

String methods

Length	Returns the number of characters in a string.
IndexOf	Finds the index of the first occurrence of a specified character or substring.
Substring	Retrieves a portion of the string based on a specified index or length.
ToUpper and ToLower	Converts a string to uppercase or lowercase, respectively
Trim	Removes leading and trailing whitespace characters from a string.
Replace	Replaces all occurrences of a specified character or substring with another.
Split	Splits a string into an array of substrings based on a specified delimiter.
Concat	Concatenates multiple strings into a single string.
Format	Formats a composite string by replacing placeholders with corresponding values.
Compare	Compares two strings and returns an integer indicating their relative order.
Contains	Determines whether a specified character or substring exists within a string.
StartsWith and EndsWith	Checks if a string starts or ends with a specified character or substring.
PadLeft and PadRight	Adds padding characters to the left or right of a string to achieve a specified length.
IsNullOrEmpty and IsNullOrWhiteSpace	Checks if a string is null, empty, or consists only of whitespace characters.
Join	Concatenates an array of strings into a single string, using a specified delimiter.
ToCharArray	Converts a string to an array of characters.
CompareTo	Compares two strings and returns an integer indicating their relative order.
Substring (overload with start index and length)	Retrieves a substring from a string, starting at a specified index and with a specified length.
Remove	Deletes a specified number of characters from a string, starting at a specified position.



By Ziggiboy
cheatography.com/ziggiboy/

Published 15th May, 2023.
Last updated 16th May, 2023.
Page 1 of 4.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

String methods (cont)

Equals Compares two strings for equality, taking into account culture-specific or case-sensitive comparisons.

Logical And Arithmetic Operators

&&	AND
	OR
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo
++	Increase 1
--	Decrease 1

Arrays

```
string[] planets =
{
    "Mer cur y", "Ven us", "Mar s",
    "Ear th", "Jup ite r", "Sat urn ",
    "Ura nus ", "Nep tun e", "Plu to"
};

for (int i = 0; i < planet s.L e -
ngth; i++)
{
    Consol e.W rit eLi ne( pla net s[i]);
}

foreach (string planet in planets)
{
    Consol e.W rit eLi ne( pla net);
}
```

array sort & reverse:

```
string[] names = { " Jan e", " Fra nk", " Ali ce",
" Tom " };

Array.S o r t (na mes);
foreach (string name in names)
{
    Consol e.W r ite (name + " ");
}

Array.R e v e r s e(n ames);
foreach (string name in names)
{
    Consol e.W r ite (name + " ");
}
```

Arrays (cont)

```
> }
```

Switch case

```
switch(expression) {
    case consta nt- exp res sion1 :
        statem ent(s);
        break;
    case consta nt- exp res sion2 :
    case consta nt- exp res sion3 :
        statem ent(s);
        break;

    / you can have any number of case statements /
    default : / Optional /
        statem ent(s);
}
```

While loop

```
while(condition)
```

```
{
    statement(s);
}
```

Here, statement(s) may be a single statement or a block of statem-ents. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

do while loop

```
do
{
    statement(s);
} while( condition );
```

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop execute once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop execute again. This process repeats until the given condition becomes false.

Classes and objects

```
--Classes--
class Car
{
    string color = " red ";
}

--Object--
class Car
{
    string color = " red ";
    static void Main(s tring[] args)
    {
        Car myObj = new Car();
        Consol e.W rit eLi ne( myO bj.c olor);
    }
}

--Multiple Objects--
class Car
{
    string color = " red ";
    static void Main(s tring[] args)
    {
        Car myObj1 = new Car();
        Car myObj2 = new Car();
        Consol e.W rit eLi ne( myO bj1.co lor);
        Consol e.W rit eLi ne( myO bj2.co lor);
    }
}
```

Comparison Operators

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Comments

```
// Single line
/* Multiple
lines */
/// XML comments on single line
```

Assignment

=	Assignment
+=	Add
-=	Subtract

Lists

```
List<int> number = new()
{
    4, 6, 8, 1, 10, 2, 7, 3, 9, 5
};
number.Ad d(12);
number.Ad d(11);
number.Re mov e(1);
number.So rt();
foreach (int num in number)
{
    Consol e.W rit eLi ne( num);
}

-----

List<s tri ng> word = new()
{
    " Del ta", " Cha rli e", " Bra vo",
    " Fox tro t",
    " Ech o", " Alp ha", " Gol f"
};
word.A dd( " Ind ia");
word.A dd( " Hot el");
word.R emo ve( " Alp ha");
word.S ort();
foreach (string phonetic in word)
{
    Consol e.W rit eLi ne( pho netic);
}
```

Casting

Implicit Casting (automatically) converting a smaller type to a larger type size
char -> int -> long -> float -> double.

Explicit Casting (manually) converting a larger type to a smaller size type
double -> float -> long -> int -> char.

Implicit Casting:

```
int myInt = 9;
double myDouble = myInt; // Automatic casting: int to double
```

```
Console.WriteLine(myInt); // Outputs 9
Console.WriteLine(myDouble); // Outputs 9
```

Explicit Casting:

```
double myDouble = 9.78;
int myInt = (int) myDouble; // Manual casting: double to int
```

```
Console.WriteLine(myDouble); // Outputs 9.78
Console.WriteLine(myInt); // Outputs 9
```

for loop

```
for ( init; condition; increment )
{
    statement(s);
}
```

The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.

Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the for loop.

After the body of the for loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.

The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again testing for a condition). After the condition becomes false, the for loop terminates.

Data types

bool Represents a Boolean value, either true or false.

char Represents a single Unicode character.

string Represents a sequence of characters.

Data types (cont)

int Represents signed integers (whole numbers) within the range -2,147,483,648 to 2,147,483,647.

double Represents double-precision floating-point numbers with decimal values. Sufficient for storing 15 decimal digits

decimal Represents decimal numbers with high precision and a larger range.

float Represents single-precision floating-point numbers with decimal values. Sufficient for storing 6 to 7 decimal digits

long Represents signed integers with a larger range than int, from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

DateTime Represents dates and times.

Example method

```
namespace MyApplication
{
    class Program
    {
        static void Main(s tring[] args)
        {
            MyMeth od();
        }
        static void MyMethod()
        {
            Consol e.W rit eLi ne( "I have a red banana ");
        }
    }
}
```

Ternary operator

condition ? consequent : alternative

```
string GetWeatherDisplay(double tempInCelsius) => tempInCelsius < 20.0 ? "Cold." : "Perfect!";
```

```
Console.WriteLine(GetWeatherDisplay(15)); // output: Cold.
```

```
Console.WriteLine(GetWeatherDisplay(27)); // output: Perfect!
```

