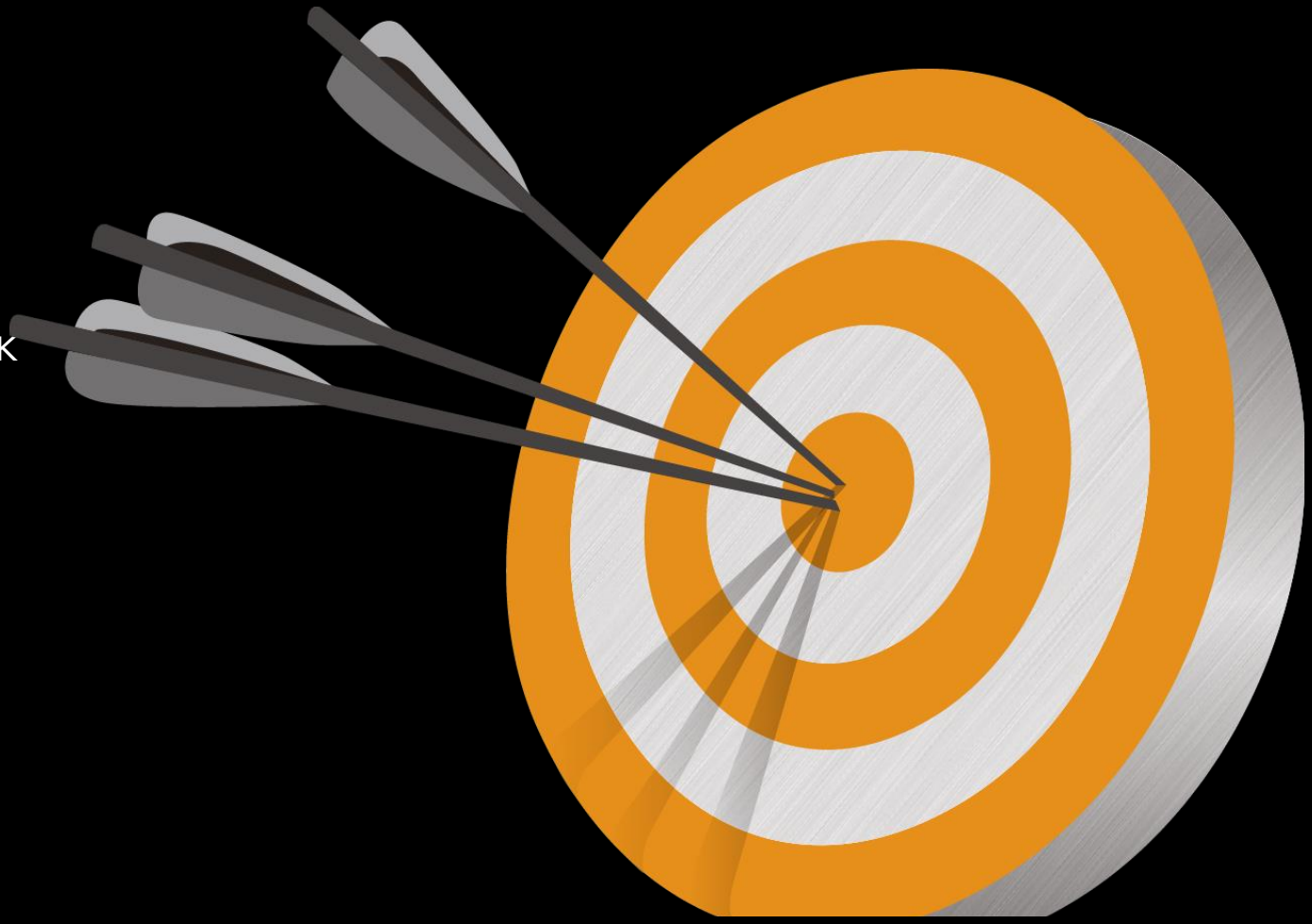
ЛЕКЦИЯ 10

- 04.10.2023
- Строки

ЦЕЛИ ЛЕКЦИИ

- Разобраться с реализацией строк в C# (класс String)
- Изучить методы работы со строками в C#



