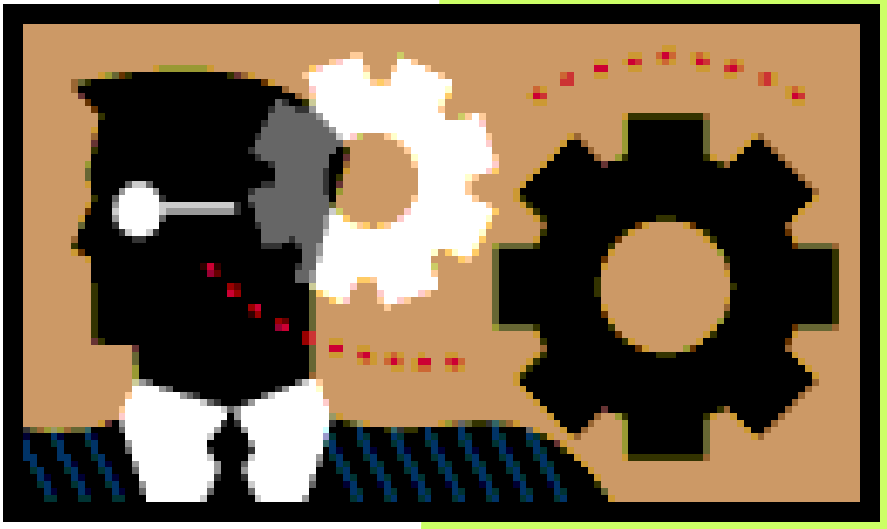

# آموزش سی شارپ
# C#.NET

## کار با رشته ها

## طریقه ایجاد کردن رشته ها:

راه معمول تعریف رشته در سی شارپ استفاده از دو علامت نقل قول است.

("abcdef") که رشته مورد نظر ما در بین این دو علامت تایپ می شود.

```
string newString = "This is a string literal";
```

رشته هایی که در بین علامت نقل قول احاطه شده اند می توانند شامل escape characters باشند از قبیل "\n"، "\t" و غیره.

"\n" برای رفتن به خط بعدی در یک رشته مثلا اگر رشته را به صورت زیر تعریف کنیم:

```
string newString = "mohammad\nbashiry";
```

هنگام چاپ این رشته و نمایش در خروجی به صورت زیر چاپ خواهد شد:

```
Mohammad
Bashiry
```

و همچنین "\t" برای رفتن به ۸ کاراکتر بعد در رشته و..

برای آشنایی بیشتر با این کاراکترهای کنترلی می توانید کتاب آموزش C یا C++ را مطالعه کنید و یا به Help خود سی شارپ مراجعه کنید..

ممکن است این سوال پیش بیاید که اگر ما بخواهیم خود کاراکتر بک‌اسلش "\" را چاپ کنیم باید چکار کرد؟

برای پاسخ به این سوال : باید از دو کاراکتر بک‌اسلش پشت سر هم استفاده نمایید. \\

```
string newString = "mohammad\\bashiry";
```

که خروجی این کد به صورت زیر خواهد شد:

```
Mohammad\bashiry
```

در صورتیکه قبل از علامت نقل قول از @ استفاده کنید دیگر \ را به عنوان کاراکتر کنترلی در نظر نمی‌گیرد و می توانید رشته ها به همان صورت که می خواهید وارد کنید.

```
string literalOne =
"\\\\MySystem\\MyDirectory\\ProgrammingC#.cs";
string verbatimLiteralOne =
@"\\MySystem\MyDirectory\ProgrammingC#.cs;"
```

در صورتیکه بخواهید در این نوع رشته( یا Verbatim string literals) به خط بعد بروید دیگر نمی‌توانید از کاراکتر کنترلی استفاده کنید بلکه مانند زیر باید تعریف کنید:

```
string verbatimLiteralTwo = @"Line One
Line Two";
```

در صورتیکه در حالت اول یعنی وقتی که از علامت @ استفاده نکردیم مانند زیر تعریف خواهیم کرد:

```
string literalTwo = "Line One\nLine Two";
```

**متد ToString:**

یکی از راه های رایج برای ایجاد یک رشته استفاده از متد ToString () در سی شارپ است. که مقدار بازگشتی یک رشته است. مثلا فرض کنید بخواهید عدد 5 را به رشته تبدیل کنید تا بتوانید با آن به صورت رشته برخورد کنید . برای اینکار کافی است از این متد استفاده کنید. (به همین راحتی)

```
int myInteger = 5;
```

```
string integerString = myInteger.ToString( );
```

که مقدار بدست آمده در رشته با نام integerString ریخته خواهد شد.

