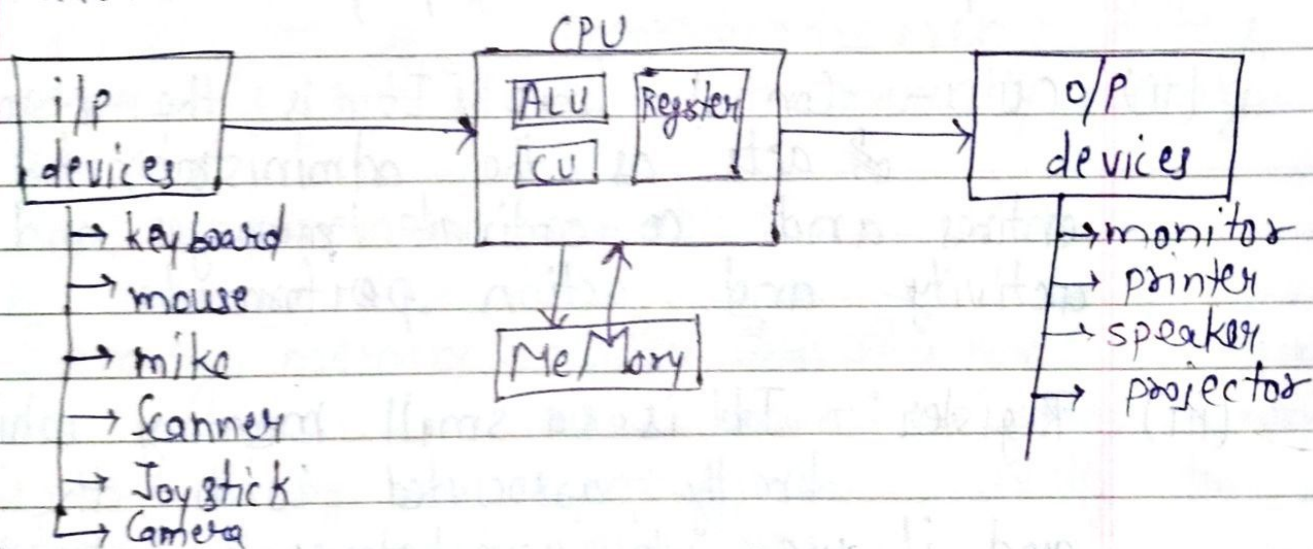


Programming for problem Solving

- ✓ Fundamentals ✓
- ✓ Numeral System ✓
- ✓ Programming ✓

Computer is electronic device which is processing & storing of the data.

Data :- Data is a raw facts & figures where as unformation is meaningful processed data.



1) Input devices:- These are the devices which are used to take the input (data) from users & pass into central unit. The Analog input such as keypress or mouse click generate the electric signal which then transform to digital form.

eg- keyboard, mouse, scanner etc.

2) Output devices:- These are the devices which are used to received process data

from the central unit & send it to users.
eg- monitor, printer, speaker etc.

CPU :- It is a component where further actual process is done. It is catalyse into many parts

(ii) ALU :- ~~Arithmetical~~ Logical unit. It is the segment of the processor where the all the calculation, logical decision, reasoning & decision making happen. Every data is actually process in this segment

(iii) CU :- Control unit; It is the segment of acts as the administrator for the entries and co-ordinate/manages and all activity and action performed.

(iii) Register :- It is a small memory which is directly associated with the CPU and it use to save/store the intermediate values

Program :- Program is the set of the instructions that the computer follows in order to perform the particular task. Indirectly we ~~store~~ give the instructions and obtain the result.

Memory :- Memory is a store media which is used to store the data and program. It can be primary memory, secondary memory and cache memory further based on the retainment of information. Memory can be Volatile and non-volatile.

(i) Volatile memory :- This type of memory keep information as long as power is supplied. It is just temporary memory.

(ii) Non-Volatile memory :- This type of memory keep ^{off} information even after power is supplied. It is permanently memory.

Type of memory

(i) Primary memory :- ~~It is a~~ This type of memory is used by the CPU directly to store data and instruction. also known as main memory

RAM :- (Random access memory) (Read/Write memory)
It is the memory which used by CPU to store data and instruction. It is currently working upon. This data can be access standonly and can be written.

SRAM (Static Ram)

- 1) It is used series of transistor (flip flop configuration)
- 2) It doesn't require to be Refresh.
- 3) It is much fast.
- 4) It is expensive.
- 5) eg - Cache memory

DRAM (Dynamic Ram)

- It primarily used capacitor to store the data.
- It requires periodic Refreshing as it leaks data.
- It is not much fast.
- It doesn't expensive eg - main memory ram

ROM:- Read only memory. It is the memory where the data and program are burned on to the chip that is permanent.

Types of ROM

- (i) PROM (Programmable ROM)
- (ii) EPROM (Erasable " ")
- (iii) EEPROM (Electrically " ")

(i) PROM (Programmable ROM) \Rightarrow It is the type of ROM which can be program by the user only once.

2) EPROM \Rightarrow In this case the contains of the ROM can be erase a few times

by using a special device which emits ultra violet rays.

3) EEPROM \Rightarrow In this case the contains are erases by using a burst of electric current enhances longevity. ~~reliable~~ and reliable

Cache memory \Rightarrow Cache memory is a type of memory which exist b/w RAM and Register and is used to store data and programs that are frequently used by the CPU. It is much faster than RAM but as lower capacity and is expensive

Secondary memory \Rightarrow It is mass store device which is used to store user data and programs.

It is non-volatile and as a higher capacity compare to primary memory but is also slow and is inexpensive. eg hard disk, SD card, Pen drive, CD/DVD, Magnetic tape disk

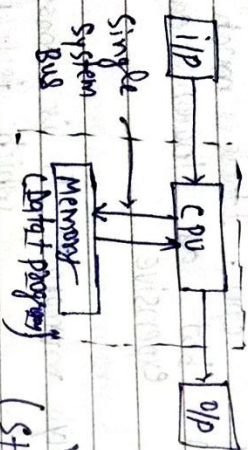
STORED PROGRAM ARCHITECTURE (SPA) \Rightarrow

Also known as von Neumann architecture it is a architecture model based on shared program. Concept states that the data and the instruction which operate of data are ~~stored~~ and

9. 60 minutes - 11.00 AM

connected to the CPU through up single system bus, the CPU would sequentially access the data and program from the memory.

memory. Due to a single system bus the CPU would only access data and program one at the time. This problem is known as von-neumann Bottleneck and the solution to the problem is to use separate bus for each data and program.



Van-Neumann
(Stored prog. concept)

Algorithms :- Algorithm is a list of steps in sequence to solve a problem. There are two ways to write a algorithms.

1.1 Pseudo code
3.7 Flow chart

↳ Pseudo code :- It is a ^{finite} list of 'english' like instructions in sequence to perform a specific task.

Q. Write a pseudo code to make tea.

Solve- instant

- (iii) Add water in container
- (iv) Put the container on top of lighter gas.
- (v) Add sugar and tea leave in container.
- (vi) Add milk to the container
- (vii) Boil for few time.
- (viii) Stop

Rules of writing Pseudocode

(i) It must always 'start' and must always 'stop'.

(II) It must be finite and numbered.

(III) Each step must be self explanatory.

(IV) It must not contain ambiguous (confused)

and long sentence.

(V) $\mathbb{I}^{\mathbb{I}}$ and \mathbb{S} contain mathematical expression.

(1) ~~Set~~ Sequence Pseudocode

0-1 Water an algorithm to add two number.

(ii) Take two numbers (input A, B)

(iii) Add expected numbers $(C = A+B)$

(v) Stop.

8. Write an algorithm to calculate area and circumference of a circle.

Solve-(i) Start

- (iii) input R
- (iv) $A = \text{pio} * R * R$
- (v) $C = 2 * \text{pio} * R$
- (vi) ~~Set~~ Set $\text{pio} := 3.14$
- (vii) Show A and C
- (viii) Stop

Q. Write an algorithm to calculate temp. in °C when Fahrenheit is give.

Ans-

- (i) Start
- (ii) input F
- (iii) ~~else~~ set $C := (F - 32) * 5/9$
- (iv) Show C
- (v) Stop

(ii) Selective Algorithm \Rightarrow

Q. Write an algorithm to find maximum b/w two numbers.

Ans- (i) Start

(ii) input A, B

3.1 if $A > B$ then

3.2 Show A

3.3 else

3.4 Show B

3.5 end if

(4) Stop

Q. Write an algorithm to check if a number is negative or pos.

Ans- (1) Start

(2) input A

3.1 if $A < 0$ then

3.2 Show "negative"

3.3 else

3.4 Show "positive"

3.5 end if

Q. Write an algorithm to find minimum b/w three numbers.

Ans-

(i) Start

(ii) input A, B, C

(iii) if $C < B$ then if $C < A$ then

3.2 Show C

3.3 else if

3.4 ~~Show B~~ $B < A$ and $B < C$ then

3.5 Show B

3.6 else

3.7 Show A

3.8 end if

(4) Stop

Q. Write an algorithm to check if a person can vote.

Ans- (i) Start

(ii) input age

(iii) if age ≥ 18 then

3.1 print you can go for vote

3.2 else

3.3 print you can't go for vote

3.4 end if

(iv) Stop

If condition :-

If Cond" then

else if Cond" then

else if Cond" then

else if Cond" then

end if

Q. Write an algorithm to print Day of the week depending upon using input.

Ans-

```
(i) Start
(ii) Input A
    if A = 1 then
        print "Monday"
    else if A = 2 then
        print "Tuesday"
    else if A = 3 then
        print "Wednesday"
    else if A = 4 then
        print "Thursday"
    else if A = 5 then
        print "Friday"
    else if A = 6 then
        print "Saturday"
    else if A = 7 then
        print "Sunday"
    else
        print "Invalid day"
    end if
(iii) Stop
```

Q. Write an algorithm to find minimum b/w three no.

Ans-

```
1 Start
2 input A, B, C
3 if (A < B) and (A < C) then
    print "A is smallest"
4 else if (B < A) and (B < C) then
    print "B is smallest"
5 else
    print "C is smallest"
6 end if
7 Stop
```

Q. Write an algorithm to check a person can vote and date is 28 Aug 2018.

Ans-

```
1 Start
2 input age and date = 28/8/18
3 if date = 28 Aug 2018
4 if age >= 18 and (date = 28/8/2018) then
    print "person can vote"
5 else if (age < 18) and (date is not 28/8) then
    print "person cannot vote"
6 end if
7 Stop
```

(3)

Iterative procedure :- This type of code is implemented when we want to repeat certain step a fix no. of times.
It's used the key word 'repeat' and 'write'.

Q.1 Write an algorithm to show your name 100 times.

Ans-

```
(i) Start
(ii) Set count := 1
(iii) Repeat step 4, 5 until count <= 100
(iv) Show "India"
(v) Count = count + 1
(vi) Stop
```

Q.2 Write an algorithm to print counting from 1 to 10.

Ans-

```
1 Start
2 Set count := 1
3 Repeat step 4, 5 until count <= 10
4 Show "count"
5 Count = count + 1
6 Stop
```


Q.3 Write an algorithm to print multiplication table of 7.

Ans-1 Start
 2 set count := 1
 3 Repeat 3.2, 3.3 until CV <= 7
 3.2 Print Count
 3.3 count = count + 1
 4 Stop

Q.4 Write an algorithm to show table of

on input value 'n'.

Ans-1 Start
 2 Set count := 1
 3 Repeat 3.2, 3.3 until CV <= n
 3.2 Print Count * n
 3.3 count = count + 1
 4 Stop

Q.5 Write an algorithm to find

(I) Sum of first 150 numbers

(II) Sum of first n numbers

(III) Sum of numbers from 100 to 500

Ans-1 Start

2 Set count = 1, S = 0

3 Repeat step 4, 5 until count <= 50

3.2 Print count + 1

3.3 count = count + 1

4 Stop

5 S = S + a

6 a = a + 1

7 Print S

8 Stop

(II) 1 Start

2 a := 1, S := 0

3 Repeat 4, 5 until a <= n

4 S = S + a

5 a = a + 1

6 print S

7 Stop

(III) 1 Start

2 a := 100, S := 0

3 Repeat 4, 5 until a <= 500

4 S = S + a

5 a = a + 1

6 print S

7 Stop

Q. Write an algorithm to calculate factorial of

a number.

Ans-1 Start

2 input n, S = 1

3 Repeat 4, 5 until a <= n

4 a = n * (n-1) S = S * a

5 a = a - 1

6 print S

7 Stop

Q. Write an algorithm 2 no. of LCM.

(a * b) = LCM(a, b) * gcd(a, b)

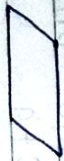
Flow chart \Rightarrow It is a diagrammatic representation of an algorithm. It is specifically drawn to perform certain task.

Symbols of flow chart :-

1. Begin / End :-



2. Input / output :-



3. Processing :-



4. Flow :-



(i) Begin / End :- This symbol marks the start or stop of the flowchart and it is represented by a oval.

(ii) Input / output :- This symbol is used to implement input / output statement and is represented by a parallelogram.

(iii) Process :- This symbol is used to implement calculation and assignment. It is represented by a rectangular.

(iv) Decision :-



This symbol is used to perform analysis on the statement which can either be true or false. It is generally used if an statement. And it is represented by a ~~rectangle~~ rhombus (diamond).

(v)

Connector (end of page) :- This symbol is used in flowchart to connect the flowchart on to the page. It is represented by an circle.

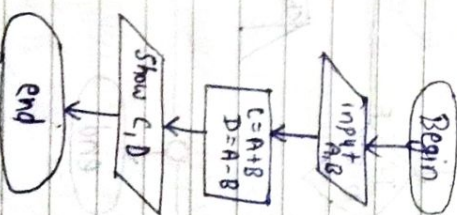


(vi) Flow line :-

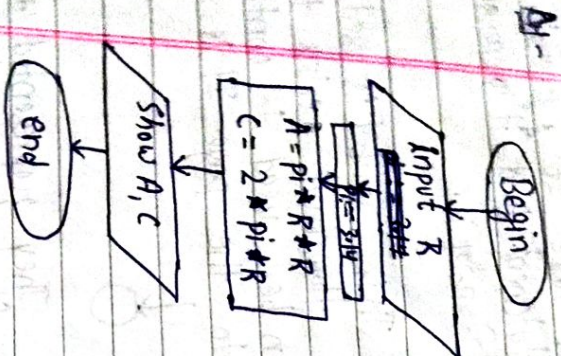


They represented the flow of the task. They generally in the downward direction and can not be curved. They are represented by arrows line.

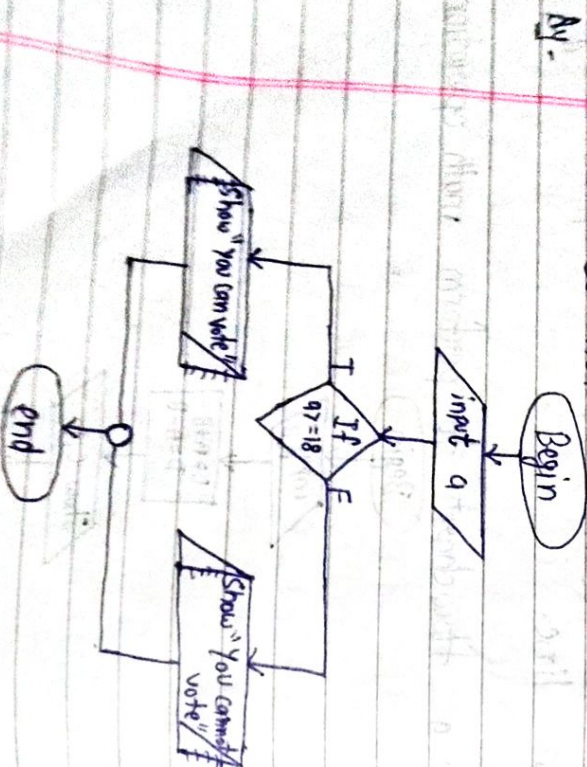
Q. Draw a flowchart to perform math operation.



