018 -
LITERALS IN C

Literals are the constant values assigned to the constant variables. We can say that the literals represent the fixed values that cannot be modified.

It also contains memory but does not have references as variables.

const int = 10;

The above statement is a constant integer expression in which 10 is an integer literal.

There are four major types of literals that exist in C programming :-

Integer literal
Float literal
Character literal
String literal

const constant_name = value;

2. The #define preprocessor is also used to define constant.

#define CONSTANT_VALUE VALUE

There are four major types of literals that exist in C programming :-

INTEGER LITERAL

It is a numeric literal that represents only integer type values. It represents the value neither in fractional nor exponential part.

It can be specified in the following three ways:

Decimal number (base 10) :- It is defined by representing the digits between 0 to 9. For example, 45, 67, etc.

Octal number (base 8) :- It is defined as a number in which 0 is followed by digits such as 0,1,2,3,4,5,6,7. For example, 012, 034, 055, etc.

Hexadecimal number (base 16) :- It is defined as a number in which 0x or 0X is followed by the hexadecimal digits (i.e., digits from 0 to 9, alphabetical characters from (a-z) or (A-Z)).

> L or l: It is a size qualifier that specifies the size of the integer type as long.
>
> U or u: It is a sign qualifier that represents the type of the integer as unsigned. An unsigned qualifier contains only positive values.

FLOAT LITERAL

It is a literal that contains only floating-point values or real numbers.

These real numbers contain the number of parts such as integer part, real part, exponential part, and fractional part.

The floating-point literal must be specified either in decimal or in exponential form.

The decimal form must contain either decimal point, exponential part, or both.

If it does not contain either of these, then the compiler will throw an error.

The decimal notation can be prefixed either by '+' or '-' symbol that specifies the positive and negative numbers.

The exponential form is useful when we want to represent the number, which is having a big magnitude.

It contains two parts, i.e., mantissa and exponent.

For example, the number is 2340000000000, and it can be expressed as 2.34e12 in an exponential form, where 2.34 is mantissa and 12 is exponent.

The systax to define a float literal in exponential form is :-

const float constant_name = [+/-] <Mantissa> <e/E> [+/-] <Exponent>
// +1e23, -9e2,

The following are the rules for creating a float literal in exponential notation:

In exponential notation, the mantissa can be specified either in decimal or fractional form.

An exponent can be written in both uppercase and lowercase, i.e., e and E.

We can use both the signs, i.e., positive and negative, before the mantissa and exponent.

Spaces are not allowed.

CHARACTER LITERAL

A character literal contains a single character enclosed within single quotes.

If multiple characters are assigned to the variable, then we need to create a character array.

If we try to store more than one character in a variable, then the warning of a multi-character character constant will be generated.

STRING LITERAL

A string literal represents multiple characters enclosed within double-quotes.

It contains an additional character, i.e., '\0' (null character), which gets automatically inserted.

This null character specifies the termination of the string.

We can use the '+' symbol to concatenate two strings.
