

Национальный исследовательский университет ИТМО

Кафедра программных систем

**Лабораторная работа 1**

Выполнил: Орел

Даниил Максимович

Группа № K3221

Проверил: Иванов С.Е.

Санкт-Петербург

2020

**Цель работы:**

* Для квадратной матрицы NxN, размер которой задается, с помощью массивов реализовать методы сложения, перемножения, нахождения обратной матрицы.

**Ход работы:**

1. **В ходе выполнения задания был создан статичный класс MatrixActions, содержащий в себе следующие функции:**
   1. Общие функции программы, упрощающие работу с ней:

//Функция, генерирующая матрицу целых чисел в интервале от (-10; 10)

static public int[,] GenerateMatrix(int rows, int columns)

{

Random rnd = new Random();

int value;

int[,] matrix = new int[rows,columns];

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

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

{

value = rnd.Next(-10, 10);

matrix[i, j] = value;

}

PrintMatrix(matrix);

return matrix;

}

//Функция вывода матрицы на экран

static public void PrintMatrix(int[,] matrix)

{

int rows = matrix.GetUpperBound(0) + 1;

int columns = matrix.Length / rows;

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

{

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

{

Console.Write(matrix[i, j] + "\t");

}

Console.WriteLine();

}

}

//Функция вывода матрицы дробных чисел на экран

static public void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetUpperBound(0) + 1;

int columns = matrix.Length / rows;

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

{

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

{

Console.Write(Math.Round(matrix[i, j], 2) + "\t");

}

Console.WriteLine();

}

}

* 1. Сложение матриц:

static public void MatrixAddition(int[,] matrix1, int[,] matrix2)

{

int[,] matrix;

int rows\_matrix1 = matrix1.GetUpperBound(0) + 1;

int columns\_matrix1 = matrix1.Length / rows\_matrix1;

int rows\_matrix2 = matrix2.GetUpperBound(0) + 1;

int columns\_matrix2 = matrix2.Length / rows\_matrix2;

if (isMatricesValidForAddition(rows\_matrix1, columns\_matrix1, rows\_matrix2, columns\_matrix2))

{

matrix = new int[rows\_matrix1, columns\_matrix1];

for (int i = 0; i < rows\_matrix1; i++)

for (int j = 0; j < columns\_matrix1; j++)

{

matrix[i, j] = matrix1[i, j] + matrix2[i, j];

}

Console.WriteLine("Результат сложения: ");

PrintMatrix(matrix);

}

else Console.WriteLine("Невозможно выполнить сложение, т.к. количество строк и столбцов первой матрицы не соответсвует количеству строк и столбцов второй матрицы");

}

//Функция, проверяющая совпадают ли размерности складываемых матриц

static private bool isMatricesValidForAddition(int rows1, int columns1, int rows2, int columns2)

{

if (rows1 == rows2 && columns1 == columns2)

return true;

else

return false;

}

* 1. Перемножение матриц:

static public void MatrixMultiply(int[,] matrix1, int[,] matrix2)

{

int[,] matrix;

int rows\_matrix1 = matrix1.GetUpperBound(0) + 1;

int columns\_matrix1 = matrix1.Length / rows\_matrix1;

int rows\_matrix2 = matrix2.GetUpperBound(0) + 1;

int columns\_matrix2 = matrix2.Length / rows\_matrix2;

if (isMatricesValidForMultiply(columns\_matrix1, rows\_matrix2))

{

matrix = new int[rows\_matrix1, columns\_matrix2];

for (var i = 0; i < rows\_matrix1; i++)

{

for (var j = 0; j < columns\_matrix2; j++)

{

matrix[i, j] = 0;

for (var k = 0; k < columns\_matrix1; k++)

{

matrix[i, j] += matrix1[i, k] \* matrix2[k, j];

}

}

}

Console.WriteLine("Результат умножения: ");

PrintMatrix(matrix);

}

else Console.WriteLine("Умножение не возможно! Количество столбцов первой матрицы не равно количеству строк второй матрицы.");

}

//Проверка совпдает ли количество столбцов первой матрицы с количеством строк второй матрицы:

static private bool isMatricesValidForMultiply(int columns1, int rows2)

{

if (rows2 == columns1)

return true;

else

return false;

}

* 1. Нахождение обратной матрицы:

static public void CalculateInverseMatrix(int[,] matrix)

{

int k = 1;

double[,] algebraic\_complement\_matrix;

int rows = matrix.GetUpperBound(0) + 1;

int columns = matrix.GetUpperBound(1) + 1;

int[,] minor;

if (isSquare(rows, columns))

{

int determinant = CalculateDeterminant(matrix, rows);

Console.WriteLine("Определитель = " + determinant);

if (determinant != 0)

{

minor = new int[rows - 1, rows - 1];

matrix = FindTransparentMatrix(matrix);

algebraic\_complement\_matrix = new double[rows, rows];

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

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

{

algebraic\_complement\_matrix[i, j] = k \* CalculateDeterminant(GetMatr(matrix, minor, i, j, rows), rows - 1);

k = -k;

}

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

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

{

algebraic\_complement\_matrix[i, j] /= (double)determinant;

}

Console.WriteLine("Обратная матрица: ");

PrintMatrix(algebraic\_complement\_matrix);

}

else Console.WriteLine("Матрица вырожденная и не имеет обратной матрицы");

}

else Console.WriteLine("Матрица не квадратная! Только квадратная матрица имеет обратную.");

}

//Проверка: является ли данная матрица квадратной?

static private bool isSquare(int rows, int columns)

{

if (rows == columns)

return true;

else return false;

}

//Вычисление определителя

static public int CalculateDeterminant(int[,] matrix, int rows)