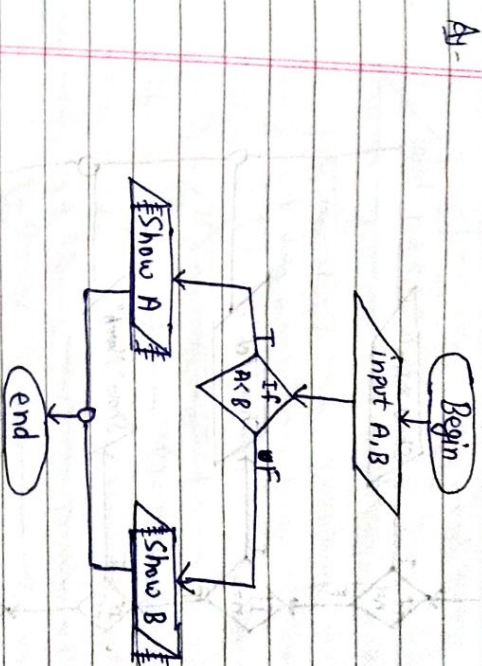
Q. Draw a flowchart to calculate area of circle its circumference.



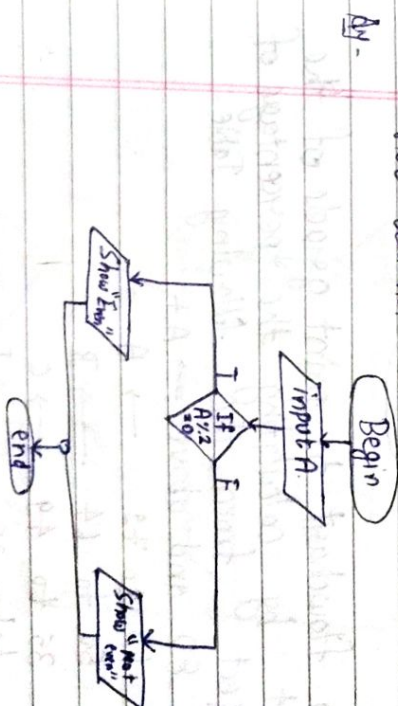
Q. Draw a flowchart to check if a person can vote or not.



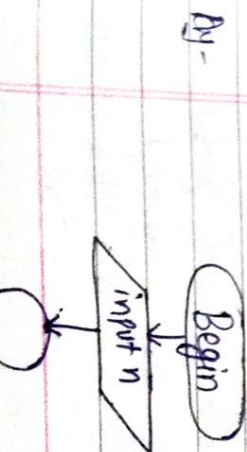
Q. Draw a Flowchart to find minimum b/w two numbers.

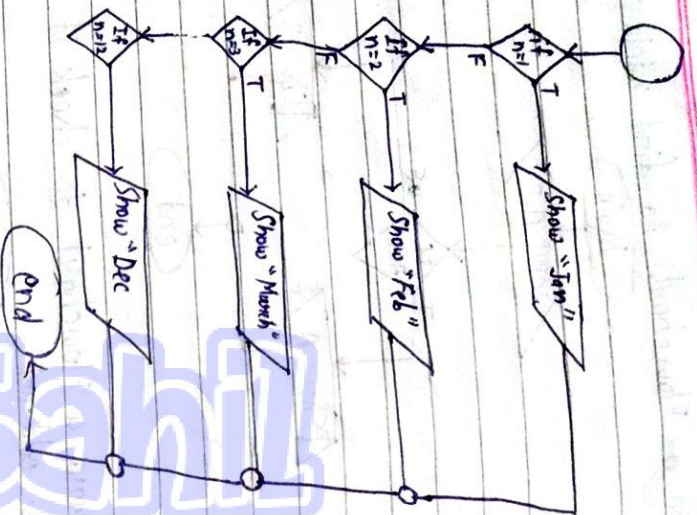


Q. Draw a Flowchart to check if a no. is even or not.



Q. Draw a Flowchart to print month of the year depending upon input.



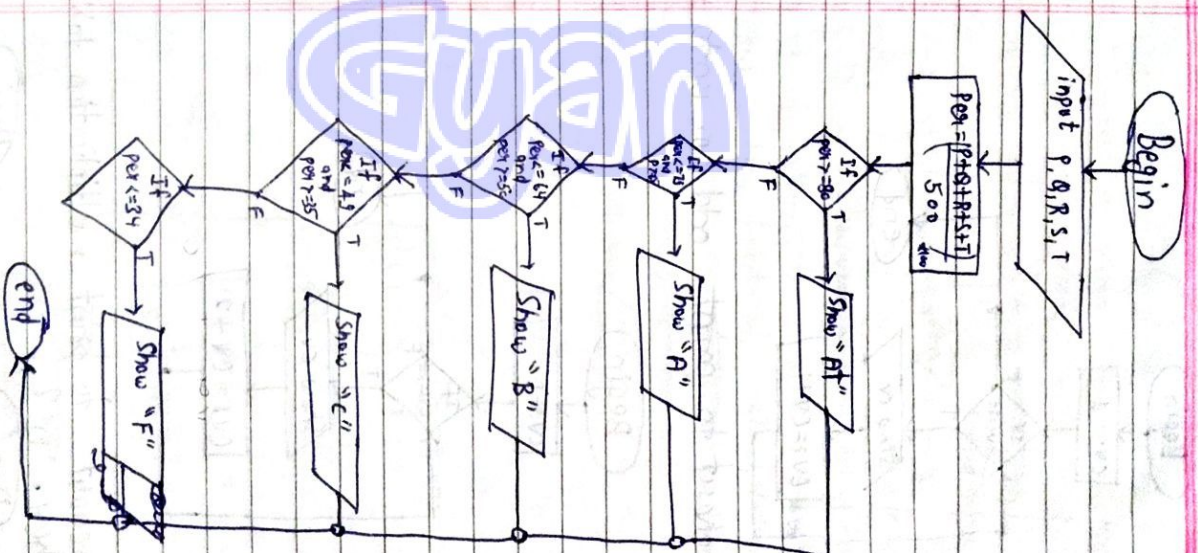


- Q. Draw a flowchart to print grade of the student by calculating the percentage of 5 subject from the following table.

80 and above \rightarrow A+
 65 to 79 \rightarrow A
 50 to 64 \rightarrow B
 35 to 49 \rightarrow C
 below 35 \rightarrow F

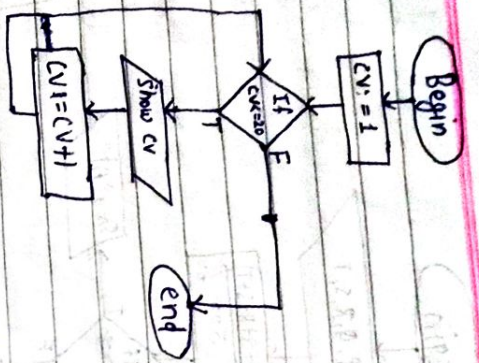
Begin

Begin



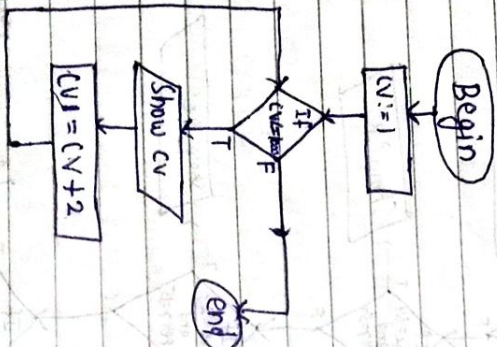
- Q. Write Draw a flowchart to print number from 1 to 20.

AI-



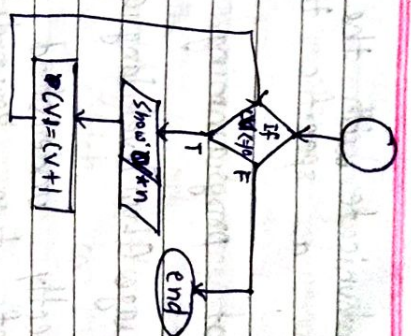
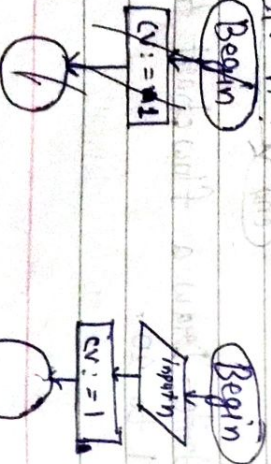
Q. Draw a flowchart to print odd no. upto on the 1000.

Ans-



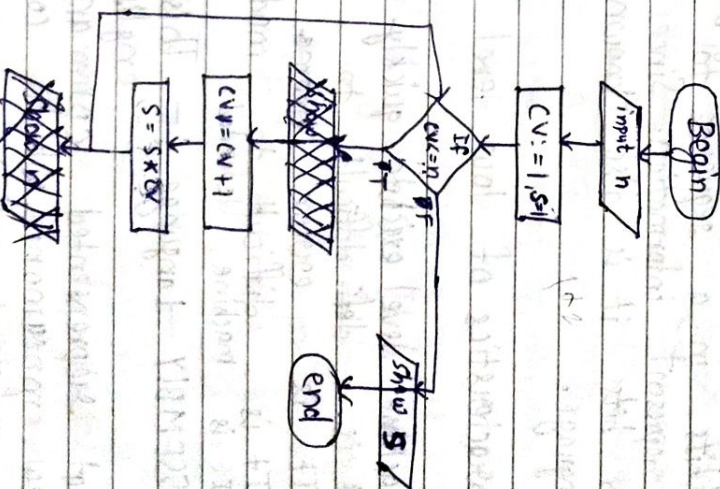
Q. Draw a flowchart to print multiplication table of any number 'n'?

Ans-



Q. Draw a flowchart to calculate factorial of a no.

Ans-



Programming languages \Rightarrow A programming language is that it is the set of rules of or regulation following a specific grammar to generate programs in

order to communicate with the computer to perform a specific task / activity.

Categories of programming language \Rightarrow

- (i) Machine language (Low level language) :- It is a language which is directly understood by the computer. It consists of two keywords (bits) represented by 0 & 1. This language uses these bits in a sequence to generate & represent information. Since, it has only two bits it is also known as binary language.

Characteristics of low level language

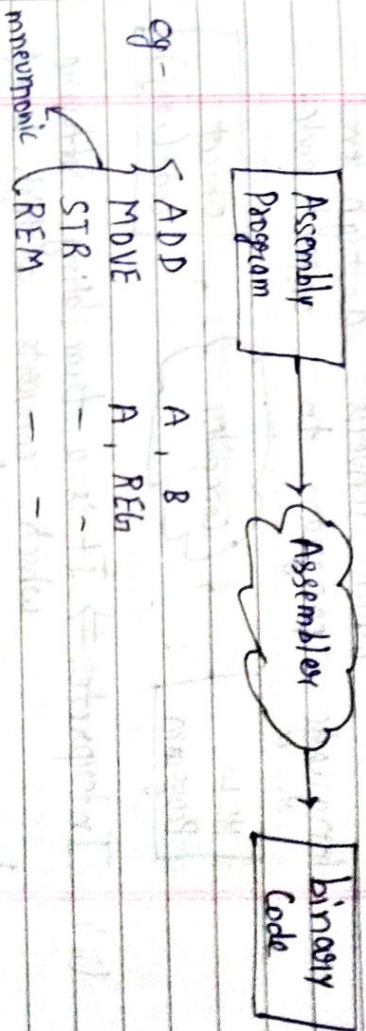
- 1) Machine level exists quickly.
- 2) It is ~~not~~ difficult to understand.
- 3) It is error ~~prone~~ prone.
- 4) It is difficult to modify.
- 5) It is machine dependent.

2.2 ASSEMBLY Language :- This language uses English like words that represented commands, known as mnemonics which contain instructions to perform actions.

- Characteristics of assembly
- 1) It is comparatively easy to understand.
 - 2) It is less error prone.

- 3) It is comparatively slow.
- 4) It is machine dependent.

Since, assembly language is not in binary, for the computer, understood it requires a translating software (translator) known as assembler which converts assembly program to binary code.



- (3) High level Language :- It is uses symbolic code containing English text and mathematical symbols which makes it easy to read & write programs.

Characteristics of HL :-

- (i) It is machine independent.
- (ii) They are easier to understand and write.
- (iii) Errors are few and can be easily remove.
- (iv) It takes more space and slow to execute.
- (v) They are may and may not be able to generate specific hardware oriented programs.

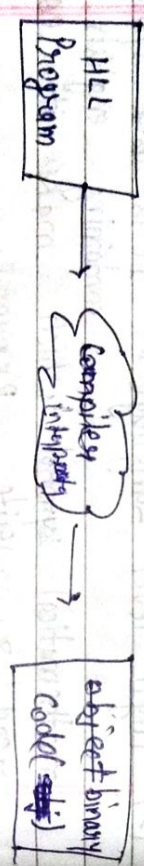
Since, a program is written in high level. It requires translation. There are two types of translation for high level languages.

- (i) Compiler
- (ii) Interpreter

1) Compiler \Rightarrow It is a translating software which converts a high level language program to object code.



(ii) Interpreter \Rightarrow It is a translating software which converts a high level language program to binary code.



#

Compiler	Interpreter
1) It translates entire code in a single step.	1) It translates code line by line
2) The errors are produced at all one	2) The next error is shown only when previous error is removed.
3) Compiler execute fast.	3) Interpreter execute comparatively slow.

- 4) debug is difficult
- 5) Compiler produces intermediate code.
- 5) Interpreter does not produce object code.
- 4) Debugging is easier.

C programming

Overview of C \Rightarrow C is a versatile, powerful, portable general purpose

programming language which provides a rich set of build in function and library to write complex program.

\rightarrow It was designed in 1972 by Dennis Ritchie and is designed to write Unix operating system.

C is a mid level language which supports and provide

- (i) Structural Programming
- (ii) Functional Programming
- (iii) Low level (Mid wise) programming
- (iv) High level paradigm

Advantage of C \Rightarrow

- (i) It supports hardware oriented hardware independent design and program.
- (ii) It is extremely stable.
- (iii) It is processor much faster.
- (iv) It is small with only 32 keywords

Disadvantage of C \Rightarrow

- (i) It is not suitable for writing lengthy programs.
- (ii) It gives less importance to data.

- (iii) It can not replicate real world entities properly.
- (iv) It is less secure.

Uses of C \Rightarrow

- (i) C provide an excellent base on system level programming and thus is use to write compiler & interpreter.
- (ii) Language library
- (iii) Hardware oriented program
- (iv) Software drivers
- (v) Operating system module

Problem Specification \Rightarrow
& Modelling of a program

- (i) Defining a problem:— In order to write a program it is required to properly specify the problem for which the program is written.

The problem must be such so that it is complete and free from any ambiguity (Confused).

- (ii) Specifying an analysis:— Once a problem is define strategy need to be build based on decision taken for the problem and a proper workout plan needs to be generated.

- (iii) Designing the solution:— Based on the plane requirements needs to

be specify such as store require data structure used etc.

(4.) Writing codes

- (5) Testing and maintaining :- Once the program is written it is thoroughly tested individually and as a whole to find out any anomalies and generate a solution.

Structure of C program \Rightarrow It specify the way a C program

is written.

- (1) comments
 - 1) Preprocessor directives
 - 2) Global declarations
 - 3) Function declarations
 - 4) ~~main~~ main functions
- {
- local declarations
- other statements
- }
- (5) User defined functions
- {
- local declarations
- other statements
- }

- (10) Comments \Rightarrow These are the statements which provide information about the program to the user. Comments are not read by the compiler. There are two types of comments :-

(i) Single line comment $//$:- They comment out an entire line.

(ii) Multiline comment $/* */$:- They comment out multiple lines.

- (1) Preprocessor directives \Rightarrow These are preprocessor commands that are executed before the compiler translation. They always being with pound symbol (#)

eg-

```
#include <stdio.h>
// (command) (header file)
// preprocessor directive

# define MAX 10
```

- (2) Global declaration \Rightarrow These declarations are visible to be entire program.