СТРОКИ В C#

Строка (string, System.String) – это последовательная **неизменяемая** (immutable) коллекция символов (типа char) в кодировке UTF-16, используемая для представления текста

- **Строковые литералы в C# записываются в двойных кавычках:**

```
string line = "This is a string.";
```

Максимальный размер строки в C# – 2Гб или в около миллиарда символов

ОБЩАЯ ИНФОРМАЦИЯ О СТРОКАХ C#

- Строки в C# являются **ссылочными типами**
- **В конце строки нет терминального символа ‘\0’**
- **Строка может содержать внутри себя произвольное количество символов ‘\0’**
- Свойство `Length` возвращает общее количество объектов `Char` в строке

`string` – псевдоним **`System.String`**

НЕИЗМЕНЯЕМОСТЬ СТРОК

```
string str = "Let eat";
str += " bee!";
Console.Write(str);
```

Память, выделенная под это будет доступна для сборщика мусора после выполнения назначения ссылки в следующей строке

В этой строке неявно создан новый экземпляр для хранения строки после конкатенации

ОПЕРАЦИЯ ИНДЕКСАЦИИ И СТРОКИ

- Индексы начинаются с 0
- Доступ для чтения к отдельным символам строки (т.е. может быть только r-value)
- В итерировании по строке, опять же с доступом для чтения

```
string str = "Let eat";
str += " bee!";
str[5] = 'B';
Console.Write(str);
```

```
string str = "Let eat";
str += " bee!";
Console.Write(str[5]);
```

ОПИСАНИЕ И ИНИЦИАЛИЗАЦИЯ СТРОК

```
// Описание без инициализации
        string message1;
        // Инициализация значением null
        string message2 = null;
        // Инициализация пустой строкой
        string message3 = System.String.Empty;
        // Инициализация строковым литералом
        string oldPath = "c:\\Program Files\\Microsoft Visual Studio 8.0";
        // Инициализация буквальным строковым литералом
        string newPath = @"c:\Program Files\Microsoft Visual Studio 9.0";
        // Использование анонимного типа, внутри методов
        var temp = "I'm still a strongly-typed System.String!";
        // Константная строка
        const string message4 = "You can't get rid of me!";
        // Явное использование конструктора String()
        // для создание строки из char*, char[] или sbyte*
        char[] letters = { 'A', 'B', 'C' };
        string alphabet = new string(letters);
```

Вместо “”

Буквальный строковый литерал

escape-последовательности внутри буквальных строковых литералов **не обрабатываются**

СОЗДАНИЕ ОБЪЕКТОВ ТИПА STRING

```
string line1 = "Это строка 1";
char[] chars = { 'М', 'а', 'с', 'с', 'и', 'в' };
string line2 = new string(chars);                    // "Массив"
string line3 = new string('W', 1);                   // "W"
string line4 = new string('7', 3);                   // "777"
string line5 = 242.ToString();                       // "242"



string line;
line += "asdfg"; // Ошибка компиляции – line не инициализирована.
```

ПУСТЫЕ И NULL-СТРОКИ

- Методы работы со строками можно вызывать для пустых строк, так как они допустимы System.String
- Строка NULL не ссылается на экземпляр System.String объекта, и любая попытка вызова метода в строке NULL вызывает исключение NullReferenceException.
  - Допустимо использовать строки NULL в операциях объединения и сравнения с другими строками

ПРИМЕР. ПУСТЫЕ И NULL-СТРОКИ

```
string str = "hello";
string nullStr = null;
string emptyStr = String.Empty;
```

```
string tempStr = str + nullStr;  // Output of the following line: hello

bool b = (emptyStr == nullStr);  // Output of the following line: False

// The following line creates a new empty string.
string newStr = emptyStr + nullStr;

// Null strings and empty strings behave differently. The following
// two lines display 0.
Console.WriteLine(emptyStr.Length);
Console.WriteLine(newStr.Length);
// The following line raises a NullReferenceException.
//Console.WriteLine(nullStr.Length);
```

