**Правительство Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

**Отчет к домашней работе №1**

**По дисциплине**

**“Компьютерный практикум по алгебре в среде Mathematica”**

Работу выполнил

Студент группы БПИ-197 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Синенко

подпись, дата

Работу проверил \_\_\_\_\_\_\_\_\_\_\_\_\_ И. Ю. Захарьев

подпись, дата

Москва 2020

Содержание

[Постановка задачи 3](#_Toc33960317)

[Уточнение задачи 3](#_Toc33960318)

[Математическое описание 4](#_Toc33960319)

[Реализация программы 5](#_Toc33960320)

[Тестирование программы 6](#_Toc33960321)

[Использованная литература 9](#_Toc33960322)

[Приложение 10](#_Toc33960323)

# Постановка задачи

1. Необходимо написать программу на одном из предложенных языков, которая по заданной матрице будет находить каноническую матрицу посредством метода Гаусса.
2. Вывести матрицу в консоль в преобразованном виде.

# Уточнение задачи

1. Написать программу, на C# в интегрированной среде разработке Visual Studio, которая будет принимать на вход матрицу произвольных размеров, обрабатывать некорректно введённые (то есть введённые не по алгоритму ввода[[1]](#footnote-1), не числа, противоречащие определению матрица[[2]](#footnote-2)). При некорректном вводе пользователь получает сообщение о том, что его входные данные неверны, и программа даёт возможность повторного ввода.
2. На основе введённой матрицы, применяя метод Гаусса, программа будет преобразовывать её в канонический вид.
3. Программа будет выводить на консоль матрицу в преобразованном виде.
4. На экран выведется решение3 введенной системы линейных уравнений.

# Математическое описание

Метод Гаусса применяется для решения СЛАУ. Этот метод используется для решения совместных систем, то есть имеющих хотя бы единственное решение [1, стр.22].

Сам метод состоит из двух этапов:

1)Прямой ход метода Гаусса заключается в преобразовании исходной матрицы в ступенчатую матрицу вида (C|D), где C – преобразованная матрица системы, а D – преобразованная матрица свободных членов системы.

2)Обратный ход метода Гаусса, его цель в приведении матрицу (C|D) к виду (E|F), где E – единичная матрица.

Предъявим сам алгоритм преобразования матрицы в канонический вид:

0-ый шаг: двигаемся из левого верхнего угла.

