Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Основы информатики»

Отчет по лабораторной работе № 10

«Вычисление обратной матрицы методом Гаусса-Жордана»

Выполнил:
студент группы ИУ5-13
Терентьев Владислав
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Козлов А. Д.
Подпись и дата:

Москва, 2019 г.

1. Постановка задачи

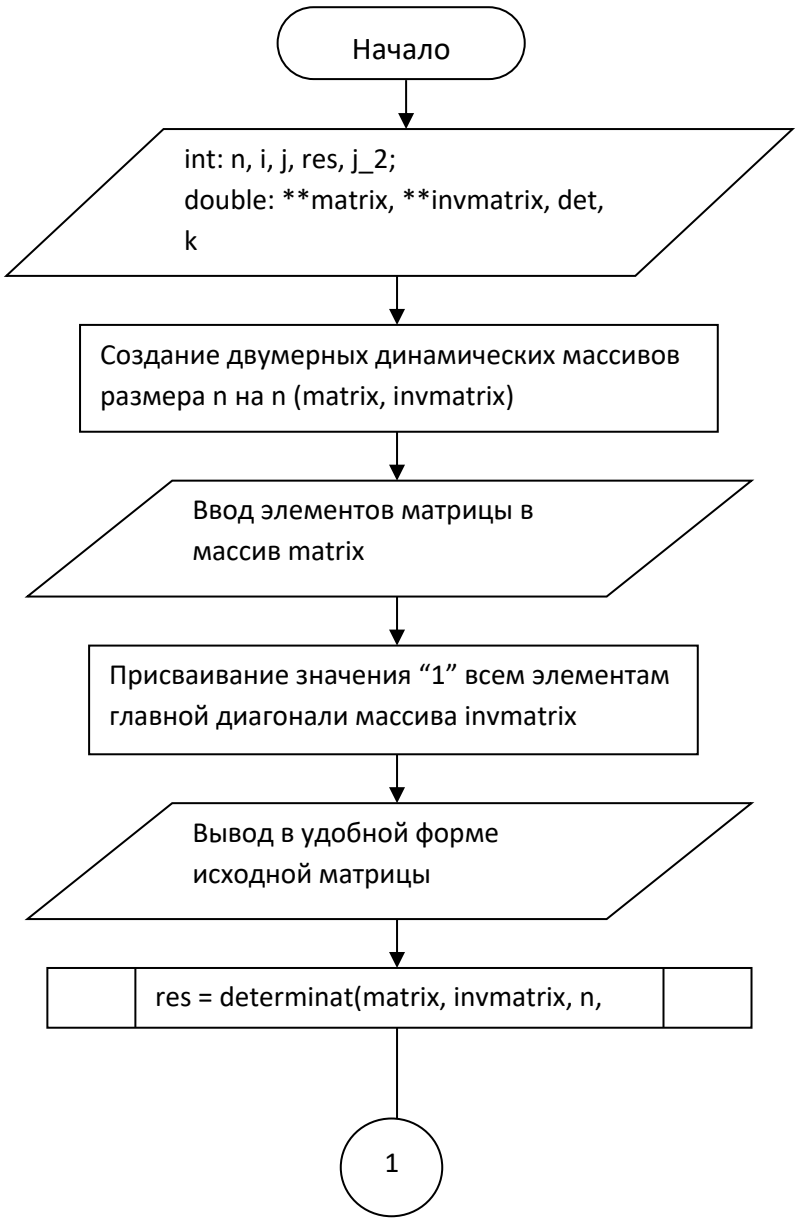
Написать программу для вычисления и вывода обратной матрицы к введенной квадратной матрице методом Гаусса-Жордана.

