**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Основы программирования»

Отчет по лабораторной работе №10

«Вычисление обратной матрицы методом Гаусса-Жордана»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-12Б  Казицин Алексей |  | преподаватель каф. ИУ5  Козлов А.Д. |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

Постановка задачи

В математике все доказательства, связанные с получением обратной матрицы, полностью основаны на применении определителей. Этот процесс ясно показывает способ вычисления каждого элемента обратной матрицы, но не эффективен при обращении матриц большого порядка, поскольку требует излишних вычислений.

На практике для получения обратной матрицы используются методы, основанные на решении системы линейных алгебраических уравнений.

В лабораторной работе рассматривается вычисление обратной матрицы методом Гаусса-Жордана. Гаусс предложил метод решения систем линейных алгебраических уравнений путем последовательного исключения переменных, а затем Жордан использовал его для нахождения обратной матрицы, добавив преобразование единичной матрицы.

**Цель работы:**

- разработка программы, реализующей алгоритм вычисления обратной матрицы методом Гаусса-Жордана;

- использование разработанных функций для решения систем линейных уравнений.

***Задание***. Создать функцию для вычисления обратной матрицы по методу Гаусса-Жордана. Размер матрицы передавать в функцию в качестве параметра. Для упрощения алгоритма следует присоединить единичную матрицу справа к исходной и выполнять все преобразования над объединенной матрицей размером N\*2N. Обратная матрица получится на месте единичной в столбцах N…2N, а на месте исходной матрицы в столбцах 0…(N-1) должна получиться единичная матрица.

Включить в алгоритм проверку на существование обратной матрицы. Для этого в в прямом ходе перед делением выполнить проверку на ноль элементов главной диагонали исходной матрицы. Если элемент равен 0, то нужно поменять местами текущую строку с одной из нижележащих строк, в которой элемент в соответствующем столбце не равен 0.

Если таких строк нет, то выдать сообщение: «Обратная матрица не существует».

Применить функцию для решения системы линейных алгебраических уравнений.

Шаги выполнения задания.

Часть 1.

* 1. Разработать схему алгоритма для вычисления обратной матрицы методом Гаусса-Жордана.
  2. Разработать программу, реализующую метод Гаусса-Жордана. Для ускорения процесса разработки программы на этом шаге следует сконцентрироваться на реализации алгоритма и не использовать динамическую память для хранения матриц и не оформлять реализацию алгоритма в виде функции.
  3. Отладить программу, используя контрольный пример.
  4. Оформить алгоритм в виде функции для вычисления обратной матрицы по методу Гаусса-Жордана. Размер матрицы передавать в функцию в качестве параметра.

Часть 2.

* 1. Выполнить тестирование программы. Проверку правильности результатов вычислений выполнять путем умножения полученной обратной матрицы на исходную (в результате должна получиться единичная матрица). Для реализации контроля разработайте функцию, реализующую произведение прямоугольных матриц. В процессе выполнения программы на экран должы выводиться следующие данные: исходная матрица, промежуточные результаты после прямого и обратного ходов, обратная матрица и результат умножения прямой матрицы на обратную. Для печати матрицы использовать функцию из лабораторной работы 10.
  2. Доработать алгоритм для случая, когда на главной диагонали появляется ноль. Включить в алгоритм проверку на существование обратной матрицы.
  3. Применить функцию вычисления обратной матрицы для решения системы линейных алгебраических уравнений.

Разработка алгоритма

**Описание используемых переменных и функций с указанием наименования, типа (int, float, и т.п.) и назначения в программе:**

double \*\*mat – исходная матрица;

bool flag – флаг, проверка на то, существует ли обратная матрица;

int n – размер матрицы;

double a, b, c – вспомогательные переменные (для работы в функциях)

double \*svob – массив для хранения свободных членов в решении СЛАУ;

double \*\*mat\_ob – обратная матрица;

double \*\*mat\_ed – единичная матрица;

