Introduction and Definitions

1- Computer System

A **Computer** is an electronic device that can input data, process data and output data (Data are any kind of information that can be codified in some manner and input into the computer) accurately and at great speed then it can perform computations and make logical decisions faster than human beings can. A computer system consists of:-

1**) Hardware**: - Various devices comprising a computer like; Keyboard, screen, mouse, disks, memory, CD-ROM, and processing units.

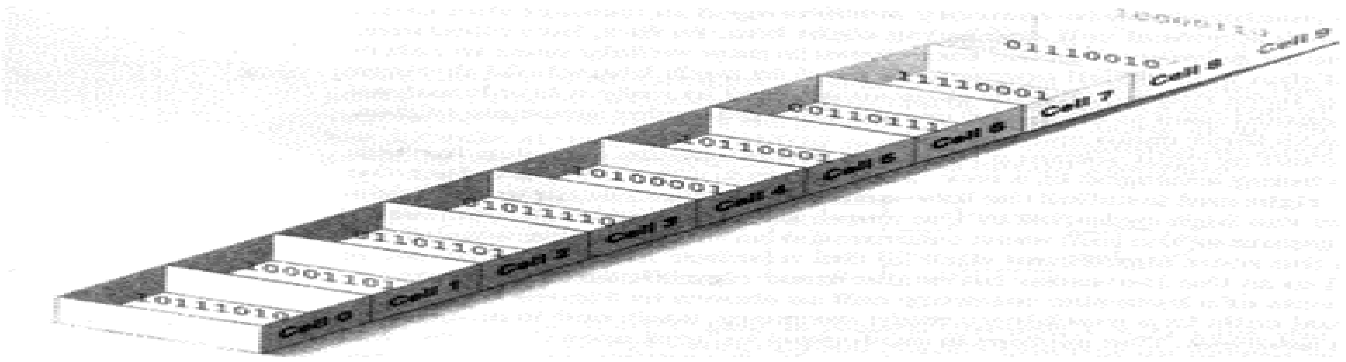
2) **Software: -** Programs that run on a computer.

Programs (also called applications) is a sequence of instructions written in some form of programming language. These instructions tell a computer what to do, and generally how to do it. Programs can contain anything from instructions to solve math problems; Operating Systems, Office Programs, Web Browsers & Email Clients.

A solution to a problem is called an **algorithm;** it describes the sequence of steps to be performed for the problem to be solved.

Programming language:-is a means of communication between a human being (programmer) and a computer. Programming languages can basically be divided in to three categories:-

1- **Low level :-**The lower level in computer "languages" are:-
Machine code (also called binary) is the lowest form of a low-level language. Machine code consists of a string of 0s and 1s, which combine to form meaningful instructions that computers can take action on.



Assembly language (also called ASM), is just above machine code on the scale from low level to high level. It is a human-readable translation of the machine language instructions the computer executes.

01	LOAD
02	ADD
03	STORE
04	HALT

2- **Middle level: -** As C++ and Java languages which are combination between low level language instructions and high level languages instructions.

3- **High level**:-As Basic, Pascal, FORTRAN…etc. which a single statement translates into one or more machine language instructions.
The high level language and middle level language are translated into machine language which is made up of binary-coded instructions, that is used directly by the computer called compiler program.

A Compiler:- is a program used to translate the source code, one instruction at time, into machine code.

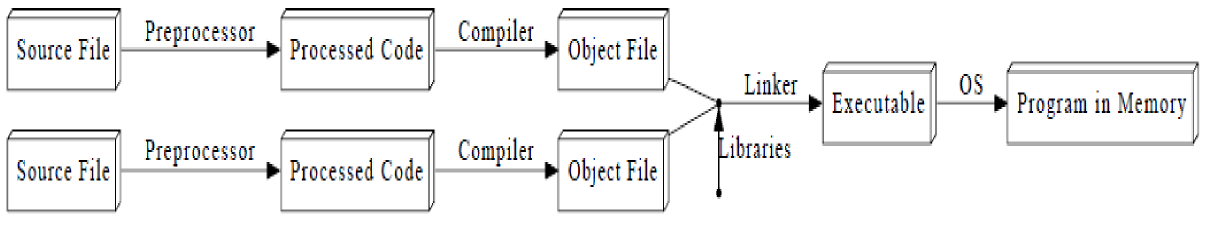
2- History of C++ Language:

In 1970 two programmers, Brian Kernighan and Dennis Ritchie, created a new language called C. (The name came about because C was preceded by the old programming language they were using called B.) C was designed with one goal in mind: writing operating systems. In 1980 Bjarne Stroustrup started working on a new language, called "C with Classes." This language improved on C by adding a number of new features, the most important of which was classes. This language was improved, augmented, and finally became C++, as an enhancement to the C programming language. Enhancements Started with the addition object-oriented concepts like classes, followed by, among many features, virtual functions, operator overloading, and exception handling. Today, C++ is nearly twice the size of the C language. Needless to say, C++ is one of the most powerful computer languages ever devised.

