Программирование на C#

Семинар №2

Модуль №2

Тема:

Массивы объектов.

Виды членов класса. Инкапсуляция полей класса.

Задания преподавателя к семинару

- Познакомиться с глубоким и поверхностным копированием массива ссылок
- Поработать с различными членами класса
- Познакомиться с инкапсуляцией

Полезные материалы к семинару

- Array clone https://learn.microsoft.com/ru-ru/dotnet/api/system.array.clone?view=net-6.0
- Array copy https://learn.microsoft.com/ru-ru/dotnet/api/system.array.copy?view=net-6.0
- Поверхностное и глубокое копирование https://www.geeksforgeeks.org/difference-between-shallow-and-deep-copy-of-a-class/
- Члены класса https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/members

Задачи с решениями. Demo

Demo 01. Задание

Определить класс **правильных** многоугольников, в котором конструктор с умалчиваемыми значениями аргументов играет роль конструктора умолчания. Поля класса – число сторон (целое) и радиус вписанной окружности (вещественный). Свойства – периметр и площадь многоугольника. Общедоступный метод **PolygonData()** формирует и возвращает строку со значениями полей и свойств объекта.

В основной программе создать массив правильных многоугольников. Размер массива вводить с клавиатуры. Значения полей заполняются случайно (стороны в диапазоне [3;10], радиус в диапазоне [10;50)). Затем СКОПИРОВАТЬ данный массив. В скопированном массиве, у каждого многоугольника увеличить каждое поле в 2 раза. Вывести оба массива.

Demo 01. Описание класса Polygon.

```
public class Polygon
{
    private int _numberOfSides; // Количество сторон.
    private double _radius; // Радиус вписанной окружности.

    public Polygon(int numberOfSides = 5, double radius = 13.5)
    {
        _numberOfSides = numberOfSides;
        _radius = radius;
    }
    // Свойство для получения периметра.
    public double Perimeter => 2 * _numberOfSides * _radius * Math.Tan(Math.PI / _numberOfSides);
    // Свойство для получения площади.
    public double Area => Perimeter * _radius / 2;

    public string PolygonData()
    {
        return $"N = {_numberOfSides}; R = {_radius:F3}; P = {Perimeter:F3}; S = {Area:F3}";
    }

    public void EnLarge(int k) // Увеличение полей в k раз.
    {
        _numberOfSides *= k;
        _radius *= k;
    }

    public Polygon Copy() // Копирование многоугольника.
    {
        return new Polygon(_numberOfSides, _radius);
    }
}
```

Demo 01. Описание статических методов класса Progam.

```
public static void ReadInt(string message, out int result) // Метод чтения целого числа.
{
    Console.WriteLine(message);
    while (!int.TryParse(Console.ReadLine(), out result) || result < 0)
    {
        Console.WriteLine("Некорректный ввод. Попробуйте снова.");
    }
}

public static Polygon[] PolygonCopy(Polygon[] polygons) // Метод глубокого копирования
массива многоугольников.
{
    Polygon[] polygonsCopy = new Polygon[polygons.Length];
    for (int i = 0; i < polygons.Length; i++)
    {
        polygonsCopy[i] = polygons[i].Copy();
    }

    return polygonsCopy;
}
```

Demo 01. Описание статических методов класса Progam.

```
public static void EnlargePolygons(Polygon[] polygons, int k = 2) // Метод увеличения многоугольника в k раз.
{
    for (int i = 0; i < polygons.Length; i++)
    {
        polygons[i].EnLarge(k);
    }
}


public static void PrintPolygons(Polygon[] polygons) // Метод печати многоугольника.
{
    foreach (var polygon in polygons)
    {
        Console.WriteLine(polygon.PolygonData());
    }

    Console.WriteLine(new string('-', Console.WindowWidth));
}
```