## دستکاری رشته ها :

کلاس string در همین خصوص برای دستکاری، مقایسه، جستجوی و ... رشته‌ها وجـود دارد. در جدول زیر لیست تمام اعضای این کلاس را مشاهده می‌کنید:

| فیلدها یا متدها | توضیحات |
|---|---|
| Empty | فیلد عمومی ثابت برای نمایش رشته های تهی |
| Compare( ) | Overloaded public static method that compares two strings.<br>(مقایسه دو رشته – حساس به حروف کوچک و بزرگ) |
| CompareOrdinal() | Overloaded public static method that compares two strings without regard to local or culture.<br>مقایسه دو رشته بدون ملاحظه محلی و یا Culture بودن |
| Concat( ) | Overloaded public static method that creates a new string from one or more strings.<br>برای اتصال دو رشته با هم از این متد استفاده می‌شود |

| | |
|---|---|
| Copy( ) | Overloaded public static method that creates a new string by copying another.<br>ایجاد رشته جدید با کپی در رشته دیگر |
| Equals( ) | Overloaded public static method that determines if two strings have the same value.<br>اگر دو رشته مقدار یکسانی دارند. |
| Format( ) | Overloaded public static method that formats a string using a format specification.<br>استفاده از یک فرمت خاص |
| Intern( ) | Overloaded public static method that returns a reference to the specified instance of a string. |
| IsInterned( ) | Overloaded public static method that returns a reference for the string. |
| Join( ) | Overloaded public static method that concatenates a specified string between each element of a string array. |
| Chars( ) | The string indexer. |

| | |
|---|---|
| | شمارنده رشته (موقعیت کاراکترها) |
| Length( ) | The number of characters in the instance.<br>گرفتن طول یک رشته |
| Clone( ) | Returns the string. |
| Compareto( ) | Compares this string with another.<br>مقایسه یک رشته با رشتهٔ دیگر |
| CopyTo( ) | Copies the specified number of characters to an array of Unicode characters. |
| EndsWith( ) | Indicates whether the specified string matches the end of this string.<br>جستجوی یک رشته در انتهای رشته ای دیگر |
| Equals( ) | Determines if two strings have the same value.<br>مقایسه دو رشته ( |
| Insert( ) | Returns a new string with the specified string inserted.<br>درج در رشته ای دیگر |
| LastIndexOf() | Reports the index of the last occurrence of a specified character or string within the |

| | |
|---|---|
| | string. |
| PadLeft( ) | Right-aligns the characters in the string, padding to the left with spaces or a specified character. |
| PadRight( ) | Left-aligns the characters in the string, padding to the right with spaces or a specified character. |
| Remove( ) | Deletes the specified number of characters.<br>حذف |
| Split( ) | Returns the substrings delimited by the specified characters in a string array.<br>قسمت قسمت کردن یک رشته بر اساس یک کاراکتر<br>خاص و ریختن بخش های تقسیم شده در یک آرایه |
| StartsWith() | Indicates if the string starts with the specified characters.<br>نشان دادن یک رشته اگر رشته با کاراکترهای خاص شروع<br>می‌شود |
| SubString( ) | Retrieves a substring.<br>دریافت یک زیر رشته |
| ToCharArray() | Copies the characters from the |

| | |
|---|---|
| | string to a character array.<br>کپی کاراکترهای یک رشته به یک آرایهٔ رشته‌ای |
| ToLower( ) | Returns a copy of the string in lowercase.<br>تبدیل کل رشته به حروف کوچک |
| ToUpper( ) | Returns a copy of the string in uppercase.<br>تبدیل کل رشته به حروف بزرگ |
| Trim( ) | Removes all occurrences of a set of specified characters from beginning and end of the string.<br>حذف کاراکترهای خاص از یک رشته (مثلا حذف تمام فاصله های خالی) |
| TrimEnd( ) | Behaves like Trim, but only at the end.<br>حذف از انتها |
| TrimStart( ) | Behaves like Trim, but only at the start.<br>حذف از ابتدا |

