Subject Code → CSIT 308

SYLLABUS

Programming For Problem Solving

Module - 1 :

(a) Introduction to Components of a Computer System : Memory, Processor, I/O devices, Storage, Operating System, Concept of assembler, Compiler, interpreter & loader, linker.

(b) Idea of Algorithm : Representation of algorithm, Flow Chart, Pseudocode with examples, from algorithm to programs, Source Code.

(c) Structure of C Programs : Writting & executing the I C Programs, Syntax & logical errors in Compilation, object and executable Code.

(d) Components of C language : Standard I/O in C, fundamentals data types, Variables and Memory locations, Storage classes.

Module - 2 :

(a) Arithmetic expression & Precedence : Operatus and expression using numeric & relational operatus, Mixed operants, type Conversion, logical operatus, Bit operations, assignment operator, operator presidenc & associativity.

b) Conditional Branching : Applying if an else with Statement, Nesting if and else, Use of break & default in Switch Statement.

Module -3 :

a) Interaction and loops : Use of While, Do While & for loops, Multiple loop Variables, Use of Break & Continue Statement.

b) Function : Introduction, Types of functions, function with array, Passing parameters to function, Call by Value, Call by reference, Recursive function.

Module -4 :

a) Arrays : Array notation & representation, Manipulating array elements, Using multidimensional arrays, Character arrays & Strings, Structure, Union, Numerated data types, array Structures, Passing arrays to function.

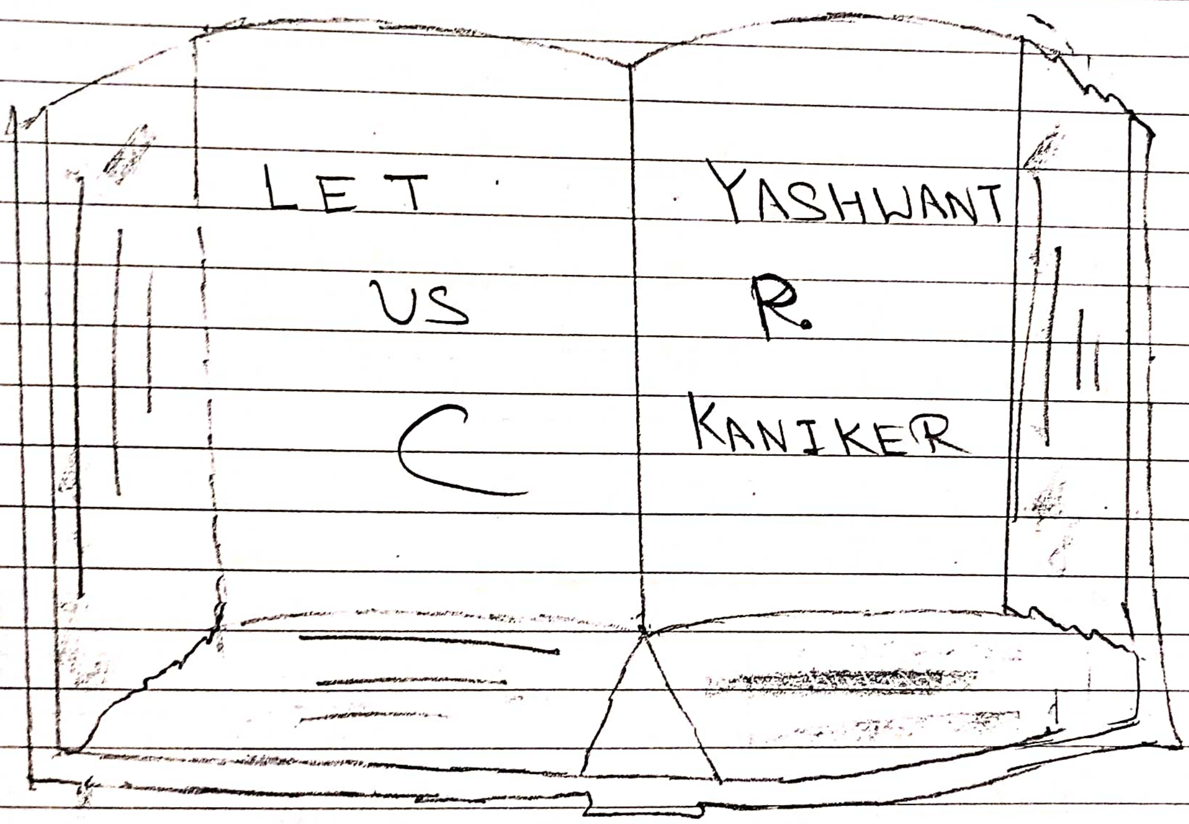
b) Basic Algorithms : Searching & Basics, Sorting algorithm (Bubble, insertion & Selection), finding roots of equation, notion of order of Complexity.

Module - 5 :

a) Pointers : Introduction, Declaration, Application, Introduction to dynamic memory allocation (MALLOC, CALLOC, REALLOC, FREE), Use of pointers in Self-refrential

Structure, Notion of linked list (No implimentation).

(b) File Handling : File I/O function, Standard C Preprocessor, defining & Calling Macros in C, Command line arguments.



Date
22/8/22

Introduction to Components of a Computer System :

Computer :- A Computer is an electronic device which takes data & instruction as input, Stores them, processes them & gives meaningful results as output which Can also be termed as information.

Input → Store → Processing → Store → output

Function of a Computer :

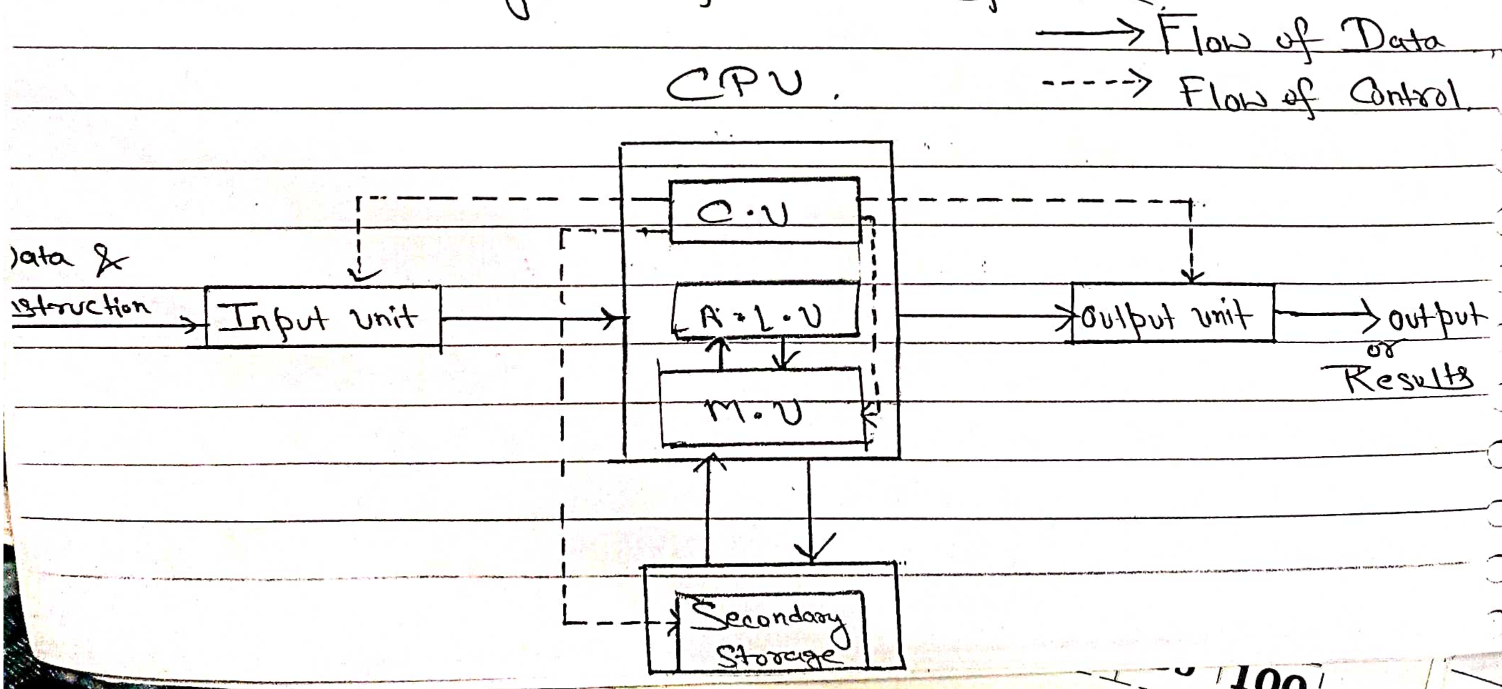
1. Storage
2. Processing
3. Output
4. Input

5. Controlling

Components of a Computer :

(i) Hardware
(ii) Software

Block diagram of a Computer.



