Министерство образования и науки Российской федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Кубанский государственный технологический университет

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Информатика и программирование» .

на тему Разработка приложения по решению систем линейных уравнений .

Выполнил студент группы 15-КБ-ПИ1 Ручка Артем Алексеевич .

Допущен к защите .

Руководитель, нормоконтролер работы В.А. Мурлина .

Защищен Оценка .

Члены комиссии .

.

.

Краснодар

2015

Министерство образования и науки Российской федерации Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Кубанский государственный технологический университет (ФГБОУ ВПО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

УТВЕРЖДАЮ

Зав. кафедрой ИСП д-р техн. наук

профессор Л.А. Видовский

« » 2015г

ЗАДАНИЕ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

Студенту Ручка А.А. группы 15-КБ-ПИ1 курса 1 .

института компьютерных систем и информационной безопасности .

направления 090303 - Прикладная информатика .

Тема работы: Разработка приложения по решению систем линейных уравнений.

Содержание задания: разработать программу в среде VISUAL STUDIO 2010 на языке С#, решающую систему линейных уравнений методами обратной матрицы, Гаусса, Крамера.

Объем работы:

а) пояснительная записка к проекту с.;

б) программы .

Рекомендуемая литература: Павловская Т.А. С# Программирование на языке высокого уровня Учебник для вузов – СПб.: Питер. 2014 – 432с.

Срок выполнения работы с “10” ноября по “25”декабря 2015 г

Срок защиты “15” декабря 2015 г

Руководитель работы к.т.н., доцент В.А. Мурлина

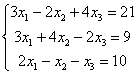
Задание принял студент Ручка А.А.

**Реферат**

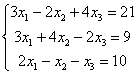
Пояснительная записка курсовой работы - 30 с., 3 рис., 7 источников.

VISUAL STUDIO 2010, C#, МЕТОД ОБРАТНОЙ МАТРИЦЫ, МЕТОД КРАМЕРА, МЕТОД ГАУССА.

Объектом исследования является матрица



Цель работы: разработать программу в среде VISUAL STUDIO 2010 на языке С#, решающую данную систему уравнений



В результате работы получен результат http://www.mathprofi.ru/f/pravilo_kramera_matrichnyi_metod_clip_image155.gif

Содержание

[Введение 5](#_Toc437896808)

[1 Нормативные ссылки 6](#_Toc437896809)

[2 Постановка задачи 7](#_Toc437896810)

[3 Описание методов вычислительной математики 8](#_Toc437896811)

[3.1 Метод обратной матрицы 8](#_Toc437896812)

[3.2 Метод Крамера 9](#_Toc437896813)

[3.3 Метод Гаусса 9](#_Toc437896814)

[4 Описание методов программирования 10](#_Toc437896815)

[4.1 Оператор выбора Switch 10](#_Toc437896816)

[4.2 Цикл с параметром For 10](#_Toc437896817)

[4.3 Оператор Break 10](#_Toc437896818)

[4.4 Оператор if 10](#_Toc437896819)

[4.5 Цикл с предусловием While 10](#_Toc437896820)

[5 Блок схема программы 11](#_Toc437896821)

[6 Листинг программы 15](#_Toc437896822)

[7 Описание программы 24](#_Toc437896823)

[8 Результат тестирования программы 26](#_Toc437896824)

[Заключение 29](#_Toc437896825)

[Список используемых источников 30](#_Toc437896826)

# Введение

Последние десятилетия характеризуются бурным развитием вычислительной техники. Расширяются области применения вычислительных машин и совершенствуются методы их использования. Созданы универсальные языки программирования и разработаны мощные операционные системы.

Сейчас невозможно представить себе какую-либо область деятельности, обходящуюся без применения компьютерной техники. Компьютеры используются при проведении различных инженерных расчетов, при решении экономических задач, в процессе управления производством, при получении оценок производственных ситуаций и во многих других случаях.

