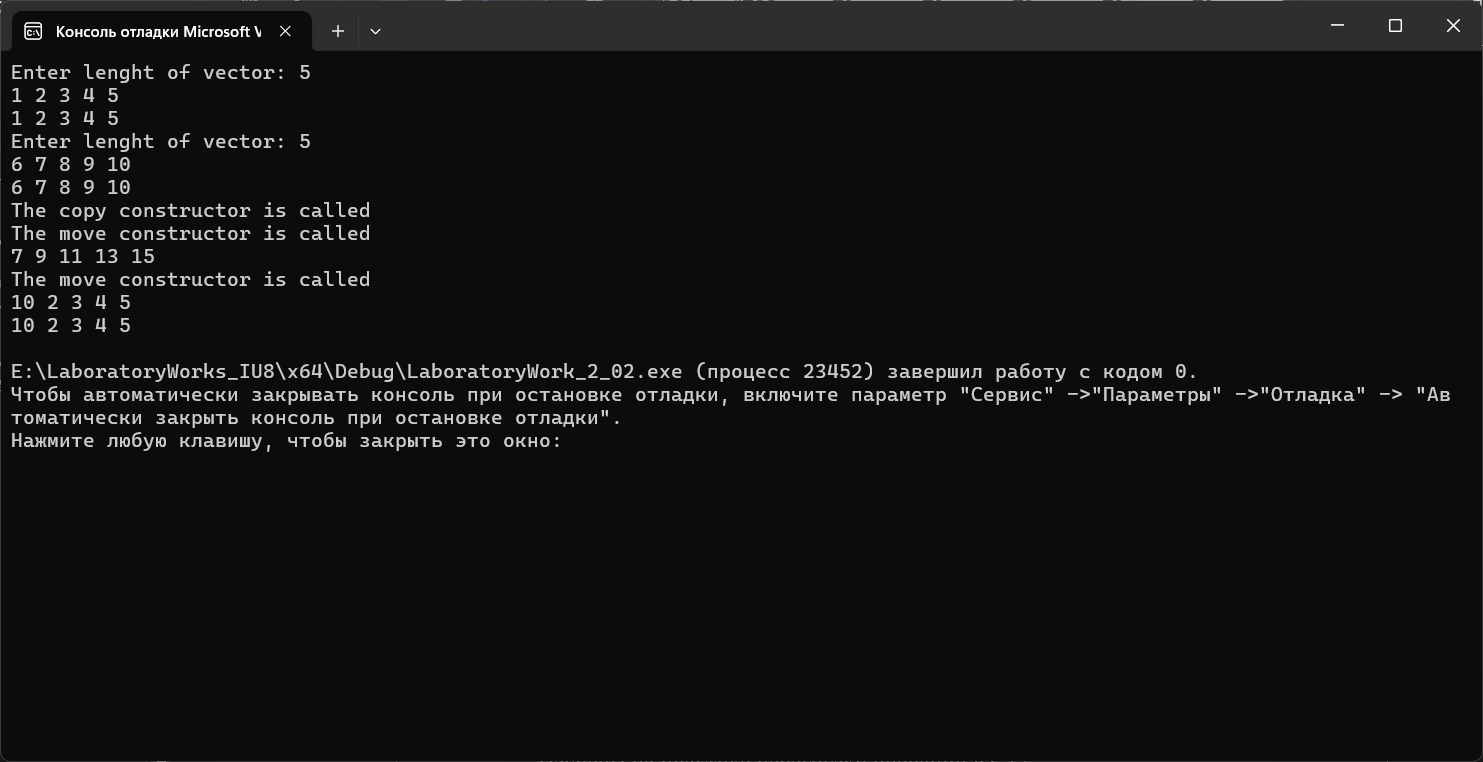
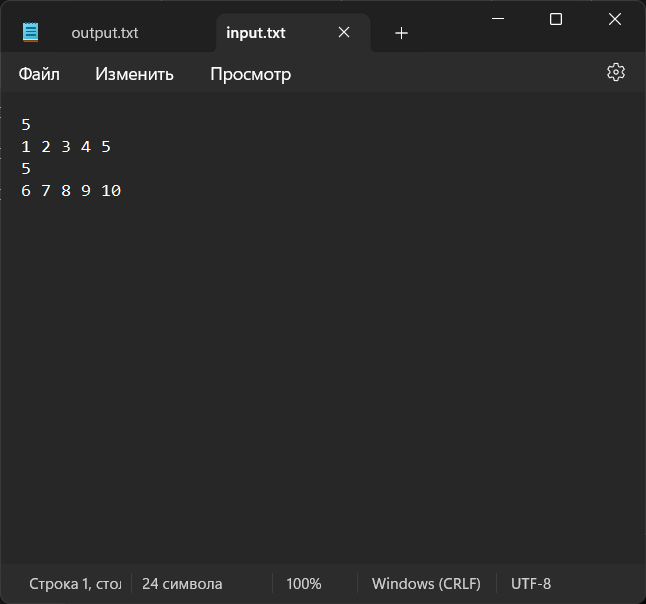