Central processing unit (CPU) :- It is Called "the Brain of Computer" as it Controls operation of all parts of Compute
It Consist of three Components : Control unit (CU), Arithmetic logic unit (A.L.U), Memory unit (M.U).

Control unit (CU) :- this part of CPU extracts instructio Performs execution, maintains & directs operations of entire System.
It performs following function :
① Controls all activities of Computer.
② Supervises flow of data within CPU.
③ Directs flow of data within CPU.
④ Transfers data to arithmetic & logic unit.
⑤ Transfers results to memory.
⑥ Fetches results from memory to output devices.

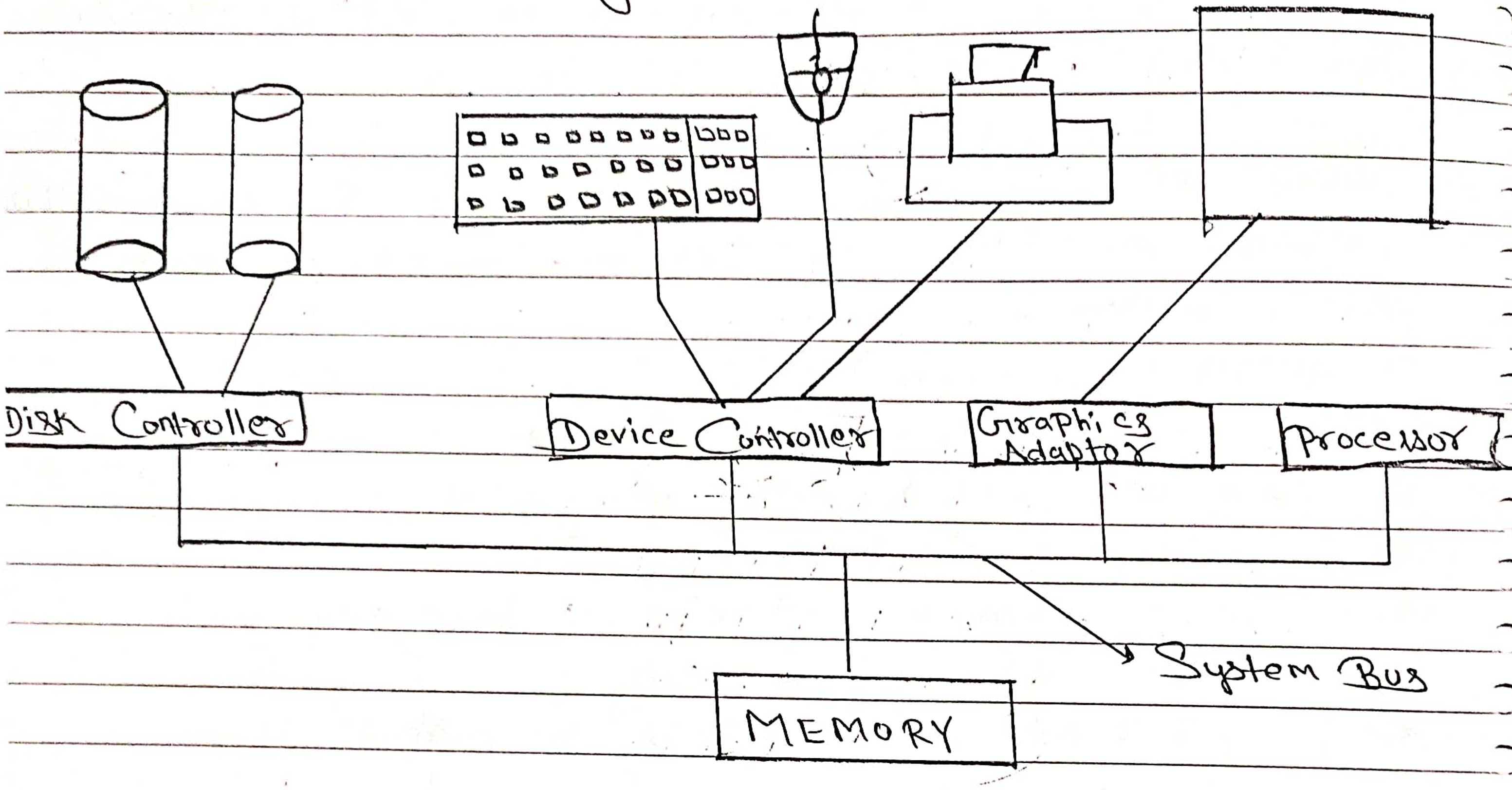
Arithmetic logic unit (ALU) :- Data entered into Computer is sent to RAM, from where it is then sent to ALU, where rest of data processing takes place. All types of processing such as Comparisons, decision making & processing of non-numeric information takes place here & once again data is moved to RAM.

Memory unit (MU) :- This is unit in which data & instructions given to Computer as well as results giver by Computer are stored.
* Unit of memory is "Byte".

1 Byte = 8 Bits

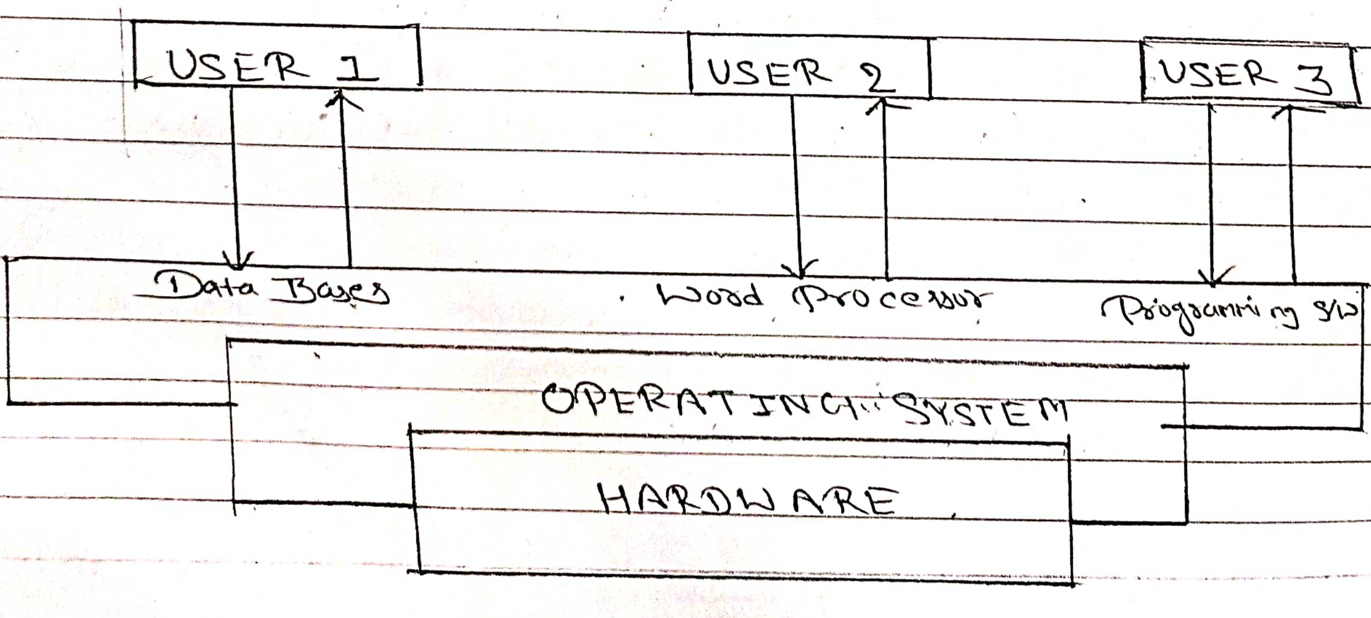
B b

A Modern Day Computer



Components of Computer System :-

1. Hardware
2. Software
3. People / user.
4. Data.



Database → A database is an organized Collection of Structured information, or data typically Stored electronically in a Computer System.

Word processor → A word processor is a type of Software application used for Composing, editing, formatting & Printing documents.

Programming Software → It is a Software which helps the programmer in developing other Software. Compilers, assemblers, debuggers, interpreters etc. are examples of programming Software.

Operating System → operating System is a Software that Controls System's hardware & interacts with user and application Software.

Memory and Storage

Memory : Computer memory is just like a human Brain. It is used to store data/information & instruction. It Can ~~store~~ both input & output can be Stored here.

Characteristics of main memory:

1. It is faster Computer memory as Compare to Secondary memory.
2. It is Semiconductor memories.
3. It is usually a Volatile memory.
4. It is main memory of the Computer.
5. A Computer System Cannot run without primary memory.

$$\begin{tikzcd}
 & & \boxed{\text{RAM}} \arrow[r] & \text{Volatile. / Temporary.} \\
\text{Storage Types} \arrow[r] & \text{① Primary} \arrow[ur] \arrow[dr] & & \\
 & & \boxed{\text{ROM}} \arrow[r] & \text{Non volatile. / Permanent}
\end{tikzcd}$$

$$\begin{tikzcd}
 & \text{Hardisk.} \\
 & \text{Pendrive.} \\
\text{② Secondary memory} \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] & \text{SD Card} \\
 & \text{etc.}
\end{tikzcd}$$

1. Primary memory : It is also Known as main memory of the Computer System. It is used to Store data and programs or instruction during Computer operation. It uses Semiconductor technology & hence is Commonly Called Semiconductor memory.

