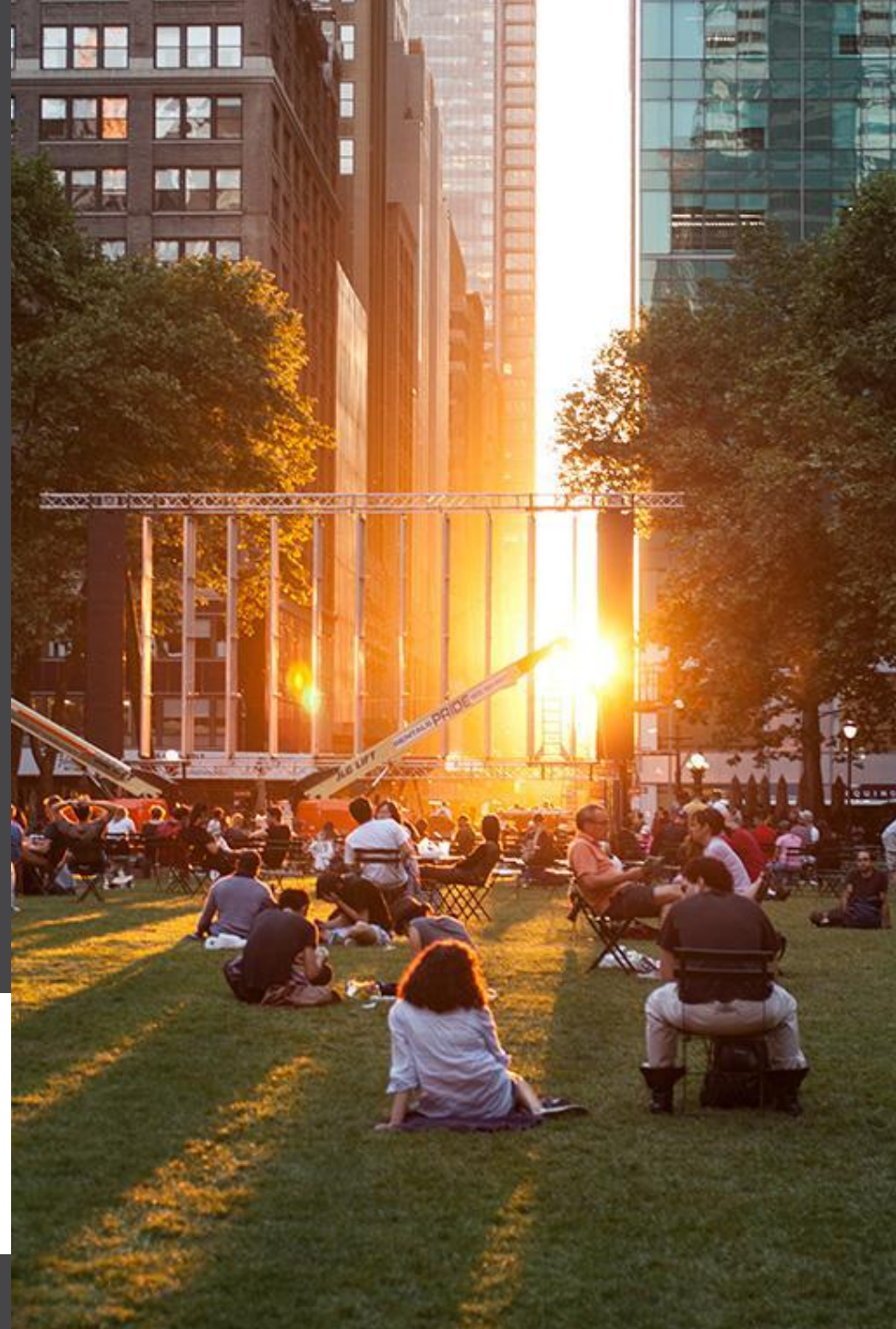
C#

Strings

04

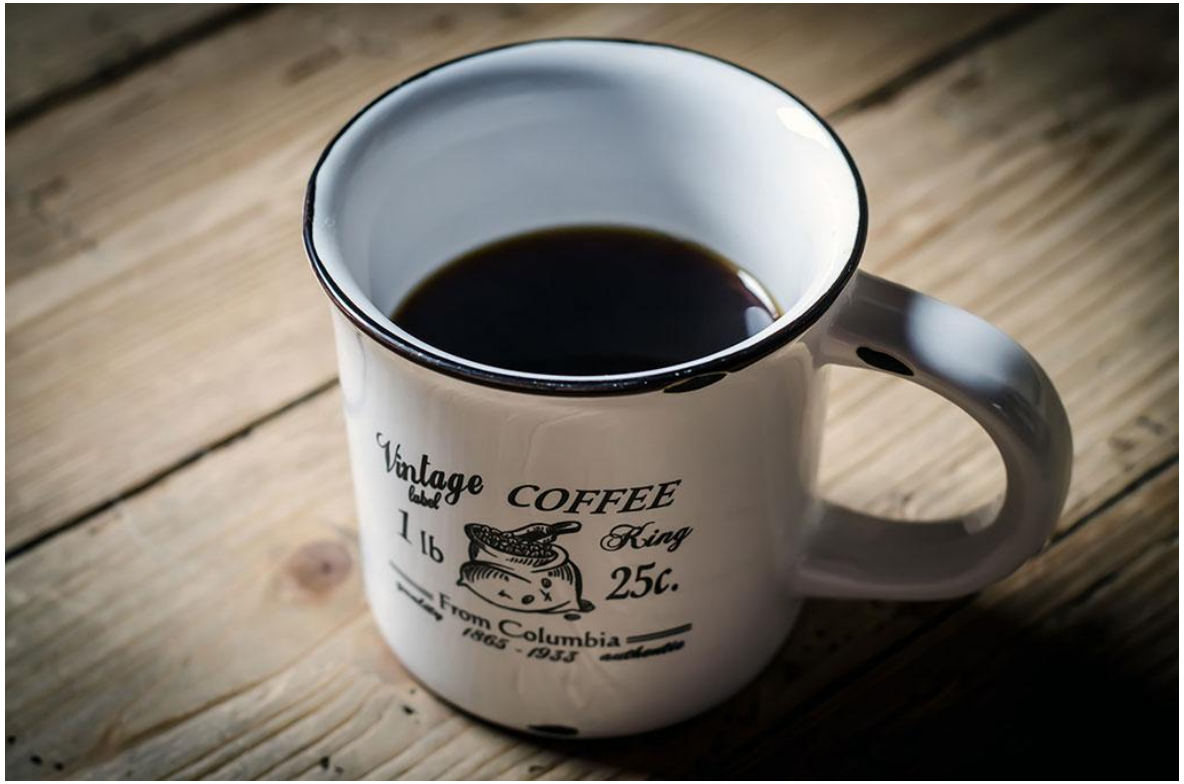


EXTERNAL TRAINING .NET/WEB

EPAM Saratov · AUTUMN 2019

ПЛАН ЗАНЯТИЯ

- Символы
- Строки
- Форматирование строк



СИМВОЛЫ

- Символы в .NET представлены структурами **`Char`**, содержащими 16-битное беззнаковое число: код символа из таблицы Unicode;
- Некоторые символы таблицы Unicode (например, суррогатные пары) записываются последовательностью из нескольких структур **`Char`**;
- Для создания объекта символа применяются апострофы и один из способов:
  - По образцу:
    ```
    char a = 'A';
    ```
  - По шестнадцатеричному коду:
    ```
    char b = '\u0041';
    ```
  - Escape-последовательность (управляющий символ):
    ```
    char c = '\n';
    ```
  - Приведение числа — кода символа — к типу **`char`**:
    ```
    char c = (char)65;
    ```
- К символам применимы операции сложения и вычитания.