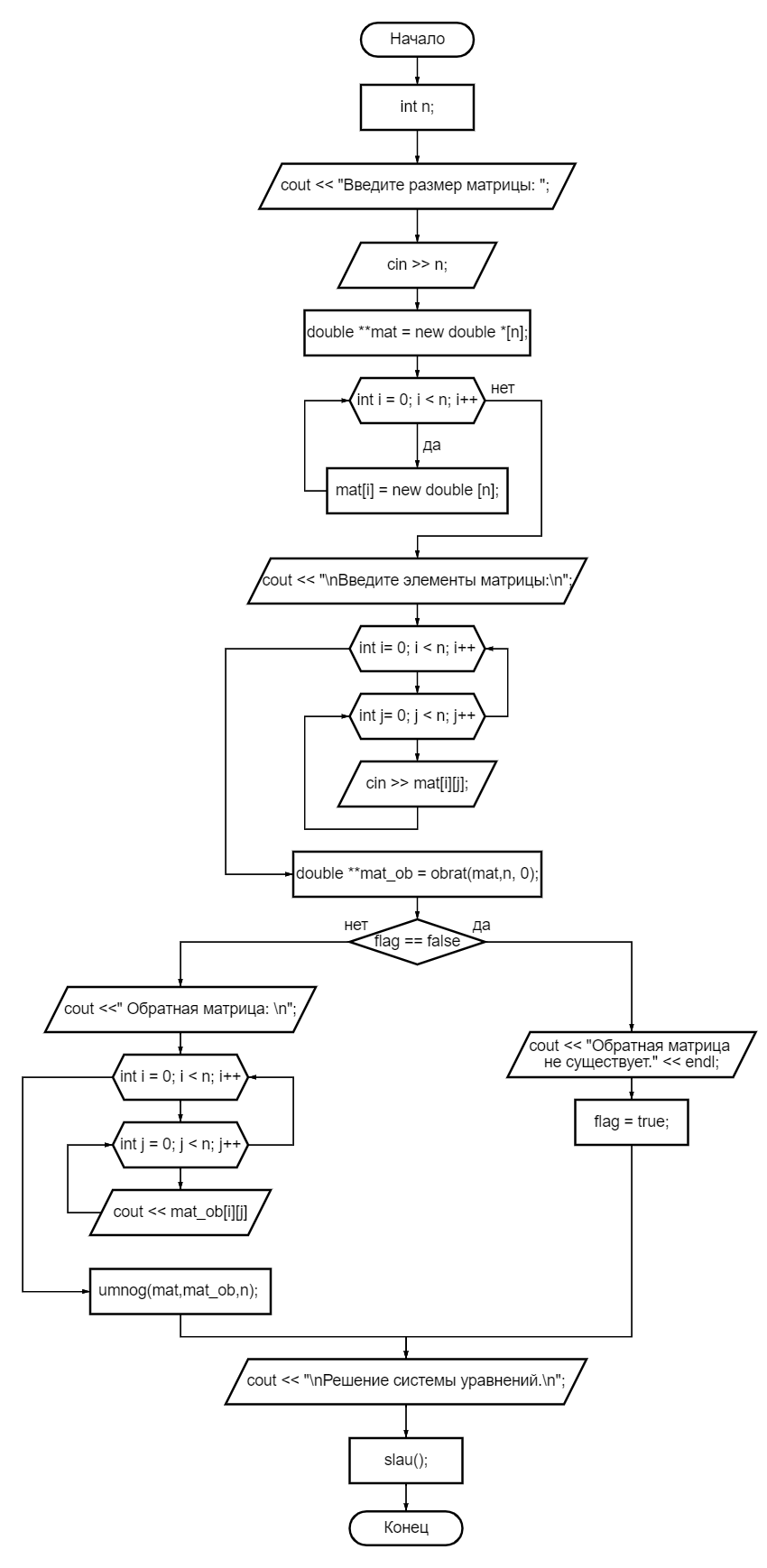
double \*\*mat\_help – вспомогательная матрица (справа исходная, слева единичная);

double \*mas – массив для получения нулевых элементов в матрице;

int i, j, k – счетчики в циклах;

int tip – параметр в функции создания обратной матрицы (нужен чтобы в решении СЛАУ не выводились промежуточные результаты, так как они нужны только в первой части задачи);

Блок-схема алгоритма (int main)



Текст программы

#include <iostream>

#include <iomanip>

#include <cmath>

#include <windows.h>

using namespace std;

double \*\*obrat(double \*\*mat, int n, int tip);

void umnog(double \*\*mat1, double \*\*mat2, int n);

void umnog(double \*\*mat1, double \*mat2, int n);

void slau();

bool flag = true;

int main(){

SetConsoleCP(65001);

SetConsoleOutputCP(65001);

int n;

cout << "Введите размер матрицы: ";

cin >> n;

double \*\*mat = new double \*[n];

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

mat[i] = new double [n];

//---------------------Ввод матрицы-------------------------

cout << "\nВведите элементы матрицы:\n";

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

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

cin >> mat[i][j];

//------------------Нахождение обратной матрицы----------------------

double \*\*mat\_ob = obrat(mat,n, 0);

if (flag == false){

cout << "Обратная матрица не существует." << endl;

flag = true;

}else{

cout <<" Обратная матрица: \n";

for (int i = 0; i < n; i++){

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

cout << setw(8) << fixed << setprecision(3) << mat\_ob[i][j] << " ";

cout <<"\n";

}

umnog(mat,mat\_ob,n); // Проверка умножением

}

//-------------------------Решение СЛАУ----------------------------

cout << "\nРешение системы уравнений.\n";

slau();

system("pause");

return 0;