Системы линейных уравнений появляются почти в каждой области прикладной математики. В некоторых случаях эти системы уравнений непосредственно составляют ту задачу, которую необходимо решать, в других случаях задача сводится к такой системе.

Чтобы быстро справиться с решением системы линейных уравнений, можно воспользоваться средствами вычислительной техники - составить программу на языке программирования.

В данной курсовой работе рассматривается возможность решения систем линейных уравнений матричным способом, методом Крамера и методом Гаусса с помощью программы, созданной на языке программирования C#.

**1 Нормативные ссылки**

В данной пояснительной записке использованы ссылки на следующие стандарты:

ГОСТ Р 1.5-2004. Стандарты национальные РФ. Правила построения, изложения, оформления и обозначения.

ГОСТ 2.301-68 ЕСКД. Форматы.

ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления.

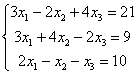
ГОСТ 7.12-93 СИБИД. Библиографическая запись. Сокращения слов на русском языке. Общие требования и правила.

ГОСТ 7.9-95 СИБИД. Реферат и аннотация. Общие требования.

ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления.

**2 Постановка задачи**

В данной курсовой работе необходимо написать программу в среде разработки VISUAL STUDIO 2010 на языке C#, способную решить данную систему уравнений



методами обратной матрицы, Крамера и Гаусса.

# 3 Описание методов вычислительной математики

## 3.1 Метод обратной матрицы

Матричныйметод решения (метод решения через обратную матрицу) систем линейных алгебраических уравнений с ненулевым определителем состоит в следующем.

Пусть дана система линейных уравнений с n неизвестными (над произвольным полем):


{ \begin{cases}
a_{11}x_1+ \ldots +a_{1n}x_n=b_1, \\
\cdots \cdots \cdots \cdots \cdots \cdots \cdots, \\
a_{n1}x_1+ \ldots +a_{nn}x_n=b_n;
\end{cases} 
}


Тогда её можно переписать в матричной форме:

AX = B, где A — основная матрица системы, B и X — столбцы свободных членов и решений системы соответственно:


A = 
\begin{pmatrix}
a_{11} & a_{12} & \ldots & a_{1n} \\
a_{21} & a_{22} & \ldots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots  \\
a_{n1} & a_{n2} & \ldots & a_{nn}
\end{pmatrix},

B = 
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_n
\end{pmatrix},

X = 
\begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{pmatrix}


Умножим это матричное уравнение слева на A^{-1} — матрицу, обратную к матрице A: A^{-1}\left( AX \right) = A^{-1}B

Так как A^{-1}A = E, получаем X = A^{-1}B. Правая часть этого уравнения даст столбец решений исходной системы. Условием применимости данного метода (как и вообще существования решения неоднородной системы линейных уравнений с числом уравнений, равным числу неизвестных) является невырожденность матрицы A. Необходимым и достаточным условием этого является неравенство нулю определителя матрицы A:

\det A \ne 0.

## 3.2 Метод Крамера

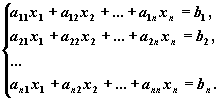
Если определитель D=det *A* матрицы системы *Ax=b* отличен от нуля, то система имеет единственное решение *x1 , x2 , ..., xn*, определяемое *формулами Крамера*

*xi =*D*i /* D, *i=1,2, ..., n*,

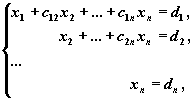
где D*i* - определитель матрицы *n* -го порядка, полученной из матрицы *A* системы заменой *i* -го столбца столбцом правых частей *b*.

## 3.3 Метод Гаусса

*Метод Гаусса* применим для решения системы линейных алгебраических уравнений c невырожденной матрицей системы. Идея метода Гаусса состоит в том, что систему *n* линейных алгебраических уравнений относительно *n* неизвестных *x1 , x2 , ..., xn*



приводят последовательным исключением неизвестных к эквивалентной системе с треугольной матрицей



решение которой находят по рекуррентным формулам:

*xn =dn , xi = di -*S *nk=i+1 cik xk , i=n-1, n-2, ...,1*.

