Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управление»

Курс «Основы программирования»

Отчет по лабораторной работе №8
«Вычисление обратной матрицы методом Гаусса-Жордана»

Выполнил:
Студент группы ИУ5-11Б
Алехин Сергей
Подпись и дата:

Проверил:
Преподаватель каф. ИУ5
Правдина Анна Дмитриевна
Подпись и дата:

Москва, 2018 г.

Задание

Разработка функции, реализующей алгоритм вычисления обратной матрицы методом Гаусса-Жордана.

Разработка функции умножения матриц.

Использование разработанных функций для решения систем линейных уравнений.

Планируемое время выполнения работы - 8 часов.

Разработка алгоритма

Входные переменные:

1) int iMatrixSize – размер матрицы;
2) int iNumBeforeDot – количество знаков до запятой;
3) int iNumAfterDot – количество знаков после запятой;
4) double **dMatrix – матрица;
5) double **dInitialMatrix – начальная матрица нужна для перемножения;
6) double **dInverseMatrix – обратная матрица;

Функции:

1) void print – вывод матрицы на экран с возможностью выбора типа вывода;
    a. *Входные переменные:*
        i. double **dMatrix – матрица;
        ii. int iOutputOption – тип вывода на экран (fixed/scientific);
        iii. int iNumBeforeDot – количество знаков до запятой;
        iv. int iNumAfterDot – количество знаков после запятой;
        v. int iMatrixSize – размер матрицы.
    b. Локальные переменные:
        i. int iColumn – номер текущего столбца;
        ii. int iNumLength – длина слова (нужно для setw);
        iii. int iNumColumnsInRow – количество слолбцов в строке;
2) double** InverseMartixCount – нахождение обратной матрицы;
    a. *Входные переменные:*
        i. double **dMatrix – передаваемая матрица;
        ii. int n – размер матрицы.
    b. *Локальные переменные:*
        i. double **dInverseMatrix – обратная матрица;
        ii. double A – значение матрицы, нужно для получение 0 в матрице;
        iii. int m – номер строки с ненулевым элементом.
    c. *Возвращаемое значение:*
        i. double **dInverseMatrix – обратная матрица.
3) double** MatrixMult – произведение матриц.
    a. *Входные переменные:*
        i. double **dInitialMatrix – начальная матрица;
        ii. double **dInverseMatrix – обратная матрица;
        iii. int n – размер матрицы.
    b. *Локальные переменные:*
        i. double **dMatrix – произведение матриц.
    c. *Возвращаемое значене:*
        i. double **dMatrix – произведение матриц.

Текст программы

Файл Lab7.cpp

```
#include <iostream>
#include <iomanip>
```

```
#include <algorithm>
using namespace std;
int main()
{
	setlocale(0, "RUSSIAN");
	double **dMatrix, **dInitialMatrix, **dInverseMatrix;
	int iMatrixSize, iNumAfterDot, iNumBeforeDot;
	cout << "Введите размер матрицы = ";
	cin >> iMatrixSize;
	dMatrix = new double*[iMatrixSize];
	for (int i = 0; i < iMatrixSize; i++)
		dMatrix[i] = new double[iMatrixSize];
	dInitialMatrix = new double*[iMatrixSize];
	for (int i = 0; i < iMatrixSize; i++)
		dInitialMatrix[i] = new double[iMatrixSize];
	cout << "Введите матрицу A" << endl;
	for (int i = 0; i < iMatrixSize; i++)
		for (int j = 0; j < iMatrixSize; j++)
		{
			cin >> dMatrix[i][j];
			dInitialMatrix[i][j] = dMatrix[i][j];
		}
	cout << "Количество знаков до запятой = ";
	cin >> iNumBeforeDot;
	cout << "Количество знаков после запятой = ";
	cin >> iNumAfterDot;
	cout << "Матрица A" << endl;
	print(dMatrix, 1, iNumBeforeDot, iNumAfterDot, iMatrixSize);
	dInverseMatrix = InverseMatrixCount(dMatrix, iMatrixSize);
	if (dInverseMatrix == 0) return 0;
	for (int i = 0; i < iMatrixSize; i++)
		delete[] dMatrix[i];
	delete []dMatrix;
	cout << "Количество знаков до запятой = ";
	cin >> iNumBeforeDot;
	cout << "Количество знаков после запятой = ";
	cin >> iNumAfterDot;
	cout << "Обратная матрица A" << endl;
	print(dInverseMatrix, 1, iNumBeforeDot, iNumAfterDot, iMatrixSize);
	dMatrix = MatrixMult(dInitialMatrix, dInverseMatrix, iMatrixSize);
	cout << "Произведение A * A^-1" << endl;
	print(dMatrix, 1, 0, 0, iMatrixSize);
	for (int i = 0; i < iMatrixSize; i++)
		delete[] dMatrix[i];
	delete[]dMatrix;
	for (int i = 0; i < iMatrixSize; i++)
		delete[] dInitialMatrix[i];
	delete[]dInitialMatrix;
	for (int i = 0; i < iMatrixSize; i++)
		delete[] dInverseMatrix[i];
	delete[]dInverseMatrix;
	return 0;

}
```

Файл str.h

```
#pragma once
void print(double **, int, int, int, int);
```

```
double** InverseMatrixCount(double **, int);
double** MatrixMult(double **, double **, int);
```

