C - language

* Algorithm & Flowchart :-

Math + Electrical = computer

program :- set of instruction arranged in proper sequence.

Algorithm :- Mathematical way of writing the program.

algorithm & flowchart are language independent.

The steps in the algorithm are always numbered.

1st instruction is

i) start

& ends with

End.

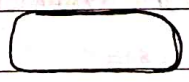
2) No needs of define variables in algorithms.
3) Printf() / promp are not in algorithms
4) The Instruction in algorithm are language independent.

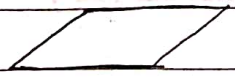
* Flowchart :-

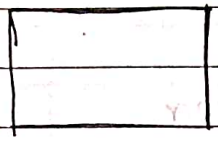
Representation of program in graphyical way.

* <u>understand the flow</u> :-

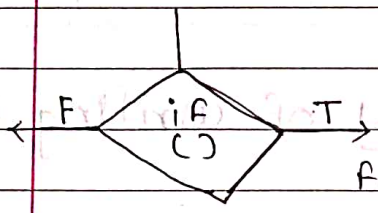
i) Symbols in flowchart are always unidirectional (only single directional)

 → start / End

 → perallelogram is used for showing I/o statements.

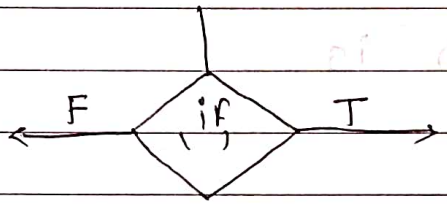


Rectangle is used for showing a processing statement.

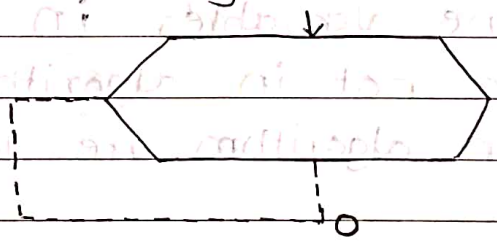


Diamond is used for if statement has incoming arrow one for T & one for F ∴ It is decision making statement.

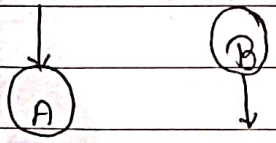
other use of diamond is while / do-while statement.



for representing for loop use hexagon.



Iterative statement :- for / while / do-while



connectors

Every connectors is labeled A to Z
To extends the program to another page program when flow chart is to big.

Note→ Symbols should be connected one another & every line so should be unidirectional.

* Introduction :-

i) The C language is imperative procedural & general purpose in nature was developed by Dennis M. Ritchie at AT&T Bell Laboratories in 1972.

ii) The C language is so east to learn & understand.

* The C character set :-

i) The c character set refers to a set of all valid characters so that use in the source program for forming words, expression & numbers that contains all characters.

ii) They include digits, alphabets, Special symbols etc.

iii) The C language provides support for 256 characters.

iv) Digits → 0 to 9

Alphabets → A to Z a to z

Symbols → ' ~ @ ! # $ ^ * %. & [] () { }
<> + = _ - / \ | ; : " , . ? $
and white space.

* C keywords :-

i) keywords are predefined, reserved word used that have special meanings to compilers.

ii) keywords are parts of syntax & they can't be used as an identifiers.

Following is the list of keywords present in C language.

auto	double	int	struct	Total : 32 keywords in C-language.
break	else	long	switch	
case	enum	register	typedof	
char	extern	return	union	
continue	for	signed	void	
const	ff	static	while	
do	goto	sizeof	volatile	
default	float	short	unsigned	

* C Identifiers :-

i) C Identifiers represent the name in the C program given to variables, function, array, structures, unions, labels etc.

ii) It can be composed of letters such as upper/lower case, underscore, digits, but the starting letter should be ~~opt~~ alphabet or an underscore.

Types :- i) Internal Identifiers
ii) External Identifiers.