**4 Описание методов программирования**

## 4.1 Оператор выбора Switch

Оператор switch — это оператор управления, выбирающий из списка возможных вариантов *раздел переключения*, для выполнения, содержащегося в нём кода.

Оператор switch включает один или несколько разделов переключения. Каждый раздел переключения содержит одну или несколько меток case, за которыми следует один или несколько операторов.

## 4.2 Цикл с параметром For

С помощью цикла for можно циклически выполнять оператор или блок операторов до тех пор, пока указанное выражение не будет являться false. Этот тип цикла полезен для перебора массивов и других приложений, в которых известно заранее, сколько раз необходимо повторить цикл.

## 4.3 Оператор Break

Оператор break завершает ближайший внешний цикл или оператор switch, в котором он появляется. Управление передается оператору, следующему за завершенным оператором (если таковой имеется).

## 4.4 Оператор if

Условный оператор if служит для ветвления программы. В зависимости от некоторого условия выполняется тот или другой набор команд.

## 4.5 Цикл с предусловием While

Оператор while выполняет оператор или блок операторов, пока определенное выражение не примет значение false.

Цикл whileможет быть завершен, если оператор break, goto, return или throw передает управление за пределы цикла.

**5 Блок схема программы**









# 6 Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication7

{

class Program

{

static void Main(string[] args)

{

int a = 1;

while (a != 2)

{

Console.WriteLine("Выберите метод:");

Console.WriteLine("1) Матричный метод");

Console.WriteLine("2) Метод Крамера");

Console.WriteLine("3) Метод Гаусса");

Console.WriteLine("4) Выход");

Console.Write("Ваш выбор? ");

char x = (char)Console.Read();

Console.ReadLine();

switch (x)

{

case '1': matrix(); break;

case '2': kramer(); break;

case '3': gauss(); break;

case '4': a = 2; break;

}

}

}

static void matrix()//матричный метод

{

int n;

Console.Write("Введите размерность матрицы: ");

n = Convert.ToInt32(Console.ReadLine());

double[] b = new double[n]; // вектор правых частей

double[] x = new double[n]; // вектор решения

double[,] matrix = new double[n, n];// основная матрица

for (int i = 0; i < n; i++)// заполняем матрицу

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

{

Console.Write("A[{0},{1}] = ", i, j);

matrix[i, j] = Convert.ToDouble(Console.ReadLine());

}

double[,] An = new double[n, n];

for (int i = 0; i < n; i++)//клонируем ее

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

An[i, j] = matrix[i, j];

double[,] Ak = new double[n, n];

Console.WriteLine("Введите вектор правой части:");

for (int i = 0; i < n; i++) //заполняем вектор правой части

{

Console.Write("b[{0}] = ", i);

b[i] = Convert.ToDouble(Console.ReadLine());

}

Console.WriteLine("Исходная система уравнений: ");

printMatrix(matrix, n, b); // распечатываем матрицу;

Console.WriteLine();

double det = Determinant(An, n); // вычисляем определитель матрицы;

Console.WriteLine("Определитель равен = {0}", det);

Console.WriteLine();

T(n,An,matrix);

if (det == 0)

Console.WriteLine("Матрица вырождена");

else

{

for (int i = 0; i < n; i++)//создаем матрицу миноров

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

{

double A = Determinant(GetMinor(An, i, j, n), n - 1);

Ak[i, j] = Math.Pow(-1, i + j) \* A;

Ak[i, j] = Ak[i, j] / det;

}

for (int i = 0; i < n; i++)//находим вектор решения

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

{

x[i] = x[i] + Ak[i, a] \* b[a];

}

Console.WriteLine();

for (int i = 0; i < n; i++)//выводим вектор решения на экран

{

x[i] = Convert.ToInt32(x[i]);

Console.WriteLine("X[{0}]= {1}", i + 1, x[i]);

}

}

Console.WriteLine();

