Программирование на C#

Семинар №1

Модуль №2

Тема:

Классы. Члены классов

Объекты.

Задания преподавателя к семинару

- Научиться объявлять классы в программах на C#
- Немного познакомиться с типом DateTime

Полезные материалы к семинару

- `System.DateTime` - (https://docs.microsoft.com/ru-ru/dotnet/api/system.datetime?view=net-5.0)
- Стандартные форматы DateTime - https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/standard-date-and-time-format-strings
- Настраиваемые форматы DateTime - https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/custom-date-and-time-format-strings
- Свойства - https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/properties

Demo 01. Пример. Описание Класса и Создания Объекта

```
1. class Cat
2. {
3.     // Состояние.
4.     string _name;
5.     DateTime _birthDay;
6.     string _breed;
7.
8.     // Поведение.
9.     public string Meow()
   => "Meow-meow-meow";
10.}
```

```
1. class Program
2. {
3.     static void Main()
4.     {
5.         Cat cat1 = new Cat();
6.         Cat cat2 = new();
7.
   Console.WriteLine(cat1.Meow());
8.
   Console.Write(cat2.Meow());
9.     }
10.}
```

Demo 01. Поля Класса Открыты

```
1. class Cat
2. {
3.     // Состояние.
4.     public string Name;
5.     public DateTime
   BirthDay;
6.     public string Breed;
7.
8.     // Поведение.
9.     public string Meow()
   => "Meow-meow-meow";
10.}
```

Делать поля открытыми – плохая идея.

Необходимо:

ToDo 01.

Убрать public из полей

Создать свойства доступа к полям

Demo 01. Как Работать с DataTime.

```
1.  public string Name { get;set; }
2.  public DateTime BirthDay { get; set;}
3.  public string Breed { get; set; }
```

ToDo 02. Добавить конструктор

Добавим проверку корректности даты рождения:

```
1.  public DateTime BirthDay {
2.          get { return _birthDay; }
3.          set {
4.              if (value <
    DateTime.Now)
5.                  {
6.                      _birthDay = value;
7.                  }
8.          }
9.      }
```

Demo 02. Шестнадцатеричные Цифры Числа

```
//Представление неотрицательных целых
1.    public class HexNumber {
2.        uint number;// Целое неотрицательное число
3.        char[] hexView;// Шестнадцатеричное представление
4.        public HexNumber(uint n) { //конструктор
5.            number = n;
6.            hexView = series(n);
7.        }
8.        public HexNumber() : this(0) { }// конструктор
      //умолчания
9.    // СВОЙСТВА КЛАССА: Number, HexView, Record
10.   // МЕТОД Series, ВОЗВРАЩАЮЩИЙ МАССИВ ШЕСТНАДЦАТЕРИЧНЫХ
      ЦИФР
11.   }   // HexNumber
```

```
1.  public uint Number
2.  {  // Свойство: десятичное целое
3.       get { return number; }
4.       set
5.       {
6.           number = value;
7.           hexView = Series(value);
8.       }
9.  }
10. public char[] HexView
11. {  // Свойство: массив символов-цифр
12.          get { return hexView; }
13.  }
```

Demo 02. Шестнадцатеричные Цифры Числа

```
1.  public string Record
2.    { // Свойство: строковое представление
    //(шестнадцатеричное) числа
3.      get
4.      {
5.        string str = new String(hexView);
6.        return "0x" + str;
7.      }
8.    }
```

```
1.  // Возвращает массив шестнадцатеричных цифр числа-
    параметра.
2.      char[] Series(uint num)     {
3.          int arLen = num == 0 ? 1 : (int)Math.Log(num,
    16) + 1;
4.          char[] res = new char[arLen];
5.          for (int i = arLen - 1; i >= 0; i--) {
6.              uint temp = (uint)(num % 16);
7.              if (temp >= 0 & temp <= 9) res[i] =
    (char)('0' + temp);
8.              else res[i] = (char)('A' + temp % 10);
9.              num /= 16;
10.         }
11.         return res;
12.     }   // series
```

