**History of C++**

C++ is an OOP language .It was developed by Bjarne Stroustrup at AT&T Bell laboratories in USA, in the early 1980's. Stroustrup wanted to combine the best of features of C and Simula67 and create a more powerful language that could support object oriented programming features and still retain the power & elegance of C, the result was C++. Therefore, C++ is an extension of C with a major addition of the class construct feature of Simula67. Since the class was a major addition to the original C language, Stoutstrup initially called the new language ‘C with Classes’. However, later in 1983, the name was changed to C++. The idea of C++ comes from the C increment operator ++, thereby suggesting that C++ is an augmented version of C.

During the early 1990's the language underwent a number of improvements and changes. In November 1997, the ANSI/ISO standards committee standardized these changes and added several new features to the language specifications.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM FEATURES**

Like C, the C++ program is a collection of function. The above example contain only one function main(). As usual execution begins at main(). Every C++ program must have a main(). C++ is a free form language. With a few exception, the compiler ignore carriage return and white spaces. Like C, the C++ statements terminate with semicolons.

1. **Comments**

Comments are an important part of any program. They are used to give the more explanations about the program statements. The compiler ignores comments, so they do not add to the file size or execution time of the executable program. Comments start with a double slash (//) and terminate the end of the line.

The comments /\* and \*/ are used for multiline comments. We can use either or both styles in our programs.

// this is an example of

//C++ Program to illustrate

//Some of its features

OR

/\* this is an example of

C++ program to illustrate

Some of its features \*/

1. **Input using (Cin) or (>>) or ( extraction ) or |(get from operator)**

The cin is a predefined object in C++ that corresponds to the standard input stream. The input stream means the flow of data from input device. Here the input stream represents the keyboard.



The operator >> is known as extraction or get from operator. It extracts (or takes) the value from the object on its left and places it in the variable on its right.

Ex: cin>>a

1. **Output using (Cout) or (<<) or (insertion) or (put to operator)**

The cout statement displays the string written within the quotation marks on the screen. <<. The cout is a predefined object that represents the standard output stream in C++. The word stream means flow of data. The output stream means flow towards the output. The operator << is called the insertion or put to operator. It inserts the contents of the variable on its right to the object on its left.

Ex: cout<< “Welcome to C++ programming language.”



1. **Cascading of I/O Operators**

The multiple use of << or >> operators in one statement is called as cascading. The extraction operator >> can be cascaded with cin, that allows the user to enter a series of values. Ex: cin>>num1>>num2>>num3;

The insertion operator << can be cascaded with cout that displays many variables values and strings in a single line. Ex: cout<<”Mark1=”<<m1<<”Mark2=”<<m2;

**Example for cout <<**

* Cout << “Sum = “ << sum << “\n”
* << “Average = “ << average << “\n”;

(This is one statement but provides two line of output. )

If you want only one line of output, the statement will be:

* Cout << “Sum = “ << sum << “,”
* << “Average = “ << average << “\n”;
* ***The output will be:***
* ***Sum = 14, average = 7***

1. **The iostream File**

We have used the following #include directive in the program:

#include <iostream>

This directive causes the preprocessor to add the contents of the iostream file to the program. It contains declarations for the identifier cout and the operator «. Some old versions of C++ use a header file called iostream.h. This is one of the changes introduced by ANSI C++. (We should use iostream.h if the compiler does not support ANSI C++ features.) The header file iostream should be included at the beginning of all programs that use input/output statements.

* **E.g:- <float.h> new version <cfloat>**
* **<math.h> new version <cmath>**

<stdio.h>, <iomanip.h> , <string.h>, <memory> , <utility>

1. **Namespace**

Namespace is a new concept introduced by the ANSI C++ standards committee. This defines a scope for the identifiers that are used in a program. For using the identifiers defined in the namespace scope we must include the using directive, like

**using namespace std;**

Here, std is the namespace where ANSI C++ standard class libraries are defined. All

ANSIC++programs must include this directive. This will bring all the identifiers defined in std to the current global scope. Using and namespace are the new keywords of C++.

7) **Return Type of main()**

In C++, main () returns an **integer value** to the operating system. Therefore, every main () in C++ should end with a return (0) statement; otherwise a warning an error might occur. Since main () returns an integer type for main () is explicitly specified as **int.**

**Note that the default return type for all function in C++ is int. The following main without type and return will run with a warning:**

**main ()**

{

---------------.

