Program:

A set of instructions that tells a computer what to do is known as **program**. A computer works according to given instructions in a program. Computer programs are written in programming languages. A person who develops a program is known as **programmer**.

Software:

A set of programs that enables a computer to perform special tasks is known as **software**. It cannot be touched by us. Software is not executed without hardware. It can be reinstalled. It is affected by computer viruses.

Eg: Microsoft Office, Visual studio, Microsoft Visio etc.

Hardware:

Physical parts of a computer are called **hardware**. It can be touched by us. Hardware cannot perform any task without software. It can be replaced. It is not affected by computer viruses.

Eg: Monitor, Speakers, and CPU etc.

Classes of Computers:

There are 3 main classes of computers

1. PC's:

   A PC's is relatively small computer and is designed to be used by one person at a time. Most home computers are PC's but PC's are widely used in businesses, Industries & Science.

2. Workstations:

   Workstations are relatively larger and more powerful computer than PC's. You may think of it as an Industrial strength PC's.

3. Mainframes:

   Mainframes are widely larger computer that typically requires some supporting staff and is shared by more than one user at a time.

Structure of Computer:

Network:

Network consists of numbers of computers connected to each other so that they may share resources such as printers and may share information.

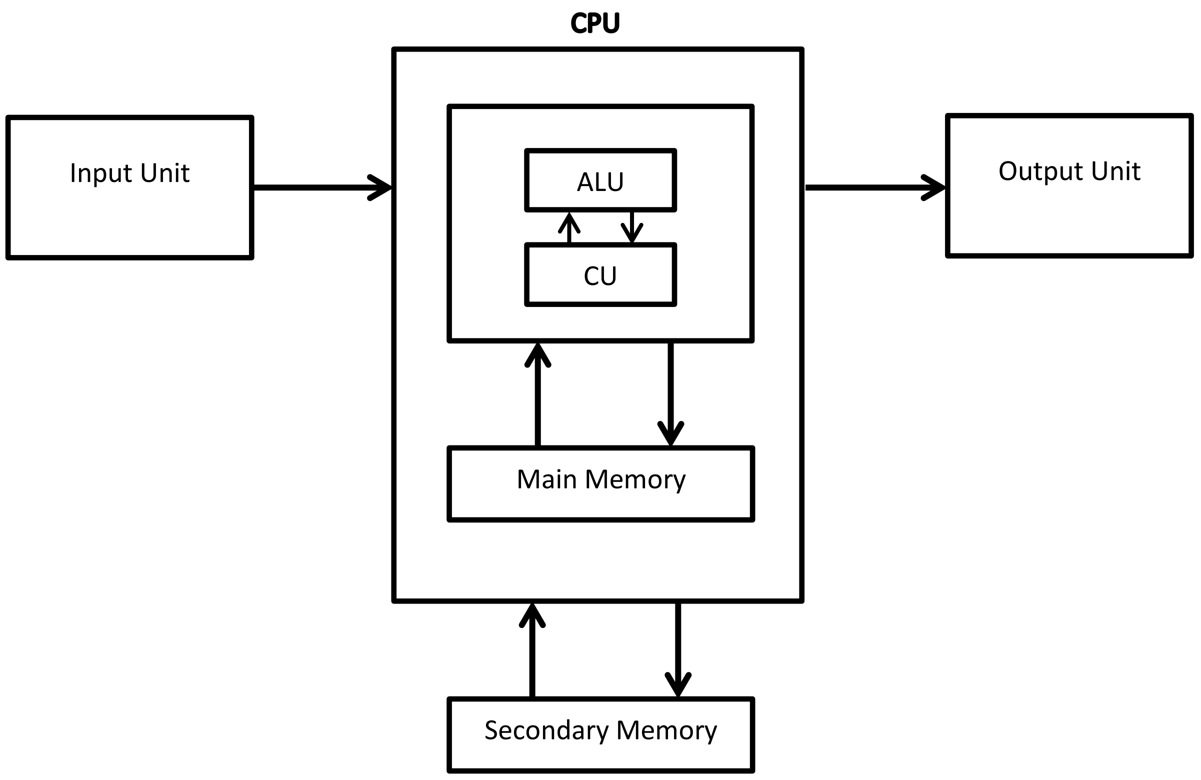
Computer Architecture:

The way in which various components of computer are connected to each other is known as computer architecture.

Design:

In 1951, Von-Neumann and his team proposed the design of store program computer, upon which modern general purpose computers are based. The key elements of Von- Neumann are: data & instructions both are stored in main memory.

Components of Computer Architecture:



1) CPU:

It is the most important part of a computer and it is also known as brain of the computer.