}

static void kramer()//метод крамера

{

int n;

Console.Write("Введите размерность матрицы: ");

n = Convert.ToInt32(Console.ReadLine());

double[] b = new double[n]; // вектор правых частей

double[] x = new double[n]; // вектор решения

double[,] matrix = new double[n, n];// основная матрица

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

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

{

Console.Write("A[{0},{1}] = ", i, j);

matrix[i, j] = Convert.ToDouble(Console.ReadLine());

} // заполняем матрицу

double[,] An = new double[n, n];// клонируем основную матрицу

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

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

An[i, j] = matrix[i, j];

Console.WriteLine("Введите вектор правой части:");

for (int i = 0; i < n; i++) //заполняем вектор правой части

{

Console.Write("b[{0}] = ", i);

b[i] = Convert.ToDouble(Console.ReadLine());

}

Console.WriteLine("Исходная система уравнений: ");

printMatrix(matrix, n, b); // распечатываем матрицу;

Console.WriteLine();

double det = Determinant(An, n); // вычисляем определитель матрицы;

Console.WriteLine("Определитель равен = {0}", det);

Console.WriteLine();

if (SLAU\_kramer(n, matrix, b, x, An) == 1)

{

Console.WriteLine("Система не имеет решения");

return;

}

else

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

{

x[i] = Convert.ToInt32(x[i]);

Console.WriteLine("x[{0}] = {1}", i, x[i]);

}

Console.WriteLine();

}

static void gauss()//метод гаусса

{

//создание матрицы

int n, k;

double kof;

Console.Write("Введите размерность матрицы: ");

n = Convert.ToInt32(Console.ReadLine());

double[,] a = new double[n, n];// основная матрица

double[] x = new double[n];

double[] b = new double[n];

double[,] array = new double[n, n + 1];

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

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

{

Console.Write("A[{0},{1}] = ", i, j);

a[i, j] = Convert.ToDouble(Console.ReadLine());

} // заполняем матрицу

Console.WriteLine("Введите вектор правой части:");

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

{

Console.Write("b[{0}] = ", i);

b[i] = Convert.ToDouble(Console.ReadLine());

}

Console.WriteLine();

Console.WriteLine("Исходная система уравнений: ");

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

{

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

{

Console.Write("{0:0}\*X[{1}] ", a[i, j], j);

}

Console.Write("= ");

Console.WriteLine("{0:0}", b[i]);

}

Console.WriteLine();

for (k = 0; k < n - 1; k++)//приводим к треугольному виду

{

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

{

kof = -1 \* a[i, k] / a[k, k];

b[i] = b[i] + kof \* b[k];

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

{

a[i, j] = a[i, j] + a[k, j] \* kof;

}

}

}

Console.WriteLine("Матрица, приведенная к треугольному виду");

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

{

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

{

Console.Write("{0:0}\*X[{1}] ", a[i, j], j);

}

Console.Write("= ");

Console.WriteLine("{0:0}", b[i]);

}

Console.WriteLine();

//вычисление результата

x[n - 1] = b[n - 1] / a[n - 1, n - 1];

double sum = 0;

for (int z = n - 2; z >= 0; --z)

{

sum = 0;

for (int j = z + 1; j < n; j++)

{

sum = sum + a[z, j] \* x[j];

x[z] = (b[z] - sum) / a[z, z];

}

}

Console.WriteLine("Решение: ");

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

Console.WriteLine("Х[{0}] = {1}", i + 1, Convert.ToInt32(x[i]));

Console.WriteLine();

}

public static double[,] GetMinor(double[,] An, int a, int b, int n)//находим миноры

{

double[,] buf = new double[n-1,n-1];

for (int j = 0, q = 0; q < n-1; j++, q++)

for (int i = 0, p = 0; p < n-1; i++, p++)

{

if (i == a) i++;

if (j == b) j++;

buf[p, q] = An[i, j];

}

return buf;

