БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №1

**Метод Гаусса решения СЛАУ**

Выполнила:

Мастыкина Елизавета

2 курс 6 группа

Преподаватель:

Горбачева Ю.Н.

Минск, 2021

**Содержание**

Постановка задачи………………………………………………………....5

Алгоритм решения………………………………………………………....6

Листинг программы………………………………………………………..8

Вывод программы………………………………………………………….17

Выводы……………………………………………………………………..19

**Постановка задачи**

Написать и отладить программу, реализующую метод Гаусса с выбором главного элемента по столбцу для численного решения систем линейных алгебраических уравнений (СЛАУ); Найти обратную к ней матрицу и сделать проверку; вычислить вектор невязки.. Предусмотреть сообщения, предупреждающие о невозможности решения указанных задач с заданной матрицей A порядка n.

**Алгоритм решения**

Первоначально имеем уравнение вида Ax=f;

Запишем его в развернутом виде:

a11x1+a12x2+a13x3+a14x4=f1

a21x1+a22x2+a23x3+a24x4=f2

a31x1+a32x2+a33x3+a34x4=f3

a41x1+a42x2+a43x3+a44x4=f4

Решение этого уравнения найдем по схеме единственного деления.

1. Выбираем строку с максимальным элементо в первом столбце, перестановкой строк меняем выбранную строку с первой. Выбираем ведущий элемент а11(без ограничения общности можно считать его ненулевым);

2. Поделив первое уравнение на а11 , получаем

x1+a(1)12x2+a(1)13x3+a(1)14x4=g1

где  a(1)1j=a(0)1j/a(0)11 ,j=2,…,5;

3. Исключаем переменную x1 из всех последующих уравнений, начиная со второго, путём вычитания уравнения 2, умноженного на коэффициент, стоящий при x1 в соответствующем уравнении. Получаем

a(1)11x1+a(1)12x2+a(1)13x3+a(1)14x4=f(1)1

a(1)22x2+a(1)23x3+a(1)24x4=f(1)2

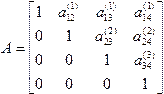
a(1)32x2+a(1)33x3+a(1)34x4=f(1)3

a(1)42x2+a(1)43x3+a(1)44x4=f(1)4

Где а(1)ij=a(0)ij-a(0)1j\*a(0)i1 ,i=2,…,4, j=2,…,4

4. Рассматриваем элементы второго стобца, среди них выбираем максимальный. Строку в которой стоит максимальный элемент. Меняем местами со втрой. Выбираем ведущий элемент во втором уравнении a(1)22 и так далее.

Если a(0)11,a(1)22,a(2)33,a(3)44 ненулевые, то матрица А примет треугольный вид:



Из системы 3 отыскиваем https://helpiks.org/helpiksorg/baza8/823318767786.files/image558.gif следующим образом

x4=f(4)4

x3=f(3)3-a(3)34x4

x2=f(2)2-a(2)23x3-a(2)24x4

x1=f(1)1-a(1)12x2-a(1)13x3-a(1)14x4

Процесс приведения матрицы к треугольному виду называется прямым ходом метода Гаусса, а нахождение корней по формулам - обратным ходом метода Гаусса.

**Листинг программы**

#include <iostream>

#include <iomanip>

using namespace std;

int i, j, k;

double\*\* CopyMatrix(double\*\* Matrix, int rows, int cols) {

int i = 0;

int j = 0;

double\*\* result = new double\* [rows];

while (i < rows) {

result[i] = new double[cols];

j = 0;

while (j < cols) {

result[i][j] = Matrix[i][j];

j++;

}

i++;

}

return result;

}

void printMatrix(double\*\* matrix, int rows, int cols);

double calculateAccuracy(double\* x, int dim)

{

double\* y = new double[dim];

double m, n;

double pogr;

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

{

y[i] = i + 1;

}

n = 0;

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

{

if (abs(y[i] - x[i]) > n)

{

n = abs(y[i] - x[i]);

}

}

pogr = n / y[dim - 1];

return pogr;

}

void matrixMultiplication(double\*\* first, double\*\* second, int dim)

{

double\*\* result = new double\* [dim];

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

{

result[i] = new double[dim];

}

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

{

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

{

result[i][j] = 0;

for (int k = 0; k < dim; k++)

{

result[i][j] += (first[i][k] \* second[k][j]);

}

}

}

cout << "Проверка: умноженеи исходной матрицы на обратную" << endl;