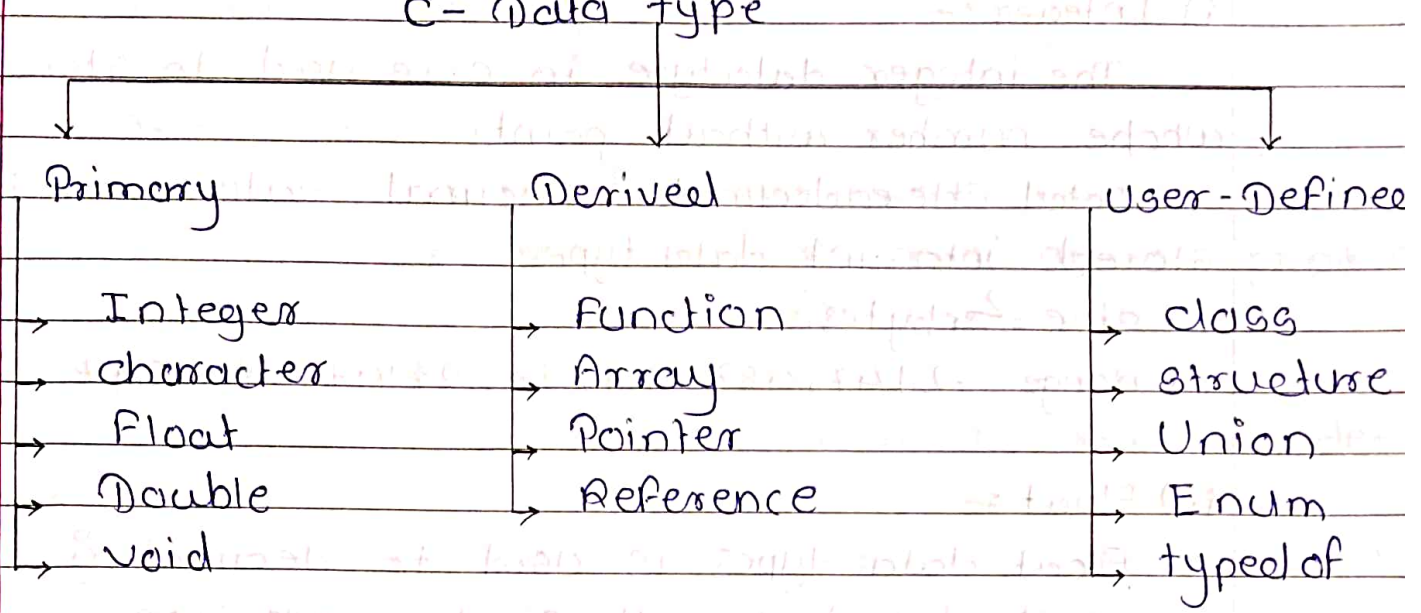
Rules :-

a) A valid identifiers can have letters, digits & underscore.

b) The first letter should be either a letter or an underscore.

c) you can't use keywords as a identifiers

* Data types.



i) Each variable in c has an associated data type. It specifies the type of data that variable can store like integer, character, float, double etc.

ii) Each data types require different amount of memory has some specific operations which can be performed over it.

A] Primary / Primitive data type :-

These are most basic data types that are used for representing simple values such as integer, float, characters etc.

B] Derived Types :-

The data types that are derived from primitive or built in data types are refered to as derived data types.

C] User Defined Types :-

The user defined data types are defined by the user himself.

A] primary Data types :-

i) Integer :-

The integer datatype in c is used to store the whole number without point.

Octal, Hexadecimal & Decimal values can be stored into int data type.

- size = 2 bytes.
- Range :- -2,147,483,648 to 2,147,483,647.

ii) Float :-

Float data types is used to decimal & exponential values with single precision.

- Range :- $1.2E^{-38}$ to $3.4E+38$
- size :- 4 bytes.

iii) Characters :-

Characters data type allows it's veriable to store only a single character.

- Range = (0 to 255) or (-128 to 127)
- size = 1 byte

iv) Double :-

A Double data types is used to store decimal number with double precision & capable of holding of 64 bits of decimal no. or Floating points.

- Range: ~~8 bytes~~ $1.7E^{-308}$ to $1.7E+308$
- size : 8 bytes

v) void :-

The void data type in c is used to specific that no value is present.

It doesn't provide result value to caller.

It has no values & no operations & is used to represent nothing.

* C variables :-

i) A variable is an entity whose values may or not may change during execution of program

* Rules for defining variables :-

(i) A variable can have alphabets, digits & under score.

(ii) A variable name can start with alphabets & underscore only. It can't start with digit.

iii) No whitespace is allowed within the variable name.

(iv) A variable name must not be any reserved word or keyword eg. int, float etc.

* Types of variables :-

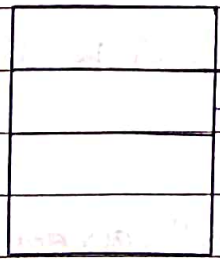
a] Integer variable :-

i) Keyword int is used to represent integer variable.

Syntax :- int list_of_variables;

ii) Variable should follow rules & identifiers.

e.g. int, more, min, sum, num;



→ more address in primary memory.

For computer it has specific

C compiler use 2 bytes (16 bits) to stored integer

b] Float variables:-
Float main, more, sum, num.
It requires 4 bytes & double requires 8 byte.

c] Character variables:-
char min, more, sum, num.
• 1 byte required to store in turbo-C.

Q. Find the area of ~~triangle~~ reet.

→
```
#include<stdio.h>
void main()
{
   int b,l, area;
   clrscr();
   printf("Interter the number for b & l : ");
   scanf("%d%d",&b,&l);
   area= b*l;
   printf("Area of ~~triangle~~ reet : %d", area);
   getch();
}
```

Q. Find the area of triangle:-

→
```
#include<stdio.h>
#include<conio.h>
void main()
{  int b,h, area; float area;
   clrscr();
   printf("Enterth number for b & h \n");
   scanf("%d%d", &b,&h);
   area= 1.0/2 * b * h;
   printf("Area of Triangle = %d \n", area);
   getch();
}
```

28/08/2023

* Constants:-
  i) Constant is an entity whose value is never be changed during execution of program
  ii) Constant is always have a datatype, there is no void type constant.
  iii) we declared a constant by using:-
    a) const keyword.
    b) #define preprocessor.

* Difference bet^n constant & variable:

Constant	Variable.
i) A constant is a variable / value that cannot be changed once it defined.	A variable is a name associated with some memory location.
ii) It hold the fixed value which we can retrieve later but cannot change.	A variable is used to hold some value that can be changed according to the requirement.
iii) constant are generally stored in the text segment as they are read only.	The variable are stored inside a data segment, heap or stack depending on the envirnment it is declared in.
iv) we can only assign a value to the constant while defining it.	we can assign value to the variable anytime.
v) constant can be defined by using #define or const keyword.	variable can only be defined using the standard variable defination syntax.
e.g. #define pi 3.14	e.g. ~~var~~ int var = 25;
const ~~int~~ float pi = 3.14;	var = 10;

a] Integer constant :-

i) Their is not allowed any symbol in integer constant.

e.g. const int a = $250; — Not valid.

const int a = 250; — valid.

ii) Integer variable take 2 byte storage but Integer constant take 4 byte storage.

b] Float constant :-

i) For Float constant their is decimal point value is mandatory.

e.g. const float a = 3.14;

ii) A float constant always stored double of float value.

