First Look At C++ Program

Let us start with a simple example of a C++ program that prints a string on the screen.

Program to print a string on the screen.

```
// My first C++ program
#include<iostream.h> or <iostream>
  using namespace std;
  main()
   {
   cout<<"welcome to C++ Programming";
   return 0;
   }
```

// My first C++ Program

This is a comment. All lines beginning with // are comments.

#include<iostream.h> or #include<iostream>

Statements that begin with # sign are directives for the preprocessor. That means, these statements are processed before compilation takes place. The #include<iostream.h> statement tells the compiler's preprocesor to include the header file iostream or iostream.h.

using namespace std;

This statement includes all C++ Standard libraries.

main()

The main() function is the point by where all C++ programs begin their execution.

cout<<"welcome to C++ Programming";

The cout is the standard output stream in C++ and the above statement inserts a sequence of characters - "welcome to C++ Programming" into the output stream (ie the screen of the monitor)

return 0;

The return instruction makes the main() to finish and it returns

the case it is returning 0 (zero).
Returning 0 is the way of telling that program has terminated normally ie. it has not found any errors during its execution.

Reason ~~Reason to~~ :

Why Include iostream.h ?

The header file iostream or iostream.h is included in every C++ program to implement input/output facilities.

I/O facilities are implemented through iostream which is I/O Library

Predefined streams in I/O Library

Files are implemented as streams of bytes.
Input operations are supported by istream class and output-stream operations are supported by ostream class.
The predefined stream objects for input, output and error are as follows.

cin (console input) : This is istream class object tied to standard input.

cout (console output) : This is an ostream class object tied to standard output.

cerr (console error) : This is an ostream class object tied to standard error.

Comments in a C++ Program:

The purpose of comments is only to allow the programmer to insert description to enhance readability or understandability of the program.

There are two ways to insert comments in C++ programs:

(i) Single line comments with //

The compiler simply ignores everything following // in the same line.

(ii) Multiline or block comments with /*...*/ :

Everything that falls between /* and */ is considered a comment even though it is spread across many lines.

```
// A sample program
#include <iostream>     // includes header files
main()
 {
 cout << "Hello!";
    /* This is a simple program with just one output statement
       written for the purpose of explaining comments */
 }
```

I/O operators:

Output operator "<<" is also called stream insertion operator is used to direct a value to standard output.

Example:

```
cout <<" The sum of 35+7= ";
cout << 35+7;
```

The two statements will produce the following output:

The sum of 35+7=42

Input operator: ">>" also known as stream extraction operator is used to read a value from standard input.

Example:

```
# include <iostream>
using namespace std;
main()
 {int value1, value2, sum;
 cout<<" Enter first value";
 cin>>value1;
 cout <<" Enter second value";
 cin >> value2;
  sum = value1 + value2;
  cout <<" The sum of given values is:";
  cout << sum;
  return 0;
 }
```

Output: Enter first value: 8
Enter second value: 9
The sum of given values is: 17

Cascading of I/O Operators

The multiple use of input or output operators (">>" or "<<") in one statement is called cascading of I/O operators.

Example: `cout << "The result of 8-2 is " << 8-2;`
`cin >> value1 >> value2;`

Role of compiler:

The compiler's job is to analyze the program code for "correctness". If the meaning of the program is correct then compiler can not detect errors.

Some common forms of program errors are given below:

1. Syntax errors:

Syntax refers to the formal rules governing the construction of valid statements in a language.

Syntax errors occur when rules of a programming language are misused ie, when a grammatical rule of C++ is violated.

Eg:
```
main()
{
int a,b:
cin>>a>>b;
cout<<a+b,
return 0
}
```

`int a,b:` → This statement is terminated by : rather than ;
`cout<<a+b,` → This statement is terminated by , rather than ;
`return 0` → missing ; in this statement.

2. Semantics Error

Semantic errors occur when statements are not meaningful. Semantics refers to the set of rules which give the meaning of a statement.

Eg: `x * y = z;` will result in a semantical error because an expression can not come on the left side of an assignment statement

3. Type Errors:

Data in C++ has an associated data type. The value 7 for instance, is an integer. 'a' is a character constant and "hi" is a string.

If a function is given wrong type of data, type error is signalled by the compiler.

If string was given in place of integer then it is a type error.

4. Runtime Errors (Execution errors).

A runtime error is that occurs during the execution of a program. It is caused of some invalid operation taking place.

Eg: If a program is trying to open a file which does not exist, divide by zero are some runtime errors.

5. Logical Errors

A logical error is that error which causes a program to produce incorrect or undesired output.

Eg: If we are trying to print the table of a number 5 and if we write as follows

```
ctr = 1
while(ctr>10)        → Logical error
 {
 cout << n * ctr;
   ctr = ctr + 1;
 }
```

Here the loop will not be executed even once as the condition ctr > 10 is not fulfilled at all. Therefore no output will be produced. Such an error is logical error.