

**Computer :** A computer is a machine that can be programmed to carry out sequences of arithmetic or logical operations (computation) automatically.

**Or**

**Computer is an electronic device that takes input from users process it and produces the output.**

**Use of C Language?**

- Development of video games.
- Applications using graphical user interfaces.
- Databases and computer operating systems.
- Browsers on the internet.
- To develop Compilers.

**Advantages of computer: 1)speed 2)accuracy  
3)storage 4)versatile**

A computer system is divided into two categories:  
Hardware and Software.

Hardware refers to the physical and visible components of the system such as a monitor, CPU, keyboard and mouse.

Software: Software is a collection of programs.  
Software refers to a set of instructions which enable the hardware to perform a specific set of tasks.

Example:windows 10 , Linux,  
Ms-word

Software are of two types :system and application software

## operating system (OS)

An **operating system (OS)** is [system software](#) that manages [computer hardware](#) and [software](#) resources, and provides common [services](#) for [computer programs](#).

or

An **Operating System (OS)** is an interface between a computer user and computer hardware.

Functions of O-S:

**Device Management** is another important function of the operating system. Device management is responsible for managing all the hardware devices of the computer system. It may also include the management of the storage device as well as the management of all the input and output devices of the computer system.

**Security management** :The security management function of an operating system helps in implementing mechanisms that secure and protect the computer system internally as well as externally. Therefore an operating system is responsible for securing the system at two different levels which are internal security and external security.External security is related with network.

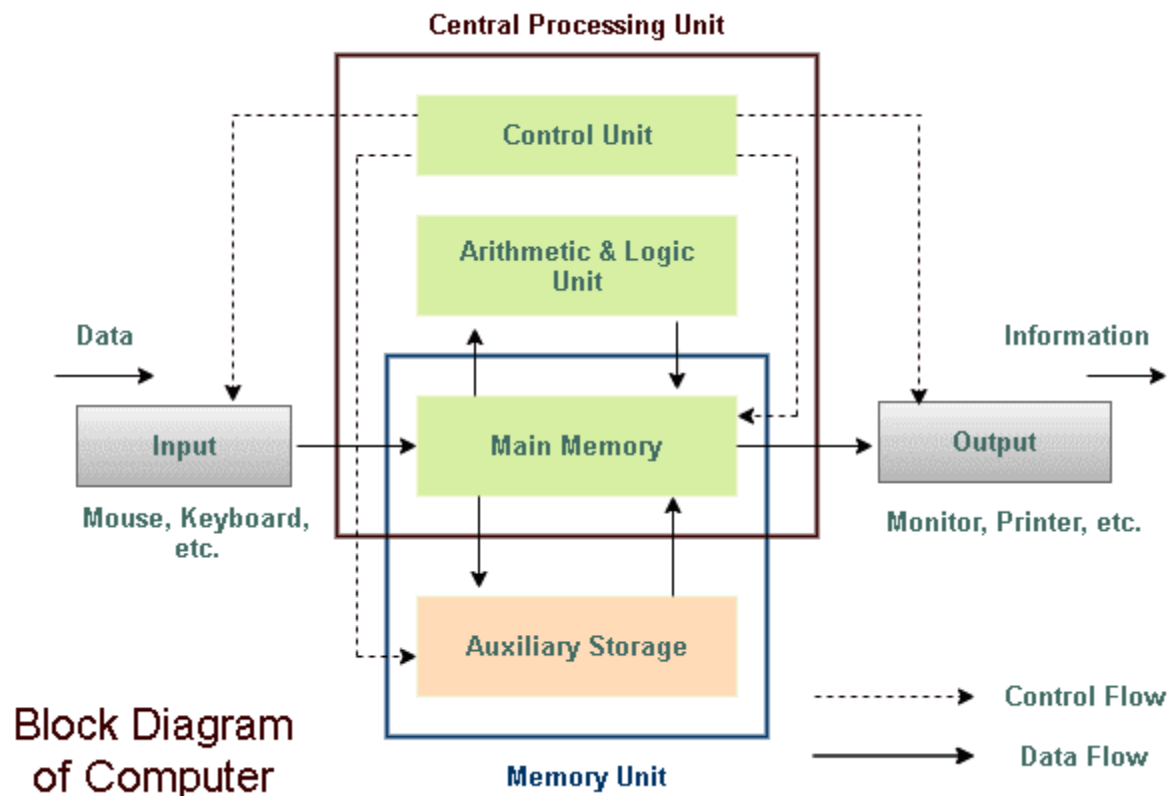
For example:it provides userid and password for login.