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

{

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

{

cout << setw(15) << result[i][j];

}

cout << endl;

}

}

void findReverse(double\*\* a, int dim)

{

//прямой ход

int j\_max;

double t;

double\*\* matrix = new double\*[dim];

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

{

matrix[i] = new double[dim \* 2];

}

double\*\* result = new double\* [dim];

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

{

result[i] = new double[dim];

}

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

{

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

{

matrix[i][j] = a[i][j];

}

}

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

{

for (int j = dim; j < dim\*2; ++j)

{

if (i == j - dim)

{

matrix[i][j] = 1;

}

else

{

matrix[i][j] = 0;

}

}

}

for (int k = 0; k < dim; k++)

{

j\_max = k;

for (j = k; j < dim; j++)

if (fabs(matrix[j\_max][k]) < fabs(matrix[j][k]))

j\_max = j;

//переставляем строки

for (i = 0; i < dim \* 2; i++)

{

t = matrix[k][i];

matrix[k][i] = matrix[j\_max][i];

matrix[j\_max][i] = t;

}

//выполнение вычислений

for (j = dim\*2 -1; j >= k; j--)

matrix[k][j] = matrix[k][j] / matrix[k][k];//делим на коэффиц. a[1][1],a[2][2] и т.д.

for (i = k + 1; i < dim; i++)

for (j = dim \* 2 - 1; j >= k; j--)

matrix[i][j] = matrix[i][j] - matrix[i][k] \* matrix[k][j];//вычитаем из 2 ур-ия 1-ое умнож. на коэфф. и т.д. \*/

}

//обратный ход

for (int i = dim - 1; i >= 0; --i)

{

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

{

result[i][j] = matrix[i][dim + j ];// -1

for (int k = dim - 1; k > i; --k)

{

result[i][j] -= matrix[i][k] \* result[k][j];

}

}

}

cout << "Обратная матрица" << endl;

printMatrix(result, dim, dim);

matrixMultiplication(result, a, dim); // проверяем правильно ли нашли обратную

}

void multiply(double\*\*matrix, int dim)

{

double \*vector = new double[dim];

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

{

vector[i] = i + 1;

}

double\* result = new double[dim];

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

{

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

{

result[i] = 0;

for (int k = 0; k < dim; k++)// columns\_1 = lines\_2

{

result[i] += matrix[i][k] \* vector[k];

}

}

}

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

{

matrix[i][dim] = result[i];

}

}

void printMatrix(double\*\* matrix, int dim)

{

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

{

for (int j = 0; j < dim+1; ++j)

{

cout << setw(15) << setprecision(10) << matrix[i][j] ;

}

cout << endl;

}

}

void printMatrix(double\*\* matrix, int rows, int cols)

{

//cout << "Matrix" << endl;

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

{

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

{

cout << setw(15) << setprecision(5) << matrix[i][j];

}

cout << endl;

}

}

void randomMatrix(double\*\* matrix, int dim)

{

srand(time(0));

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

{

for (int j = 0; j < dim+1; ++j)

{

matrix[i][j] = -100 + rand() % ( 200+1);

}

}

}

void InputMatrix(double\*\* matrix, int rows, int cols) {

cout << "Введите матрицу" << endl;

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

{

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

{

cin >> matrix[i][j];

}

}

}

void TaskSecond(double\*\* matrix, int dim);

double Task(int p) {

int dim = p;

double\* x = new double[dim];

double\*\* matrix = new double\* [dim];

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

{

matrix[i] = new double[dim + 1];

}

randomMatrix(matrix, dim);

multiply(matrix, dim); // вычисляем столбец решений

//прямой ход

int j\_max;

double t;

for (int k = 0; k < dim; k++)

{

j\_max = k;

for (j = k; j < dim; j++)

if (fabs(matrix[j\_max][k]) < fabs(matrix[j][k]))

j\_max = j;

//переставляем строки

for (i = 0; i < dim + 1; i++)

{

t = matrix[k][i];

matrix[k][i] = matrix[j\_max][i];

matrix[j\_max][i] = t;

}

//если максимальный элемент нулевой

int count;

if (matrix[j\_max][k] == 0)

{

count = 0;

for (i = 0; i < dim; i++)

{

if (matrix[k][i] == 0)

count++;

if (count == dim && (matrix[k][dim + 1] == 0))//если коэфф в строке = 0 и своб член = 0

double eps = 1e-12;

if (count == dim && (abs(matrix[k][dim + 1] )<= eps))//если коэфф в строке = 0 и своб член = 0