\rightarrow declaration is introduction of a new entity to the program.

- (3) Function declaration \Rightarrow These declaration specify the user created function which are supposed to be used later in the program.

(4.) Main function \Rightarrow This function is ~~sup~~ supply by the compiler and is the first function which is processed by it.

\rightarrow Function:— function is ~~that~~ it is a group of statement written within curly bracket to perform a certain task.

o Block of a function:— Represented by the body of the function a curly braces is

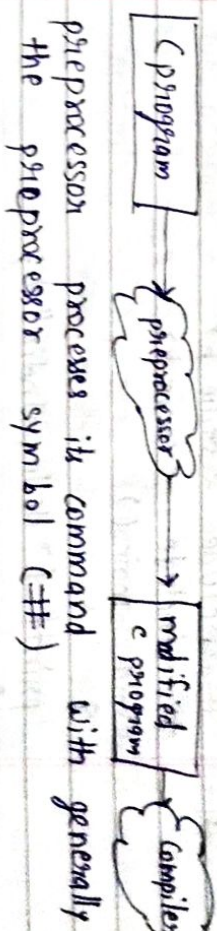
o ~~the~~ Local declaration:— These declaration are limited within block.

o Other statements:— They specify the task to be perform.

(5.) User define functions \Rightarrow These are the created by the user for certain ~~perfor~~ purpose.

eg - `#include <stdio.h>`
`void main()`
`{`
`printf("HELLO = hello = Hello")`
`}`
 (:: C is case sensitive language)

C preprocessor:— It is the software which is executed before the compiler and access C program as input and processes any C preprocessor commands within it.

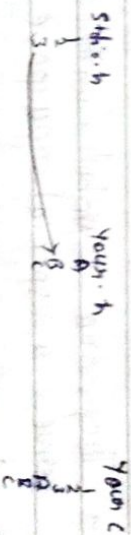


• Preprocessor directives:—

(i) File inclusion \Rightarrow This directive uses the include command to include contain of specify header file.

Header File:— This are ~~preprocessor~~ ^{written} library program function declaration and other statements

eg - `#include <stdio.h>` // standard input/output
`#include <conio.h>` // console input/output



eg - `#include <math.h>`
`#include <ctype.h>`
`#include <graphics.h>`

(iv) More substitution

Q. WAP to print Hello, your name?

Ans- #include <stdio.h>
#include <conio.h>

void main()

{
printf ("Hello, India");
}

C token:- These are the element which make up the program

(i) C character set :-

{
 alphabets a---z lower case
 A---Z upper case
 digits 0-9
 special characters , * , : , & etc
 symbols + , - , ! , < etc

(ii) Delimiters :- They are symbol with which serve a special purpose in program.

(A)

[] → It is used to represent an array.

(B) curly { } → They mark the beginning and of a block.

(C) parentheses () → They represent a function and its arguments.

(c) semicolon ; → It represents end of a statement

(d) Unary star * → It represents a pointer.

(e) dot operator . → Refers to member access.

(iii) Constants :- They are fixed value which remain same through out the program.

Integer	Real (float)	Character	String
eg- 1, 2, 3, 4	eg- 7.32, 53.4, -0.62	eg- 'a', 'A', 'Z', 'z'	eg- "Hello"

(iv) data type :- Data types define the type of data the compiler about the values.

Integer	Float	double	char	void
→ 2 Bytes	→ 4 Bytes	→ 8 Bytes	→ 1 Byte	→ Nothing
Signed: -32768 to 32768	Signed: -1.7x10 ⁻³⁸ to 1.7x10 ³⁸	Signed: 10 ⁻³⁰⁸ to 10 ³⁰⁸	Signed: -128 to 127	
Unsigned: 0 to 65536				

(v) Reserved word :- Also known as keywords are known to the compiler.

They can not be used as identifiers eg- int, float, double, char, void, if, else, for, etc.

There are the total of 32 keywords in ANSI C

(vi) Identifiers:— These are the names given by the user to their user data. Identifiers are created by us and can't be keywords.

Rules for creating identifiers:—

- 1) They are case sensitive.
 - 2) They cannot begin with a number.
 - 3) They cannot be a keyword.
 - 4) They cannot contain special characters other than underscore.
 - 5) Identifiers cannot have space b/w them and must always be a single word.
- Predefined class names can be used as identifiers.
- eg- `hello` , `prog@mx` , `Byte37`
`37ByteX`

(vii) Variables:— variables are main storage location having a type, an address and a value. They can have only a single value at a time but it can vary through out the program.

Declaring a variable:— In order to use the variable it must first be declared

Syntax:

datatype variable name ;
OR
datatype variable name = value ;

eg- `int num ;`

`num = 7 ;`
`float b = 7.68 ;`
`char x, y, z ;`
`y = 'M' ;`

Operators & Expressions ⇒

Operators:— These are the symbols which are used to perform operation based on mathematics, comparison, evaluation and deduction. operation can be unary (having one operand), binary (having two operands) and ternary (having three operands). They are category into

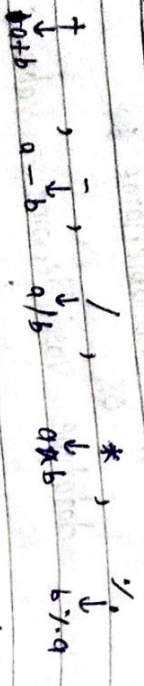
- (i) Arithmetic operators
- (ii) Relational operators
- (iii) Logical operators
- (iv) Bit wise operators
- (v) Assignment operators
- (vi) Compound operators (Short hand)
- (vii) Increment operators
- (viii) Decrement operators

Assignment operators:— This operators represent value from RHS to LHS. $\text{RHS} \rightarrow \text{LHS}$ is used to a sign

eg- `num = 7 ;`

int a, b, c;
a = 12;
b = 7;
c = 9;

2. Arithmetic operators :-



unary - $\rightarrow -b$

3. Relational operators :- These operators always evaluate to true or false

true (1) & false (0)

unary !	$c = !a$	0
<	$c = a < b$	0
>	$c = a > b$	1
<=	$c = a <= b$	0
>=	$c = a >= b$	1
=	$c = a == b$	0
!=	$c = a != b$	1

4. Logical operators :- These operators combine & evaluate

unary expression

(logical and) && $(a > b) \&\& (a > c)$ T

(logical or) || $(a > b) || (a > c)$ F

F

5. Compound operators :- These are short hand operators

which represents entire expression

$a += b$
 $a -= m$
 $a *= b$
 $a /= q$
 $a \% = d$

6. Increment operators :- These operators are used to increase/decrease

value of a variable by 1. They can be either prefix (applied before the operand) or postfix (applied after operand).

$++a$
 $--a$

prefix increment / decrement \Rightarrow

eg - $int a;$

$a = 7;$

$++a;$

$a = 8;$

eg - $a = 1;$

$--a;$

$a = 0;$

$a = 1$

eg - $int a, b;$

$a = 7;$

$b = ++a;$

$a = 8;$

eg - $a = -5$

$b = --a;$

$a = a - 1$
 $b = a$

eg-
 int a=4;
 int b;
 b=++a;
 b=-a;
 // a=5, b=5
 // a=4, b=4

Postfix:-
 int a=4;
 int b;
 b=a+1;
 b=a-1;
 a=a+1;
 a=a-1;

eg-
 int x,y,z;
 x=7;
 y=-1;

Z=++x; \rightarrow x=x+1
 x=y--; \rightarrow x=y
 y=++z; \rightarrow z=z+1
 z=-y; \rightarrow z=-y
 x=8, y=8, z=8

0. $i-2 = 1 - (-1) = 2$
 $i-2 = 2$
 $i = 3$
 $i+2 = 1+2 = 3$
 $i = 3$

1. Bitwise operators \Rightarrow These operators evaluate in binary

bitwise and & xor XOR negate ~ left shift << right shift >>

for 15 bits \Rightarrow [8421] (eg-7 \Rightarrow 0111)

Expression:- These are the statement which evaluate to something that is produce a result.

Types of expression:-

1. Arithmetic expression :- $C = a + b \times z$
2. Relational expression :- $a < b$
3. Logical expression :- $a < b \ \&\& \ a < c$
4. Bitwise expression :- $a \&b$
5. Increment, decrement :- $++a, --b$

(i) Compound expression :- $a + b$

~~Type~~ Conversion \Rightarrow Also known as type casting it is the process of converting one data type to other data type. It is of two type :-

- (i) Implicit / internal
- (ii) Explicit / external

1) Implicit :- This type of casting is done by the compiler automatically based on the type of data.

eg-
 $\text{int } a = 5;$
 $\text{int } b = 2;$
 $\text{float } c;$
 $c = a / b;$

Output
 $c = 2.0$
 $\left\{ \begin{array}{l} \text{--} \end{array} \right. c = \text{float}$
 $\frac{a}{b} = \text{int} = 2.5$
 datatype = 2

eg-
 $\text{int } a = 5;$
 $\text{float } b = 4.5;$
 $\text{float } c;$

$c = a + b;$
 Output
 $c = 9.5$
~~int a+b=9~~

(ii) Explicit :- In this process conversion from one data type to other is done manually by the user by the following syntax :-

eg-
 $\text{int } a = 5;$
 $\text{int } b = 2;$
 $\text{float } c;$
 $c = (\text{float}) a / b;$

(target datatype) expression;
 eg-
 $\text{int } a = 5;$
 $\text{float } b = 4.5;$
 $\text{float } c;$
 $c = (\text{int})(a + b);$
 $[c = 9.5]$

Operator Precedence & associativity \Rightarrow

Operator precedence :- It specifies which operator must be performed first in an expression

- (i) Brackets $() [] \{\}$
- (ii) Member access $., * \&$
- (iii) Unary operators $!, +, -, ++, --, \&, *$

(iv) Member access via pointer \rightarrow

(v) Multiplicative $*, \%, /$

(vi) Additive $+, -$

(vii) Relational $<, >, <=, >=, ==, !=$

(viii) Bitwise $\&, |, \sim$

(ix) Logical $\&\&, ||$

(x) Assignment
(xi) Screenshot
(xii) Gamma

$+ = , - = , * = , / = , \% =$

Q. $7 + 5 * 7 - 2 / 2 = 41$
 Q. $7 * 5 / 2 - 3 + 4 = 18$

Associativity \Rightarrow When ever multiple operators have same precedence then the order is determined by its associativity which can either be right to left or left to right.

Unary operators have right to left associativity. Where as binary operators have left to right.

Conditional Operator \Rightarrow [?:] This operator is used to be have like if then

else statement and provides result based on the truthfulness of condition

Syntax:

Condition? True part : false part

eg-

```
int a, b, c;
a = 5; b = 7;
c = (a > b) ? a : b;
if (a > b)
  c = a;
else
  c = b;
```

Input / Output - C library provides a variety of build in function which can be used to performed input/output operations. This function mostly define in `stdio.h` can be category into formatted function and unformatted function.

	Input	Output
Formatted	<code>scanf()</code>	<code>printf()</code>
unformatted	<code>getc()</code> <code>getch()</code> <code>gets()</code>	<code>putc()</code> <code>putch()</code> <code>puts()</code>

Formatted Function \Rightarrow These functions describe the input output operation to data. In addition to way the information is formatted or appears.

Unformatted Function \Rightarrow These functions describe the input/output operation to data.

Formatted output function :- C provides a function in which `printf` is mostly used. `printf()` function is the formatted output function which is used to output the argument passed to the standard output device.

Syntax: `printf("output string");`

and `printf("control specifier", variable);`

eg- `printf("Hello");` →

output
Hello

Format specify / conversion specify ⇒

Also known as control / time / conversion specifier. Provides a place holder to specify value of the corresponding variable.

They begin with % and a character each representing a specific data type.

- d, i - signed integer
- u - unsigned integer
- c - character
- f - float
- e, E - float in exponential form
- s - string
- p - pointer value

eg- `int a;`

`a=5;`

`printf("%d", a);`

`printf("%s");`

Each specifier acts as a substitution for exactly for a variable.

eg- `int a;`

`float b;`

`char c;`

`a=10; b=7.5; c='z';`

`printf("%d %f %c", a, b, c);`

output
10 7.5 z

Escape sequences:- Use primary in printf have a special and are used to print characters that are not directly available. They always begin with a back slash (\).

- (i) \a - alert
- (ii) \t - tab
- (iii) \n - newline
- (iv) \r - return to first column
- (v) \b - return to first column previous line
- (vi) \f - vertical tab

eg- `int a;` , `float b;` , `char c;`

`printf("%d\n\t%f", a, b, c);`

output
10
7.5 z

eg- a.

Write a program to perform math operation

on `a=10` , `b=7`

Ans- `#include <stdio.h>`

`#include <conio.h>`

`void main()`

{


```

int a, b, ad, su, ml, di, mo;
ad = a + b;
su = a - b;
ml = a * b;
di = a / b;
mo = a % b;
printf ("%.d + %.d = %.d\n", a, b, ad);
printf ("%.d - %.d = %.d\n", a, b, su);
printf ("%.d * %.d = %.d\n", a, b, ml);
printf ("%.d / %.d = %.d\n", a, b, di);
printf ("%.d % %.d = %.d\n", a, b, mo);
getch();
}

```

Output

```

10 + 7 = 17
10 - 7 = 3
10 * 7 = 70
10 / 7 = 1
10 % 7 = 3

```

Q. Write a program to calculate a simple interest.

```

// include <stdio.h>
// include <conio.h>
void main()
{
    int p, n, t, si;
    p = 1.5;
    n = 2.5;
    t = 0.5;
    si = (p * n * t) / 100;
}

```

```

printf ("%.f * %.f * %.f / 100 = %.f", p, n, t, si);
getch();
}

```

Output

$$\frac{1.5 * 2.5 * 0.5}{100} = 0.01875$$

Q. Write a program to calculate area & perimeter of rectangular.

```

// include <stdio.h>
// include <conio.h>
void main()
{

```

```

    int l, w, area, pa;
    l = 5;
    w = 2;
    area = l * w;
    pa = 2 * (l + w);

```

```

printf ("%.d\n %.d\n %.d\n", l, w, area);
printf ("%.d\n %.d\n", l, w, pa);
getch();
}

```

Output

```

5
2
10
14

```


input function
`scanf()` \Rightarrow It is a formatted input function which is used to accept information from the standard input device.