**File management:** File management in operating system is nothing but software that manages or handles the files ( video, audio, docs, pdf, text, etc.) present in computer software. The operating system's file system can manage individuals and groups of files present in the computer system.<sup>28</sup>

Memory management is **the process of controlling and coordinating a computer's main memory**. It ensures that blocks of memory space are properly managed and allocated so the operating system (OS), applications and other running processes have the memory they need to carry out their operations.

Digital Computer: **Digital Computer** is a machine or a device that helps to process any kind of information.

Digital computer operates on two binary digits 0 and 1. 0 for false and 1 for true. These are called binary digits.



## Input Unit

The Input Unit consists of input devices such as a mouse, keyboard, scanner, joystick, etc. These devices are used to input information or instruction into the computer system. Like other electronic machines, a computer takes inputs as raw data (binary data) and performs necessary processing giving out processed data. Therefore, the input unit is the medium of communication that takes data from us to the computer in an organized manner for processing.

The Input Unit performs the following major functions:

- The input unit converts the inputted data or instructions into binary form for further processing.
- Input Unit transmits the data to the main memory of the computer.

## Output Unit

The output unit consists of devices that are used to display the results or output of processing. The output data is first stored in the memory and then displayed in human-readable form through output devices. Some of the widely used output devices are Monitor, Printer, and Projector.

The OUTPUT Unit performs the following major functions:

The output unit accepts the data or information in binary form from the main memory of the computer system.

- The output unit converts the binary data into a human-readable form for better understanding.

## Central Processing Unit

CPU or Central Processing Unit is known as the brain of the computer system. It is an electronic hardware device that processes all the operations (e.g., arithmetic and logical operations) of the computer. In other words, all the major calculations, operations or comparisons are performed inside the CPU. It is also responsible for handling the operations of several other units.

CPU is divided into three parts:

- 1)ALU
- 2)CU
- 3)Memory Registers

## Control Unit

As the name suggests, the control unit of a CPU controls all the activities and operations of the computer. It is also responsible for controlling input/output, memory, and other devices connected to the CPU.

The control unit acts like the supervisor which determines the sequence in which computer programs and instructions are executed. It retrieves instructions from memory, decodes the instructions, interprets the instructions and understands the sequence of tasks to be performed

accordingly. It further transmits the instructions to the other parts of the computer system to execute them. In short, the control unit determines the sequence of operations to execute the given instructions.

## Arithmetic & Logic Unit

The data inputted through input devices is stored in the primary storage unit. The Arithmetic Logic Unit (ALU) performs arithmetic and logical operations.

The arithmetic unit controls simple operations such as **addition, subtraction, division, and multiplication**.

## Memory Unit

Memory Unit is an essential part of the computer system which is used to store data and instructions before and after processing. The memory unit transmits the information to other units of the computer system when required.

The memory unit consists of various registers like accumulator, memory buffer register etc.

# Algorithmn and Flowchart

An algorithm is a finite set of instructions carried out in a specific order to perform a particular task.

Algorithn characteristic:

- Input: An algorithm requires some input values. An algorithm can be given a value other than 0 as input.
- Output: At the end of an algorithm, you will have one or more outcomes.
- Unambiguity: A perfect algorithm is defined as unambiguous, which means that its instructions should be clear and straightforward.
- Finiteness: An algorithm must be finite. Finiteness in this context means that the algorithm should have a limited number of instructions, i.e., the instructions should be countable.
- Effectiveness: Because each instruction in an algorithm affects the overall process, it should be adequate.

Algorithn to add two numbers:

- 1) Start.
- 2) Accept Number one and Number two.
- 4) Add both the numbers.
- 5) Print the result.
- 6) End

## Algorithm to swap two numbers:

```
Declare a variable a,b and c as integer;  
Read two numbers a and b;  
c=a;  
a=b;  
b=c;  
Print a and b
```

Algorithm to find area of circle:

STEP 1: Take radius as input from the user using std input.