در مثال زیر از مهمترین متدهای جدول بالا استفاده شده است از قبیل:

Compare()، Concat()، Copy()، Insert()، EndsWith() و IndexOf

مثال کار با رشته ها: **(کد شماره ۱)**

```
namespace Programming_CSharp
{
    using System;
    public class StringTester
    {
        static void Main()
        {
            // create some strings to work with
            string s1 = "abcd";
            string s2 = "ABCD";
            string s3 = @"Liberty Associates, Inc.
provides custom .NET development,
on-site Training and Consulting";
            int result; // hold the results of comparisons
            // compare two strings, case sensitive
            result = string.Compare(s1, s2);
            Console.WriteLine(
            "comprecompare s1: {0}, s2: {1}, result: {2}\n",
            s1, s2, result);
            // overloaded compare, takes boolean "ignore case"
            //(true = ignore case)
            result = string.Compare(s1, s2, true);
            Console.WriteLine("compare insensitive\n");
            Console.WriteLine("s4: {0}, s2: {1}, result: {2}\n",
            s1, s2, result);
            // concatenation method
            string s6 = string.Concat(s1, s2);
            Console.WriteLine(
            "s6 concatenated from s1 and s2: {0}", s6);
```

```
                // use the overloaded operator
                string s7 = s1 + s2;
                Console.WriteLine(
                "s7 concatenated from s1 + s2: {0}", s7);
                // the string copy method
                string s8 = string.Copy(s7);
                Console.WriteLine(
                "s8 copied from s7: {0}", s8);
                // use the overloaded operator
                string s9 = s8;
                Console.WriteLine("s9 = s8: {0}", s9);
                // three ways to compare.
                Console.WriteLine(
                "\nDoes s9.Equals(s8)?: {0}",
                s9.Equals(s8));
                Console.WriteLine(
                "Does Equals(s9,s8)?: {0}",
                string.Equals(s9, s8));
                Console.WriteLine(
                "Does s9==s8?: {0}", s9 == s8);
                // Two useful properties: the index and the length
                Console.WriteLine(
                "\nString s9 is {0} characters long. ",
                s9.Length);
                Console.WriteLine(
                "The 5th character is {1}\n",
                s9.Length, s9[4]);
                //  test  whether  a  string  ends  with  a  set  of
characters
              Console.WriteLine("s3:{0}\nEnds with Training?: {1}\n",
```

```
                s3,
                s3.EndsWith("Training"));
                Console.WriteLine(
                "Ends with Consulting?: {0}",
                s3.EndsWith("Consulting"));
                // return the index of the substring
                Console.WriteLine(
                "\nThe first occurrence of Training ");
                Console.WriteLine("in s3 is {0}\n",
                s3.IndexOf("Training"));
                // insert the word excellent before "training"
                string s10 = s3.Insert(103, "excellent ");
                Console.WriteLine("s10: {0}\n", s10);
                // you can combine the two as follows:
                string s11 = s3.Insert(s3.IndexOf("Training"),
                "excellent ");
                Console.WriteLine("s11: {0}\n", s11);
            }
        }
    }
```

خروجی

```
compre s1: abcd, s2: ABCD, result: -1
compare insensitive
s4: abcd, s2: ABCD, result: 0
s6 concatenated from s1 and s2: abcdABCD
s7 concatenated from s1 + s2: abcdABCD
s8 copied from s7: abcdABCD
```

```
s9 = s8: abcdABCD
Does s9.Equals(s8)?: True
Does Equals(s9,s8)?: True
Does s9==s8?: True
String s9 is 8 characters long.
The 5th character is A
s3:Liberty Associates, Inc.
provides custom .NET development,
on-site Training and Consulting
Ends with Training?: False
Ends with Consulting?: True
The first occurrence of Training
in s3 is 103
s10: Liberty Associates, Inc.
provides custom .NET development,
on-site excellent Training and Consulting
s11: Liberty Associates, Inc.
provides custom .NET development,
on-site excellent Training and Consulting
```

