Федеральное государственное автономное образовательное учреждение высшего образования «**Уральский федеральный университет** имени первого Президента России Б. Н. Ельцина»

Отчет по лабораторной работе №1.2

дисциплины «Алгоритмы и анализ сложности»

Студент: Волков Илья Евгеньевич  
Группа: РИ-200004 (АТ-08)  
Преподаватель: Рыбкина Мария Николаевна

2022 год.

Оглавление

[Задание 3](#_Toc101780260)

[Теоретическая часть 3](#_Toc101780261)

[Инструкция пользователя 4](#_Toc101780262)

[Инструкция программиста 4](#_Toc101780263)

[Тестирование 5](#_Toc101780264)

[Листинги 10](#_Toc101780265)

[Вывод 14](#_Toc101780266)

# Задание

Лабораторная работа №1.2

Алгебраический алгоритм вычисления обратной матрицы:

1. Написать функцию вычисления определителя квадратной матрицы, раскрывая определитель по строке.  
2. Написать функцию вычисления обратной матрицы, с использованием алгебраических дополнений.  
3. Написать функцию умножения матрицы на вектор.  
4. Решить квадратную систему линейных уравнений методом обратной матрицы.

Функция выдает ответы:  
– решение в случае его единственности;  
– нет решений или решений бесконечно много.

# Теоретическая часть

Есть несколько способов решения данной задачи:

1. Найти готовую библиотеку. Я не выбрал этот способ, потому что скорее всего так выполнять задание нельзя.

2. Найти готовое решение и, основываясь на нем, написать свое.

3. Полностью написать все с нуля самостоятельно.

Я выбрал третий способ. Мне знакома теория матриц, потому искать готовое решение не было смысла. Однако некоторую теорию все же пришлось поискать в интернете.

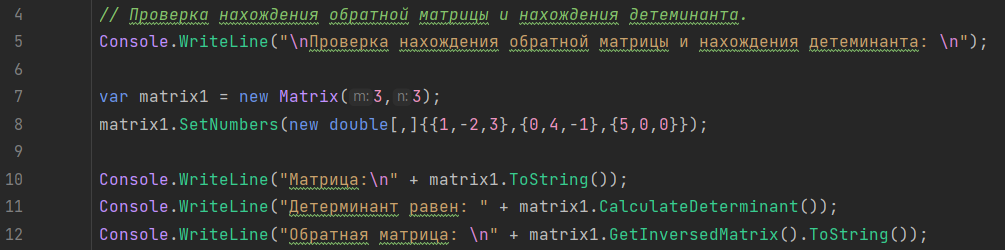
# Инструкция пользователя

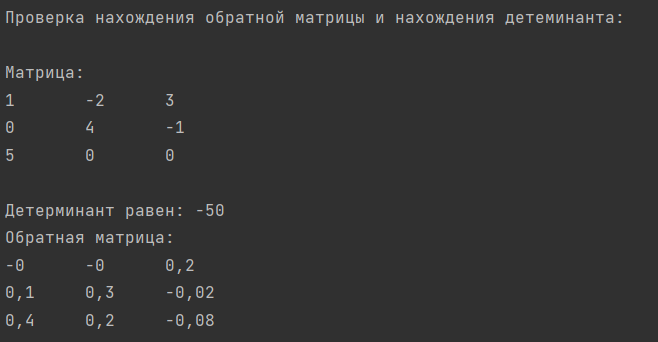
Для запуска софта используется файл InverseMatrix.exe.  
Откроется окно с примерами работы программы. В ПО нет пользовательского интерфейса, т.е. кроме как кодом управлять им нельзя.

# Инструкция программиста

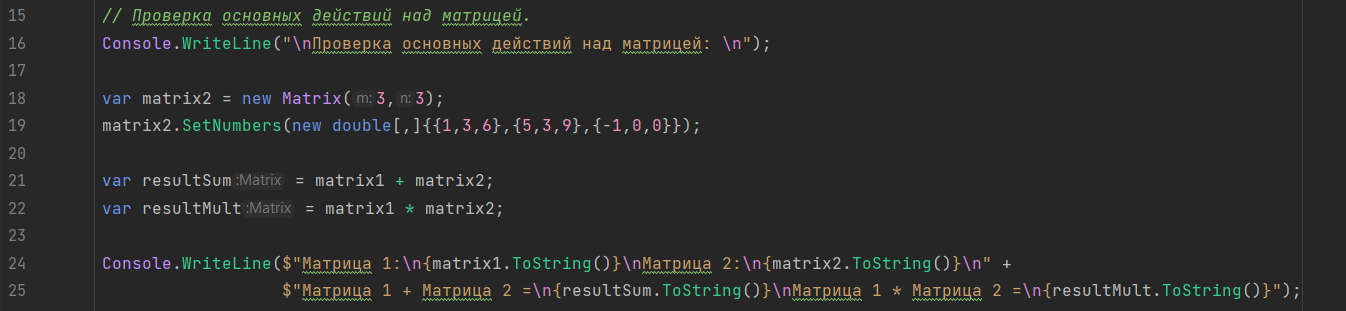
Для запуска софта используется файл InverseMatrix.exe.   
Для запуска решения используется файл InverseMatrix.sln (или InverseMatrix.csproj). Класс матрицы реализован в файле Matrix.cs. Работать с классом можно из любой точки программы, например, из Program.cs. Примеры работы с классом представлены в Program.cs.