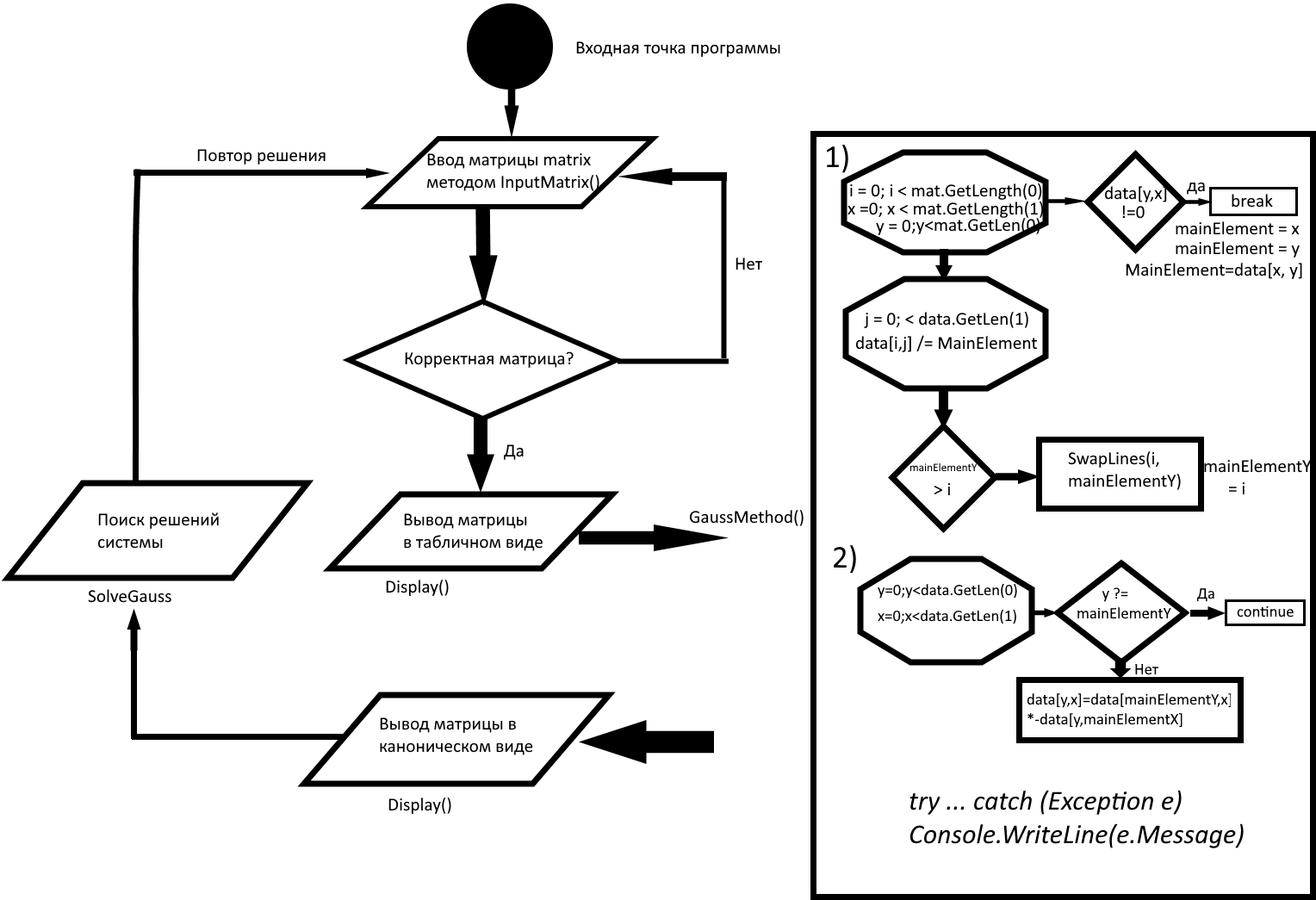
1-ый шаг: если текущий элемент равен 0, то переходим к шагу 2, иначе элемент объявляется ведущим. Теперь прибавляем строку с ведущим элементом к остальным так, чтобы все элементы, расположенные выше и ниже обратились в ноль. А именно, пусть ведущий элемент [А]***i j****,* тогда для k-ой строки (k!=i) берем λ= **-**[А]***k j*** */* [А]***i j***.Так называемое элементарное преобразование II типа. После i-ю строку умножаем на [А]***i j***!=0. Так называемое элементарное преобразование [[3]](#footnote-3)III типа. Выбираем новый текущий элемент, смещаясь в матрице на 1 столбец вправо и на 1 строку вниз и переходим к следующему шагу, повторяя шаг 1.

2-ой шаг: если текущий элемент равен нулю просматриваем все элементы под ним. Если все элементы равны 0, то переходим к шагу 3. Если в k-ой строке есть элементы не равные нулю, то меняем текущую и k-ю строку местами. Так называемое элементарное преобразование I типа и переходим к шагу 1.

3-ий шаг: если текущий элемент и все элементы под ним равны нулю, то меняем текущий столбец, смещаясь на 1 вправо. Если это возможно переходим к шагу 1, если нет, то конец алгоритма.

