BASIC STRUCTURE OF C++ PROGRAM

1. Header files

The sub directory called INCLUDE contains header files. These files are text files, like the ones you generate with a word processor or the turbo C++Editor. Header files can be combined with your program before it is compiled. Each header files has a **.h** files extension.

```
#include “stdio.h”
```

OR

```
#include <stdio.h>
```

.
This is the method to write the header files in the C++Editor.

2. Function Definition

All C++ programs are divided into units called „functions‟. Every C++ program consists of one or more function. Consider the following program:

```
void main (void)
{
   printf (“ this is a program” ) ;
}
```

The above function program has only one function “main”, it is the first function executed.
The word “void” preceding “main” specifies that the function “main” will not return a value.
The second “void” in parenthesis specifies that the function takes no argument.

3. Delimiters

The braces after the function definition signal the beginning and end of the body if the function. The opening brace **({)** indicates a block of code that forms a distinct unit is about to begin. The closing brace **(})** terminates the block of code.
Braces are also used to delimit blocks of code ion situations other than function. They are used in loops and decision making statements within programs.

4. Statement terminator

A statement in C++language is terminated with a semicolon.

5. The printf( ) function

The **printf( )** functions causes the phrases in quotes to be printed on screen. The printf function is always followed by parenthesis containing the phrase to be printed surrounded by quotes. As C++language distinguishes between uppercase and lowercase characters, thus the function **PRINTF( )** and **Printf( )** are not the same as **printf( )** .

printf() can be used to print numbers , string or characters.

Consider the following program line is:

printf(“this is a program”) ;

printf() can be used to print numbers , string or characters.

6. Format Specifiers

Format Specifiers tell the printf statement where to put the text and how to display the text. The various format specifiers are:

%d => integer
%C=> character
 %f => floating point etc.

7. Printing numbers

The printf( ) functions uses a unique format for printing constants and variable.
Consider the program:

```
void main (void)
{
   printf (“ number one %d” , 1) ;
}
```

It will print „1‟ instead of „%d‟. Similarly you can use the printf function to generate output according to your desired format.

8. Printing characters

The printf function can also print characters separately by utilizing its specific format. Another way to define the character:

```
void main (void)
{
   printf (“ %C is a character” , „a‟ ) ;
}
```

This program will print „a‟ instead of „%c‟. Not only a single character is printed using this function but you can print number of character at a time in a single statement and even string and character are also printed using appropriate format. Consider the following program:

```
void main (void)
{
   printf (“ %C is a pronounced as %s ” , „j‟ , “jay”) ;
}
```

The output of the above program: **j is pronounced as jay**

9. Escape Sequences

Escape Sequence causes the program to escape from the normal interpretation of a string, so That the next character is recognized as having a special meaning. The back slash “\” Character is called the Escape Character”. The escape sequence includes the following:

\n => new line
\b => back space
\r => carriage return
\” => double quotations
\\ => back slash   etc.