∴ float variable - 4 byte

float constant - 8 byte.

c] Character constant :-

A character constant value must be in single quate,

e.g. const char a = 'A';

Note :-

Every character constant stored as a integer in complier. that's way it take 2 byte storage.

d] string constant :-

A string constant value must be in double quate.

e.g. const string a = "Ram";

e] Symbolic constant :-

We use name for value is called symbolic constant. Symbolic constant are created by using keyword "const";

e.g. const float pi = 3.14;

by default every symbolic constant is integer constant.

```
const a = 3;
```

here a is an integer constant.

* #define pi 3.14

There is complier define the type of constant on value assigned to constant which is followed by #define preProcesser. (Preprocessor = Before the execution of program.

we can use #define preprocessor anywhere of the program.

#define does not terminated.

e.g.

```
#include <stdio.h>
#define pi = 3.14
void main()
{
    int a=10;
    clrscr();

    #define pi = 3.14
    printf("%d", pi);      --- print 3.14
    getch();
}
```

* Scope of variables :-
i) Global scope of variable.
ii) Local scope of variable.

* Operator :-
Operations is are the special types of symbols those used performing various operation on /meaningful operation on operands.

* Types of operator :-
A] Arithmatic operators :-
i) Arithmatic operators are used to perform basic mathematical operation on operands.
∴ + → Addition
− → Substraction
* → Multiplication
/ → Division (Qit
% → Remainder.
ii) Every Arithmatic operators always binary in nature means, in every arithmatic operators required minimum two operands in expression.
e.g.

a + b = c
operands ↑ operands operands
operator

iii) Modulus (%) oprators are only used for interger operations.
some types of arithmatical operation
a) Integer arithimatical operation.
In which both the operands are integer & result also integer.

Integer value operator Integer value = Integer value

b) Real arithmatical operations :-
In which both the operands will be float & result of expression always float.
∴ Float value operator Float value = Float value.

c) Mixed arithmatical operations :-
In which one of operands is integer & another is ~~integer~~ float so that's way result is float.

∴ Float value operator Float value = Float value

B] Relational operator :-
It is also known as comparision operator. because the it is used to compare one or more operands.

i) > → greater
ii) >= → greater or equals to
iii) < → less than
iv) <= → less than or equals to
v) == → equal-equal to
vi) != → Not equal to

The result of relational operator is always written in integer, because it's always binary (1/0)

e.g. int a = 2, int b = 3;
printf("%d", a > b) ---- o/p = 0 - False

int c = 0
printf("%d", a < b > c) ---- o/p = 1 - true.

c] Logical operators :-

Logical operators are used for compound cond^n, Means. if the we want common result from multiple expression.

e.g.

cond^n 1 && cond^n 2

&& → And operator
|| → or operator
! → Nor operator.

Truth Tables.

a	b	a&&b	a||b	a	!b
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

a) && (And operator) :-

When the both cond^n of are true then result is true otherwise result is false.

b) || (OR operator) :

When minimum one operation (cond^n) is true the result is true.

c) ! (Not operator);

Its reverse the results.

True → False
False → True.

D) Ternary operators(conditional operator)

It use (? :) symbols only.

Ternary operator check the conditon when it's true the befor the ":" colon part is execut other wise after the colon is executer.

It ~~as~~ may or not to be assigned value to ~~vatu~~ variable.

```
expresion ? expresion 2 : expression 3 ;
conditon   (true)        (false)
```

a. WAP to find largest of two numbers using Ternary operators.

```
void main ()
    {
        int n1, n2, lar;
        clrscr();
        printf("\nIt Enter the two numbers : ");
        scanf("%d%d", &n1, &n2);

        lar = (n1>n2)? n1 : n2;
        printf("\nIt The largest number is %d ", lar);

        getch();
    }
```

E) Assignment operator :-

Assignment operator is used to assigne the value to result when left operand is equal to result operand :-

e.g. a = a+2;

Here we use assignment operator like:

a+=2;

i) `+=` → Addition
ii) `-=` → substraction
iii) `/=` → Division
iv) `*=` → multiplication.
v) `%=` → modulus.

e.g.
```
void main()
{
    int a=10;
    clrscr();
    printf("a %d \n", a);
    a+=2;
    printf("a %d \n", a);
    getch();
}
```

F) Increament / Decrement operator :-
When we want increase/decrease the value of integer variable by 1, then use Increament/ Decreament operator is.
Inc / Dec operator is only applied to variable (integer, float), not a constant.
`++` → Increament
`--` → Decreament
Here, this have two type also described.
i) post inc/dec (after the variable) → a++
ii) pre inc/dec (before the variable → ++a;

e.g of post inc/dec.
```
void main()
{
    int a=10, b;
    clrscr();
    b= a++;
```

```
printf("a : %d , b : %d ", a , b);
getch();
}
```

∴ Here frist value of a used to assign to b. then added it self by 1;

a = 11, b = 10',

e.g. on pre increcment.

```
void main()
    {
        int a = 10; b;
        clrscr();
        b = ++a;
        printf("a : %d , b : %d", a , b);
        getch();
    }
```

∴ Here the frist is increment in a by 1 then added to be.