Syntax
`scanf ("control specifier", variable address);`

eg-
`int a; float b; char c;`

`scanf ("%f%f%c", &b, &c, &a);`

Q. WAP to add two integers?

Ans-
`#include <stdio.h>`

`void main()`

`{`

`int n1, n2, s;`

`printf ("Enter two numbers");`

`scanf ("%d%d", &n1, &n2);`

`s = n1 + n2;`

`printf ("sum is %d", s);`

`}`

Q. WAP to calculate area of triangle.

Ans-

`#include <stdio.h>`

`#include <conio.h>`

`void main()`

`{`

`int b, h, ar;`

`printf ("Enter two numbers");`
`scanf ("%d%d", &b, &h);`
 $ar = \frac{1}{2} * b * h;$

`printf ("The area is %f", ar);`
`getch();`

Q. WAP to calculate determinate.

Ans-
`#include <stdio.h>`

`#include <conio.h>`

`void main()`

`{`

`int b, a, c, de;`

`printf ("Enter three numbers");`

`scanf ("%d%d%d", &b, &a, &c);`

$de = b^2 - 4ac;$

`printf ("The determinate is %d", de);`

`getch();`

`}`

Q. WAP to swap two number.

Ans-
`#include <stdio.h>`

`#include <conio.h>`

`void main()`

`{`

`int a, b, e;`

`e = a;`

`a = b;`

`b = e;`

`printf ("Enter two number");`


```
printf ("Enter two numbers\n a=%d, b=%d", a, b);
scanf ("%d %d", &a, &b);
```

```
e = a;
a = b;
```

```
b = e;
printf ("a=%d, b=%d", a, b);
getch();
```

13) Without third variable

```
# include <stdio.h>
# include <conio.h>
void main()
```

```
{
    int a, b;
    printf ("Enter any two numbers");
    scanf ("%d %d", &a, &b);
    b = a + b;
    a = b - a;
```

```
b = b - a;
printf ("a=%d, b=%d", a, b);
```

```
getch();
}
```

Unformatted function :-

```
getc() - Input a character
```

```
getchar() - " " as echo
```

```
getche() - "no echo
```

```
getch() - "with echo
```

```
get() - "String
```

```
putc() - o/p a character
putchar() - " "
puts() - o/p a string
```

```
eg- void main()
```

```
{
    char c, f;
    f = getc();
    putc(f);
    putchar(f);
```

getc() & getche() :- These are the input

used to accept a single character. The difference is that getche() also echo the input where as getch doesn't.

puts() and putchar() :- These function are used to input and display a

eg- string.

```
{ void main()
```

```
{ char *c;
    getc();
    putc(c);
```

```
}
```


Macro Substitution

Macro :- These are symbolic constant which are implemented using define command and provide a name for a value or expression.
It is of two types:-

- 1) Macro without argument.
- 2) Macro with ~~out~~ argument.
- 3) Macro without argument:-

Syntax:-

```
# define      macro name      value
# define      PI      3.14
# define      MAX      100
# define      const      0.55
```

(ii) Macro with argument:-

Syntax:- # define macro name(x) expression

eg:-

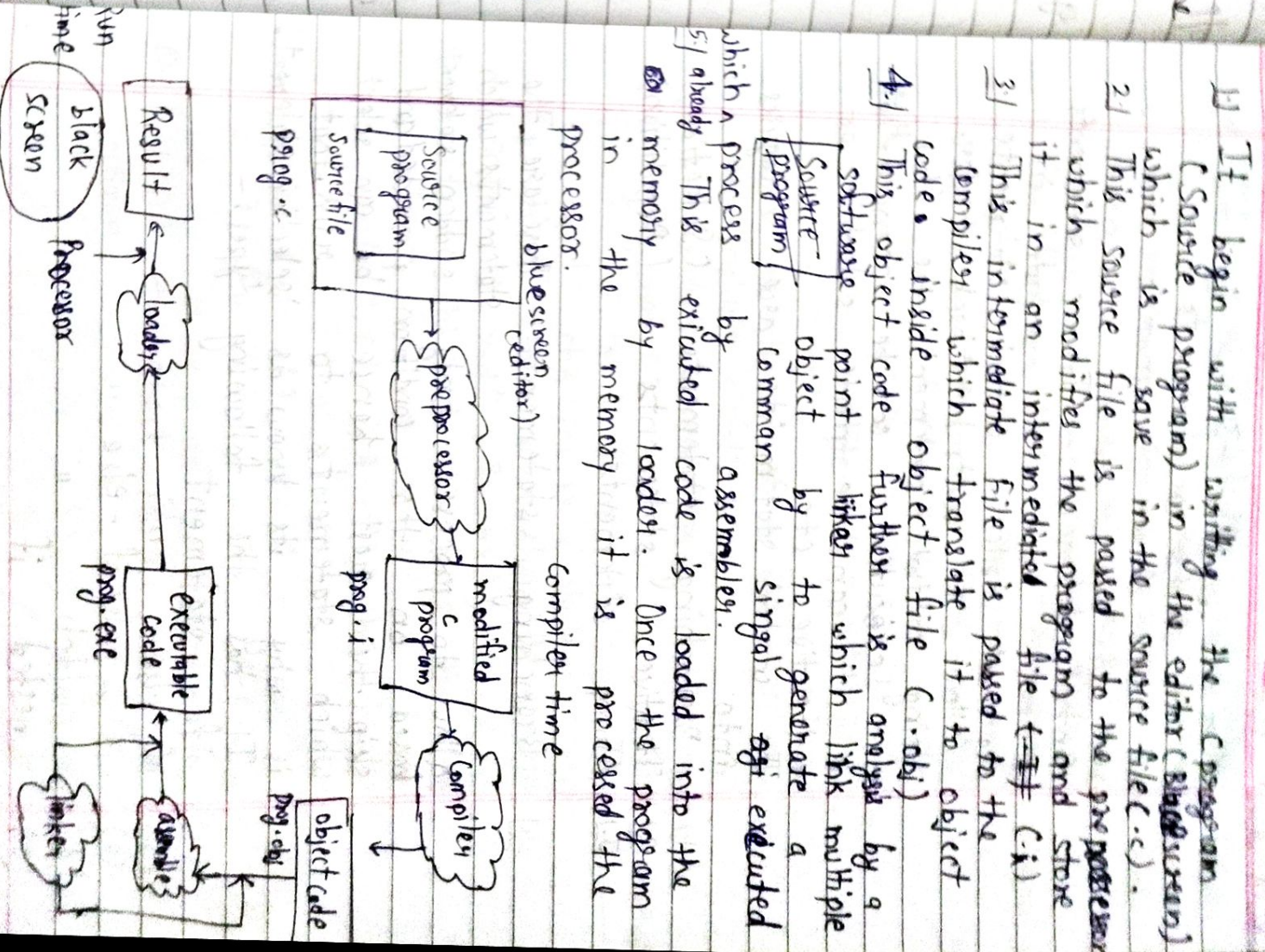
```
# define      sqn(x)      x * x
# define      peri(a,b)      2 * (a+b)
```

eg:-

```
printf("%d", sqn(7));
p = peri(7,5);  $\Rightarrow$  p = 2 * (7+5);
```

Program life cycle \Rightarrow Program life cycle is

the process of processing the program and the step involved until its program.



Unit-3

Control statements Array & function

Control statements:- In C statements are executed in sequential order. Control statements allow to control that back order.

Using control statement the flow of execution can be changed that is it can be made selective or iterative etc.

The control statements are categories into

1. Branching statements (Selection)
2. Iterative statements (Repetition)
3. Jump statements

1. Branching statement:- These are the statements which

allow to process only sudden statement based on the condition provided and skip the rest hence we can select which statements to run that is why its known as 'selection statement'. It has the following types:-

- (a) if statement
- (b) if else "
- (c) if - else if - else "
- (d) switch "
- (e) nested if

(a) If statements => It is a simple branch the statements specify in its block only when the condition satisfy true.

-> Condition must always evaluate to either true (1) or false (0).
Syntax:-

```
if (condition)
{
    statements;
}
```

Q. WAP to check if a value is positive.

```
#include <stdio.h>
void main()
{
    int chk;
    printf ("Enter a number");
    scanf ("%d", &chk);
    if (chk > 0)
    {
        printf (" %d is +ve", chk);
    }
    printf (&getch());
}
```

Output
37 < -37 < 37 is +ve

Q. WAP to check if a person is senior citizen.

Q. -
include <stdio.h>
include <conio.h>
void main()
{

int a;
printf ("Enter any number");
scanf ("%d", &a);
if (a > 65)

{
printf (" %d is a senior citizen", a);
}

getch();
}

Q. WAP to a if a no. exist b/w 12 & 18.

include <stdio.h>
include <conio.h>
void main()
{

int a;
printf ("Enter any number");
scanf ("%d", &a);
if (a > 12 && a < 18)

{
printf (" %d is exist", a);
}

getch();
}

2.1 If else statement \Rightarrow

Syntax:-

(True)

if (condition)
{
Statement;
}

else

(false)

{
Statement;
}

Q. -

WAP to check if a number is even or odd.
include <stdio.h>
include <conio.h>
void main()
{

int n;
printf ("Enter any number");
scanf ("%d", &n);
if (n % 2 == 0)

{
printf (" %d is even", n);
}

else

{
printf (" %d is odd", n);
}

getch();
}

3.1 If else if else statement \Rightarrow These statements are incremented when they are more than 2 options to choose from.

syntax: if (condition)

```
{
    statement;
}
```

```
else if (condition)
{
}
```

```
opt-② {
    statement;
}
```

```
else {
    statement;
}
```

Q. WAP to check if a number is positive, -ve or zero.

Ans- #include <stdio.h>

#include <conio.h>

void main()

{

int n;

printf ("Enter any number");

scanf ("%d", &n);

if (n > 0)

{

printf ("%d is +ve");

}

```
else if (n < 0)
```

```
{
```

```
printf ("%d is -ve");
```

```
}
```

```
else
```

```
{
    printf ("%d is zero");
}
```

```
getch();
}
```

Output
Enter any number 2
n is +ve

Q. WAP to print grade of the student based on percentage.

per > 85

A+

70 < per <= 85

A

55 < per <= 70

B

per <= 55

F

Ans-

#include <stdio.h>

#include <conio.h>

void main()

{

int per;

printf ("Enter any number");

scanf ("%d", &per);

if (per > 85)

{

printf ("student will gain A+");

}

else if (per > 70) && (per <= 80)

{

printf ("A");

}


```

3 else if (1 per > 55) && (per <= 70)
3
3 printf ("B");
3
3 else
3
3 printf ("F");
3
3 getch();
3
    
```



Q. WAP to calculate roots of a quadratic eqⁿ.

Ans -

```

# include <stdio.h>
# include <conio.h>
# include <math.h>
void main()
{
    int a, b, c, d, x1, x2;
    printf ("Enter any three number\n");
    scanf ("%d %d %d", &a, &b, &c);
    d = b*b - 4*a*c;
    if (d > 0)
        printf ("roots are real and distinct");
    else if (d < 0)
        printf ("roots are imaginary");
    }
    
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

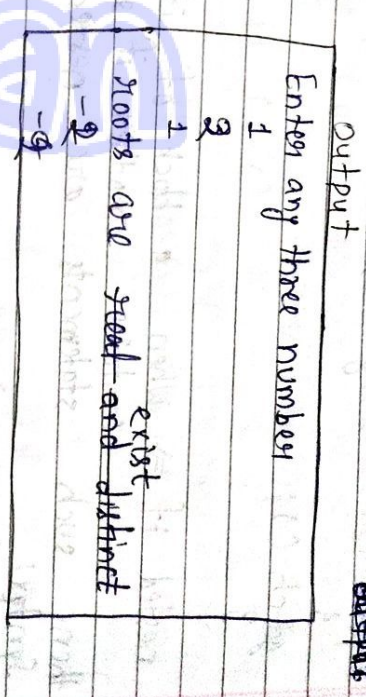
$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

```

if (d > 0)
    printf ("roots are real and distinct");
else if (d < 0)
    printf ("roots are imaginary");
    
```

```

3 else
3
3 printf ("roots are exist");
3
3 printf ("%d %d %d", x1, x2);
3
3 getch();
3
    
```



Q. WAP to input sides of a triangle and check if it is isosceles, equilateral and scalene.

Ans -

```

# include <stdio.h>
# include <conio.h>
void main()
{
    int a, b, c;
    printf ("Enter any three number\n");
    scanf ("%d %d %d", &a, &b, &c);
    if (a == b && b == c && c == a)
        printf ("triangle is equilateral");
    }
    
```

$$a = b = c$$

$$a = 2, b = 2, c = 2$$

```

if (a == b && b == c && c == a)
    printf ("triangle is equilateral");
    
```



```

if (a==b || b==c || c==a)
{
    printf("triangle is isosceles");
}
else
{
    printf("triangle is scalene");
}
getch();

```

4. nested if \Rightarrow when a block of if statement contains another if statement then such statements are nested if.

Syntax:

level-0 if (condition1)

level-1 if (condition2)

if

else

else

Q. WAP to find minimum among 3 numbers using nested if.

Ans- #include <stdio.h>
#include <conio.h>
void main()

```

{
    int a, b, c;
    printf("Enter any three numbers\n");
    scanf("%d %d %d", &a, &b, &c);
    if (a < b)
    {

```

```

        if (a < c)
        {
            printf("%d is minimum", a);
        }
        else
        {
            printf("%d is minimum", b);
        }
    }
}

```

printf("%d is minimum", a);

else if (a < b < a)

printf("%d is minimum", b);

printf("%d is minimum", b)

else

printf("%d is minimum", c)

getch();

5.1

Switch statements \Rightarrow It is a branching statement which is used to select one of the options (case) depending upon the assignment provided. This assignment can only be either an integer or a character.

Syntax :-
switch (variable)

```
{
    case 1 : stmt;
              break;
    case 2 : stmt;
              break;
    default : stmt;
}
```

The difference b/w else if and switch statement is that else if statement can be used to select a choice from a given range of value where as switch statement provides fixed option from which one can be selected.

Q. WAP to print day of week depending upon user input.

Ans -

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int w;
```

```
printf ("Enter day number\n");
scanf ("%d", &w);
switch (w)
```

```
{
    case 1 : printf ("Monday");
              break;
```

```
    case 2 : printf ("Tuesday");
              break;
```

```
    case 3 : printf ("Wednesday");
              break;
```

```
    case 7 : printf ("Sunday");
              break;
    default : printf ("invalid day no");
}
```

```
getch();
}
```

Break statement \Rightarrow It is a control statement which is used to forcefully exit out of a switch block or loop block.

Q. WAP to create arithmetic calculator using switch

Ans -

```
Case:
#include <stdio.h>
#include <conio.h>
void main ()
{
```

```
    char op;
    int a, b, c, d;
```



```

printf ("Enter any a");
printf ("1. Add\n 2. sub\n 3. mult\n 4. div\n 5. mod\n");
printf ("Enter any operator");
scanf ("%d", &d);
printf ("Enter any two number\n");
scanf ("%d %d", &a, &b);
switch (d)
{
case 1:
    c = a + b;
    printf ("%d", c);
    break;
case 2:
    c = a - b;
    printf ("%d", c);
    break;
case 3:
    c = a * b;
    printf ("%d", c);
    break;
case 4:
    c = a / b;
    printf ("%d", c);
    break;
case 5:
    c = a % b;
    printf ("%d", c);
    break;
default:
    printf ("invalid choice");
    getch();
}

```

Output
1. Add
2. Sub
3. Mult
4. div
5. Mod
Enter any operator
Enter any two no.
5
4
20

Q. WAP to ~~check~~ check if a character is consonant or a vowel.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    char v;
    printf ("Enter any character alphabet\n");
    scanf ("%c", &v);
    switch (v)
    {
case 'a':
    printf ("%c is vowel", v);
    break;
case 'e':
    printf ("%c is vowel", v);
    break;
case 'i':
    printf ("%c is vowel", v);
    break;
case 'o':
    printf ("%c is vowel", v);
    break;
case 'u':
    printf ("%c is vowel", v);
    break;
default:
    printf ("%c is not vowel", v);
    getch();
}
}

```

Output
Enter any character alphabet
z
z is not vowel

2.1 Iterative statements \Rightarrow These are the looping statements which

are used to repeat a given set of statements a fixed no. of times. They are implemented by

- (i) for
- (ii) while
- (iii) do while

III 'for' loop:— It is a simple loop which repeats a certain number of statement fixed no. of time.

Syntax: $\begin{array}{l} \text{for (initialization; condition; iteration)} \\ \{ \\ \text{statements;} \\ \} \end{array}$

(a) Initialization \Rightarrow This step specify the beginning of the loop or from where the loop starts.

(b) Condition \Rightarrow This step specifies the condition ~~certain~~ will exist until ~~the~~ loop will run

(c) Iteration \Rightarrow This step specifies the movement of the loop from start to finish.

Note

To work with the loop be need to ~~interment~~ implement the loop variable

Q. WAP to print counting 1 to 50.

Ans- $\begin{array}{l} \# \text{ include <stdio.h>} \\ \# \text{ include <conio.h>} \\ \text{void main()} \\ \{ \end{array}$

$\begin{array}{l} \text{int i;} \\ \text{for (i = 1 ; i <= 50 ; i = i + 1)} \\ \{ \\ \text{printf ("%d\n", i);} \\ \} \end{array}$

Q. WAP to print odd no. from 50 to 500

Ans- $\begin{array}{l} \# \text{ include <stdio.h>} \\ \# \text{ include <conio.h>} \\ \text{void main()} \\ \{ \end{array}$

$\begin{array}{l} \text{int i;} \\ \text{printf ("Enter odd no. 50 to 500\n");} \\ \text{for (i = 51 ; i <= 500 ; i = i + 2)} \\ \{ \\ \text{printf ("%d\n", i);} \\ \} \\ \text{getch();} \\ \} \end{array}$

Q. WAP to find all the even & odd no. b/w 1 to 100.

Ans- # include <stdio.h>
include <conio.h>
void main()
{

int i;
if (i%2==0)

for (i=1; i<=100; i=i+2)
{ printf ("%d", i);
printf (" even");
}

else

for (i=1; i<=100; i=i+2)
{ printf ("%d", i);
printf (" odd");
}

getch();
}

Q. WAP to print multiplication table of 7.

Ans- # include <stdio.h>
include <conio.h>
void main()
{
int i;
for (i=1; i<=7; i=i+1)

printf ("%d", i);
}

getch();
}

int i, n;
scanf ("%d", &n);
for (i=1; i<=n; i++)

printf ("%d", n*i);
getch();
}

2.] 'while' loop \Rightarrow It is ^{entry} control loop which

is used to iterate a block of statements certain number of time.

Although It can also be used to repeat statements to infinity in a control manner.

Syntax:-

initialization;
while (condition)
{

statements;
iteration;
}

Q. WAP to print even numbers from 100 to 0.

Ans- # include <stdio.h>
include <conio.h>
void main()