3- Programs from Conception to Execution:-

C++ programs are written in a high-level language using letters, numbers, and the other symbols you find on a computer keyboard. Computers actually execute a very *low-level language* called *machine code*. So, before a program can be used, it must undergo several transformations:-

1- ***Editing a file*:-** Programs start out as an idea in a programmer's head. A programmer uses a **text editor** to create or modify files containing C++ code. Code is also known as **source code**. A file containing source code is called a **source file**.
2- ***Preprocessing****:-*which applies some modifications to the source code, before being fed to the compiler.
3- ***Compiling:-****Translates source code into* ***object code (machine code).****Checks for* ***syntax errors*** *and* ***warnings***
4- ***Linking:-***Combines the program object code with other object code to produce the executable file (a set of machine-language instructions).The other object code can come from the **standard Library**, other libraries, or object files that you have created.
5- ***Loading:-***loading the program into memory.
6- ***Running:-***Run the program.



The main structure of C++ programs:-

```
1.      // program description (header comment)
2.      #include directives (preprocessor directives)
3.      int main ( )
4.      {
5.      Program statement (function body);
6.      Constant declarations;
7.      Variable declarations;
8.      Executable statement;
9.      return 0
10.     }
```

A **comment** is descriptive text used to help a reader of the program understand its content. Improve program readability. Ignored by compiler. Single-line comment.
Line (2)
Commands starting with a hash ('#') are **"Pre-processor Directives"**, The **#include** directive tells the compiler to include some already existing C++ code in your program. The included file is then linked with the program. If you do not include it you cannot use these objects or anything else "declared" in this file
C++ Standard Library header files:-

1. Each contains a portion of the Standard Library.
2. Function prototypes for the related functions
3. Definitions of various class types and functions
4. Constants needed by those functions
5. Header file names ending in **.h** , <iostream.h> declares the cout, cin and endl objects. <math.h> contains function prototypes for math library function.

cin
- Standard input stream (**cin** >> *variableName1* >> *variableName2* >> *variableName3* ;)
- Normally keyboard

cout
- Standard output stream (**cout** << a string or an expression ;) or (**cout** << S_1 <<S_2<<S_3<<**endl**;). A string is any sequence of characters enclosed in double-quotes. The symbol << is an output operator which takes an output stream as its left operand and an expression as its right operand, and causes the value of the latter to be sent to the former.
- Normally computer screen

Line (3)
Overall structure of a C++ program contains one function named main (), called the **driver function,** All other functions are invoked from main (), No main function then nothing to run & cannot build the program.

Line (4)

Function body: The statements inside a function (enclosed in braces). Each statement inside the function must be terminated with a semicolon. A *statement* is simply a single line of code that performs some action. Many programmers also use the term *expression* when referring to a statement.
Line (5)

return: A keyword causing the appropriate value to be returned from the function. The statement return 0 in the main( ) function causes the program to end.

The program below shows the main structure of C++ programs and when run, simply
outputs the message Hello World.
Line (6)
This brace marks the end of the body of main.

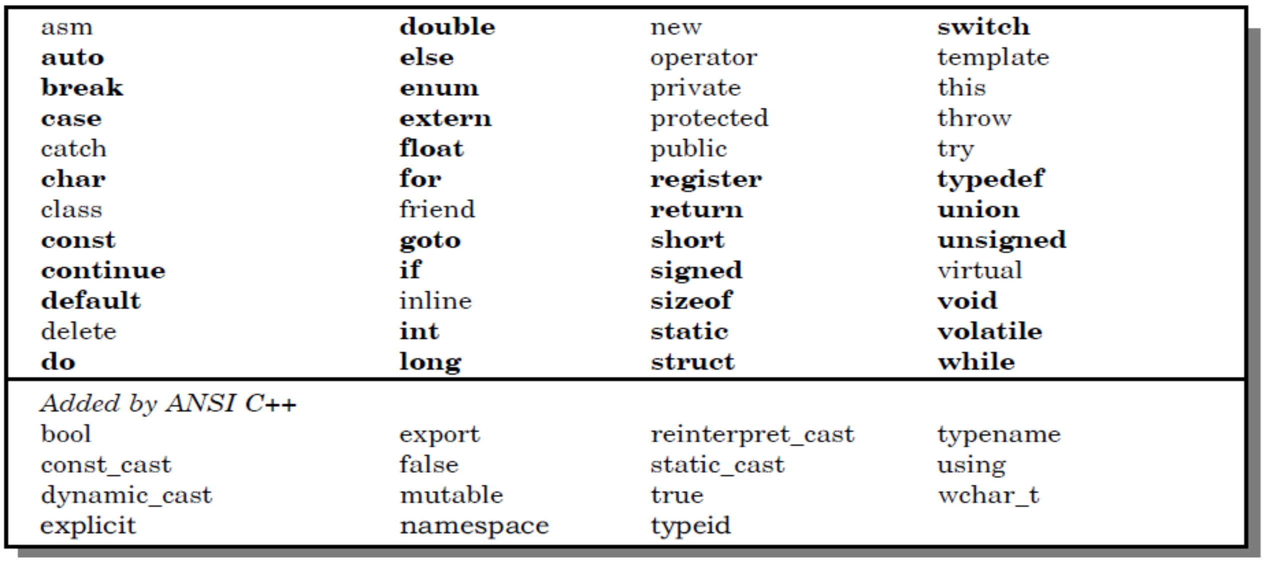
```
// This program prints awelcome message.
#include <iostream.h>
int main()
{
cout << "Hello world\n";
return 0;}
```

