

Software

- [C++ Variables, Literals and Constants](#)
- [C++ Data Types](#)
- [data science topics](#)
- [Python Strings](#)

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
<code>\b</code>	Backspace
<code>\f</code>	Form feed

<code>\n</code>	Newline
<code>\r</code>	Return
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab
<code>\\</code>	Backslash
<code>\'</code>	Single quotation mark
<code>\"</code>	Double quotation mark
<code>\?</code>	Question mark
<code>\0</code>	Null Character

String Literals

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

<code>"good"</code>	string constant
<code>""</code>	null string constant
<code>" "</code>	string constant of six white space
<code>"x"</code>	string constant having a single character
<code>"Earth is round\n"</code>	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.

C++ Data Types

In this tutorial, we will learn about basic data types such as int, float, char, etc. in C++ programming with the help of examples.

In C++, data types are declarations for variables. This determines the type and size of data associated with variables. For example,

```
int age = 13;
```

Here, *age* is a variable of type `int`. Meaning, the variable can only store integers of either 2 or 4 bytes.

C++ Fundamental Data Types

The table below shows the fundamental data types, their meaning, and their sizes (in bytes):

Data Type	Meaning	Size (in Bytes)
<code>int</code>	Integer	2 or 4
<code>float</code>	Floating-point	4
<code>double</code>	Double Floating-point	8
<code>char</code>	Character	1
<code>wchar_t</code>	Wide Character	2
<code>bool</code>	Boolean	1
<code>void</code>	Empty	0

Now, let us discuss these fundamental data types in more detail.

1. C++ int

- The `int` keyword is used to indicate integers.
- Its size is usually 4 bytes. Meaning, it can store values from - **2147483648 to 2147483647**.
- For example,

```
int salary = 85000;
```

2. C++ float and double

- `float` and `double` are used to store floating-point numbers (decimals and exponentials).
- The size of `float` is 4 bytes and the size of `double` is 8 bytes. Hence, `double` has two times the precision of `float`. To learn more, visit C++ float and double.
- For example,

```
float area = 64.74;  
double volume = 134.64534;
```

As mentioned above, these two data types are also used for exponentials. For example,

```
double distance = 45E12 // 45E12 is equal to 45*10^12
```

3. C++ char

- Keyword `char` is used for characters.
- Its size is 1 byte.
- Characters in C++ are enclosed inside single quotes `' '`.
- For example,

```
char test = 'h';
```

Note: In C++, an integer value is stored in a `char` variable rather than the character itself. To learn more, visit [C++ characters](#).

4. C++ wchar_t

- Wide character `wchar_t` is similar to the `char` data type, except its size is 2 bytes instead of 1.
- It is used to represent characters that require more memory to represent them than a single `char`.
- For example,

```
wchar_t test = L'א' // storing Hebrew character;
```

Notice the letter L before the quotation marks.

Note: There are also two other fixed-size character types `char16_t` and `char32_t` introduced in C++11.

5. C++ bool

- The `bool` data type has one of two possible values: `true` or `false`.
- Booleans are used in conditional statements and loops (which we will learn in later chapters).
- For example,

```
bool cond = false;
```

https://www.youtube.com/embed/zB9RI8_wExo

6. C++ void

- The `void` keyword indicates an absence of data. It means "nothing" or "no value".
- We will use void when we learn about functions and pointers.

Note: We cannot declare variables of the `void` type.

Putting it all together

Run the following repl to see its results. Looking at the `main.cpp` file, a range of variables of different data types are declared at the top.

`cout` is used with the `+` operator to output the sum of two integers. `cout` is also used to display a message that includes the contents of the string variables, and lastly an if statement is used to present a message based on which letter is stored in the char variable.

Try doing each of the following, running the between each change:

1. Change the string variables to reflect your name
2. Change the values of the integers
3. Change the operator used in the output to `*`
4. Change the letter stored in the char variable to a grade you would like to receive
5. Change the integer variables to float, and assign decimal values.

<https://replit.com/@ChesterWhitwell/C-variables-and-operators?lite=true>

C++ Type Modifiers

We can further modify some of the fundamental data types by using type modifiers. There are 4 type modifiers in C++. They are:

1. `signed`
2. `unsigned`
3. `short`
4. `long`

We can modify the following data types with the above modifiers:

- `int`
- `double`
- `char`

C++ Modified Data Types List

Data Type	Size (in Bytes)	Meaning
<code>signed int</code>	4	used for integers (equivalent to <code>int</code>)
<code>unsigned int</code>	4	can only store positive integers
<code>short</code>	2	used for small integers (range -32768 to 32767)
<code>unsigned short</code>	2	used for small positive integers (range 0 to 65,535)
<code>long</code>	at least 4	used for large integers (equivalent to <code>long int</code>)
<code>unsigned long</code>	4	used for large positive integers or 0 (equivalent to <code>unsigned long int</code>)
<code>long long</code>	8	used for very large integers (equivalent to <code>long long int</code>).
<code>unsigned long long</code>	8	used for very large positive integers or 0 (equivalent to <code>unsigned long long int</code>)
<code>long double</code>	12	used for large floating-point numbers
<code>signed char</code>	1	used for characters (guaranteed range -127 to 127)
<code>unsigned char</code>	1	used for characters (range 0 to 255)

Let's see a few examples.

```
long b = 4523232;
long int c = 2345342;
long double d = 233434.56343;
short d = 3434233; // Error! out of range
unsigned int a = -5; // Error! can only store positive numbers or 0
```

Derived Data Types

Data types that are derived from fundamental data types are derived types. For example: arrays, pointers, function types, structures, etc.

We will learn about these derived data types in later tutorials.

data science topics

Topic : Introduction to data science

- Data scientist skills
- Data scientist responsibilities
- data vs information

Topic : Presenting data science projects

- Planning
- performing
- Data visualization
- Data analytics

Topic : Ethics of data science

- Code of ethics
- Privacy, transparency and security
- Consent
- Fairness and Algorithms

Topic : Data manipulation

- What is data manipulation?
- Tools
- Techniques

Topic : Artificial intelligence

- AI history
- Three phases of AI
- Learning processes
- Strong AI vs. weak AI
- AI ethics
- Case studies
 - business

- education
- healthcare
- finance

Topic : Machine Learning

- Mathematics for machine learning
 - Linear algebra and discrete mathematics
 - Multivariate calculus
 - Statistics
- Machine learning algorithms
- Machine learning pipeline
- Optimisation for machine learning

Topic : Deep learning

- Deep learning vs machine learning
- Natural language processing (NLP)
- Self learning
 - Self-learning agents in games

Python Strings

<https://replit.com/@ChesterWhitwell/String-methods?lite=true>

<https://trinket.io/embed/python/d4301ef9ac>