C++ Variables, Literals and Constants

In this tutorial, we will learn about variables, literals, and constants in C++ with the help of examples.

Content from https://www.programiz.com/cpp-programming/variables-literals.

C++ 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). For example,

```
int age = 14;
```

Here, *age* is a variable of the `int` data type, and we have assigned an integer value 14 to it.

Note: The `int` data type suggests that the variable can only hold integers. Similarly, we can use the `double` data type if we have to store decimals and exponentials.

We will learn about all the data types in detail in the next tutorial.

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

```
int age = 14;   // age is 14
age = 17;       // age is 17
```

Rules for naming a variable

- A variable name can only have alphabets, numbers, and the underscore `_`.

- A variable name cannot begin with a number.
- It is a preferred practice to begin variable names with a lowercase character. For example, *name* is preferable to *Name*.
- A variable name cannot be a keyword. For example, `int` is a keyword that is used to denote integers.
- A variable name can start with an underscore. However, it's not considered a good practice.

Note: We should try to give meaningful names to variables. For example, *first_name* is a better variable name than *fn*.

C++ 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.

Here's a list of different literals in C++ programming.

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`.

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`

Characters

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

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	Characters
`\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

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

We will learn about strings in detail in the C++ string tutorial.

C++ Constants

In C++, we can create variables whose value cannot be changed. For that, we use the `const` keyword. Here's an example:

```
const int LIGHT_SPEED = 299792458;
LIGHT_SPEED = 2500 // Error! LIGHT_SPEED is a constant.
```

Here, we have used the keyword `const` to declare a constant named `LIGHT_SPEED`. If we try to change the value of `LIGHT_SPEED`, we will get an error.

A constant can also be created using the `#define` preprocessor directive. We will learn about it in detail in the C++ Macros tutorial.

Revision #4
Created 2022-05-03 02:05:46 UTC by Chester Whitwell
Updated 2022-05-03 02:11:04 UTC by Chester Whitwell