УПРАВЛЯЮЩИЕ СИМВОЛЫ

Символ	Значение
\n	Перевод строки
\r	Возврат каретки
\t	Горизонтальная табуляция
\\	Обратная косая черта
\'	Апостроф
\"	Кавычки
\a	Звуковой сигнал
\b	Возврат на шаг назад
\f	Перевод страницы
\v	Вертикальная табуляция

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;
- Строка может содержать суррогатные пары, записанные при помощи управляющей последовательности **`\U`** (например, **`"\U0001D11E"`**).

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 (число объектов **`char`**)
`Contains()`	Проверяет, содержится ли в строке подстрока
`Compare()`	Сравнивает две строки (меньше/больше/равны)
`Equals()`	Сравнивает две строки на равенство
`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

КЛЮЧЕВЫЕ СВОЙСТВА И МЕТОДЫ

Метод	Действие
`StartsWith()`	Проверяет, начинается ли строка с подстроки
`EndsWith()`	Проверяет, заканчивается ли строка на подстроку
`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**

		ВРЕМЯ ВЫПОЛНЕНИЯ		ЗАТРАТЫ ПАМЯТИ	
Число слов	Длина	String	StringBuilder	String	StringBuilder
3	15	0.163	0.22	60	32
4	20	0.252	0.373	100	96
5	25	0.336	0.39	150	96
6	30	0.464	0.463	210	96
7	35	0.565	0.591	280	224
15	75	1.779	1.125	1 200	480
20	100	2.697	1.354	2 100	480
25	125	3.811	1.571	3 250	480
50	250	11.45	3.03	12 750	992
90	450	32.13	5.419	40 950	2 016

МЕТОДЫ И СВОЙСТВА КЛАССА **STRINGBUILDER**

Метод	Действие
`Length`	Длина строки
`Capacity`	Емкость строки
`Append()`	Добавляет строку или текстовое представление объекта
`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 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 в од
```

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";
```

Поскольку \ считается обычным символом, escape-последовательности не работают.
Для вывода знака кавычки " её следует задублировать: ""

ФОРМАТИРОВАНИЕ СТРОК

```
// 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();
```

ПАРАМЕТРЫ ФОРМАТИРОВАНИЯ

Параметр	Формат
C, c	Финансовый ($, €, ₽)
D, d	Целочисленный
E, e	Экспоненциальный (научный)
F, f	Вещественный
G, g	Общий числовой
N, n	Стандартное форматирование
P, p	Процентный
X, x	Шестнадцатеричный

ПРИМЕР ФОРМАТИРОВАНИЯ

```
Console.WriteLine("C:{0,14:C}\t{0:C4}", 12345.678);
Console.WriteLine("D:{0,14:D}\t{0:D6}", 123);
Console.WriteLine("E:{0,14:E}\t{0:E8}", 12345.6789);
Console.WriteLine("G:{0,14:G}\t{0:G10}", 12345.6789);
Console.WriteLine("N:{0,14:N}\t{0:N4}", 12345.6789);
Console.WriteLine("X:{0,14:X}", 1234);
Console.WriteLine("P:{0,14:P}", 0.9);
```

```
C:   12 345,68 p.        12 345,6780 p.
D:            123        000123
E: 1,234568E+004        1,23456789E+004
G:     12345,6789        12345,6789
N:      12 345,68        12 345,6789
X:            4D2
P:         90,00%
```

НАСТРОЙКИ ФОРМАТИРОВАНИЯ

Описатель формата	Имя
0	Знак-заместитель нуля
#	Заместитель цифры
.	Разделитель
,	Разделитель групп и масштабирование чисел
'строка'	Разделитель строк-литералов
;	Разделитель секций