}

static void T(int n, double[,] An,double[,] matrix)//транспонируем матрицу

{

Console.WriteLine("Транспонированная матрица: ");

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

{

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

{

An[i, j] = matrix[j, i];

Console.Write(An[i, j] + " \t ");

}

Console.WriteLine();

}

}

//метод вычисления определителя матрицы;

static private double Determinant(double[,] An, int n)

{

if (n == 1) // если матрица первого порядка, то ее определитель это единственный элемент этой матрицы;

return An[0, 0]; // возвращаем его;

if (Zero(An, n)) // если есть нулевой столбец, то определитель равен 0;

return 0;

int kol; // число перестановок строк;

sort(An, n, out kol); // сортируем матрицу по первым элементам строк;

for (int i = 1; i < n; ++i) // итерация методом Гаусса;

{

double index = An[i, 0] / An[0, 0];

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

An[i, j] -= index \* An[0, j];

} // где зануляется первый столбец с 1 по n - 1 строки;

return An[0, 0] \* Math.Pow(-1, kol) \* Determinant(subMatrix(An, n), n - 1);

// возвращаем определитель рекурсивно;

}

//метод для перестановки 2х линейных массивов;

static private void swapArray(double[,] An, int index, int lengthOfMatrix)

{

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

swapDouble(ref An[index, i], ref An[index + 1, i]);

}

//метод для перестановки 2х элементов;

static private void swapDouble(ref double Buffer1, ref double Buffer2)

{

double Tmp = Buffer1;

Buffer1 = Buffer2;

Buffer2 = Tmp;

}

// сортировка матрицы по первым элементам строк

static private void sort(double[,] An, int n, out int kol)

{

kol = new int();

bool flagOfSwapping = true;

int numberOfIteration = new int();

while (flagOfSwapping) // цикл выполняется пока есть хотя бы одна перестановка в ходе итерации;

{

flagOfSwapping = false;

for (int i = 0; i < n - 1 - numberOfIteration; ++i)

if (An[i, 0] < An[i + 1, 0])

{

swapArray(An, i, n);

flagOfSwapping = true;

++kol;

}

numberOfIteration++;

}

}

//метод вывода матрицы на экран;

static private void printMatrix(double[,] matrix, int n, double[] b)

{

Console.WriteLine();

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

{

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

Console.Write("{0}\*X[{1}] ", matrix[i, j],j);

Console.WriteLine("= {0}", b[i]);

}

}

//метод выделения подматрицы размером n - 1;

static private double[,] subMatrix(double[,] matrix, int n)

{

double[,] subMatrix = new double[n - 1, n - 1];

for (int i = 1; i < n; ++i) // выделяется подматрица со строк [1, n);

for (int j = 1; j < n; ++j) // .. cтолбцов [1; n);

subMatrix[i - 1, j - 1] = matrix[i, j];

return subMatrix;

}

//метод проверки первого столбца на равенство нулевому;

static private bool Zero(double[,] matrix, int n)

{

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

if (matrix[i, 0] != 0)

return false;

return true;

}

//метод вычисления методом крамера

static int SLAU\_kramer(int n, double[,] matrix, double[] b, double[] x, double[,] An)

{

double det1 = Determinant(An, n);

if (det1 == 0) return 1;

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

{

equal(n, An, matrix);

change(n, i, An, b);

x[i] = Determinant(An, n) / det1;

}

return 0;

}

//клонирование матрицы

static void equal(int n, double[,] An, double[,] B)

{

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

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

An[i, j] = B[i, j];

}

//метод замены столбца вектором правой части

static void change(int n, int N, double[,] An, double[] b)

{

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

An[i, N] = b[i];

}

}

}

**7 Описание программы**

Программа создавалась для решения систем линейных уравнений методами обратной матрицы, Крамера и Гаусса.

Логика работы программы состоит в дискретной реализации всех действий отдельными методами.

Главный метод состоит из оператора switch, который выбирает из списка значения меток case. В каждой из меток case происходит вызов одного из «вспомогательных» операторов.