Реализация программы

* Для работы с матрицами создан специальный класс **Matrix**.
* Для считывания матрицы с консоли, проверки на корректность введённых данных служит метод Matrix **InputMatrix**(), который возвращает матрицу, прошедшую проверки.В случае некорректных введенных данных в консоль выводится сообщение об ошибке с описанием требований к вводу при помощи метода Console.Writeline(), оповещающее об этом пользователя, и ввод повторяется.
* Для перевода матрицы в канонический вид используется метод void **GaussMethod**(), в котором по заданному в математическом описании алгоритму происходит преобразование матрицы. В случае необходимости, строки матрицы меняются местами с помощью метода void **SwapLines**(int i1, int i2).
* Для вывода на консоль матрицы, используется метод void **Display**().
* При помощи методаvoid **SolveGauss**() идет поиск свободных переменных для СЛАУ, если она имеет бесконечное количество решений. Метод возвращает список свободных переменных, в случае их наличия, либо пустой список.
* Для проверки, введена ли нулевая матрица используется метод bool **isZeroMatrix**().
* Печать результата решения СЛАУ, имеющая единственное решение, посредством метода void **DisplayParticularSolution**().
* В методеvoid **Main**(string[] args) реализован последовательный вызов методов, а также повтор решения.



Блок-схема работы алгоритма Гаусса

# Тестирование программы

Рассмотрим разные случаи ввода пользователем, демонстрируя таким образом работу метода **InputMatrix()** :

* Введены некорректные значения в строки, метод не допускает наличия буквенных символов (рис.1)
* Неверный формат ввода, то есть несоответствующий алгоритму ввода (рис.2)
* Введенные данные противоречат определению матрицы и алгоритму ввода (рис.3)
* Корректно введенная матрица. Значения на выводе округляются до трёх знаков после запятой. В памяти компьютера они хранятся в исходном виде. (рис.4)

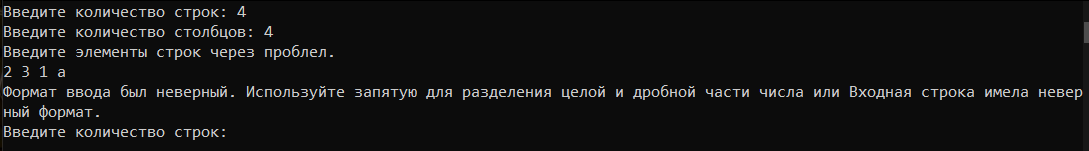


Рисунок 1 – демонстрация ввода неверных данных в программу.

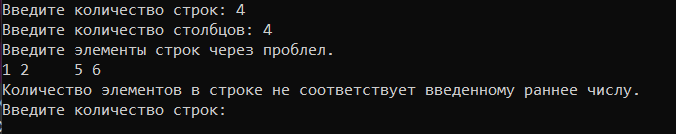


Рисунок 2 – демонстрация ввода неверных данных в программу.

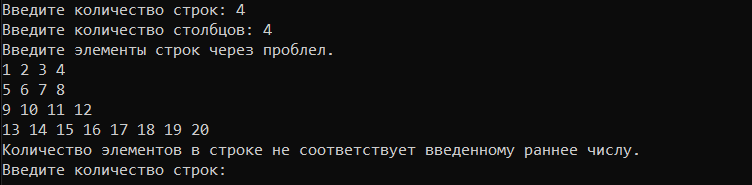


Рисунок 3 – демонстрация ввода неверных данных в программу.

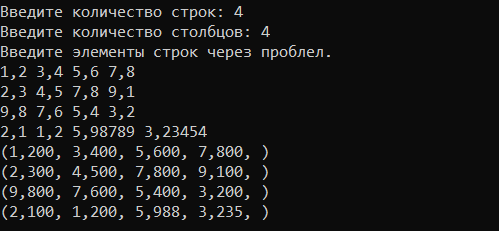


Рисунок 4 – демонстрация ввода корректных данных в программу.

