Темы урока

[**Список List<T>**](#_1nbaqgkx9q55) **1**

[**Словарь Dictionary<T1, T2>**](#_znkic6381v5) **1**

[**Очередь Queue<T>**](#_y4xdls47ty3h) **2**

[**Стек Stack<T>**](#_h5sagbglk9hz) **2**

[**Домашнее задание**](#_jrgs2cfxvax3) **3**

# Список List<T>

* Класс List<T> представляет простейший список однотипных объектов.
* Определен в неймспейсе System.Collections.Generic.
* Среди методов списка можно выделить следующие:
  + void **Add**(T item): добавление нового элемента в список
  + void **AddRange**(ICollection collection): добавление в список коллекции или массива
  + void **Clear**(): очищает список
  + int **IndexOf**(T item): возвращает индекс первого вхождения элемента в списке
  + void **Insert**(int index, T item): вставляет элемент item в списке на позицию index
  + bool **Remove**(T item): удаляет элемент item из списка, и если удаление прошло успешно, то возвращает true
  + void **RemoveAt**(int index): удаление элемента по указанному индексу index
  + void **Sort**(): сортировка списка
* Примеры работы со списками

## Самостоятельная работа

* Написать приложение, которое будет спрашивать значения типа double до тех пор, пока не введено слово “stop”.
* Когда оно введено необходимо завершить цикл запрашивания значений и рассчитать сумму и среднее арифметическое введенных величин.
* Если введено нечисловое значение
  + перехватить исключение,
  + вывести в консоль сообщение об ошибке и остановке программы,
  + пробросить оригинальный эксепшн с помощью ключевого слова throw.

# Словарь Dictionary<T1, T2>

* Описание
  + Еще один распространенный тип коллекции представляют словари. Словарь хранит объекты, которые представляют пару ключ-значение. Каждый такой объект является объектом структуры **KeyValuePair<TKey, TValue>**.
  + Благодаря свойствам **Key** и **Value**, которые есть у данной структуры, мы можем получить ключ и значение элемента в словаре.
  + Определен в неймспейсе System.Collections.Generic.
  + Имеет большинство методов сходных с методами списков.
  + Для определения есть ли в словаре элемент с заданным ключом, используется метод **HasKey**()
  + Словарь не может хранить элементы с одинаковыми ключами!
* Рассказать, объяснить примеры

## Самостоятельная работа

Написать приложение-игру.

* Программа хранит небольшой список стран и соответствующих им столиц
* Пользователя циклически спрашивают столицу страны в случайном порядке до тех пор, пока он не ошибется
* Если пользователь угадал столицу, его нужно похвалить.
* При ошибке, сообщаем, что пользователь ошибся и выходим из приложения

# Очередь Queue<T>

* Представляет коллекцию объектов, основанную на принципе  
  “первым поступил — первым обслужен” (First In, First Out: FIFO).
* Самый простой способ представить себе принцип “первым поступил, первым обслужен” – представить поезд, въезжающий в тоннель.
  + Сначала туда въедет сам поезд, потом первый вагон, потов второй, и т. д.
  + Выезжать из тоннеля они будут в том же порядке - сначала поезд, потом первый вагон и так далее.
* Очереди используются для случаев, когда не получается спроектировать приложение таким образом, чтобы оно мгновенно возвращало результат.  
  В таком случае, обычно разделяют приложение на две части
  + первая часть приложения формирует и ставит задачи в очередь
  + вторая часть приложения обрабатывает эти задачи и отсылает результат обратно.

## 

## Самостоятельная работа

Написать приложение которое будет запрашивать у пользователя целые числа для отложенного вычисления (по команде) квадратного корня до тех пор, пока пользователь не введет одну из двух команд:

* При вводе команды “run” программа должна вывести на экран расчеты по всем задачам, накопившимся в очереди.
* При вводе команды “exit” программа выводит число оставшихся задач в очереди на момент выхода и завершается.

# Стек Stack<T>

* Представляет коллекцию переменного размера экземпляров одинакового заданного типа, обслуживаемую по принципу "последним пришел — первым вышел" (Last In, First Out: LIFO)
* Стек часто используется для того, чтобы запоминать последовательности для последующего воспроизведения в обратном порядке.
* Хороший пример для объяснения – это стопка тарелок.
  + Когда вы моете посуду, вы составляете вымытые мокрые тарелки в стопку Таким образом вы создаете стек тарелок.
  + Вытирать и убирать вы их будете в обратном порядке - сначала последнюю вымытую тарелку, так как она находится на вершине стека и так далее вниз. Последней будет вытерта тарелка, вымытая первой.

## Самостоятельная работа

Написать приложение, которое будет запрашивать у пользователя одну из трех команд – “wash”, “dry”, или “exit”.

* Если пользователь вводит “**wash**”, то мы кладем в стек очередную “тарелку”.
* Если пользователь вводит “**dry**” мы смотрим, есть ли тарелки в стеке и если есть, то удаляем “тарелку” с вершины стека.
* Если пользователь вводит “**exit**”, завершаем работу программы.
* После ввода каждой команды программа должна выводить количество тарелок в стопке на вытирание.
* Если вы хотите вытереть тарелку, а тарелок в стопке для вытирания нет, выведите сообщение “Стопка тарелок пуста!”

Считаем, что в раковине бесконечное число тарелок :)

# 

# Домашнее задание

1. Написать консольное приложение, которое будет проверять расстановку круглых и квадратных скобок в строке на “правильность” по следующему алгоритму:  
     
   Строка считается корректной, если закрывающаяся скобка соответствует последней открытой, но не закрытой скобке.  
     
   Проверить алгоритм на таких примерах:  
     
   var s1 = "()"; // True  
   var s2 = "[]()"; // True  
   var s3 = "[[]()]"; // True  
   var s4 = "([([])])()[]"; // True  
     
   var s5 = "("; // False  
   var s6 = "[][)"; // False  
   var s7 = "[(])"; // False  
   var s8 = "(()[]]"; // False