Demo 02. Шестнадцатеричные Цифры Числа. Main()

```
1.  HexNumber hex;              // ссылка с типом класса
2.  hex = new HexNumber(0);   // объект класса
3.  uint number;
4.  while (true)  { // цикл для ввода разных значений числа
5.    do Console.Write("Введите целое неотрицательное число:  ");
6.    while (!uint.TryParse(Console.ReadLine(), out number));

7.    hex.Number = number;      // Изменяем объект через свойство
8.    Console.WriteLine("Свойство Number: " + hex.Number);
9.    Console.Write("Шестнадцатеричные цифры числа: ");
10.
11.   foreach (char h in hex.HexView) Console.Write("{0} ", h);
12.
13.   Console.WriteLine($"{Environment.NewLine}Шестнадцатеричная запись: " +
    hex.Record);
14.   Console.WriteLine("Для выхода нажмите клавишу ESC");
15.
16.   if (Console.ReadKey(true).Key == ConsoleKey.Escape) break;
17. }    // while
```

Demo 03. С днём рождения?

Программа проверяет скоро ли день рождения? (Без учета високосного года)
Получите от пользователя имя и дату рождения (год, месяц, число), создайте объект класса `Birthday`, описывающего сведения о человеке. Получите число дней до очередного дня рождения.
Используются возможности библиотечной структуры **`System.DateTime`** (https://docs.microsoft.com/ru-ru/dotnet/api/system.datetime?view=net-5.0)

Форматирование DateTime

Стандартные форматы:
https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/standard-date-and-time-format-strings
Настраиваемые форматы:
https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/custom-date-and-time-format-strings

Demo 03. С днём рождения?

```
1.    class Birthday   {
2.        string name; // Закрытое поле – фамилия.
3.        int year, month, day; // Закрытые поля: год, месяц, день
      //рождения
4.        public Birthday(string name, int y, int m, int d)    {
      //Конструктор
5.           this.name = name;
6.           year = y; month = m; day = d;
7.        }
8.        DateTime Date    { // закрытое свойство - дата рождения
9.           get { return new DateTime(year, month, day); }
10.       }
11.       public string Information {   //свойство-сведения о человеке
12.          get {
13.              return name + ", дата рождения " + day + ":" + month +
      ":" + year;
14.          }
15.       }

16.        public int HowManyDays { // свойство - сколько дней
      до дня рождения
17.           get {
18.               // номер сего дня от начала года:
19.               int nowDOY = DateTime.Now.DayOfYear;
20.               //  номер дня рождения от начала года:
21.               int myDOY = Date.DayOfYear;
22.               int period = myDOY >= nowDOY ? myDOY -
      nowDOY :
23.                                              365 - nowDOY +
      myDOY;
24.               return period;
25.           }}
```

```
1.    class Program    {
2.           static void Main( )  {
3.               Birthday md = new
      Birthday("Чапаев", 1887, 2, 9);
4.
      Console.WriteLine(md.Information);
5.               Console.WriteLine("До
      следующего дня рождения дней осталось: ");
6.
      Console.WriteLine(md.HowManyDays);
7.
8.               Birthday km = new
      Birthday("Маркс Карл", 1818, 5, 4);
9.
      Console.WriteLine(km.Information);
10.              Console.WriteLine("До
      следующего дня рождения дней осталось: ");
11.
      Console.WriteLine(km.HowManyDays);
12.          }
13.      }
14.   }
```

ToDo 03: С днём рождения?

1. Добавьте в класс **`Birthday`** конструктор без параметров, устанавливающий поля объекта класса в состояние «1 января 1970».
2. Добавьте в класс **`Birthday`** методы, позволяющий получить информацию о дне рождения со следующими форматами представления даты: **`DD Month YYYY`**, **`DD-MM-YY`**.
3. Решите проблему високосного года (учтите верно количество дней до дня рождения).
4. Добавьте методы для получение текущего значения и изменения значения поля name.

