≈ C Language :- The language were developed in 1970s at Bell laboratories by Dennis ritchie. It was design for programming in the operating system called unix.
After the advent of C the unix operating system was rewritting it.

≈ Characteristics of C :-
It is the middle level language. It has the simplicity of high level language as well as the low level language. This aspect of C ~~market~~ makes it suitable for writing both application programming and system programming. It is an excellent efficient and general purpose language for most of the application such as mathematical, scienstific, bussiness and system software application.

Programming language :-
Before learning any language it is it is important to know various types of languages and their features.
The programming language can be classified into two types :-

1. Low - level language
2. High - level language

Low - level language :-
The language in this catagory are the machine language and assembly language.

Machine level language :-
Computers can understand only digital signals which are in 0 and 1.
The instructions given to the computer can be only in binary codes. Computer can understand only machine language. The machine language consist of instructions that are in binary 0 and 1.

Assembly language :-
In assembly language instructions are given in English like words such as MOV, ADD, SUB etc. It is easy to write and understand assembly program since a computer can understand only machine language. The translator that is use for translating is called assembler.

High-level language :-

For translating a high level language program into machine language compiler or interpreter is used. Every language has its own compiler or interpreter. Some language in this catagory : FORRAN, COBOL, PASCAL etc.

Translators :-

① Compilers.
② Assembler
③ Interpreter

1. **Compiler :-** Compiler and Interpreter are used to covert the code of high level language into machine language. This high level language is known as source program and corresponding machine language program is known as object program both compiler and Interpreter perform the same task. But ~~their~~ there is a difference in their working. A Compiler search all the error of the program and list them and the Interpreter checks the error of program statement by statement.

✡ Structure of C Program :-

C program are a collection of one or more functions every function is a collection of statements and perform specific task.

```
Comments
preprocessors directives
  Global variables
main ( ) function
  {
local variables.
  Statements
  - - - - - -
  }
function 1 ( ).
  {
local variables
  Statements
  - - - - - - -
  }
function 2 ( ).
  {
local variables.
  Statements
  - - - - - - -
  }
```

Comments :- Comments can be place anywhere in a program and are inclosed b/w the delimeters /* and */. comments are generaly used for documentation purpose.

Preprocess directive :- The commonly used preprocessors directive are # include and # define. # include can be used for including header files. # define can be used for define symbolic constants and macros.

Every C program has one or more functions. If a program has only one function than it must be main function. Execution of C program start with main( ) function. It has two parts declaration of local variables and statements.

✡ Elements of C :-

Every language has some basic elements and grammatical rules. Before understanding programming it must know the basic elements of C language.

The Basic elements are. characterset, variables, data types, Constants, key words, variables declaration,

expressions, statements etc.
All of these are used to construct a C Program.

(1) C Character Set :-

Alphabets :- A B C D ------ Z.
a b c d ------ z

Digits :-

Digits :- 0, 1 ------ 9

✡ Special characters :→

Character	Meaning	Character	Meaning
+	Plus sign	-	Tilde.
*	Asterisk	%	Percentage sign.
\	Backward Slash	/	Forward slash
<	less than sign	=	equal to
>	greater than sign	_	under score
(	left Parent thesis	}	Right Braces
{	left Braces	)	Right Parent thesis
[	left Bracket	]	Right Bracket
,	Comma	.	Period
?	Question mark	"	Double quotes
&	Ampersand	;	Semi colon
@	At the rate	!	Exclamation
'	Single goutes	|	vertical Bar Sign
$	Dollar sign.	^	Caret sign
:	Colon	#	Hash sign

Delimeters :-

Delimeters are used for syntatic meaning. These are given below.

:	Colon	used for label
;	Semi Colon	End of statements
( )	Parenthesis	used in expressions
[ ]	Square bracket	used for array
{ }	Curly bracket	used for block of statements
#	Hash	Preprocessor Directive
,	Comma	Variable Delimeter

Reserved words / Key words :-

These are certain words that are reserve for doing specific task. These words are known as keywords. They are always written in lower case. There are only 32 keywords in C which are given below:

Auto	break	Case	Char.
Const.	Continue	Default	Do
Double	else	Enum	Extern
Float	for	Go to	if
int.	long	Register	Return
Short	Signed	Size of	Static
Struct	Switch	Type def.	union
unsigned	void	Volatile	while.

* Identifiers :- Keyword are the predefine and can't be changed by the user. While Identifier are user words and are used to give names to entities like variables, arrays, functions, structure etc. Rules for naming Identifiers are given below as :-

1. The name should consist of only alphabates (both upper and lower case) digit and under score sign (___).
2. First character should be an alphabate or underscore.
3. The name should not be a keyword.
4. Since C is case sensitive. The upper case and lower case letters consider different.

for example :- Code
code
CODE
are the three diff. identifire.

Some examples of valid identifires name given below :-

Value
a
net_play
rec1
_data
MARKS

Some of invalid identifiers are given below :-

5bc
int
rec#
reg no.

* Data types :-

C supports different types of data. There are four fundamental data types in C which are :-

Int
Chara
float
double

Int :- Int is used to share integer value.

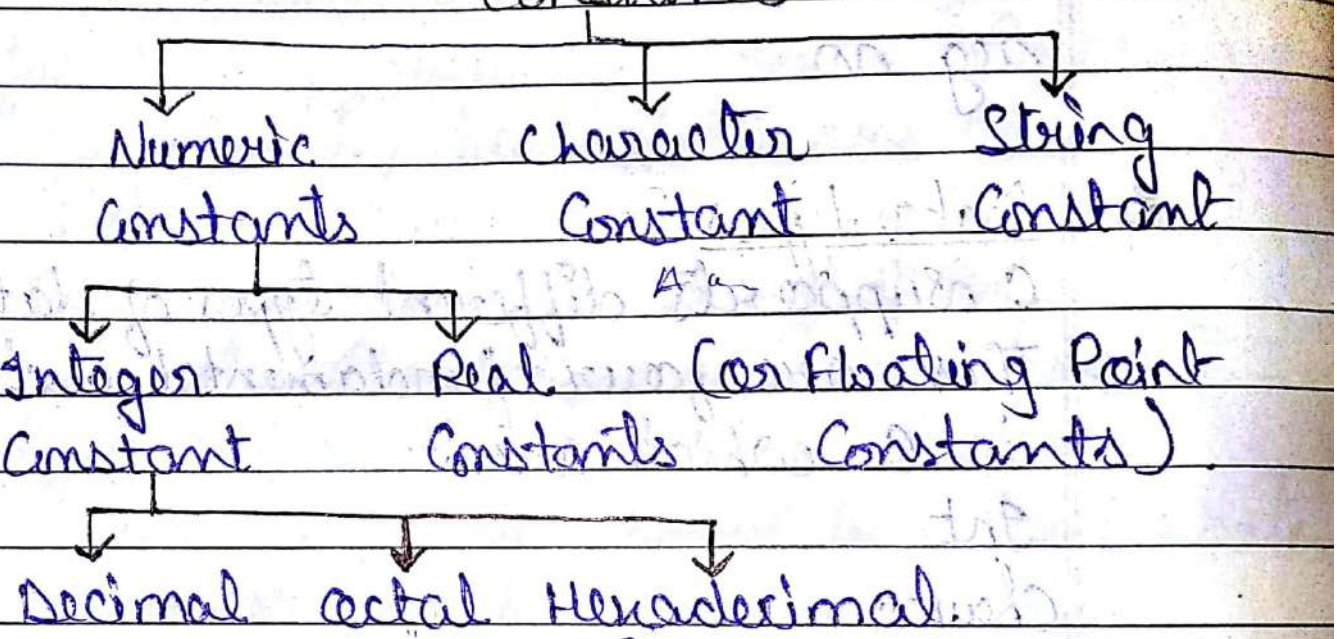
Charac :- Character is used to store any size single character.

Float :- Float is used for storing single precision floating point number.

Double :- Double is used for storing double precision floating point no.

Type	Storage size	Value range.
Char.	1 byte	-128 to 127
Int	2 byte	-32,768 to 32,767.
Float	4 byte	3.4E-38 to 3.4E+38
double	8 byte	1.7E-308 to 1.7E+308

☆ Constants :- Constant is a value that cannot be changed during execution of a program. There are three types of constants.