ПУСТЫЕ И NULL-СТРОКИ – ПРИМЕР 2

```
 // The null character can be displayed and counted, like other chars.
string s1 = "\x0" + "abc";
string s2 = "abc" + "\x0";
// Output of the following line: * abc*
Console.WriteLine("*" + s1 + "*");
// Output of the following line: *abc *
Console.WriteLine("*" + s2 + "*");
// Output of the following line: 4
Console.WriteLine(s2.Length);
```

ИНДЕКСАЦИЯ СТРОК

Строки полностью поддерживают индексы и диапазоны:

```
using System;

string indexingDemo = "Yet another string...";
Console.WriteLine(indexingDemo[4]);
Console.WriteLine(indexingDemo[^4]);
Console.WriteLine(indexingDemo[4..]);
Console.WriteLine(indexingDemo[..^18]);
Console.WriteLine(indexingDemo[4..11]);
```

Вывод:

a
g
another string...
Yet
another

КОНКАТЕНАЦИЯ СТРОК

- Конкатенация (объединение) строк может осуществляться с помощью операций + и +=:

```
string strConcat = "Hello";
strConcat = strConcat + " Software";
strConcat += " Engineering!";
Console.WriteLine(strConcat);
```

Вывод:
Hello Software
Engineering!

Строки в C# неизменяемы и конкатенация всегда ведёт к созданию новой строки. По этой причине крайне НЕ рекомендуется выполнять конкатенацию в цикле

СВЯЗЬ СТРОК И ССЫЛОК НА НИХ

```
string first = "Foo";
string second = first;
first += "Bar";
Console.Write($"first:: {first} second:: {second}");
```

Что выведет этот код?

СТРОКА КАК КОНТЕЙНЕР СИМВОЛОВ

- Так как строка фактически является неизменяемым контейнером для символов, её можно обходить посимвольно с помощью циклов for и foreach:

```
using System;
string line1 = "example1";
for (int i = 1; i <= line1.Length; ++i) {
    Console.Write(line1[^i] + " ");
}
Console.WriteLine();
string line2 = "0123";
foreach (char letter in line2) {
    Console.Write(letter + "->" + (int)letter + " ");
}
string incorrect = "example3";
incorrect[0] = 'E'; // Ошибка компиляции - строка неизменяема.
```

Обход с использованием индексации с конца.

Вывод:

1 e l p m a x e

0->48 1->49 2->50 3->51

МАССИВЫ СТРОК

Как и в случае с другими типами в C#, допустимо создавать массивы строк:

```
string[] lines = new string[] { "ноль", "один", "два", "три",
                "четыре", "пять", "шесть", "семь", "восемь", "девять"};
```

```
using System;
string[] starArray = new string[4];
for (int i = 0; i < starArray.Length; i++)
{
    starArray[i] = new string('*', i + 1);
}
foreach (string startSet in starArray)
{
    Console.Write("\t" + starSet);
}
```

Вывод:

```
*     **     ***     ****
```

ПАРАМЕТР MAIN() – МАССИВ СТРОК

У Main() может быть параметр – массив строк, передаваемый приложению в момент запуска:

```
class Program {
    static void Main(string[] args) {
        int sum = 0;
        if (args.Length == 0) {
            System.Console.WriteLine("Нет аргументов в командной строке!");
            return;
        }
        for (int i = 0; i < args.Length; i++) {
            sum += int.Parse(args[i]);   // Осторожно: возможны исключения.

        System.Console.WriteLine("Сумма чисел = " + sum);
    }
}
```

При желании допускается указать параметры командной строки в Visual Studio: Project → имя_проекта Properties → Debug → Command line arguments

СТРОКИ В SWITCH-ВЫРАЖЕНИЯХ

- Аналогично другим типам данных строки могут использоваться в выражениях switch:

```
static string TranslateDay (string day) => day.ToLower() switch
{
    "monday" => "понедельник",
    "tuesday" => "вторник",
    "wednesday" => "среда",
    "thursday" => "четверг",
    "friday" => "пятница",
    "saturday" => "суббота",
    "sunday" => "воскресенье",
    _ => throw new ArgumentException("This is not a valid day.");
};
```

ФОРМАТИРОВАНИЕ И ИНТЕРПОЛЯЦИЯ СТРОК: ПРИМЕР

```
string s = "Hello there.";
System.Console.WriteLine($"{s.ToUpper()}");      // К верхнему регистру.
System.Console.WriteLine("{0}", s);              // Исходная строка.
```

Вывод:
HELLO THERE.
Hello there.

```
using System;

string myPet = "Spot";
int age = 4;
string color = "black and white";
Console.WriteLine("My dog's name is {0}. He is {1} years old. His color is {2}.", myPet, age, color);
Console.WriteLine($"My dog's name is {myPet}. He is {age} years old. His color is {color}.");
```

Вывод:
My dog's name is Spot. He is 4 years old. His color is black and white.
My dog's name is Spot. He is 4 years old. His color is black and white.