∴ a = 11; b = 11;

6] Bitwise operator :- & , | , ^

Bitwise operator is very complex to understand. Their one three operator.

i) & → and = all cond^n one true = true

ii) | → or = at least one cond^n is true = t

iii) ^ → xor = when the both the one diffent then output true(

i) & Bitwise operator :-

e.g. int a = 10, b = 8 ;

then c = a & b = ?

Page No.:
Date:

weigth of bytes. ... 16 8 4 2 1

For a = 10 — 0 1 0 1 0
For b = 8 — 0 1 0 0 0
For c = a & b — 0 1 0 0 0

then c = 8 ;

ii) Bitwise or :-

e.g. int a = 10 ; b = 8 ;
then c = a | b ?

→

weigth of bytes. ... 16 8 4 2 1

For a = 10 — 0 1 0 1 0
For b = 8 — 0 1 0 0 0
then c = a | b — 0 1 0 1 0

c = 10 ;

iii) xor Bitwise :-

e.g. int a = 10 ; b = 8 ;
then c = a ^ b ?

→

weigth of bytes — 16 8 4 2 1

For a = 10 — 0 1 0 1 0
For b = 8 — 0 1 0 0 0
then c = a ^ b — 0 0 0 1 0

c = 2

IMP

Q. WAP to test wether the given number is odd or even by bitwise & ternary operator.

→
```
#include<conio.h>
#include<stdio.h>
void main()
{
    int a = 10;
    clrscr();

    (a&1) ? printf("odd no.") : printf("even no")
    getch();
}
```

Q. WAP to test wether the given number is odd or even by by using XOR operator.

→
```
#include<conio.h>
#include<stdio.h>
void main()
{
    int a=11;
    clrscr();

    (a^1)==(a+1) ? printf("even no."); printf("odd no.")
    getch();
}
```

	8	4	2	1
a	1	0	1	1
				1
a^1 =	1	0	1	1

12

	1	1	0	0
				1
	1	1	0	1

Date: 30/08/2023

H]

Shift operator :-

i) Shift operator is only used for integer data type variables for moving to left / right side it.

a) >> ⟶ right shift
b) << ⟶ left shift

e.g. of right shift

a = 10 ;

Byte weightage :- 32 16 8 4 2 1

a = 10 0 0 1 0 1 0

If a <<>> 2

then add two zero value at last position & shifted starting of two zero.

∴ 1 0 1 0 [0 0] ← added.

a >>

a<<2 = 40

If ~~a <<~~ a>>2.

then 0 0 1 0 [1 0] ← shifted.

a>>2 = 2.

i] Special operators :-

a] Comma operators :-

It is used grouping of some element which homogenous in nature.

Ex. WAP to swap values of two variables :-

```
#include< stdio.h>
#include < conio.h>
void main()
    {
        int a, b, t;
        clrscr();
        printf("Enter two numbers : ");
        scanf("%d%d", &a, &b);

        t=a, a=b, b=t ; -- // use of comma operator

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

IMP→ Some swaping techniqus :-

i)	ii)	iii)
a = a+b	a = a*b	a = a^b
b = a-b	b = a/b	b = a^b
a = a-b	a = a/b	a = a^b

b] sizeof operator :-

i) size of operator is look ~~work~~ as function but it's work as a operator.

ii) sizeof operator is return the interger size of storage of variable/ constant.

Some important example of sizeof operator.

i)
sizeof(3.14)
∴ It is float constant ∴ then it's return 8bit for it's byte value.

ii)
int a = 30;
size(a)
∴ It return 2 because it is an integer type value.

iii)
sizeof(30);
It return 4 because it is an integer constant.

iv)
~~size~~ int a=10, b=20;
sizeof(a+b);
∴ Here the sizeof operator does not solve the expression, its only return value, it 4 2 because both's are integer.

* Expression :-

i) Expresion is collection of operators & operands.
ii) Expresion have ~~an~~ at least one any type of operator.
iii) Expression may have one or more one operand which is operators performs operations.

iv) Pricrity of executing of expression is Bodmas Rule

i) Solve bracket first.
ii) / , %
iii) *
iv) +
v) −

e.g. a = 12 * (4/5)
= 12 * 0 ∴ Here division operator is used then zero.
a = 0

a = 12 + (4/5)
a = 12 + 0 :

IMP Type casting (Type conversion) :-
Type casting is a process in which we convert one datatype to another datatype.
casting have two types.
i) Implicit casting
ii) Explicit casting.

1] Implicit casting :-
The casting which is done by complier automatically during the execution of program.
e.g. 3 + 5.5
3.0 + 5.5
8.5
8.5

Here we get two o/p. when user wants o/p in integer format with by assign the value then output is zero, otherwise o/p is 8.5

ii) char c;
c = 'c' + 1 - - ?
Here complier automatically gets ASCII code of c. its 98 then in which add 1.
∴ 99 & 99 is the ASCII code of D.
then c = D :

ii] Explicit casting :-

i) In Explicit casting, we force / applied casting before the execution of program using the type cast operator. [( )].

e.g.

~~int a=(int) 5 + 3.5;~~

i) int a = (int) 3.5 + 5;

int a = 3 + 5;

a = 8;

Here we use integer type casting operator i.e. (int).

ii) int a = int (3 + 5.5);

a = 8;

here the o/p is same but procedure is different.

* I/O statement :-

A] printf();

i) printf statements containg two parts.

a) string part.

b) variable part.

ii) printf statement return the how much colums requried for display the string in integer format.

e.g. a = printf("%d", 1234)

o/p = a = 4;

Now printf("%d", printf("%d", 1234));

∴ Here the complier executes inner printf() which o/p is 1234, then return the column size is 4

then final o/p is 12345.

B] Scanf():

i) It is used to get value from user.

ii) It get only interger, Float & character inputs

iii) scanf() contain two parts

a) control string → format specifier contains

%d = interger , %ld = long integer

%f = Float , %lf = Double Float.

b) variable list :- in which we ge, stored the values (input) from user.

iv) scanf() return the interger value which is corresponding to values read by user.

v) scanf() use for validation also.

eg. void main()
{
int a, b, c, sum t;
clrscr();
printf("Enter the three values");
t = scanf("%d%d%d", &a, &b, &c);

if(t != 3)
{
printf("In one or more than one inpu
is are incorrect !");
getch();
return 0; // exit(0);
}
Sum = a+b+c;

