

НИЯУ МИФИ

Отчет по курсу СГМ

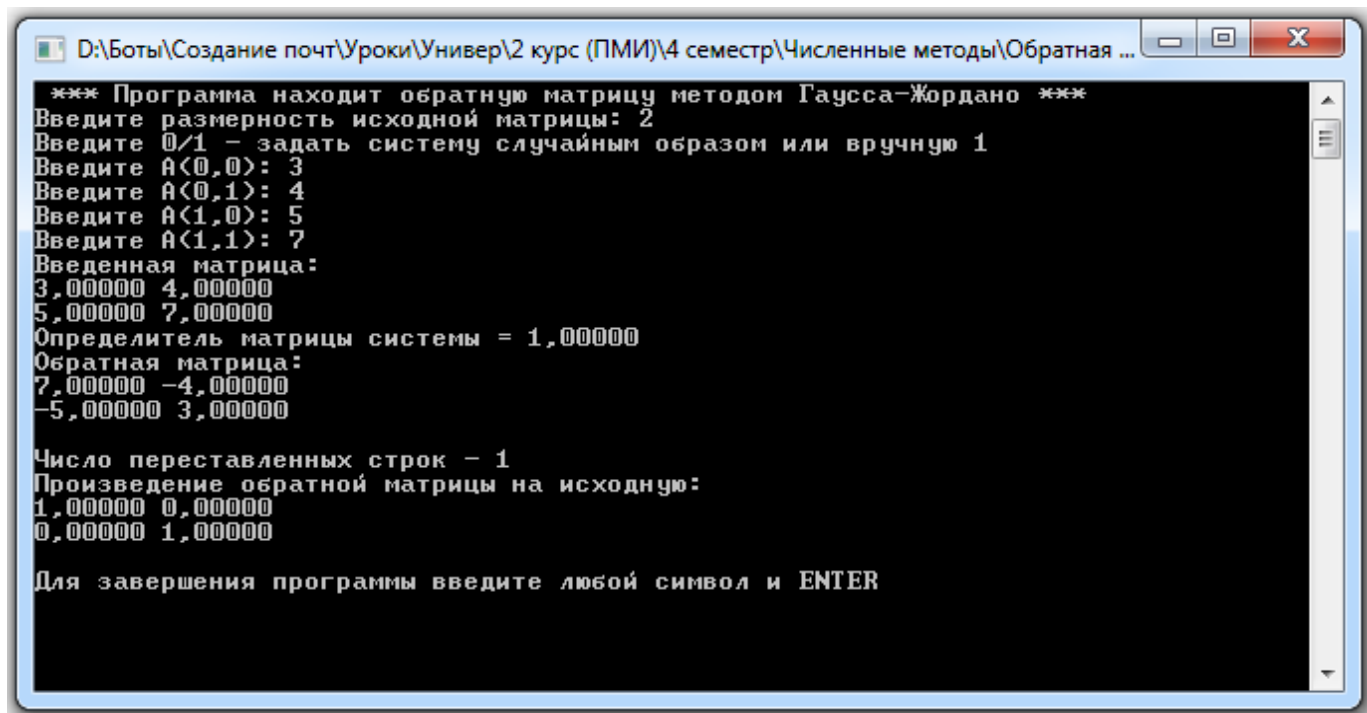
«Нахождение обратной матрицы методом Гаусса-
Жордана»

Выполнил: студент группы Б17-511 Чудновец Иван

Проверил: Козин Рудольф Глебович

2019

Скриншот работы программы:



```
D:\Боты\Создание почт\Уроки\Универ\2 курс (ПМИ)\4 семестр\Численные методы\Обратная ...
*** Программа находит обратную матрицу методом Гаусса-Жордано ***
Введите размерность исходной матрицы: 2
Введите 0/1 - задать систему случайным образом или вручную 1
Введите A<0,0>: 3
Введите A<0,1>: 4
Введите A<1,0>: 5
Введите A<1,1>: 7
Введенная матрица:
3,000000 4,000000
5,000000 7,000000
Определитель матрицы системы = 1,000000
Обратная матрица:
7,000000 -4,000000
-5,000000 3,000000
Число переставленных строк - 1
Произведение обратной матрицы на исходную:
1,000000 0,000000
0,000000 1,000000
Для завершения программы введите любой символ и ENTER
```

