Блок 1. Базовый C#

1.3 — Строки

План занятия

- Символы
- Строки
- StringBuilder
- Итернирование

Символы

- Символы в .NET представлены структурами **Char**, содержащими 16-битное беззнаковое число: код символа из таблицы Unicode;
- Некоторые символы таблицы Unicode (например, суррогатные пары) не вмещаются в 16 бит, они представляются в виде пары структур Char;
- Для создания объекта символа применяются апострофы и один из способов:
  - По образцу:
    ```
    char a = 'A';
    ```
  - По шестнадцатеричному коду:
    ```
    char b = '\0x41';
    ```
  - При помощи escape-последовательности:
    ```
    char c = '\n';
    ```
- К символам применимы операции сложения и вычитания.

Ключевые методы типа Char

Метод	Действие
IsDigit()	Символ является цифрой
IsLetter()	Символ является буквой
IsNumber()	Символ является числом
IsPunctuation()	Символ является знаком пунктуации
IsSeparator()	Символ является разделителем
IsLower()	Символ относится к нижнему регистру
IsUpper()	Символ относится к верхнему регистру
ToLower()	Преобразование к нижнему регистру
ToUpper()	Преобразование к верхнему регистру

```
bool b1 = char.IsDigit('1'); // true
char c1 = char.ToUpper('t'); // T
bool b2 = char.IsPunctuation("Это строка", 3); // false
```

Строки

- Представляют собой неизменяемые последовательности символов;
- Не существует способа изменить строку или её часть;
- За хранение отвечает класс String;
- Для создания объекта строки применяются следующие способы:
  - Явное создание по образцу при помощи кавычек;
  - Конструктор класса String;
  - Метод ToString() объекта любого класса.

Способы создания строки

```
// объявление с инициализацией значением Сайт
string s1 = "Сайт";

// строка в куче из пяти повторяющихся символов
string s2 = new string('s', 5);

string s3_1 = new string("s1"); // так нельзя
string s3_2 = new string(); // так тоже нельзя
// а так можно:
string s3; // это и есть объявление пустой строки в куче

// преобразование в символьный массив явно заданной строки
char[] chars = "Привет посетителям портала!".ToCharArray();

// объявление строки с инициализацией символами из массива
string s4 = new string(chars);

// объявление с копированием подстроки Привет
// (с нулевого шесть символов)
string s5 = new string(chars, 0, 6);
```

Ключевые свойства и методы

Метод	Действие
Length	Свойство, возвращает длину строки.
Contains()	Проверяет, содержится ли в строке подстрока
Format()	Позволяет задать форматирование строки.
Insert()	Вставляет в строку подстроку
Remove()	Удаляет из строки символы
Replace()	Заменяет в строке все вхождения подстроки на какой-либо другой текст
Substring()	Вырезает из строки подстроку
StartsWith()	Проверяет, начинается ли строка с подстроки
EndsWith()	Проверяет, заканчивается ли строка на подстроку

Ключевые свойства и методы

Метод	Действие
ToCharArray()	Конвертирует строку к массиву символов
ToUpper()	Переводит все символы строки в верхний регистр (в т. ч. символы национальных алфавитов).
ToLower()	Переводит все символы строки в нижний регистр. Корректно работает в том числе и с русскими символами.
Trim()	Удаляет из начала и конца строки пробельные, либо другие спецсимволы
IndexOf()	Определяет номер позиции первого вхождения подстроки в строку
LastIndexOf()	Определяет номер позиции последнего вхождения подстроки в строку

Строки – неизменяемые объекты!

- Потокобезопасность
- Неизменность
- Сокращение затрат памяти (интернирование)

```
string s = "Hello world!";
s.Replace(' ', '_');
Console.WriteLine(s);
```

```
string s = "Hello world!";
s = s.Replace(' ', '_');
Console.WriteLine(s);
```

Неправильная модификация строк

- При каждой модификации строки создается отдельный объект
- Сколько строк будет создано?
- Сколько памяти будет занято?

```
int n = 10000;
string str = string.Empty;
for (int i = 0; i < n; i++)
{
    str += "*";
}
```

```
int n = 10000;
string str = new string('*', n);
```

Правильная модификация строк

- Для сложения большого числа разных строк используйте класс StringBuilder:

```
int n = 10000;
var sb = new StringBuilder("1");
for (int i = 2; i <= n; i++)
{
    sb.Append(", ").Append(i.ToString());
}
string s = sb.ToString();
```

Сравнение String и StringBuilder

Число слов	Длина	String		StringBuilder	
		Время	Затраты памяти	Время	Затраты памяти
3	15	0.163	30	0.22	16
4	20	0.252	50	0.373	48
5	25	0.336	75	0.39	48
6	30	0.464	105	0.463	48
7	35	0.565	140	0.591	112
15	75	1.779	600	1.125	240
20	100	2.697	1050	1.354	240
25	125	3.811	1625	1.571	240
50	250	11.45	6375	3.03	496
90	450	32.13	20475	5.419	1008

Методы и свойства класса StringBuilder

Метод	Действие
Length	Длина строки
Capacity	Емкость строки
Append()	Добавляет строку к текущей строке.
AppendFormat()	Добавляет форматированную строку
Insert()	Вставляет подстроку в строку
Remove()	Удаляет символы из текущей строки
Replace()	Заменяет в строке все вхождения подстроки на какой-либо другой текст
ToString()	Преобразует объект в строку

Когда сложение строк – не преступление

- Литеральные строки

```
string s = "Это " +
    "длинная текстовая " +
    "строка " +
    "которую не удобно " +
    "записать в одной " +
    "строке программы.";
```

```
string s = "Это длинная текстовая строка которую не удобно записать
```

Буквальные строки (оператор @)

Строка, помеченная @, воспринимается буквально, без учета управляющих символов:

```
string s1 = "c:\\temp\\myfile.txt";
string s2 = @"c:\temp\myfile.txt";

string s3 = "Это текст\r\nна несколько\r\nстрок";
string s4 = @"Это текст
на несколько
строк";
```

Использование string.Format

```
// Спорно
s = s1 + "[" + s2 + "]=" + s3;

// Предпочтительно
s = string.Format("{0}[{1}]={2}", s1, s2, s3);
```

```
public static string Format(IFormatProvider provider, string format,
{
    if (format == null || args == null)
    {
        throw new ArgumentNullException((format == null) ? "format"
    }
    StringBuilder stringBuilder = new StringBuilder(format.Length +
    stringBuilder.AppendFormat(provider, format, args);
    return stringBuilder.ToString();
```

Итернирование

- *Интернирование строк* — это механизм, при котором одинаковые литералы представляют собой один объект в памяти.

Итернирование

```
var s1 = “HelloWorld";
var s2 = " HelloWorld ";
var s3 = " Hello" + " World ";
Console.WriteLine(object.ReferenceEquals(s1, s2));//true
Console.WriteLine(object.ReferenceEquals(s1, s3));//true
```

Итернирование

- **public static String Intern(String str);**
- **public static String IsInterned(String str);**

Демонстрация

Спасибо за внимание!

Контактная информация:
Дмитрий Вересkун
Инструктор
EPAM Systems, Inc.
Адрес: Саратов, Рахова, 181
Email: Dmitry_Vereskun@epam.com
http://www.epam.com