```
    printf("sum = %d", sum);
    getch();
    return(0);
}
```

// in c use return() or exit() for the logical end which is appears multiple times

Q. Why return 0 in c?

→ As assumption 0 is good return value in c but we can use any interger value.

∴ return 0 ✓

return 1 ✓

return -1 ✓

05/09/2023

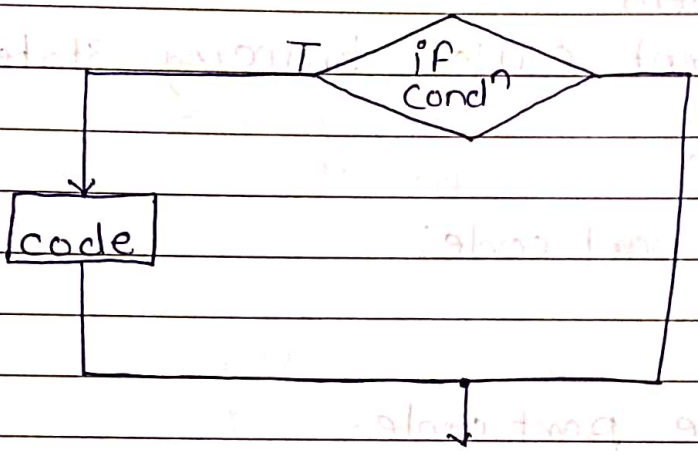
Unit - II

i) Simple if statement :-

There is only one part i.e. if condition block is executed when condition is true otherwise program continues from next step.

Syntax :-

```
if (cond^n) { }
```



∴ If the if statement have non-zero value then its true condition.

e.g. if(-1) ——— True
if(1) ——— True
if(0) ——— False

Q WAP to find smallest from three numbers using simple if statement.

→
```
void main()
{
    int a, b, c, small;
    clrscr();
    printf("Enter the three numbers : ");
    scanf("%d%d%d", &a, &b, &c);
    small = a;
```

```
        if(small > b)    small = b;
        if(small > b     small = c;

        printf("\n small number = %d", small);

        getch();
    }
```

ii) if-else statement:-

This statement called branching statement

e.g.

```
if(cond^1)
{   True part code;
}
else
{   False part code.
}
```

iii) Nested if statements:-

syntax:-

```
if(cond^1)
{
    if(cond^2)
    {
    }
    else
    {
    }
}
else
{   IF(cond^3)
    {
```

```
        else
        {
        }
    }
```

iii) else IF ladder :-

Syntax :-

```
if( cond^n 1)
{  #code
}
else if( cond^n 2)
{   # code
}
:
else
{
}
```

v) switch case statements :-

Syntax.

```
switch ( Expression ( interger/~~float~~ character ))
{
    case n1;
     //code
    break ;
     :

    default :
      code
}
```

* Looping statement :-

Loops in c is used to execute the block of code several times according to the condition given in the loop.

Types of loops :-

i) for statement.
ii) do-while statement
iii) while statement.

i] For loop :-

It execute the code until condition is false.

There is three parameters in loof statement.

a) Initialization
b) condition
c) Increment / Decrement.

Syntaxe :-

```
for( initialization; condition ; inc/dec)
{
    //code to be executed
}
```

The for loop also called per-tested loop. It is better to use for loop if the number of iteration is known in advanced.

Q. WAP to print Fiboranci series with n numbers.

→
```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a, b, c, n, i=3;
    clrscr();
```

```
printf(" Enter the total numbers : ");
scanf(" %d", &n);
a=0;
b=1;
printf(" %d \n %d ", a, b);

for( ;i<=n ; --i)
{
   c=a+b;
   printf(" \n %d", c);
   a=b;
   b=c;
}

getch();
}
```

Q. WAP to add odd numbers, even numbers seperatly from the range of number 1 to 50.

→

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

void main()
{
   int i , odd=0,even=0;
   clrscr();

   for(i=1 ; i<=50 ; i++)
   {
      (i%2==0) ? even+=i : odd+=i;
   }
```

```
        printf("\n Even numbers sum %d", even);
        printf("\n Odd numbers sum %d", odd);

        getch();
    }
```

Q. WAP to find factorial of the number :

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

void main()
{
    int n, fact=1
    clrscr();

    printf("Enter number : ");
    scanf("%d", &n);

    for(; n>1; )
    {
        fact *= n--;
    }
    printf("\n Factorial is : %d", fact);

    getch();
}
```

Q write a program to find largest number among the 'n' non-zero positive integer.

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

```
int n, no, lar=0;
clrscr();
printf("Enter the size : ");
scanf("%d", &n);

for( ; n>0; n--)
{
    printf("Enter the number : ");
    scanf("%d", &no);

    if(lar<no)
      lar=no;
}
printf("\n largest number :%d", lar);

getch();
}
```

Q. write the program to test the number is prime or not?

① ⟶

```
int main()
{
    int no, i, flag=1;
    clrscr();
    printf("Enter the number : ");
    scanf("%d", &no);

    for(i=2; i<=no/2; i++)
    {
        if(no%i==0)
        {  flag=0;
           break;
        }
    }
```

```
    if( flag==0)
        printf("%d is NOT prime number :", no);
    else
        printf("%d is Prime number : ", no );

    return(0);
    getch();
}
```

ii) By counter method.