В методе matrix() происходит решение системы уравнений методом обратной матрицы:

double det = Determinant(An, n); // вычисляем определитель матрицы;

if (det == 0)

Console.WriteLine("Матрица вырождена");

else

{

for (int i = 0; i < n; i++)//создаем матрицу миноров

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

{

double A = Determinant(GetMinor(An, i, j, n), n - 1);

Ak[i, j] = Math.Pow(-1, i + j) \* A;

Ak[i, j] = Ak[i, j] / det;

}

for (int i = 0; i < n; i++)//находим вектор решения

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

{

x[i] = x[i] + Ak[i, a] \* b[a];

}

В методе kramer() происходит решение системы уравнений методом Крамера:

double det = Determinant(An, n); // вычисляем определитель матрицы;

if (SLAU\_kramer(n, matrix, b, x, An) == 1)//если определитель равен 0 то

{

Console.WriteLine("Система не имеет решения");

return;

}

else// иначе решаем методом крамера

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

{

x[i] = Convert.ToInt32(x[i]);

Console.WriteLine("x[{0}] = {1}", i, x[i]);

}

В методе gauss() происходит решение системы уравнений методом Гаусса:

for (k = 0; k < n - 1; k++)//приводим к треугольному виду

{

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

{

kof = -1 \* a[i, k] / a[k, k];

b[i] = b[i] + kof \* b[k];

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

{

a[i, j] = a[i, j] + a[k, j] \* kof;

}

}

}

//вычисление результата

x[n - 1] = b[n - 1] / a[n - 1, n - 1];

double sum = 0;

for (int z = n - 2; z >= 0; --z)

{

sum = 0;

for (int j = z + 1; j < n; j++)

{

sum = sum + a[z, j] \* x[j];

x[z] = (b[z] - sum) / a[z, z];

}

}

8 Результат тестирования программы

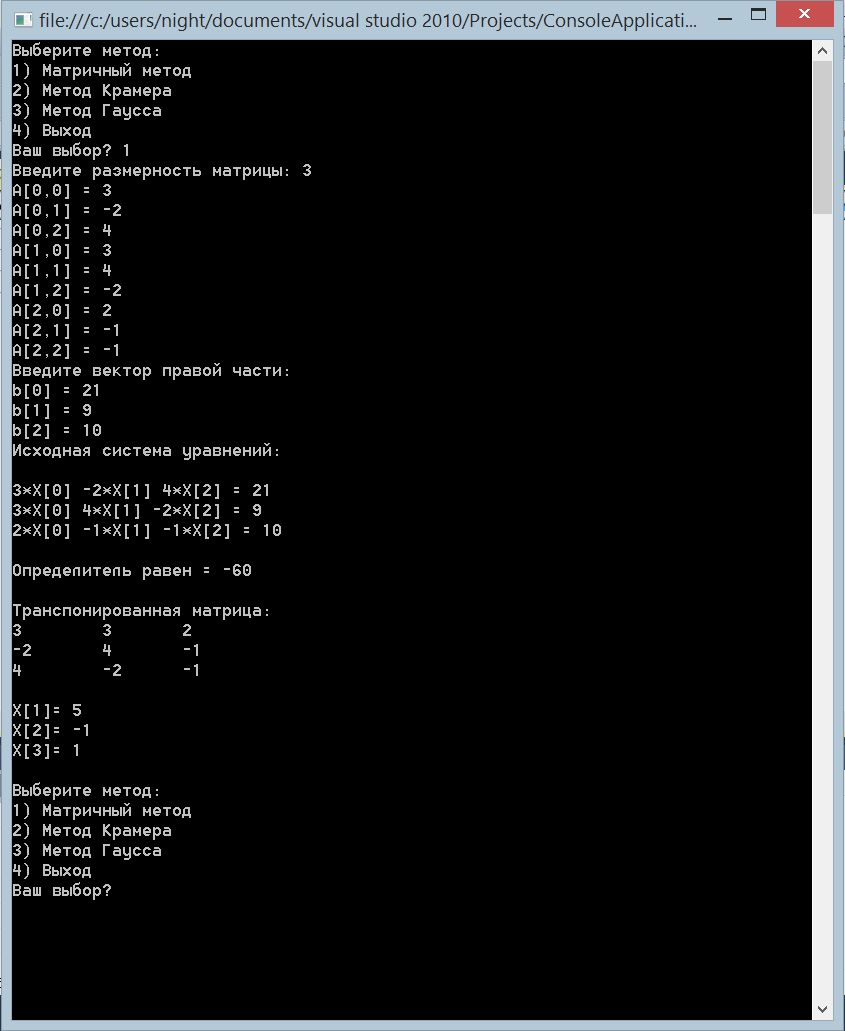
Результат решения системы методом обратной матрицы представлен на рисунке 1. 