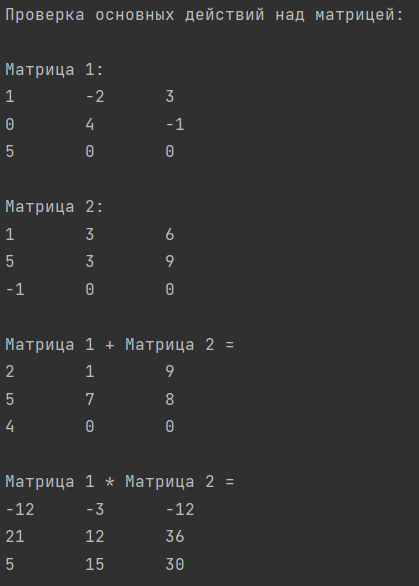
# Тестирование

1.

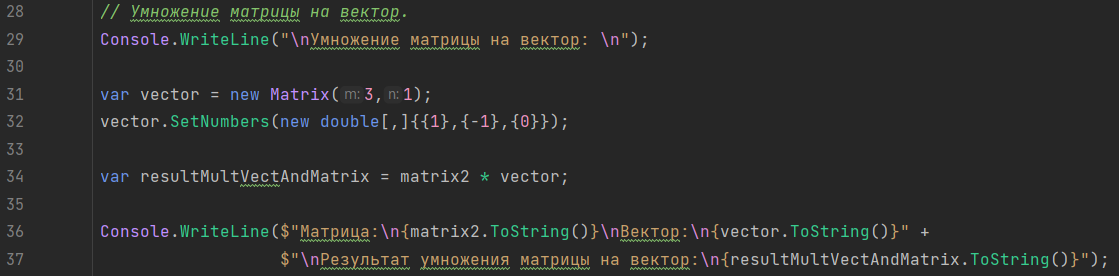
Вывод:

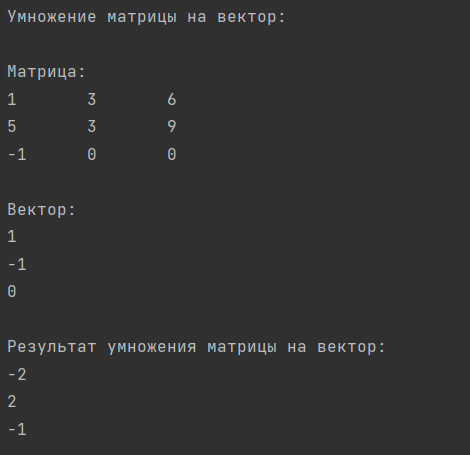
2.

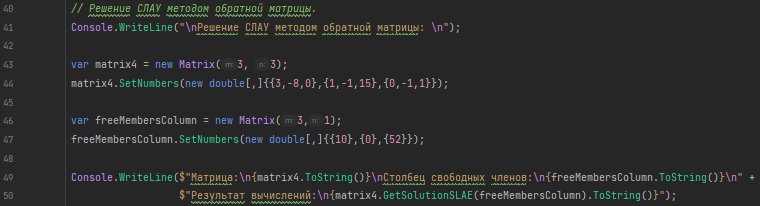


Вывод:

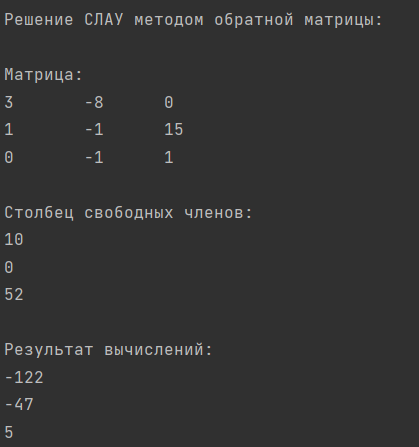
3.