\_--------------

}

**8) variables :**

Variables are the name given to identify the elements in the program.

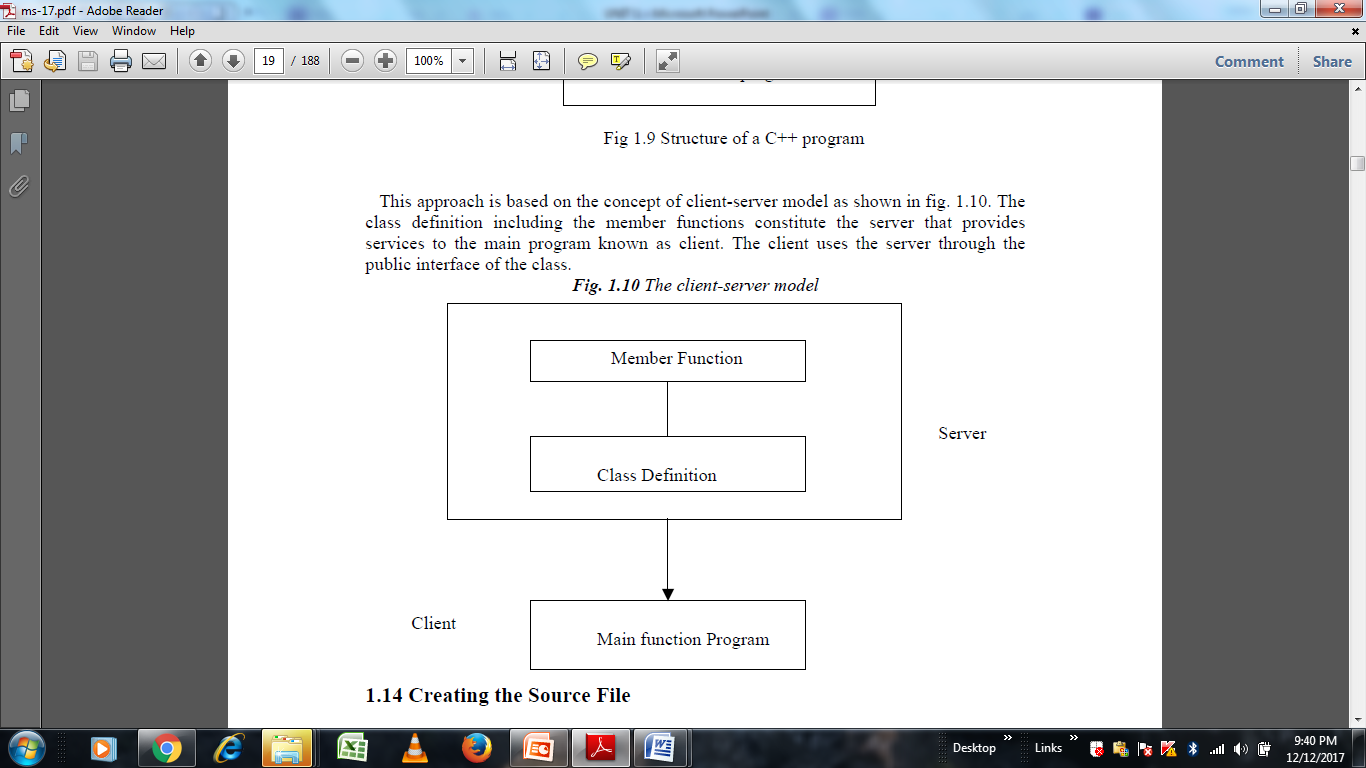
**Float num1,num2,sum, avg;**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**2) Structure of C++ Program**

**It is a common practice to organize a program into three separate files.** The class declarations are placed in a header file and the definitions of member functions go into another file. This approach enables the programmer to separate the abstract specification

 of the interface (class definition) from the implementation details (member functions definition). Finally, the main program that uses the class is placed in a third file which "includes" he previous two files as well as any other files required. This approach is based on the concept of Class definition **client-server model as shown in**



The class definition include member function constitute the server that provides services to main program known as client. The client uses the server through the public interface of the class.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*