Рисунок 1.

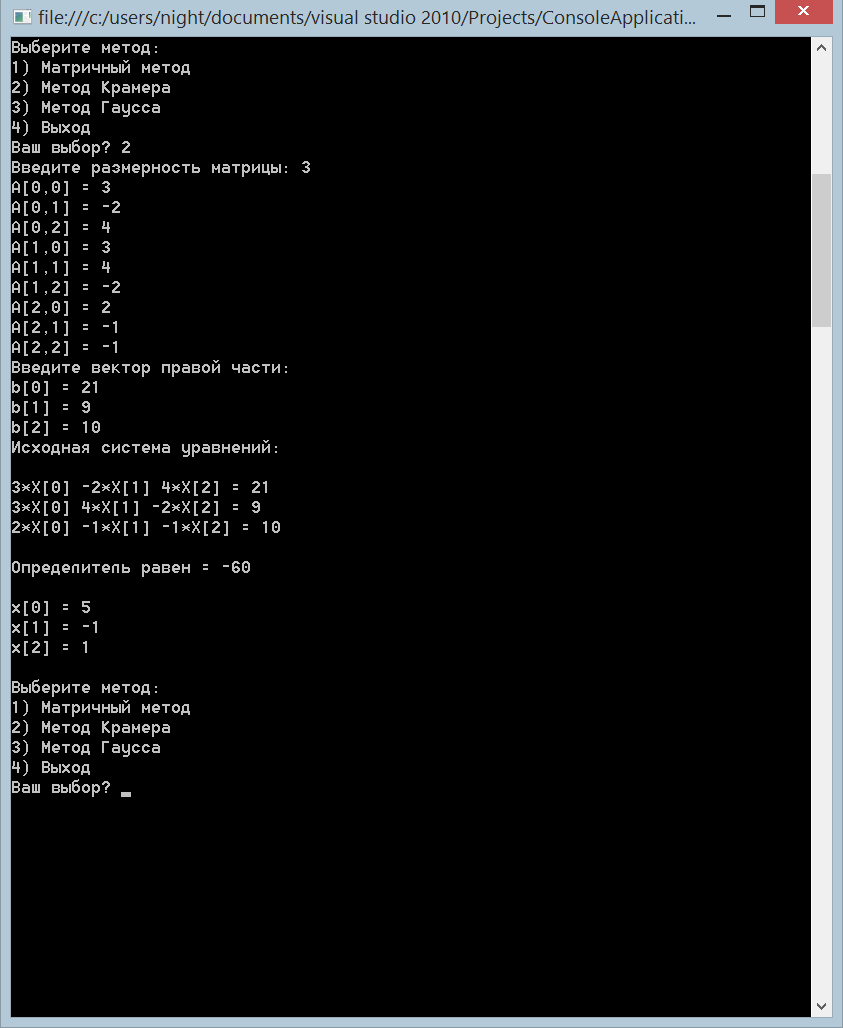
Результат решения системы методом Крамера представлен на рисунке 2. 

Рисунок 2.