```

3
{
  int i;
  i = 100;
  while (i >= 1)
  {
    printf("%d\n", i);
    i = i - 2;
  }
  getch();
}

```

Output

```

100
98
...

```

```

3
{
  getch();
}

```

Q. WAP to input & print numbers until 0 is pressed.

Ans-
#include <stdio.h>
#include <conio.h>
void main()

```

{
  int i;
  printf("Enter any number\n");
  scanf("%d", &i);
  while (i != 0)
  {
    printf("%d", i);
    scanf("%d", &i);
  }
}

```

Output

```

17 17
23 23
0

```

31 'do while' loop \Rightarrow It is a exit control loop a certain no. of statements.

Syntax:-

```

do
{
  initialization;
}

```

```

while (condition);

```

Although while & do while perform the same, the major difference is while is an 'entry control' loop where as do while is 'exit control' loop which means in while condition is checked before entering the block, where as in do while the condition is checked after the do while block is executed.

\rightarrow This process causes the do while block to be executed at least one and the while block to be executed zero times.

Q. WAP to print counting from 5 to 50.

Ans-
#include <stdio.h>
#include <conio.h>
void main()
{
 int i;


```

l=5;
do
{
printf ("%d\n", l);
l=l+1;
} while (l<=50);
getch();
}

```

Q. WAP to find the sum of first 50 natural numbers.

Ans-

```

#include <stdio.h>
#include <conio.h>

void main()
{

```

```

int l, s=0;
l=1;
do
{
printf ("%d\n", s);
s=s+l;
l=l+1;
} while (l<=50);
getch();
}

```

(3.) Jump statement (other statement) \Rightarrow

(a) Break statement :- It terminates in active loop or a switch statement.

eg-

```

int c;
for (c=0; c<=10; c=c+2)
{
if (c==6)
{ break; }
else
{ printf ("%d", c);
}
}

```

output

0
2
4

(b) Continue :- This statements when executed skips the following statements and moves to be next iteration.

eg-

```

int c;
for (c=0; c<=10; c=c+2)
{
if (c==6)
{ continue; }
printf ("%d", c);
}

```

output

0
2
4
8
10

Q. WAP to print odd no. from 1 to 100.

Ans-

```

#include <stdio.h>
#include <conio.h>

void main()
{
int c;
for (c=1; c<=100; c=c+1)
{
if (c%2 == 0)

```



```

} continue; }
printf ("%d", c);
}

```

(c) go to statement \Rightarrow It is a control transfer statement which makes the control jump to specify label. A label is an identifier provided by the user.

\rightarrow When the control transfers to the label it is a forward jump otherwise it is a backward jump.

Backward

```

label:
if (condition)
goto label;

```

Q. WAP to find sum of n numbers (until zero is pressed)

Ans - #include <stdio.h>
#include <conio.h>

```

void main()
{

```

```

    int s=0;
    int n;

```

```

    printf ("Enter a no.");
    scanf ("%d", &n);
    s = s + n;

```

```

printf ("Press number to continue, 0 to exist");
scanf ("%d", &n);
if (n != 0)
    go to start;
printf ("%d", s);
}
getch();

```

~~★~~ **nested loop** \Rightarrow When a loop repeats occurs inside another loop it is a nested loop

● **nested for** :- When for repeats inside another for. The loop which is outside is the outer loop & the loop which is inside the other loop is the inner loop. For each value of outer loop the inner loop complete itself for example if outer loop goes from 1 to 10 & inner loop goes from 1 to 7 then for each value from 1 to 10 the inner loop runs 7 times combining the total run to $10 \times 7 = 70$ steps.

Eg -

```

int n, c;
row  $\rightarrow$  for (r=0; r<10; r++)
{
    coloum  $\rightarrow$  for (c=0; c<10; c++)
    {
        printf ("%d", c);
        //
    }
}

```


Q. WAP to print the following pattern

Ans-

```
#include <stdio.h>
#include <conio.h>
void main()
{
```

```
    int n, c;
```

```
    for (n=1; n<=4; n++)
```

```
    { printf("\n");
```

```
    for (c=1; c<=4; c++)
```

```
    { printf("*");
```

```
    }
```

```
}
```

```
    getch();
}
```

Q. WAP to print a following pattern

```
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
```

Ans-

```
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
```

```
    int n, c;
```

```
    for (n=1; n<=4; n++)
```

```
    { printf("\n");
```

```
    for (c=1; c<=4; c++)
```

```
    {
```

```
        printf("%d", n);
```

```
    }
```

```
    getch();
```

```
}
```

09

```
int n, c;
```

```
int m=1;
```

```
{ printf("%d", m);
```

```
    m++;
```

```
}
```

Q. WAP to print following pattern:-

```
1 2 3 4
5 6 7 8
9 10 11 12
```

Ans-

```
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
```

```
    int n, c;
```

```
    int m=1;
```

```
    for (n=1; n<=3; n++)
```

```
    { printf("\n");
```

```
    for (c=1; c<=4; c++)
```

```
    {
```

```
        printf("%d", m);
```

```
        m++;
```

```
    }
```

```
    getch();
```


Q. WAP to print following pattern

```

1 3 5 7
2 11 13 15
17 19 21 23
25 27 29 31

```

```

//
#include <stdio.h>
#include <conio.h>
void main()
{

```

```

    int n, c;
    int m=1;
    for (n=1; n<=4; n++)
    {
        printf (" %d", m);
        for (c=1; c<=4; c++)
        {
            printf (" %d", m);
            m=m+2;
        }

```

```

    }
    getch();
}

```

Q. WAP to print the following pattern

```

* * *
* * *
* * *

```

```

//
#include <stdio.h>
#include <conio.h>
void main()
{

```

```

    int n, c;
    for (n=1; n<=4; n++)
    {
        printf (" %d", m);
        for (c=1; c<=4; c++)
        {
            printf (" *");
        }

```

```

    }
    getch();
}

```

Q. WAP to print the following pattern

```

1
2 3
4 5 6
7 8 9 10

```

```

//
#include <stdio.h>
#include <conio.h>
void main()
{

```

```

    int n, c;
    int m=1;
    for (n=1; n<=4; n++)
    {
        printf (" %d", m);
        for (c=1; c<=n; c++)
        {
            printf (" %d", m);
            m++;
        }

```

```

    }
    getch();
}

```


Q. WAP to print following pattern

```

1
2 2
3 3 3
4 4 4 4

```

Ans-
#include <stdio.h>
#include <conio.h>
void main()
{

```

    int n, c;  
    for (n=1; n<=4; n++)  
    {  
        for (c=1; c<=n; c++)  
        {  
            printf("%d", n);  
        }  
        printf("\n");  
    }  
    getch();  
}
```

Q. WAP to print following pattern

```

1
2 2
3 3 3
4 4 4 4

```

Ans-
#include <stdio.h>
#include <conio.h>
void main()
{

```

    int n, c;  
    for (n=1; n<=4; n++)  
    {  
        for (c=1; c<=n; c++)  
        {  
            printf("%d", n);  
        }  
        printf("\n");  
    }  
    getch();  
}
```

Arrays \Rightarrow These are a derived datatype where multiple values of same type are stored under a single name.

It is basically collection of homogeneous element stored in a sequential & continuous memory location.

Type of arrays:-

1. 1-Dimensional 2. Multi-Dimensional

1. 1-Dimensional \Rightarrow It is a array variant which stores a single row, multiple columns.

Declaration of 1-D \Rightarrow

Syntax:-

eg- int marks[50]; // integer array
data type variable name[size];


```
float per[100]; // float array
char name[10]; // char array
```

When declaring an array the brackets specify the size of array where as during the use through the entire program the brackets specify index.

Index is the location of a particular element in an array. It begins with zero & goes upto size - 1.

Q. WAP to declare & initialize a 10th element integer array

Ans-

```
int a[10];
a[0] = 16;
a[1] = 47;
a[2] = 56;
a[3] = 64;
a[4] = 77;
a[5] = 88;
a[6] = 99;
a[7] = 100;
a[8] = 101;
a[9] = 102;
```

Q. WAP to input & print 10th value of an array

Ans-

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10];
```

```
int i;
for (i=0; i<9; i++)
```

```
{
    printf ("Enter element %d", i);
    scanf ("%d", &a[i]);
}
```

```
for (i=0; i<9; i++)
```

```
{
    printf ("%d\n", a[i]);
}
```

```
getch();
```

output
enter element 0
1
18

Q. WAP to input 100 integers & print in reverse order.

Ans-

```
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
    int a[100];
```

```
int i;
for (i=0; i<99; i++)
```

```
{
    printf ("Enter element %d", i);
    scanf ("%d", &a[i]);
}
```

```
for (i=99; i>0; i--)
```

```
{
    printf ("%d\n", a[i]);
}
```

```
getch();
```


Q. WAP to input an Array of 10 element & print their sum.

```

Ans- #include <stdio.h>
#include <conio.h>
void main()
{
    int a[10];
    int i, s=0;
    for (i=0; i<=9; i++)
    {
        printf("Enter element %d\n", i);
        scanf("%d", &a[i]);
    }
    for (i=0; i<=9; i++)
    {
        printf("%d", s);
        s = s + a[i];
    }
    printf("%d", s);
    getch();
}

```

* Q. WAP to find maximum in array of 20 element.

```

Ans- #include <stdio.h>
#include <conio.h>
void main()
{
    int a[20];
    int i;
    int max=0;

```

```

    for (i=0; i<=19; i++)
    {
        scanf("%d", &a[i]);
    }

```

```

    for (i=0; i<=19; i++)
    {
        if (max < a[i])
        {
            max = a[i];
        }
    }
    printf("The maximum is %d", max);
    getch();
}

```

* Q. WAP to find smallest & largest value in array of 100 elements.

```

Ans- #include <stdio.h>
#include <conio.h>
void main()
{
    int a[100];
    int i;
    int max=0;
    int min=32767;
    for (i=0; i<=99; i++)
    {
        scanf("%d", &a[i]);
    }
    for (i=0; i<=99; i++)
    {
        if (max < a[i])

```



```

{
    max = a[i];
}
}
for else printf("The maximum is %d", max);
for (i=0; i<9; i++)
{
    if (min > a[i])
    {
        min = a[i];
    }
    printf("The minimum is %d", min);
    getch();
}

```

Q. WAP to input an element of array & print all the even numbers.

```

#include <stdio.h>
void main()
{

```

```

    int a[1000];
    int n;
    int i;
    printf("Enter size of array\n");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
}

```

```

for (i=0; i<n; i++)
{
    if (a[i] % 2 == 0)
    {
        printf("number is even");
    }
    getch();
}

```

Q. WAP to arrange 10 element of an array in ascending (sorting) order.

```

#include <stdio.h>
void main()
{

```

```

    int a[10];
    int i, j;
    printf("Enter elements\n");
    for (i=0; i<9; i++)
    {
        scanf("%d", &a[i]);
    }
    for (i=0; i<9; i++)
    {
        for (j=i+1; j<9; j++)
        {
            if (a[j] < a[i+1])
            {
                e = a[j];
                a[j] = a[i+1];
            }
        }
    }
}

```


$a[i+1] = e;$

```
3 }  
3 {  
3     for (i=0; i<9; i++)  
3         printf ("%d", a[i]);  
3 getch();  
3 }
```

2D multi Dimensional Array \Rightarrow

ex - include, 2-D, 3D array.

Multi dimensional arrays contain more than one row within a table and as the dimension the no. of D also increase

2-D array: - when a array is more than row & multiple column it is 2-D array.

Syntax: - $(++i \dots ++j)$

data type variable name $[row\ size][column\ size];$

eg -

```
int table [3][4];  
float mat [5][5];  
char courtaia [5][7];
```

2-D array a generally represented in terms of matrix -

Q. WAP to initialize 2D integer array.

Ans -

```
int a[10][10] = { {1,2,3,3, 2,4,5,6,3, 1,15,4,1,3, 2,14,17,1,2,3, 3, 0, 0}}
```

```
a[0][0] = 1;  
a[3][1] = 17;  
a[2][2] = 9;
```

Q. WAP to input & print a 5x4 integer array.

Ans -
#include <stdio.h>
#include <conio.h>
void main ()

```
2  
int a[5][4];  
int x, c;  
for (x=0; x<5; x++)
```

```
2  
for (c=0; c<4; c++)
```

```
2  
3 scanf ("%d", &a[x][c]);
```

```
3  
for (x=0; x<5; x++)
```

```
3  
for (c=0; c<4; c++)
```

```
3  
3 printf ("%d", a[x][c]);
```

```
3  
3 getch();  
3 }
```


Q. WAP to print an identity matrix

Ans-

```
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
    int a[4][4];
```

```
    int n, c;
```

```
    for (n=0; n<4; n++)
        printf("\n");
```

```
    for (c=0; c<4; c++)
```

```
    {
        if (n==c)
```

```
        {
```

```
            a[n][c] = 1;
```

```
        }
```

```
        else
```

```
        {
```

```
            a[n][c] = 0;
```

```
        }
```

```
    }
```

```
}
```

```
for (n=0; n<4; n++)
```

```
{
```

```
    printf("\n");
```

```
    for (c=0; c<4; c++)
```

```
    {
        printf("%d", a[n][c]);
```

```
    }
```

```
}
```

```
getch();
```

```
}
```

(i) Matrix operations

(1) Addition

add[n][c] = a[n][c] + b[n][c];

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

Q. WAP to add two matrix.

Ans-

```
#include <stdio.h>
#include <conio.h>
```

```
void main()
```

```
{
```

```
    int a[3][3], b[3][3], add[3][3];
```

```
    int n, c;
```

```
    for (n=0; n<3; n++)
```

```
    {
        printf("\n");
```

```
        for (c=0; c<3; c++)
```

```
        {
```

```
            scanf("%d", &a[n][c]);
```

```
        }
```

```
    }
```

```
    for (n=0; n<3; n++)
```

```
    {
```

```
        printf("\n");
```

```
        for (c=0; c<3; c++)
```

```
        {
            scanf("%d", &b[n][c]);
```

```
        }
```

```
    }
```

```
    for (n=0; n<3; n++)
```

```
    {
```

```
        printf("\n");
```

```
        for (c=0; c<3; c++)
```

```
        {
```



```

3 add[a][c] = a[n][c] + b[n][c];
3
3 for (x=0; x<3; x++)
3 {
3     for (c=0; c<3; c++)
3     {
3         printf("%d", add[a][c]);
3     }
3 }
3 getch();
3

```

Q8. ~~Ans-~~

WAP to multiple 2 matrix

```

# include <stdio.h>
# include <conio.h>
void main()
{

```

```

    int a[2][2], b[2][2], mult[2][2];
    int s, n, c;
    for (n=0; n<1; n++)
    {
        for (c=0; c<1; c++)
        {

```

```

            scanf("%d", &a[n][c]);
        }
    }

```

```

    for (n=0; n<1; n++)
    {
        for (c=0; c<1; c++)
        {

```

```

        scanf("%d", &b[n][c]);
    }
    for (n=0; n<1; n++)
    {
        for (c=0; c<1; c++)
        {
            mult[n][c] = 0;
            for (s=0; s<1; s++)
            {
                mult[n][c] = mult[n][c] + (a[n][s] * b[s][c]);
            }
        }
    }

```

```

    for (n=0; n<1; n++)
    {
        for (c=0; c<1; c++)
        {
            printf("%d", mult[n][c]);
        }
    }
    getch();
}

```

Functions:- These are self contain block of statements under a name and perform a certain task. By simplifying function the code is organised better and can be re-use.