(a) RAM → It is a Volatile memory, It Stores information based on the power Supply. If the power Supply fails / intrupted / Stopped, all the data & information on this memory will be lost. Ram is used for booting up or Start the Computer. It temporarily Stores program / data.

Two types

① S RAM (Static RAM)

② D RAM (Dynamic RAM)

2. Secondary Memory : It is also Known as Auxiliary memory and backup memory. It is non-Volatile memory & used to Store a large amount of data or information. The data or information Stored in Secondary memory is permanent, & it is Slower than primary memory. A CPU Cannot acess Secondary memory directly. The data or information from the auxiliary memory is first transferred to the main memory & then CPU Can acess it

3. Cache Memory : It is type of high Speed Semiconductor memory that Can help the CPU run faster. Between the CPU and the main memory, it Serves as a buffer. It is used to Store data & programs that the CPU uses the most frequently.

Advantage → ① It is faster than the main memory.
② It takes less time to access as Compare to the main memory.
③ It Keeps the program that Can run in Short amount of time.
④ It Stores data in temporary use.

Disadvantage → ① Because of Semiconductor, It is Very expensive.
② Size is Small.

Volatile & Non-Volatile Memory

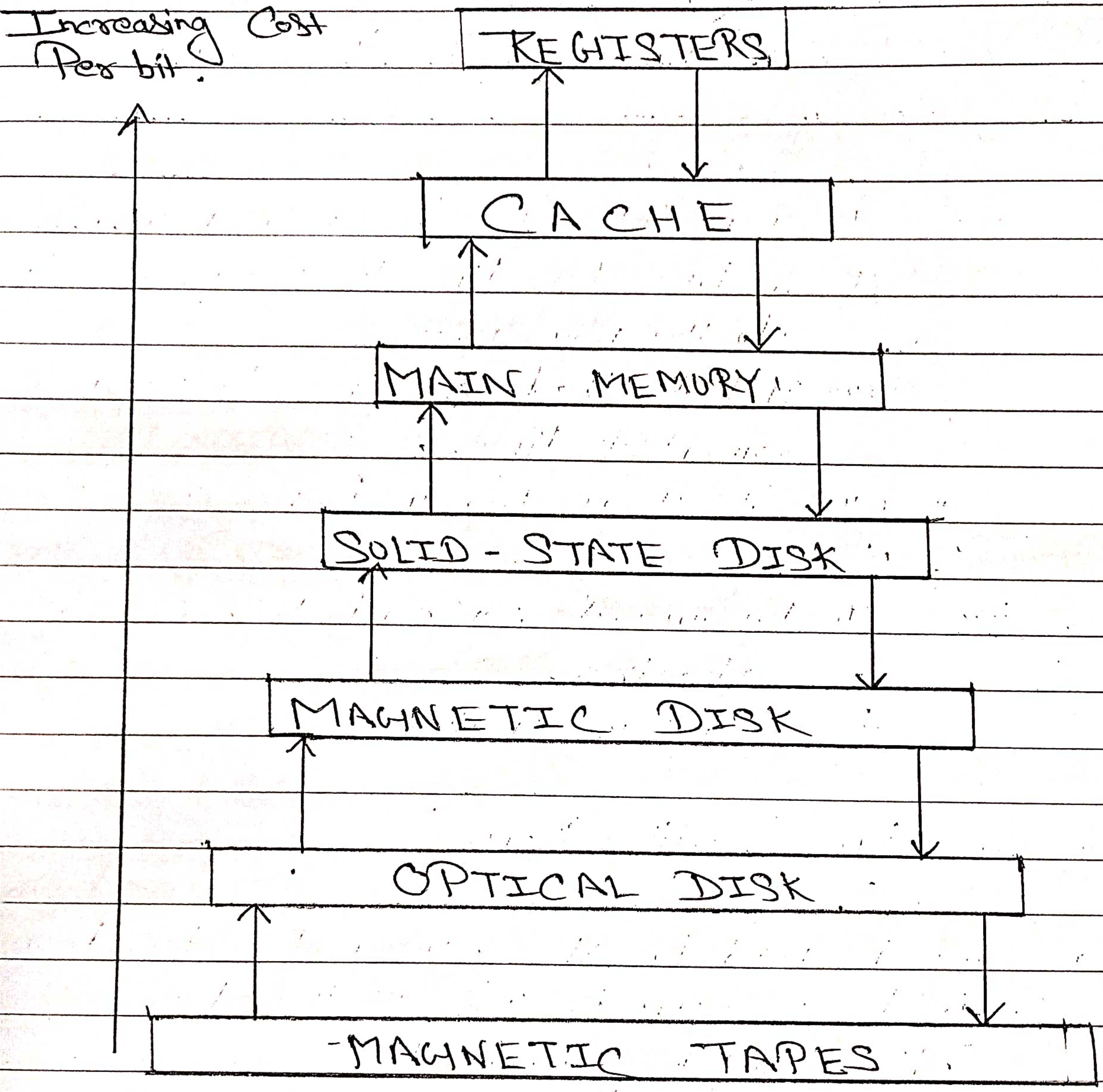
Volatile memory is used to Store information based on power Supply. if power Supply is off, all the data & information on this Computer will be lost. for example, RAM (Random Acess memory).

Whereas non-Volatile memory is used to Store information even when the power Supply is off. for example, ROM (Read only memory)

Criteria for Storage dividation :
① Access time - the time requrie to locate.
② Cost per bits Storage
③ Storage Capacity.

$$\boxed{\text{Cost} \uparrow \propto \frac{1}{\text{Storage} \downarrow}}$$

Memory Hierarchy



Operating System :

It has many defination, In general operating System is a huge Software that manage Computer ~~to~~ hardware & also performs Various function. It provides an environment in which a user Can execute programs Conveniently & efficiently. It acts as a Intermediate between user of a Computer & hardware. It provides GUI (Graphic user Interface) i.e Desktop, Icons.

Views of operating System :

1. User View
2. System or Hardware view .

$$\begin{tikzcd}
\boxed{\text{INSTRUCTION}} \arrow[r] & \boxed{\text{FUNCTION}} \arrow[r] & \boxed{\text{PROGRAMS}} \arrow[r] & \boxed{\text{MODULES}} \arrow[d] \\
 & & & \boxed{\text{SOFTWARE}}
\end{tikzcd}$$

User Interface

1. User .
2. System or application programs
3. operating System
4. Hardware .

Components of operating System :-

1. Process management → related to CPU
2. Memory management → related to RAM
3. Input / output Management
4. File Management
5. Secondary Storage management
6. Security
7. Command Interpreter .

1. Process Management : In multiprogramming Environment, the OS decides which process gets the processor when & for how much time .
   - → Keeps tracks of processor & Status of process .
   - → Allocates the processor (CPU) to a process
   - → De-allocates processor when a process is no longer requir

2. Memory Management : Memory management refers to management of primary memory (RAM) or main memory .

→ Keeps tracks of primary memory i.e What part of it are in use by whom, what part are not in use.

→ In multiprogramming, the OS decides which process will get memory when & how much.

→ Allocates the memory when a process requests it to do so.

→ De-allocates the memory when a process no longer needs it or has been terminated.

3. I/O Management : I/O Device management provides an abstract level of H/w devices & keep the details from application to ensure proper use of devices, to prevent errors, & to provide users with convenient & efficient programming environment.

→ Hide the details of H/w devices

→ Manage main memory for the devices using cache, buffer, & spooling.

→ Maintain & provide custom drivers for each divice.

4. File Management : It is one of the visible services of an operating system. Mostly files represent data, source & object forms, & programs. Data files can be any type like alphabetic, numeric, & alphanumeric.

→ File Creation and deletion.

→ Directory Creation & deletion.

→ The Support of primitives for manipulating files & directories

→ Mapping files onto Secondary Storage.

→ file ~~bacop~~ bacup.

