**Библиотека для работы с матрицами, реализованная на языке C#.**

**Подробная документация.**

**Версия 1.0**

**Последняя актуализация 03.11.2020**

*Содержание*

***Класс Determinant***2

2

2

2

***Класс MatrixClass***3

3

4

4

5

5

***Примеры использования библиотеки***8

***Класс Determinant***

***(<public> <abstract> class Determinant)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Поля*** | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Имя*** | ***Назначение*** |
| private | static | List<string> | history | Информация о преобразованиях в матрице |
| Double[,] | table | Матрица |
| Int | size | Размер матрицы |
| Double | det | Определитель матрицы |
| Short | eps | Точность расчетов  (10-12) |
| ***Методы*** | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Имя*** | ***Назначение*** |
| public | static | void | CreateDetTable  (Double[,]) | Создает матрицу |
| private | DetFind  () | Вычисляет детерминант матрицы, приводя ее к ступенчатому виду |
| ***Свойства*** | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Имя*** | ***Назначение*** |
| public | static | List<string> | GetHistoryChangeTable | Возвращает историю изменений матрицы |
| Double[,] | GetTable | Возвращает матрицу |
| Int | GetSize | Возвращает размер (квадратной) матрицы |
| Double | GetDeterminant | Возвращает определитель матрицы |
| Short | Eps | Возвращает/Задает точность, к которой привести полученные значения в матрице и определитель |

***Класс MatrixClass***

***(<public> class MatrixClass)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Поля*** | | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Имя*** | | ***Назначение*** |
| public |  | *Enum* | ValueName | | *Перечисляет типы данных в классе. Используется в* PrintValue |
| private | Short | eps | | Точность расчетов  (10-10) |
| Double[,] | table | | Матрица |
| Double[] | lineSolution | | Значения равенств линейных уравнений  (для решения СЛАУ) |
| Double[,] | transpositionTable | | Транспонированная матрица |
| inverseTable | | Обратная матрица |
| gausTable | | Матрица после преобразований по методу Гаусса |
| Double[] | gausSolution | | Значения равенств линейных уравнений  после преобразований по методу Гаусса |
| string | answerGaus | | Ответ, полученный в результате решения СЛАУ по методу Гаусса |
| answerKramer | | Ответ, полученный в результате решения СЛАУ по методу Крамера |
| answerMatrix | | Ответ, полученный в результате решения СЛАУ матричным методом |
| Double | det | | Определитель |
| Int | rowCount | | Кол-во строк |
| columnCount | | Кол-во столбцов |
| string | size | | Размеры, представленные в форме строк  ({i}x{j}), где  i = rowCount,  j = columnCount |
| ***Переопределение операторов*** | | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Оператор*** | | ***Назначение*** |
| public | static | MatrixClass | +  (MatrixClass  matrix1,  MatrixClass matrix2) | | Складывает две матрицы и возвращает объект MatrixClass с новой матрицей. |
| -  (MatrixClass  matrix1,  MatrixClass matrix2) | | Вычитает две матрицы и возвращает объект MatrixClass с новой матрицей. |
| \*  (MatrixClass  matrix1,  Double k) | | Умножает матрицу на заданное число и возвращает объект MatrixClass с новой матрицей. |
| \*  (MatrixClass  matrix1,  MatrixClass matrix2)  (Перегрузка оператора \*) | | Умножает две матрицы и возвращает объект MatrixClass с новой матрицей. |
| ***Конструкторы*** | | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Имя*** | | ***Назначение*** |
| public |  |  | MatrixClass  (Int \_size,  Double[,] value,  Double[] \_lineSolution = null) | | Создает квадратную матрицу с возможностью привязать к ней значения равенств линейных уравнений для решения СЛАУ |
| MatrixClass  (Int \_row,  Int \_column,  Double[,] value) | | Создает произвольную матрицу |
| ***Методы*** | | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Имя*** | | ***Назначение*** |
| private |  | void | PrintValue  (ValueName valueName, string name = “”) | | Печатает выбранное ValueName в консоль.  (Используется для наглядного представления данных или отладки) |
| Solve  () | | Решает СЛАУ тремя способами  (Методом Гаусса, Крамера и матричным методом) |
| InverseMatrix  () | | Находит обратную матрицу |
| TranspositionMatrix  () | | Транспонирует матрицу |
| KramerMethod  () | | Решает СЛАУ методом Крамера |
| GausMethod  () | | Решает СЛАУ методом Гаусса |
| MatrixMethod  () | | Решает СЛАУ матричным методом |
| HistoryParsing  (List<string> \_history,  Bool forInverse = false,  Double[,] tb = null) | | Обрабатывает информацию об изменениях матрицы при ее приведении к ступенчатому виду, отображая это в значениях равенств линейных уравнений или для единичной таблицы, если этот метод вызывается методом InverseMatrix  (Используется в InverseMatrix*,* GausMethod) |
| ***Свойства*** | | | | | |
| ***Доступ*** | ***Ключевые слова*** | ***Тип данных*** | ***Имя*** | ***Назначение*** | |
| public |  | Short | Eps | Возвращает/Задает точность, к которой привести результаты решений | |
| Double[,] | GetMatrix | Возвращает матрицу | |
| GetInverseMatrix | Возвращает обратную матрицу | |
| GetTransMatrix | Возвращает транспонированную матрицу | |
| GetChangeGausMatrix | Возвращает матрицу, приведенную к ступенчатому виду, т.е с обнуленными элементами под и над главной диагональю | |
| Double[] | GetSolution | Возвращает значения равенств линейных уравнений (если таковые были заданы в конструкторе квадратных матриц) | |
| GetGausSolution | Возвращает значения равенств линейных уравнений  после преобразований по методу Гаусса | |
| string | GetAnswerKramer | Возвращает результат решения СЛАУ по методу Крамера | |
| GetAnswerGaus | Возвращает результат решения СЛАУ по методу Гаусса | |
| GetAnswerMatrix | Возвращает результат решения СЛАУ матричным методом | |
| GetSize | Возвращает размеры матрицы, представленные в форме строк  ({i}x{j}), где  i = rowCount,  j = columnCount | |
| Int | GetRowCount | Возвращает количество строк в матрице | |
| GetColumnCount | Возвращает количество столбцов в матрице | |
| Double | GetDetMatrix | Возвращает определитель матрицы | |