TYPES:-

- 1) library function
- 2) user defined function

1) Library function \Rightarrow These are build in function which are already defined in library files. And can be accessed through header file.

eg- printf, scanf, sqrt etc.

2) User defined function \Rightarrow These are the functions created and written by the user. To implement user define function we follow these steps:-

- (i) Function declaration
- (ii) Function definition
- (iii) Function call

(i) Function declaration \Rightarrow It is the process of declaring function a proto type.

Syntax:-
returntype function name (arguments);

(ii) Function definition \Rightarrow This step defines the block of statements of the function.

Syntax:

returntype function name (arguments)
{

// block of statements;
}

3) Function call \Rightarrow To execute the function it must be called. Function calling is most done inside main but it can happen inside other function as well.

Syntax:-
function name (parameters);

Arguments - parameters \Rightarrow These are the values the function. It can be many. which we give to

Return value \Rightarrow This value is provided by the function to us. It can only be one.

Category for implementing function \Rightarrow

- (i) no return; no argument
- (ii) with return; no argument
- (iii) no return; with argument
- (iv) with return; with argument

(i) no return; no argument \Rightarrow In this case function takes no parameters and returns no value

Q. WAP to create a function to add two no.

Ans-
include <stdio.h>
include <conio.h>


```
void add ();
void main ()
```

```
{
    clrscr ();
```

```
    add ();
```

```
    getch ();
}
```

```
void add ()
```

```
{
    int a, b, c;
```

```
    printf ("Enter two no.");
    scanf ("%d%d", &a, &b);
```

```
    c = a + b;
    printf ("%d", c);
}
```

Q. WAP to perform arithmetic operation using function.

Ans -

```
#include <stdio.h>
void main ()
```

```
{
    void add ();
```

```
    void sub ();
```

```
    void mult ();
```

```
    void div ();
}
```

```
clrscr ();
add ();
sub ();
mult ();
getch ();
}
```

```
void add ()
```

```
{
    int a, b, c;
```

```
    scanf ("%d%d", &a, &b);
    c = a + b;
```

```
    printf ("%d", c);
}
```

```
void sub ()
```

```
{
    int d, e, f;
```

```
    scanf ("%d%d", &d, &e);
    f = d - e;
```

```
    printf ("%d", f);
}
```

```
void mult ()
```

```
{
    int g, h, i;
```

```
    scanf ("%d%d", &g, &h);
    i = g * h;
```

```
    printf ("%d", i);
}
```

```
void div ()
```

```
{
    int j, k, l;
```

```
    scanf ("%d%d", &j, &k);
    l = j / k;
```

```
    printf ("%d", l);
}
```


(ii) with return ; no argument:— function in the case its no argument but return the value.

Syntax:
 returntype functionname ();

Q. WAP to find biggest no. b/w 2 number.

Ans-
#include <stdio.h>
#include <conio.h>
int biggest ();
void main ()

```

{
    int z;
    z = biggest ();
    printf ("%d", z);
    getch ();
}
int biggest ()
{
    int a, b;
    scanf ("%d%d", &a, &b);
    if (a > b)
        return a;
    else
        return b;
}

```

In this case

(iii) no return & with argument → function accept ~~but~~ argument

Q. WAP to find biggest no. b/w 2 number.

Ans-
#include <stdio.h>
#include <conio.h>
void biggest (int x, int y);
void main ()

```

{
    int a, b;
    scanf ("%d%d", &a, &b);
    biggest (a, b);
    getch ();
}
void biggest (int x, int y)
{
    if (x > y)
        printf ("%d", x);
    else
        printf ("%d", y);
}

```

Q. WAP to find square & cube of the function.

Ans-
#include <stdio.h>
#include <conio.h>
int square (int);
void main ()

```

{
    int a;
    scanf ("%d", &a);
    square (a);
    getch ();
}

```



```
scanf ("%d", &a); cube(a);
square(a);
getch();
```

```
int square (int x)
{
    return x = a * a;
    return x;
}

int cube (int y)
{
    return y = a * a * a;
    return y;
}
```

Q. WAP to create a function swap with exchange 2 integers.

Ans - #include <stdio.h>

```
#include <conio.h>
void swap (int, int);
void main ()
```

```
{
    int a, b;
    printf ("Enter two numbers\n");
    scanf ("%d %d", &a, &b);
    swap (a, b);
    getch();
}
```

```
void swap (inta, inty)
{
    inte;
    e = x;
}
```

```

scanf x = y;
scanf y = x;
printf ("%d\n", x, y);
```

Pass by value \Rightarrow With in function the parameters can be passed by value or by reference. This concept is known as (i) call by variable (Pass by value) (ii) call by address (Pass by reference)

(i) Pass by value :- It is the concept in which the values pass b/w the functions are not the actual value but they are copies of those value which are sent to be function. Therefore any changes made by the function to the values are made to the copies. And thus doesn't affect the original values.

eg - void swap (int, int);
void main ()

```
{
    int a, b;
    scanf ("%d %d", &a, &b);
    swap (a, b);
    printf ("a = %d, b = %d", a, b);
    getch();
}
```

```
void swap (inta, intb)
```

Pass by reference

Q. WAP to find sum of n number.

Ans-
int total();
void main()

```

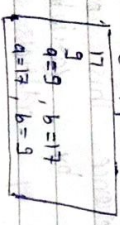
{
    int sum;
    sum = total();
    printf("%d", sum);
}
int total()
{

```

```

    int a;
    c=a;
    a=b;
    b=c;
    printf("%d", a+b);
}

```



```

    int a[10000];
    int n,i;
    int s=0;
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
        s = s + a[i];
    }
    return s;
}

```

Q. WAP to find factorial of a number.

Ans-
#include <stdio.h>
#include <conio.h>

```

int fac(int);
void main()
{

```

```

    int n,z;
    z = fac(n);
    printf("%d", z);
}
int fac(int)
{

```

```

    int n,q;
    int s=1;
    for(a=1; a<n; a++)
    {
        scanf("%d", &n);
        s = s * a;
    }
    return s;
}

```

Q. WAP to generate fibonacci series.

Ans-
#include <stdio.h>
#include <conio.h>

```

void fib();
void main()
{
    fib();
}

```

```

    fib();
    getch();
}

```


~~void fib()~~

Output
0 1 1 2 3 5 8

```
int a=0;
int b=1;
C=a+b;
printf("%d", C);
```

~~a=b;~~
~~b=C;~~

void fib()

```
int a=0, b=1;
int i, n;
scanf("%d", &n);
printf("%d", a, b);
for(i=0; i<n; i++)
{
    C=a+b;
    a=b;
    b=C;
}
```

printf("%d", C);

}

Q. WAP to check if a no. is palindrome or not.

Ans-
include <stdio.h>
include <conio.h>
int main()

void main()

```
int n;
scanf("%d", &n);
n = palin(n);
if (n == 1)
```

printf("no. is palindrome");

else

printf("no. is not palindrome");

getch();

int palin(int n)

{
 int t;

int rem;

int rev=0;

t=n;

while (t>0)

{
 rem = t%10;

rev = 10*rev + rem;

t = t/10;

if (n==rev)

{
 return 1;

else
 {
 return 0;

432 → 234
432/10 = 2
432%10 = 43
432/10 = 43
43/10 = 4
43%10 = 4
43/10 = 4
4/10 = 0
4%10 = 4
4/10 = 0

Q. WAP to check no. is armstrong or not.

```

// not
#include <stdio.h>
#include <conio.h>
#include <math.h>
int armstrong(int);
void main()
{
    int n, n1;
    scanf("%d", &n);
    n1 = armstrong(n);
    if (n == n1)
        printf("no. is armstrong");
    else
        printf("no. is not armstrong");
    getch();
}

int armstrong(int n)
{
    int temp, sum = 0;
    while (n > 0)
    {
        temp = n % 10;
        sum = sum + temp * temp * temp;
        n = n / 10;
    }
    if (sum == n)
        return 1;
    else
        return 0;
}

```

else { return 0; }

Recursion ⇒ It is the process in which a function call itself. Here by initiating a indefinite loop.

Q. WAP to calculate a factorial of a no. using recursion.

```

// not
#include <stdio.h>
#include <math.h>
int fact(int);
void main()
{
    int n;
    scanf("%d", &n);
    printf("fact of %d is %d", n, fact(n));
}

int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

UNIT - 4

(Pointers, structure & File handling)

Pointers \Rightarrow Pointers are of pointer type. Whose variable can store address of some other variables.

Characteristic of pointer : —

- (i) Its value is an address.
- (ii) It can be used to refer value of other variable.
- (iii) It can also refer elements of arrays.
- (iv) It can directly access at work with memory address

Syntax : —

datatype * ^{indirection} variable name;

eg -

```
int * ptr;  
float * xyz;  
char * n;
```

Q. WAP to demonstrate a basic pointer.

Ans - void main()

{

int * p;

int a;

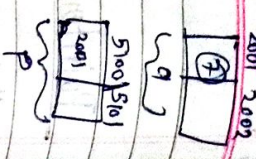
a = 7;

p = &a;


```

7 → printf ("%d", a);
2001 → printf ("%d", &a);
2001 → printf ("%u", p);
500 → printf ("%u", &p);
7 → printf ("%d", *p);
3 getch();

```



Where ~~the~~ ~~refer~~ ~~to~~ address and ~~the~~ ~~operation~~ ~~which~~ ~~points~~ ~~to~~ be the value inside the address.

Q. WAP to add using pointer.
 Ans =

```

1 void main()
2 {
3     int *p, *q;
4     int a, b, c;
5     a = 7;
6     b = 5;
7     p = &a;
8     q = &b;
9     c = *p + *q;
10    printf ("%d", c);
11    *p = 19;
12    printf ("%d", a);
13 }

```

Q. WAP to multiple using pointer.
 Ans -
 # include <stdio.h>
 # include <conio.h>

```

void main()
{
    int *p, *q;
    int a, b, c;
    a = 6;
    b = 5;
    p = &a;
    q = &b;
    c = *p * *q;
    printf ("%d", c);
}

```

Q. WAP to swap to no. using pointer & function.
 Ans -

```

1 # include <stdio.h>
2 # include <conio.h>
3 void swap (int*, int*);
4 void main()
5 {
6     int a, b;
7     scanf ("%d %d", &a, &b);
8     printf ("%d %d", a, b);
9     swap (&a, &b);
10    printf ("%d %d", a, b);
11 }
12 void swap (int* p, int* q)
13 {
14     int c;
15     c = *p;
16     *p = *q;
17     *q = c;
18 }

```

Q. WAP to swap to no. using pointer & function.
 Ans -

Q. WAP to swap to no. using pointer & function.
 Ans -

★ Pass by reference :- Also known as call by address

/ pointer is the concept of parameter passing in which the address of variable is passed to the function.

Since the address is passed any changes made by the function to the variable are permanent & are reflected inside the calling function as well that is the original value ~~are~~ is passed so the changed made are seen in the original value.

Pointers & arrays :- Arrays are closely related to pointer

through the fact that pointer can be ~~are~~ used to represent the entire array.

Using the concept of arithmetic the pointer can move and traverses through the entire array.

Q. WAP to print ~~an~~ element on array

Ans -

```
#include <stdio.h>
void main()
{
    int a[] = {5, 16, 7, 9, 14};
    int *p;
    p = &a[0];
    // p = a;
```

```
printf("%d", *p);
// 5
p = p + 1;
printf("%d", *p);
// 16
printf("%d", *(p+1));
// 7
printf("%d", *p+1);
// 17
```

Q. WAP to input an array of 10 element integer using pointer.

Ans -

```
#include <stdio.h>
void main()
{
    int i, a[10];
    int *p;
    p = &a[0];
    for (i = 0; i < 10; i++)
    {
        scanf("%d", p+i);
    }
    for (i = 0; i < 10; i++)
    {
        printf("%d", *(p+i));
    }
}
```

Q. WAP to add an array of 10 integer

Ans -

```
#include <stdio.h>
#include <conio.h>
```



```

void main ()
{
    int i, a[10];
    int s=0;
    int *p;
    p = &a[0];
    for (i=0; i<10; i++)
    {
        scanf ("%d", p+i);
    }
    for (i=0; i<10; i++)
    {
        s = s + *(p+i);
    }
    printf ("%d", s);
    getch();
}
    
```

IMP

Structures \Rightarrow These are user define data type whose variable can store values of different type.

It is basically a collection of heterogeneous elements.

A structure is user define because its variable can not be directly used like the integer variable or float variable.

Before creating a variable the struct must be define which will contain the variable with its data types.

So in order to use the variable the structure must define its members. To define a structure :-

Syntax :-

```

struct tagname
{
    type member 1;
    type member 2;
}
    
```

Advantage of structure :-

(i) It can hold multiple values of different type.

(ii) It can group & organised similar information.

eg- Struct student

```

char name[10];
int roll;
int year;
char branch[20];
    
```

To create a structure variable use the following syntax:

eg- struct tagname variable name;
 struct student s1, s2, s3; --;

→ To access the members of structure we must use dot(.) operator in the form
variable name . member

eg- s1.roll

Q. WAP to input & print student detail.

```
#
#
# struct student
{
    char name[10];
    int roll;
    int year;
    char branch[20];
};

void main()
{
    struct student s1;
    printf("Enter name");
    scanf("%s", s1.name);
    printf("Enter roll");
    scanf("%d", &s1.roll);
    printf("Enter year");
    scanf("%d", &s1.year);
    printf("Enter branch");
    scanf("%s", s1.branch);
    printf("Student details are:");
    printf("Name: %s, Roll: %d, Year: %d, Branch: %s", s1.name, s1.roll, s1.year, s1.branch);
}
```

Q. WAP to print % of a student

```
#
#
# struct student
{
    char name[10];
    int sem;
    int ph, ch, cp, c, m;
};

void main()
{
    int total;
    float per;
    printf("Enter details");
    gets(s.name);
    scanf("%d", &s.sem);
    scanf("%d", &s.ph);
    scanf("%d", &s.ch);
    scanf("%d", &s.cp);
    scanf("%d", &s.c);
    scanf("%d", &s.m);
    total = s.ph + s.ch + s.cp + s.c + s.m;
    per = (float)total / 5;
    printf("Percentage: %f", per);
}
```

Array with in structure ⇒

```
Q.
#
# struct student
{
    char name[10];
    int roll;
    int year;
    char branch[20];
};

void main()
{
    struct student s1;
    printf("Enter name");
    scanf("%s", s1.name);
    printf("Enter roll");
    scanf("%d", &s1.roll);
    printf("Enter year");
    scanf("%d", &s1.year);
    printf("Enter branch");
    scanf("%s", s1.branch);
    printf("Student details are:");
    printf("Name: %s, Roll: %d, Year: %d, Branch: %s", s1.name, s1.roll, s1.year, s1.branch);
}
```

Enter name
Sahil
28
Sahil
28


```

char name[10];
int sem;
int m[5];
}

```

```

void main()
{
}

```

```

int i, total;
Add per;
printf("Enter details");
gets(s.name);

```

```

scanf("%d", &s.m[i]);
total = total + s.m[i];
per = (total / 5);

```

```

printf("%f", per);
}

```

```

}

```

Q. WAP to calculate a bill of purchasing

of 5 books mentioning 5% GST.