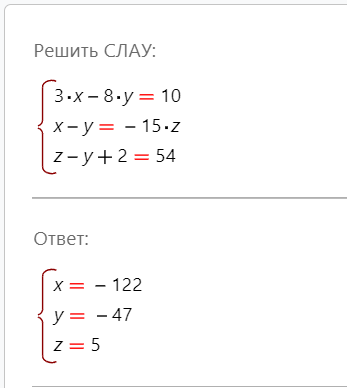
Вывод:  


4. 

Вывод:

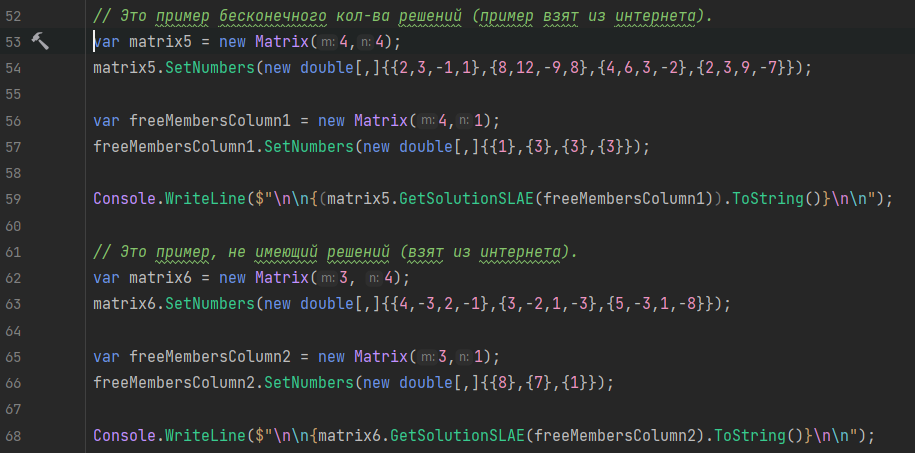


Проверка решения:



Ресурс: https://mathforyou.net/online/equation/sle/inverse/

5.



Вывод: (на каждый пример)



# Листинги

Листинг класса Matrix.cs:

using System.Text;  
  
namespace InverseMatrix.classes;  
  
public class Matrix  
{  
 private double[,] numbers;  
  
 private int m;  
 private int n;  
   
 private bool IsSquareMatrix => this.m == this.n;  
  
 public Matrix(int m, int n)  
 {  
 this.m = m;  
 this.n = n;  
 this.numbers = new double[m, n];  
 this.MakeActionOverEveryNumber((i, j) => this.numbers[i, j] = 0);  
 }  
   
 public double this[int x, int y]   
 {  
 get => this.numbers[x, y];  
 set => this.numbers[x, y] = value;  
 }  
   
 public void SetNumbers(double[,] numbers)  
 {  
 this.numbers = numbers;  
 }  
  
 public static Matrix operator \*(Matrix matrix, double value)  
 {  
 var result = new Matrix(matrix.m, matrix.n);  
   
 result.MakeActionOverEveryNumber((i,j) => result[i,j] = matrix[i,j] \* value);  
 return result;  
 }  
  
 public static Matrix operator \*(Matrix matrix1, Matrix matrix2)  
 {  
 if (matrix1.n != matrix2.m)  
 throw new ArgumentException("Не перемножаемные матрицы!");  
   
 var result = new Matrix(matrix1.m, matrix2.n);  
   
 result.MakeActionOverEveryNumber((i, j) =>  
 {  
 for (var h = 0; h < matrix1.n; h++)  
 {  
 result[i, j] += matrix1[i, h] \* matrix2[h, j];  
 }  
 });  
  
 return result;  
 }  
  
 public static Matrix operator +(Matrix matrix1, Matrix matrix2)  
 {  
 if (matrix1.m != matrix2.m || matrix1.n != matrix2.n)  
 throw new ArgumentException("Складываться могут только матрицы одинаковых размеров!");  
  
 var result = new Matrix(matrix1.m, matrix2.n);  
   
 result.MakeActionOverEveryNumber((i,j)=>result[i,j] = matrix1[i,j] + matrix2[i,j]);  
 return result;  
 }  
  
 public string ToString()  
 {  
 StringBuilder result = new StringBuilder();  
   
 for(var i = 0; i < m; i++)  
 {  
 for(var j = 0; j < n; j++)  
 {  
 result.Append(Math.Round(this[i, j], 5) + "\t");  
 }  
  
 result.Append('\n');  
 }  
  
 return result.ToString();  
 }  
  
 public double CalculateDeterminant()  
 {  
 if (!this.IsSquareMatrix)  
 throw new ArgumentException("Не существует детерминанта не квадратной матрицы!");  
  