Код программы:

```
#include <iostream>
#include <cmath>
#include <time.h>
using namespace std;
#define B(i,j) MM[i*n+j] //исходная система
#define A(i,j) M[i*n+j] //приводимая система
#define E(i,j) ME[i*n+j] //под единичную матрицу
#define L(i,j) LL[i*n+j] //массив для номеров переставленных столбцов
float *M, *MM, *ME, *LL, det = 1;
int n, m; //размерность и число перестановок
int n1, n2;
void fine();
void select_Amax(int k);
void test();
int main()
{
    setlocale(LC_ALL, "");
    int i, j, c;
    char s[2];
    cout<<" *** Программа находит обратную матрицу методом Гаусса-Жордано *** ";
    cout<<"\nВведите размерность исходной матрицы: ";
    cin >> n;
    M = (float *)malloc(n*n * sizeof(float)); //матрица - A
    MM = (float *)malloc(n*n * sizeof(float)); //копия A - B
    ME = (float *)malloc(n*n * sizeof(float)); //для теста единичной матрицы
    LL = (float *)malloc(2 * n * sizeof(float)); //хранит номера столбцов
    n1 = n - 1; //чтобы обращаться к компонентам матриц A, B, L
    cout<<"Введите 0/1 - задать систему случайным образом или вручную ";
    cin >> c;
    switch (c) {
    case 0:
        //инициализация датчика п.с. чисел текущим временем
        srand(time(NULL));
        for (i = 0; i < n; i++)for (j = 0; j < n; j++)B(i, j) = 0.5 - rand() / (RAND_MAX +
1.0); //[-0.5;0.5]
        break;
    default:
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                printf("Введите A(%d,%d): ", i, j);
                cin >> B(i, j);
            }
        }
    }
    printf("Введенная матрица:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%.5f ", B(i, j));
            A(i, j) = B(i, j);
        }
        printf("\n");
    }
    //printf("A(n-1,n-1) = %.3f\n",A(n1,n1));
    fine();
    printf("Определитель матрицы системы = %.5f", det);
    if (fabs(det) < 1.E-10) {
        printf("\nМатрица системы ВЫРОЖДЕННАЯ !");
    }
    else {
        printf("\nОбратная матрица:\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                printf("%.5f ", A(i, j));
            }
            printf("\n");
        }
        printf("\nЧисло переставленных строк - %d", m);
    }
}
```

```

    test();
    printf("\nПроизведение обратной матрицы на исходную:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%.5f ", E(i, j));
        }
        printf("\n");
    }
    printf("\nДля завершения программы введите любой символ и ENTER");
    cin >> s;
    return 0;
}

//вычисляет произведение обратной матрицы на исходную
void test() {
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            E(i, j) = 0;
            for (k = 0; k < n; k++) {
                E(i, j) = E(i, j) + A(i, k)*B(k, j);
            }
        }
    }
}

//процедура выбора ведущего элемента в k-ой подстроке
void select_Amax(int k) {
    int i, kmax, j;
    float Amod, Amax, buf;
    if (k < (n - 1))
    {
        Amax = fabs(A(k, k)); kmax = k;
        for (j = (k + 1); j < n; j++)
        {
            Amod = fabs(A(k, j));
            if (Amod > Amax)
            {
                Amax = Amod; kmax = j;
            }
        }
        if (Amax < 1.E-10) // матрицы вырожденная
        {
            det = 0;
            return;
        }
        if (kmax != k)
        {
            for (i = 0; i < n; i++) // переставляем столбцы
            {
                buf = A(i, k);
                A(i, k) = A(i, kmax);
                A(i, kmax) = buf;
            }
            det = -det;
            L(0, m) = k; //сохраняем номера переставленных столбцов
            L(1, m) = kmax;
            m++;
        }
    }
    det = det * A(k, k);
}

// нахождение обратной матрицы методом Гаусса-Жордана
void fine()
{
    int i, j, k, m0, m1;
    float buf;
    det = 1; m = 0;
    for (k = 0; k < n; k++)
    {

```

```

select_Amax(k); // выбор ведущего элемента
if (det == 0) return;
for (i = 0; i < n; i++)
{
    if (i != k)
    {
        A(i, k) = -A(i, k) / A(k, k);
        for (j = 0; j < n; j++)
        {
            if (j != k)
            {
                A(i, j) = A(i, j) + A(i, k)*A(k, j);
            }
        }
    }
}
for (j = 0; j < n; j++)
{
    if (j != k)
    {
        A(k, j) = A(k, j) / A(k, k);
    }
}
A(k, k) = 1 / A(k, k);
}
if (m > 0) // переставляем строки в обратной матрице
{
    for (i = (m - 1); i >= 0; i--)
    {
        m0 = L(0, i);
        m1 = L(1, i);
        for (j = 0; j < n; j++)
        {
            buf = A(m0, j);
            A(m0, j) = A(m1, j);
            A(m1, j) = buf;
        }
    }
}
}

```