1. Numeric Constants :- Numeric constant consist of numeric digits that may or may not have decimal point. These are the rules for defining numeric constants.

i) Numeric constants should have atleast one digit.

ii) No comma or space is alow within the numeric constants.

iii) Numeric constants can either be positive or negative but default sign is always positive.

There are two types of Numeric constants.

a). Integer constants :- Integer constants are nos. which have no decimal points. These are three types of integer constants base on different no. system.

i) Decimal constant :- (0, 1, 2 ---9)
Base = 10.

ii) Octal Constant :- (0, 1, 2 ---7), Base = 8.

iii) Hexadecimal :- (0, 1, 2 ---9), (A, B, C, D, E, F), Base = 16.

Some valid decimal Integer constants are
123, 3705, 38467, 456896.

Some invalid decimal Integer constant are given below :-

2.5	Illegal character (.).
3#5	Illegal " (#).
98 5	No space alow.
0925	First digit cannot be Zero.
8,354	Comma is not alow.

- In octal Integer constant First digit must be 0.
example :- 05, 07, 088, 0789.

- In Hexadecimal Integer constant first two characters should be 0x or 0x
example :- 0x, 0X23, 0XFFF.

b. Real (Floating Point) Constant :-
Floating Point constants are Numeric constants that contains decimal point Some valid floating constants are :-
. 0.5 , 5.3.

2. Character :- Co character constant is a single character that is enclosed within single quotes.
Some valid character constant are:
'1' , '#'.

3. String Constant :- String constant has 0, 1 or more than 1 character. A string constant is enclose within double quotes.
Some valid string character Constant are:
"8", "Kumar", "593", " ", "A".

4. Variables :- Variable is a name that can be use to store value. Variable can take different value but one at time. These value can be changed during execution of program.
A data type is associated with each variables. The data type of the variable decide what value it can take.

✡. **Declaration of variables :-**

Declaration of variables specify its name and data type. The type and range of value that a variable can store depend upon its datatype. The syntax of declaration of variable is

datatype variable name ;

Here datatype may be int., float, char., double etc..

Some example of declaration of variable are :- Int x;

float salary;

char great;

Here x is a variable of type int., salary is a variable of type float, great is a variable of type character.

We can also declare more than one variable in a single declaration.

eg :- (Int. x, y, z, total;)

Here Int. x, y, z, total are all variables of type Int.

UNIT-2.

PAGE :
DATE : / /

Input Output Statement :-

C is a functional Programming language in which to perform these Input output operations. C Provid a library functions. This library is called in statement input output library. It is denoted by stdio.h. The header file containing such library functions is called stdio.h

Some of the standard input output functions in C are given below.

1. Scanf( )
2. Printf( )
3. getchar( )
4. putchar( )
5. gets( )
6. puts( )
7. getch( ).

These are two types of Input output Statement which are as.

1. Formatted I/o Statements
2. Unformatted I/O Statements

Formatted Input Output statement enable the user to especific the type of data.

On the other hand the unformatted Input output Statement do not especify the type of data.

PAGE:
DATE: / /

Type of I/O Statement	Input Statement	Output Statement
Formatted	Scanf( )	printf( )
Unformatted	getchar( )	putchar( )
	gets( )	puts( )

✡. Reading Input data :-
Input data can be entered into the memory from a Standard input device (Keyboard), C provide the Scanf( ) library functions for entering input data. This function can takes all type of value (Numeric, character, String) as input. The Scanf( ) functions can be written as. ~~Scanf("% Control String", & address 2, ----);~~
Scanf("% Control string", & address2, --);

This function should have atleast two parameters.
First Parameter is a control String
Second " is a Address of variable.

✡. 1. Control String :- Control string contains Conversion specification Character. It should be within double quotes.

Each character group is a combination of the (%) Symbole and one of the conversion. This conversion Specification character may be one or more. It depend on the number of variables, we want to Input.

2. Address of Variables :-
The address of variables is found by preceding the variable name by an ampersand (&) sign. This sign is called the address operator.
It gives the starting address of the variable name in memory.
A string variable is not procede by ampersand (&) sign to get the address.

✡. Conversion Specification character :-

%c → a single character
%d → a decimal integer
%f → a floating Point numbers
%lf → a long range of floating point nos
%o → an octal Numbers.
%x → a hexadecimal no.
%s → String

PAGE:
DATE: / /

4. Some example of scanf( ) are as :-

```
# Include < stdio.h>
    main ( )
      {
      Int marks ;
      ---------
    Scanf ("%d", & marks );
      }
```

The control string contains only one conversion specification character "%d" which implies that an integer value should be entered as input. The entered value will be stored in the variable marks.

Example 2:-

```
# Include <stdio.h>
    main ( )
      {
      Char str[30] ;
      ----------
    Scanf ("%s", str );
      }
```

Example 3:-

```
# Include < stdio.h>
    main ( ).
      {
```

```
int basic;
float hra;
char grade;
Scanf("%d%f%c", &basic, &hra, &grade);
}
```

✡. Writting output Data :-

output data can be written from computer memory to standard output device using printf library function. C provides the printf() to display the data on the monitor. With this function all type of values (Numeric, char, or String) can be written as output.

✓ The printf() is including in stdio.h
The printf() can be written as :-

```
printf("control String", Variable, Variable);
```

In this function the control string contains conversion specifications characters and text it. It should be enclosed within double quotes. The name of variable should no be proced by an ampersand (&) Sign.

PAGE:
DATE: / /

☆. Some examples of printf( ) are as:-

1. # include <stdio.h>
main( ).
{
printf("C is excellent \n");
}

Output = C is excellent.
The control string has only text and no conversion specification character. Hence the output is only text.

2. # include <stdio.h>
main( ).
{
int Basic = 2000;
printf("%d", basic);
}

output = 2000.

3. # include <stdio.h>
main( )
{
char ch = '5';

DATE: / /

printf("%c", ch);

}

✡. getch( ) :- getch is used to get a character from consolue. It can be used to hold program execution. Its primary purpose which is to wait until the users enter a character.

✡. putchar( ) :-
Putchar( ) out one character at a time to the standard output. Putchar( ) is used to write a character on standard output screen. In a C program we can use putchar( ) as below :-

putchar(char);
↳ Character is a variable or value.

✡ getchar( ) :- getchar( ) is used to get / read a character from keyboard input. In a C program we can use getchar( ) as below :-

getchar(char);

PAGE:
DATE: / /

```
#include<stdio.h>
    main()
      {
    char ch;
printf("enter a character");
     ch = getchar();
printf("The entered character is");
     Putchar(ch);
      }
```

✡. **Clrscr :-**
Clrscr means clean screen. It will clear the output of the previous program.

✡. **Operators :-** An operator specify an operation to be perform. C include a large no. of operators that fall under several different categories which are as :-

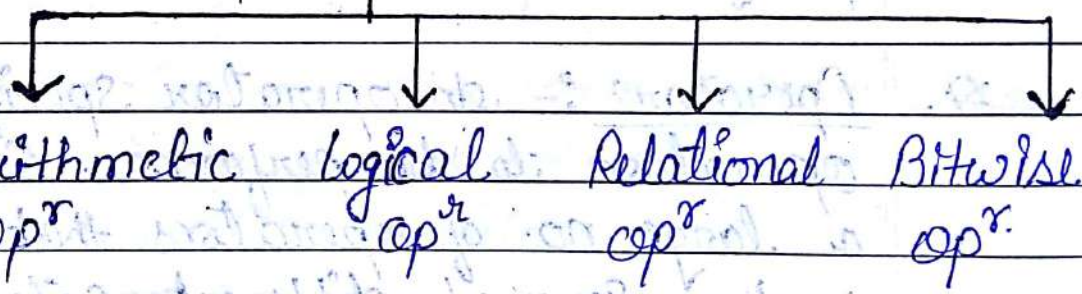
DATE: / /

1. UNARY Operators :-
An operator that act upon any on operand is known as UNARY operator.

✡. UNARY Minus :-
Any possible positive operand associated with UNARY MINUS operator gets its value changes to negative.

for ex :- a = 3.
b = 4.
c = a + (-b)
c = 3 - 4 = -1

2. BINARY Operator :-



2. BINARY op^r :-
(i) Arithmatic op^r :- Arithmetic op^r are used to perform the basic arithmatic operator operations. Such as addition, Subtraction, Multiplication, and division.

operator	Purpose
+	Addition
−	Subtraction
X	Multiplication
/	division

DATE: / /

(ii) **Relational opr:-** These are used to compare two operands. They define the relationship existing between two constants or variables.
They result in either a true or false value. The value of true is non zero (1) and the value of false is (0).