Файл func.cpp

```
#include <iostream>
#include <iomanip>
#include <algorithm>
using namespace std;
double** MatrixMult(double **dInitialMatrix, double **dInverseMatrix, int n)
{
	double **dMatrix = new double*[n];
	for (int i = 0; i < n; i++)
		dMatrix[i] = new double[n];
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			dMatrix[i][j] = 0;
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			for (int k = 0; k < n; k++)
				dMatrix[i][j] += dInitialMatrix[i][k] * dInverseMatrix[k][j];
	return dMatrix;
}
void print(double **dMatrix, int iOutputOption, int iNumBeforeDot, int iNumAfterDot, int
iMatrixSize)
{
	int iColumn = 0, iNumLength, iNumColumnsInRow;
	if (iOutputOption == 1) iNumLength = iNumBeforeDot + iNumAfterDot + 2;
	else iNumLength = iNumAfterDot + 7;
	iNumColumnsInRow = floor(79 / (iNumLength));
	if (iNumColumnsInRow > iMatrixSize) iNumColumnsInRow = iMatrixSize;
	while (iColumn < iMatrixSize)
	{
		for (int i = 0; i < iMatrixSize; i++)
		{
			for (int j = iColumn; j < min(iMatrixSize, iColumn + iNumColumnsInRow);
j++)
			{
				if (iOutputOption == 1) cout << fixed << setw(iNumLength) <<
setprecision(iNumAfterDot) << dMatrix[i][j] << ' ';
				else cout << scientific << setw(iNumLength) <<
setprecision(iNumAfterDot) << dMatrix[i][j] << ' ';
			}
			cout << endl;
		}
		iColumn += iNumColumnsInRow;
		cout << endl;
	}
}
double** InverseMatrixCount(double **dMatrix, int n)
{
	double **dInverseMatrix = new double*[n];
	for (int i = 0; i < n; i++)
		dInverseMatrix[i] = new double[n];
	double A;
	int m;
	for (int i = 0; i < n; i++)
```

```
	for (int j = 0; j < n; j++)
		if (i == j) dInverseMatrix[i][j] = 1;
		else dInverseMatrix[i][j] = 0;
for (int i = 0; i < n; i++)
{

	if (dMatrix[i][i] != 0)	A = dMatrix[i][i];
	else
	{
		m = -1;
		for (int j = i; j < n; j++)
		{
			if (dMatrix[j][i] != 0)
			{
				m = j;
				break;
			}
		}
		if (m == -1)
		{
			cout << "Обратная матрица не существует";
			return 0;
		}
		for (int j = 0; j < n; j++)
		{
			swap(dMatrix[i][j], dMatrix[m][j]);
			swap(dInverseMatrix[i][j], dInverseMatrix[m][j]);
		}

		A = dMatrix[i][i];
	}
	for (int j = 0; j < n; j++)
	{
		dMatrix[i][j] /= A;
		dInverseMatrix[i][j] /= A;
	}
	for (int j = i; j < n - 1; j++)
	{
		A = -dMatrix[j + 1][i];
		for (int k = 0; k < n; k++)
		{
			dMatrix[j + 1][k] += dMatrix[i][k] * A;
			dInverseMatrix[j + 1][k] += dInverseMatrix[i][k] * A;
		}
	}

}
for (int i = n - 1; i >= 0; i--)
{
	for (int j = i; j >= 1; j--)
	{
		A = dMatrix[j - 1][i];
		for (int k = 0; k < n; k++)
		{
			dMatrix[j - 1][k] += dMatrix[i][k] * (-A);
			dInverseMatrix[j - 1][k] += dInverseMatrix[i][k] * (-A);
		}
```

```
            }
        }
        return dInverseMatrix;
}
```

Анализ результатов

№	Входные данные	Полученный результат
1	Обычная матрица	Количество знаков до запятой = 1 Количество знаков после запятой = 1 Матрица A 1.0 2.0 3.0 4.0 4.0 2.0 5.0 6.0 1.0 3.0 6.0 3.0 5.0 2.0 4.0 5.0 Количество знаков до запятой = 2 Количество знаков после запятой = 2 Обратная матрица A -0.20 -0.20 0.00 0.40 0.76 -1.24 0.11 0.82 -0.49 0.51 0.22 -0.36 0.29 0.29 -0.22 -0.24 Произведение A * A^-1 1 -0 0 0 -0 1 0 0 0 -0 1 -0 -0 -0 0 1
2	Первый элемент матрицы равен 0	Количество знаков до запятой = 1 Количество знаков после запятой = 1 Матрица A 0.0 5.0 6.0 2.0 7.0 2.0 8.0 9.0 0.0 3.0 4.0 7.0 2.0 4.0 5.0 0.0 Количество знаков до запятой = 2 Количество знаков после запятой = 2 Обратная матрица A -0.63 -0.04 0.23 0.64 -0.90 -0.30 0.64 1.04 0.97 0.25 -0.60 -0.89 -0.17 -0.02 0.21 0.06 Произведение A * A^-1 1 0 -0 0 0 1 -0 0 -0 -0 1 0 -0 0 0 1

3	Обратной матрицы не существует	Количество знаков до запятой = 1 Количество знаков после запятой = 1 Матрица A 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0 4.0 1.0 0.0 0.0 0.0 0.0 Обратная матрица не существует

В этой работе мы научились находить обратную матрицу методом Гаусса-Жордана, научились работать с двумерными динамическими массивами, научились перемножать матрицы.