{

cout << "Система имеет бесконечное множество решений" << endl;

return ;

}

}

}

//выполнение вычислений

for (j = dim; j >= k; j--)

matrix[k][j] = matrix[k][j] / matrix[k][k];//делим на коэффиц a[1][1],a[2][2] и тд

for (i = k + 1; i < dim; i++)

for (j = dim; j >= k; j--)

matrix[i][j] = matrix[i][j] - matrix[i][k] \* matrix[k][j];//вычитаем из 2 ур-ия 1-ое умнож на коэфф и тд \*/

}

//обратный ход

for (int i = dim - 1; i >= 0; --i)

{

x[i] = matrix[i][dim];

for (int j = dim - 1; j > i; --j)

{

x[i] -= matrix[i][j] \* x[j];

}

}

double max = 0;

for (i = 0; i < dim; i++)

{

if (abs(x[i] - i - 1) > max) {

max =abs( x[i] - i - 1);

}

}

double accuracy = max / dim;

return accuracy;

}

void TaskFirst() {

setlocale(LC\_ALL, "Russian");

int dim = 10;

//cin >> dim;

double\* x = new double[dim];

double\*\* matrix = new double\* [dim];

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

{

matrix[i] = new double[dim + 1];

}

double\*\* matrix\_copy = new double\* [dim];

//InputMatrix(matrix, dim, dim );

randomMatrix(matrix, dim);

multiply(matrix, dim); // вычисляем столбец решений

matrix\_copy = CopyMatrix(matrix, dim, dim);

cout << "Исходная матрица:" << endl;

printMatrix(matrix, dim);

//прямой ход

int j\_max;

double t;

for (int k = 0; k < dim; k++)

{

j\_max = k;

for (j = k; j < dim; j++)

if (fabs(matrix[j\_max][k]) < fabs(matrix[j][k]))

j\_max = j;

//переставляем строки

for (i = 0; i < dim + 1; i++)

{

t = matrix[k][i];

matrix[k][i] = matrix[j\_max][i];

matrix[j\_max][i] = t;

}

//если максимальный элемент нулевой

int count;

if (matrix[j\_max][k] == 0)

{

count = 0;

for (i = 0; i < dim; i++)

{ double eps = 1e-12;

if (matrix[k][i] == eps)

count++;

if (count == dim && (abs(matrix[k][dim + 1] )<= eps))//если коэфф в строке = 0 и своб член = 0

{

cout << "Система имеет бесконечное множество решений" << endl;

return ;

}

else if (count == dim && abs( matrix[k][dim + 1]) != eps)//если коэфф в строке = 0 и своб член не равен 0

{

cout << "Система не имеет решений" << endl;

return ;

}

}

}

//выполнение вычислений

for (j = dim; j >= k; j--)

matrix[k][j] = matrix[k][j] / matrix[k][k];//делим на коэффиц a[1][1],a[2][2] и тд

for (i = k + 1; i < dim; i++)

for (j = dim; j >= k; j--)

matrix[i][j] = matrix[i][j] - matrix[i][k] \* matrix[k][j];//вычитаем из 2 ур-ия 1-ое умнож на коэфф и тд \*/

}

cout << "Матрица после приведения к треуголному виду:" << endl;

printMatrix(matrix, dim);

//обратный ход

for (int i = dim - 1; i >= 0; --i)

{

x[i] = matrix[i][dim];

for (int j = dim - 1; j > i; --j)

{

x[i] -= matrix[i][j] \* x[j];

}

}

//вывод решения

cout << "Найденные решения" << endl;

for (i = 0; i < dim; i++)

cout << setprecision(20) << x[i] << endl;

double max = 0;

for (i = 0; i < dim; i++)

{

if (abs(x[i] - i - 1 )> max) {

max = abs(x[i] - i - 1);

}

}

double accuracy = max / dim;

accuracy = calculateAccuracy(x, dim);

cout << "Погрешность равна: " << setprecision(20) << accuracy << endl;

TaskSecond(matrix\_copy, dim);

}

void TaskSecond( double\*\* matrix, int dim)

{

findReverse(matrix, dim);

}

void TaskThird()

{

cout << setw(15) << "Порядок" << "\t"<< setw(15) << "Относительная погрешность "<< endl;

for (int i = 7; i <= 107; i += 10)