STEP 2: Calculate the area of circle using,  $\text{area} = (3.14) * r * r$

STEP 3: Print the area to the screen using the std output.<sup>19</sup>

Algorithm to find greatest of 3 numbers

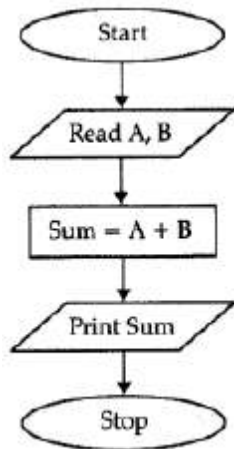
1. Ask the user to enter three integer values.
2. Read the three integer values in num1, num2, and num3 (integer variables).
3. Check if num1 is greater than num2.
4. If true, then check if num1 is greater than num3.
  1. If true, then print 'num1' as the greatest number.
  2. If false, then print 'num3' as the greatest number.
5. If false, then check if num2 is greater than num3.
  1. If true, then print 'num2' as the greatest number.
  2. If false, then print 'num3' as the greatest number.

**Flowchart:** A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm,

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectagle represents a process
	Decision	A diamond indicates a decision

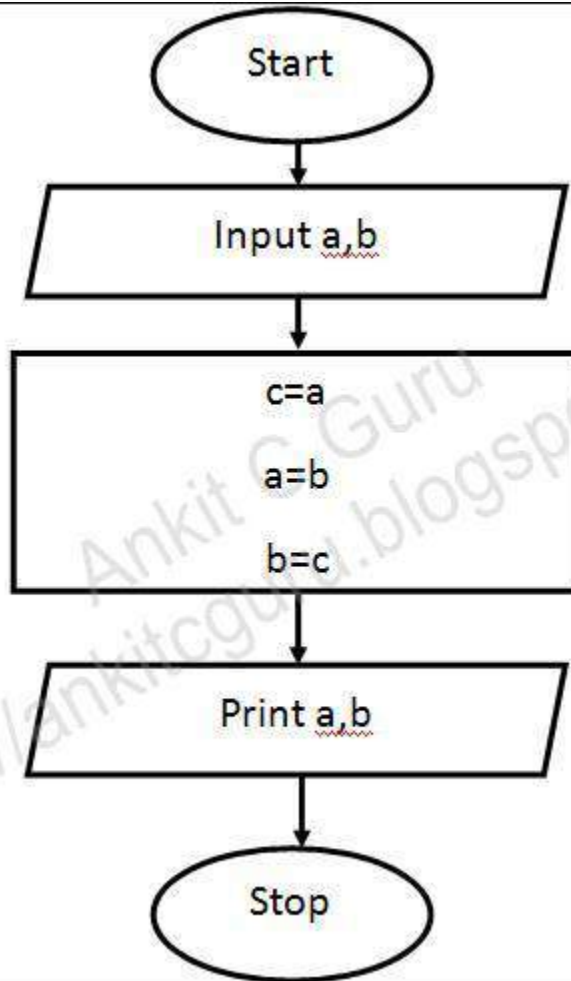
Flowchart to add two numbers



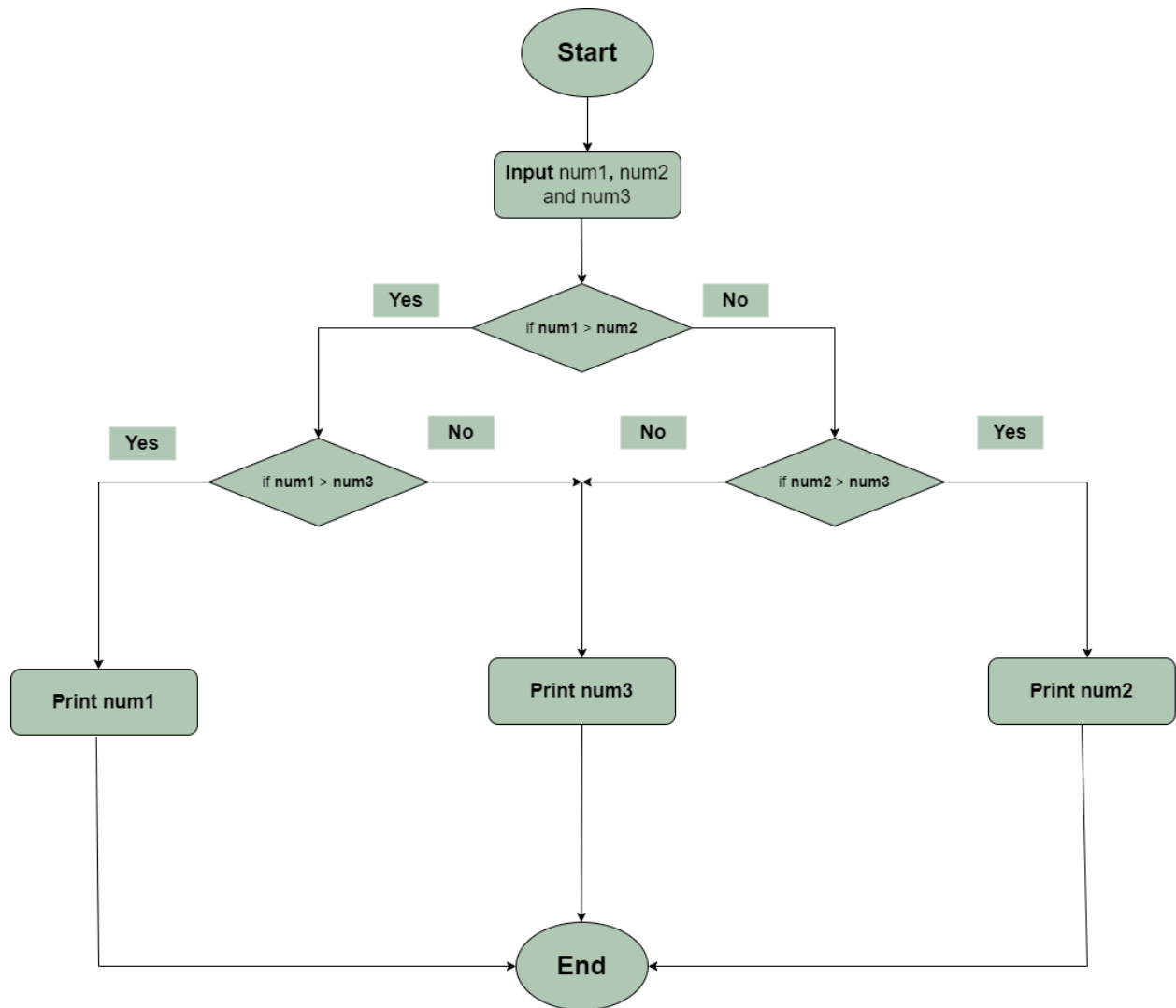


Flowchart to swap two numbers:

---



## Flowchart to find greatest of three numbers

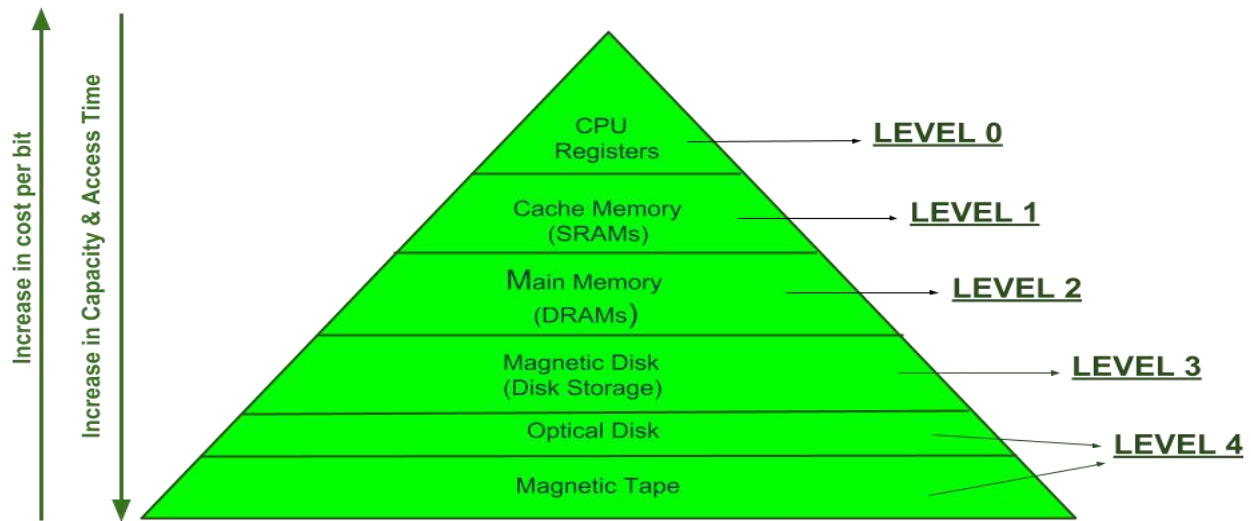


### Header file in C:

A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files. There are two types of header files: the files that the programmer writes and the files that comes with your compiler.

