Программирование на C#

Семинар №5

Модуль №2

Тема:

Индексаторы, как члены классов

Полезные материалы

1. Обзор индексаторов https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/indexers/
2. Правила и примеры перегрузки операций https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/operator-overloading

Индексаторы

- *ИНДЕКСАТОРЫ ИСПОЛЬЗУЮТСЯ ДЛЯ ИНКАПСУЛЯЦИИ ЗНАЧЕНИЯ, ПРЕДСТАВЛЕННОГО МАССИВОМ*
- *ИНДЕКСАТОР РАБОТАЕТ АНАЛОГИЧНО СВОЙСТВУ И ПРЕДОСТАВЛЯЕТ ГЕТТЕРЫ И СЕТТЕРЫ ДЛЯ ОБЕСПЕЧЕНИЯ КОНТРОЛЯ НАД ЧТЕНИЕМ, ЗАПИСЬЮ И ИЗМЕНЕНИЯМИ ЗНАЧЕНИЙ МАССИВА*
- *ИНДЕКСАТОР СОЗДАЕТ «ОБЁРТКУ» НАД МАССИВОМ, КОТОРАЯ ЗАЩИЩАЕТ ЕГО*
  - *ОТ СОХРАНЕНИЯ НЕКОРРЕКТНЫХ ДАННЫХ В МАССИВ*
  - *ИСПОЛЬЗОВАНИЯ НЕВЕРНОГО ИНДЕКСА МАССИВА*
  - *ИЗМЕНЕНИЯ ССЫЛКИ НА МАССИВ ИЗВНЕ*

```
ключевое. слово      список параметров
              ↓                ↓
ReturnType this[Type param1,...]

{                ↑                ↑
            get          квадратные скобки
            {
                        ...
            }
            set
            {
                        ...
            }
}
```

<МОДИФИКАТОР ДОСТУПА> <ТИП ДАННЫХ> *THIS [INT INDEX]* {

    GET { /* ВОЗВРАЩАЕТ ЗНАЧЕНИЕ ИЗ МАССИВА, ДОСТУПНОЕ ПО ИНДЕКСУ INDEX */ }

    SET { /* УСТАНАВЛИВАЕТ ЗНАЧЕНИЕ МАССИВА ПО ИНДЕКСУ INDEX */ }

}

Перегрузка индексаторов

```
class MyClass
{
   public string this[int index]
   {
      get { ... }
      set { ... }
   }
   public string this[int index1, int index2]
   {
      get { ... }
      set { ... }
   }
   public int this[float index1]
   {
      get { ... }
      set { ... }
   }
}
```

Demo 01. Расписание на неделю

- Разработать класс Schedule, в котором неделя представлена при помощи индексатора со строковым индексом – днём недели.
- Протестировать класс в консольном приложении

https://repl.it/@Maksimenkova/ClassesObjects08

Demo 01. Расписание На Неделю

```
class Schedule {// Начало занятий в разные дни недели
   public string this[string day] {
      get {
         switch (day) {
            case "понедельник": return days[0] == null ? "Нет занятий" : days[0];
            case "вторник": return days[1] == null ? "Нет занятий" : days[1];
            case "среда": return days[2] == null ? "Нет занятий" : days[2];
            case "четверг": return days[3] == null ? "Нет занятий" : days[3];
            case "пятница": return days[4] == null ? "Нет занятий" : days[4];
            case "суббота": return days[5] == null ? "Нет занятий" : days[5];
            case "воскресенье": return days[6] == null ? "Нет занятий" : days[6];
            default: return "Ошибка в обращении!";
         }
      }
   }

   string[] days = new string[7]; // все null по умолчанию

   public Schedule(params string[] d) {
      for (int i = 0; i < d.Length; i++)
          days[i] = d[i];
   }
} // Schedule
```

Заголовок индексатора, принимает в качестве индекса string

Demo 01. Расписание На Неделю

```
class Program {
        static void Main() {
            Schedule Module_2 = new Schedule("9_00", "10_30",
                                             null, "15_00", "13_40");
            Console.WriteLine("Начало занятий в модуле 2:");
            Console.WriteLine("понедельник: \t" + Module_2["понедельник"]);
            Console.WriteLine("вторник: \t" + Module_2["вторник"]);
            Console.WriteLine("среда:    \t" + Module_2["среда"]);
            Console.WriteLine("четверг: \t" + Module_2["четверг"]);
            Console.WriteLine("пятница: \t" + Module_2["пятница"]);
            Console.WriteLine("суббота: \t" + Module_2["суббота"]);
            Console.WriteLine("воскресенье: \t" + Module_2["воскресенье"]);
        }
    }
```

Demo 02. Математическая Функция

Класс с индексатором представляет математическую функцию одного аргумента, определенную только в пределах от X_{min} до X_{max}. Вне этого интервала функция равна нулю.

Для определенности, пусть класс представляет отрезок функции sin(x) на интервале $[X_{min},X_{max}]$.