СПЕЦИФИКАЦИЯ ФОРМАТА СТРОК

C# поддерживает форматирование строк через специальный синтаксис с фигурными скобками:

```
{ index[,alignment][:formatString[precisionSpecifier]] }
```

- `index` – номер аргумента;
- `alignment` – ширина поля;
- `formatString` – спецификатор формата;
- `precisionSpecifier` – спецификатор точности.

```
using System;
int num = 23, den = 6;
string result, // ссылка на строку, с результатом
form = "Числитель: {0}, знаменатель: {1}, дробь: {0}/{1} == {2}";
result = string.Format(form, num, den, num / den);
Console.WriteLine(result);
```

Вывод:
Числитель: 23, знаменатель: 6, дробь: 23/6 == 3

СПЕЦИФИКАТОРЫ ФОРМАТА И ТОЧНОСТИ

Спецификатор	Формат	Роль спецификатора точности
C или c	Валютный	Количество десятичных разрядов.
D или d	Целочисленный	Минимальное число цифр.
E или e	Экспоненциальный	Число разрядов после точки.
F или f	С фиксированной точкой	Число разрядов после точки.
G или g	Наиболее короткий между E и F	Аналогично E и F
N или n	Численный с отступами	Минимальное число цифр.
P или p	Процентный	Количество десятичных разрядов.
R или r	Строка, посимвольно равная такому же числу	Игнорируется.
X или x	Шестнадцатеричный	Минимальное число цифр.

https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-

ПРИМЕР ФОРМАТИРОВАНИЯ (1)

```
double dou = 1234.567;
string form = "Спецификация E4: {0:E4}, спецификация F4: {0:F4}";
string result = string.Format(form, dou);
System.Console.WriteLine(result);
```

Вывод:
Спецификация E4: 1,2346E+003, спецификация F4: 1234,5670

```
int num = 23;
string form = "Основание 10: {0,-4:D}\r\nОснование 16: {0,4:X}";
System.Console.WriteLine(form, num);
```

Вывод:
Основание 10: 23
Основание 16:   17

```
int num = 23;
string form = "Основание 10: {0,-4:D3}\r\nОснование 16: {0,4:X3}";
System.Console.WriteLine(form, num);
```

Вывод:
Основание 10: 023
Основание 16: 017

ПРИМЕР ФОРМАТИРОВАНИЯ 2

```
double completion = 0.57;
string form = "Процесс загрузки: {0:P4}";
System.Console.WriteLine(form, completion);
```

Вывод:
Процесс загрузки: 57,0000 %

```
decimal currency = 2365700000;
string form = "На Вашем счету: {0,3:C2}";
System.Console.WriteLine(form, currency);
```

Вывод:
На Вашем счету: 2 365 700 000,00 ₽

ЧЛЕНЫ КЛАССА STRING (1)

Член	Тип	Краткое описание
Length	Свойство	Возвращает количество символов в данной строке.
Contains	Метод	Возвращает true, если указанная подстрока присутствует в данной строке, иначе false.
ToLower	Метод	Возвращает копию строки, все символы которой переведены в нижний регистр.
ToUpper	Метод	Возвращает копию строки, все символы которой переведены в верхний регистр.
Insert	Метод	Возвращает копию строки, для которой в заданное место вставлена указанная строка.
Remove	Метод	Возвращает копию строки, из которой удалена указанная подстрока.
Replace	Метод	Возвращает копию строки, в которой заданная подстрока заменена другой подстрокой.
Substring	Метод	Возвращает обрезанную с заданной позиции копию строки.

