

## **BASICS OF C++**

### **Introduction of C++**

C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming.

C++ is regarded as a middle-level language, as it comprises a combination of both high-level and low-level language features.

C++ was developed by Bjarne Stroustrup starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983.

C++ is a superset of C, and that virtually any legal C program is a legal C++ program.

Note – A programming language is said to use static typing when type checking is performed during compile-time as opposed to run-time.

### **Object-Oriented Programming**

C++ fully supports object-oriented programming, including the four pillars of object-oriented development –

- Encapsulation
- Data hiding
- Inheritance
- Polymorphism

### **Standard Libraries**

Standard C++ consists of three important parts –

- The core language giving all the building blocks including variables, data types and literals, etc.
- The C++ Standard Library giving a rich set of functions manipulating files, strings, etc.
- The Standard Template Library (STL) giving a rich set of methods manipulating data structures, etc.

### **The ANSI Standard**

The ANSI standard is an attempt to ensure that C++ is portable; that code you write for Microsoft's compiler will compile without errors, using a compiler on a Mac, UNIX, a Windows box, or an Alpha.

The ANSI standard has been stable for a while, and all the major C++ compiler manufacturers support the ANSI standard.

## **Learning C++**

The most important thing while learning C++ is to focus on concepts.

The purpose of learning a programming language is to become a better programmer; that is, to become more effective at designing and implementing new systems and at maintaining old ones.

C++ supports a variety of programming styles. You can write in the style of Fortran, C, Smalltalk, etc., in any language. Each style can achieve its aims effectively while maintaining runtime and space efficiency.

## **Use of C++**

C++ is used by hundreds of thousands of programmers in essentially every application domain.

C++ is being highly used to write device drivers and other software that rely on direct manipulation of hardware under realtime constraints.

C++ is widely used for teaching and research because it is clean enough for successful teaching of basic concepts.

Anyone who has used either an Apple Macintosh or a PC running Windows has indirectly used C++ because the primary user interfaces of these systems are written in C++.

## **C++ Environment Setup**

### **Local Environment Setup**

If you are still willing to set up your environment for C++, you need to have the following two softwares on your computer.

#### **Text Editor**

This will be used to type your program. Examples of few editors include Windows Notepad, OS Edit command, Brief, Epsilon, EMACS, and vim or vi.

Name and version of text editor can vary on different operating systems. For example, Notepad will be used on Windows and vim or vi can be used on windows as well as Linux, or UNIX.

The files you create with your editor are called source files and for C++ they typically are named with the extension .cpp, .cp, or .c.

A text editor should be in place to start your C++ programming.

## **C++ Compiler**

This is an actual C++ compiler, which will be used to compile your source code into final executable program.

Most C++ compilers don't care what extension you give to your source code, but if you don't specify otherwise, many will use .cpp by default.

Most frequently used and free available compiler is GNU C/C++ compiler, otherwise you can have compilers either from HP or Solaris if you have the respective Operating Systems.

## **Installing GNU C/C++ Compiler**

### **UNIX/Linux Installation**

If you are using Linux or UNIX then check whether GCC is installed on your system by entering the following command from the command line –

```
$ g++ -v
```

If you have installed GCC, then it should print a message such as the following –

Using built-in specs.

Target: i386-redhat-linux

Configured with: ../configure --prefix=/usr .....

Thread model: posix

gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)

If GCC is not installed, then you will have to install it yourself using the detailed instructions available at <https://gcc.gnu.org/install/>

### **Mac OS X Installation**

If you use Mac OS X, the easiest way to obtain GCC is to download the Xcode development environment from Apple's website and follow the simple installation instructions.

Xcode is currently available at [developer.apple.com/technologies/tools/](http://developer.apple.com/technologies/tools/).

## **Windows Installation**

To install GCC at Windows you need to install MinGW. To install MinGW, go to the MinGW homepage, [www.mingw.org](http://www.mingw.org), and follow the link to the MinGW download page. Download the latest version of the MinGW installation program which should be named MinGW-<version>.exe.

While installing MinGW, at a minimum, you must install gcc-core, gcc-g++, binutils, and the MinGW runtime, but you may wish to install more.

Add the bin subdirectory of your MinGW installation to your PATH environment variable so that you can specify these tools on the command line by their simple names.

When the installation is complete, you will be able to run gcc, g++, ar, ranlib, dlltool, and several other GNU tools from the Windows command line.

## **C++ Basic Syntax**

When we consider a C++ program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what a class, object, methods, and instant variables mean.