 *// База* if (n == 2)  
 {  
 return this[0, 0] \* this[1, 1] - this[0, 1] \* this[1, 0];  
 }  
   
 double result = 0;  
  
 *// Рекурсия* for (var j = 0; j < n; j++)  
 {  
 result += (j % 2 == 0 ? 1 : -1) \* this[0, j] \*  
 this.CalculateMinor(0, j);  
 }  
  
 return result;  
 }  
  
 public Matrix GetInversedMatrix()  
 {  
 if (!this.IsSquareMatrix)  
 throw new ArgumentException("Обратная матрица есть только у квадратных матриц!");  
  
 var det = CalculateDeterminant();  
  
 if (det == 0)  
 throw new ArgumentException("Обратная матрица есть только у невырожденных матриц!");  
   
 var result = new Matrix(m, n);  
  
 this.MakeActionOverEveryNumber((i, j) => result[i, j] = ((i + j) % 2 == 0 ? 1 : -1) \* CalculateMinor(i, j) / det);  
  
 result = result.CreateTransposeMatrix();  
 return result;  
 }  
  
 public Matrix GetSolutionSLAE(Matrix freeMembersVector)  
 {  
 if (freeMembersVector.m != m && freeMembersVector.n != 1)  
 throw new ArgumentException("Матрица свободных членов должна быть вектором!");  
  
 if (!this.isHasSolutions(freeMembersVector))  
 throw new ArgumentException("Система не имеет решений или решений бесконечно много!");  
  
 return this.GetInversedMatrix() \* freeMembersVector;  
 }  
  
 *// Логика тут не проста:  
 //  
 // Если detA != 0, то его ранг равен N; если (есть хотя бы один) detA\* != 0, то его ранг равен N;  
 // Где A\* - расширенная матрица или ExtendedA; N - кол-во переменных;  
 // Если ранги обоих матриц и N равны, то система имеет единственное решение.  
 //  
 // ps: Если матрица изначально не квадратная, то решений бесконечно много!* private bool isHasSolutions(Matrix vector)  
 {  
 if (!this.IsSquareMatrix)  
 return false;  
   
 var detA = this.CalculateDeterminant();  
 if (detA == 0)  
 return false;  
   
 var ExtendedA = new Matrix(m, n + 1);  
 ExtendedA.CopyNumbersFrom(this);  
  
 for (var j = 0; j < m; j++)  
 ExtendedA[j, 3] = vector[j, 0];  
  
 for (var c = 0; c < n + 1; c++)  
 {  
 if (ExtendedA.WithoutColumn(c).CalculateDeterminant() != 0)  
 return true;  
 }  
  
 return false;  
 }  
   
 private void CopyNumbersFrom(Matrix matrix)  
 {  
 for (var i = 0; i < matrix.m; i++)  
 for (var j = 0; j < matrix.n; j++)  
 this[i, j] = matrix[i, j];  
 }  
   
 private void MakeActionOverEveryNumber(Action<int, int> act)  
 {  
 for(var i = 0; i < m; i++)  
 {  
 for(var j = 0; j < n; j++)  
 {  
 act(i, j);  
 }  
 }  
 }  
   
 private Matrix CreateTransposeMatrix()  
 {  
 var result = new Matrix(n, m);  
 result.MakeActionOverEveryNumber((i, j) => result[i, j] = this[j, i]);  
 return result;  
 }  
   
 private double CalculateMinor(int i, int j)   
 => this.WithoutColumn(j).WithoutRow(i).CalculateDeterminant();  
   
 private Matrix WithoutRow(int row)  
 {  
 if (row < 0 || row >= m)  
 throw new ArgumentException("Не существующий индекс строки!");  
  
 var result = new Matrix(m - 1, n);  
   
 result.MakeActionOverEveryNumber((i, j) => result[i, j] = i < row ? this[i, j] : this[i + 1, j]);  
 return result;  
 }  
  
 private Matrix WithoutColumn(int column)  
 {  
 if (column < 0 || column >= n)  
 throw new ArgumentException("Не существующий индекс колонки!");  
  
 var result = new Matrix(m, n - 1);  
   
 result.MakeActionOverEveryNumber((i, j) => result[i, j] = j < column ? this[i, j] : this[i, j + 1]);  
 return result;  
 }   
}

# Вывод

Я сумер реализовать класс матрицы со всеми необходимыми методами. Программа работает достаточно быстро, хотя есть способы ее оптимизации (например, сохранять уже посчитанный определитесь, чтобы не пересчитывать его). В процессе выполнения пришлось много вспоминать теории матриц, однако большую часть я знал, что очень мне помогло.