**توضیحات کامل کد بالا :**

در شروع کد سه رشته را به ترتیب زیر تعریف کردیم:

```
string s1 = "abcd";
string s2 = "ABCD";
string s3 = @"Liberty Associates, Inc.
```

```
provides custom .NET development,
on-site Training and Consulting";
```

دو رشته اول ← String Literals

رشته سوم ← Verbatim String literals

(در مورد این دو نوع تعریف رشته قبلا توضیحات داده شده است)

در ادامه کار با استفاده از متد Compare() دو رشته s1 و s2 با هم مقایسه شدند:

```
int result; // hold the results of comparisons
// compare two strings, case sensitive
result = string.Compare(s1, s2);
Console.WriteLine(
"comprecompare s1: {0}, s2: {1}, result: {2}\n", s1,
s2, result);
```

**توضیح در مورد دستور Console.WriteLine**

همانطور که در کد بالا دقت می کنید این دستور برای فرستادن نتیجه به خروجی است. به این صورت کاربری که با برنامه کار می کند می تواند نتایج را در خروجی که همان مانیتور است ببیند.

{0} یعنی پارامتر اول ← همان s1

{1} یعنی پارامتر دوم ← همان s2

{2} یعنی پارامتر سوم ← یعنی result

همانطور که دیدید خروجی این دستور به صورت بدست آمد:

**comprecompare s1: abcd, s2: ABCD, result: -1**

### توضیح در مورد خروجی دستور:

این مقایسه یک مقایسه حساس به متن است یعنی در بین حروف کوچک و بزرگ تفاوت قائل می‌شود. بر اساس نتیجه مقایسه سه حالت خروجی دارد

**مقادیر بازگشتی:**

0>: رشته اول از رشته دوم کمتر است (خروجی منفی)

در این مثال: abcd<ABCD و خروجی ۱- (بر اساس کاراکتر اسکی)

0 : اگر دو رشته با هم یکسان باشند

0<: رشته اول بزرگتر از رشته دوم است (خروجی مثبت)

**به بررسی ادامه کد می پردازیم:**

```
result = string.Compare(s1,s2, true);
Console.WriteLine("compare insensitive\n");
Console.WriteLine("s4: {0}, s2: {1}, result: {2}\n",
s1, s2, result);
```

اگر بیشتر به کد بالا دقت کنید می توانید بفهمید که با مورد قبلی فقط در کلمه true تفاوت دارد. در صورتی که از کلمه true به شکل بالا استفاده کنیم حساسیت به حروف کوچک و بزرگ نخواهیم داشت. پس خروجی کد فوق به صورت زیر خواهد بود.

```
compare insensitive
s4: abcd, s2: ABCD, result: 0
```

در ادامه کد دستور زیر را خواهیم داشت:

```
string s6 = string.Concat(s1,s2);
```

همانگونه که قبلا اشاره شد این دستور دو رشته s1 و s2 را به هم می‌چسباند. همچنین به جای دستور بالا می‌توان از دستور زیر استفاده کرد.

```
string s7 = s1 + s2;
```

در این دستور دو رشته با استفاده از عملگر + به هم متصل شده‌اند.

در زیر خروجی هر دو دستور را می بینید:

```
s6 concatenated from s1 and s2: abcdABCD
s7 concatenated from s1 + s2: abcdABCD
```

فکر کنم این دستور هم متوجه شده باشید.

- پس به بررسی ادامه کد خواهیم پرداخت:

```
string s8 = string.Copy(s7);
```

برای ایجاد یک کپی از رشته دو راه در پیش داریم:

**یکی** با استفاده از ثابت copy است. همانند دستور بالا که عملکردش نگهداری یک کپی از رشته s7 در رشته s8 می‌باشد.

و **راه دوم** به صورت زیر میتوان یک کپی از رشته را در رشته دیگر نگهداری کرد:

```
string s9 = s8;
```

دوباره خروجی این دو دستور مثل هم است:

```
s8 copied from s7: abcdABCD
s9 = s8: abcdABCD
```

- در کلاس string در .NET سه راه برای تست تساوی دو رشته در پیش و روی ما گذاشته است.