{

int det = 0;

int k = 1;

int[,] new\_matrix = new int[rows, rows];

if (rows == 1)

return matrix[0, 0];

else if (rows == 2)

{

det = matrix[0, 0] \* matrix[1, 1] - matrix[1, 0] \* matrix[0, 1];

return det;

}

else

{

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

{

det += k \* matrix[i,0] \* CalculateDeterminant(GetMatr(matrix, new\_matrix, i, 0, rows), rows - 1);

k = -k;

}

return det;

}

}

//Получение матрицы путем вычеркивания i-ой строки и j-ого столбца

static public int[,] GetMatr(int[,] matrix, int[,] p, int i, int j, int m)

{

int ki, kj, di, dj;

di = 0;

for (ki = 0; ki < m - 1; ki++) { // проверка индекса строки

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < m - 1; kj++) { // проверка индекса столбца

if (kj == j) dj = 1;

p[ki,kj] = matrix[ki + di, kj + dj];

}

}

return p;

}

//Получение транспонированной матрицы

static public int[,] FindTransparentMatrix(int[,] matrix)

{

int rows = matrix.GetUpperBound(0) + 1;

int columns = matrix.Length / rows;

int temp;

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

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

{

if(i > j)

{

temp = matrix[i, j];

matrix[i, j] = matrix[j, i];

matrix[j, i] = temp;

}

}

Console.WriteLine("Транспонированная матрица: ");

PrintMatrix(matrix);

return matrix;

}

1. **Код класса Program:**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Матрица A: ");

Console.Write("Введите количество строк в матрице: ");

int rows\_array1 = int.Parse(Console.ReadLine());

Console.Write("Введите количество столбцов в матрице: ");

int columns\_array1 = int.Parse(Console.ReadLine());

int[,] array1 = MatrixActions.GenerateMatrix(rows\_array1, columns\_array1);

Console.WriteLine("-------------------------------------------------------");

Console.WriteLine("Матрица B: ");

Console.Write("Введите количество строк в матрице: ");

int rows\_array2 = int.Parse(Console.ReadLine());

Console.Write("Введите количество столбцов в матрице: ");

int columns\_array2 = int.Parse(Console.ReadLine());

int[,] array2 = MatrixActions.GenerateMatrix(rows\_array2, columns\_array2);

Console.WriteLine("-------------------------------------------------------");

Console.WriteLine("A + B : ");

MatrixActions.MatrixAddition(array1, array2);

Console.WriteLine("-------------------------------------------------------");

Console.WriteLine("A \* B : ");

MatrixActions.MatrixMultiply(array1, array2);

Console.WriteLine("-------------------------------------------------------");

Console.WriteLine("Нахождение матрицы обратной данной: ");

MatrixActions.PrintMatrix(array1);

MatrixActions.CalculateInverseMatrix(array1);

}

}

1. **Результат работы программы:**

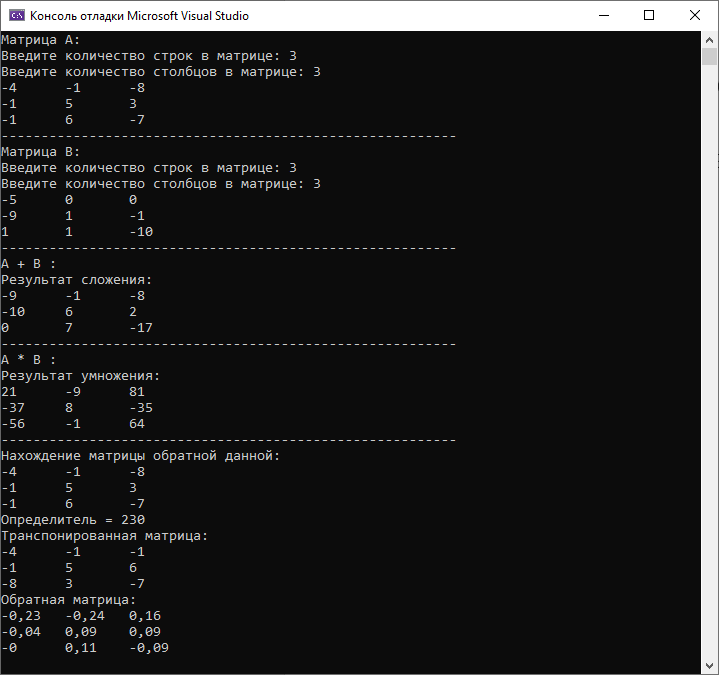


Рисунок 1 – Создание и обработка двух матриц 3x3

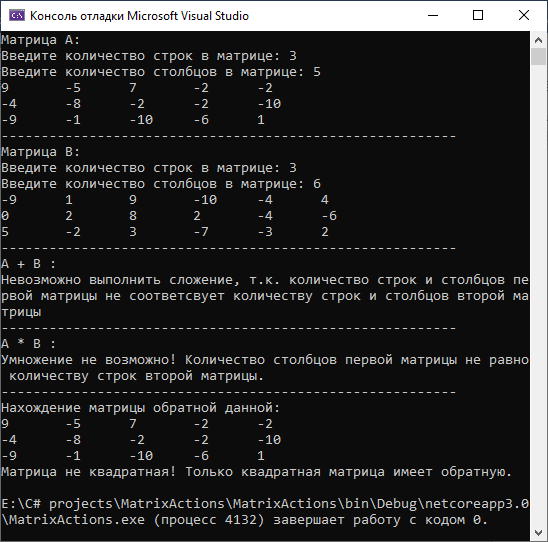


Рисунок 2 – Создание и обработка матриц 3x5 и 3x6

**Вывод:**

В ходе выполнения данной практической работы была создана программа, выполняющая основные действия с матрицами: сложение и перемножение матриц, а также нахождение обратной матрицы.