Operator	Meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	equal to
!=	is not equal to

(iii) **Logical operators:-**
This class of C operators is used to take decisions these are three such operators they are
AND
OR
NOT
The first two are binary operator and the last one is unary operator. The result of these operations either true or false.

PAGE:
DATE: / /

Operator	Meaning	Precedence	Associativity
&& (Double Ampersand)	Logical AND	2	Left to Right
|| (Double pipline)	Logical OR	3	"
! Enclamation	Logical NOT	1	"

(iv) Bitwise operator :-

All data items are stored in the computer memory as a sequence of bits (0, 1). C has the ability to support the manuplation of data at the bit level. Bitwise operator are used for operations on individual bits ; Bitwise operator, oprande on integer only. The bitwise op^r are as:-

Bitwise operator	Meaning.
&	Bitwise AND (Ampersand)
|	Bitwise OR (Pipeline)
~	One's compliment (Tilde)
<<	Left SHift
>>	Right SHift
^	Bitwise XOR (CARET).

PAGE:
DATE: / /

Imp 3. **TERNARY operator :- (conditional opr):-**

The Ternary opr is a conditional opr in C. There is the only one opr in C. It takes three operands. The Symbole "?" is use as ternary operator in C.

<expression>?<value 1>:<value 2>;

↳ Relational Expression.

Example :-

C = a<b ? a : b;

Here C will be Sign the value of a if a is less than b otherwise it will assign the value of b.

UNIT=3

Control Statements :-

In C program statements are executed sequentially in the order in which they appear in the program. But sometime we may want to use a condition for executing only a part of program.

Control Statement enable us to specify the order in which the various instructions in the program to be executed.

Control Statement define how the control is transfer to other part of the program.

PAGE:
DATE: / /

C language support four types of control statement which are as.

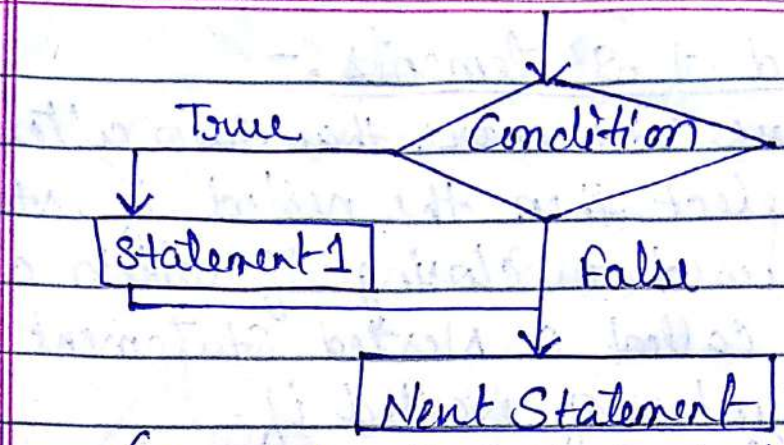
1. If-else.
2. go-to
3. Switch
4. loop (while, do-while, for).

If-else :- This is a by directional conditional control statements. This statements is used to test a condition and take one of the two possible action.

If the condition is true then a single statement or a block of statements is executed (one part of the program). otherwise another single statements or a block of statements is executed (other part of the programe).

```
If (Condition)
  {
  Statements;
  }
```

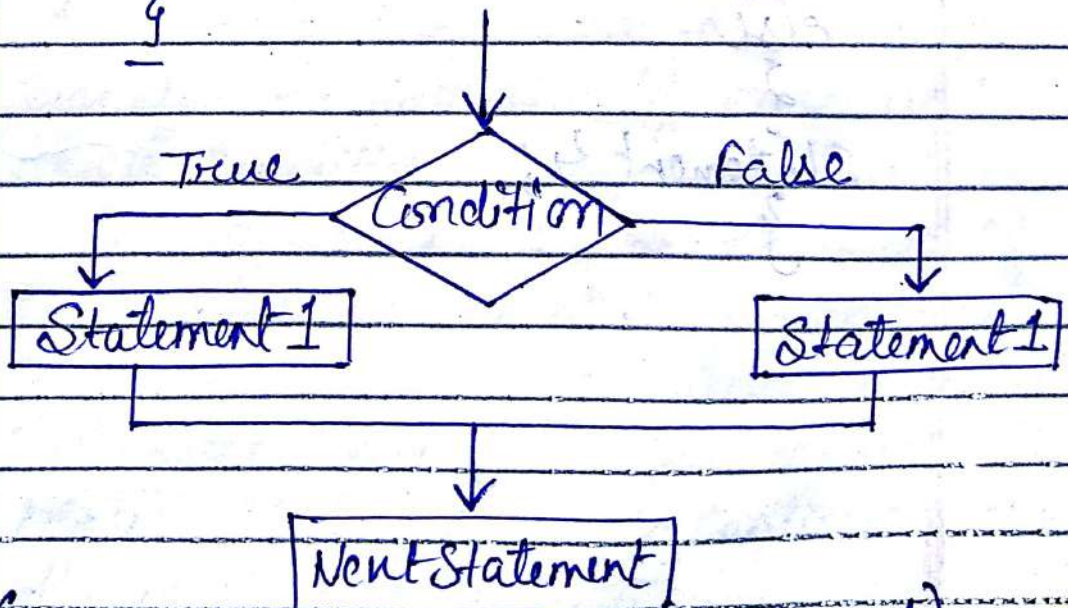
There can be a single statement or a block of statement after If part.



(Flow chart of if control statement)

Here if condition is true then statement 1 is executed. if it is false the next statement which is immediately after the if control statement is executed.

```
if (Condition)
  {
Statements;
  }
else
  {
Statements;
  }
```



(Flow chart of if else control statement)

