МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

По лабораторной работе № 5

по дисциплине

Программирование

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соколова Э.С.

(подпись)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бушуев Д.В

(подпись)

22-ИВТ-4

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2023

**Задание:** Реализовать класс Матрица. Реализовать метод для транспонирования матрицы, перегрузить операции >> и << для ввода/вывода матрицы.

**Код программы:**

**Файл Lab5.cpp**

#include <iostream>

#include "Matrix.h"

int main()

{

int column = 0, string = 0;

std::cout << "Enter the number of rows and columns" << std::endl;

std::cin >> string >> column;

matrix a(string, column);

std::cout << "Original matrix" << std::endl << a;

a.TransposeMatrix();

while (true)

{

int choice = 0;

std::cout << "Current matrix" << std::endl << a;

std::cout << "Operation's:" << std::endl

<< "1) Change Matrix" << std::endl

<< "2) Change Values of Current Matrix" << std::endl

<< "3) Transpose Matrix" << std::endl

<< "0) Close Program" << std::endl;

std::cin >> choice;

if (choice != 0)

{

switch (choice)

{

case 1:

system("cls");

std::cout << "Enter the number of rows and columns" << std::endl;

std::cin >> string >> column;

a.ChangeMatrix(string, column);

std::cin >> a;

break;

case 2:

system("cls");

a.SetMatrix();

break;

case 3:

system("cls");

a.TransposeMatrix();

break;

default:

break;

}

}

else

{

break;

}

}

return 0;

}

**Файл Matrix.h**

#pragma once

class matrix

{

private:

int\*\* mat;

unsigned int column, string;

inline int\*\* AllocateMemory();

inline int\*\* InvertMatrixMemory(const int& string, const int& column);

inline int\*\* ClearMemory(int\*\* del);

std::ostream& PrintMatrix(std::ostream& stream) const;

std::istream& SetMatrix(std::istream& stream);

public:

friend std::ostream& operator << (std::ostream& stream, const matrix& other);

friend std::istream& operator >> (std::istream& stream, matrix& other);

/\*

\* Конструктор инициализирует поля класса для работы с динамической памятью

\*/

matrix();

matrix(const int& string, const int& column);

/\*

\* Деструктор корректно очищает память при завершении работы функции

\*/

~matrix();

int\*\* ChangeMatrix(const int& string, const int& column);

int\*\* TransposeMatrix();

int\*\* SetMatrix();

void PrintMatrix() const;

};

**Файл MatrixRealisation.cpp**

#include <iostream>

#include "Matrix.h"

// Конструктор без параметров

matrix::matrix()

{

column = 0, string = 0;

mat = nullptr;

}

// Конструктор с параметрами

matrix::matrix(const int& string, const int& column)

{

this->column = column, this->string = string;

mat = AllocateMemory(); SetMatrix();

}

// Деструктор

matrix::~matrix()

{

if (mat != nullptr)

{

for (int i = 0; i < string; i++)

{

delete[] mat[i];

}

delete[] mat;

}

}

std::ostream& operator << (std::ostream& stream, const matrix& other)

{

return other.PrintMatrix(stream);

}

std::istream& operator >> (std::istream& stream, matrix& other)

{

return other.SetMatrix(stream);

}

// Функция изменения матрицы

int\*\* matrix::ChangeMatrix(const int& string, const int& column)

{

if (column == 0 || string == 0) // при неверных данных, ничего не изменять

{

return mat;

}

else if (mat != nullptr) // если матрица существует

{

mat = ClearMemory(mat);

this->column = column, this->string = string;

return mat = AllocateMemory();

//return mat = SetMatrix();

}

else

{

this->column = column, this->string = string;

return mat = AllocateMemory();

//return mat = SetMatrix();

}

}

inline int\*\* matrix::AllocateMemory()

{

if (column == 0 || string == 0)

{

return nullptr;

}

else

{

try

{

mat = new int\* [string];

}

catch (std::bad\_alloc& ba)

{

std::cout << "Error: " << ba.what() << std::endl;

}

for (int i = 0; i < string; i++)