При корректном вводе матрица при помощи метода Gauss будет приведена к каноническому виду (рис. 5):

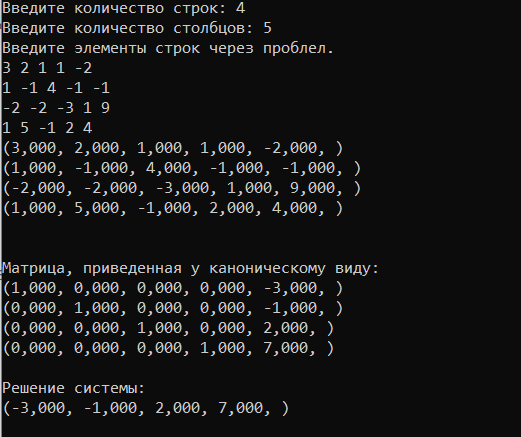


Рисунок 5 – демонстрация работы метода Гаусса в программе.

Корректность работы Gauss проверим при помощи метода RowReduce в Wolfram Mathematica (рис. 6):

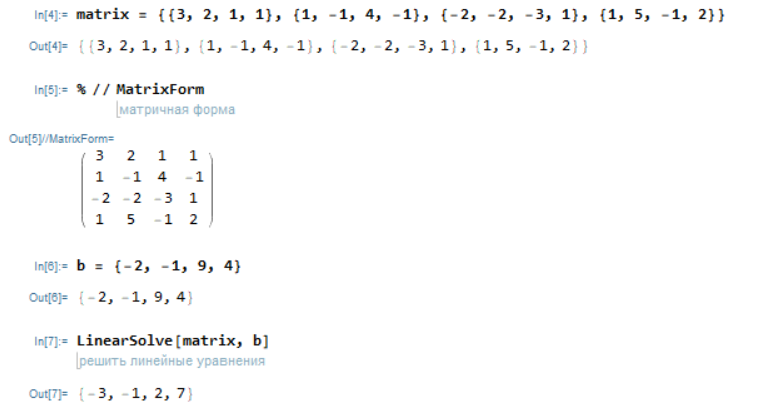


Рисунок 6 – демонстрация работы метода RowReduce в Wolfram Mathematica.

Как видно из Рисунка 6, ответы совпадают.

2)Решим СЛАУ (рис. 7) с единственным решением методом Гаусса в написанной программе и сравним полученный ответ с результатом работы метода Reduce в Wolfram Mathematica:

* Исходная СЛАУ:

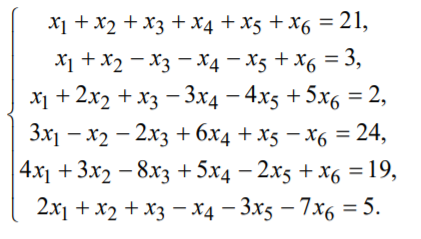


Рисунок 7 – рассматриваемая СЛАУ.

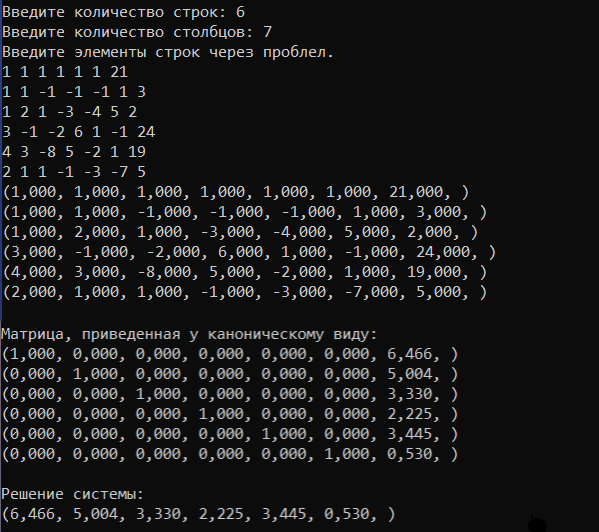
* Решение СЛАУ в написанной программе (рис. 8): 

Рисунок 8 – нахождение решения СЛАУ в программе.

