Практическая работа №2. Разработка типов на C#.

Задание. Реализуйте пользовательский тип `Matrix` для работы с матрицами произвольного размера, а также возможность работы с матрицами через консоль (ввод, вывод, операции).

Рекомендации

Данные. Структура для хранения данных (элементов матрицы) может быть реализована в виде прямоугольного массива (`double[,] data`), массива массивов (`double[][] data`) или одномерного массива (`double[] data`).

Внешний пользователь типа Matrix не должен напрямую работать с внутренней структурой данных.

Конструкторы. Реализуйте несколько конструкторов для инициализации нового объекта:

```
public Matrix(int nRows, int nCols) { .. }
public Matrix(double[,] initData) { ..}
```

Свойства. Реализуйте доступ к элементам матрицы через свойство-индексатор:

```
public double this[int i, int j] {..}
```

Размер матрицы доступен только для чтения через свойства:

```
public int Rows {.. }
public int Columns { .. }
// размер квадратной матрицы
public int? Size { .. }
```

Реализуйте несколько булевых свойств:

```
// Является ли матрица квадратной
public bool IsSquared { .. }
// Является ли матрица нулевой
public bool IsEmpty { .. }
// Является ли матрица единичной
public bool IsUnity { .. }
// Является ли матрица диагональной
public bool IsDiagonal { .. }
// Является ли матрица симметричной
public bool IsSymmetric { .. }
```

Операторы. Реализуйте стандартные матричные операции через перегрузку операторов:

```
public static Matrix operator+(Matrix m1, Matrix m2)
public static Matrix operator-(Matrix m1, Matrix m2)
public static Matrix operator*(Matrix m1, double d)
public static Matrix operator*(Matrix m1, Matrix m2)
```

Реализуйте операторы преобразования типов:

```
public static explicit operator Matrix(double[,] arr)
```

Методы. Реализуйте несколько экземплярных методов (н-р, для транспонирования матрицы, для вычисления следа матрицы):

```
public Matrix Transpose() {.. }
public double Trace() { ..}
```

Реализуйте переопределение метода `ToString` для преобразования матрицы в строку:

```
public override string ToString() { .. }
```

Статические методы. Реализуйте статические методы для порождения единичной и нулевой матрицы определенного размера:

```
public static Matrix GetUnity(int Size) { .. }
public static Matrix GetEmpty(int Size) { .. }
```

Реализуйте статические методы для создания матрицы по строчке в определенном формате.

```
public static Matrix Parse(string s) { .. }
public static bool TryParse(string s, out Matrix m){ ..}
```

Формат строки может быть таким: элементы строчки матрицы разделены пробелами, после запятой начинается новая строчка. Например,

«1 2 3, 4 5 6, 7 8 9»

Для разбиения строки на подстроки используйте метод `Split` класса `String`.

Метод `Parse` должен генерировать исключение `FormatException`, если формат некорректный.

Работа с матрицами

Реализуйте возможность работы с матрицами через консоль. Можно использовать либо консольное меню для выбора действий, либо строку ввода.

Пример консольного меню:

```
Работа с матрицами
------------------------------
     1 – Ввод матрицы
     2 - Операции
     3 – Вывод результатов
     0 - Выход
------------------------------
```

В случае работы с матрицами через строку ввода:

```
> m1 = 1 1 1, 2 2 2, 3 3 3
matrix created (3 x 3)..

> m2 = 3 3 3, 2 2 2, 1 1 1
matrix created (3 x 3)..

> m3 = m1 + m2
matrix calculation completed..

> m3
4 4 4
4 4 4
4 4 4
```