{

cout << setw(15) << i << "\t"<< setw(15) << Task(i) << endl;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

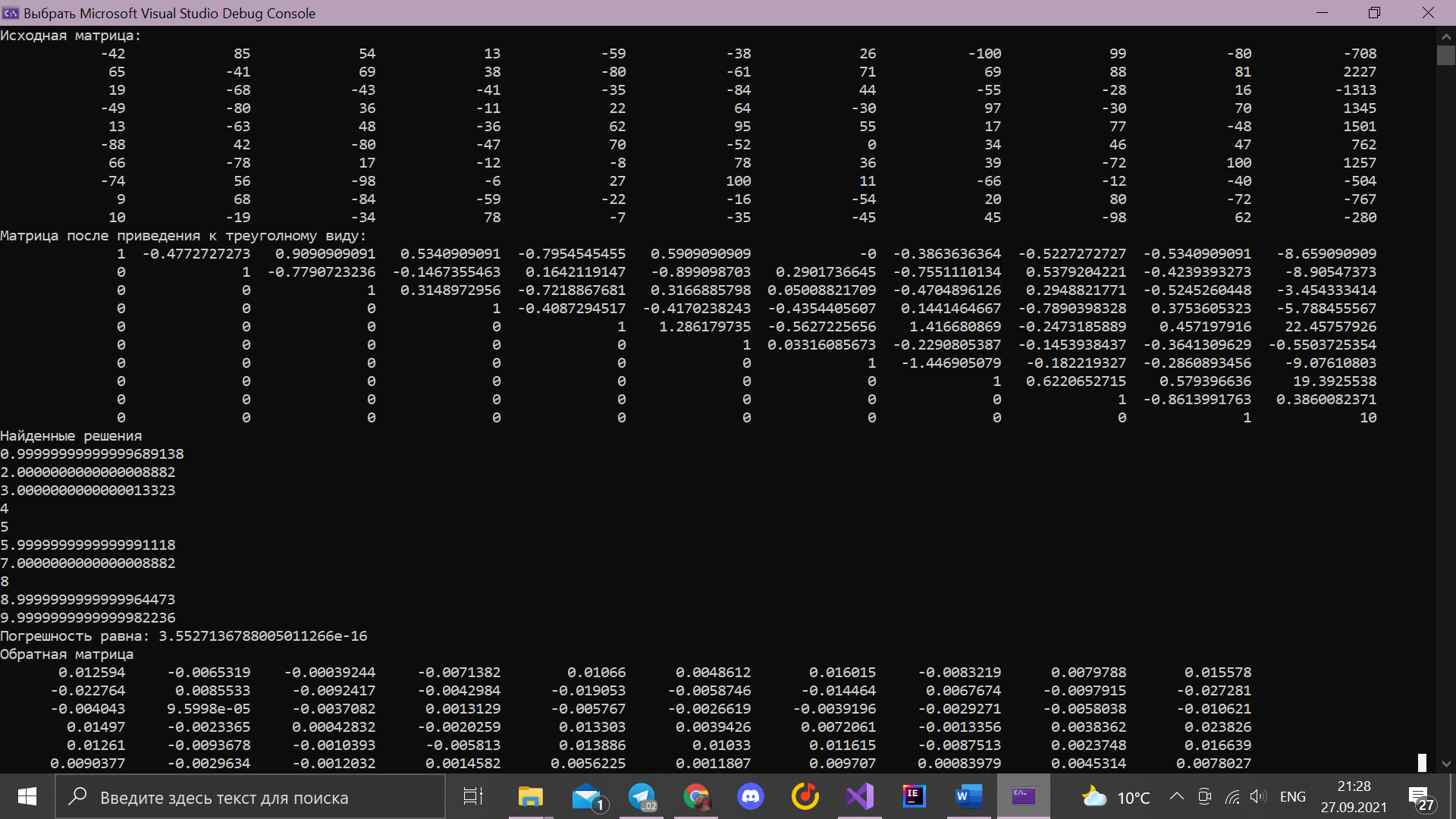
TaskFirst();