* Решение СЛАУ в Wolfram Mathematica (рис. 9):

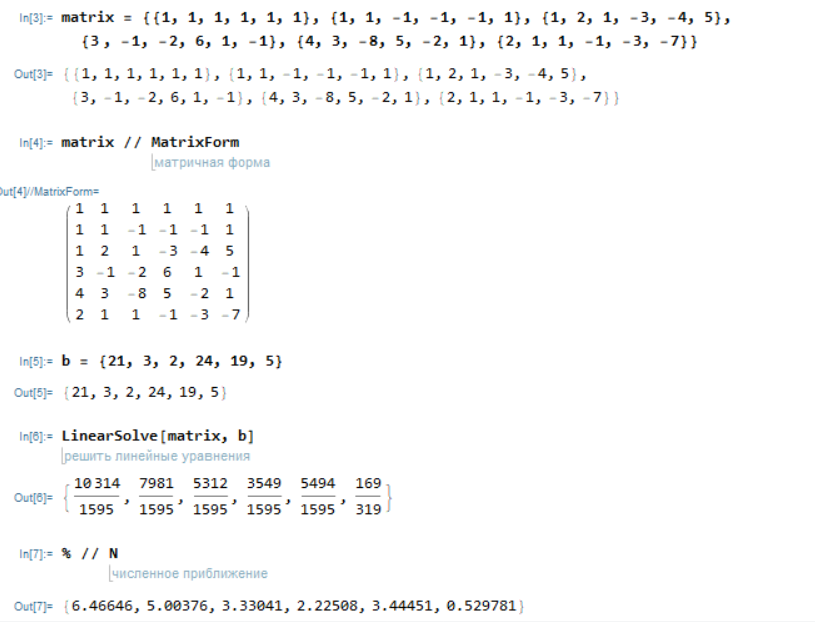


Рисунок 9 – нахождение решения СЛАУ в Wolfram Mathematica.

Ответы совпадают!

# Использованная литература

1. Введение в алгебру: Часть 1, Основы алгебры (второе издание) / А.И. Кострикин, изд-во Московского центра непрерывного математического образования, 2011, 271 страница.
2. Линейная алгебра: учебное пособие 2-ое издание / Г.С. Шевцов, исп. и доп. – М.:Гардарики , 1999, 360 страниц.