Рисунок 1 – Запуск программы с считыванием и выводом в консоль



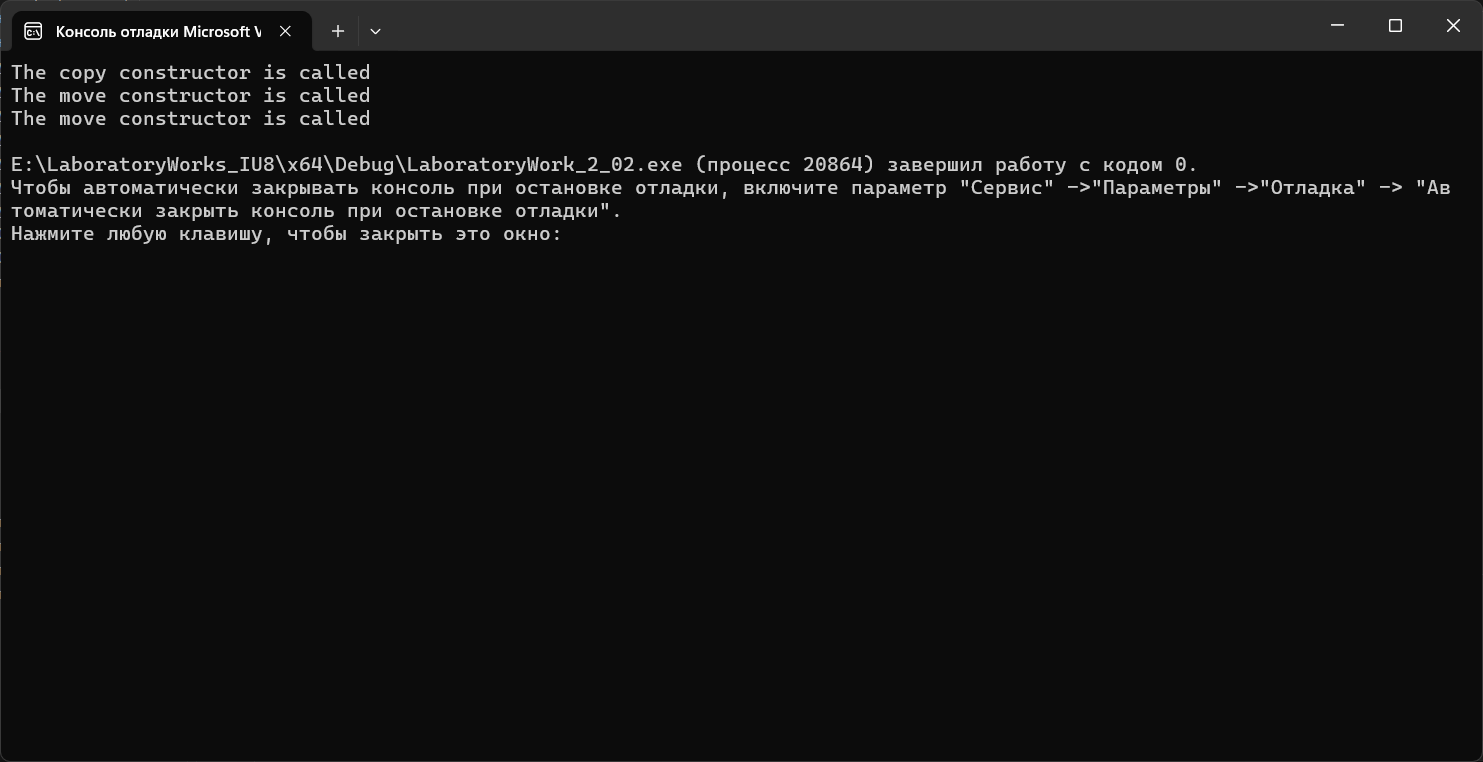


Рисунок 2-3 – Запуск программы с считыванием из файла input.txt и выводом в файл output.txt

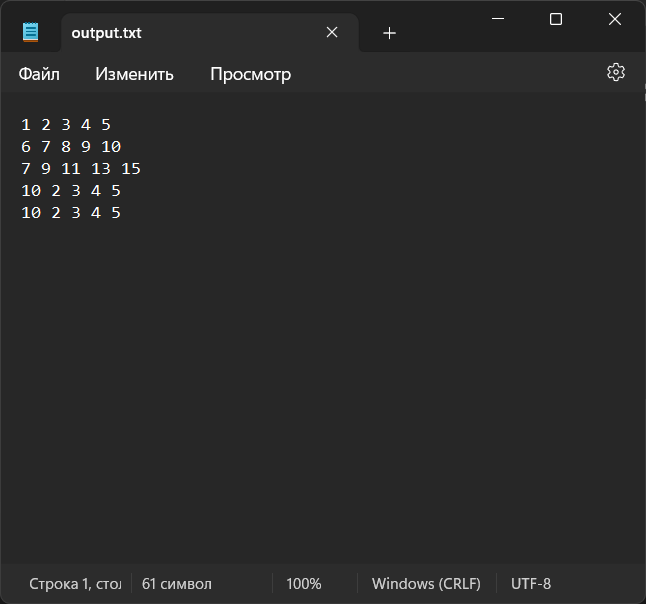


Рисунок 4 – Выходные данные

Заключение

Задачи лабораторной работы были решены, результаты проверены. Изучены на практике перегрузки операций в С++.