```
int main()
{
    int i, no, count=0;
    clrscr();
    printf("Enter the numbers : ");
    scanf("%d", &no);

    for(i=2; i<=no/2; i++)
    {
        if(no%i==0){
            count++;
            break;}

    }
    (count!=0) ? printf("Not Prime number") :
                 printf("Prime number :");

    getch();
    return 0;
}
```

iii)

```
int main()
{
    int i, no ;
    clrscr();
    printf("Enter the numbers : ");
    scanf("%d", &no);

    for( i=2 ; i<=no/2 ; i++)
    {
        if( no%i==0)
        {
            printf("Not prime ");
            getch();
            return 0;
        }
    }
    printf("Prime number ");
    getch();
    retun 0;

}
```

Q. ~~print~~ Find sum when sum = 1 + 1/2 + 1/3 --- 1/n;

→

```
int main()
{
    int i, n;
    float sum= 0;
    clrscr();
    printf("Enter the number : ");
    scanf("%d", &n);
```

```
for( ; n>0; n--)
    sum = sum+ 1.0/n;

printf("Total = %d ", sum);
getch();
return 0;
}
```

Q WAP to print common divisor of 2 non-zero positive integer numbers -

→
```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n1, n2, i;
    clrscr();
    printf("Enter two numbers : ");
    scanf("%d%d", &n1, &n2);

    if( n1>n2)  a=a^b, b=a^b, b=a^b;
    for(i=1; i<=n1/2; i++)
    {
        if( n1%i==0 && n2%i==0)
            printf("\n %d ", i);

    }
    getch();
}
```

Q. WAP to evaluate a series $s = 1/1! + 2/2! + \cdots + \frac{n}{n!}$

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

```
void main()
{
    int n, i;
    float sum=0, f=1;
    clrscr();
    printf("Enter the size of series : ");
    scanf("%d", &n);

    for(i=1; i<=n; i++)
    {
        f = f*i;
        sum= sum+ i/f;
    }
    printf("\n sum = %f", sum);
    getch();
}
```

* while statement =

i) It is working similler to for loop.

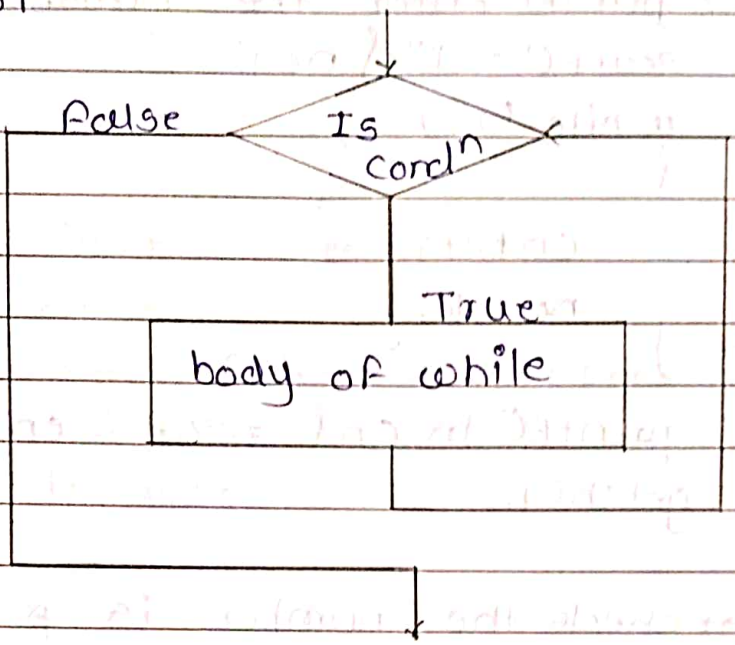
ii) when we want to execute the specific ~~editi~~ code with perticular condition but we don't idea about How much iteration required for it then we use while loop.

iii) In while loop, First condition is check the block of code is executed until the the condition is False.

Syntax :-

```
while(condition)
{
    // code
    update the value of contral variable
}
```

Flow chart:



Q. WAP to print number from 1 to 100 which are divisible by 4 & 6;

→
```
void main()
{
    int i=1;
    clrscr();
    for(while(i<=100)
    {
        if(i%4==0 && i%6==0)
            printf("%d", i);
        i++;
    }
    getch();
}
```

Q. WAP to count digits in a ger number.

→
```
void main()
{   int no,cnt=0;
    clrscr();
```

```
printf("Enter the number : ");
scanf("%d", &no);
while (no)
{
    cnt++;
    no /= 10;
}
printf("\n cnt = %d", cnt);
getch();
}
```

Q. WAP to check the number is Armstrong or not

→