**Example: `stdio.h`, `conio.h`, `math.h`**

Memory is used for data storage.



### MEMORY HIERARCHY DESIGN

Hierarchy Design is divided into 2 main types:

1. **External Memory or Secondary Memory** – Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
2. **Internal Memory or Primary Memory** – Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

<p>RAM data is volatile, which means if the system crashes or some interrupt occurs, the RAM data is lost.</p> <p>RAM is a read-write memory which means the data stored on RAM can be modified.</p> <p>RAM is faster as compared to ROM because the CPU access data directly from it.</p>	<p>ROM data is not volatile, which means the data stored in ram is permanent.</p> <p>ROM is a read-only memory which means data stored on ROM cannot be modified.</p> <p>ROM is the slower device as compared to RAM because the CPU access data indirectly.</p>
--	--

**Cache Memory:**Cache memory is a high speed memory which is placed in between CPU and main memory for faster processing.

**Register Memory:**This memory is used for temporary data storage. For Eg: add 2 numbers data is stored in registers.

## Language Translators:

Language translators allow computer programmers to write sets of instructions in specific programming languages. These instructions are converted by the language translator into machine code. The computer system then reads these machine code instructions and executes them.

### Assembler

An assembler translates assembly language into machine code. Assembly language consists of mnemonics code. For example, add a,b

**Compiler:** A compiler is a computer program that translates code written in a high-level language into a low-level language, machine code. It scans entire code at once and displays errors.

### Interpreter

An interpreter program executes other programs directly, running through the program code and executing it line-by-line. As it analyses every line, an interpreter is slower than running compiled code

### Interpreter

Translates program one statement at a time.

The overall execution time is comparatively slower than compilers.

No Object Code is generated, hence are memory efficient.

Programming languages like JavaScript, Python, Ruby use interpreters.

### Compiler

Scans the entire program and translates it as a whole into machine code.

The overall execution time is comparatively faster than interpreters.

Generates Object Code which further requires linking, hence requires more memory.

Programming languages like C, C++, Java use compilers.

## Errors and its types

**Errors** are the problems or the faults that occur in the program. There are commonly five types of errors exist in C programming:


1. Syntax error
2. Run-time error
3. Linker error
4. Logical error
5. Semantic error

### **Syntax error**

Syntax errors are also known as the compilation errors as they occurred at the compilation time. These errors are mainly occurred due to the mistakes while typing when programmer does not follow the grammar rule of the programming language. These errors can be easily corrected.

Commonly occurred syntax errors are: using variable without its declaration, missing the semicolon (;) at the end of the statement.

```
...  
int a = 6, b = 7;  
c = a + b;  
printf("%d", c);  
...
```

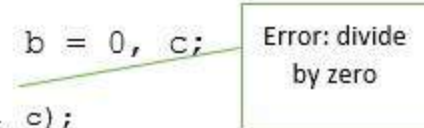


Error: c is undefined

### **Run-time error / Execution Error**

The errors exist during the execution-time even after the successful compilation known as run-time errors. When the program is running, and it is not able to perform the operation is the main cause of the run-time error. The division by zero is the common example of the run-time error. These errors are very difficult to find, as the compiler does not point out to these errors.

```
...  
int a = 6, b = 0, c;  
c = a / b;  
printf("%d", c);  
...
```



Error: divide by zero

## Linker error

Linker errors are mainly generated when the executable file of the program is not created. This can be happened either due to the wrong function prototyping or usage of the wrong header file.

```
#include<studio.h>
#include<conio.h>
void main()
{
    printf("enter first number:");
    .....
}
```

Error: No such  
header file

## Logical error

The logical error is an error that leads to an undesired output. These errors produce the incorrect output, but they are error-free, known as logical errors. The occurrence of these errors mainly depends upon the logical thinking of the developer.

```
...
int a = 6, b = 7, avg;
c = a + b / 2;
printf("%d", c);
...
```