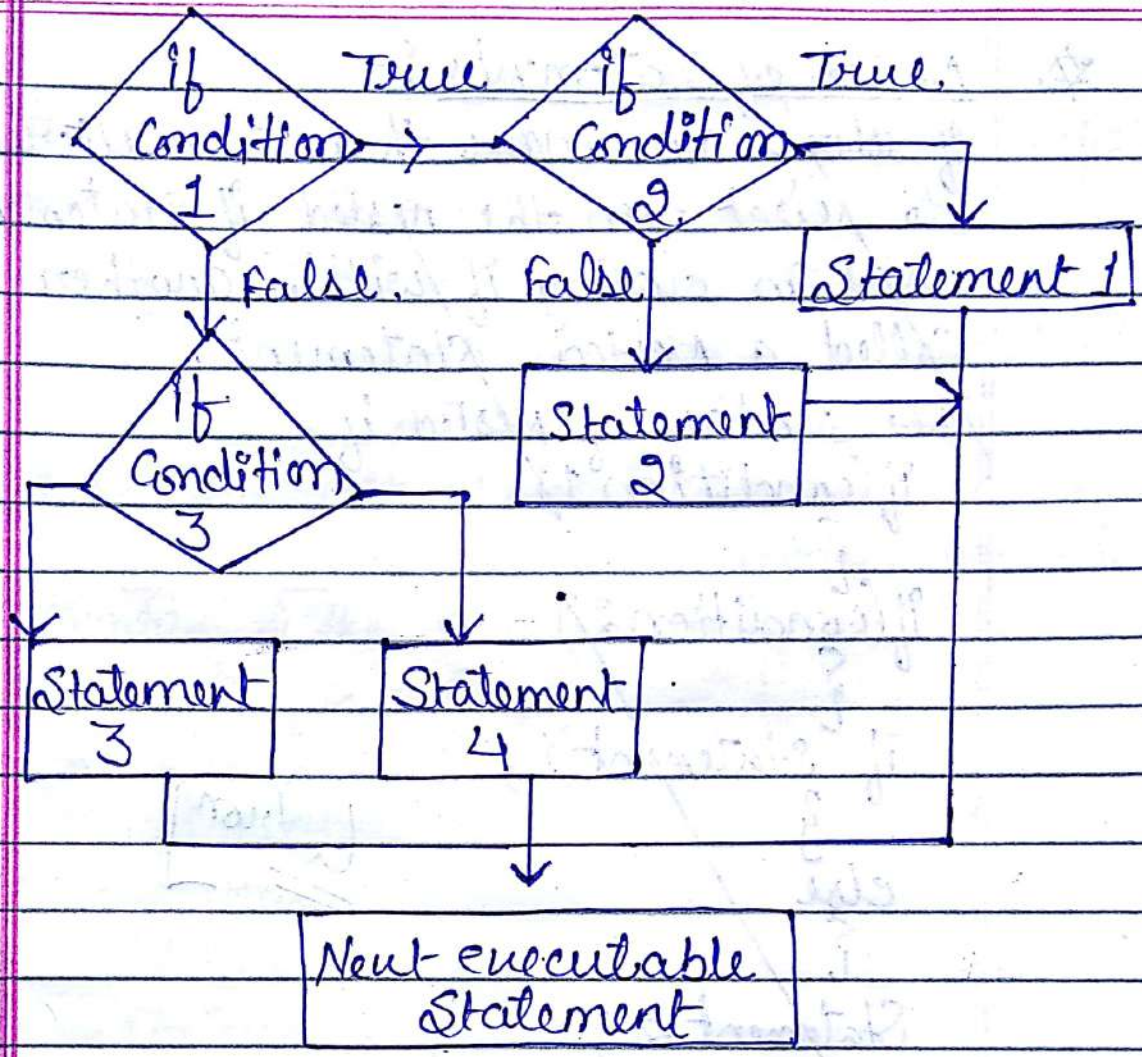
PAGE:
DATE: / /

✡. Nested of Statements :-

If there are more than two alteranative to select then the nested if statement are used in closing if within another if is called a Nested Statement.

The Syntax of Nested if.

```
if (condition 1)
  {
if (Condition 2)
  {
if Statement 1;
  else
  {
Statement 2;
  }
  }
else if (condition 3)
  {
Statement 3;
  }
  else
  {
Statement 4;
  }
```



Statement 1 is executed if condition 1 or condition 2 is true. If condition 1 is true and condition 2 is false statement 2 is executed.

If condition 1 is false the control transfer to the else if part and check condition 3. If condition 3 is true statement 3 is executed if it is false statement 4 is executed if both condition 1 and condition 3 are false control comes out of this Nested Statement and continues with the next statement.

PAGE:
DATE: / /

✡ Switch-Case Statement :-

The switch case statement is a multiple branch selection statement which test the value of an expression against list of integer, or constants. A switch statement allow you to select a block of statement, several altranatives statement general syntax of the switch statement is.

```
Switch (expression)
  {
Case value 1;
  Statement ;
  Statement;
   break ;
 Case value 2 ;
  Statement ;
  Statement ;
   break;
  ===
Case value n;
  Statement;
  Statement;
  break ;
  }
```

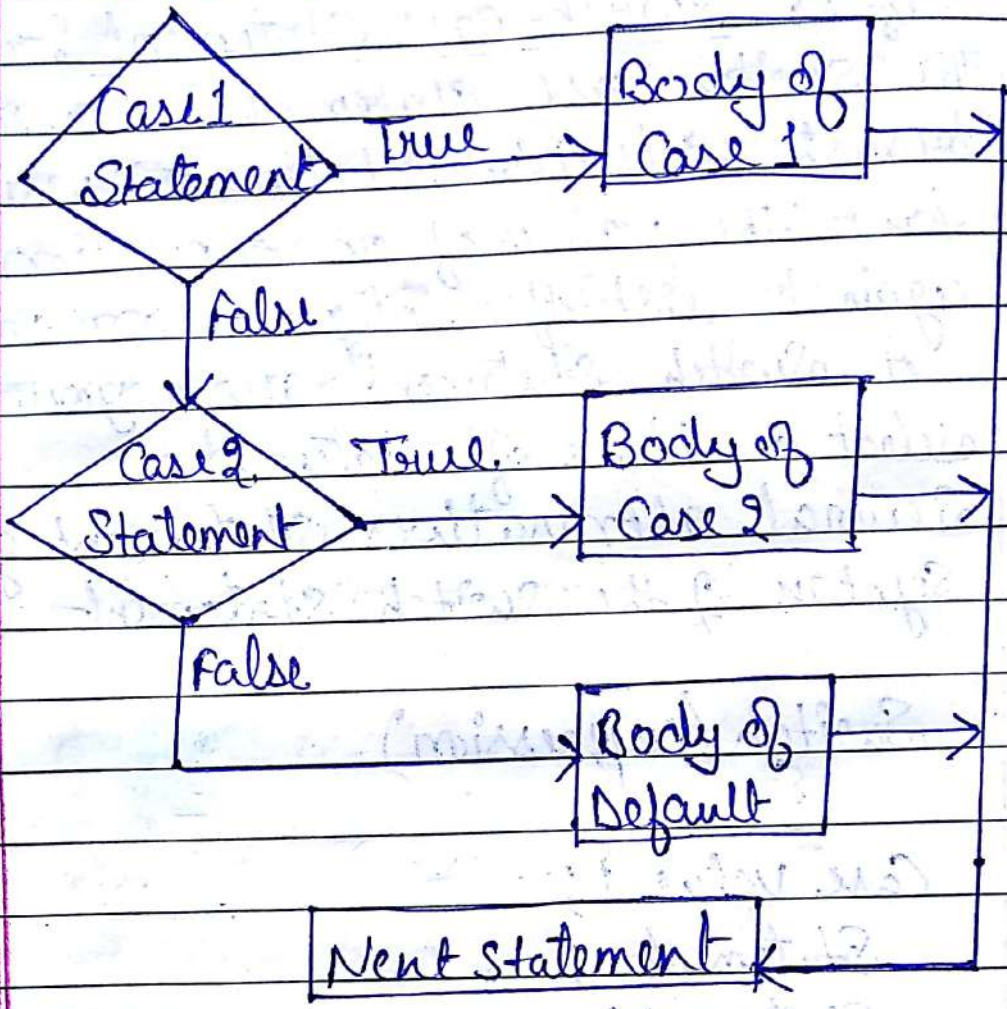


Fig:- Flow chart of switch case statement.

✡ LOOPS :- loops are use when we want to execute a part of the program or a block of statements several times.

for example :- Suppose we want to print "C is the best" ten times. One way to get the desire output we can write ten Printf statement which is not preferable otherway is used loop.

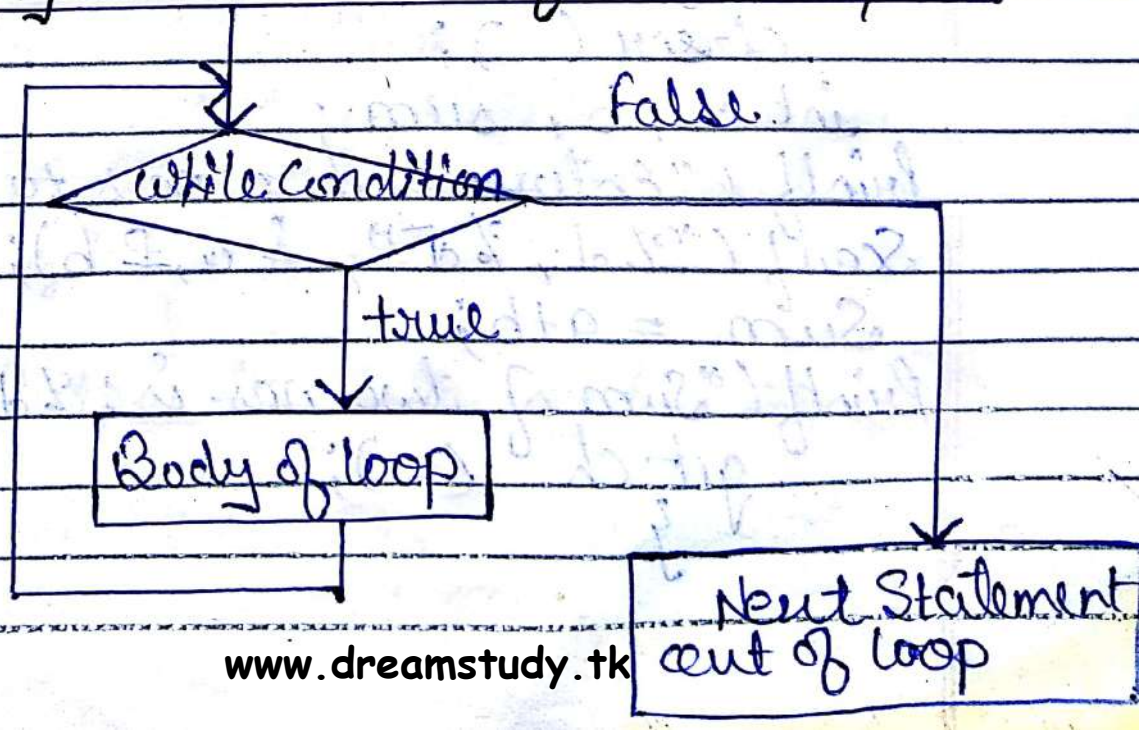
using loop we can write one

loop statement and one printf statement.
And this approach is definitly best bitter than the first one.
There are three loop statements in C.
i) while
ii) do-while
iii) for

1). **While loop**:- The while statement can be written as:-

```
while (condition)
  {
Statement;
Statement;
  }
```

Fig:- Flow chart of while loop.