```
void main()
{
    int no, temp, sum=0, r;
    clrscr();
    printf("Enter the number : ");
    scanf("%d", &no);
    temp = no;
    while(no)
    {
        r = no % 10;
        sum = sum + r*r*r;
        no /= 10;
    }
    if(sum == temp)
        printf("Enter the Armstrong number : ");
    else
        printf("Not armstrong number");
    getch();
}
```

Q. WAP to check whether the given number is pallindrom or not.

→
```
void main()
{
    int no, r, temp, rev = 0;
    clrscr();
    printf("Enter the number : ");
    scanf("%d", &no);
    temp = no;

    while(no)
    {
      r = no % 10;
      rev = (rev * 10) + r;
      no /= 10;
    }
    if( rev == temp)
        printf(" Pallindrom number ");
    else
        printf(" Not pallindrom number ");

    getch();
}
```

* do-while loop :-

i) When we want the block of code executed at least ones without cond^n then we use do-while in there.

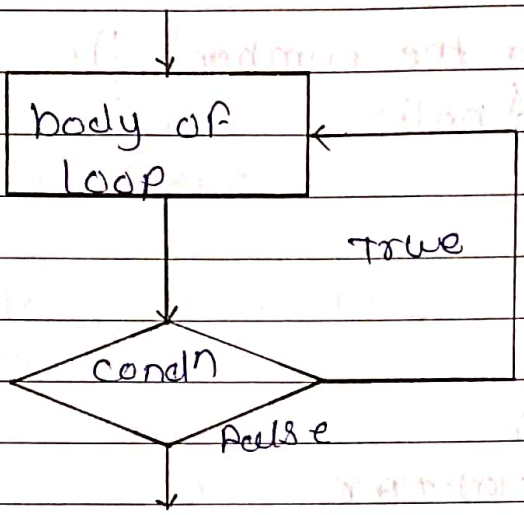
ii) Do while loop terminated with semicolon.

iii) Flow →

body → condition.

Syntax :- do
{
//code
} while(condition);

Flow chart



* Nested for loop

Q. writer the program that multiplication tables for numbe 1 to 10;

→
```
void main()
{
    int i, j;
    clrscr();
    for(i=1; i<=10; i++)
    {
        for(j=1; j<=10; j++)
        {
            printf("%4d", i*j);
        }
        printf("\n");
    }
    getch();
}
```

Q. write a program to demonstrate use of continue statement.

→ void main();
{
int i;
clrscr();
for(i=0; i<=10; i++)
{
printf("%4d", i);
continue;
printf("%4d", i*5);
}
getch();
}

* Break statement -
i) break statement use for terminating looping structure.

* Drawback -
ii) C is structured programming language but goto make it unstructured.
iii) take more to execute.
iii) Goto statement have a label.

Q. void main()
{ int i=1;
clrscr();
Next: if(i<=10)
{
printf("%d", i);
i++;
goto Next;
}
getch();
}

Q. Label below the goto statement.

→
```
void main()
{
    int i=0;
    clrscr();

    xyz: if(i>9) goto abc;
         i++;
         printf("%d \t", i);
         goto xyz;
    abc: getch();
}
```

Q. WAP to add digits of number using if & goto statement.

→
```
void main()
{
    int no, sum=0;
    clrscr();
    printf("Enter the number : ");
    scanf("%d", &no);

Label: sum = sum + no % 10;
    if(no)
    {  no /=10;
       goto label;
    }
    printf(" sum of digits = %d", sum);
    getch();
}
```

* Jumping in & jumping out from the looping statement

* Array:-

Array is collection of homogeneous element which are stored in continuous memory location.

- Types of Array:

i) Single dimensional array:
ii) Multi-dimensional array (2-d array):

* Single (one) dimensional array:-

When the array elements are stored in one row with continuous memory location.

e.g.

```
int num[10]
```

1000 →	num[0]
1002 →	num[1]
1004	
1018	num[9]

* Intillization 1D array :-

i) Compile Time
ii) Run Time

- Compile Time - also called static intilization.

In which we intilize all the value to array at compile Time.

e.g. `int array[5]={1,2,3,4,5};`

Imp→ i) If int num[10] = {0}.
Here all the values of ~~elements~~ array are zero.

ii) If int num[10] = {1,2};
then only 1, 2 are the first & second element of the array respectively. others are zero.

iii) int num[] = {1, 2, 3, 4}.
Here compiler automatically assign size of array as 4.

iv) If int num[] = {0,0,0,0} then compiler counts writes zero upto it's size.

- Run-Time intilization / Dynamic intilization.
  int array[10];

Q. write a program to find smallest number from array with size 10.

→
```
void main()
{
    int num[10]={ 10,5, 8, 11, 20, 40, 70, 90, 1, 2};
    int i, small;
    clrscr();
    small = num[0];
    for(i=1; i<10; i++)
    {
        if( small > num[i])
            small = num[i];
    }
    printf("Smallest number : %d", small );
    getch();
}
```

Q. WAP to create an array of 10 element & count odd & even number from it

→
```
void main()
{
    int no[10], even=0, odd=0;
    clrscr();
    printf("Enter 10 elements of an array:");
    for(i=0; i<10; i++)
    {
        scanf("%d", &no[i]);
        (no[i]%2==0)? even++: odd++;
    }
    printf("Total even numbers : %d", even);
    scanf("Total odd numbers : %d", odd);
    getch();
}
```

Q. To find largest number from 2d array of size 3×3

→
```
void main()
{   int a[3][3]={2,1,10,11,3,6,7,8,9};
    clrscr();   int i,j,s largest=a[0][0];
    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
        {
            if(largest > a[i][j])
            {   largest = a[i][j];
            }
        }
    }
    printf("largest number : %d", largest);
    getch();
}
```

* Multiplication of Matrix :-

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$

$$a*b = \begin{bmatrix} 3+2+6 & 2+6+3 & 1+2+3 \\ 9+1+4 & 6+3+2 & 3+2+6 \\ 6+3+2 & 4+9+1 & 2+6+3 \end{bmatrix} = \begin{bmatrix} 11 & 11 & 6 \\ 14 & 11 & 11 \\ 11 & 14 & 11 \end{bmatrix}$$