Error: bracket is  
not used in  
formula

## Semantic error

Semantic errors are the errors that occurred when the statements are not understandable by the compiler. The following can be the cases for the semantic error:

```
...
int a = 6, b = 7, c;
a + b = c;
printf("%d", c);
...
```

Error:  
expression  
wrong

## Flow of C program:

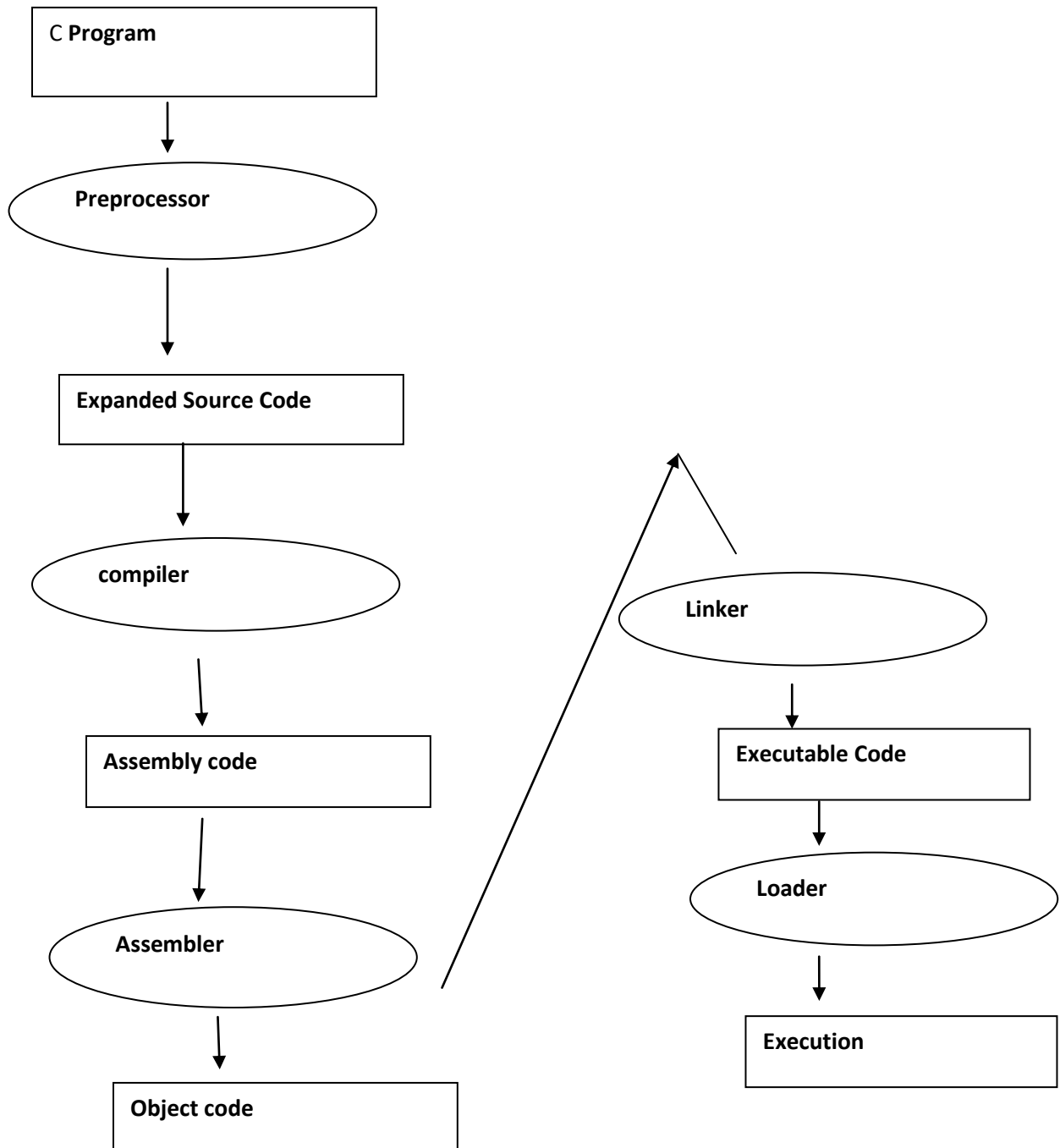
1) C program (source code) is sent to preprocessor first. The preprocessor is responsible to convert preprocessor directives into their respective values. The preprocessor generates an expanded source code.

