# **C++ Declare Multiple Variables**

## Declare Many Variables

To declare more than one variable of the **same type**, use a comma-separated list:

### Example

int x = 5, y = 6, z = 50;  
cout << x + y + z;

## One Value to Multiple Variables

You can also assign the **same value** to multiple variables in one line:

### Example

int x, y, z;  
x = y = z = 50;  
cout << x + y + z;

# **C++ Identifiers**

## C++ Identifiers

All C++ **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

**Note:** It is recommended to use descriptive names in order to create understandable and maintainable code:

### Example

// Good  
int minutesPerHour = 60;  
  
// OK, but not so easy to understand what **m** actually is  
int m = 60;

The general rules for naming variables are:

* Names can contain letters, digits and underscores
* Names must begin with a letter or an underscore (\_)
* Names are case-sensitive (myVar and myvar are different variables)
* Names cannot contain whitespaces or special characters like !, #, %, etc.
* Reserved words (like C++ keywords, such as int) cannot be used as names

# **C++ Constants**

## Constants

When you do not want others (or yourself) to change existing variable values, use the const keyword (this will declare the variable as "constant", which means **unchangeable and read-only**):

### Example

**const** int myNum = 15;  // myNum will always be 15  
myNum = 10;  // error: assignment of read-only variable 'myNum'

You should always declare the variable as constant when you have values that are unlikely to change:

### Example

**const** int minutesPerHour = 60;  
**const** float PI = 3.14;

## Notes On Constants

When you declare a constant variable, it must be assigned with a value:

### Example

Like this:

const int minutesPerHour = 60;

This however, **will not work**:

const int minutesPerHour;  
minutesPerHour = 60; // error

# **C++ User Input**

## C++ User Input

You have already learned that cout (see-out) is used to output (print) values. Now we will use cin (see-in) to get user input.

cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).

In the following example, the user can input a number, which is stored in the variable x. Then we print the value of x:

### Example

int x;   
cout << "Type a number: "; // Type a number and press enter  
cin >> x; // Get user input from the keyboard  
cout << "Your number is: " << x; // Display the input value

#### **Good To Know**

cout is pronounced "see-out". Used for **output**, and uses the insertion operator (<<)

cin is pronounced "see-in". Used for **input**, and uses the extraction operator (>>)