- **Object** – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating. An object is an instance of a class.
- **Class** – A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.
- **Methods** – A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variables** – Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

## C++ Program Structure

Let us look at a simple code that would print the words *Hello World*.

```
#include <iostream>
using namespace std;

// main() is where program execution begins.
int main() {
    cout << "Hello World"; // prints Hello World
    return 0;
}
```

Let us look at the various parts of the above program –

- The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header <iostream> is needed.
- The line using namespace std; tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.
- The next line '// main() is where program execution begins.' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.
- The line int main() is the main function where program execution begins.
- The next line cout << "Hello World"; causes the message "Hello World" to be displayed on the screen.
- The next line return 0; terminates main( )function and causes it to return the value 0 to the calling process.

## Compile and Execute C++ Program

Let's look at how to save the file, compile and run the program. Please follow the steps given below –

- Open a text editor and add the code as above.
- Save the file as: hello.cpp
- Open a command prompt and go to the directory where you saved the file.
- Type 'g++ hello.cpp' and press enter to compile your code. If there are no errors in your code the command prompt will take you to the next line and would generate a.out executable file.
- Now, type 'a.out' to run your program.
- You will be able to see 'Hello World' printed on the window.

```
$ g++ hello.cpp
```

```
$ ./a.out
```

```
Hello World
```

Make sure that g++ is in your path and that you are running it in the directory containing file hello.cpp.

You can compile C/C++ programs using makefile. For more details, you can check our 'Makefile Tutorial'.

## Semicolons and Blocks in C++

In C++, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

For example, following are three different statements –

```
x = y;
```

```
y = y + 1;
```

```
add(x, y);
```

A block is a set of logically connected statements that are surrounded by opening and closing braces. For example –

```
{
```

```
cout << "Hello World"; // prints Hello World
```

```
return 0;
```

```
}
```

C++ does not recognize the end of the line as a terminator. For this reason, it does not matter where you put a statement in a line. For example –

```
x = y;
```

```
y = y + 1;
```

```
add(x, y);
```

is the same as

```
x = y; y = y + 1; add(x, y);
```

## C++ Identifiers

A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores, and digits (0 to 9).

C++ does not allow punctuation characters such as @, \$, and % within identifiers. C++ is a case-sensitive programming language. Thus, Manpower and manpower are two different identifiers in C++.

Here are some examples of acceptable identifiers –

```
mohd zara abc move_name a_123  
myname50 _temp j a23b9 retVal
```

## C++ Keywords

The following list shows the reserved words in C++. These reserved words may not be used as constant or variable or any other identifier names.

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

## Trigraphs

A few characters have an alternative representation, called a trigraph sequence. A trigraph is a three-character sequence that represents a single character and the sequence always starts with two question marks.

Trigraphs are expanded anywhere they appear, including within string literals and character literals, in comments, and in preprocessor directives.

Following are most frequently used trigraph sequences –

Trigraph	Replacement
??=	#
??/	\
??'	^
??(	[
??)	]
??!	
??<	{
??>	}
??-	~

All the compilers do not support trigraphs and they are not advised to be used because of their confusing nature.

## Whitespace in C++

A line containing only whitespace, possibly with a comment, is known as a blank line, and C++ compiler totally ignores it.

Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins.

### **Statement 1**

```
int age;
```

In the above statement there must be at least one whitespace character (usually a space) between int and age for the compiler to be able to distinguish them.

### **Statement 2**

```
fruit = apples + oranges; // Get the total fruit
```

In the above statement 2, no whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish for readability purpose.

## Comments in C++

Program comments are explanatory statements that you can include in the C++ code. These comments help anyone reading the source code. All programming languages allow for some form of comments.

C++ supports single-line and multi-line comments. All characters available inside any comment are ignored by C++ compiler.

C++ comments start with `/*` and end with `*/`. **For example –**

```
/* This is a comment */
```

```
/* C++ comments can also  
* span multiple lines  
*/
```

A comment can also start with `//`, extending to the end of the line. **For example** –

```
#include <iostream>  
using namespace std;  
  
main() {  
    cout << "Hello World"; // prints Hello World  
  
    return 0;  
}
```

When the above code is compiled, it will ignore `// prints Hello World` and final executable will produce the following result –

Hello World

Within a `/*` and `*/` comment, `//` characters have no special meaning. Within a `//` comment, `/*` and `*/` have no special meaning. Thus, you can "nest" one kind of comment within the other kind. For example –

```
/* Comment out printing of Hello World:
```

```
cout << "Hello World"; // prints Hello World
```

```
*/
```

## C++ Data Types

While writing program in any language, you need to use various variables to store various information. Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

You may like to store information of various data types like character, wide character, integer, floating point, double floating point, boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

### Primitive Built-in Types

C++ offers the programmer a rich assortment of built-in as well as user defined data types. Following table lists down seven basic C++ data types –

Type	Keyword
------	---------



Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

Several of the basic types can be modified using one or more of these type modifiers –

- signed
- unsigned
- short
- long

The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value which can be stored in such type of variables.