  - **راه اول:** استفاده از متد Equals() (بهترین روش)

```
Console.WriteLine("\nDoes s9.Equals(s8)?: {0}",
```

```
s9.Equals(s8));
```

در اینجا s9 با s8 مقایسه می‌شوند

- **راه دوم:** فرستادن هر دو رشته به عنوان پارامتر به متد Equals()

```
Console.WriteLine("Does Equals(s9,s8)?: {0}",
string.Equals(s9,s8));
```

- **راه سوم:** مقایسه از طریق عملگر "=="

```
Console.WriteLine("Does s9==s8?: {0}", s9 == s8);
```

در هر یک از این حالات نتیجه مقداری است درست یا غلط (Boolean)

در زیر خروجی‌ها را به ترتیب ملاحظه می‌کنید:

```
Does s9.Equals(s8)?: True
Does Equals(s9,s8)?: True
Does s9==s8?: True
```

این هم از این دستور

در **خط بعدی کد** اصلی به علامت عملگر اندیس یا [] بر می‌خوریم:

```
Console.WriteLine("\nString s9 is {0} characters
long.,s9.Length);
Console.WriteLine("The 5th character is {1}\n",
s9.Length, s9[4]);
```

عملگر اندیس کاراکتری را که در محل ذکر شده وارد شده است را بر می‌گرداند مثلا برای مثال بالا موقعیت **پنجم** (اندیسها از صفر شروع می‌شوند) از رشته s9 کاراکتر A است.

Length هم طول رشته را به ما می‌دهد.

طبق نکات گفته شده خروجیها چیزی شبیه زیر می‌باشد:

```
String s9 is {8} characters long.
The 5th character is A
```

متد EndsWith() : یافتن یک زیر رشته در انتهای رشته دیگر

منظور اینست که به سی شارپ می‌گوئیم اگر در آخر فلان رشته فلان کلمه بود True برگرداند.

حتما به خاطر دارید که در کد اصلی یا همان کد شماره ۱ مقدار رشته s3 را به صورت زیـر تعریـف کردیم:

```
string s3 = @"Liberty Associates, Inc.
provides custom .NET development,
on-site Training and Consulting";
```

این رشته با کلمه Consulting خاتمه می‌یابد نه با کلمه Training

```
// test whether a string ends with a set of
characters
Console.WriteLine("s3:{0}\nEnds with Training?:
{1}\n",s3,s3.EndsWith("Training") );
Console.WriteLine("Ends with Consulting?: {0}",
s3.EndsWith("Consulting"));
```

در نتیجه طبق مطالب گفته شده در بالا خروجی به صورت زیر می‌باشد:

```
s3:Liberty Associates, Inc.
provides custom .NET development,
on-site Training and Consulting
Ends with Training?: False
Ends with Consulting?: True
```

متد IndexOf() : این متد موقعیت یک زیر رشته را در رشته ما پیدا می‌کند.

متد Insert() : درج یک زیر رشته جدید در یک کپی از رشته اصلی

در زیر اولین رخداد کلمه Training در رشته S1 مد نظر ما می‌باشد.

```
Console.WriteLine("\nThe first occurrence of Training
");
Console.WriteLine ("in s3 is {0}\n",
s3.IndexOf("Training"));
```

این هم از خروجی این:

```
The first occurrence of Training
in s3 is 103
```

- متد Insert (): با استفاده از این متد میتوانیم در موقعیت دلخواه کلمه مورد نظرمان را درج کنیم. برای این مثال میخواهیم قبل از کلمه Training کلمه excellent را درج نماییم به طوری که بعد از آن یک فاصله قرار گیرد و بعد از فاصله کلمه Training ظاهر شود. بهترین راه استفاده از همین متد یعنی Insert () است.