5. Secondary Storage Management : Most programs, like compilers, Assemblers, Sort routines, editors, formatters & so on, are stored on the disk until loaded into memory, &

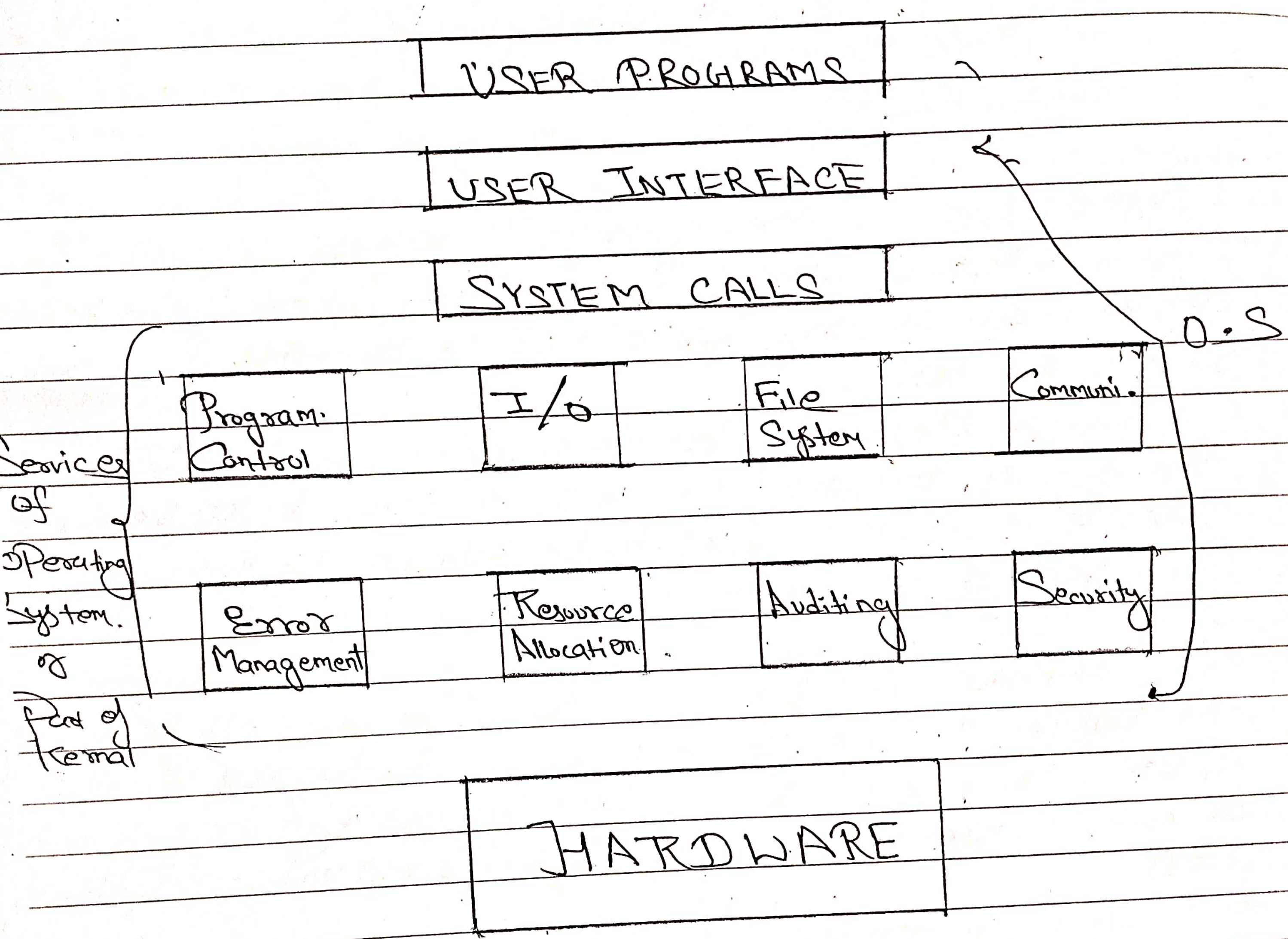
then use the disk as both the Source & destination of their processing.
→ Free Space Management
→ Storage allocation

6. Security : The operative System is primarily responsible for all task & activities happen in the Computer System. The various processes in an operating System must be protected from each other's activities.
→ Security Mangement refers to a mechanism for Controlling the acess of programs, processes or users to the resource defined by a Computers Control to be imposed, together with Some means of enforcement.

7. Command Interpreter : The Command interpreter is the primary interface between the user & the rest of the System. It executes a user ~~to~~ Command by Calling one or more number of underlying System programs or System Calls.
→ Command interpreter System allows human users to interact with the operating System & provides Convenient programming environment to the users.

System Calls

System Call is the Special function i.e used by the process to request action (or Services) from the operating System which Cannot be Carried out by Normal function. System Calls provide the end face between the process & the operating System.



When the process is being run, if the process required Certain action which need to be Carried out by OS, the process has to go Called the function which can interact with the Kernal to Complete the actions. the Special type of function Call is Known as System Calls in OS. System calls Provide the interface So that the process can Communicate with the operating System.

Note : Kernal is the Core Component of the operating System that has Complete Control of the hardware.

Types of System Calls

There are mainly five - types of System Calls Available : They are as follows :-

① Process Control → It handles the System Calls for process Creation, deletion, etc.
ex load, Execute, abort, Wait Signal events for process

② File Management → File Manipulation events like Creating deleting, reading & Writing etc are being classified Under file Management System Calls.

③ Device Management → This System Calls are being Used to request the device, release the device, logically attached & detached the device.

④ Information Maintainance → This type of System Call is Used to maintain the information about the System like time & date.

⑤ Communication → In order to have Interprocess Communications like Send or recieve the message, Create or delete the Communication Connection, & to transverse Status information etc, Communication System Calls are used.

Types of Computer Users :

① Normal users
② Programmers
③ Database Administrators
④ System Administrators.

Programming languages :-

① High level — App Software
② Low level — System Software.

Programming language translator :

① Assembler — Assembly language — low level.
② Interpreter } High-level language.
③ Compiler. }

① Assembler → Assembly language is in between the high level language & Machine language. It is Closer to machine language ~~is in b~~ than high level language. This language is not easily readable & Understandable by the programmer like a high level language. The assembler works as a translator in Converting the assembly language programe to Machine Code.

② Interpreter → An interpretor is also a high level language translator that Converts high level programs into machine Codes, Convert the Source Code to machine Code line by line. As it Checks line by line, the Scanning time is lower.

ex Python, Ruby, PHP, Perl based translator.

③ Compiler → A Compiler is a language translator that Converts high level programs into machine Understandable machine Codes. In this process, the Compiler Converts the whole program to machine Code at a time. If there are any Syntactic or Semantic error, the ~~Compilers~~ Computers will indicate them. It checks the whole program & displays all errors.

ex C, C++

Types of Errors :-

① Syntactical — Syntax, Errors.
② Symentical — logical.

How C Programs get translated.

While doing C Programming a user writes a program in C language (High level language). The C Compiler, Compiles the program and translates it to assembly program (low-level language) & Assembler than translates the Assembly program into machine Code (object Code). A linker tool is then used to link all the parts of the program together for execution (Executable machine Code).
A loader loads All of them into memory & then the program is executed.

Programming language translators :-

$$\begin{tikzcd} \text{① (Programmers} \arrow[r] & \text{Source Code} \arrow[r] \arrow[dr] & \text{HLL} \\ & & \text{LLL.} \end{tikzcd}$$

$$\begin{tikzcd} \text{② Machine} \arrow[r] & \text{object Code} \arrow[r] & \text{Machine Code} \end{tikzcd}$$

i.e 0001111 ....

1. Source Code : It refers to high level Code or assembly Code which is generated by human / Programmer Source Code is easy to read & modify. It is Written by programmer by using any high level language or intermediate language. In simple way, Source Code is a set of instruction / Commands & Statements which is Written by a programmer by using Computer programming language C, C++, Java, Python.

2. Object Code : It refers to low level Code which is Understandable by machine. object Code is generated from Source Code after going through Compiler or other translator. object Code Contains sequence of machine Understandable instructions to which Central processing unit Understands & executes. Some object file examples are Common object file format (COFF), COM files & ".exe" files.

Instruction Execution Cycle

① Fetch
② Decode
③ Execute

① Fetch → The next instructions is fetched from the memory address that is Currently Stored in the program Counter & Stored into the instruction Cycle.

② Decode → During this Stage, the encoded instruction presented in the instruction register is interpreted by the ~~doctor~~ decoder

③ Execute → The Control unit of the CPU passes the decoded information as a sequence of Control Signals to the relevant functional Units of the CPU to perform the actions required by the instruction, Such as reading Values from registers, passing them to the ALU to perform Mathematical or logic functions on them, & Writing the result back to a register.

Linker & Loader.

→ Linker is an important utility program that takes the object files, produced by the assembler & Compiler & other Code to join them into a single executable file.

→ A loader is a Vital Component of an operating System that is accountable for loading programs & libraries ..

$$\underset{\substack{\downarrow \\ \text{Pre-Processor Director}}}{\#}\ \text{include} < \underbrace{\text{Studio.h}}_{\substack{\text{Header file} \\ \text{or} \\ \text{library file}}} > \left\{ \begin{array}{c} \text{Printf} \\ \text{Scanf} \\ \text{Cin} \\ \text{Cont} \end{array} \right\}$$

Algorithms, Flowchart & Pseudocode

Algorithms → An algorithm is a Step by Step procedure to solve a problem. It is Complex to Understand, Use plain text, easy to debug, difficult to Construction, does not follow any rules.