```

#include <stdio.h>
#include <conio.h>

```

```

struct bill
{

```

```

}

```

```

char name[10];

```

```

int book[5];

```

```

}

```

```

void main()
{

```

```

}

```

```

int i, total;
float g;
printf("Enter details");
for(i=0; i<5; i++)
{
}

```

```

scanf("%d", &b.book[i]);
total = total + b.book[i];

```

```

g = (total * 5 / 100);

```

```

printf("%f", g);
}

```

Array of structure :- A structure variable represented an array of structure and can be used to represented multiple complex structure.

Q. WAP to calculate percentage of 60 student.

```

#include

```

```

struct student
{

```

```

char name[10];

```

```

int sem;

```

```

int m[5];

```

```

int total;

```

```

float per;

```

```

}

```

```

void main()
{

```

```

}

```



```

int i, j;
for (i=0; i<60; i++)
{
    gets(sci[0].name);
    scanf("%d", &sci[0].sem);
    for (j=0; j<5; j++)
    {
        scanf("%d", &sci[0].m[j]);
        total = total + sci[0].m[j];
    }
    sci[0].total = sci[0].total + sci[0].m[j];
    sci[0].per = (float)sci[0].total / 5;
}
for (i=0; i<60; i++)
{
    printf("%v.f", sci[0].per);
}

```

Q. WAP to calculate bill of 50 books.

```

#include <stdio.h>
#include <conio.h>
 clrscr();
 char bname[10];
 char author[10];
 int price;
 int b[50];

```

```

void main()
{
    int i, total=0;
    printf("Enter details");
    for (i=0; i<50; i++)
    {
        gets(b[i].bname);
        gets(b[i].author);
        scanf("%d", &b[i].price);
        total = total + b[i].price;
    }
    printf("%v.d", total);
}

```

Q. Create a structure for a cricket player specify its name, team, match and runs in each match. The program for 11 players which prints name, team and batting average of each players.

Q. MAP to print book detail using pointer to structure.

Ans

```

struct book
{
    char bname[10],
    char bauthor[10],
    int price;
} b;

void main()
{

```

```

    struct book *p;

```

```

    printf("enter book name\n");

```

```

    gets(p->bname);

```

```

    printf("enter author\n");

```

```

    gets(p->bauthor);

```

```

    printf("enter price\n");

```

```

    scanf("%d", &p->price);

```

```

    printf("%s\n%s\n%d", p->bname, p->bauthor,

```

```

        p->price);
}

```

Q. MAP to copy data of one structure variable to another.

Ans-

```

struct book
{

```

```

    char bname[10];

```

```

    char author[10];

```

```

    int price;
} b1, b2;

```

Pointer to structure:- Similar to work a regular variable a pointer can also point to a structure variable for this the pointer variable must be of a type structure and the pointer variable can access the member of structure using de-references operation →


```
void main()
```

```
{ struct book *p;
```

```
p = &b2;
```

```
printf("enter book name\n");
```

```
gets(p->bname);
```

```
printf("enter author name\n");
```

```
gets(p->bauthor);
```

```
printf("enter price\n");
```

```
scanf("%d", &p->price);
```

```
strcpy(b1.bname, b2.bname);
```

```
strcpy(b1.bauthor, b2.bauthor);
```

```
strcpy b1.price = b2.price;
```

```
printf("%s\n%s\n%d", b1.bname, b1.bauthor, b1.price);
```

File & multi file handling mgmt:-

File \Rightarrow It is a collection of data in stored in multiple format.

File provides a permanent medium to store data even when the program is not running.

File is generally store in secondary storage and thus is non-volatile and the program access / store data in the file.

There are two types of file.

(i) Text file

(ii) Binary file \leftarrow variable length

(i) Text file :-

Also known as ASCII files is a file which represents information as a stream of readable address character.

File Handling function :- In order to work operations need to be perform on the file various.

(i) Reading :- It is the process of taking data from the file to the program.

(ii) Writing :- It is the process of taking the data from the program and storing in file.

(iii) Appending :- It is the process of adding editing contain to and exiting file with data.

(iv) opening :- It is the process of opening a file to perform operation.

(v) Closing :- It is the process of closes the file and save the result.

File operation are handled by various file handling function which are already define in stdio.h

(1) File opening :- To open a file a file pointer must be created.

Syntax -
FILE *fp; pointer;

fopen :- once the file pointer be created then become the file using function

Syntax -
fp = fopen ("file name.txt", "mode");

mode of a file :- file mode specifies how the file is open in terms of operation

(i) r ⇒ Known as read mode. It open the file for reading if does not create the new file.

(ii) w ⇒ (write mode) This modes open the file to write the new data. Create a new file if does not exist.

(iii) a ⇒ (append) open the file for adding more data to existing file. Create a new file if it does

not exist

(iv) r+ (read & write mode) ⇒ Opens the file for reading and writing.

(v) w+ (write & read mode) ⇒ Opens the file for writing & reading.

(vi) a+ (read & append) ⇒ Opens the file for reading & appending.

(2) File closing :- Uses the function fclose to close the file

Syntax -
fclose (file pointer);

Q. WAP to open and close a file.

Ans -
#include <stdio.h>
#include <conio.h>
void main()

{

FILE *fp;

fp = fopen ("Hello.txt", "w");

if (fp == NULL)

{

printf ("cannot open file");

exit (0);

}

fclose (fp);

}

Reading and writing :-

Q. WAP to write a character on to the file.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void main()
{
```

```
    char ch;
    FILE *fp;
    fp = fopen("Hello.txt", "w");
    if (fp == NULL)
```

```
    {
        printf("cannot open file");
        exit(0);
    }
```

```
    ch = getc();
    putc(ch, fp);
    fclose(fp);
}
```

(i) 3

Q. WAP to read a character from a file.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void main()
{
```

```
    char ch;
    FILE *fp;
    fp = fopen("Hello.txt", "r");
    if (fp == NULL)
```

```
    {
        printf("cannot open file");
        exit(0);
    }
```

Q.

WAP to print multiple data on to the file.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void main()
{
```

```
    char a;
    FILE *p;
    p = fopen("Hello.txt", "w");
    if (p == NULL)
```

```
    {
        printf("cannot open file");
        exit(0);
    }
```

```
    while (ch = getc())
```

```
    {
        fputc(ch, p);
        ch = getc();
    }
```

3

fputc & fgetc ⇒ These character i/o functions are file functions which are used to read/write to/from the file.

syntax:

```
fputc (char *var, const FILE *var);
char *var = fgetc (FILE *);
```


Q. `feof()` \Rightarrow It is a end of file function generally implemented in text files and is used to obtain the end of file parameter. It returns a non zero value when end of file (EOF) is reached otherwise it returns zero.

Syntax :-
`feof (FILE *)`

Q. WAP to read data from the file and count the no. of character.

Ans -
 # include <stdio.h>
 # include <conio.h>
 # include <stdlib.h>
 void main()
 {

char a, n=0;
 FILE *fp;
 fp = fopen("Read.txt", "r");
 if (fp == NULL)
 {

printf("cannot open file");
 exit(0);
 }

while (feof(fp) == 0)
 {
 a = fgetc(fp);

n++;
 printf("%d", n);
 }
 fclose(fp);
 }

Q. WAP to count word in file.

Ans -
 # include <stdio.h>
 # include <conio.h>
 # include <stdlib.h>
 void main()
 {

char a, n=0;
 FILE *fp;
 fp = fopen("count.txt", "r");
 if (fp == NULL)
 {

printf("cannot open file");
 exit(0);
 }

a = fgetc(fp);
 while (feof(fp) == 0)
 {

printf("%c", a);
 if (a == '\n' || a == '\0')
 {

n++;
 }

a = fgetc(fp);
 printf("n words = %d", n);
 fclose(fp);
 getch();
 }

f printf & f scanf \Rightarrow These functions are similar to printf and scanf but are file based.

fprintf :- These function is used to write data of any type on to the file.

fscanf :- This function is used to input (read) single data of any type from the file.

Syntax →
 . `fscanf (FILE*, "conversion specifier", variable)`

`fscanf (FILE*, "conversion spec", address of var)`

Q. WAP to print your roll no. on to a file.

Ans-
 #include <stdio.h>
 #include <conio.h>
 #include <stdlib.h>
 void main()

```

{
    FILE *fp;
    fp = fopen ("roll.txt", "w");
    if (fp == NULL)
    {
        exit (0);
    }

```

```

    scanf ("%d", &a);
    fprintf (fp, "%d", a);
    fclose (fp);
}

```

Variable length files are (fprintf, scanf)

Binary file :-

These are the files in which information is represented as single form & binary form. The information present in these files are generally unreadable. The modes for these files are

rb	
r+b	
wb	
w+b	
ab	
a+b	

extension →

.dot
 .bak

Variable length files :- These files information is represented on the type and can be categorized based on type. Every single piece of information is implemented individually in a separate line.

These files are variable length because every single line may represent information of different type.

Q. WAP to write student information on to a file.

```

# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
void main()
{
    void main()

```

FILE *fp;

Student student char name[10];
int roll;

~~char~~ float per; "wb");
fp = fopen ("Student.dat", "wb");
printf ("enter name");
gets ("name");
printf ("enter roll & per");
scanf ("%d %f", &roll, &per);
fprintf (f, "%s\n", name);
fprintf (f, "%d\n", roll);
fprintf (f, "%f", per);
fclose (f);

3

ABC
25
72.3

Q. WAP to read student information from the file and print as output

#include <stdio.h>
#include <conio.h>

void main ()

FILE *f;
char name[10];
int roll;

float per;

fp = fopen ("Student.dat", "rb");

fscanf (f, "%s\n", &name);
fscanf (f, "%d\n", &roll);
fscanf (f, "%f", &per);
~~fclose (f);~~

3

fprintf (f, "%s\n", name);
fprintf (f, "%d\n", roll);
fclose (f);

Fixed length files \Rightarrow These are similar to the difference that each file with these data is represented as a same size. And these file serve as a collector of records.

VLF

FLF

(i) Data is represented individually. where as

(ii) where as data is represented a group of record.

(ii) Each Data has a different size.

(iii) Each record is of same size

Block I/O \Rightarrow

These input/output file functions data in a single step.

fwrite :- This function writes the complete record (like of a structure) on to the file.

fread :- This function reads the complete record on to the file.

Syntax -

fwrite (address of variable, number, file ptr);
fread (address of var., no., file pointer);

Q. WAP to create a student structure and print it in a file.

Ans -

```
#include <stdio.h>
#include <conio.h>

void main()
{
    char name[10];
    int roll;
    float per;
    FILE *f;
    struct student
    {
        int x;
        char *name;
        int roll no;
        char branch[3];
    } s;
    void main()
    {
        FILE *f;
        f = fopen("student.dat", "wb");
        scanf("%d", &s.roll no);
        gets(s.name);
        scanf("%d", &s.roll no);
```

size of () :- This operator returns size of a variable or datatype in Bytes

gets(s.name);

fwrite (s, size of (s), 1, f);

Q. WAP to read and copy student structure from a file.

Ans -

```
#include <stdio.h>
#include <conio.h>
struct student
{
    int x;
    char *name;
    char branch[3];
    int s, e;
    void main()
    {
        FILE *f;
        f = fopen("student.dat", "rb");
        fread (f, size of (f), 1, f);
        printf ("%d", s, e, name, branch);
        fclose (f);
    }
}
```

Q. WAP to copy data of file to other

Ans -

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char ch;
```



```
FILE *f, *p;
f = fopen("ABC.txt", "w");
p = fopen("XYZ.txt", "w");
while (feof(f) != 0)
{
    ch = fgetc(f);
    fputc(ch, p);
}
fclose(f);
fclose(p);
```

Void pointer & NULL pointer:-

NULL pointer \Rightarrow It is a special pointer which points to null and is generally used to deal with stray pointers.

eg- `int *a = NULL;`

Void pointer \Rightarrow It is a generic pointer which can store address

of any variable of any type. To prefer the value using void pointer, the void pointer must be type casted.

eg-
`void *a;`
`int b;`
`char c;`
`float d;`

