C Variables, Constants and Literals

Variables

In programming, a variable is a container (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name (identifier). Variable names are just the symbolic representation of a memory location. For example:

int playerScore = 95;
Here, playerScore is a variable of int type. Here, the variable is assigned an integer value 95.

The value of a variable can be changed, hence the name variable.

```
char ch = 'a';
// some code
Ch = 'l';
```

Rules for naming a variable

- A variable name can have only letters (both uppercase and lowercase letters), digits and underscore.
- The first letter of a variable should be either a letter or an underscore.
- There is no rule on how long a variable name (identifier) can be. However, you may run into problems in some compilers if the variable name is longer than 31 characters.

C is a strongly typed language. This means that the variable type cannot be changed once it is declared. For example:

```
int number = 5;      // integer variable
number = 5.5;        // error
double number;       // error
```

Here, the type of number variable is int. You cannot assign a floating-point (decimal) value 5.5 to this variable. Also, you cannot redefine the data type of the variable to double. By the way, to store the decimal values in C, you need to declare its type to either double or float.

Literals

Literals are data used for representing fixed values. They can be used directly in the code. For example: 1, 2.5, 'c' etc.

Here, 1, 2.5 and 'c' are literals. Why? You cannot assign different values to these terms.

1. Integers

An integer is a numeric literal(associated with numbers) without any fractional or exponential part. There are three types of integer literals in C programming:

decimal (base 10)
octal (base 8)
hexadecimal (base 16)

For example:

Decimal: 0, -9, 22 etc
Octal: 021, 077, 033 etc
Hexadecimal: 0x7f, 0x2a, 0x521 etc
In C programming, octal starts with a 0, and hexadecimal starts with a 0x.

2. Floating-point Literals

A floating-point literal is a numeric literal that has either a fractional form or an exponent form. For example:

-2.0
0.0000234
-0.22E-5
Note: E-5 = 10-5

3. Characters

A character literal is created by enclosing a single character inside single quotation marks. For example: 'a', 'm', 'F', '2', '}' etc.

4. Escape Sequences

Sometimes, it is necessary to use characters that cannot be typed or has special meaning in C programming. For example: newline(enter), tab, question mark etc.

In order to use these characters, escape sequences are used.

Escape Sequences

Escape Sequences	Character
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\?	Question mark
\0	Null character

For example: \n is used for a newline. The backslash \ causes escape from the normal way the characters are handled by the compiler.

5. String Literals

A string literal is a sequence of characters enclosed in double-quote marks. For example:

```
"good"                //string constant
""                  //null string constant
"      "            //string constant of six white space
"x"                  //string constant having a single character.
"Earth is round\n"        //prints string with a newline
```

• Constants

If you want to define a variable whose value cannot be changed, you can use the const keyword. This will create a constant. For example,

```
const double PI = 3.14;
```

Notice, we have added keyword const.

Here, PI is a symbolic constant; its value cannot be changed.

```
const double PI = 3.14;
PI = 2.9; //Error
```

You can also define a constant using the #define preprocessor directive.