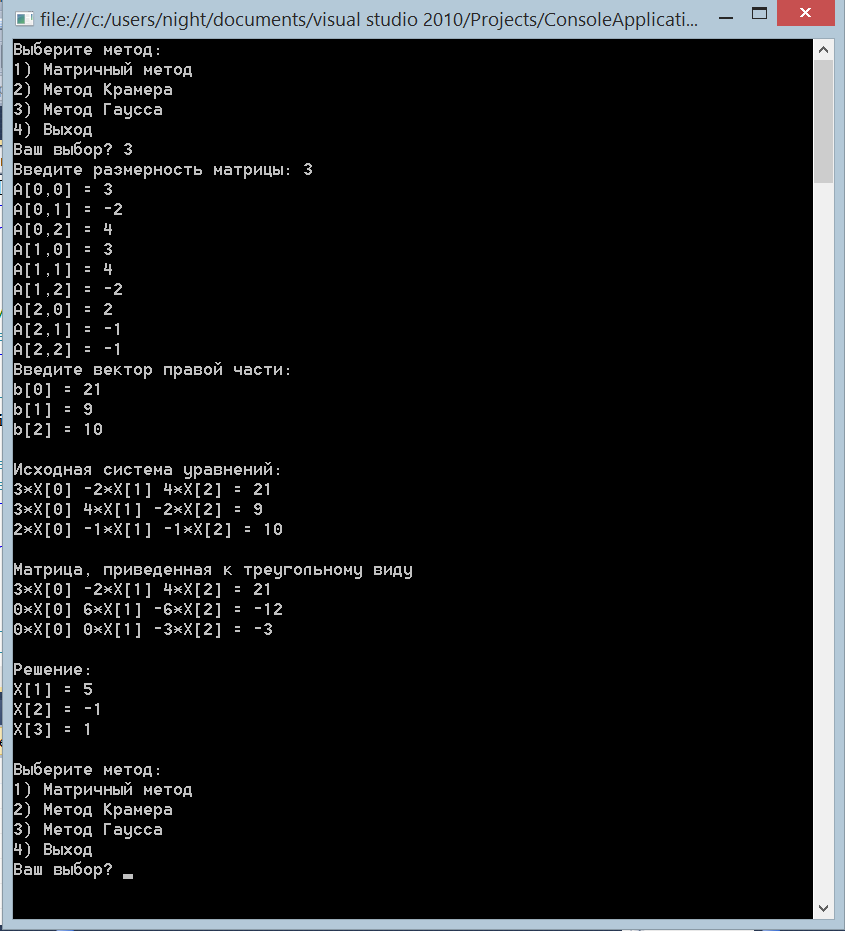
Результат решения системы методом Гаусса представлен на рисунке 3. 

Рисунок 3.

# Заключение

В данной работе решена задача решения систем линейных уравнений.

В процессе создания работы разработан алгоритм решения поставленной задачи. По этому алгоритму на языке C# составлена и отлажена программа.

В ходе тестирования были получены результаты работы программы методом обратной матрицы, Крамера и Гаусса.

Программа является полностью работоспособной, что подтверждается результатами её тестирования.

Достоинства: решает различные системы уравнений разными методами.

Недостатки: одновременно вычисляет только одним методом, при вычислении другим методом необходимо каждый раз заново вводить матрицу.

# Список используемых источников

1. Павловская Т.А. С# Программирование на языке высокого уровня Учебник для вузов – СПб.: Питер. 2014 – 432с.
2. http://www.mathprofi.ru/pravilo\_kramera\_matrichnyi\_metod.html
3. http://www.exponenta.ru/educat/class/courses/la/theme3/theory.asp#4
4. http://www.exponenta.ru/educat/class/courses/la/theme2/theory.asp#2
5. https://ru.wikipedia.org/wiki/Матричный\_метод
6. https://ru.wikipedia.org/wiki/Метод\_Гаусса\_—\_Жордана
7. http://www.cleverstudents.ru/systems/solving\_systems\_Gauss\_method.html