2) Expanded source code is sent to compiler which compiles the code and converts it into assembly code.

3) The assembly code is sent to assembler which assembles the code and converts it into object code. Now a simple.obj file is generated.

4) The object code is sent to linker which links it to the library such as header files. Then it is converted into executable code. A simple.exe file is generated.

5) The executable code is sent to loader which loads it into memory and then it is executed. After execution, output is sent to console.





## Difference Between Object Code and Executable Code

**Object code** is a sequence of statements in binary that is generated after compiling the source program. In contrast, an **executable code** is a file or a program that indicates tasks according to encoded instructions which are directly executed by the CPU.

### Linker

A linker is a useful utility tool that combines object files and other code created by the assembler and compiler into a single executable file.

The primary role of a linker is to create executable files.

Linking is the process of putting together multiple parts of code and source code to create executable code.

It takes the object code created by an assembler as input.

### Loader

A loader is an operating system component responsible for loading applications and libraries.

The primary goal of a loader is to load executable files into the main memory.

The process of loading executable programs into the main memory for subsequent execution is known as loading.

It accepts linker-generated executable modules.

Source code is generally understood to mean programming statements that are created by a programmer with a text editor or a visual programming tool and then saved in a file.

**Pseudocode** is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool. The rules of Pseudocode are reasonably straightforward. All statements showing "dependency" are to be indented. Pseudocode to add two numbers:

1. BEGIN.
2. NUMBER s1, s2, sum.
3. OUTPUT("Input number1:")
4. INPUT s1.
5. OUTPUT("Input number2:")
6. INPUT s2.
7. sum=s1+s2.
8. OUTPUT sum.

**Data Types in c:** Data types refers to type of information represented by a variable in memory.

For eg: `int a; // a is interger type variable`

Following are the primitive data types in c:

Keyword	Data type	Size(in bytes)/Range	Format specifier
Int	Integer	2(-32768 to +32768)	%d
Float	Floating Point or decimal	4(3.4E-38 to 3.4E+38)	%f
Double	Floating Point or decimal(large size)	8 (1.7E-308 to 1.7E+308)	%lf
Char	Character	1(-127 to +128)	%c

**Components of C Language:**

**1)Tokens:** Tokens are the smallest unit of a program. Each and every punctuation and word that you come across in a C program is token.

**2)Keywords:** A keyword is a reserved word that have special meaning to the C compiler. .You cannot use it as a variable name, constant name, etc. There are only 32 reserved words (keywords) in the C language. Example: `int ,float, switch`

**3)Identifiers:** Identifiers" are the names we supply for variables, arrays, functions, and labels in our program. Identifier names must differ from any keywords. **Example:** `int a[10];`

**4)Variabes:** Variable is a location in primary memory that is referred by name and is used for storing information.

Eg: `int a ; // here a is a variable`

## Structure of a C program

Most programming languages have a structure, including the C language. A C program is divided into six sections:

- 1) Documentation
- 2) Link
- 3) Definition
- 4) Global Declaration
- 5) Main() Function
- and 6) Subprograms.

While the main section is compulsory, the rest are optional in the structure of the C program.

Section	Description
Documentation	Consists of the description of the program, programmer's name, and creation date. These are generally written in the form of comments.
Link	All header files are included in this section which contains different functions from the libraries. A copy of these header files is inserted into your code before compilation.
Definition	Includes preprocessor directive, which contains symbolic constants. E.g.: #define allows us to use constants in our code. It replaces all the constants with its value in the code.
Global Declaration	Includes declaration of global variables, function declarations, static global variables, and functions.
Main() Function	For every C program, the execution starts from the main() function. It is mandatory to include a main() function in every C program.
Subprograms	Includes all user-defined functions (functions the user provides). They can contain the inbuilt functions and the function definitions declared in the Global Declaration section. These are called in the main() function.

**Comments:** A comment is basically a text note that gives an explanation about the source code. Furthermore, they act as documentation in the source code. We include comments to increase the readability of the program.

# Types of Comments

There are two ways to add comments in C:

1. `//` - Single Line Comment
2. `/*...*/` - Multi-line Comment

Example:

```
/**          //Documentation
 * file: age.c
 * author: Rajesh Kumar
 * description: program to find our age.
 */
```

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

```
#define BORN 2000    //Definition
```

```
int p=10; //Global Declaration
```

```
void main(void)      //Main() Function
{
    int current = 2021;
    printf("Age: %d", current);
}
```

```
int age(int current) //Subprograms
```

```
{    return current - BORN;
}
```

## Standard I/O in C:

C language has standard libraries that allow input and output in a program. The **stdio.h** or **standard input output library** in C that has methods for input and output.