Demo 01. Описание Main.

```
static void Main(string[] args)
{
    ReadInt("Введите количество элементов массива.", out int n);
    Polygon[] polygons = new Polygon[n];
    Random random = new Random();
    for (int i = 0; i < n; i++) // Заполнение массива случайными многоугольниками.
    {
        int rnd = random.Next(0, 5);
        switch (rnd)
        {
            case 0: // Заполняется умалчиваемыми аргументами.
                polygons[i] = new Polygon();
                continue;
            case <= 3: // Заполняется заданными аргументами.
                polygons[i] = new Polygon(random.Next(3, 11), 10 + random.NextDouble() * 40);
                continue;
            default: // Радиус задаётся умалчиваемым аргументом.
                polygons[i] = new Polygon(random.Next(3, 11));
                break;
        }
    }

    Polygon[] polygonsCopy1 = (Polygon[])polygons.Clone(); // Неглубокое копирование.
    Polygon[] polygonsCopy2 = new Polygon[polygons.Length];
    Array.Copy(polygons, polygonsCopy2, polygons.Length); // Неглубокое копирование.
    Polygon[] polygonsCopy3 = PolygonCopy(polygons); // Глубокое копирование.
```

Demo 01. Описание Main.

```
Console.WriteLine("Изменяем 1 копию.");
EnlargePolygons(polygonsCopy1);

Console.WriteLine("Изначальный массив:");
PrintPolygons(polygons);
Console.WriteLine("Копия 1:");
PrintPolygons(polygonsCopy1);
Console.WriteLine("Копия 2:");
PrintPolygons(polygonsCopy2);
Console.WriteLine("Копия 3:");
PrintPolygons(polygonsCopy3);
```

Вывод:

```
Введите количество элементов массива.
3
Изменяем 1 копию.
Изначальный массив:
N = 10; R = 90.474; P = 587.936; S = 26596.479
N = 10; R = 27.000; P = 175.457; S = 2368.665
N = 12; R = 26.171; P = 168.299; S = 2202.259
-----------------------------------------------------------------------------------------------
Копия 1:
N = 10; R = 90.474; P = 587.936; S = 26596.479
N = 10; R = 27.000; P = 175.457; S = 2368.665
N = 12; R = 26.171; P = 168.299; S = 2202.259
-----------------------------------------------------------------------------------------------
Копия 2:
N = 10; R = 90.474; P = 587.936; S = 26596.479
N = 10; R = 27.000; P = 175.457; S = 2368.665
N = 12; R = 26.171; P = 168.299; S = 2202.259
-----------------------------------------------------------------------------------------------
Копия 3:
N = 5; R = 45.237; P = 328.666; S = 7433.942
N = 5; R = 13.500; P = 98.083; S = 662.062
N = 6; R = 13.085; P = 90.658; S = 593.151
-----------------------------------------------------------------------------------------------
```

Demo 01. Описание Main.

```
Console.WriteLine("Изменяем 2 копию.");
EnlargePolygons(polygonsCopy2);

Console.WriteLine("Изначальный массив:");
PrintPolygons(polygons);
Console.WriteLine("Копия 1:");
PrintPolygons(polygonsCopy1);
Console.WriteLine("Копия 2:");
PrintPolygons(polygonsCopy2);
Console.WriteLine("Копия 3:");
PrintPolygons(polygonsCopy3);
```

Вывод:

```
--------------------------------------------------------------------------------
Изменяем 2 копию.
Изначальный массив:
N = 20; R = 180.948; P = 1146.375; S = 103717.158
N = 20; R = 54.000; P = 342.110; S = 9236.981
N = 24; R = 52.342; P = 330.764; S = 8656.353
--------------------------------------------------------------------------------
Копия 1:
N = 20; R = 180.948; P = 1146.375; S = 103717.158
N = 20; R = 54.000; P = 342.110; S = 9236.981
N = 24; R = 52.342; P = 330.764; S = 8656.353
--------------------------------------------------------------------------------
Копия 2:
N = 20; R = 180.948; P = 1146.375; S = 103717.158
N = 20; R = 54.000; P = 342.110; S = 9236.981
N = 24; R = 52.342; P = 330.764; S = 8656.353
--------------------------------------------------------------------------------
Копия 3:
N = 5; R = 45.237; P = 328.666; S = 7433.942
N = 5; R = 13.500; P = 98.083; S = 662.062
N = 6; R = 13.085; P = 90.658; S = 593.151
--------------------------------------------------------------------------------
```