Flowchart → A flow Chart is a diagram Created with different Shapes to Show the flow of data. easy to understand, Symbol shapes are used, hard to debug, Simple to Construct, follows rules to be Constructed.

CMM → Capability Maturity Model $\quad$ level 1-5

A typical programming task Can be divided into two phases:

① Problem Solving phase : In which an order Sequence of Steps that describes Solution of problem is produced. This Sequence of Steps is Called an algorithm.

② Implementation phase : In which a programmer implements the program in some programming language.

Steps in problem Solving :

Following may be Considered as Some of the Steps in problem Solving.

① First produced a general Algorithm (one Can also use Pseudo Code).

② Refine the algorithm Successively to get Step by Step detailed algorithm i.e Very Close to a Computer language.

(Pseudo Code is an artificial & Informal language that helps programmers develop algorithm.

Example - 1

Write an algorithms to determinant a Student's final grade & Indicate whether he/she is passing or failing. The final grade is Calculated as the average of 4 Marks.

$$\left.\begin{matrix} M_1 \\ M_2 \\ M_3 \end{matrix} \quad + \right\} \longrightarrow \text{GRADE} \qquad \frac{M_1 + M_2 + M_3 + M_4}{4}$$

Pseudo Code :-

```
→ Input Set of 4 Marks
→ Calculate their average by Summing & dividing by 4.
→ If average is below 50
      Print "FAIL"
  Else,
      Print "PASS"
```

Algorithm :-

```
Step 1 : Input M1, M2, M3, M4
Step 2 : $\text{GRADE} \underbrace{\leftarrow}_{\text{Equal to.}} (M1+M2+M3+M4)/4$
Step 3 : If (GRADE < 50) then
             Print "FAIL"
         Else,
             Print "PASS"
```

(Dictionary)
A flow chart is a Schematic representation of a Sequence of Operations, as in a manufacturing process or Computer program.

(Technical)
A flow Chart is a graphical representation of the Sequence of operation in an ~~f~~ information System or a program. Information System flow Chart Show how data flows from Source documents ~~to the Com~~ through the Computer to final distribution to users. Program flow Charts Show

The ess functions & Subroutin

Flow li

the Sequence of instruction in a single program or Sub-routine. Different Symbols are used to draw each type of flowchart.

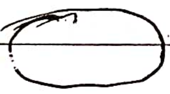
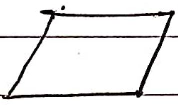
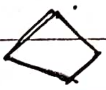
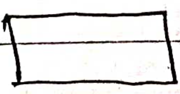
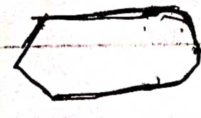
A flow Chart:

① Shows logic of ~~an~~ algorithm

② Emphasizes individual Steps & their interConnections.

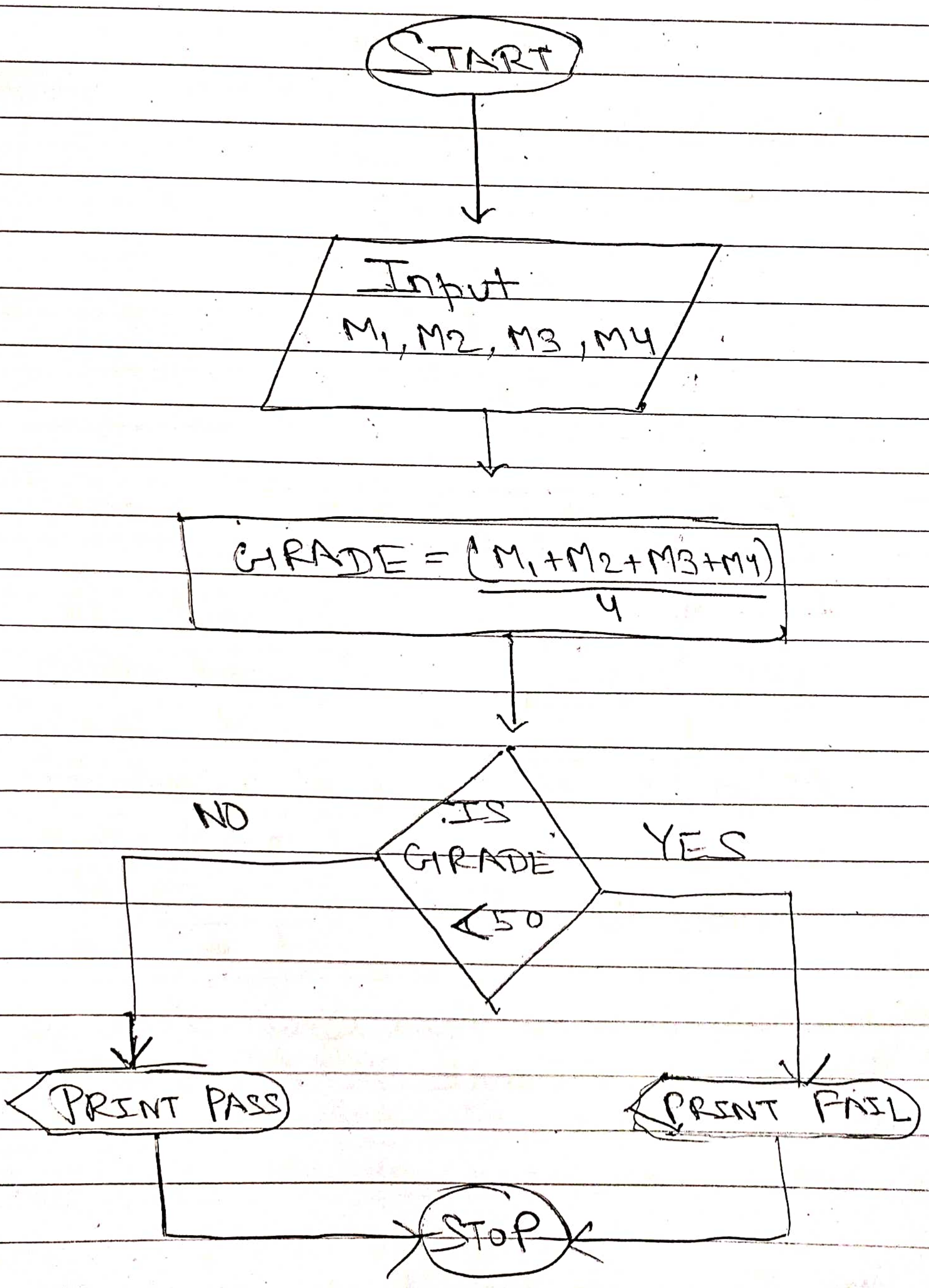
③ For ex - Control flow from one action to the next.

Flowchart Symbols:

Name	Symbol	Use in flow chart.
Oval or Terminal		Denotes the begining or end of the program.
Parallelogram		Denoting input & output operation.
Diamond		Denotes a decision or (Branch) to be make: The program Should Continue along one of the two routes. (for ex - If / Then / else)
Rectangle		Denotes a process to be Carried out (for ex:- addition, Subtraction, division, etc.)
Hybrid		Denotes an output operation

Flow line ← ↑ ↓ → Denotes the direction ~~or the~~ of the logic flow in the program

Flow of logic :
(i) Sequence
(ii) Decisions
(iii) Loops



Q Write the algorithm & draw of flowchart to Convert the length ~~inject~~ in feet into CM.

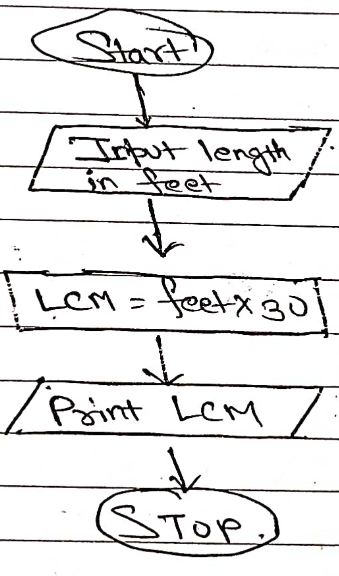
Algorithm

Step 1 : Take / Input the Value of length in feet.
Step 2 : Multiply the Value of length in feet by 30
Step 3 : Display the Value after Calculation.

Pseudocode

~~Step 1 :~~ Input length in feet
L'CM = feet × 30
Print LCM.

Flowchart.



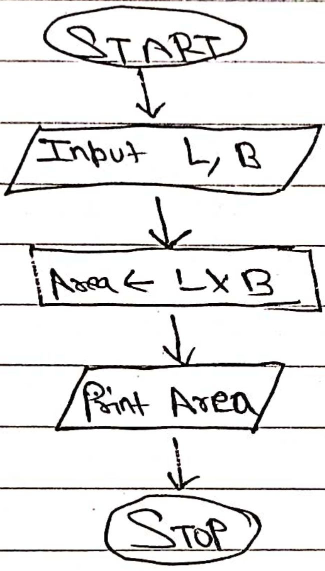
Q Write an algorithm and draw a flow Chart that will read the two sides of a rectangle & Calculate its area

Algorithm :
Step 1 : Take the Input for length & breadth of rectangle
Step 2 : Multiply the Values of length & breadth to get the area of the rectangle
[illegible]

Pseudocode

```
Input L, B
Area ← L x B
PRINT AREA.
```

Flow chart



Relational operators

(i)	Less than	<
(ii)	Greater than	>
(iii)	Equals to	= =
(iv)	Less than or equal to	< =
(v)	Greater than or equal to	> =
(vi)	Not equal to	! =
(vii)	Not	!