ЧЛЕНЫ КЛАССА STRING (2)

Член	Тип	Краткое описание
Split	Метод	Возвращает массив строк, полученный путём деления указанной строки на подстроки с использованием заданных разделителей.
Join	Статический Метод	Объединяет несколько строк в одну через указанный разделитель.
Concat	Статический Метод	Производит конкатенацию двух и более строк, может конкатенировать строковое представление произвольных объектов.
Trim	Метод	Возвращает копию строки, из которой удалены все вхождения указанного(ых) символа(ов) в начале и в конце.
Format	Статический Метод	Возвращает отформатированную строку (подробнее).
Compare	Статический Метод	Осуществляет сравнение строк по заданным правилам.
Equals	Метод	Осуществляет проверку строк на равенство по заданным правилам.
ToCharArray	Метод	Копирует символы данной строки в массив символов.

ЧЛЕНЫ КЛАССА STRING (3)

Член	Тип	Краткое описание
IndexOf	Метод	Возвращает индекс первого вхождения подстроки в строку или -1.
LastIndexOf	Метод	Возвращает индекс последнего вхождения подстроки в строку или -1.
IndexOfAny	Метод	Возвращает индекс первого вхождения одного из перечисленных символов или -1.
StartsWith	Метод	Возвращает true, если указанная строка начинается с заданного символа/строки или false в противном случае.
EndsWith	Метод	Возвращает true, если указанная строка заканчивается заданным символом/строкой или false в противном случае.
PadLeft	Метод	Возвращает копию строки, в которую добавлено указанное количество символов слева.
PadRight	Метод	Возвращает копию строки, в которую добавлено указанное количество символов справа.

МЕТОД SPLIT() – ПРИМЕР 1

```
string example = "5 great men attacks of 1997 happened over 20 years ago.";
Console.WriteLine(CountIntsInSplitString(example, " "));

static int CountIntsInSplitString(string s, params string[] delimiters) {
    int count = 0;
    if (delimiters.Length == 0) {
        delimiters = new[] { " " };
    }
    string[] splitString = s.Split(delimiters, StringSplitOptions.RemoveEmptyEntries);
    foreach (string word in splitString) {
        if (int.TryParse(word, out _)) {
            ++count;
        }
    }
    return count;
}
```

Пустые вхождения строки будут удаляться в результате разделения.

Если не нужно сохранять результат в out-переменную, то можно проигнорировать его

ПРИМЕР МЕТОД SPLIT()

```
using System;

string line = "hi there! this, is: a string.";
char[] delimiters = { ' ', '!', ',', ':', '.' };
string[] words = line.Split(delimiters, StringSplitOptions.RemoveEmptyEntries);
Console.WriteLine($"Word Count: {words.Length}\r\nThe Words...");
foreach (string word in words)
{
    Console.WriteLine($"{word}");
}
```

Вывод:
Word Count: 6
The Words...
hi
there
this
is
a
string

ПРИМЕР МЕТОД JOIN()

```
using System;

string sent = "De gustibus non est disputandum"; // О вкусах не спорят.
string[] words = sent.Split(' ');
Console.WriteLine("word.Length = " + words.Length);
foreach (string word in words)
{
    Console.Write("{0}\t", word);
}
sent = string.Join("-:-", words);
Console.WriteLine("\n" + sent);
```

Вывод:

word.Length = 5
De gustibus non est disputandum
De-:-gustibus-:-non-:-est-:-disputandum

ПРЕОБРАЗОВАНИЯ С УЧАСТИЕМ СТРОКОВОГО ТИПА

Для формирования строки не пригодна операция приведения типов:

(тип) первичное_выражение

```
int n = 8, m = 3;
Console.WriteLine(m.ToString() + n.ToString() + " попугаев");
```

Результат вывода на консоль:
38 попугаев

Средства получения значений из строк:

- Методы класса System.Convert,

```
public static int ToInt32(string value) {
  if (value == null)
    return 0;
  return int.Parse(value, CultureInfo.CurrentCulture);
}
```

- Метод static Parse()
- Метод static TryParse()