ПРИМЕР РАСШИРЕННОГО ФОРМАТИРОВАНИЯ

```
Console.WriteLine("{0:plus ####.0000;minus 0000.####;zero}",
    123.456);
Console.WriteLine("{0:plus ####.0000;minus 0000.####;zero}",
    -123.456);
Console.WriteLine("{0:plus ####.0000;minus 0000.####;zero}",
    0.0);
Console.WriteLine("{0,50:#### тысяч 000 целых.###}",
    2222123.4);
Console.WriteLine("{0,-50:#### тысяч 000 целых.###}",
    123.4896318);
```

```
plus 123,4560
minus 0123,456
zero
                                          2222 тысяч 123 целых,4
 тысяч 123 целых,49
```

ПРИМЕР ФОРМАТИРОВАНИЯ ДАТЫ

```
DateTime dt = new DateTime(2011, 3, 9, 16, 5, 7, 123);
Console.WriteLine("t: {0:t}", dt);
Console.WriteLine("T: {0:T}", dt);
Console.WriteLine("d: {0:d}", dt);
Console.WriteLine("D: {0:D}", dt);
Console.WriteLine("f: {0:f}", dt);
Console.WriteLine("F: {0:F}", dt);
Console.WriteLine("g: {0:g}", dt);
Console.WriteLine("G: {0:G}", dt);
Console.WriteLine("m: {0:m}", dt);
Console.WriteLine("y: {0:y}", dt);
Console.WriteLine("o: {0:o}", dt);
Console.WriteLine("s: {0:s}", dt);
Console.WriteLine("u: {0:u}", dt);
Console.WriteLine("U: {0:U}", dt);
```

```
t: 16:05
T: 16:05:07
d: 2011-03-09
D: 9 марта 2011 г.
f: 9 марта 2011 г. 16:05
F: 9 марта 2011 г. 16:05:07
g: 2011-03-09 16:05
G: 2011-03-09 16:05:07
m: 9 марта
y: Март 2011
o: 2011-03-09T16:05:07.1230000
s: 2011-03-09T16:05:07
u: 2011-03-09 16:05:07Z
U: 9 марта 2011 г. 12:05:07
```

НАСТРОЙКИ ФОРМАТИРОВАНИЯ ДАТЫ

Описатель формата	Описание
d dd ddd dddd	День
f ff fff ffff	Доли секунд
F FF FFF FFFF	Доли секунд (без нулей)
h hh	Часы в 12-часовом формате
H HH	Часы в 24-часовом формате
m mm	Минуты
M MM	Месяц
s ss	Секунды
y yy yyy yyyy	Год
Z ZZ ZZZ	Смещение времени

ПРИМЕР НАСТРОЙКИ ФОРМАТИРОВАНИЯ ДАТЫ

```
DateTime dt = new DateTime(2008, 3, 9, 16, 5, 7, 123);
Console.WriteLine("{0:\\y-\t y yy yyy yyyy}", dt);
Console.WriteLine("{0:\\M-\t M MM MMM MMMM}", dt);
Console.WriteLine("{0:\\d-\t d dd ddd dddd}", dt);
Console.WriteLine("{0:\\h\\H-\t h hh H HH}", dt);
Console.WriteLine("{0:\\m-\t m mm}", dt);
Console.WriteLine("{0:\\s-\t s ss}", dt);
Console.WriteLine("{0:\\f-\t f ff fff ffff}", dt);
Console.WriteLine("{0:\\F-\t F FF FFF FFFF}", dt);
Console.WriteLine("{0:\\z-\t z zz zzz}", dt);
```

```
y-      8 08 2008 2008
M-      3 03 мар Март
d-      9 09 Вс воскресенье
hH-     4 04 16 16
m-      5 05
s-      7 07
f-      1 12 123 1230
F-      1 12 123 123
z-      +4 +04 +04:00
```