Q Write an algorithm & draw a flowchart that reads two values, determines the largest value and print the largest value with an identifying message.

Structure of C Programs.

Documentation Section
Definition Section
Global Declaration Section
Main( ) Function Section

{

Declaration Part
Executable Part

}

Sub program Section.

Function 1	
Function 2	{ user-defined functions }
Function n	

Every C program has only one main function, exact
one Main function & It have more than one in
Sub main function.

A C Program may Contain one or more Sections
Which are as follows :

① Documentation Section :- The documentation Section
Consist of a Set of Comment lines giving the name of the
program, the author & other details, Which the
programmer would like to use later.

(ii) Definition Section :- The definition Section defines all Symbolic Constant.

(iii) Global Declaration Section :- There are Some Variables that are used in more than one function. Such Variables are Called global Variables & are declared in the global declation Section i.e outside of all the functions. This Section also declaration of all user define function

Every C Program must have one main() function. This Section Contains two parts,

(a) Declation Part
(b) Executable part.

The declaration part declares all the Variables used in executable part. There is atleast one Statement in the executable part. These two parts must appear between the opening & closing braces.
The program execution becames at the opening brace & ends with the closing brace. The closing brace of the main() function Section is the logical end of the Program. All Statement in the declaration & executable parts end with the Semi Column (;).

The Subprogram Section Contains all the user-defined functions that are Called in the main function. User defined function are generally place ~~immediately~~ exactly after the main function, Although they may appear in any order.

Indentation → Improve readability.

Return 0 or 1 ← int main ( )

Main (Void)
int main ( )
Same { Void main (Void)
Void main ( )

All Sections, except the main() function Section may be absent when they are not require.

```
# include < Stdio . h >
Void    Main ( )
    {
        /* Printing Starts */
        Print f ("Hello. World") ;
          /* Printing ends */
      }
```

Function → A function is a Self ~~Contain~~ Contain block of Code which performs a particular task.

Compile → Alt + F9
Run → Ctrl + F9
output → Alt + F5

C Programs

Program to find area of a Circle.

```
/* Program to Calculate the area of
      a Circle */
    /* Title ( Comment ) */

# include <stdio.h>   /* LIBRARY FILE */
   Void Main( )       /* Function Heading */
```

output
→ Area of Circle
Variable Require
→ rad
→ area

```
{        Spare      space
          ↓          ↓
   float radius, area;               /* Variable Declaration */
   Printf ("Radius = ? ");           /* output Statement (Prompt) */
   ~~Print~~ Scanf ("%f", &Radius;        /* Input Statement */
   area = 3.14159 * radius * radius; /* Assignment Statement */
   Printf (" Area = %f", area];      /* output Statement */

}
```

Contents

$$\begin{tikzcd} & \text{① Intege.} \\ \text{① Primary (Buit in)} \arrow[ur] \arrow[r] \arrow[dr] & \text{② Real or floating point} \\ & \text{③ Character Constant} \end{tikzcd}$$

Variables → Variables are Small, temporary Storage locations which are used to Store Constant Values. Variables exist as long as the programs is Running.

%f → format String or Control String,
%d
%s
%c
& → address of operator.

15/09/22

The following features should be pointed out in this last Program:

① The program is typed in lower Case. either upper or lower Case Can be used, though it is Customary to type ordinary instruction in lower Case.

Most Comments are also typed in lower Case, though Comments are Sometimes typed in ~~upper Case~~ Uppercase for emphasizes, or to distinguish ~~Some~~ Certain Comments from the instructions.

(2) The first line is a Comment that identifies the purpose of the program (3) The Second line Contains a refrence to Special file (Stdio.h) Which Contains information that must be included in the program when it is Compiled. The inclusion of this required informatio will automatically be handled by the Compiler.

(4) The third line is a heading for the function ~~Main~~ name. the empty parentheses (Brackets) following the ~~Main~~ name of the function. indicate that this function ~~that~~ does not include any argument.

(5) The remaining five lines of program are indented ~~And~~ an enclose within ~~appear~~ a pair of braces (Curly) These five lines Comprise the Compound Statement within ~~name~~ main.

(6) The first indented line is a Variable declaration it Stablishes the Symbolic names radius & area as floating point Variables.

(7) The remaining four indented lines are expression Statement. The Second indented line (Printf) generates a request for information (namely, a value for the radius). This Value is entered into the Computer Via the third indented line (Scanf).

(& → amphersant)

* – Scanf ~~it~~ always 2 type ~~f~~ Parameters hote hai

<u>User defined functions</u>

(i) Declare the prototype.
(ii) Call the function.
(iii) Define the function.

\n . ~~[illegible]~~ → backward slash n
~~/n~~ → ~~forward slash~~
n .
→ New line Character.

Printf ("Hello \n World")
→ Display → Hell
World &

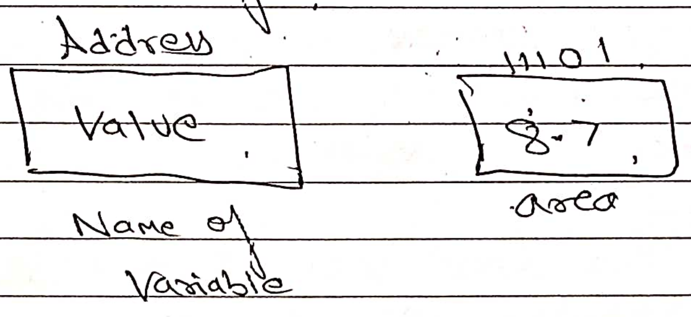
Parameters → n
Commas → n-1

~~Printf ("Area = %f'&~~

Printf (" Area = %f ", area);
→ variables jiski value Display karna hai
→ format string.

Scanf (" %f ", & radius);
→ address of

<u>Variables</u> →



Scanf (" %f ", & area);

⑧ The four indented line is a particular type of expression statement called an <u>assignment statement</u> this statement causes the area to be calculated from the given value of the radius. Within this statement the [(*) astricks] represent multiplication signs.

(9) The last indented line (Printf) Causes the Calculated Value of the area to be displayed. The numerical Value will be preceded by a brief level.
पहले

(10) Notice that, each expression Statement within the Compound Statement ends with the Semi colon (;).
This is required for all expression Statement.

(11) Finally, notice the liberal use of Spacing and indentation, Creating wide Space withing the program
विस्तृत स्थान रखना
The blank lines Separate different parts of the program into logically identifiable Components, & the ~~identification indicates Sub-o~~ indentation indicates Sub-ordinate relationships among the Various instructions. These features are not gramamatically essentials, but their presence ~~is~~ Strongly encouraged as a matter of good programming practice.

Program

I. Write a program in C to print an integer entere by the user.

```
/* Programming to display an integer */
#include<Stdio.h>
# include<Conio.h>
Void main()
{
    Clr.scr();
int num1 ;
Printf(" enter the number to be displayed ");
 Scanf(" %d", &num1);
 Printf("The integer number is = %d", num1);
 getch();
}
```

2. Write a program in C to add two integers number & show the results.

```
/* Programming to Sum of two integer numbers */
 #include<Stdio.h>
 #include <Conio.h>
 Void main()
{
      Clrscr();
      int num1, num2, Sum;
     Printf(" enter two numbers to be added");
      Scanf(" %d %d", &num1, &num2);
      Sum = num1 + num2 ;
      Printf(" Sum = %d", Sum);
      getch();
}
```

3. Write a program in C to Swap the Contents of two integer Variables. Values for Both Variables will be entered by the User.

```
/* Programming to Swap the Values of Variables */
 #include<Stdio.h>
 # include <Conio.h>
```

two variables

```
void main ()
{
    int a,b;
    Clrscr();
    Printf(" Enter First number: ");
    Scanf(" %d", &a);
    Printf(" Enter the Second number: ");
    Scanf(" %d", &b);
    (a = a+b);
    (b = a-b);
    (a = a-b);
    Printf(" Value after Swapping: ");
    Printf(" Value of a %d", a);
    Printf(" Value of b %d", b);
    getch();
}.
```

a = 20 , b = 10

a = a + b = 20 + 10 = 30

b = a - b = 30 - 10 = 20

a = a - b = 30 - 20 = 10.

a = 20 , b = 10

a = 10 , b = 20 } after Swap.

three variable

temp = a

a = b

b = temp

Date
20/09/22.