It is also known as Processor or Microprocessor. It is small chip fixed on the motherboard. It process or manipulates data according to given instructions and converts them into useful information.

Component of CPU:

CPU has two components:

i. ***Arithmetic & Logic Unit (ALU):***

- **Arithmetic Unit (AU):** Perform all arithmetic operations such as **+, - , / , *, %.**
- **Logic Unit (LU):** Perform all logical operators such as a>b, 3<4**.**

ii. ***Control Unit (CU):***

- It controls all activities of computer.
- It fetches instructions and data from the main memory.
- Decode instructions
- Execute instructions

2) Main Memory:

Main Memory is called RAM (Random Access Memory). It is a small chip fixed on the motherboard. It is used to store data and instructions. The data and instructions to be processed are first loaded into the main memory and then they are processed by the CPU. After processing, the results are also stored on the main memory. The main memory is the working area of the computer. A computer cannot work without main memory.

3) Secondary Memory:

Secondary Memory are normally housed in a single cabinet.

4) Input/output Unit:

The I/O unit is used to control different Input/output operations and Input/output devices such as keyboard, mouse, printers etc. It controls the communication between the processor and peripheral devices.

Difference between Algorithm & Flowchart

Algorithm	Flowchart
• The step by step procedure of program is called Algorithm	• The graphical representation of any algorithm is called flowchart
• It is easy to modify	• It is difficult to modify
• It is less time consuming	• It is more time consuming
• Proper steps are used in algorithm	• Proper shapes are used in flowchart

Difference between High Level Language & Low Level Language

High Level Language	Low Level Language
• High level languages are close to human languages.	• Low level languages are close to computer hardware but far from human languages
• High level languages are written in English like words such as Print.	• Low level Languages are written in binary codes or symbols
• It is easy to understand.	• It is difficult to understand
• It is easy to modify.	• It is difficult to modify.
• It takes too much time in execution	• It takes less time in execution
• Deep knowledge of hardware is not required to write a program	• Deep knowledge of hardware is required to write a program

Difference between Source Code & Object Code

Source Code	Object Code
• Source code is written in high level languages or assembly languages	• Object code is written in machine language.
• It is easy to understand	• It is difficult to understand
• It is easy to modify	• It is difficult to modify

Difference between Compiler & Interpreter

Compiler	Interpreter
• Compiler convert a computer program into machine code as a whole	• Interpreter convert a program into machine code statement by statement
• It creates the object code	• It does not create the object code
• It is fast in execution	• It is slow in execution
• It displays syntax error as a whole	• It displays syntax error on every statement

Identifiers:

The **identifiers** are the names used to represent variable, constants, types, functions and labels in a program. **Identifier** is an important feature of all computer languages.

An **identifier** in C++ may consist of 31 characters. If the name of identifier is longer than 31 characters, the first 31 will be used. The remaining characters will be ignored by C++ compiler. Some important rules for identifier name are as follows.

- The first character must be an alphabetic or underscore (_).
- The identifier name must consist of only alphabetic character, digits or underscore.
- The reserved word cannot be used as identifier name.

Types of identifiers:

C++ provides the following types of identifiers.

- **Standard Identifiers:**

  A type of identifier that has special meaning in C++ is known as **standard identifier**.

Example:

cout and **cin** are examples of standard identifiers.

- **User-defined Identifiers:**

  A type of identifier that is defined by the programmer to access the memory location is known as **user-defined identifier**.

Example:

Some examples of user-defined identifiers are **a**, **marks** and **age** etc.

Keywords:

Keyword is a word in C++ language that has predefined meaning and purpose,. The meaning and purpose of a keyword is defined by the developer of the language. It cannot be changed or predefined by the user. **Keywords** are also known as **reserved words**. There are different types of keywords in C++ language. The total numbers of keywords is 63.

For Example:

bool, int, float, break etc.

Errors in Programing Languages:

There are 3 types of errors in Programing Languages:

1. **Syntax Error:**

As we know that every programming language has its own rules which are known as syntax. **Syntax error** is a type of error in which programmer does not follow these rules.

Eg: When a programmer uses ‘Cout’ instead of ‘cout’.
If a terminating statement is missing at the end of the line.
Misspelled Keyword

2. **Logical Error:**

Logical error is a type of error in which a programmer writes a poor program. A program will be executed if it has logical error but it will not give the right output.

Eg: If the programmer used ‘&&’ operator instead of ‘| | operator’.
If the programmer used ‘>’ operator instead of ‘< operator’

3. **Run Time Error:**

Run Time error is a type of error that occurs during execution of a program.

Eg: When a user enter the wrong data type.

Data Types:

The data types define a set of values and a set of operations on those values.