In this example the effect is that the string "Hello word" is sent to cout, causing it to be printed on the computer monitor screen.

Token:-
Is the smallest element in the C++ language. It may be a single character or a sequence of characters to form a single item.Tokens can be:-

- ❖ String
- ❖ constants
- ❖ Keywords
- ❖ Names (identifiers)
- ❖ Punctuation
- ❖ Operators

☒ **Keywords**:-
are the commands that make up the C++ languge. These words should be written in small letters or lowercase letters. Each keyword has a predefined purpose in the language(Words with special meaning to the compiler). Some of the keyword used in c++ are int, float, char, double, long, void, for, while, do, if, else ..etc as shown in the table(1).
Table (1) C++ Keywords.



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

☒ **Identifiers and Constants** :-
Identifier:- refer to names of variables(A variable is simply a place in memory set aside to hold data of a particular type. It is a specific section of the computer's memory that has been reserved to hold some data. It is called a variable because its value or content can **vary**.), functions, array, and various other user-defined , greated by the programmer. They are fundamental requirement of any language. Each language has it's own rules for naming these identifiers. The following rules are:-

1. Only alphabetic characters, digits and underscores are permitted.
2. The name cannot start with a digit.
3. Uppercase and lowercase letters are distinct.
4. A declared keyword cannot be used as a variable name
5. Maximum length of an identifier = 1024 characters.

Correct	**Incorrect**
Count	1count
test23	hi!there
high_balance	High.....balance
Xsum	Tax-rate
X_sum	W1.1

Constants (often referred to as Literals):- are data items that never change their value during the execution of the program. The following types of literals are available in C++:-

(i) integer-constants:-

Integer constants are whole numbers without any fractional part. It may contain either + or – sign, but decimal point or commas does not appear in any integer constant. C++ allows three types of integer constants:-

1. Decimal (Base 10) :-It consists of sequence of digits and should not begin with 0 (zero). For example 124, - 179, + 108.
2. Octal (Base 8):- It consists of sequence of digits starting with 0 (zero). For example, 014,012
3. Hexadecimal (Base 16):- It consists of sequence of digits preceded by ox or OX. For example OXD, OXC.

(ii) character-constants:-

A character constant in C++ must contain one or more characters and must be enclosed in single quotation marks. For example ‘A’, ‘9’, etc.

C++ allows nongraphic characters which cannot be typed directly from keyboard, e.g., backspace, tab, carriage return etc. These characters can be represented by using a escape sequence. An escape sequence represents a single character. The following table gives a listing of common escape sequences as shown in Table (2).

Table (2) Escape Sequence Nongraphic Character

\a	Bell (beep)
\b	Backspace
\f	Formfeed
\n	Newline or line feed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\?	Question mark
\ \	Backslash
\ ’	Single quote
\ ”	Double quote
\ xhh	Hexadecimal number (hh represents the number in hexadecimal)
\ 000	Octal number (00 represents the number in octal)
\0	Null

Character:
Each character in the computer is represented by ASCII (American Standard Code for Information Interchange). For example, the character *A* has the ASCII code 65, and the character *a* has the ASCII code 97.

Character	ASCII	Integer equivalent
new line,\n	0001010	10
$	0100100	36
%	0100101	37
3	0110011	51
A	1000001	65
A	1100001	97
B	1100010	98

A tatal of 128 characters can be represented in ASCII.
Symbolic Constant:-
Is declared by using the keyword const. It is ususlly defined with uppercase identifier. For example:
Const double PI=3.1415;
Note : the value of symbolic constant cannot be changed.