Learning C Language

Natural language.

$$\begin{tikzcd}\boxed{\text{ALPHABETS}} \arrow[r] & \boxed{\text{WORDS}} \arrow[r] & \boxed{\text{SENTENCES}} \arrow[r] & \boxed{\text{PARAGRAPH}}\end{tikzcd}$$

'C' Language

$$\begin{tikzcd}\boxed{\text{CHARACTER SET}} \arrow[r] & \boxed{\begin{array}{l}\text{VARIABLES}\\ \text{KEY WORDS}\end{array}} \arrow[r] & \boxed{\text{INSTRUCTION}} \arrow[r] & \boxed{\text{PROGRAM}}\end{tikzcd}$$

i) 'C' Character Set :

(a) Alphabets

$$\begin{tikzcd} & \text{A–Z} \\ \text{Alphabets} \arrow[ur] \arrow[dr] & \\ & \text{a–z}\end{tikzcd}$$

(b) Digits — 0-9

(c) Special Symbols

ii) ~~a. Variables :-~~
(a) Constants :-

$$\begin{tikzcd} & \text{'C' Constants} \arrow[dl] \arrow[dr] & \\ \text{Primary Constant} & & \text{Secondary Constant}\end{tikzcd}$$

Primary Constant
(Build-in Constants)
→ Integer
→ Real
→ Character

Secondary Constant
(User-defined Constants)
→ Pointers
→ Arrays
→ Structure
→ [illegible]

'C' Constants Can be divided into two Measure Categories:
(i) Primary Constants
(ii) Secondary Constant.

Rules for Constructing integer Constants :-
→ following are the rules for Constructing integer Constant -
1. An integer Constant Must have atleast 1 digit.
2. It must not have a decimal point.
3. It Can be either positive or negative.
4. If no sign preceeds an integer Constant, It is assume to be positive
5. No Commas or Blanks are allowed within an integer Constant.
6. The allow~~able~~ range for integer Constant is −32768 to 32767.

Rules for Constructing Real Constants :-
→ Real Constants are often Called floating point Constants. The real Constant Can be written in two forms — Fractional form & Exponential form.

Following rules must be observed While Constructing real Constants expressed in fractional form :

1. A real Constant must have atleast 1 digit.
2. It must have a decimal point.
3. It Could be either positive or negative.
4. Default sign is positive
5. No Commas or Blanks are allowed within a real Constant.

ex Real
+325.64 −32.76
261.0 −48.5792

−2,147,483,648 to 2147,483,647 → 32 bit Supports for Integer Constant

~~−9,223,372,036,854,775,808 to~~ → 64 bit Supports for Integer Constant.

$$\begin{tikzcd}
& \text{Primary Constant} \arrow[dl] \arrow[dr] & & \\
\text{Numeric Constants} \arrow[d] \arrow[dr] & & \text{Character Constants} \arrow[d] \arrow[dr] & \\
\text{Integer} & \text{Real (or floating Value)} & \text{Single Character Constants} & \text{String Constant.}
\end{tikzcd}$$

A real number may als be enpresed in enponential (or Scientific) notation.
For ex,
The Value 215.65 may be written as in enponential notation $\overset{\downarrow}{2.1565\,e^2}$ → means 10^2
e^2 means multiply by 10^2.
The general form of such number is
Mantissa e exponent.

The mantiss is either a real number enpressed in decimal notation or an integer. The enponent is an integer number with an optional (+) or (−) sign. The letter 'e' Separating the mantissa & the enponent Can be written in either lower or Upper Case.
$\underline{\text{Ex}}$ 2.1565 E 2 → $2.1565\,E^2$
Since the enponent Causes the decimal point to "float" this notation is said to represent a real number in floating point form.

Examples of legal floating points Constant are

0.65e4 1.5e+5

12e-2 3.18E3

-1.2E1

Single Character Constant →

A Single Character Constant (or Simply character Constant) Contains a Single Character & Close within ~~appe~~ a pair of Single Code marks.

ex

'5' 'x' ';' ' '

Note that the Character Constant '5' is not the Same as number 5. The last Constant (' ') is a blank Space. Character Constant has integer Value known as ASCII Values

ASCII →

Since, each Character Constant represents an integer Value, it is also possible to ~~for~~ Perform arithematic operations on Character Constants.

String Constant →

A String Constant is a sequence of Character and Closed in 'double Coats'. The Characters may be letters, numbers, Special characters & blank Space.

ex

"Hello" "1987" "WELL DONE" "?...!"

"5+3"

Date.
23-09-22.

C

BackSlash Character Constant :

C Supports Some Special backSlash Character Constants that are used in output functions.

~~(a) 7~~

Constant		Meaning
\a		audible alert (bell)
\b		backspace
\n		new line
\f		formfeed
\r		Carrage return
\t		Horizontal tap.
\v		Vertical tap.
\' (Slash Single cout)		Single Cort
\" (double cout)		double Cout
\? (Question mark)	—	Question mark
\\	—	Backslash
\0		null Value

: Variables :

A Variable is a data name that may be used to store a data Value. Unlike Constant that remain unchanged during the execution of a program, a Variable may take different Values ~~at~~ different time during execution.

A Variable name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program

ANSI.

(_) → underscore

[_average] → कुछ Systems मे चल जायेगा]

Some examples of such names are :

Average
height
Total
Counter_1
Class_Strength

Variable names may consist of letters, digits, & the underscore, subject to the following condition :

1. They must begin with a letter. Some systems permit underscore as the first character.
2. ANSI Standard recognizes a length of 31 characters. However, length should not be normally more than 8 characters, since only the first 8 characters are treated as significant by many compilers.
3. Upper case and lower case are significant. i.e, the variable Total is not the same as total or TOTAL.
4. It should not be a keywords.
5. ~~Write space~~ White space is not allowed.

Some examples of valid variable name are :

John	Value	T_raise.
Delhi	X1	ph_Value.
mark	Sum1	distance.

Invalid examples include :

123	(area)
%	25th.

Variable name	Valid ?	Remark
First_tag	Yes	
Char	No	Char is Keywords
Price $	No	dollor sign is ~~ke~~ special char
group one	No	White space not allowed
average_number	Yes	
int_type	Yes.	

Data Types

C language is rich in datatypes. Storage representation and machine instruction to handle Constants differ from machine to machine. The Variety of data types available allow the programmer to select the type appropriate to the needs of the application as well as the machine.

ANSI C Supports free classes of data types :

1. Primary (or fundamental) data types,
2. derived data types,
3. User defined data types.

All C Compilers Support 5 fundament data types :- namely Integer (Int), Character (Char), floating point (float), double-precision floating point (doubble) & Void. Many of them also offer extended data types such as long int and long double.

Data types

[illegible]

Primary Data Types

Integral Type.

Integer.

Signed	Unsigned.
int	unsigned int
Short int	unsigned Short int
long int	unsigned long int

Character.

- Char
- Signed char
- unsigned char

Floating Point Type.

Float	double	long double

Void

for 16 bit

Data type	Range of Values.
Char	−128 to 127
int	− 32,768 to 32,767
float	$3.4e^{-38}$ to $3.4e^{+38}$
double	$1.7e^{-308}$ to $1.7e^{+308}$.

/* Program to Calculate Simple Interest */

~~#include<stdio.h>~~
~~#[illegible]include~~

```
#include <Stdio.h>
#include <Conio.h>
Void main()
    {
     Clrscr();
     int P, N;
     float R, SI;
     Printf (" Enter principal amount ");
     Scanf (" %d", = &P) ;
    Printf (" Enter no. of years ") ;
     Scanf (" %d ", = &N ) ;
    Printf (" Enter Rate of interest ") ;
    Scanf (" %f", = &R) ;
     SI = (P * R * N) / 100
     Printf (" Simple interest = %f ", SI);
     getch () ;
   }
```

/* A Program
/* Ramesh's Bs = input at
DA = 40 % of Bs
HRA = 20 % of Bs
WAP to Calculate GS. (Gross Salary).

```
   #include <Stdio.h>
   #include <Conio.h>
Void main()
 {
Clrscr(); float BS, DA, HRA, GS ;
   Printf (" Enter Ramesh's Basic Salary") ;
   Scanf (" %f", &BS) ;
DA = BS * 0.4);
HRA = BS * 0.2;
GS = BS + DA + HRA;
Printf (" Gross Salary is : %f", GS); getch();
}
```

/ Program to Calculate Simple Interest /

```
#include <Stdio.h>
#include <Conio.h>
Void main()
{
    int P, ~~n~~;
    float Si, r;
    Clrscr();
Prinf (" Enter the Values for principal, number of
    years & rate of enterest to Calculate Simple
  Interest :");
 Scanf (" %d % d %f" &P , &n, &r);
 Printf(" Simple interest = %f", Si);
 getch();
}
```

29-09-22.