Integer Data Types:

Integer data is numeric value with no decimal point or fraction. It includes positive and negative values. The minus sign “-” is used to indicate negative value. If no sign is used, the value is positive by default.

Example:

Some examples of integer values are 10, 520 and -20 etc.

Types:

C++ provides different types of integer data. These are as follows:

1. **int Data Type:**
   int data type is used to store integer values. It takes two or four bytes in memory depending on the computer and compiler being used. Its range is from -32768 to 32768.
2. **short int Data Type:**
   short int data type is used to store integer values. It takes two bytes in memory. Its range is from -32768 to 32768.
3. **unsigned int Data Type:**
   unsigned int data type is used to store only positive integer values. It takes two bytes in memory. Its range is from 0 to 65,535.
4. **long int Data Type:**
   long int data type is used to store large integer values. It takes four bytes in memory. Its range is from -2,147,483,648 to 2,147,483,648.
5. **unsigned long int Data Type:**
   unsigned long int data type is used to store large positive integer values. It takes four bytes in memory. Its range is from 0 to 4,294,967,295.

Real Data Types:

Real data is numeric value with decimal point or fraction. It is also called floating point number. It includes positive and negative values. The minus sign “-” is used to indicate negative value. If no sign is used, the value is positive by default.

Example:

Some examples of integer values are 10.5, 5.3 and -10.91 etc.

Types:

C++ provides different types of real data. These are as follows:

1. **float Data Type:**
   float data type is used to store real values. It takes four bytes in memory. Its range is from 3.4×10^{-38} to $3.4 \times 10^{+38}$.

2. **double Data Type:**
   double data type is used to store real values. It takes eight bytes in memory. Its range is from 1.7×10^{-308} to $1.7 \times 10^{+308}$.

3. **long double Data Type:**
   long double data type is used to store large real values. It takes ten bytes in memory. Its range is from 1.7×10^{-4932} to $1.7 \times 10^{+4932}$.

Character Data Types:

Char is used to store a character value. It takes 1 byte in memory. It is used to represent a letter, number, punctuation mark and few other symbols.

Example:

‘a’ , ‘x’ , ‘#’ , ‘5’ etc.

bool Data Types:

The bool data type takes 1 byte and store a value of true (1) and false (0).

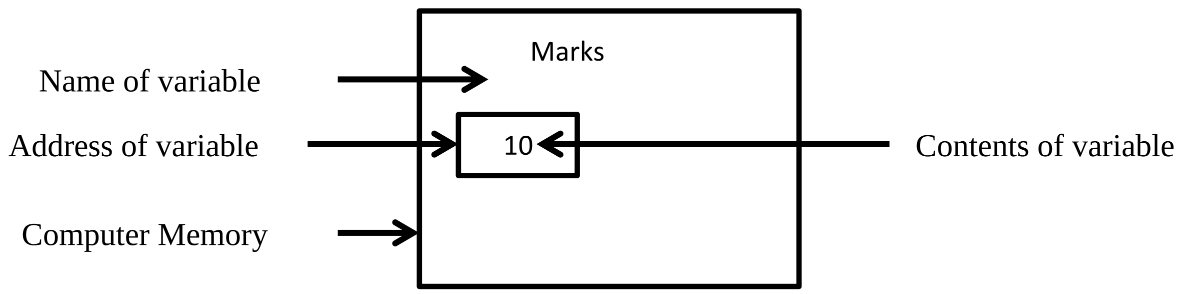
Example:

bool cond = “false”

Variables:

A variable is a named memory location and memory cell. It is used to store program's input data and its computational results during execution. The value of a variable may change during execution of program. However, the name of variable cannot be changed.

The variables are created in RAM. RAM is a temporary memory. That is why the data stored in variables is also temporary. It can only be used and processed during the execution of program,. The data stored in variable is automatically removed when program ends.



In the above figure:

Name of variable: It refers to an identifier that represents a memory location.
Address of variable: It refers to the memory location of the variable.
Contents of variable: It refers to the value stored in memory location referred by variables.

Variables Declaration:

The process of specifying variable names and its types is called variable declaration.

Syntax:

The Syntax of declaring variable is as follows:

```
data_type variable_name;
```

data_type: It indicates type of data that can be stored in variable.
Variable_name: It refers to the memory location of the variable.

Example:

Different types of variables are declared as follows:

```
int marks;
float average;
char grade;
double salary;
```

Rules for Declaring Variables:

Following are some rules for naming variables in C++ language.

- Variable may include letters, numbers and underscore (_).
- The first character of variable must be letter or underscore.
- Blank Spaces are not allowed in variable names.
- Both upper and lower cases are allowed.
- Special symbols cannot be used as variable name.
- Reserved word cannot be used as a variable name.
- A variable can be up to 31 characters.