Вы, конечно же заметили, что создание дополнительных методов для получения доступа к закрытым полям класса довольно утомительно. Изучите свойства: https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/properties и замените методы доступа к закрытым членам класса свойствами.
Воспользуйтесь ссылкой с готовыми набросками кода: https://repl.it/@Maksimenkova/ClassesObjects01

Demo 04: Точка На Плоскости

Класс "точка на плоскости" (**Point**):

Автореализуемые вещественные свойства **X** и **Y** задают декартовы координаты точки.

Свойства для чтения **ρ** и **φ** - полярные координаты точки.

Конструктор общего вида и конструктор умолчания.

В основной программе создать два объекта класса **Point** (две точки),

ввести данные о третьей точке и вывести сведения о трех точках

в порядке возрастания их расстояний от начала координат.

Конец работы программы – ввод двух нулевых значений координат

$$\varphi = \begin{cases} \arctan\left(\frac{y}{x}\right), x > 0, y \geq 0 \\ \arctan\left(\frac{y}{x}\right) + 2\pi, x > 0, y < 0 \\ arctan\left(\frac{y}{x}\right) + \pi, x < 0 \\ \frac{\pi}{2}, x = 0, y > 0 \\ \frac{3\pi}{2}, x = 0, y < 0 \\ 0, x = 0, y = 0 \end{cases}$$

Demo 04. Точка на плоскости

```
1.  class Program    {
2.  class Point {
3.    public double X { get; set; }
4.    public double Y { get; set; }
5.    public Point(double x, double y) { X = x; Y = y; }
6.    public Point() : this (0,0) { } // конструктор
    умолчания
7.    public double Ro {
8.       get {
9.          //ToDo 04: реализовать свойство
10.      }
11.   }
12.   public double Fi {
13.      get {
14.         //ToDo 05: реализовать свойство
15.      }
16.   }
17.   public string PointData   {   // СВОЙСТВО
18.     get  {
19.        string maket = "X = {0:F2}; Y = {1:F2}; Ro =
    {2:F2}; Fi = {3:F2} ";
20.        return string.Format(maket, X, Y, Ro, Fi);
21.     }
22.   }
```

```
1.  static void Main()        {
2.      Point a, b, c;
3.      a = new Point(3, 4);
4.      Console.WriteLine(a.PointData);
5.      b = new Point(0,3);
6.      Console.WriteLine(b.PointData);
7.      c = new Point();
8.      double x = 0, y = 0;
9.      do    {
10.         Console.Write("x = ");
11.         double.TryParse(Console.ReadLine(), out x);
12.         Console.Write("y = ");
13.         double.TryParse(Console.ReadLine(), out y);
14.         c.X = x; c.Y = y;
15. // ToDo 06: В основной программе создать
    два объекта класса (две точки), ввести
    данные о третьей точке и вывести сведения
    о трех точках в порядке возрастания их
    расстояний от начала координат.
16.     } while (x != 0 | y != 0);
17. }
```

Demo 05. Магические Квадраты.

Магический квадрат – это матрица, в которой равны суммы элементов каждой из строк, каждого из столбцов и обеих диагоналей. В этом задании вам предстоит написать код для определения, является ли квадрат магическим.

Файл ***Square.cs*** содержит заготовку для класса, представляющего квадратную матрицу. Он содержит заголовки для конструктора, задающего размер матрицы, и для метода, читающего значения в квадрат, печатающего квадрат, находящего сумму отдельной строки/столбца/диагонали и определяющего, является ли квадрат магическим. Заготовка для считывания вам дана, нужно написать все остальные методы.

Файл ***SquareTest.cs*** содержит заготовку программы, читающей значения квадрата из файла ***magicData.txt*** и сообщающей, является ли квадрат магическим. Заполните оставшиеся методы, учитывая комментарии к коду. Обратите внимание, что главный метод считывает только размер квадрата, после создания квадрата указанного размера он вызывает метод ***ReadSquare*** для считывания значений элементов квадрата. Метод ***ReadSquare*** требует массив строк (прочитанных из файла) в качестве параметра.