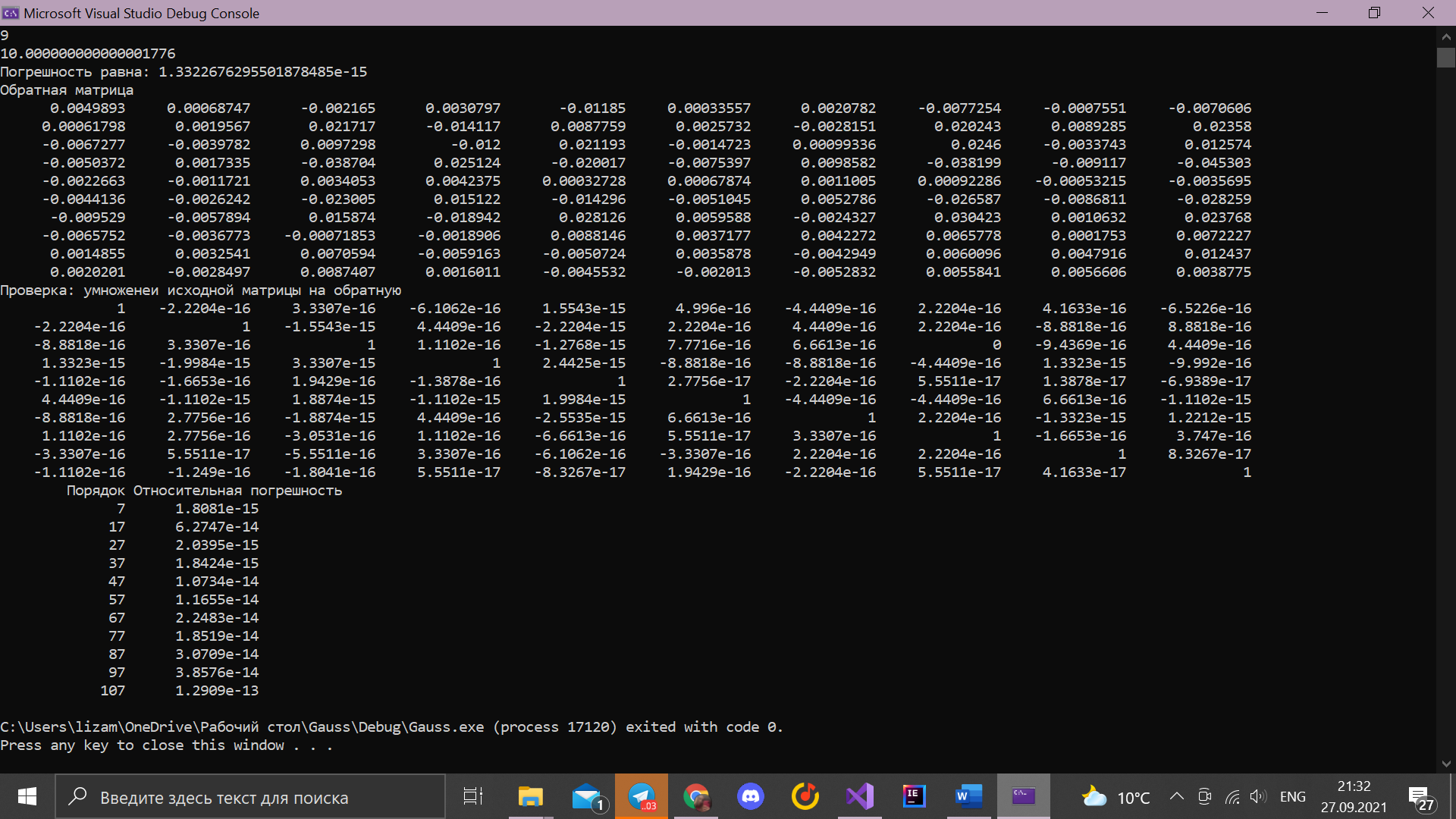
TaskThird();

return 0;

}

**Результат вывода программы**

****

****

| **Порядок матрицы** | **Относительная погрешность** |
| --- | --- |
| **7** | **1.8081e-15** |
| **17** | **6.2747e-14** |
| **27** | **2.0395e-15** |
| **37** | **1.8424e-15** |
| **47** | **1.0734e-14** |
| **57** | **1.1655e-14** |
| **67** | **2.2483e-14** |
| **77** | **1.8519e-14** |
| **87** | **3.0709e-14** |
| **97** | **3.8576e-14** |
| **107** | **1.2909e-13** |

**Выводы**

При увеличении порядка матрицы относительная погрешность вычислений изменяетяс незначитьельно и зависит от выбора ведущих элементов. Метод Гаусса с выбором ведущего элемента по столбцу позволяет уменьшить относительную погрешность вычислений.