Темы урока

Строки в C#

- Тип string представляет последовательность, состоящую из нуля или более символов в кодировке Юникод. string является псевдонимом для типа System.String в .NET

Сравнение

- **==**
- **!=**
- Equals
- Equals с параметром StringComparison.InvariantCultureIgnoreCase

Escape-последовательности (\uXXXX)

- Escape-последовательность строк в C# – это группа символов, которая позволяет закодировать в строке символы, используя порядковый номер символа или “псевдоним” символа.

- Они использоваться в тех случаях, когда другими средствами такой символ закодировать затруднительно. Например, если он является “неотображаемым” или “управляющим”: Enter, Tab, двойная кавычка.

Буквальные строковые литералы (@)

- Вообще строковыми литералами называют строчки, написанные в кавычках в коде “как есть”.
    - Это те самые строковые значения, которые либо сохраняются в переменных, либо просто выводятся на экран. Такие литералы еще называют **регулярными**.
    - Рекомендуется использовать регулярные строковые литералы, когда вы предполагаете использование escape-символов.
- **Буквальные** строковые литералы

Построение строк

Конкатенация строк

- Объединение строк с переменными других типов также возможно с помощью **оператора +**. При объединении переменные других типов будут **автоматически** приведены к строке с помощью метода ToString().

Форматирование строк

- Форматирование позволяет сформировать строку используя шаблон и объекты с возможностью указать формат. Для этого используется метод **String.Format()**.
- Вместо переменных значений используют **шаблоны** с порядковыми номерами в круглых скобках: {0}, {1}, и т.д.
- Можно также указывать формат для некоторых переменных, например можно указать сколько выводить знаков после запятой или в каком формате указывать дату и время. Формат указывается через двоеточие после индекса переменной:
    - {0:#.###}
    - {1:dd.MM.yyyy HH:mm}
    - и т.д. Все виды возможных форматов для различных типов данных можно найти здесь: https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/formatting-types

Интерполяция строк

- Специальный знак $ в начале строки делает интерполированную строку.
- При вычислении интерполированной строки в результирующую элементы заменяются строковыми представлениями результатов выражений.
- Эта возможность доступна в C# 6 и более поздних версиях.

Самостоятельная работа

Написать приложение, которое будет запрашивать у пользователя 2 вещественных (дробных) числа, а затем будет выводит результат математических действий над ними. Строки вывода должны формироваться **различными** методами:

- перемножение – конкатенацией
- сложение – форматированием
- вычитание – интерполяцией

Проверку ошибок входных данных не делаем (будем рассчитывать на корректный ввод данных).

Пример выполнения с результатом:

```
Enter two real numbers to multiply them:
12.6
3.14
12.6 * 3.14 = 39.564
12.6 + 3.14 = 15.74
12.6 - 3.14 = 9.46
```

Решение:

```
Console.WriteLine("Enter two real numbers to multiply them:");
var d1 = double.Parse(Console.ReadLine());
var d2 = double.Parse(Console.ReadLine());
Console.WriteLine(d1 + " * " + d2 + " = " + d1 * d2);
Console.WriteLine("{0} + {1} = {2}",d1, d2, d1 + d2);
Console.WriteLine($"{d1} - {d2} = {d1 - d2}");
```

Поиск по строкам

- Contains
- StartsWith
- EndsWith
- IndexOf
- LastIndexOf
- Рассказать, объяснить примеры.

Модификация строк

- Replace
- Substring
- Split
- ToLower, ToLowerInvariant
- ToUpper, ToUpperInvariant

Очистка строк и проверка на пустоту

- Trim
- TrimStart
- TrimEnd
- IsNullOrEmpty
- IsNullOrWhiteSpace

Построение строк (string.Join)

- Рассказать, объяснить примеры.

Самостоятельная работа на обработку строк

Дана строка с “грязными” пробелами:

```
string text = "   lorem    ipsum    dolor      sit   amet  ";
```

Необходимо произвести над ней следующие операции:

1. “Очистить” исходную строку от лишних пробелов в начале, в конце строки, а также между словами, а также поднять регистр второго слова:
    - “ lorem ipsum dolor sit amet ” → “lorem IPSUM dolor sit amet”
2. Удалить из исходной строки последнее слово и пробелы перед ним:
    - “ lorem ipsum dolor sit amet ” → “ lorem ipsum dolor sit”

Результаты по каждому пункту вывести отдельной строкой.