```
a = &b;
a = &c;
a = &d;
printf("%f", (float*)a);
```


Unit-2 Number system

Number system is a mathematical location of ~~expr~~ expressing no. It is of two type.

1] non-positional

2] positional

1] Non-positional :- In this system the position of digit does not matter.

eg- Roman no.

2] Positional :- In this position call the digit is important.

Radix or base \Rightarrow For a particular positional no. system base represent the no. of unique digit to express a value.

(a) Decimal NS :- It has a base of 10 that is this system is expressed as a combination of 10 unique symbols from 0 to 9.

Numeric value are expressed in terms of power of 10 with starting from 0 and determining the position of each unique digit.

eg

10^2 10^1 10^0
hundred ten unit
 10^{-1} 10^{-2}
one tenth one hundredth

(b) Binary

NS \Rightarrow It has a base 2. i.e. numeric values can be represented only 2 no. mainly symbol 0 & 1.

(c) Octal

NS \Rightarrow It has a base 8th with unique digit from 0 to 7

(d) Hexadecimal \Rightarrow It has a base 16 with unique digit 0 to 9 & A to F.

★ Conversion :-

Number system conversion can be term ~~that~~ (i) other no. system to decimal (ii) decimal to other no. system (iii) one no. system to other no. system

(1) Other no. system to decimal :- For this

conversion multiplication operation perform and the source no. system values are expressed that power of its base.

(2) binary to decimal

eg- $(1011)_2 \rightarrow (??)_{10}$

$$= 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 1$$

$$= 4 + 0 + 1$$

$$= (5)_{10}$$

eg- $(1111.10)_2 \rightarrow (??)_{10}$

$$= 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1 + 2^{-1} \times 1 + 2^{-2} \times 0$$

$$= 8 \times 1 + 4 \times 1 + 2 \times 1 + 1 \times 1 + \frac{1}{2} \times 1 + \frac{1}{2^2} \times 0$$

$$= 8 + 4 + 2 + 1 + 0.5$$

$$= (15.5)_{10}$$

(b) Octal to decimal :-

In this case positions are represented as power of 8.

eg- $(561)_8 \rightarrow (??)_{10}$

$$= 8^2 \times 5 + 8^1 \times 6 + 8^0 \times 1$$

$$= 16 \times 5 + 8 \times 6 + 1 \times 1$$

$$= 380 + 48 + 1$$

$$= (369)_{10}$$

eg- $(4312.16)_8 \rightarrow (??)_{10}$

Ans- $8^3 \times 4 + 8^2 \times 3 + 8^1 \times 1 + 8^0 \times 2 + 8^{-1} \times 1 + 8^{-2} \times 6$

$$= 512 \times 4 + 64 \times 3 + 8 \times 1 + 1 \times 2 + \frac{1}{8} \times 1 + \frac{1}{64} \times 6$$

$$= 2048 + 192 + 8 + 2 + \frac{1}{8} + \frac{3}{32}$$

$$= 2250 + 0.12 + 0.09$$

$$= (2250.218)_{10}$$

(c) Hexadecimal to decimal :- In this case expressed by multiplication factor of power of 16.

eg- $(AA)_{16} \rightarrow (??)_{10}$

Ans- $16^1 \times 10 + 16^0 \times 10$

$$= 160 + 10 = (170)_{10}$$

eg- $(43F.3B)_{16} \rightarrow (??)_{10}$

Ans- $16^2 \times 4 + 16^1 \times 3 + 16^0 \times 15 + 16^{-1} \times 3 + 16^{-2} \times 11$

$$= 256 \times 4 + 16 \times 3 + 1 \times 15 + \frac{3}{16} + \frac{11}{256}$$

$$= 1024 + 48 + 15 + 0.18 + 0.05$$

$$= (1087.23)_{10}$$

2058
192
360
312

(ii) Decimal to other no. system: — In 4

to convert the decimal division operation is performed successfully on the question & the result is passed by remainder.

(a) Decimal to binary: — Here the

is 2. which divides the no. success until a value lower than 2 is obtained.

eg- $(27)_{10} \rightarrow (??)_2$

2	27	1	MSD
2	13	1	
2	6	0	
2	3	1	
2	1	0	
1	0	0	1

$(11011)_2$

operation quotient remainder

$27/2 = 13$ remainder 1
 $13/2 = 6$ remainder 1
 $6/2 = 3$ remainder 0
 $3/2 = 1$ remainder 1
 $1/2 = 0$ remainder 1

$(11011)_2$

Q. $(35)_{10} \rightarrow (??)_2$

2	35	1
2	17	1
2	8	0
2	4	0
2	2	0
1	1	1

$(100011)_2$

Q. $(135)_{10} \rightarrow (??)_2$

2	135	1
2	67	1
2	33	1
2	16	0
2	8	0
2	4	0
2	2	0
1	1	1

$(10000111)_2$

Q. $(135.23)_{10} \rightarrow (??)_2$

$0.23 \times 2 = 0.46$
 $0.46 \times 2 = 0.92$
 $0.92 \times 2 = 1.84$

$(10000111.001)_2$

Q. $(147.381)_{10} \rightarrow (?)_8$

Ans -

147	3
8 147	8 3
18	2

$(223)_8$

$0.381 \times 8 = 3.048$
 \downarrow
 $0.048 \times 8 = 0.384$
 \downarrow
 $(223.308)_8$

Q. $(4372.1134)_{10} \rightarrow (?)_{16}$

Ans -

4372	11	34
16 4372	16 11	16 34
273	6	2

$(10B4.1A)_{16}$

$0.1134 \times 16 = 1.8144$
 \downarrow
 $0.8144 \times 16 = 13.0304$
 \downarrow
 $(1114.1D)_{16}$

3] CONVERSION FROM ONE NO. SYSTEM TO OTHER:-

The decimal system act as the intermediate i.e. initially the provided no. is convert into decimal no. system & then

decimal system is converted into the target no. system.

Q. $(135.23)_8 \rightarrow (?)_2$

Ans -

Step-I

$(57)_8 \rightarrow (2)_2$

Step-II

$(47)_{10} \rightarrow (2)_2$

$5 \times 10^1 + 7 \times 10^0 = 40 + 7 = 47$

2	47	1
2 47	2 1	
23	0	

$(101111)_2$

Q. $(110011.101)_2 \rightarrow (?)_8$

Ans -

Step-I

$(110011.101)_2 \rightarrow (2)_8$

$1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

$32 + 16 + 0 + 0 + 4 + 2 + 0 + \frac{1}{2} + \frac{1}{4}$

$59 + 0.5 + 0.125 = (59.625)_{10}$

$(101.57)_{10}$

$(101.57)_{10}$

$(101.57)_{10}$

$$(51.625)_{10} \rightarrow (?)_8$$

$$\begin{array}{r} 8 \overline{) 51.625} \\ 40 \\ \hline 11 \\ 8 \\ \hline 3 \end{array}$$

$$= 63$$

$$0.625 \times 8 = 5.000$$

$$(63.5)_8$$

$$Q. (1101111)_2 \rightarrow (?)_{16}$$

$$Ans - (1101111)_2 \rightarrow (?)_{10}$$

$$2^6 \times 1 + 2^5 \times 1 + 2^4 \times 0 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1$$

$$64 + 32 + 8 + 4 + 2 + 1$$

$$= (111)_{10}$$

$$\frac{96}{111} \rightarrow (?)_{16}$$

$$\begin{array}{r} 16 \overline{) 111} \\ 80 \\ \hline 31 \end{array}$$

$$Q. (347.55)_8 \rightarrow (?)_{16}$$

$$(6F)_{16}$$

$$Ans - Step-1 (347.55)_8 \rightarrow (?)_{10}$$

POPUL

POPUL
231
0.078
0.103

$$8^3 \times 3 + 8^2 \times 4 + 8^1 \times 7 + 8^0 \times 5$$

$$64 \times 3 + 32 + 1 \times 7 + \frac{5}{8} + \frac{5}{64}$$

$$192 + 32 + 7 + 0.625 + 0.078$$

$$(231.0703)_{10}$$

$$Step-10 (231.703)_{10} \rightarrow (?)_{16}$$

$$\begin{array}{r} 16 \overline{) 231} \\ 160 \\ \hline 71 \end{array}$$

$$(E3)_{16}$$

$$0.703 \times 16 = 11.248$$

$$0.248 \times 16 = 3.968$$

$$(E3.B4)_{16}$$

$$Trick: \quad 3 \quad 4 \quad 7 \quad 5 \quad 5$$

$$00001110011101101101$$

$$\begin{array}{r} 0.703 \\ \times 16 \\ \hline 11.248 \end{array}$$

$$(i) (341.227)_8 \rightarrow (?)_2$$

$$(ii) (FA31)_{16} \rightarrow (?)_{10}$$

$$(iii) (8412)_{10} \rightarrow (?)_8$$

$$(iv) (10111.110)_2 \rightarrow (?)_{10}$$

$$(v) (32)_x \rightarrow (53)_6$$

Ans (i)

$$(341.227)_8 \rightarrow (??)_{10}$$

$$3 \times 8^2 + 4 \times 8^1 + 1 \times 8^0 + 2 \times 8^{-1} + 2 \times 8^{-2} + 7 \times 8^{-3}$$

$$64 \times 3 + 8 \times 4 + 1 \times 1 + \frac{2}{8} + \frac{2}{64} + \frac{7}{512}$$

$$192 + 32 + 1 + 0.25 + \frac{1}{32} + \frac{7}{512}$$

$$225 + 0.25 + 0.03 + 0.013$$

$$(225.293)_{10}$$

$$(225.293)_{10} \rightarrow (??)_2$$

2	225	1
2	112	0
2	56	0
2	28	0
2	14	0
2	7	1
2	3	1
2	1	1

$$(1110001)_2$$

$$0.293 \times 2 = 0.586$$

$$0.586 \times 2 = 1.172$$

$$(1110001.01)_2$$

(ii)

$$(FA31)_{16} \rightarrow (??)_{10}$$

$$15 \times 16^3 + 10 \times 16^2 + 3 \times 16^1 + 1 \times 16^0$$

$$+ 2560 + 48 + 1$$

$$61440 + 2560 = (64000)_{10}$$

(iii)

$$(8412)_{10} \rightarrow (??)_8$$

8	8412	3
8	1051	4
8	131	3
8	16	0
2		2

$$(20334)_8$$

(iv)

$$(10111.110)_2 \rightarrow (??)_{16}$$

$$(10111.110)_2 \rightarrow (??)_{16}$$

$$2^4 \times 1 + 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1 + 2^{-1} \times 1 + 2^{-2} \times 1 + 0 \times 2^{-3}$$

$$16 + 0 + 4 + 2 + 1 + 0.5 + 0.25$$

$$(23.75)_{10}$$

$$(23.75)_{10} \rightarrow (??)_{16}$$

16	23	7
1		7

$$16$$

$$0.75 \times 16 = 12$$

$$(17.C)_{16}$$

(v)

$$(32)_{10} \rightarrow (53)_8$$

$$(53)_8$$

Ans =

$$(32)_x \rightarrow (1?)_{10} = (53)_8 \rightarrow (1?)_{10}$$

$$x^1 \times 3 + x^0 \times 2 = 5 \times 8 + 3 \times 8^0$$

$$3x + 2 = 40 + 3$$

$$3x = 41$$

$$x = \frac{41}{3}$$

Complement of a number \Rightarrow Primarily used in digital

no. is the process to the obtain mathemathical subtraction using other mathemathic operation.

For a certain base system with radix R complement can be either $(R)^n$ or $(R)^n - 1$ complement.

on 50 $(R)^n$ - 1 complement.

For example the binary no. system their is 2's complement and 1's complement

eg- for decimal their can be 10's complement or 9's complement.

eg- for hexadecimal their can be 16's complement or 15's complement

To calculate R complement we can do $(R)^n - N$

To calculate $R-1$ complement we can do $(R)^n - N - 1$

For ~~the~~ N is the number

n is no. of digits

R is the base or Radix

Q. find 9's and 10's complement of 437

Ans- 10's complement $(R)^n - N$

$$(10)^3 - 437$$

$$1000 - 437$$

$$(563)_{10}$$

9's complement $(R)^n - N - 1$

$$(10)^3 - 437 - 1$$

$$999 - 437$$

$$(562)_{10}$$

Q. Find R 's & $(R-1)$'s complements of $(10111)_2$

Ans- R 's complement $R^n - N$

$$(2)_{10}^5 - (10111)_2$$

$$(32)_{10} - (10111)_2$$

$$(32)_{10} - (10111)_2$$

$$(32)_{10} - (23)_{10} - 1$$

$$(9)_{10}$$

$$(1001)_2$$

$$(1000)_2$$

Q. Find $(R-1)$'s complement of $(3A12)_{16}$

$$(16)_{16}^4 - (3A12)_{16} - 1$$

$$R^n - N - 1$$

$$(16)_{16}^4 - (3A12)_{16} - 1$$

$$(65536)_{16} - (3A12)_{16} - 1$$

$$(50663)_{16}$$

$$\begin{array}{r} 16 \\ 16 \\ \hline 16 \end{array} \quad \begin{array}{r} 50669 \\ 3168 \\ \hline 197 \\ 12 \end{array} \quad \begin{array}{r} 114 \\ 5 \\ \hline 119 \end{array}$$

$$(C5FD)_{16}$$

- Binary arithmetic \Rightarrow

$(7A31)_{16}$
R's complement

R' is complement

$(R^{-1})^s$ complement

$$(16)_0^4 - (7A31)_{16}$$

$$(\psi_0 - (\psi_3))_{11} = 1$$

(65536)₁₀ - (B1281)₁₀

65336 - 3/28/11

$$(34255)_{10}$$

(34254)₁₀

(D6F)₁₆

(D6E)₁₆

Binary arithmetic :-

(ii) Binary addition \Rightarrow

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$0 + 1 + 0$$

$$1 + 1 = 10$$

↑
canity

Ans - (ii)

$$\begin{array}{r} 153 \\ 209 \\ \hline 362 \end{array}$$

$$\begin{array}{r} 6536 \\ 31281 \\ \hline 34255 \end{array}$$

$$\begin{aligned} & \mathbf{a} \cdot \mathbf{1} \quad (11110011)_2 + (100011)_2 \\ & \mathbf{a} \cdot \mathbf{2} \quad (10111100)_2 + (100011)_2 \\ & \mathbf{a} \cdot \mathbf{3} \quad (10001111)_2 + (11001111)_2 \end{aligned}$$

$$8.3 \quad (10001111)_2 + (1000.11)_2$$

$$8.2 \ (10111100)_2 + (100011)_2$$

$$(1110011)_2 + (100011)_2 = (10010110)_2$$

~~110011~~
100000

$$\begin{array}{r} 1100010 \\ \hline \end{array}$$

(ii)

1011.1100
1000.11

10100.1000

$$(1010011001)_2$$

三

10001111

11001111

10101110

$$(10101110)_2$$

(ii) Binary subtraction \Rightarrow

$$0 - 0 = 0$$

1-0

$$\frac{0}{10}$$

1-1

Вот так

Q.1

$$\begin{array}{r} 1010 \\ - 010 \\ \hline 1000 \end{array}$$

$$(1010)_2 - (010)_2 = (1000)_2$$

Q.2

$$\begin{array}{r} 11010 \\ - 110 \\ \hline 01000 \end{array}$$

Q.3

$$\begin{array}{r} 1000 \\ 0011 \\ \hline 0101 \end{array}$$

Q.4

$$(a) (10001111)_2 - (110111)_2$$

$$(b) (11100010)_2 - (00100111)_2$$

$$(c) (010011001)_2 - (1001110)_2$$

Ans- (a)

$$\begin{array}{r} 10001111 \\ - 110111 \\ \hline 010011000 \end{array}$$

(b)

$$\begin{array}{r} 111001010 \\ - 001000111 \\ \hline 101111011 \end{array}$$

(c)

$$\begin{array}{r} 010011001 \\ - 1001110 \\ \hline 001001011 \end{array}$$

(iii) Binary multiplication \Rightarrow

$$0 \times 0 = 0$$

$$1 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 1 = 1$$

Q.1

$$\begin{array}{r} 101 \\ \times 11 \\ \hline 101 \\ 101 \\ \hline 1111 \end{array}$$

Q.2

$$(1101)_2 \times (111)_2$$

Ans- (1)

(2)

$$\begin{array}{r} 110101 \\ \times 1101 \\ \hline 110101 \\ 110101 \\ 110101 \\ 110101 \\ \hline 10111011 \end{array}$$

(iv) Binary division :-

$$g - (0100)_2 + (00010)_2$$

$$\sqrt{10} \quad \sqrt{100} \quad \sqrt{10}$$

10

000

g- $(10)_2 \div (10)_2$

$$10 \sqrt{10} 10$$

$$\frac{10}{00}$$

$$(10)_2 \div (10)_2 = (1)$$

eg - $(101010)_2 \div (0110)_2$

$$\begin{array}{r} 110 \overline{) 101010} \\ \underline{1111} \end{array}$$

110

1001

$$\underline{110}$$

0110

110

0000

$$(101010)_2 \div 11 = (0110)_2 = (111)_2$$

Representation of channel

data consists of text like character & symbols and numbers. This data is generally represented & transmitted b/w systems using a group of bits. Also known as binary code. These binary codes are category into weighted and non weighted codes.

(1) Weighted code:—In these code the position of each bit is represented as weights. And thus, the digit position changes so does its weight.

BCD- 8421

BCD - 8429

(ii) Non-weighted code : — This coding method does not obey the weight position rule and thus the weight is not determined through the digits position.

eg- excess-3

↓

BCD (Binary code decimal) :- It is known binary

Coded decimal

and is a coding technic to represent decimal value into binary digits.

It is a $4 B_n^{(1000)}$ where each decimal digit is represented as a combination

of 4 binary digit as per following table

decimal digit	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

eg- (36)₁₀ → (?)_{BCD}

(00111000)_{BCD}

eg- (174)₁₀ → (?)_{BCD}

(0001011101000001)_{BCD}

Excess-3 :- To remove the short

coming of BCD

which was not able to represent

highest numbers excess-3 code

is used. In this code each

decimal digit is added by

the value of 3 & its binary

equivalent is obtain.

eg- (174)₁₀ → (?)_{XS3}

$$\begin{array}{r} 174 \\ +3 \quad +3 \quad +3 \\ \hline 5074 \end{array}$$

(010100001101010101010101)_{XS3}

ASCII :- It is known as American

interchange. It is standard code for information

code to represent char, symbol & number

to binary. Each individual for ASCII

is assigned a decimal value & a binary

value. Initially a seven bit ASCII

code containing character A to Z, lower

and upper case- special characters,

special symbol, printable & non-printable.

It is known as ASCII-7 which

represent each char- using 7 bits. further and additional & external version of ASCII is also present known as ASCII-8. which uses 8 bits to represent each character. It provides an additional 128 characters (printable & non printable)

EBIC DIC:-

It is known as extended binary coded decimal for information. It is an 8 bit character format to represent no. char. in system and was primary designed to be used with computer and can primary contain 256 characters.

000 0000 to 001 1111 → special non printable (0 to 31)

010 0000 to 010 1111 → special characters (32 to 47)

011 0000 to 100 0000 → numbers & math symbols (48 to 64)

100 0001 to 101 1010 → A-Z (65 to 90)

110 0001 to 111 1010 → a-z (97 to 122)

100 0001	A	65
100 0010	B	66
100 0011	C	67
100 0100	D	68
100 0101	E	69
100 0110	F	70
100 0111	G	71
100 1000	H	72
100 1001	I	73

eg-
INDIA
↓ ↓ ↓ ↓ ↓
73 76 68 73 65

100 1001 100 1110 100 0100 100 1001 100 0001

Er Sahil Ka Guan