}

double \*\*obrat(double \*\*mat, int n, int tip){

double a,b,c;

double \*mas= new double[2\*n];

// Создание единичной матрицы

double \*\*mat\_ed= new double \*[n];

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

mat\_ed[i]= new double [n];

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

for(int j= 0; j < n; j++){

if(i == j) mat\_ed[i][j] = 1;

else mat\_ed[i][j] = 0;

}

// Создание вспомогательной матрицы

double \*\*mat\_help= new double \*[n];

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

mat\_help[i]= new double [2\*n];

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

for(int j= 0; j < (2\*n); j++){

if(j < n)

mat\_help[i][j] = mat[i][j];

else mat\_help[i][j] = mat\_ed[i][j-n];

}

// Прямой ход, в конце цикла получается верхняя треугольная матрица

for(int k= 0; k < n; k++){

if (mat\_help[k][k] == 0){ //проверка диагональных элементов на равенство нулю (существует ли обратная матрица)

flag = false;

for(int i= k+1; i < n; i++){

if(mat\_help[i][k] != 0){

flag = true;

for(int j= 0; j < n\*2; j++){

c= mat\_help[i][j];

mat\_help[i][j] = mat\_help[k][j];

mat\_help[k][j] = c;

}

break;

}

}

}

if (flag == false){

break;

}

for(int i= k; i < n; i++){

if(i == k){

for(int j= 0; j < 2\*n; j++)

mas[j] = mat\_help[i][j];

}

for(int j= k; j < 2\*n; j++){

if(i == k){

if(j == k) a = mat\_help[k][k];

mat\_help[i][j]= mat\_help[i][j]/a;

}else{

if(j == k) b = mat\_help[i][j];

mat\_help[i][j] = mat\_help[i][j]-mas[j]\*b/a;

}

}

}

}

if (flag == false){

return 0;

}else{

if (tip == 0){

//Промежуточный вывод прямого хода

cout << "Прямой ход (промежуточный результат):" << endl;

for (int i = 0; i < n; i++){

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

cout << setw(8) << fixed << setprecision(3) << mat\_help[i][j] << " ";

cout <<"\n";

}

cout << endl;

}

// Обратный ход (прямой ход, только зануляем правые верхние элементы)

for (int k = n-1; k >= 0; k--)

for(int i = k; i >= 0; i--){

if(i == k){

for(int j = 0; j < 2\*n; j++)

mas[j] = mat\_help[i][j];

}

for(int j = k; j < 2\*n; j++){

if(i == k){

if(j == k) a = mat\_help[k][k];

mat\_help[i][j] = mat\_help[i][j]/a;

}else{

if(j == k) b = mat\_help[i][j];

mat\_help[i][j] = mat\_help[i][j]-mas[j]\*b/a;

}

}

}

double \*\*mat\_ob = new double\*[n];

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

mat\_ob[i] = new double [n];

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

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

mat\_ob[i][j] = mat\_help[i][j+n];

if (tip == 0){

//Промежуточный вывод обратного хода

cout << "Обратный ход (промежуточный результат):" << endl;

for (int i = 0; i < n; i++){

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

cout << setw(8) << fixed << setprecision(3) << mat\_ob[i][j] << " ";

cout <<"\n";

}

cout << endl << endl;

}

return mat\_ob;

}

}

void umnog(double \*\*mat1, double \*\*mat2, int n){

double s;

double \*\*mat = new double \*[n];

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

mat[i] = new double [n];

for(int i = 0; i < n; i++){

for(int j = 0; j < n; j++){

s= 0;

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

s = s+mat1[i][k]\*mat2[k][j];

if(i == j)mat[i][j] = 1;else

mat[i][j] = s;

}

}

cout <<" Умножение обратной на исходную: \n";

for (int i = 0; i < n; i++){

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

cout << setw(8) << fixed << setprecision(3) << abs(mat[i][j]) << " ";

cout <<"\n";

}

}

void umnog(double \*\*mat1, double \*mat2, int n){

double s;

double \*mat = new double[n];

for (int i = 0; i < n; i++){

s= 0;

for (int j = 0; j < n; j++){

s = s+mat1[i][j]\*mat2[j];

}

mat[i]= s;

}

cout << "\nРезультат:\n";

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

cout << "x" << i+1 << " = " << mat[i] << "\n";

}

void slau(){

int n;

cout << "Введите количество уравнений в системе - ";

cin >> n;

double \*\*mat= new double \*[n];

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

mat[i]= new double [n];

for (int i = 0; i < n; i++){

cout << "Введите коэффициенты уравнения: " <<i+1 <<"\n";

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

cin >> mat[i][j];

}

double \*\*mat\_ob = obrat(mat,n, 1);

double \*svob = new double [n];

cout << "Введите свободные члены уравнений:\n";

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

cin >> svob[j];

umnog(mat\_ob, svob, n);

}

Анализ результатов