{

try

{

mat[i] = new int[column];

}

catch (std::bad\_alloc& ba)

{

std::cout << "Error: " << ba.what() << std::endl;

}

}

return mat;

}

}

// Выделение памяти под инвентированную матрицу

inline int\*\* matrix::InvertMatrixMemory(const int& string,const int& column)

{

int\*\* p = nullptr;

if (column == 0 || string == 0)

{

return p;

}

else

{

try

{

p = new int\* [column];

}

catch (std::bad\_alloc& ba)

{

std::cout << "Error: " << ba.what() << std::endl;

return p = nullptr;

}

for (int i = 0; i < column; i++)

{

try

{

p[i] = new int[string];

}

catch (std::bad\_alloc& ba)

{

std::cout << "Error: " << ba.what() << std::endl;

return p = nullptr;

}

}

return p;

}

}

inline int\*\* matrix::ClearMemory(int\*\* del)

{

if (del != nullptr)

{

for (int i = 0; i < string; i++)

{

delete[] del[i];

}

delete[] del;

}

return del = nullptr;

}

// Транспонирование матрицы

int\*\* matrix::TransposeMatrix()

{

if(mat != nullptr)

{

int\*\* p = InvertMatrixMemory(string, column), temp;

for (int i = 0; i < string; i++)

{

for (int j = 0; j < column; j++)

{

p[j][i] = mat[i][j];

}

}

mat = ClearMemory(mat);

mat = p;

temp = column;

column = string;

string = temp;

std::cout << "Transpose completed!" << std::endl;

return mat;

}

else

{

std::cout << "Cannot transpose matrix as it does not exist" << std::endl;

return nullptr;

}

}

int\*\* matrix::SetMatrix()

{

system("cls");

for (int i = 0; i < string; i++)

{

for (int j = 0; j < column; j++)

{

std::cout << "You are editing matrix type "

<< string << 'x' << column << std::endl;

std::cout << "Current element is " << i + 1 << j + 1 << std::endl;

std::cin >> mat[i][j];

system("cls");

}

}

return mat;

}

void matrix::PrintMatrix() const

{

if (mat == nullptr)

{

std::cout << "There is no matrix available to display here" << std::endl;

}

else

{

for (int i = 0; i < string; i++)

{

for (int j = 0; j < column; j++)

{

std::cout << mat[i][j] << ' ';

}

std::cout << std::endl;

}

}

}

std::ostream& matrix::PrintMatrix(std::ostream& stream) const

{

if (mat == nullptr)

{

return stream << "There is no matrix available to display here";

}

else

{

for (int i = 0; i < string; i++)

{

for (int j = 0; j < column; j++)

{

stream << mat[i][j] << ' ';

}

stream << std::endl;

}

return stream;

}

}

std::istream& matrix::SetMatrix(std::istream& stream)

{

system("cls");

for (int i = 0; i < string; i++)

{

for (int j = 0; j < column; j++)

{

std::cout << "You are editing matrix type "

<< string << 'x' << column << std::endl;

std::cout << "Current element is " << i + 1 << j + 1 << std::endl;

stream >> mat[i][j];

system("cls");

}

}

return stream;

}

**Файл MatrixTest.cpp**

#include "pch.h"

#include "CppUnitTest.h"

#include "..\Lab5\Matrix.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace MatrixTest

{

TEST\_CLASS(MatrixTest)

{

public:

TEST\_METHOD(No\_Matrix\_transpose)

{

matrix a;

int\*\* p = a.TransposeMatrix();

Assert::IsNull(p);

}

TEST\_METHOD(No\_Matrix\_wrong\_change)

{

matrix a;

int\*\* p = a.ChangeMatrix(0,0);

Assert::IsNull(p);

}

TEST\_METHOD(No\_Matrix\_Colums)

{

matrix a(0,1);

int\*\* p = a.TransposeMatrix();

Assert::IsNull(p);

}

TEST\_METHOD(No\_Matrix\_Colums\_Set)

{

matrix a(0, 1);

int\*\* p = a.SetMatrix();

Assert::IsNull(p);