### scanf()

The scanf() method, in C, reads the value from the console as per the type specified. **Syntax:**

**scanf("%X", &variableOfXType);** where %X is the [format specifier in C](#). It is a way to tell the compiler what type of data is in a variable and & is the address operator in C, which tells the compiler to change the real value of this variable, stored at this address in the memory.

### printf()

The printf() method, in C, prints the value passed as the parameter to it, on the console screen. **Syntax:**

**printf("%X", variableOfXType);** where %X is the [format specifier in C](#). It is a way to tell the compiler what type of data is in a variable and & is the address operator in C, which tells the compiler to change the real value of this variable, stored at this address in the memory.

***Simple C Program two add two numbers:***

```
#include<stdio.h>
#include<cono.h>
void main()
{
    Int a =10,b=20;
    clrscr();
    c=a+b;
    printf("addition is %d",c);
}
```

***Simple C Program two add two numbers using user input:***

```
#include<stdio.h>
#include<cono.h>
void main()
{
    int a ,b,c;
    clrscr();
    printf("enter two numbers");
    scanf("%d%d",&a,&b);
    c=a+b;
    printf("addition is %d",c);
    getch();
}
```

***Simple C Program two find average of 3 numbers using user input:***

```
#include<stdio.h>
#include<cono.h>
void main()
{
    int a ,b,c,d;
    clrscr();
    printf("enter three numbers");
    scanf("%d%d%d",&a,&b,&c);
    d=(a+b+c)/3;
    printf("addition is %d",c);
    getch();
}
```



## System and Application software differences

**Software:** It is defined as a collection of programs. Programs are a set of instructions. Eg. ZOOM, MS-Word

<b>Definition</b>	<b>System Software</b> is the type of software which is the interface between application software and system.	<b>Application Software</b> is the type of software which runs as per user request..
<b>Necessity</b>	<b>System software</b> are essential for operating the computer hardware.	<b>Application software</b> are not essential for the operation of the computer.
<b>Examples</b>	Examples of system software include operating systems, compilers, assemblers, debuggers, drivers, etc.	Examples of application software include word processors, web browsers, media players, etc.

## High Level vs Low Level Language

<b>Parameter</b>	<b>High-Level Language</b>	<b>Low-Level Language</b>
<b>Basic</b>	These are programmer-friendly languages that are easy to understand,	These are machine-friendly languages that are very difficult to understand by human beings
<b>Ease of Execution</b>	These are very easy to execute.	These are very difficult to execute.
<b>Debugging</b>	It is very easy to debug(find errors) these languages.	A programmer cannot easily debug(find errors) in these languages.

**Python ,C,C++ are high level languages.**



**Operating System:** It is defined as interface between user and hardware.  
Or

**An operating system (OS) is system software that manages computer hardware and software resources.**

**Example: Windows Operating System, Android OS**

**Types of Operating System:**

### **Batch Operating System**

Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs are batched together and run as a group.

### **Multi-Tasking/Time-sharing Operating systems**

Time-sharing operating system enables people located at a different terminal(shell) to use a single computer system at the same time. The processor time (CPU) which is shared among multiple users is termed as time sharing.

### **Real time OS**

A real time operating system time interval to process and respond to inputs is very small. Examples: Military Software Systems, Space Software Systems are the Real time OS example.

### **Distributed Operating System**

Distributed systems use many processors located in different machines to provide very fast computation to its users.

### **Network Operating System**

Network Operating System runs on a server. It provides the capability to serve to manage data, user, groups, security, application, and other networking functions.

### **Mobile OS**

Mobile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices.

Algorithm and Flowchart questions:

1. Write algo and flowchart to add two numbers.
2. Write algo and flowchart to find area of circle.
3. Write algo and flowchart to find simple interest.
4. Write algo and flowchart to find greatest of three numbers.
- 5. Write algo and flowchart to swap two numbers using third variable.**
- 6. Write algo and flowchart to swap two numbers using third variable.**
- 7. Write algo and flowchart to calculate area of triangle.**