Keywords

Keywords are the words whose meaning has already been explained to the C Compiler (or in a broad sence to the Computer). The Keywords Cannot be used as variable names because if we do so, we are trying to assign a new meaning to the Keywords, Which is not allowed by the C Compiler. The Keywords are also Called 'Reserved words'

There are only 32 Keywords available in C and they are as follows :

Auto	else	int	Static	Volatile
break	enum	long	Struct	While.
Case	entream	register	Switch	
Char	float	return	typedef	
Const	for	Short	Union	
Continue	goto	Signed	Unsigned	
default	if	Size of	Void	
double				
do				

Rules that are applicable to all C Programs.

1) Each instruction in a C Program is written as a Separate Statement. therefore, a Complete C program would Comparise of a Series of Statements.

2) The Statements in a program must appear in the Same order in which we wish them to be executed; Unless of Course the logic of the problem demands a deliberate

) Jump or transfer of Control to a Statement, Which is out of Sequence :

③ Blank Spaces may be inserted between two words to improve the rediability of the Statement . However, no blank Spaces are allowed within a Variable , Constants or Keywords

④ All Statements are entered in Small Case letters .

⑤ C has no Specific rules for the position at which a Statement is to be Written thats Why it is often Called a free form language .

⑥ Every C Statement must end with a Semi Colon (;) thus, Semicolon acts as a Statement terminator .

Types of Instruction in C :-

There are different types of instructions Supported by C . and Understanding of these instruction is necessary in order to write programs effectively .

~~foot~~ following are the types of instructions in C :-

① Type declaration instructions :
② Input / output Instructions
③ Arithematic instruction
④ Control instruction .

Variable → (I) Name.
(II) Address
(III) Value.

1. Type declaration instruction :

These types of instructions are used to declare the type of variables used in a C program.

2. Input / Output instruction :

These types of instructions are used to take input from the user & to give output (or Display results) back to the users.

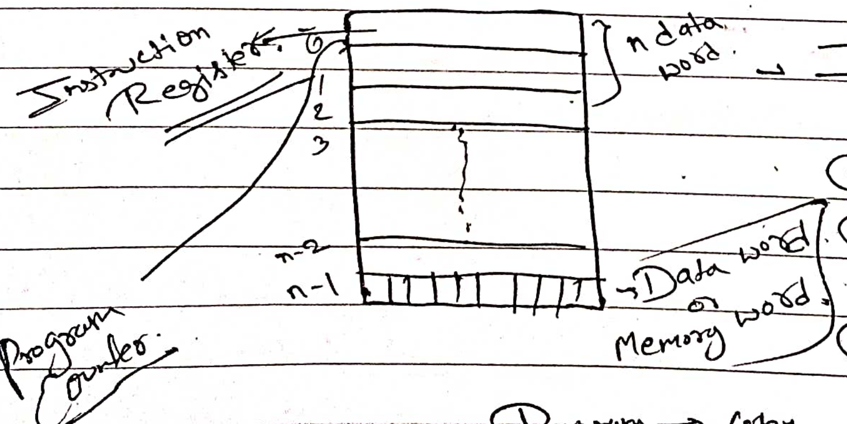
3. Arithematic instruction :

These instructions perform arithematic operations between Constants & Variables.

4. Control instructions :

These instructions are used to Control the Sequence of execution of Various Statements in a C Program.

{ Program Counter is a CPU register which holds the ~~value~~ adress of the next instruction to be executed. }



→ Instruction Cycle

(I) Fetch.
(II) Decode ← Mamonics
1, 0, 1, 0.
(III) Execute.

Program → Copy → Process

H/W → <Programs / Concept of C>

Process
↙ ↘
PCB — Actual Content
↓
Pointed Circuit B
Process Control Block.

14/10/22. CSIT.

1001	1001
Value	600
num 1	num 1.

~~Input~~ → Output Instruction.

Printf ("Message to be displayed");

output →

Printf ("The message related to the output & or one or more format Stri (or Control String"), one or more Variable-name

$$\begin{tikzcd}
\text{Admit.} \arrow[r] & \text{New} \arrow[r, leftrightarrow, "\text{Loaden in RAM}"] & \text{Ready} \arrow[r, "\text{Dispatch}"] & \text{Running} \arrow[r] \arrow[dl, "\text{I/O Request}"] & \text{terminated} \\
 & & \text{Blocked.} \arrow[u, "\text{I/O Request.}"] & &
\end{tikzcd}$$

S/W Development.

Analysis → Design → Coding → Testing → Implementatio
↓
Maintainance.

Input instruction

Scanf (" $\underbrace{\text{one or more formal String(s)"}}_{\text{I st argument}}$, $\underbrace{\text{one or more Variable\_names preceeded by address of operator for every Variable\_name}}_{\text{II nd argument.}}$);

20-10-22

Arithmetic Instructions :

A C Arithematic instruction Consist of a Variable name on the left hand side of equal to(=) and Variable names & Constants on the right hand side of equal to (=).

The Variables & Constant appearing on the right hand side of equal to(=) are Connected by arithematic operators like +, -, *, and /

Ex

```
int ad ;
float Kot , deta, alpha, beta, gamma ;
ad = 3200 ;
Kot = 0.0056 ;
deta = alpha * beta / gamma + 3.2 * 2/5 ;
```

Here, ① * (astrics) , / (slash) , - (Minus) , + (plus) are the arithematic operators.

② = $\overset{is}{\Rightarrow}$ assignment operator.

③ 2, 5 & 3200 are integer Constants.

④ 3.2 & 0.0056 are real Constants.

⑤ ad is an integer Variable

⑥ Kot, deta, alpha, beta, gamma are real Variables

The Variables & Constants together are Called 'operants' that are operated upon by the 'Arithematic operators' & the result is assign, using the assignment operator, to the Variable on left hand side.

a C arithmetic Statatement Could be of three types & they are as follows :

① Integer mode arithmetic Statement
This is an arithematic Statement in which all operants are either integer Variable or integer Constants.

(II) Real Mode arithematic Statement
This is an arithematic Statement in which all operants are either real Constants or real Variables

(III) Mix Mode arithematic Statement
This is an arithematic Statement in which Some of the operants are integers & Some of the operan are real.

It is Very important to understand how the executi of arithematic Statement takes place.

firstly, the right hand Side is evaluated using Constants and the numerical Value Stored in the Variable names. This Values then assigned to the Variable on the left hand side.

Though, Arithematic ~~St~~ Instruction look Simple to use one often Comments mistakes in writing the

Note the following points Carefully.
① C allows only one Variable ~~of~~ on left hand side of = i.e Z = K+l ; is Valid but K+l = Z ; an invalid instruction.

for Quotient → /
for remainder → %

के अलावा

(2) In addition to the division operator / C also provides a modular division operator (%) & this operator returns the remainder on dividing one integer with another. Thus the expression 10/2 yields 5 where as 10 % 2 yields 0. note that the Modulus Operator Cannot be applied on float. Also note that the ~~Modulus operator~~ on using modulus operator the sign of the remainder is always same as a sign of ~~is~~ the numerator thus,

```
 numerator
   ↓      ↓denominator
  -5  % 2   yields   -1
Where as   5  % -2   yields   1
           ↓     ↓
     numerator   denominator
```

(3) An arithematic instruction is often used for storing character constand and character variables.

```
Char a, b, c ;
 a = 'F';
 b = 'c' ;
 c= '*' ;
```

When we do this the ~~askey~~ ~~as~~ ASCII values of the character stored in the variables. ASCII values are used to represent any character in memory.

) Arithematic operations can be performed on ints, ~~is~~ floats & chars.

```
Char x, y ;
int z ;
x = 'a' ;
y = 'b' ;
z = x+y ;
```

⑤ No operature is assume to be present. It must be written explicitely. in the following examples, multiplication operator after b must be explicitely written.

$a = c.d.b(xy)$ → user arithematic statemen

$a = c * d * b * (x * y)$ → for C.

⑥ Unlike other high level language, there is no operator for performing exponentiation operation. Thus, followin statements are invalid.

$$\left.\begin{aligned} a &= 3**2 \\ b &= 3^\wedge 2 = 3^2. \end{aligned}\right\} \times$$

If you won't do the exponentiation, we get ~~go~~ it done in the following way :-

```
#include <math.h>
main()
{
    int a;
    a = pow(3,2);
    printf(" %d ", a);
}
```

Here Pow() function is a standard library function. It is being used to raise 3 to the power of 2.
#include <math.h> is a pre-processor directive.
It is being used here to ensure that the Pow function works correctly.

In order to effectively develop C programs, It will be necessary to understand the rules that are used for the implicit conversion of floating point and integer value in C.

5) 20 (.04

These are mentioned below.
Note them Carefully.

1. An arithematic operation between an~~t~~ integer and integer always yields an integer ~~reg~~ result.

2) An operation between a real & real always yield a real result.

3) An operation between an integer and real always yields a real result. In this operation the integer is first promoted to a real and then the operation is Performed. Hence, the result is real.

Operation	Result.
5/2	2
5.0/2	2.5
5/2.0	2.5
5.0/2.0	2.5
2/5	0
2.0/5	0.4.