(iii) floating-constants
They are also called real constants. They are numbers having fractional parts. They may be written in fractional from or exponent form. A real constant in fractional form consists of signed or unsigned digits including a decimal point between digits. For example 3.0, -17.0, -0.627 etc. A real constant in exponent form has two parts: a mantissa and an exponent. The mantissa is either an integer or a real constant followed by letter E or e and the exponent which must be an integer. For example 2E03, 1.23E07.
(iv) String Literals
A sequence of character enclosed within double quotes is called a string literal. For example “COMPUTER” .
Punctuators
The following characters are used as punctuators in C++:-
Brackets [ ] opening and closing brackets indicate single and multidimensional array subscript.
Parentheses ( ) opening and closing brackets indicate functions calls, function parameters for grouping expressions etc.
Braces { } opening and closing braces indicate the start and end of a compound statement.
Comma , it is used as a separator in a function argument list.
Semicolon ; it is used as a statement terminator.
Colon : it indicates a labelled statement or conditional operator symbol.
Asterisk * it is used in pointer declaration or as multiplication operator.
Equal sign = it is used as an assignment operator.

Data types :-
Are used to specify the types of values that will be contained in variables.
In C++, data types are classified into:-
- Numeric data types are either integers (short, int, long) or floating point values (float, double, long double).
- Nonnumeric data types are alphabetic and special characters (char).

Type	Size(Bytes) (Bits)	Range
Char	1 (8bits)	–128 to 127
Int	2or 4 (16 or 32)	–32,768 to 32,767 or–2,147,483,648 to 2,147,483,647
Short int	2(16)	–32,768 to 32,767
Long int	4 (32)	–2,147,483,648 to 2,147,483,647
Float	4 (32)	1.18E–38 to 3.40E+38
Double	8 (64)	2.23E–308 to 1.79E+308
Long double	10 (80)	3.37E–4932 to 1.18E+4932

4- <u>Variable declaration</u>

Before a variable appears in programs, you must declare it, the general variable declaration is:-
For example, you might declare a variable in the following manner:-
int j;
Then, you have just allocated four bytes of memory; you are using the variable j to refer to those four bytes of memory. You are also stating that the only type of data that j will hold, is whole numbers. (The int is a data type that refers to integers). Now, whenever you reference j in your code, you are actually referencing the contents being stored at a specific address in memory. You can declare more than one variable on a single line.
int x, i_age, sgnal9;
All three variables are of type int.
All variables have two important attributes:
 A type: which is established when the variable is defined (e.g., integer, real, characters). Once defined, the type of a C++ variable cannot be changed.
 A value: which can be changed by assigning a new value to the variable. When a variable is define (declare), its value is undefined until it is actually assigned one. The assigning of a value to a variable for the first time is called **initialization**.
int x;
float y;
x=5;
y=4.7;
It is possible to declare a variable and assign it at the same time.
int num = 1500;
float number = 0.451;
It is important to ensure that a variable is an initialized before it is used in any computation. A default value is simply some starting value that the variable will hold, by default, if no other value is placed into it.

7. Constant declaration

With the const prefix you can declare constants with a specific type in the same way as you do with the variable, but you should assign a value at the same step.
const data type = value;
Ex: const int x=100;
const float pi=3.14;

Example(1):- Write a C++ program to read two integer numbers, then find their sum?

Solution:-

```
#include <iostream.h>
 int main()
 {
int integer1, integer2, sum; // declaration
cout << "Enter first integer\n"; // prompt
 cin >> integer1; // read an integer
 cout << "Enter second integer\n"; // prompt
 cin >> integer2; // read an integer
 sum = integer1 + integer2; // assignment of sum
 cout << "Sum is " << sum << endl; // print sum
 return 0; // indicate that program ended successfully
 }
```

Example(2):- write a C++ program that computes the area of rectangle with given fixed lengh and width?

Solution:-

```
#include <iostream.h>
 int  main()
 {
int lengh, width, area;
lengh=4;
width=5;
area=lengh* width;
cout<< " the area of rectangle is:" <<area<<end1;
return 0;
}
```

Example(3) :- write a C++ program that reads the lengh and width of a rectangle, and prints its area?

Solution:-

```
#include <iostream.h>
 void  main()
 {
int lengh, width, area;
cout<< "enter lengh:"<<end1;
cin>> lengh;

cout<< "enter width :"<<end1;

cin>> width;

area=lengh* width;
cout<< " the area of rectangle is:" <<area<<end1;
return 0;}
```

Example(4) :- write a C++ program that reads the radius of a circle and prints its area and perimeter?

Solution:-

```
# include < iostream.h>

void main ( )

{

Const double PI = 3.1415;

Double radius , area , perimeter;

Cout << " enter the radius : " << end1;

Cin >> radius ;

Area = PI * radius * radius ;

Perimeter = 2* PI * radius ;

Cout << " the area of circle: " << area << end1 ;

 Cout << " the of Perimeter circle: " << Perimeter << end1;

return 0;

}
```