ПОЛУЧЕНИЕ ЗНАЧЕНИЙ ИЗ СТРОК

```
int res = int.Parse(Console.ReadLine());
```

Вводимая строка:
" - 240 "
Значением переменной res будет -240

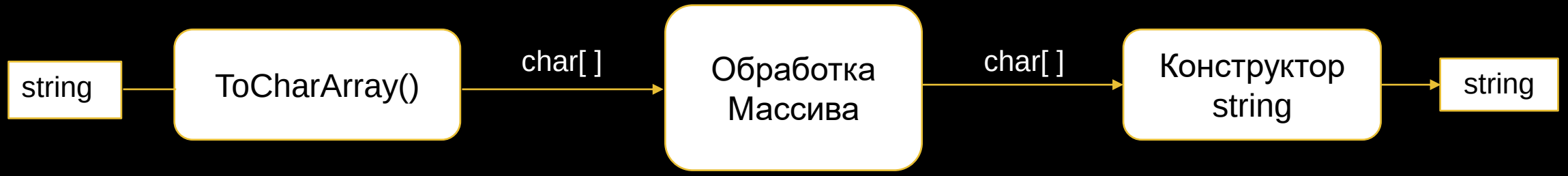
```
int res;
do Console.Write("Введите целое число: ");
while(int.TryParse(Console.ReadLine(),out res)==false);
```

Convert.ToInt32(строка);

```
string sPi = "3,14159", radius = "10,0";
double circle = 2 * Convert.ToDouble(sPi) * Convert.ToDouble(radius);
Console.WriteLine("Длина окружности="+circle.ToString());
```

Будет выведено:
Длина окружности=62,8318

РАБОТА С СИМВОЛАМИ СТРОКИ КАК С МАССИВОМ CHAR




```
string row = "0123456789";
char[] reversed;
reversed = row.ToCharArray();

Array.Reverse(reversed);

row = new string(reversed);
Console.WriteLine(row);
```

Вывод:
9876543210

ИЗМЕНЕНИЕ СОДЕРЖИМОГО СТРОК

- Изменение содержимого строки в C # [https://learn.microsoft.com/ru-ru/dotnet/csharp/how-to/modify-string-contents]

СРАВНЕНИЕ СТРОК

- Сравнение строк в C# [https://learn.microsoft.com/ru-ru/dotnet/csharp/how-to/compare-strings]

ПРИМЕР. СРАВНЕНИЕ СТРОК

```
public static int Compare (string strA, string strB)
```

```
string[] eng = { "one", "two", "three", "four" };
string res = eng[0];
foreach (string num in eng)
{
    if (string.Compare(res, num) < 0)
    {
        res = num;
    }
}
Console.WriteLine(res);
```

Вывод:
two

ПРИМЕР. СРАВНЕНИЕ СТРОК С УЧЕТОМ РЕГИОНАЛЬНЫХ СТАНДАРТОВ

```
public static int Compare(string sA, string sB, bool ignoreCase, CultureInfo);
```

```
string[] hens = {"Куропатка белая", "Куропатка тундровая",
                 "Тетерев", "Глухарь", "Рябчик" };
string res = hens[0];
foreach (string hen in hens)
{
    if (string.Compare(res, hen, true, new CultureInfo("ru")) > 0)
    {
        res = hen;
    }
}
Console.WriteLine(res);
```

Вывод:
Глухарь

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

- Raw String Literals In C# 11 (c-sharpcorner.com)
- The Complete Guide to C# 11 Raw String Literals | Guilherme Ferreira - Minimalist developer (gsferreira.com)
- Raw string literals in C# | Steve Fenton
- Строки и строковые литералы (https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/strings/)
- https://docs.microsoft.com/ru-ru/dotnet/api/system.string?view=net-5.0
- https://docs.microsoft.com/ru-ru/dotnet/api/system.string.-ctor?view=net-5.0
- https://docs.microsoft.com/en-us/dotnet/api/system.string.chars?view=net-5.0
- https://docs.microsoft.com/ru-ru/dotnet/api/system.string.compare?view=net-5.0
- https://docs.microsoft.com/ru-ru/dotnet/api/system.string.split?view=net-5.0
- https://docs.microsoft.com/ru-ru/dotnet/api/system.string.join?view=net-5.0