PAGE:
DATE: / /

Programs

Program 1:- Write a program to print the word "Hellow world".

```
#include<stdio.h>
#include<conio.h>
    void main()
    {
Printf("Hellow world");
    getch();
    }
```

Output:- "Hellow world"

Program 2:- Write a program adding of two numbers.

```
#include<stdio.h>
#include<conio.h>
    void main()
    {
    clrscr();
  int a, b, sum;
Printf("Enter any two no.s to Sum");
Scanf("%d, %d", &a, &b);
  Sum = a+b;
Printf("Sum of two nos. is: %d", Sum);
    getch();
    }
```

PAGE :
DATE : / /

Output :-

Enter any two nos. 45
55

Sum : 100.

Program 3 :- W.A.P to input two nos. and then add, Subtract, Multiply and divide them.

```
# include <stdio.h>
# include <conio.h>
    void main ( ).
    {
    clrscr ( );
int a, b, addi, Subtract, Mult, Div;
Printf ("Enter any two nos.");
Scanf ("%d %d", &a, &b);
  Sum = a+b;
  Sub = a-b;
  Mult = a*b;
  Div = a/b;
Printf ("addition: %d \n, Subtraction:
      %d \n, Multiplication: %d \n; division
      %d \n", addi, sub, mult, div);
getch ( );
    }
```

Output :- Enter any two nos. 10
5

addition : 15 , Subtraction : 5
Multiplication : 50 , division : 2

PAGE:
DATE: / /

Program 4:- W.A.P. to Calculate the Volume of box.

```
#include <stdio.h>
#include <conio.h>
    void main()
    {
clrscr();
float l, b, h, V;
Printf("Enter the length, breadth and
        height of the box");
Scanf("%f %f %f", &l, &b, &h);

V = l * b * h.
Printf("Volume of the box: %f", V);
    getch();
    }
```

Output:-

Enter the length, breadth, height of the box 5
3
4
Volume = 60

PAGE :
DATE : / /

Program 5 :- Write a program to calculate the area of a circle.

```
#include <stdio.h>
#include <conio.h>
void main( )
{
clrscr ( );
float pie = 3.14, r, area;
printf ("Enter the value of radius");
scanf ("%f", &r);
area = pie * r * r
Printf ("area of the circle : %f", area);
getch ( );
}
```

Output :-

Enter the value of radius 2
area of circle = 12.56

Program 6 :- W.A.P. to check whether the no. is even or odd.

```
#include <stdio.h>
#include <conio.h>
void main( )
{
clrscr ( );
int n;
Printf ("Enter the value n");
```

```
scanf ("%d", &n);
if (n%2 == 0);
  Printf ("No. is even");
Printf ("No. is odd");
getch ( );
  }
```

Output :-

Enter the value of n = 7.
No. is odd.

Program 7 :- W.A.P. to print the nos. from 1 to 10 using while loop

```
#include <stdio.h>.
#include <conio.h>.
  Void main ( )
    {
int i = 1;
While (i<= 10).
  {

Printf ("%d", i);
 i= i+1;
Printf ("\n");
 getch ( );.
  }
  }
```

PAGE :
DATE : / /

Output :-

1
2
3
4
5
6
7
8
9
10

Program 8 :- W.A.P. to print nos. in reverse order with a difference of 2.

```
#include <stdio.h>
#include <conio.h>
  void main( )
   {
int i= 20;
while (i>=2)
  {
printf("%d", i);
i= i-2;
printf("\n");
getch( );
  }
}
```

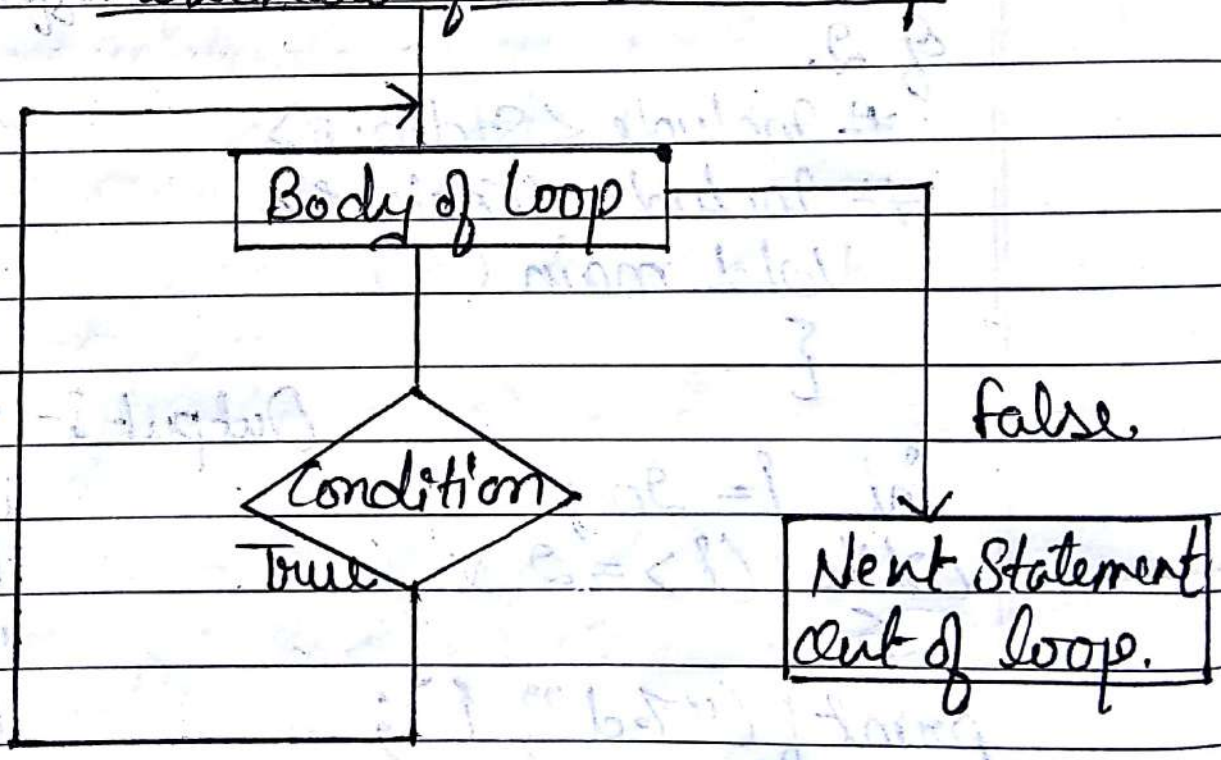
Output :- 20
18
16
14
12
10
8
6
4
2

PAGE:
DATE: / /

2. **do-while loop:-** The do-while statement is also use for looping. The body of this loop may contain a single statement or a block of statement. The Syntax of writting this loop is:

```
do
{
Statement;
Statement;
}
while(condition);
```

Flowchart of do-while loop.



Here firstly the statements inside loop body are executed and then the condition is evaluated.

PAGE:
DATE: / /

If the condition is true then again the loop body is executed and this process continue until the condition becomes false.
Note that unlike while-loop here a semicolon is placed after the condition.

Program 9:- W.A.P. to print 1 to 10 no. using do-while loop.

```
#include <stdio.h>
#include <conio.h>
void main ( )
{
int i=1;
do
{
printf ("%d", i);
i=i+1;
printf ("\n");
}
while (i<=10);
getch ( );
}
```

Output:-

1
2
3
4
5
6
7
8
9
10.