В основной программе определить объект класса для X_{min}=0, X_{max}= π и, используя его индексатор, вычислить для отрезка функции, представляемой объектом класса, определенный интеграл методом трапеций (или прямоугольников).

https://repl.it/@Maksimenkova/ClassesObjects07

Demo 02. Математическая Функция

```
using System;
class MyFunction {
    double xmi, xma;
    public MyFunction(double mi, double ma)
    {
        xmi = mi; xma = ma;
    }
    public double this[double x] {
        get {
            return x < xmi | x > xma ? 0 : Math.Sin(x);
        }
    }
}
class Program {
    static void Main()
    {
        double rmi = -5, rma = 5;
        MyFunction sin = new MyFunction(0, Math.PI);
        double s = 0, del = 0.001;
        for (double x = rmi; x < rma; x += del)
            s += sin[x];
        s *= del;
        Console.WriteLine(s);
    }
}
```

В классе определены min и max диапазона. Конструктор копирования для инициализации значений max и min. В свойство класса определяет для аргумента X геттер (или 0 или Sin(x)). Свойство работает с индексатором, указывающим на текущий экземпляр. В Main конструктор задает для экземпляра класса MyFunction диапазон от 0 до Pi. В пределах представленного диапазона (- 5 до 5) в цикле вычисляем сумму, прибавляя на каждой итерации значение функции с приращением аргумента 0.001

Self 01. Библиотека

Реализовать класс Библиотека (**Library**), предоставляющий доступ к коллекции книг (класс **Book**) через индексатор.
Предусмотреть:

- возможность создания как пустой библиотеки (конструктор без параметров), так и из готовой коллекции книг;
- метод добавления книги в библиотеку (**AddBook**);
- свойство, возвращающее количество книг в библиотеке (**N**);
- метод с целочисленным параметром **n**, возвращающий книги (в виде одномерного массива ссылок) с количеством страниц меньшим, чем **n** (**GetBooksWithTheLessAmountOfPages**);
- метод **string GetInfo()**, который возвращает строку с информацией о библиотеке (разрешается переопределить **ToString** вместо этого метода);

Каждая книга имеет следующие поля:

- количество страниц (**_countPages**);
- номер секции в библиотеке (**_sectionNumber**);

Предусмотреть невозможность изменения полей книги после её создания.

Предусмотреть наличие конструктора в классе **Book**.

Предусмотреть метод **string GetInfo()**, который возвращает строку с информацией о библиотеке (разрешается переопределить **ToString** вместо этого метода);

В самой программе необходимо создать библиотеку и заполнить её **N** (от 10 до 20) случайно сгенерированными книгами (количество страниц может быть от 1 до 500, в библиотеке от 5 до 10 секций). Необходимо вывести все книги, а после этого книги с количеством страниц меньше 200.

Self 02. Арифметическая прогрессия

Реализовать класс **ArithmeticSequence**, представляющий арифметическую прогрессию.
Класс содержит следующие элементы (другие члены класса добавлять разрешено):

Поля
- **double _start –** начальное значение последовательности;
- **double _increment –** разность прогрессии;

Конструкторы:
- **ArithmeticSequence()** – назначает **start** равным нулю, а **increment** равным единице;
- **ArithmeticSequence(double start, double increment) –** инициализирует поля класса значениями параметров;

Индексаторы:
- **double this[int index] –** элемент последовательности с порядковым номером **index (start** имеет порядковый номер 1);

Методы:
- **string GetInfo() –** возвращает строку с информацией о последовафтельности (разрешается переопределить **ToString** вместо этого метода);
- **double GetSum(int n)** - находит сумму первых **n** членов прогрессии.

1. В основной программе создать отдельный объект типа **ArithmeticSequence** и массив из **N** объектов типа **ArithmeticSequence**, где **N** – случайное число из интервала **[5, 15].** Начальное значение последовательности генерировать случайно из диапазона **[0, 1000], a** разность прогрессии из диапазона **[1, 10].**
2. Сгенерировать случайное число **step** из диапазона **[3, 15].**
3. Вывести на экран информацию о последовательностях из массива, у которых элемент с номером **step** больше, чем у отдельной последовательности.
4. Для каждой последовательности из массива вывести сумму первых **step** членов.

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

Self 03. Геометрическая прогрессия

Описать класс **GeometricProgression**, соответствующий геометрической прогрессии.
Класс должен содержать следующие элементы (другие члены класса добавлять разрешено):

Поля:
- **double _start** – начальное значение последовательности;
- **double _increment** – знаменатель прогрессии;

Конструкторы:
- **GeometricProgression()** – назначает start равным нулю, а increment равным единице;
- **GeometricProgression(double start, double increment)** – инициализирует поля класса значениями параметров;

Индексаторы:
- **double this[int index]** – элемент последовательности с порядковым номером index (start имеет порядковый номер 1);

Методы:
- **string GetInfo()** – возвращает строку с информацией о последовательности (разрешается переопределить **ToString** вместо этого метода);
- **double GetSum(int n)** - находит сумму первых **n** членов геометрической прогрессии.

В основной программе создать объект типа **GeometricProgression** и массив из **N** объектов типа **GeometricProgression**, где **N** – случайное число из интервала **[5, 15]**.
Начальное значение последовательности генерировать случайно из диапазона **[0, 10]**, а знаменатель прогрессии из диапазона (0, 5].
Сгенерировать случайное число **step** из диапазона [3, 15]. Вывести на экран информацию о последовательностях из массива, у которых элемент с номером **step** больше, чем у отдельной последовательности.
Для каждой последовательности из массива вывести сумму первых **step** членов.
Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.