ИНТЕРПОЛЯЦИЯ СТРОК

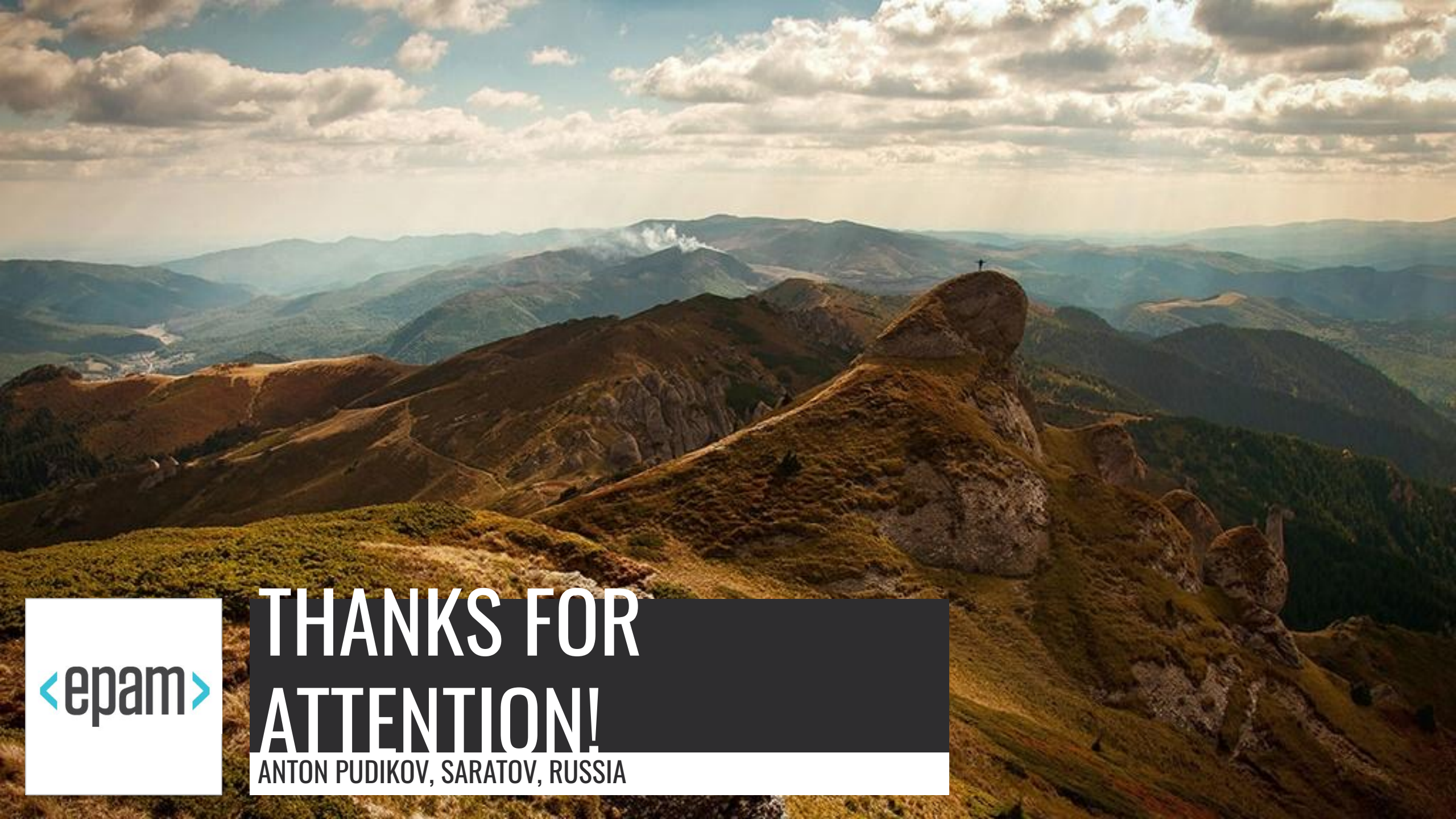
C# 6

```
string s1 = string.Format("x = {0}, y = {1}", p.X, p.Y);

string s2 = $"x = {p.X}, y = {p.Y}";
```

Если необходимо вывести в строку фигурные скобки, их нужно задублировать: **{{..}}**

Интерполяция совместима с буквальными строками: **$@"C:\{fileName}.txt"**



THANKS FOR ATTENTION!

ANTON PUDIKOV, SARATOV, RUSSIA