PAGE:
DATE: / /

3. **For loop**:- The for statement is very useful while programing in C. It has three expressions and semicolon's are use for seperating these expressions. The for statement can be written as:

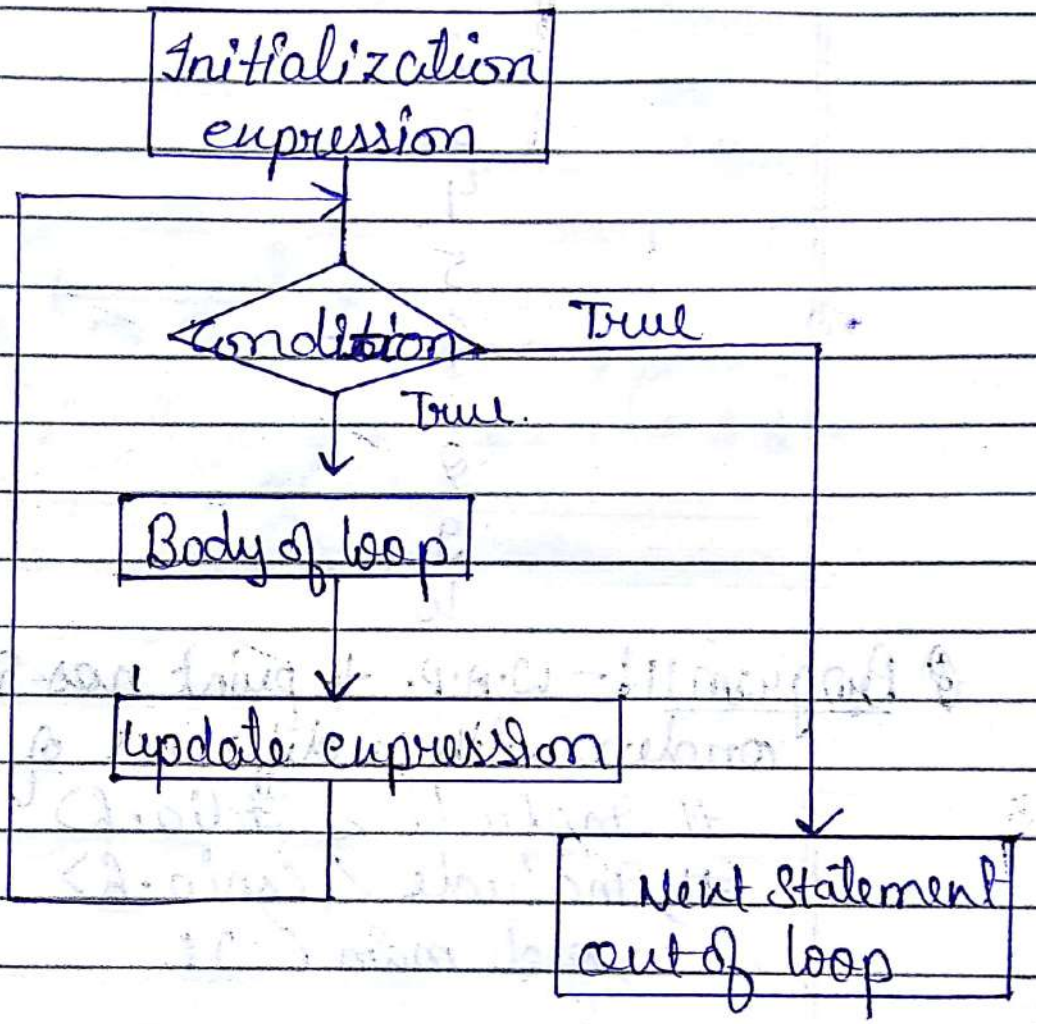
for(expression1; expression2; expression3)
    statement.

Expression 1 is an initialization of expression.
Expression 2 is a test expression.
Expression 3 is an update expression.

Expression 1 is executed when the loop start and is used to initalize loop. variables.
Expression 2 is a condition and is tested before each iteration of the loop. This condition generally use relational and logical operator.
Expression 3 is an update expression and is executed each time after the body of loop is executed.

PAGE:
DATE: / /

Flowchart of for loop:-



Program 1:- W.A.P. to print the no. from 1-to using for loop.

```
#include <stdio.h>
#include <conio.h>
    void main()
    {
  clrscr();
  int i;
  for(i=1; i<=10; i++)
    printf("%d", i);
    printf("\n");
        getch();
    }
```

PAGE:
DATE: / /

Output:-

1
2
3
4
5
6
7
8
9
10.

Program 11:- W.A.P. to print nos in reverse order with a difference of 2.

```
#include <stdio.h>
#include <conio.h>
void main ( )
{
clrscr ( );
int i;
for (i=10; i>=10; i=(-2)
printf ("%d", i);
printf ("\n");
getch ( );
}
```

Output:-

10
8
6
4
2

PAGE :
DATE : / /

UNIT - 4

PAGE:
DATE: / /

Algorithm:- To write a logical step by step method to solve the problem is a pro called algorithm. In other words algorithm is a processor for solving problems. An Algorithm include calculations, Reasoning and data processing.

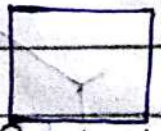
Flowchart:- A flow chart is the graphical or pictorial representation of an Algorithm. with the help of different Symbols. with algorithms we can usualy understand a program the main purpose of a flowchart is to analys different process.

Flowchart:-

* Terminal Box (Start and End).

* Input / output

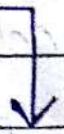
* Process / Instructions



* Decision

PAGE:
DATE: / /

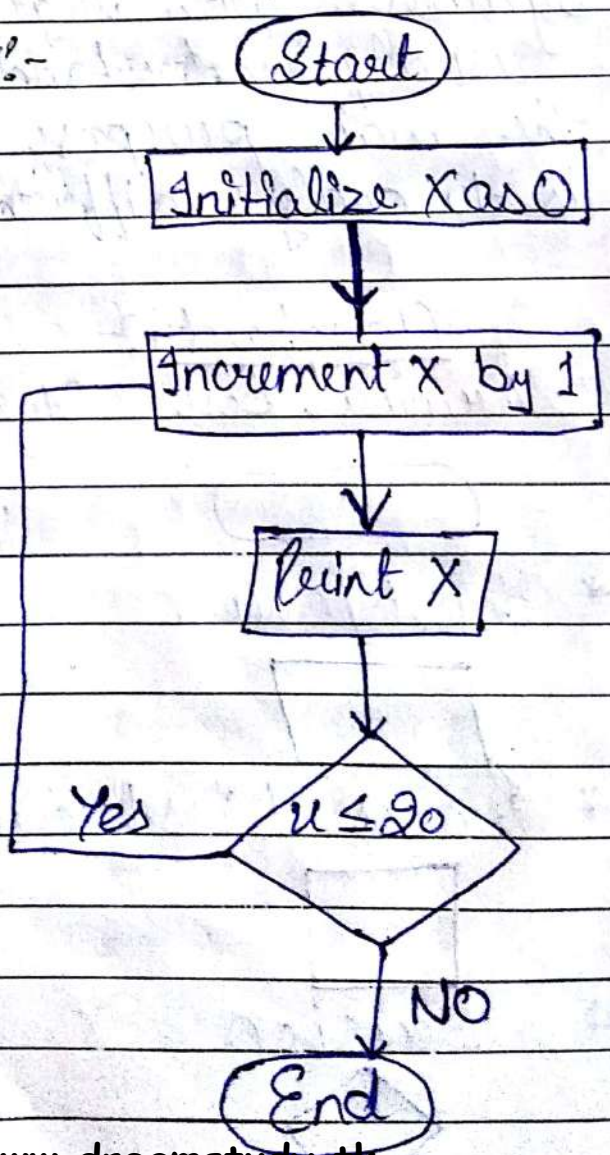
* Connector / Arrow



Example :-
Write a Algorithm to print 1 to 20

Algo :-
Step 1 :- Initialize X as 0,
Step 2 :- Increment X by 1,
Step 3 :- Print X,
Step 4 :- If X is less than or equal to 20 then go back to step 2.