Вы должны убедиться, что первые три квадрата в файле являются магическими, а остальные нет. Учтите, что значение -1 в конце файла сообщает программе об окончании чтения.

Demo 05. Магические Квадраты

```
/*
 * Square.cs
 * Определяет квадрат с методами по его созданию, заполнению
 * и подсчёту сумм элементов строк, столбцов и диагоналей,
 * и определению принадлежности квадрата к магическим
 */

public class Square {
    private int[][] _square;

    /// <summary>
    /// Создаёт новый квадрат указанного размера
    /// </summary>
    /// <param name="size">Размер квадрата</param>
    public Square(int size) { }

    /// <summary>
    /// Возвращает сумму элементов указанной строки
    /// </summary>
    /// <param name="row">Номер строки</param>
    public int SumRow(int row) { }

    /// <summary>
    /// Возвращает сумму элементов указанного столбца
    /// </summary>
    /// <param name="col">Номер столбца</param>
    public int SumCol(int col) { }

    /// <summary>
    /// Возвращает сумму элементов главной диагонали
    public int SumMainDiag() { }
    /// Возвращает сумму элементов побочной диагонали
    public int SumOtherDiag() { }
    /// <summary>
    /// Возвращает, является ли текущий квадрат магическим
    /// </summary>
    public bool Magic() { }

```

```
    /// <summary>
    /// Считывает значения элементов квадрата из консоли
    /// </summary>
    public void ReadSquare(string[] lines, int lineIndex)
    {
        for (int row = 0; row < _square.Length; row++)
        {
            string[] line = lines[lineIndex + row]
                .Split(new[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
            if (line.Length != _square.Length)
                Console.WriteLine($"Ошибка при чтении
квадрата: строка должна содержать {_square.Length} значений,
а содержит {line.Length}");
            for (int i = 0; i < _square.Length; i++)
                int.TryParse(line[i], out _square[row][i]);
        }
    }
    /// <summary>
    /// Выводит аккуратно отформатированное содержимое
квадрата
    /// </summary>
    public void PrintSquare()  { }
}
```

Demo 05. Магические Квадраты

```
1.  static void Main(string[] args)
2.      {
3.          string[] lines = System.IO.File.ReadAllLines("..\\..\\magicData.txt");    // читаем
    все строки файла в массив
4.          int lineIndex = 0;  // на какой строке файла находимся
5.          int count = 0; // считаем, на каком мы сейчас квадрате
6.          while (lines.Length > lineIndex)
7.          {
8.              int size;   // размер квадрата
9.              if (!int.TryParse(lines[lineIndex], out size))
10.             {
11.                 Console.WriteLine($"Ошибка при чтении размера квадрата: {lines[lineIndex]} -
    не число (строка {lineIndex + 1})");
12.                 return;
13.             }
14.             if (size == -1)    // в конце файла ожидается -1
15.                 return;
16.             lineIndex++;
17.             // TODO 07: создаём новый квадрат размера size
18.             // TODO 08: вызываем метод ReadSquare - считывание квадрата
19.             Console.WriteLine($"\n******** Квадрат номер {++count} ********");}}
20.             // TODO 09: выводим квадрат
21.             // TODO 10: выводим суммы элементов его строк
22.             // TODO 11: выводим суммы элементов его столбцов
23.             // TODO 12: выводим сумму элементов его главной диагонали
24.             // TODO 13: выводим сумму элементов его побочной диагонали
25.             // TODO 14: определяем и выводим, является ли квадрат магическим
```

