C++ Variable Types

A variable provides us with named storage that our programs can manipulate. Each variable in C++ has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C++ is case-sensitive −

Variable Definition in C++

A variable definition tells the compiler where and how much storage to create for the variable. A variable definition specifies a data type, and contains a list of one or more variables of that type as follows −

type variable\_list;

Here, **type** must be a valid C++ data type including char, w\_char, int, float, double, bool or any user-defined object, etc., and **variable\_list** may consist of one or more identifier names separated by commas. Some valid declarations are shown here −

int i, j, k;

char c, ch;

float f, salary;

double d;

The line **int i, j, k;** both declares and defines the variables i, j and k; which instructs the compiler to create variables named i, j and k of type int.

Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as follows −

type variable\_name = value;

Some examples are −

extern int d = 3, f = 5; // declaration of d and f.

int d = 3, f = 5; // definition and initializing d and f.

byte z = 22; // definition and initializes z.

char x = 'x'; // the variable x has the value 'x'.

For definition without an initializer: variables with static storage duration are implicitly(Auto by compiler) initialized with NULL (all bytes have the value 0); the initial value of all other variables is undefined.

## Lvalues and Rvalues

There are two kinds of expressions in C++ −

* **lvalue** − Expressions that refer to a memory location is called "lvalue" expression. An lvalue may appear as either the left-hand or right-hand side of an assignment.
* **rvalue** − The term rvalue refers to a data value that is stored at some address in memory. An rvalue is an expression that cannot have a value assigned to it which means an rvalue may appear on the right- but not left-hand side of an assignment.

Variables are lvalues and so may appear on the left-hand side of an assignment. Numeric literals are rvalues and so may not be assigned and can not appear on the left-hand side. Following is a valid statement −

int g = 20;

But the following is not a valid statement and would generate compile-time error −

10 = 20;