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-2,147,483,648 to 2,147,483,647
signed long int	8bytes	same as long int
unsigned long int	8bytes	0 to 4,294,967,295
long long int	8bytes	-(2 <sup>63</sup> ) to (2 <sup>63</sup> )-1
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	
double	8bytes	
long double	12bytes	
wchar_t	2 or 4 bytes	1 wide character

The size of variables might be different from those shown in the above table, depending on the compiler and the computer you are using.

Following is the example, which will produce correct size of various data types on your computer.

```
#include <iostream>
using namespace std;

int main() {
    cout << "Size of char : " << sizeof(char) << endl;
    cout << "Size of int : " << sizeof(int) << endl;
    cout << "Size of short int : " << sizeof(short int) << endl;
    cout << "Size of long int : " << sizeof(long int) << endl;
    cout << "Size of float : " << sizeof(float) << endl;
    cout << "Size of double : " << sizeof(double) << endl;
    cout << "Size of wchar_t : " << sizeof(wchar_t) << endl;

    return 0;
}
```

This example uses endl, which inserts a new-line character after every line and << operator is being used to pass multiple values out to the screen. We are also using sizeof() operator to get size of various data types.

When the above code is compiled and executed, it produces the following result which can vary from machine to machine –

```
Size of char : 1
Size of int : 4
Size of short int : 2
Size of long int : 4
Size of float : 4
Size of double : 8
Size of wchar_t : 4
```

## **typedef Declarations**

You can create a new name for an existing type using typedef. Following is the simple syntax to define a new type using typedef –

```
typedef type newname;
```

For example, the following tells the compiler that feet is another name for int –

```
typedef int feet;
```

Now, the following declaration is perfectly legal and creates an integer variable called distance – feet distance;

## Enumerated Types

An enumerated type declares an optional type name and a set of zero or more identifiers that can be used as values of the type. Each enumerator is a constant whose type is the enumeration.

Creating an enumeration requires the use of the keyword `enum`. The general form of an enumeration type is –

```
enum enum-name { list of names } var-list;
```

Here, the `enum-name` is the enumeration's type name. The list of names is comma separated.

**For example**, the following code defines an enumeration of colors called `colors` and the variable `c` of type `color`. Finally, `c` is assigned the value "blue".

```
enum color { red, green, blue } c;  
c = blue;
```

By default, the value of the first name is 0, the second name has the value 1, and the third has the value 2, and so on. But you can give a name, a specific value by adding an initializer. For example, in the following enumeration, `green` will have the value 5.

```
enum color { red, green = 5, blue };
```

Here, `blue` will have a value of 6 because each name will be one greater than the one that precedes it.

## 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 –

There are following basic types of variable in C++ as explained in last chapter –

Sr.No	Type & Description
1	<b>bool</b> Stores either value true or false.
2	<b>char</b>

	Typically a single octet (one byte). This is an integer type.
3	<b>int</b> The most natural size of integer for the machine.
4	<b>float</b> A single-precision floating point value.
5	<b>double</b> A double-precision floating point value.
6	<b>void</b> Represents the absence of type.
7	<b>wchar_t</b> A wide character type.

C++ also allows to define various other types of variables, which we will cover in subsequent chapters like Enumeration, Pointer, Array, Reference, Data structures, and Classes.

Following section will cover how to define, declare and use various types of variables.

## 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 initialized with NULL (all bytes have the value 0); the initial value of all other variables is undefined.

## Variable Declaration in C++

A variable declaration provides assurance to the compiler that there is one variable existing with the given type and name so that compiler proceed for further compilation without needing complete detail about the variable. A variable declaration has its meaning at the time of compilation only, compiler needs actual variable definition at the time of linking of the program.

A variable declaration is useful when you are using multiple files and you define your variable in one of the files which will be available at the time of linking of the program. You will use extern keyword to declare a variable at any place. Though you can declare a variable multiple times in your C++ program, but it can be defined only once in a file, a function or a block of code.