# Приложение

1. **using** System;
3. **namespace** GausseMethod
4. {
5. **class Program**
6. {
7. *// Входная точка программы*
8. **static** **void** Main(**string**[] args)
9. {
10. **do**
11. {
12. Matrix matrix = InputMatrix();
13. matrix.Display();
14. Console.ReadLine();
15. **matrix.GaussMethod();**
16. Console.WriteLine("Матрица, приведенная у каноническому виду:");
17. matrix.Display();
18. matrix.SolveGauss();
19. } **while** (Console.ReadKey(**true**).Key != ConsoleKey.Escape);
20. **}**
21. *// Метод для ввода матрицы*
22. **static** Matrix InputMatrix()
23. {
24. **int** i, j;
25. **Console.Write("Введите количество строк: ");**
26. **if** (!**int**.TryParse(Console.ReadLine(), **out** i) || i < 2)
27. {
28. Console.WriteLine("Неверный ввод. " +
29. "Количество строк это целое положительное число больше 2.");
30. **return InputMatrix();**
31. }
32. Console.Write("Введите количество столбцов: ");
33. **if** (!**int**.TryParse(Console.ReadLine(), **out** j) || j < 2)
34. {
35. **Console.WriteLine("Неверный ввод. " +**
36. "Количество столбцов это целое положительное число больше 2.");
37. **return** InputMatrix();
38. }
40. **Matrix matrix = new Matrix(i, j);**
41. Console.WriteLine("Введите элементы строк через проблел.");
42. **for** (**int** y = 0; y < i; y++)
43. {
44. **string**[] inp = Console.ReadLine().Trim().Split(' ');
45. **if (inp.Length != j)**
46. {
47. Console.WriteLine("Количество элементов в строке не соответствует введенному раннее числу.");
48. **return** InputMatrix();
49. }
50. **try**
51. {
52. **for** (**int** x = 0; x < j; x++)
53. {
54. matrix[y, x] = **double**.Parse(inp[x]);
55. **}**
56. }
57. **catch** (Exception e)
58. {
59. Console.WriteLine("Формат ввода был неверный. Используйте запятую для разделения целой и дробной части числа или " + e.Message);
60. **return InputMatrix();**
61. }
62. }
63. **return** matrix;
64. }
65. **}**
66. *// Класс, характеризующий матрицу*
67. **class** Matrix
68. {
69. *// Двухмерный массив для хранения матрицы*
70. **double[,] data;**
71. **public** **double** **this** [**int** i, **int** j]
72. {
73. **get** { **return** data[i, j]; }
74. **set** { data[i, j] = **value**; }
75. **}**
76. *// Свойство, которое считает ранг матрицы*
77. **public** **int** Rank
78. {
79. **get**
80. **{**
81. **int** count = 0;
82. **for** (**int** i = 0; i < data.GetLength(0); i++)
83. {
84. **bool** flag = **false**;
85. **for (int j = 0; j < data.GetLength(1); j++)**
86. {
87. **if** (data[i, j] != 0)
88. flag = **true**;
89. }
90. **if (flag) count++;**
91. }
92. **return** count;
93. }
94. }
95. ***// Свойство, которое считает количество переменных***
96. **public** **int** VariablesCount
97. {
98. **get**
99. {
100. **return data.GetLength(1) - 1;**
101. }
102. }
103. **public** Matrix(**int** i, **int** j)
104. {
105. **data = new double[i, j];**
106. }
107. *// Матод Гаусса для приведения матрицы к каноническому виду*
108. **public** **void** GaussMethod()
109. {
110. **try**
111. {
112. **for** (**int** i = 0; i < data.GetLength(0); i++)
113. {
114. *// Индексы главного элемента*
115. **int mainElementX = 0, mainElementY = 0;**
116. **bool** isFound = **false**;
117. **for** (**int** x = 0; x < data.GetLength(1); x++)
118. {
119. **for** (**int** y = i; y < data.GetLength(0); y++)
120. **{**
121. **if** (data[y, x] != 0)
122. {
123. mainElementX = x;
124. mainElementY = y;
125. **isFound = true;**
126. **break**;
127. }
128. }
129. **if** (isFound) **break**;
130. **}**
132. **double** MainElement = data[mainElementY, mainElementX];
133. **for** (**int** j = 0; j < data.GetLength(1); j++)
134. {
135. **data[i, j] = data[i, j] / MainElement;**
136. }
138. *// Если главный элемент ниже рассматриваемой строки, то они меняются местами в матрице*
139. **if** (mainElementY > i) SwapLines(i, mainElementY);
140. **mainElementY = i;**
142. **for** (**int** y = 0; y < data.GetLength(0); y++)
143. {
144. **if** (y == mainElementY)
145. **continue;**