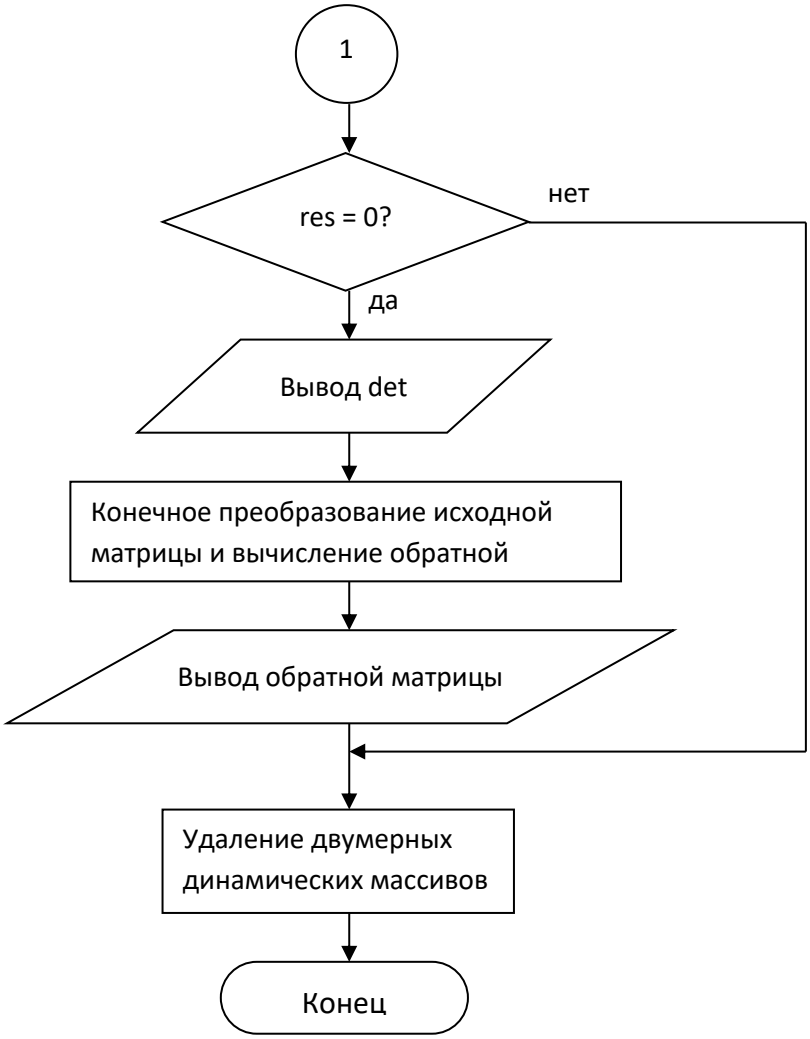
2. Разработка алгоритма

Описание переменных: переменные типа int: i, j, j_2 – для цикла; n – количество столбцов (строк) квадратной матрицы; res – для возвращаемого значения функции determinat.
типа double: det – значение определителя; k – элемент матрицы.
типа double **: matrix – для динамического двумерного массива со значениями исходной матрицы; invmatrix - для динамического двумерного массива со значениями обратной матрицы.

Описание функций: функции типа int: determinat – 4 входных параметра [двумерный динамический массив (исходная матрица), двумерный динамический массив (обратная матрица), число(количество столбцов/строк квадратной матрицы), адрес переменной для значения определителя], вычисляет определитель и возвращает значение -1, если определитель равен 0, или возвращает значение 0, если определитель не равен 0.

Схема алгоритма:





3. Текст программы

```
#include "pch.h"
#include <iostream>

using namespace std;

int determinat(double **, double **, int, double &);

int main()
{
	int n, i, j, res, j_2;
	double **matrix, **invmatrix, det, k;

	setlocale(LC_ALL, "Russian");
	cout << "Введите количество строк (столбцов) квадратной матрицы: ";
	cin >> n;
	matrix = new double *[n];
	invmatrix = new double *[n];
	for (i = 0; i < n; i++) {
		matrix[i] = new double[n];
		invmatrix[i] = new double[n];
	}
	cout << endl << "Введите элементы матрицы:" << endl;
	for (i = 0; i < n; i++) {
		for (j = 0; j < n; j++) {
			cin >> matrix[i][j];
			if (j == i) {
				invmatrix[j][j] = 1;
			}
			else {
```

```
                        invmatrix[i][j] = 0;
                    }
                }
        }
        cout << endl << "Исходная матрица: " << endl;
        for (i = 0; i < n; i++) {
                for (j = 0; j < n; j++) {
                        cout << matrix[i][j] << "\t \t";
                }
                cout << endl;
        }
        res = determinat(matrix, invmatrix, n, det);
        if (res == 0) {
                cout << endl << "Определитель: " << det << endl << endl;
                for (j = 0; j < n; j++) {
                        k = matrix[j][j];
                        for (j_2 = 0; j_2 < n; j_2++) {
                                if (j_2 >= j) {
                                        matrix[j][j_2] = matrix[j][j_2] / k;
                                }
                                invmatrix[j][j_2] = invmatrix[j][j_2] / k;
                        }
                }
                for (j = n - 1; j > 0; j--) {
                        for (i = j - 1; i >= 0; i--) {
                                for (j_2 = 0; j_2 < n; j_2++) {
                                        invmatrix[i][j_2] = invmatrix[i][j_2] - matrix[i][j] *
invmatrix[j][j_2];
                                }
                        }
                }
                cout << "Обратная матрица: " << endl;
                for (i = 0; i < n; i++) {
                        for (j = 0; j < n; j++) {
                                cout << invmatrix[i][j] << "\t \t";
                        }
                        cout << endl;
                }
        }
        cout << endl;
        for (i = 0; i < n; i++) {
                delete[] matrix[i];
                delete[] invmatrix[i];
        }
        delete[]matrix;
        delete[]invmatrix;
        system("pause");
}

int determinat(double **matrix, double **invmatrix, int n, double &det) {
        int i, j, i_0, j_2;
        double k, buf;
        det = 1;
        for (j = 0; j < n; j++) {
                for (i = j; (i < n) && (fabs(matrix[i][j]) < 1.0e-15); i++) {}
                i_0 = i;
                if (i_0 == n) {
                        cout << endl << "Определитель равен: 0" << endl;
                        return -1;
                }
                else {
                        for (j_2 = 0; j_2 < n; j_2++) {
                                buf = matrix[i_0][j_2];
                                matrix[i_0][j_2] = (-1)*matrix[j][j_2];
                                matrix[j][j_2] = buf;
```

```
                        buf = invmatrix[i_0][j_2];
                        invmatrix[i_0][j_2] = (-1)*invmatrix[j][j_2];
                        invmatrix[j][j_2] = buf;
                }
                det = det * matrix[j][j];
                for (i = j + 1; i < n; i++) {
                        if (fabs(matrix[i][j]) >= 1.0e-15) {
                                k = -matrix[i][j] / matrix[j][j];
                                for (j_2 = 0; j_2 < n; j_2++) {
                                        matrix[i][j_2] = matrix[i][j_2] + k *
matrix[j][j_2];

                                        invmatrix[i][j_2] = invmatrix[i][j_2] + k *
invmatrix[j][j_2];
                                }
                        }
                }
        }
        return 0;
}
```

4. Анализ результатов

```
Введите количество строк (столбцов) квадратной матрицы: 3

Введите элементы матрицы:
2 5 7 6 3 4 5 -2 -3

Исходная матрица:
2               5               7
6               3               4
5               -2              -3

Определитель: -1

Обратная матрица:
1               -1              1
-38             41              -34
27              -29             24

Press any key to continue . . .
```

```
Введите количество строк (столбцов) квадратной матрицы: 4

Введите элементы матрицы:
2 1 0 0 3 2 0 0 1 1 3 4 2 -1 2 3

Исходная матрица:
2               1               0               0
3               2               0               0
1               1               3               4
2               -1              2               3

Определитель: 1

Обратная матрица:
2               -1              0               0
-3              2               0               0
31              -19             3               -4
-23             14              -2              3

Press any key to continue . . .
```

```
Введите количество строк (столбцов) квадратной матрицы: 3

Введите элементы матрицы:
1 2 3 4 5 6 7 8 9

Исходная матрица:
1               2               3
4               5               6
7               8               9

Определитель равен: 0

Press any key to continue . . .
```