



**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

(Группа)

В. С. Ажгирей

(И. О. Фамилия)

Преподаватель:

М. В. Малахов

(И.О. Фамилия)

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

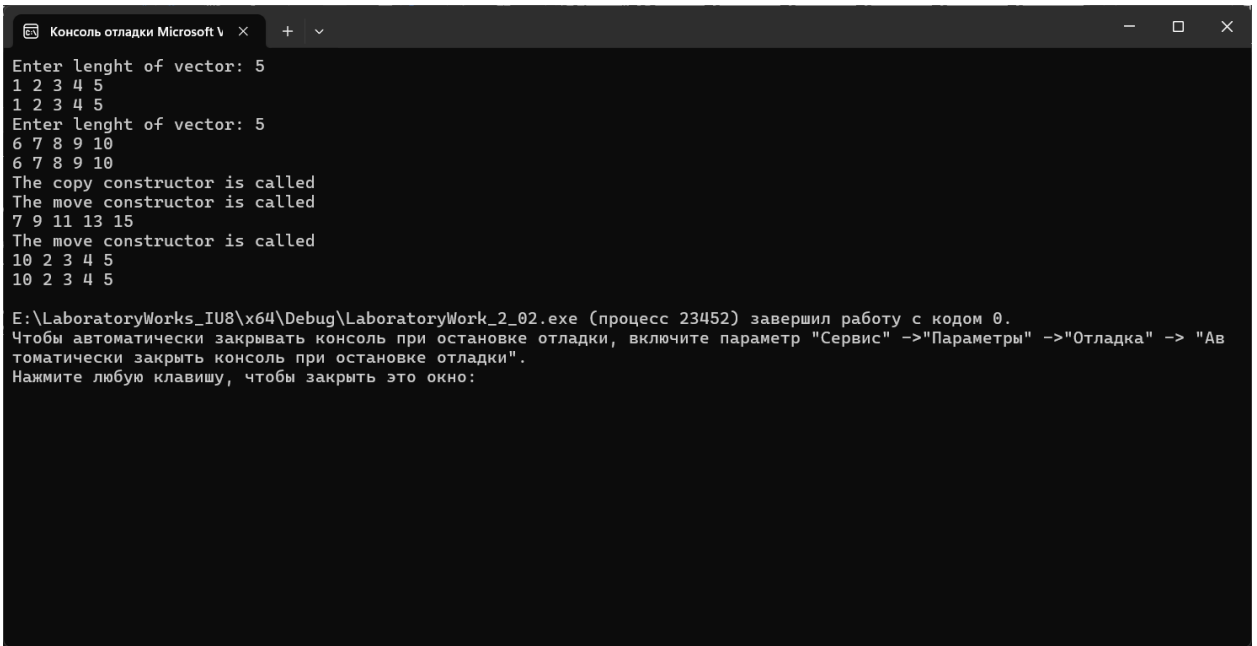
    outputFile << vector2; // Вставка объекта в поток вывода
    outputFile << vector4; // Вставка объекта в поток вывода

    inputFile.close();
    outputFile.close();