Пример выполнения с результатом:

```
lorem IPSUM dolor sit amet
   lorem    ipsum    dolor      sit
```

Решение:

```
string text = "   lorem    ipsum    dolor      sit   amet  ";

// part 1:
// split string by words
string[] words = text.Split(' ', StringSplitOptions.RemoveEmptyEntries);

// uppercase for the second word
words[1] = words[1].ToUpperInvariant();

// join words using single space char
Console.WriteLine(string.Join(' ', words));

// part 2:
// remove the spaces at the end of the line
string textClean = text.TrimEnd();

// looking for the position of the last space char
int lastSpaceIndex = textClean.LastIndexOf(' ');

// cutting string from 0 to lastSpaceIndex
textClean = textClean.Substring(0, lastSpaceIndex);

// clean up the tail
textClean = textClean.TrimEnd();

// this is it :)
Console.WriteLine(textClean);
```

Построение строк (StringBuilder)

- Класс StringBuilder позволяет эффективно с точки зрения памяти создавать и модифицировать длинные строки.
- Располагается в области видимости **System.Text**, не забывайте подключать её используя using System.Text; в начале файла.
- Основные методы:
    - **Append()** – добавляет аргумент, при необходимости конвертированный в строку, **к концу** внутренней строчки
    - **AppendFormat()** – добавляет шаблонизированную строку **к концу** внутренней строки, позволяет передать сразу шаблон форматирования и дополнительные параметры. Является просто сокращенной записью от Append(string.Format())
    - **Insert()** – позволяет вставить **в произвольное место** внутренней строки переданную параметром строку или переменную, приведенную к строке.
    - **Remove()** – позволяет удалить заданное количество символов внутренней строки начиная с любого места.
    - **Replace()** – аналогично string.Replace – ищет и в случае нахождения изменяет искомую строку заданной.
    - **ToString()** – строит и возвращает внутреннюю строку, когда создание завершено.
    - **Length** - возвращает текущую длину внутренней строки.

Самостоятельная работа (сложная)

Дана строка с “грязными” пробелами:

```
string text = "   lorem    ipsum    dolor      sit   amet  ";
```

Тут **только первая часть от предыдущего задания**, но сделать ее надо **перебором** каждого символа исходной строки (**в цикле** for или foreach) и “набора” результирующей строки в **StringBuilder**: Необходимо произвести над ней следующие операции:

1. “Очистить” исходную строку от лишних пробелов в начале, в конце строки, а также между словами, а также поднять регистр второго слова:
    a. “ lorem ipsum dolor sit amet ” → “lorem IPSUM dolor sit amet”

Пример выполнения с результатом:

```
lorem IPSUM dolor sit amet
```

Решение:

```
string text = "   lorem    ipsum    dolor      sit   amet  ";

StringBuilder sb1 = new StringBuilder();

int wordIndex = 0;
foreach (char c in text)
{
      // for the first result we should remove all the spaces at the
beginning.
      // other words, we will copy the first letter only if it is not a
space.
      if (sb1.Length == 0 && c != ' ')
      {
            sb1.Append(c); // just copy char to sb1
            wordIndex = 1;
      }
      // else if it is not the first symbol
      else if (sb1.Length > 0)
      {
            // if it is not a space
            if (c != ' ')
            {
                  // add to sb1 just c or char.ToUpper(c)
                  sb1.Append(wordIndex == 2 ? char.ToUpper(c) : c);
            }

            // if (current char is a space and at the same time
            // the last copied char to sb1 wasn't a space)
            if (c == ' ' && sb1[sb1.Length - 1] != ' ')
            {
                  wordIndex++; // increase the actual word number
                  sb1.Append(c); // add this space
            }
      }
```

```
}

// after we finish the cycling,
// at the end of sb1 we can receive the space,
// so let's check whether last char is a space
if (sb1[sb1.Length - 1] == ' ')
{
      // remove it if it's true
      sb1.Remove(sb1.Length - 1, 1);
}

Console.WriteLine(sb1.ToString());
```

Домашнее задание

1. Написать консольное приложение, которое запрашивает строку и выводит количество слов, начинающихся на букву А.

   Программа должна спрашивать исходную строку до тех пор, пока пользователь не введёт хотя бы 2 слова.

   Пример работы программы:
   ```
   > Введите строку из нескольких слов:
   > тест /это ввод пользователя/
   > Слишком мало слов :( Попробуйте ещё раз:
   > Антон купил арбуз. Алый мак растет среди травы. /это ввод пользователя/
   > Количество слов, начинающихся с буквы 'А': 3.
   > Нажмите любую клавишу для выхода…
   ```

2. Написать консольное приложение, которое запрашивает строку, а затем выводит все буквы приведенные к нижнему регистру в обратном порядке.

   Программа должна спрашивать исходную строку до тех пор, пока пользователь не введет строку, содержащую печатные символы.

   Пример работы программы:
   ```
   > Введите непустую строку:
   >      /это ввод пользователя/
   > Вы ввели пустую строку :( Попробуйте ещё раз:
   > Не до логики, голоден /это ввод пользователя/
   > недолог ,икигол од ен
   > Нажмите любую клавишу для выхода...
   ```

- Не забывать обрабатывать все предсказуемые исключения.