Flowchart :-



PAGE :
DATE : / /

Program 12 :- W.A.P to find the sum of first n no numbers :-

```
#Include <Stdio.h>
#Include <Conio.h>
    Void main ( )
      {
    clrscr ( );
int n, i, sum;
Sum = 0;
print ("Enter the value of n");
Scanf ("%d", &n);
for (i=1; i<=n; i++).
  {
Sum = Sum + i;
  }
printf ("Sum of first n numbers : %d",
                Sum);
getch ( );
  }
```

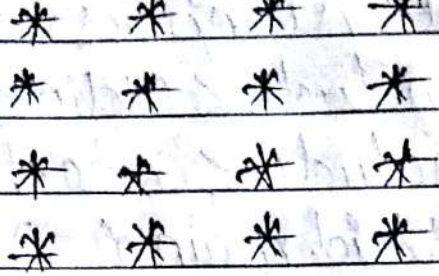
Output :-

enter the value of n 5

Sum of first n numbers is 15

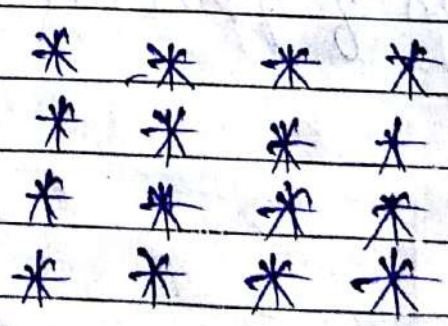
PAGE:
DATE: / /

Program 13:- W.A.P. to print the following pattern



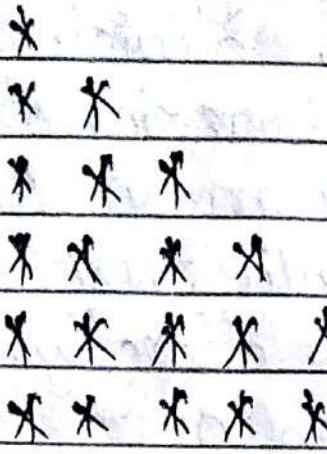
```
# Include <stdio.h>
# Include <conio.h>
  void main ( )
    {
  clrscr ( );
int i, j;
for (i=1; i<=4; i++)
   {
  for (j=1; j<=4; j++)
    {
  printf ("*");
    }
  printf ("\n");
    }

  getch ( );
   }.
```

Output:-



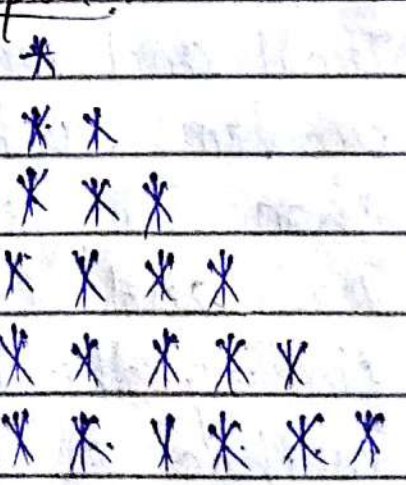
PAGE:
DATE: / /

Program 14:- W.A.P. to print the following pattern:-



```
#include <stdio.h>
#include <conio.h>
void main ()
{
clrscr ();
int i, j;
for (i=1; i<=6; i++)
{
for (j=1; j<=i; j++)
{
printf ("*");
}
printf ("\n");
}
getch ();
}
```

Output:-



PAGE:
DATE: / /

* **Divide and Conquer Algorithms :-**
In Computer Science, divide and Conquer Algorithm work by breaking down a problem into two or more sub-problems of the same time until these become simple to be solve directly. The solution to subproblem are then combined to give a solution to the original problem.
This technique is the basis of efficient algorithms for all kinds of problems such as sorting.
eg:- Quick sort, Merge sort

* **Trail and error algorithm :-**
Some complex problem can be solve by a technique that is called trail and error.
Trail and error is typically good for problems. where you have multiple chance to get the correct solution.
Trail and error is not a method of finding the best solution is it a method of finding all solutions.
One of the most powerful advantage of these techniques is that it does not require you have a

lots of knowledge. Trail and error is used to discover new things and it also play and important role in the scientific method as well.

* Steps in problem solving :-

There are seven main steps to follow when trying to solve a problem. These steps are as follow -

i) Define and Identify the problem.
2) Analyze the problem.
3) Identify possible solutions
4) Selecting the best solutions
5) Evaluating the solution
6) Develop an action plan.
7) Implement the solution.

1- Define and Identify the problem - The first step is critical. It is essential for each group member to clearly understand the problem, so that all energy will be focused in the same direction. A good way to define the problem is to write down a concise statement which summerize the problem.

2- Analyze the problem - In this stage of problem solving questions should be

PAGE:
DATE: / /

asked and information gathered. Think about how the issue affects others. It is essential to spend more time researching the problem.

3- Identifying Possible Solutions - Identifying possible solutions to the problem is sometimes referred to as finding "optional solutions", because the goal is to complete a list of all conceivable alternatives to the problem. Using the variety of creative techniques, group participants create an extensive list of problem solution.

4- Slecting the best Solutions - The starting point in any conscious attempt at rational decision making must be the recognition that a problem exists. While effective meetings are essential to getting work done, most meetings leave us still looking for a decision.

5- Evaluating Solutions - There are several ways to evaluate the chosen solutions. It is often helpful to make a T-Chart and ask members of the group to

PAGE:
DATE: / /

name the pros and cons of each solutions. This method will visually illustrate the strengths and weakness of each solution.

6- Develop an action plan - An action plan is a chart that lists the tasks that need to be done and identifies who will be responsible for each, when and what action is necessary, where to start, and how.

7- Implement the solution - Sometimes the groups who choose the solution are not the ones who will implement it. If this is the case, members who select the solution should clearly explain why they selected it to the ones who will implement it.

Unit – V	Simple Arithmetic Problems	Addition / Multiplication of integers, Determining if a number is +ve / -ve / even / odd, Maximum of 2 numbers, 3 numbers, Sum of first n numbers, given n numbers, Integer division, Digit reversing, Table generation for n, a^b, Factorial, sine series, cosine series, ${}^nC_r$, Pascal Triangle, Prime number, Factors of a number, Other problems such as Perfect number, GCD numbers etc (Write algorithms and draw flowchart), Swapping

PAGE:
DATE: / /

UNIT-6th

Functioning :-

A function is a self contain sub program that is mean to do some specific, well defined task.

A C program consists of a one or more functions. If a program has only one function then it must be the main ()

C programs have two types of function

a) library funct^n b) user defined funct^n.

1- Library funct^n — These functions are present in the C Library and they are pre-defined.

For example — To use input output funct^n like Printf ( ), Scanf ( ), We have to include stdio.h for mathematical library ( ) we have to include math.h

for string library() we have to include string.h.

2- User-defined functions :-
User can create there own function for performing any specific task of the program. These types of funct^n are called user-defined functions. To create and use these functions we should know about these three things:-

a)- Function definition.
b)- Function declaration
c)- Function calling.

a- Function definition - The function definition consist of the whole discription and code of a funct^n. It tells what the funct^n is doing and what are its input and output. A function definition consist of two parts-

i) a function header ii) function body.

The general syntax of a function definition is.

```
return-type func_name(type1 arg1, type2 arg2 ---- )
{
local variables declarations;
    Statement;
    ---------
return (expression);
}
```

PAGE :
DATE : / /

The first line in the funct^n definition is known as the funct^n header and after this body of this function is return enclosed in curly braces.
The return-type denotes the type of value that will return by the function.
The return-type is optional and if omitted it is assume to be int by default. If a funct^n does not return any value then void should be written in place of return-type.
func-name specify the name of the funct^n and it can be any valid C identifier.
The body of funct^n is a compound statement which consist of declaration of variable and statements followed by and optional return statement.

Ex-
```
int sum (int x, int y).
  {
  int S;
  S = x+y;
  return S;
  }
```

(b) - Function Call :-
A function is called by simply writing its name follow by the argument list inside the parenthesis.

PAGE:
DATE: / /

Syntax → func_name (arg1, arg2, arg3).

These arguments arg1, arg2, are called actual arguments.

Here func_name is known as the called function.

ex → main() is the calling function.
and sum() is the called function.
and a, b are the actual arguments.

2- Function declaration-

The calling function needs information about the called function. If definition of the called function is place before the calling function the declration is not needed. But generaly the function are main is place at the top and all other function are place after it. In this case function declaration is needed.

Function declaration is also known as the function prototype And it inform the compiler about the following three things-

a) Name of the function.
b) Name and type of any argument received by the function.
c) Type of value written by the funct^n.

PAGE:
DATE: / /

The general Syntax of function declaration is.

return_type func_type (type1 arg1, type2 arg2 ----).