}

};

}

**Блок схема:**

Выделение памяти в стеке под класс

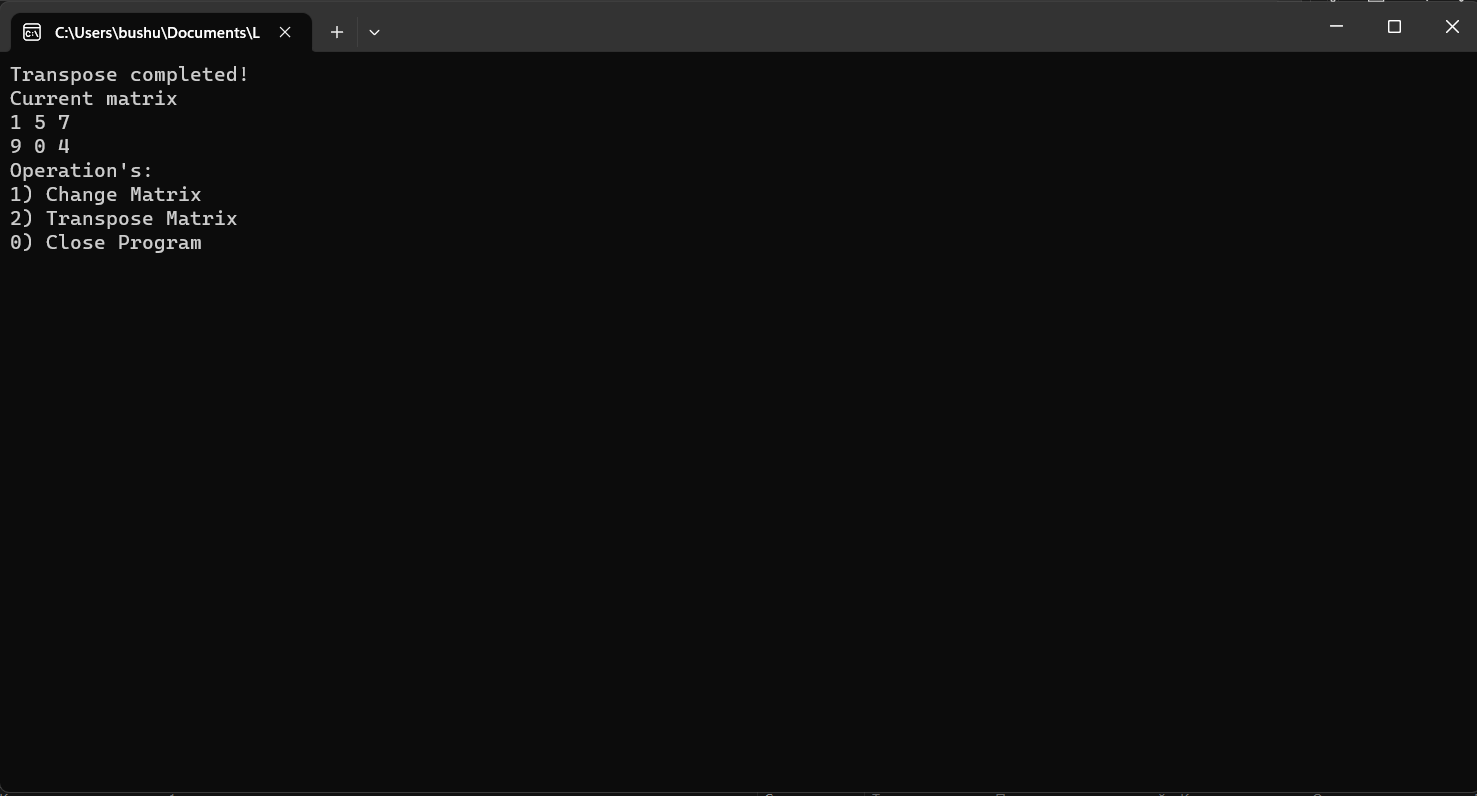
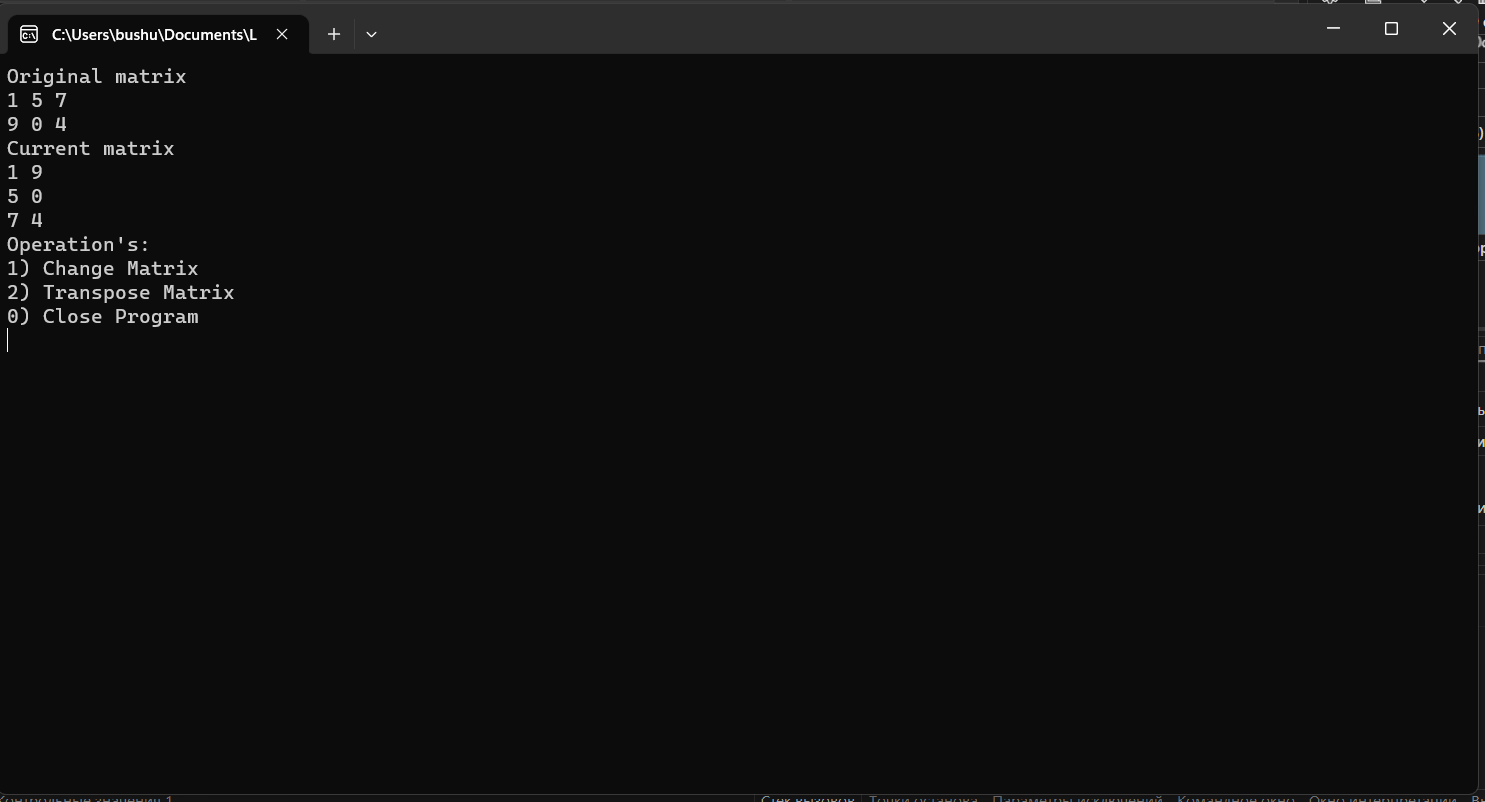
Транспонирование матрицы

Изменить матрицу

Транспонирование матрицы

Конструктор

**Результаты работы программы:**



**Вывод:** В результате данной лабораторной работы, я научился работать с классами и перегружать операторы