### Example

Try the following example where a variable has been declared at the top, but it has been defined inside the main function –

```
#include <iostream>
using namespace std;

// Variable declaration:
extern int a, b;
extern int c;
extern float f;

int main () {
    // Variable definition:
    int a, b;
    int c;
    float f;

    // actual initialization
    a = 10;
```

```
b = 20;
c = a + b;

cout << c << endl ;

f = 70.0/3.0;
cout << f << endl ;

return 0;
}
```

When the above code is compiled and executed, it produces the following result –

```
30
23.3333
```

Same concept applies on function declaration where you provide a function name at the time of its declaration and its actual definition can be given anywhere else. For example –

```
// function declaration
int func();
int main() {
// function call
int i = func();
}
```

```
// function definition
int func() {
return 0;
}
```

## 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;
```

## Variable Scope in C++

A scope is a region of the program and broadly speaking there are three places, where variables can be declared –

- Inside a function or a block which is called local variables,
- In the definition of function parameters which is called formal parameters.
- Outside of all functions which is called global variables.

We will learn what is a function and its parameter in subsequent chapters. Here let us explain what are local and global variables.

## Local Variables

Variables that are declared inside a function or block are local variables. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own. Following is the example using local variables –

```
#include <iostream>
using namespace std;

int main () {
    // Local variable declaration:
    int a, b;
    int c;

    // actual initialization
    a = 10;
    b = 20;
    c = a + b;

    cout << c;

    return 0;
}
```

## Global Variables

Global variables are defined outside of all the functions, usually on top of the program. The global variables will hold their value throughout the life-time of your program.

A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration. Following is the example using global and local variables –

```
#include <iostream>
using namespace std;

// Global variable declaration:
int g;

int main () {
    // Local variable declaration:
    int a, b;

    // actual initialization
    a = 10;
    b = 20;
    g = a + b;

    cout << g;

    return 0;
}
```

A program can have same name for local and global variables but value of local variable inside a function will take preference. **For example** –

```
#include <iostream>
using namespace std;

// Global variable declaration:
int g = 20;

int main () {
    // Local variable declaration:
    int g = 10;

    cout << g;

    return 0;
}
```



When the above code is compiled and executed, it produces the following result –

10

## Initializing Local and Global Variables

When a local variable is defined, it is not initialized by the system, you must initialize it yourself. Global variables are initialized automatically by the system when you define them as follows –

Data Type	Initializer
int	0
char	'\0'
float	0
double	0
pointer	NULL

It is a good programming practice to initialize variables properly, otherwise sometimes program would produce unexpected result.

## C++ Basic Input /Output

The C++ standard libraries provide an extensive set of input/output capabilities which we will see in subsequent chapters. This chapter will discuss very basic and most common I/O operations required for C++ programming.

C++ I/O occurs in streams, which are sequences of bytes. If bytes flow from a device like a keyboard, a disk drive, or a network connection etc. to main memory, this is called input operation and if bytes flow from main memory to a device like a display screen, a printer, a disk drive, or a network connection, etc., this is called output operation.

### I/O Library Header Files

There are following header files important to C++ programs –

Sr.No	Header File & Function and Description
1	<iostream>  This file defines the cin, cout, cerr and clog objects, which correspond to the standard input stream, the standard output stream, the un-buffered standard error stream and the buffered standard error stream, respectively.
2	<iomanip>  This file declares services useful for performing formatted I/O with so-called parameterized stream

	manipulators, such as setw and setprecision.
3	<fstream>  This file declares services for user-controlled file processing. We will discuss about it in detail in File and Stream related chapter.

### The Standard Output Stream (cout)

The predefined object cout is an instance of ostream class. The cout object is said to be "connected to" the standard output device, which usually is the display screen. The cout is used in conjunction with the stream insertion operator, which is written as << which are two less than signs as shown in the following example.

```
#include <iostream>

using namespace std;

int main() {
    char str[] = "Hello C++";

    cout << "Value of str is : " << str << endl;
}
```

When the above code is compiled and executed, it produces the following result –

Value of str is : Hello C++

The C++ compiler also determines the data type of variable to be output and selects the appropriate stream insertion operator to display the value. The << operator is overloaded to output data items of built-in types integer, float, double, strings and pointer values.

The insertion operator << may be used more than once in a single statement as shown above and endl is used to add a new-line at the end of the line.

### The Standard Input Stream (cin)

The predefined object cin is an instance of istream class. The cin object is said to be attached to the standard input device, which usually is the keyboard. The cin is used in conjunction with the stream extraction operator, which is written as >> which are two greater than signs as shown in the following example.

```
#include <iostream>

using namespace std;
```

```
int main() {  
    char name[50];  
  
    cout << "Please enter your name: ";  
    cin >> name;  
    cout << "Your name is: " << name << endl;  
}
```

When the above code is compiled and executed, it will prompt you to enter a name. You enter a value and then hit enter to see the following result –

```
Please enter your name: cplusplus  
Your name is: cplusplus
```

The C++ compiler also determines the data type of the entered value and selects the appropriate stream extraction operator to extract the value and store it in the given variables.

The stream extraction operator >> may be used more than once in a single statement. To request more than one datum you can use the following –

```
cin >> name >> age;
```

This will be equivalent to the following two statements –

```
cin >> name;  
cin >> age;
```