Demo 05. Магические Квадраты

```
3
8 1 6
3 5 7
4 9 2
7
30 39 48  1 10 19 28
38 47  7  9 18 27 29
46  6  8 17 26 35 37
5  14 16 25 34 36 45
13 15 24 33 42 44  4
21 23 32 41 43  3 12
22 31 40 49  2 11 20
4
48  9  6 39
27 18 21 36
15 30 33 24
12 45 42  3
3
6 2 7
1 5 3
2 9 4
4
3  16  2 13
6   9  7 12
10  5 11  8
15  4 14  1
5
17 24 15  8  1
23  5 16 14  7
4   6 22 13 20
10 12  3 21 19
11 18  9  2 25
7
30 39 48  1 10 28 19
38 47  7  9 18 29 27
46  6  8 17 26 37 35
5  14 16 25 34 45 36
13 15 24 33 42  4 44
21 23 32 41 43 12  3
22 31 40 49  2 20 11
-1
```

Файл *magicData.tx*t внутри папки /bin

Self 01. Прямоугольник

Описать класс **`Rectangle`**, содержащий:

- Два закрытых поля: **`height`** и **`width`** (типа **`double`**), по умолчанию поля имеют единичную длину.
- Свойства (с секциями **`get`** и **`set`**) для доступа к полям.
- Свойство, возвращающее значение периметра прямоугольника.
- Свойство, возвращающее площадь прямоугольника.
- Переопределённый метод **`ToString()`**
- Два конструктора: без параметров и конструктор с двумя параметрами типа **`double`**.

В основной программе получить от пользователя значения длин сторон двух прямоугольников, создать их и вывести на экран значения их площадей и длин сторон.

Self 02. Круг

Определить класс **Circle** с полем радиус **_r** и свойством доступа к нему, значение радиуса положительное вещественное число. В классе **Circle** описать конструктор без параметров и конструктор с вещественным параметром. Определить свойство **S** – площадь круга заданного радиуса. В основной программе получить от пользователя диапазон изменения значения радиуса: (R**min**, R**max**), R**min**, R**max** – произвольные вещественные числа и величину шага **delta** разбиения данного диапазона. Создать объект типа **Circle**, последовательно изменяя значение радиуса на **delta** вычислять и выводить на экран значение площади круга, ограниченного данной окружностью.

Self 03. LatinChar

Определить класс **LatinChar** с полем **_char** и свойством доступа к нему, значение поля – символ латинского алфавита. Значение поля по умолчанию – **‘a’**. Определить конструкторы класса. В основной программе создать объект типа **LatinChar** и, последовательно перебирая все символы из заданного пользователем диапазона [**minChar**, **maxChar**], выводить значение поля **_char** объекта.

Demo 06. Статические Члены Класса

Создать класс со статическими членами, константами и полями (readonly) только для чтения. В отладочном режиме (начиная с первого оператора класса – поставьте точку прерывания на readonly string name =) и на `static int entranceYear =` проследить (по F11) последовательность выполнения инициализаций

Demo 06. Статические Члены Класса

```
1.  class MyClassmate { // Одноклассник
2.      readonly string name = "Неизвестный"; // Фамилия
3.      readonly int birthYear = 1998; // Год рождения
4.      const int apprenticeship = 4;//Срок обучения
5.      static int entranceYear = 2016; // Год
    //поступления
6.      static MyClassmate() //Статический конструктор
7.      {
8.          entranceYear = 2015;
9.      }
10.     public MyClassmate() { } //Конструктор умолчания
11.     public MyClassmate(string name, int by)
12.     {   //Конструктор общего вида.
13.         this.name = name;
14.         birthYear = by;
15.     }
16.     public string Information()
17.     {   // Метод объекта
18.         return "Фамилия: " + name + "; возраст: " +
19.             (entranceYear - birthYear) +
20.             " лет; год окончания: " +
21.             (entranceYear + apprenticeship);
22.     }
23. }
```

```
1.  class Program {
2.      static void Main()
3.      {
4.          MyClassmate Nan = new MyClassmate();
5.          Console.WriteLine(Nan.Information());
6.          MyClassmate Bob = new
    MyClassmate("Смирнов", 1997);
7.          Console.WriteLine(Bob.Information());
8.      }
9.  }
```