    return 0;
}

```

Снимки выполнения работы программы

A screenshot of a Microsoft Visual Studio debug console window. The window title is "Консоль отладки Microsoft V". The console output shows the following sequence of events: 1. Prompt "Enter lenght of vector: 5" followed by input "1 2 3 4 5". 2. Prompt "Enter lenght of vector: 5" followed by input "6 7 8 9 10". 3. Output "The copy constructor is called". 4. Output "The move constructor is called". 5. Prompt "Enter lenght of vector: 5" followed by input "7 9 11 13 15". 6. Output "The move constructor is called". 7. Prompt "Enter lenght of vector: 5" followed by input "10 2 3 4 5". 8. Prompt "Enter lenght of vector: 5" followed by input "10 2 3 4 5". 9. Final message: "E:\LaboratoryWorks_IU8\x64\Debug\LaboratoryWork_2_02.exe (процесс 23452) завершил работу с кодом 0. Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки". Нажмите любую клавишу, чтобы закрыть это окно:". The console window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
Консоль отладки Microsoft V
Enter lenght of vector: 5
1 2 3 4 5
1 2 3 4 5
Enter lenght of vector: 5
6 7 8 9 10
6 7 8 9 10
The copy constructor is called
The move constructor is called
7 9 11 13 15
The move constructor is called
10 2 3 4 5
10 2 3 4 5
E:\LaboratoryWorks_IU8\x64\Debug\LaboratoryWork_2_02.exe (процесс 23452) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 1 – Запуск программы с считыванием и выводом в консоль

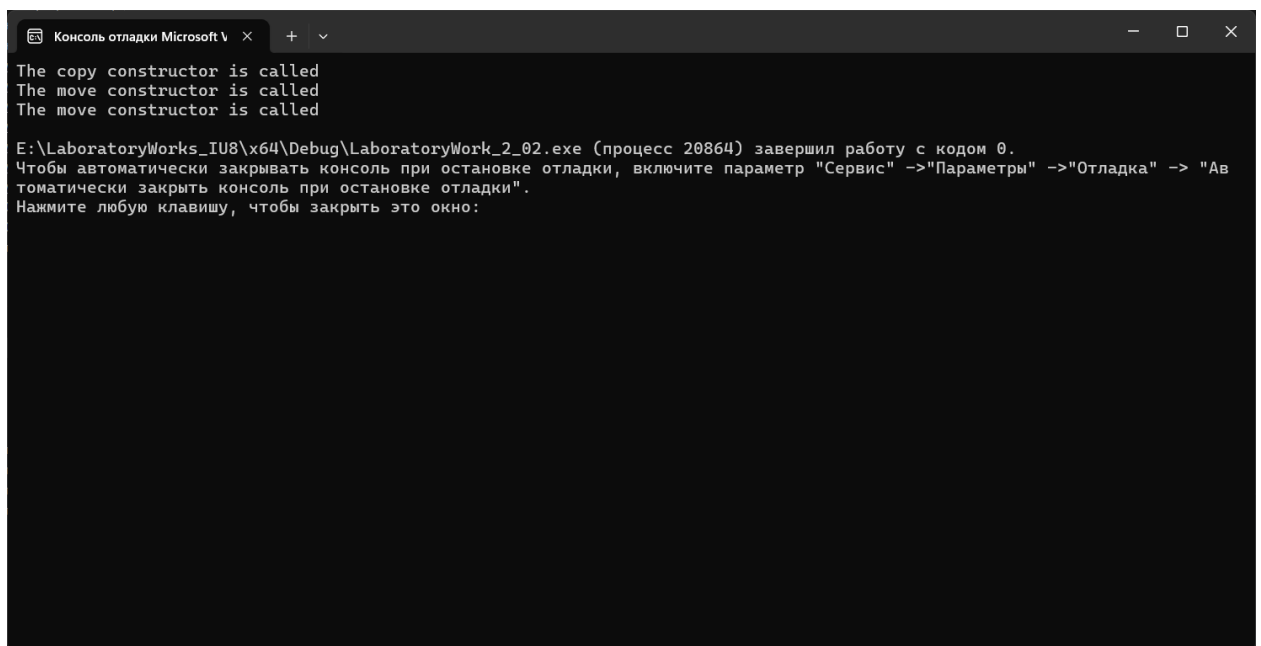
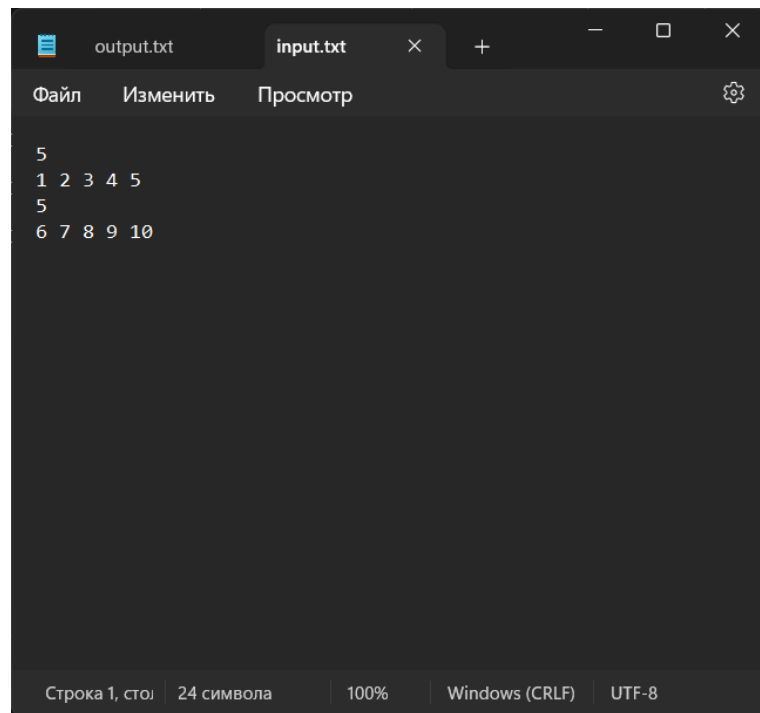
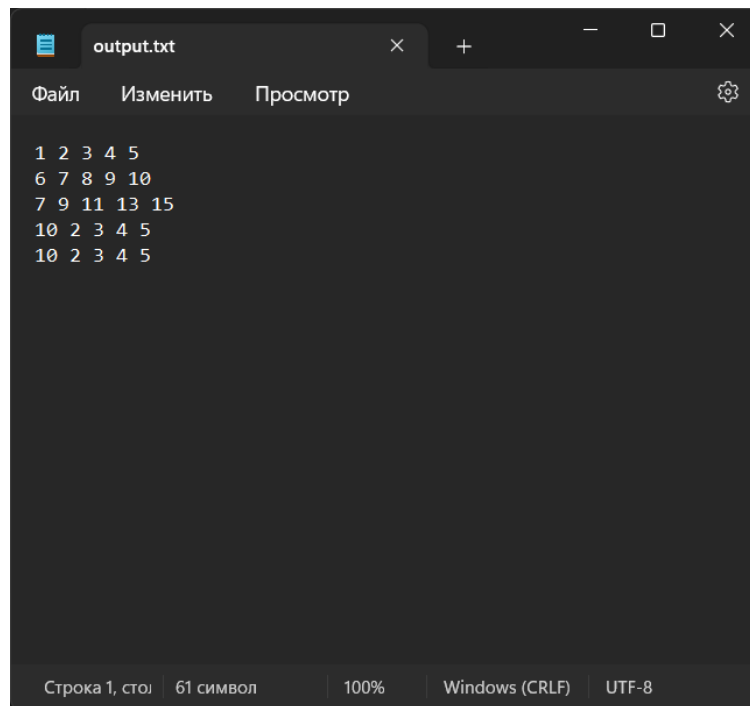


Рисунок 2-3 – Запуск программы с считыванием из файла input.txt и выводом в файл output.txt

A screenshot of a text editor window titled 'output.txt'. The window has a dark theme and a menu bar with 'Файл', 'Изменить', and 'Просмотр'. The main area contains five lines of text: '1 2 3 4 5', '6 7 8 9 10', '7 9 11 13 15', '10 2 3 4 5', and '10 2 3 4 5'. The status bar at the bottom shows 'Строка 1, столб 61 символ', '100%', 'Windows (CRLF)', and 'UTF-8'.

```
1 2 3 4 5
6 7 8 9 10
7 9 11 13 15
10 2 3 4 5
10 2 3 4 5
```

Рисунок 4 – Выходные данные

Заключение

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