Q. W.A.P. to check whether the no. is even or odd.

```
#Include <Stdio.h>.
#Include <Conio.h>.
Void ~~no~~ find (Int n);
  main ( ).
   {
int num;
printf ("Enter any no.");
Scanf ("%d", &n);
find (num);
  }
Void find (Int n).
  {
If (n%2==0).
printf ("No. is even");
 else
printf ("No. is odd");
  getch ( );
  }.
```

PAGE:
DATE: / /

Q. W.A.P. that find the larger of two nos.

```
#include <stdio.h>
#include <conio.h>
int max(int a, int b);
  main()
int x, y;
printf("Enter the value of x and y");
scanf("%d %d", &x, &y);
  max(x, y);
  }
int max(int a, int b);
  {
if (a>b)
return x; printf("a is greater");
else
return y; printf("b is greater");
getch();   getch();
  }
```

A. Local variables - The variables that are defined within the body of a function or a block and are local to that function are called local variables.

For eg:-

```
func()
  {
int a, b;
  }
```

PAGE:
DATE: / /

Here a and b are local variables which are defined within the body of the func().

✡. Global variable - The variables that are define outside any funct^n are called global variable. All functions in the program can access and modify global variables. It is useful to declare a variable global if it is to be use by many funct^n in the program.

✡. Static variable - Static variable are declare by writting keyword static infront of the declaration.

Static type var__name;

Eg:- Static int x;

Amf ✡. Actual argument / parameter - The argument which are mention in the function call are known as actual argument. Science they are the values which are actually sent to the called function.

Amf ✡. Formal arguments - The name of the arguments which are mention in the function definition are called formal dummy arguments.

PAGE:
DATE: / /

Since they are use just to hold the values that are sent by the calling function.

Eg:- void disp (int n);
main ( ) → Formal Argument.
{
int a;
disp(a);
→ Actual argument

If a function take any arguments it must declare variables that accept the value as arguments.

There are two ways to pass value to function in C language which is given below -

1. Call by value.
2. Call by reference.

1. Call by value - In call by value original value cannot be change or modify. In call by value when you pass value to the function it is local store by the function parameter in stack memory location.

```
void disp (int n);
  main ()
   {
int a = 100;
disp(a);
```

PAGE:
DATE: / /

```
Void disp (int n);
  {
  n = n + 100;
  - - - -
  - - - -
  }
```

2- Call by Reference.-

In call by reference original value is changes or modify because we pass reference (address). Address of the value is pass in the function so ~~action~~ actual and formal arguments share the same address space.

```
void disp (int *n);
  main ( )
    {
int a = 100;
 disp (&a);
void disp (int *n)
*n = *n + 100;
    }
  }
```

PAGE:
DATE: / /

Difference b/w Call by value and Call by Reference –

	Call by Value	Call by Reference
i)	This method copy original value into funct^n as a argument	i) This method copy address of arguments into funct^n as arguments
ii)	Actual and formal arguments will be created in different memory location	ii) Actual and formal argument will be created in same memory location.

Program:- W.A.P. to interchange nos. using function (By call by value method).

```
#include <stdio.h>
#include<conio.h>
void interchange(int x, int y);
  main( ).
    {
clrscr( );
int a, b;
printf("Enter any two nos.");
scanf("%d,%d", &a, &b);
interchange(a, b);
printf("\n value of a and b after
  interchange \n a=%d \n b=%d", a, b);
  getch( );
    }
```

```
void Interchange (int x, int y)
  {
printf ("value of x and y before Interchang
        \n x = %d \n y = %d", x, y);
int temp;
 temp = x;
 x = y;
 y = temp;
printf ("\n value of x and y after
        Interchanging : \n x = %d \n
        y = %d", x, y);
 }
```

Output:- Enter any two nos. 5 and 6
value of x and y before Interchanging
x = 5
y = 6
value of x and y after interchanging
x = 6
y = 5
value of a and b after Interchanging
a = 5,
b = 6.

PAGE:
DATE: / /

Program:- W.A.P. to find factorial of a no.

```
#Include <Stdio.h>
#Include<Conio.h>
Void main()
 {
 clrscr();
int n, fact=1;
printf("Enter any no. to find factorial");
Scanf("%d", &n);
 while(n!=1)
 {
fact= fact * n;
n--;
printf("factorial is: %d",fact);
  getch();
   }
```

Output :-

Enter any no. to find factorial - 5
Factorial is 120.

Program:- W.A.P. to find ~~factorial~~ the squart root of any number -

```
# Include<Stdio.h>
# Include<Math.h>
# Include.<Conio.h>
  Void main()
   {
double n, s;
```

```
printf("Enter any no.");
Scanf("%lf", &n);
  S= Sqrt(n);
printf("value of s is : %lf", S);
      getch();
  }
```

✱. Time complexity:- Time complexity depend upon the size of inputs thus it is a function of input size "n".
It should be noted that different time can arise for the same algorithm.
Usually we deals with.

1) Best case time
2) Average case time
3) Worst case time

1- Best case time- The minimum amount of time that can algorithm require for an input of size "n" is refer to as best case time complexity.

2- Average case time- It is execution of an algorithm having typing input data of size "n".

3- Worst case time complexity- The maximum amount of time needed by an algorithm for an input of size "n" is refer to

PAGE:
DATE: / /

as worst case time complexity.

✱. Space Complexity- The amount of memory needed by program to run to completion is refer to as space complexity.

✱. Recursion- Recursion is a powerful technique of writing a complicated algorithm in an easy way.
According to this technique a problem is define in terms of itself.
The function that all itself (inside function body) again and again is known as recursive function. In recursion calling function and the called function are same.

Ex:-
```
int sub(int n)
int i, a;
_____
_____
i = sub(a);
    └──> Recursion.
```

PAGE:
DATE: / /

Brainstorming In decision making -

Brainstorming is a process of for developing creative solutions to problems. Critism of ideas are withheld during the brainstorming session as the purpose is on generating varied and unusual ideals and extending or adding to these ideas. Critism is reserved for the evaluation stage of the process.

Brainstorming works by focussing on a problem and then deliberating coming up with as may solutions as possible and by the pushing the ideas as far as possible.

Pseudocode -

Pseudocode is a kind of structural english for describing algorithms. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax. At the same time, the pseudocode needs to be complete. It describe the entire logic of the algorithm so that Implementation becomes a rote mechanical task of translating line by line into source code.

Extract the next word from the line (good)
Set word to get next token (poor).

PAGE:
DATE: / /

Append the file extension to the name(good)
name = name + extension (poor).

For all the characters in the name(good)
for character = first to last (ok).

2o Programs - W.A.P. find a factorial no. using recursion :-

```
# Include <Stdio.h>
# Include <conio.h>
int fact (int no);
  main ( )
    {
clrscr ( );
int no, ans;
printf ("Enter a number");
scanf ("%d", & no.);
ans = fact (no);
printf ("factorial is : %d", ans);
getch ( );
}

int fact (int n)
{
int f;
if (n == 0)
{
return 1;
}
```

```
else
{
f = n * fact(n-1);
return f;
}
}
```

Output:- Enter a number
5
factorial is : 120.

Big oh Notation:- denoted by (O) upper bound.

Definition:- $f(n) = O(g(n))$.
If and only if there exist two positive Constant C and no. such that
$|f(n)| \leq C|g(n)|$ and $n \geq no$.
↳ Real constant.

Example:-
Let $f(n) = 3n+3$.
$g(n) = n$.
$3n+3 = O(n)$.
$|f(n)| \leq C\,|g(n)|$
$|3n+3| \leq C\,n$.

$n=1$	$C=4$	$6 \leq 4$ X
$n=2$	$C=4$	$9 \leq 8$ X
$n=3$	$C=4$	$12 \leq 12$ ✓
$n=4$	$C=4$	$15 \leq 16$ ✓

PAGE:
DATE: / /

Program 21:- W.A.P. to check no. is armstrong or not.

```
#Include<Stdio.h>
#Include<conio.h>
 Void main()
  {
clrscr( );
int number, Sum=0, rem=0, Cube=0,
 temp;
printf ("Enter any number");
Scanf ("%d", &number);
 temp = number;
While (number=0)
 {
rem = number%10;
Cube = pow(rem3);
Sum = Sum+Cube;
number= number/10;
 }
if (Sum==temp)
printf ("No. is armstrong");
 else
printf ("No. is not armstrong");
getch( );
 }
```

Output:- Enter any no. 153.
No. is armstrong.