برای استفاده از این متد به دو پارامتر نیازمندیم. پارامتر اول موقعیتی است که میخواهیم رشته جدید را در آنجا درج کنیم و پارامتر دوم همان رشته مورد نظر برای درج در رشتهٔ دیگر است.

```
string s10 = s3.Insert(103,"excellent ");
Console.WriteLine("s10: {0}\n",s10);
```

خروجی هم به شکل زیر است:

```
s10: Liberty Associates, Inc.
provides custom .NET development,
on-site excellent Training and Consulting
```

نهایتا به صورت ترکیبی زیر هم می‌توانید استفاده نمایید:

```
string s11 =
s3.Insert(s3.IndexOf("Training"),"excellent ");
```

```
Console.WriteLine("s11: {0}\n",s11);
```

**«« اتمام بررسی کد شماره ۱ »»**

## یافتن زیر رشته‌ها:

مثال -کد شماره ۲

```
namespace Programming_CSharp
{
    using System;
    using System.Text;
    public class StringTester
    {
        static void Main()
        {
            // create some strings to work with
            string s1 = "One Two Three Four";
            int ix;
            // get the index of the last space
            ix = s1.LastIndexOf(" ");
            // get the last word.
            string s2 = s1.Substring(ix + 1);
            // set s1 to the substring starting at 0
            // and ending at ix (the start of the last word
            // thus s1 has one two three
            s1 = s1.Substring(0, ix);
            // find the last space in s1 (after two)
            ix = s1.LastIndexOf(" ");
            // set s3 to the substring starting at
            // ix, the space after "two" plus one more
            // thus s3 = "three"
```

```
            string s3 = s1.Substring(ix + 1);
            // reset s1 to the substring starting at 0
            // and ending at ix, thus the string "one two"
            s1 = s1.Substring(0, ix);
            // reset ix to the space between
            // "one" and "two"
            ix = s1.LastIndexOf(" ");
            // set s4 to the substring starting one
            // space after ix, thus the substring "two"
            string s4 = s1.Substring(ix + 1);
            // reset s1 to the substring starting at 0
            // and ending at ix, thus "one"
            s1 = s1.Substring(0, ix);
            // set ix to the last space, but there is
            // none so ix now = -1
            ix = s1.LastIndexOf(" ");
            // set s5 to the substring at one past
            // the last space. there was no last space
            // so this sets s5 to the substring starting
            // at zero
            string s5 = s1.Substring(ix + 1);
            Console.WriteLine("s2: {0}\ns3: {1}", s2, s3);
            Console.WriteLine("s4: {0}\ns5: {1}\n", s4, s5);
            Console.WriteLine("s1: {0}\n", s1);
        }
    }
}
```

خروجی کد شماره ۲

```
s2: Four
s3: Three
s4: Two
s5: One
s1: One
```

حال به توضیحات کد شماره ۲ خواهیم پرداخت.

برای شروع کار یک رشته به صورت زیر خواهیم ساخت:

```
string s1 = "One Two Three Four";
```

بعد از این کار ix را به موقعیت آخرین فاصله خالی در رشته تنظیم می‌کنیم:

```
ix=s1.LastIndexOf(" ");
```

سپس زیر رشته را به یک فاصله بعد از مورد تنظیم شده set میکنیم: (در اصل کلمه Four)

```
string s2 = s1.Substring(ix+1);
```

در مرحله بعد لغت Four را از رشته S1 خارج می‌کنیم. برا این کار می‌توانید به صورت زیر عمل کنید:

```
s1 = s1.Substring(0,ix);
```

ix به آخرین فاصله باقیمانده می‌رسد. در این نقطه از لغت Three شروع میکنیم. و همینطور ادامه می‌دهیم تا s4 و s5 تولید شوند. سپس نتیجه را چاپ می‌کنیم:

```
s2: Four
s3: Three
s4: Two
s5: One
s1: One
```

## تکه تکه کردن یک رشته:

- Split () با استفاده از این متد می‌توان یک رشته را از یک کاراکتر خاص در طول رشته تکه تکه نمود و هر بخش را جداگانه در یک خانه از آرایه قرار داد و بعدا از آن استفاده کرد.

کد شماره ۳ -

```
namespace Programming_CSharp
{
    using System;
    using System.Text;
    public class StringTester
    {
        static void Main()
        {
            // create some strings to work with
            string s1 = "One,Two,Three Liberty Associates,
Inc.";
            // constants for the space and comma characters
            const char Space = ' ';
            const char Comma = ',';
            // array of delimiters to split the sentence with
            char[] delimiters = new char[]
                {
                Space,
                Comma
                };
```