146. **double** tempValue = data[y, mainElementX];
147. **for** (**int** x = 0; x < data.GetLength(1); x++)
148. {
149. data[y, x] += data[mainElementY, x]\* -tempValue;
150. **}**
151. }
152. }
153. }
154. **catch** (Exception e)
155. **{**
156. Console.WriteLine(e.Message);
157. Console.WriteLine(e.Source);
158. Console.WriteLine(e.StackTrace);
159. }
160. **}**
161. *// Метод решающий систему, если это возможно*
162. **public** **void** SolveGauss()
163. {
164. **if** (isZeroMatrix())
165. **{**
166. Console.WriteLine("Ввдена нулевая матрица, которая образует СЛАУ.");
167. **return**;
168. }
169. **int** rank = Rank;
170. **Console.WriteLine("Решение системы: ");**
171. **if** (rank == VariablesCount)
172. DisplayParticularSolution();
173. **if** (rank < VariablesCount)
174. {
175. **Console.WriteLine("Частное решение: ");**
176. DisplayParticularSolution();
177. Console.WriteLine("Набор ФСР: ");
178. DisplayFSS(rank);
179. }
180. **if (rank > VariablesCount)**
181. Console.WriteLine("Система несовместна.");
182. }
183. */// <summary>*
184. */// Метод для вывода частного решения (решения системы, когда ранг равен количеству переменных)*
185. ***/// </summary>***
186. **private** **void** DisplayParticularSolution()
187. {
188. Console.Write("(");
189. **for** (**int** i = 0; i < data.GetLength(0); i++)
190. **{**
191. Console.Write("{0:0.000}, ", data[i, data.GetLength(1) - 1]);
192. }
193. Console.WriteLine(")");
194. }
195. ***/// <summary>***
196. */// Метод для вывода ФСР*
197. */// </summary>*
198. */// <param name="rank">Ранг матрицы</param>*
199. **private** **void** DisplayFSS(**int** rank)
200. **{**
201. Console.WriteLine("Так как нашлись свободные переменные, решение будет выражаться через них.");
202. }
203. *// Метод, который меняет строки местами*
204. **public** **void** SwapLines(**int** i1, **int** i2)
205. **{**
206. **for** (**int** j = 0; j < data.GetLength(1); j++)
207. {
208. **double** temp = data[i1, j];
209. data[i1, j] = data[i2, j];
210. **data[i2, j] = temp;**
211. }
212. }
213. *// Метод, проверяющий нулевая ли матрица*
214. **public** **bool** isZeroMatrix()
215. **{**
216. **bool** flag = **true**;
217. **for** (**int** i = 0; i < data.GetLength(0); i++)
218. {
219. **for** (**int** j = 0; j < data.GetLength(1); j++)
220. **{**
221. **if** (data[i, j] != 0) flag = **false**;
222. }
223. }
224. **return** flag;
225. **}**
226. *// Метод выводящий матрицу на экран*
227. **public** **void** Display()
228. {
229. **for** (**int** i = 0; i < data.GetLength(0); i++)
230. **{**
231. Console.Write("(");
232. **for** (**int** j = 0; j < data.GetLength(1); j++)
233. {
234. Console.Write("{0:0.000}, ", data[i, j]);
235. **}**
236. Console.Write(")**\n**");
237. }
238. Console.Write(Environment.NewLine);
239. }
240. **}**
241. }

1. Алгоритм ввода: Пользователь вводит количество строк, затем количество столбцов. Далее пользователю необходимо вводить матрицу построчно через пробел, после введения первой строки перейти на следующую при помощи клавиши Enter. Ввод матрицы закончится в тот момент, когда будут введены все строки системы (количество строк известно из ввода пользователя). [↑](#footnote-ref-1)
2. Коэффициенты при неизвестных составляют прямоугольную таблицу называемую *матрицей* размера m x n (m x n-матрицей или квадратной матрицей порядка n при m = n) и сокращенно обозначаемую символом (*aij*) или простой буквой *A*. [1, стр.20].

   3 *Решение системы* – это упорядоченный набор (x1, x2, … , xn) из n чисел, при подстановке которых в уравнение системы вместо соответствующих неизвестных каждое уравнение системы превращается в тождество [2, стр.9]. [↑](#footnote-ref-2)
3. Над уравнениями системы обычно приходится проводить следующие элементарные преобразования: [2, стр.10]:

   Перемена двух строк матрицы (I тип)

   Прибавление к какой-либо строке матрицы другой её строки, умноженной на некоторое число (II тип)

   Умножение некоторой строки на отличное от нуля число (III тип) [↑](#footnote-ref-3)