Demo 01. Описание Main.

```
Console.WriteLine("Изменяем 3 копию.");
EnlargePolygons(polygonsCopy3);

Console.WriteLine("Изначальный массив:");
PrintPolygons(polygons);
Console.WriteLine("Копия 1:");
PrintPolygons(polygonsCopy1);
Console.WriteLine("Копия 2:");
PrintPolygons(polygonsCopy2);
Console.WriteLine("Копия 3:");
PrintPolygons(polygonsCopy3);
```

Вывод:

```
--------------------------------------------------------------------------------
Изменяем 3 копию.
Изначальный массив:
N = 20; R = 180.948; P = 1146.375; S = 103717.158
N = 20; R = 54.000; P = 342.110; S = 9236.981
N = 24; R = 52.342; P = 330.764; S = 8656.353
--------------------------------------------------------------------------------
Копия 1:
N = 20; R = 180.948; P = 1146.375; S = 103717.158
N = 20; R = 54.000; P = 342.110; S = 9236.981
N = 24; R = 52.342; P = 330.764; S = 8656.353
--------------------------------------------------------------------------------
Копия 2:
N = 20; R = 180.948; P = 1146.375; S = 103717.158
N = 20; R = 54.000; P = 342.110; S = 9236.981
N = 24; R = 52.342; P = 330.764; S = 8656.353
--------------------------------------------------------------------------------
Копия 3:
N = 10; R = 90.474; P = 587.936; S = 26596.479
N = 10; R = 27.000; P = 175.457; S = 2368.665
N = 12; R = 26.171; P = 168.299; S = 2202.259
--------------------------------------------------------------------------------
```

Demo 01. Описание Main.

```
Console.WriteLine("Изменяем изначальный массив.");
    EnlargePolygons(polygons);

    Console.WriteLine("Изначальный массив:");
    PrintPolygons(polygons);
    Console.WriteLine("Копия 1:");
    PrintPolygons(polygonsCopy1);
    Console.WriteLine("Копия 2:");
    PrintPolygons(polygonsCopy2);
    Console.WriteLine("Копия 3:");
    PrintPolygons(polygonsCopy3);
}
```

Вывод:

```
--------------------------------------------------------------------------------
Изменяем изначальный массив.
Изначальный массив:
N = 40; R = 361.896; P = 2278.548; S = 412298.953
N = 40; R = 108.000; P = 679.983; S = 36719.068
N = 48; R = 104.683; P = 658.685; S = 34476.663
--------------------------------------------------------------------------------
Копия 1:
N = 40; R = 361.896; P = 2278.548; S = 412298.953
N = 40; R = 108.000; P = 679.983; S = 36719.068
N = 48; R = 104.683; P = 658.685; S = 34476.663
--------------------------------------------------------------------------------
Копия 2:
N = 40; R = 361.896; P = 2278.548; S = 412298.953
N = 40; R = 108.000; P = 679.983; S = 36719.068
N = 48; R = 104.683; P = 658.685; S = 34476.663
--------------------------------------------------------------------------------
Копия 3:
N = 10; R = 90.474; P = 587.936; S = 26596.479
N = 10; R = 27.000; P = 175.457; S = 2368.665
N = 12; R = 26.171; P = 168.299; S = 2202.259
--------------------------------------------------------------------------------
```

ToDo. Разработка дополнений к коду Demo
Self. Задачи на самостоятельное решение

ToDo 01.