Arrays:

An array is a collection of different adjacent memory locations. All these memory location have one collective name and type. The memory locations in the array are also known as elements of array. The total number of elements an in array is known as length of array.

Each element in the array is accessed with reference to its position of location in the array. This position is called index or subscript. The index of first element is 0 and the index of last element is "length-1". The value of index is written in brackets along with name of array.

Declaring One Dimension Array:

A type of array in which all elements are arranged in the form of list is known as one-dimension array. It is also called single-dimensional array or linear list. It consists of one column and one row.

The process of specifying array name, length and data type is called **Array Declaration**.

Syntax:

The Syntax of declaring one-dimension array is as follows:

Data_Type identifier [Length];

Data_Type: It indicates the data types of the values to be stored in the array.
Identifiers: It indicates the name of the array.
Length: It indicates the total number of elements in the array.

Array Initialization:

The process of assigning values to array elements at the time of array declaration is called **Array Initialization**. The values are separated with commas and enclosed within braces.

Syntax:

The Syntax to initialize array is as follows:

Data_Type identifier [Length] = {List of values};

Data_Type: It indicates the data types of the values to be stored in the array.
Identifiers: It indicates the name of the array.
Length: It indicates the total number of elements in the array.
List of values: It indicates the values to initialize the array.

Declaring Two-Dimensional Array:

Two-dimensional array represents a collection of same type of data that is in the form of table or matrix. It has two dimensions that are vertical and horizontal dimension. The vertical dimension represents the rows and the horizontal dimension represents the columns.

Each element in 2-D array is referred with the help of two indexes. One index is used to indicate the row and second index indicates the column of the element.

Syntax:

The Syntax for declaring 2-D array is as follows:

Data_Type identifier [Rows] [Cols];

Data_Type: It indicates the data types of the values to be stored in the array.
Identifiers: It indicates the name of the array.
Rows: It indicates the number of rows in the array.
Columns: It indicates the number of columns in the array.

Array Initialization:

The process of assigning values to array elements at the time of array declaration is called **Array Initialization**.

Some important points to initialize two-dimensional arrays are as follows:

- The elements of each row are enclosed within braces and separated by commas.
- All rows are enclosed within the braces.

Functions:

A group of statements that together perform a task is known as function. Each function must have a unique name. The statements written in the function will be executed when it is called by its name. Every program in C++ has at least one function known as **main ( )**.

Types of Functions:

There are two types of functions:

- User Defined Function
- Built in Function

User Defined Function:

A type of function written by a programmer is known as **user defined function**. Every user-defined function has a unique name. A program may contain many user-defined functions.

Built in Function:

A type of function that is available as a part of language is known as **built-in function/ library function**. These functions are ready-made programs.

Call by Value	Call by Reference
• Call by value passes the value of actual parameter to formal parameter.	• Call by value passes the address of actual parameter to formal parameter.
• The actual and formal parameters refer to different memory locations.	• The actual and formal parameters refer to same memory location.
• Any change made by function in formal parameter does not affect the value of actual parameter.	• Any change made by function in formal parameter actually changes the value of actual parameter.
• It requires more memory	• It requires less memory
• It is less efficient	• It is more efficient

Components of Function:

Each function in C++ consists of the following components:

1. Function Prototype/ Function Declaration:

It provides information to compiler about the structure of the function to be used in a program. It ends with a semicolon. It consists of the following parts:

- Function name
- Function Return type
- Numbers and types of parameters

Syntax:

The syntax of function declaration is as follow:

Return-type Function-name (parameters);

Return-type: It indicates the type of value that will be returned by function.
Function-name: It indicates the name of the function.
Parameters: Parameters are the values that are provide to a function when the function is called.

2. Function Definition:

A set of statements that explains what a function does is called function definition. It can be written at the following places:

- Before main ( ) function
- After main ( ) function
- In a separate file

The function definition consists of two parts:

- **Function Header:**
  The first line of function definition is known as function header. It is also known as **function declarator**. It is similar to the function prototype. The only difference is that it is not terminated with semicolon.

- **Function Body:**
  The set of statements which are executed inside the function is known as function body. These statements are written in curly braces { }.

3. Function Call:

Function call is the statement that invokes a function. A function is executed only when it is called by its name.

Parameters:

Parameters are the values that are provided to a function when the function is called. The parameters are written in the parenthesis. Multiple parameters are separated by commas. The empty parameters are used if there is no parameter..

Types:

Two types of parameters are as follows:

1. **Formal parameter:**
   The parameter used in header of a function definition is called formal parameter. These parameters are used to receive values from the calling function.
2. **Actual parameter:**
   The parameter used in function call is called actual parameter. These are the actual values that are passed to the function..