Literals and Constants in C

In this tutorial you will learn all about literals and constants in C language. Also you will notice the narrow difference between literals and constants with the help of simple examples.

What are Literals in C ?

In C programming constants and literals mean fixed values which do not change in the course of the program. We very often confuse identifying constants and literals as there is a thin line lies between them. **Literal is nothing but a constant value itself** . For example:

```
int X = 10;
printf(" Learn C through Learn eTutorials!");
```

Here the number '10' and 'Learn C through Learn eTutorials' both are literals.

Different types of Literals in C

In c programming literals are broadly classified into four types:

1. Integer literals
2. float literals
3. character literals
4. String literals

Integer Literals

Integer literals in C are numeric values that represent integer type values only.No fractional or exponential values can be represented.The integer literal can be of three types:

Integer Literals	Example
Decimal Literals	10,3369,555u
Octal Literals	036,099,024L
HexaDecimal Literals	0xdEbL,0x5Au

When you observe the above examples you can see the below findings

- A literal with no prefix and base of 10 are termed as **decimal type integers**.
- A literal prefixed with '0' indicates that the number is an **octal type integer**. The base value of the number is 8.
- A literal prepended with '0x' denotes a **hexadecimal type number**. The base value of the number is 16.Hexadecimal number contains digits and alphabets.

Integer Literals appended with Qualifiers:

sign qualifiers(U or u) : are suffixed with literals to indicate the value as unsigned integer type.
size qualifiers L or l) : are suffixed with literals to denote the size of the integer type as long .

Another important feature is all the alphabets used in the integer literals are not case-sensitive i.e. 'x' and 'X' or 'e'and 'E' all mean the same.

Floating Point Literals

A floating-point literal means a literal which contains a fractional or exponential number. These literals are mainly of two types– real and complex. The real ones contain the following parts:

I. Integer part
II. Real part
III. Fractional part
iv. Exponential part

The floating- point literals can be represented in two forms:

Floating-point Literals Form	Example
Decimal Form	+10.50, -3.3, 6.19
Exponential Form	+1.6e28, -2.4e12, 3.4e-25

Character Literals

Character literals in C are single characters which are enclosed in single quotes.

```
char x ='a';
```

In this example a single character 'a' is stored inside the variable (or address) x, so 'a' is nothing but a character literal.

When we try to store more than one character to a character variable , the compiler will generate a warning which states multi-character character constants as shown below in the code fragment.

character literal with single character

```
#include <stdio.h&rt;
int main()
{
 char v = 'v';
 printf("%c",v);
}
```

Output:

```
v
```

character literal with multiple characters

```
#include <stdio.h&rt;
int main()
{
 char v = 'var';
 printf("%c",v);
}
```

Output:

```
r
```

C programs not only on simple characters, but also on special escape sequences. For example '\t' accounts for a tablespace or ' \n' for moving a cursor to the next line.

String Literals

As letters construct words in C language, an array of characters is called a string in C. So a string is composed of characters, escape sequences, symbols and white space.

A string literal in C is a literal which is enclosed within double quotes.For instance ,

```
x = "Learn";
y = "C programming";
```

In this example , x and y are string variables which store string literals " Learn" and " C Programming".

> Note :'C' allocates space for character literal while "C" will allocate space for string literal

What are Constants in C ?

Constants in C are variables whose value can't be changed after defining.In programming its is a good practice to use constants instead of literals.

Suppose you are writing a program for banking software where the minimum balance is 1000 Rs. You can define a constant say 'min_bal' and give it a value 1000. So later on the program, you don't have to recall the value again and again. Rather, you can just call the constant 'min_bal' whenever needed. Moreover, there is an opportunity for future alterations. Suppose the bank decides to increase the minimum balance to 2500 Rs, you don't require to change all the integer literals. What you have to do is just going to the line where the constant is declared and replacing the value '1000' with '2500'. This is why constants are always preferred by the programmers than literals.

How to define Constants in C ?

Constants can be defined by two ways:

- using #define preprocessor
- using keyword const.

Using #define preprocessor:

'#define' works like a function by declaring a constant name and assigning the value at once without using '='.

The syntax is :

```
#define constant_name constant_value
```

```
// Example
#define pi 3.14
```

The following example will illustrate the use of #define preprocessor to assign a constant value

```
#include <stdio.h>
#define a 5
#define b 6
#define jump '\n'
main()
{
	int c;
	c=a+b;
	printf("The sum is %c %d",jump,c);
	getch();
}
```

Output:

```
The sum is
11
```

Using the keyword const :

The 'const' is a simple keyword by which you can declare a constant using following syntax:

```
const  data_type  c
```

```
// Example
const float pi = 3.14;
```

So unlike preprocessor here we have to put semicolon (;) at the end as 'const' declaration is considered to be a statement, and assignment operator '=' is used to assign values the constants. The above program can be written like this:

```
#include <stdio.h&rt;
const int a = 5;
const int b = 6;
const char jump= '\n';

main()
{
	int c;
	c=a+b;
	printf("The sum is %c %d",jump,c);
	getch();
}
```