C# - CONSTANTS AND LITERALS

The constants refer to fixed values that the program may not alter during its execution. These fixed values are also called literals. Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal. There are also enumeration constants as well.

The constants are treated just like regular variables except that their values cannot be modified after their definition.

Integer Literals

An integer literal can be a decimal, or hexadecimal constant. A prefix specifies the base or radix: 0x or 0X for hexadecimal, and there is no prefix id for decimal.

An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.

Here are some examples of integer literals –

```
212         /* Legal */
215u        /* Legal */
0xFeeL      /* Legal */
```

Following are other examples of various types of Integer literals –

```
85         /* decimal */
0x4b       /* hexadecimal */
30         /* int */
30u        /* unsigned int */
30l        /* long */
30ul       /* unsigned long */
```

Floating-point Literals

A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part. You can represent floating point literals either in decimal form or exponential form.

Here are some examples of floating-point literals –

```
3.14159       /* Legal */
314159E-5F    /* Legal */
510E          /* Illegal: incomplete exponent */
210f          /* Illegal: no decimal or exponent */
.e55          /* Illegal: missing integer or fraction */
```

While representing in decimal form, you must include the decimal point, the exponent, or both; and while representing using exponential form you must include the integer part, the fractional part, or both. The signed exponent is introduced by e or E.

Character Constants

Character literals are enclosed in single quotes. For example, 'x' and can be stored in a simple variable of char type. A character literal can be a plain character $such as 'x'$, an escape sequence $such as '\t'$, or a universal character $such as '\u02C0'$.

There are certain characters in C# when they are preceded by a backslash. They have special meaning and they are used to represent like newline \n or tab \t. Here, is a list of some of such escape sequence codes –

Escape sequence	Meaning
\\	\ character
\'	' character
\"	" character
\?	? character
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\xhh . . .	Hexadecimal number of one or more digits

Following is the example to show few escape sequence characters –

Live Demo

```
using System;

namespace EscapeChar {
   class Program {
      static void Main(string[] args) {
         Console.WriteLine("Hello\tWorld\n\n");
         Console.ReadLine();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result –

```
Hello   World
```

String Literals

String literals or constants are enclosed in double quotes "" or with @"". A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.

You can break a long line into multiple lines using string literals and separating the parts using whitespaces.

Here are some examples of string literals. All the three forms are identical strings.

```
"hello, dear"
"hello, \
dear"
"hello, " "d" "ear"
@"hello dear"
```

Defining Constants

Constants are defined using the **const** keyword. Syntax for defining a constant is –

```
const <data_type> <constant_name> = value;
```

The following program demonstrates defining and using a constant in your program –

Live Demo

```
using System;

namespace DeclaringConstants {
   class Program {
      static void Main(string[] args) {
         const double pi = 3.14159;

         // constant declaration
         double r;
         Console.WriteLine("Enter Radius: ");
         r = Convert.ToDouble(Console.ReadLine());

         double areaCircle = pi * r * r;
         Console.WriteLine("Radius: {0}, Area: {1}", r, areaCircle);
         Console.ReadLine();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result –

```
Enter Radius:
3
Radius: 3, Area: 28.27431
```