I. Техническое задание:

Необходимо реализовать класс в соответствии с выбранным вариантом: матрицы.

II. Выполнить:

В классе должны быть определены различные конструкторы (по умолчанию, конструктор копирования).

Должны быть перегружены следующие операторы: +, -, ==, !=

В качестве второго операнда выступает объект этого же класса.

**Листинг описания классов:**

#pragma once

#include <iostream>

using namespace std;

class Matrix

{

private:

int rows\_count;

int columns\_count;

int \*matrix;

Matrix();

public:

Matrix(int rows\_count, int columns\_count)

{

this->rows\_count = rows\_count;

this->columns\_count = columns\_count;

matrix = new int[rows\_count \* columns\_count];

for (int i = 0; i < rows\_count \* columns\_count; i++)

{

matrix[i] = 0;

}

}

Matrix(const Matrix &matrix);

void setMatrixElement(int rows\_count, int columns\_count, int value);

int getMatrixElement(int rows\_count, int columns\_count);

void print();

Matrix operator +(const Matrix &m1);

Matrix operator -(const Matrix &m1);

bool operator==(const Matrix &m1);

bool operator!=(const Matrix &m1);

void operator =(const Matrix &m1);

~Matrix();

};

**Листинг реализации классов**

#include <iostream>

#include "OperatorOver.h"

Matrix::Matrix()

{

}

Matrix::~Matrix()

{

delete[] matrix;

}

Matrix::Matrix(const Matrix &m1)

{

this->columns\_count = m1.columns\_count;

this->rows\_count = m1.rows\_count;

matrix = new int[rows\_count \* columns\_count];

for (int i = 0; i < rows\_count \* columns\_count; i++)

{

matrix[i] = m1.matrix[i];

}

}

void Matrix::print()

{

cout << endl;

for (int i = 0; i < rows\_count \* columns\_count; i++)

{

cout << " " << matrix[i];

}

cout << endl;

}

Matrix Matrix::operator+(const Matrix &m1)

{

if (rows\_count == m1.rows\_count && columns\_count == m1.columns\_count)

{

Matrix result;

result.rows\_count = rows\_count;

result.columns\_count = columns\_count;

result.matrix = new int[rows\_count \* columns\_count];

for (int i = 0; i < rows\_count \* columns\_count; i++)

{

result.matrix[i] = matrix[i] + m1.matrix[i];

}

return result;

}

else

{

cout << endl << "ERROR: return \*this;" << endl;

return \*this;

}

}

Matrix Matrix::operator-(const Matrix &m1)

{

if (rows\_count == m1.rows\_count && columns\_count == m1.columns\_count)

{

Matrix result;

result.rows\_count = rows\_count;

result.columns\_count = columns\_count;

result.matrix = new int[rows\_count \* columns\_count];

for (int i = 0; i < rows\_count \* columns\_count; i++)

{

result.matrix[i] = matrix[i] - m1.matrix[i];

}

return result;

}

else

{

cout << endl << "ERROR: return \*this;" << endl;

return \*this;

}

}

bool Matrix::operator==(const Matrix &m1)

{

if (rows\_count != m1.rows\_count || columns\_count != m1.columns\_count)

return false;

for (int i = 0; i < (rows\_count \* columns\_count); i++)

{

if (matrix[i] != m1.matrix[i])

return false;

}

return true;

}

bool Matrix::operator!=(const Matrix &m1)

{

return !this->operator ==(m1);

}

void Matrix::operator =(const Matrix &m1)

{

delete matrix;

columns\_count = m1.columns\_count;

rows\_count = m1.rows\_count;

matrix = new int[rows\_count \* columns\_count];

for (int i = 0; i < rows\_count \* columns\_count; i++)

{

matrix[i] = m1.matrix[i];

}

}

void Matrix::setMatrixElement(int i, int j, int value)

{

matrix[(i \* (columns\_count)) + j] = value;

}

int Matrix::getMatrixElement(int i, int j)

{

return matrix[(i \* (columns\_count)) + j];

}

**Результаты работы программы проверки классов**

На данном скриншоте я проверял работу программы на перегрузку операторов.

