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
```

C++ Keywords

The following list shows the reserved words in C++. These reserved words are usually written in lowercase with special meanings and may not be used as constant or variable or any other identifier names. It should be noted that these reserved words may not be redefined or used other than those specified for them.

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	

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 valid identifiers:

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

Here are some examples of invalid identifiers:

```
7-up                    salim!
$2                      no#
```

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	-2,147,483,648 to 2,147,483,647

unsigned long int	8bytes	0 to 4,294,967,295
long long int	8bytes	-(2^63) to (2^63)-1
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615
Float	4bytes	
Double	8bytes	
long double	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
```

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++:

Type	Type & Description
bool	Stores either value true or false.
char	Typically a single octet (one byte). This is an integer type.
int	The most natural size of integer for the machine.
float	A single-precision floating point value.
double	A double-precision floating point value.
void	Represents the absence of type.
wchar_t	A wide character type.

C++ also allows to define various other types of variables, like **Enumeration, Pointer, Array, Reference, Data structures,** and **Classes**.

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.

Variable Initialization in C++

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 –

```
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.

Example: Try the following example :

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

  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;
}
```

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 it's 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 = 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.