***Примеры использования библиотеки***

***Задача 1.***

*Найти определитель.*

*Решение:*

*Поскольку нам не нужно проводить операции над матрицами, то нет смысла создавать объект с функционалом, который нам не пригодиться. Воспользуемся абстрактным классом Determinant.*

using System;

using MatrixLibrary;

namespace ConsoleMatrixLibrary

{

class Program

{

static void Main(string[] args)

{

double[,] A = { { 4, 2, 3},

{ 7, 1, 5},

{ 8, 9, 10}};

Determinant.CreateDetTable(A);

Console.WriteLine(Determinant.GetDeterminant); //Output -35

}

}

}

*Обращаю внимание, что библиотеку нужно подключать. Также добавлять на нее ссылку в проект (подключать dll).*

***Задача 2.***

*Найти определитель матрицы С, если он ненулевой, сложить матрицу А с матрицей B, умножить на определитель С.*

*Решение:*

*Здесь функционала класса Determinant явно недостаточно, однако заметим, что не имеет смысла создавать объект под матрицу С.*

using System;

using MatrixLibrary;

namespace ConsoleMatrixLibrary

{

class Program

{

static void Main(string[] args)

{

double[,] C = { { 1, 3},

{ 8, 3}};

double[,] A = { { 4, 2, 3},

{ 7, 1, 5},

{ 8, 9, 10}};

double[,] B = { { 1, 2, 3},

{ 4, 5, 6},

{ 7, 8, 9}};

MatrixClass a1 = new MatrixClass(3, A);

MatrixClass b1 = new MatrixClass(3, B);

a1.PrintValue(MatrixClass.ValueName.Matrix, "До преобразования");

Determinant.CreateDetTable(C);

Console.WriteLine($"Det. C = {Determinant.GetDeterminant}");

if (Determinant.GetDeterminant != 0)

{

/\*Можем создать новый объект

MatrixClass sum = a1 + b1;

Можем переприсвоить матрицу\*/

a1 += b1;

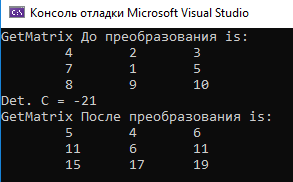
}

a1.PrintValue(MatrixClass.ValueName.Matrix, "После преобразования");

}

}

}

**

***Задача 3.***

*Решить СЛАУ. Найти обратную матрицу и убедиться в том, что найденная матрица действительно является обратной.*

*Решение:*

using System;

using MatrixLibrary;

namespace ConsoleMatrixLibrary

{

class Program

{

static void Main(string[] args)

{

double[,] arr = { { 1, -3, 5, -7},

{ 3, -5, 7, -1},

{ 5, -7, 1, -3},

{ 7, -1, 3, -5} };

double[] solution = { 12, 0, 4, 16 };

MatrixClass m1 = new MatrixClass(4, arr, solution);

//Решаем СЛАУ

m1.Solve();

//Создаем обратную матрицу

double[,] inverseMatrix = m1.GetInverseMatrix;

MatrixClass m2 = new MatrixClass(4, inverseMatrix);

//Проверяем

m2.PrintValue(MatrixClass.ValueName.Matrix, "Обратная матрица");

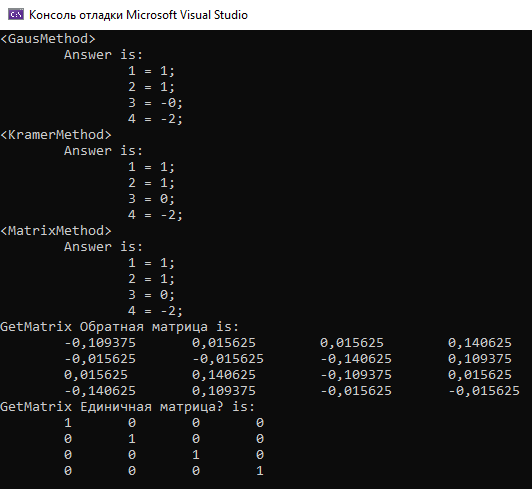
m2 \*= m1;

m2.PrintValue(MatrixClass.ValueName.Matrix, "Единичная матрица?");

}

}

}



*Как видим, после умножения мы получили единичную матрицу, а это значит, что обратная матрица была найдена верно.*