К Demo01. Программа определяет площади всех многоугольников и выводит данные о всех объектах. Данные об объекте с минимальной площадью выводится зелёным цветом, данные об объекте с максимальной площадью выводится красным цветом. Подробнее о цветах в консоли можно прочитать в документации: https://docs.microsoft.com/ru-ru/dotnet/api/system.console.foregroundcolor?view=net-6.0

Подумайте почему первые две копии и изначальный массив меняются одновременно, хотя это “копии” изначального массива, а третья копия независима от них?

Self 01. Список целых чисел.

Массив объектов другого типа может быть полем класса.

Класс *IntegerList*, представляет список целых чисел.

Поля:

- _integerList – массив целых чисел.

Методы:

- IntegerList(int size) – создаёт новый список из *size* элементов. Элементы равны 0.
- void Randomize() – заполняет список целыми числами между 0 и 100 включительно.
- void Print() – выводит список элементов и их индексы

Часто необходима возможность добавления в заполненный список. Когда список хранится в массиве, этого можно добиться созданием нового массива необходимого размера и копированием в него значений из старого. Однако, это может быть неэффективно, поэтому можно создавать новый массив, размера 2 раза больше исходного. Дополните класс *IntegerList.*

Необходимо добавить метод *IncreaseSize* и поля для отслеживания количества записанных в массив элементов.

Добавьте метод *void AddElement(int newVal)* в класс *IntegerList.* Данный метод добавляет новый элемент в список. В начале мотода *AddElement* проверьте, не является ли массив полностью заполненным. Если является, вызовите метод *IncreaseSize* перед дальнейшими действиями.

Добавьте метод *void RemoveFirst(int val)* в класс *IntegerList.* Данный метод удаляет первое вхождение элемента *val* из списка. Если указанного элемента в списке нет, метод не должен делать ничего (и ошибку он бросать тоже не должен). Удаление элемента не должно изменять размер массива, но учтите, что массив должен быть последовательным, так что после удаления элемента, необходимо сдвинуть все элементы правее него на одну позицию влево. Так же не забудьте уменьшить значение переменной, отвечающей за количество элементов в списке.

Добавьте метод *void RemoveAll(int val)* в класс *IntegerList.* Данный метод удаляет все вхождения элемента *val* из списка. Если указанного элемента в списке нет, метод не должен делать ничего (исключение он выбрасывать тоже не должен).

Протестируйте все методы.

Self 02. Видеофайлы

Реализовать класс VideoFile, представляющий видеофайл.
Класс должен содержать следующие элементы (другие члены класса добавлять разрешено):
Поля:
string _name – наименование видеофайла;
int _duration – длительность в секундах;
int _quality – качество видеофайла;
Конструкторы:
VideoFile(string name, int duration, int quality) – инициализирует поля класса значениями параметров;
Свойства:
int Size – размер видеофайла (свойство доступа);
Будем считать размер как произведение длительности на качество.
Методы:
string GetInfo() – возвращает строку с информацией о видеофайле
В основной программе создать отдельный объект типа VideoFile и массив из N объектов типа VideoFile, где N – случайное число из интервала [5, 15].
Длительность генерировать случайно из диапазона [60, 360], а качество из диапазона [100, 1000], наименование каждого видеофайла определить самостоятельно (генерировать случайную строку латинских символов длины от 2 до 9).
Вывести на экран информацию о видеофайлах из массива, размер которых больше, чем размер отдельного видеофайла.
Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводите промежуточные значения на экран.

Self 03. Линейные уравнения

Описать класс **LinearEquation**, соответствующую линейному уравнению.

Уравнение $ax + b = c$ задаётся коэффициентами a, b и c.

Определить метод, находящий корень линейного уравнения.

Разрешается добавлять члены классов, необходимые для реализации программы.

В основной программе необходимо создать массив из **N** объектов типа **LinearEquation**, где **N** – число, введенное пользователем с клавиатуры. Координаты коэффициентов a, b и c генерируются случайным образом в интервале **[-10; 10]**.

Вывести на экран информацию обо всех объектах массива.

Отсортировать массив по возрастанию корней линейного уравнения.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводите промежуточные значения на экран.