```
void main()
{
    int a[3][3], b[3][3], i, j, k, sum;
    clrscr();
    Printf("Enter elements in first matrix");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d",&a[3][3]);
        }
    }
    printf("Enter elements in second matrix");
    for(i=0;i<3;j++)
    {   for(j=0;j<3;j++)
        {
            scanf("%d",&b[3][3]);
        }
    }
```

```
printf("multiplication of matrices :\n");
for(k=0; k<3; k++)
{
    for(i=0;i<3; i++)
    {
        sum=0;
        for(j=0;j<3;j++)
        {
            sum=sum+ a[k][j]* b[j][i] ;
        }
        printf("%d\t", sum);
    }
}
getch();
}
```

* Area of circle- πr^2
* Area of rectangle = wl
* volume of cube : a^3
* strong nume = number whose sum of the factorial of digit of digit is equal to original number.

Q.1 Wap to input the 3 sides of a triangle & print it's corresponding type-

Equilateral - All sides are equal.
Isosceles - 2 equal sides.
Scalene - no equal sides

→
```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,b,c;
    clrscr();
    printf("Enter the size of triangle - ");
    scanf("%d%d%d",&a,&b,&c);

    if(a==b&&a==c)
        printf("Equilateral Triangles");
    else if(a==b||a==c||b==c)
        printf("Isosceles Triangles");
    else
        printf("Scalene Triangle");

    getch();
}
```

Q. Wap to print this type of paramid.

```
        *
      * *
    * * *
  * * * *
* * * * *
```

```
#include<stdio.h>
int main()
{
    int n, row, c, i, temp;
    printf("Enter No. of rows - ");
    scanf("%d", &n);
    temp=n;
    for( row=1; row<=n; row++)
    {
        for( i=1; i<temp; i++)
        {
            printf(" ");
            temp--;
        }
        for(c=1; c<2*row-1; c++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Q. WAP to evaluate series.

i) $p=(1*2)+(2*3)+\cdots+(9*10)$;

→

```
#include<stdio.h>
int main()
{
    int i=1;
    int sum=0;
    for( ; i<=10; i++)
    {
        sum=sum+(i*(i+1));
    } Printf("sum = %d", sum);
    return 0;
}
```

② $Q = 1/2 + 3/4 + 5/6 + \cdots 13/14$

③ $S = 2/5 + 5/9 + 8/13 \cdots + n$

④ $a = x + x^2 + x^3 + x^4 + \cdots + x^{10}$

⑤ $p = x + x^3/3 + x^5/5 + x^7/7 \cdots n$

⑥ $S = (13 * 1) + (12 * 2) \cdots (1 * 13)$

⑦ $S = 1 + 1/(2^2) + 1/3^3 + 1/4^4 + \cdots + 1/5^5$

Date :
Page :

① $Q = 1/2 + 3/4 + 5/6 + \cdots 13/14$ — $Q = ?$

② $S = 2/5 + 5/9 + 8/13 \cdots n$

③ $S = x + x^2 + x^4 + \cdots + x^{10}$

④ $P = x + x^3/3 + x^5/x^7/7 \cdots + n$

⑤ $S = (13 * 1) + (12 * 2) \cdots + (1 * 13)$

⑥ $S = 1 + 1/(2^2) + 1/3^3 + 1/4^4 + 1/5^5 \cdots$

① $Q = 1/2 + 3/4 + 5/6 + \cdots + 13/14$

→

```
void main()
{
    int i, j; float sum = 0.0;
    clrscr();

    for(i=1; j=2; i<=13; i+=2; j+=2)
    {
        sum = (float) i/j + sum;
    }
    printf("%f", sum);

    getch();
}
```

② void main :

```
{   int n, i, j = 5;
    float sum = 0.0;
    clrscr();
    printf("Enter limit : ");
    scanf("%d", &n);
```

```
for ( i=1; i<=n ; i++)
{
    sum = (float) i / j + sum;
    j = j+4 ;

    sum = (float) (i+1) / j + sum ;

    j = j+4 ;
}
printf(" sum %.f ", sum );
getch();
}
```

③ $s = x + x^2 + x^4 + \cdots + x^{10}$

→
```
#include<math.h>
void main ()
{   int n, i, sum = 0;

    clrscr();
    printf(" Enter an number: ");
    scanf(" %d ", &n);

    for ( i = 1; i<=n; i++)
    {
        sum = sum + pow(n, i);
    }
    printf(" sum = %d ", sum);
    getch();
}
```

④ $s = (13*1) + (12*2) \cdots + (1*13)$;

→

```
void main()
{
    int i=1, j=13, sum=0;
    clrscr();
    for( ; i<=13; i++)
    {
        sum = sum + (i*j);
        j--;
    }
    printf("%d", sum);
    getch();
}
```

Q. Write a program to copy content of one file to another file.

→
```
#include<stdio.h>
#include<conio.h>
void main()
    {
        FILE *ftr1, *ftr2;
        char c;
        clrscr();
        ftr1 = fopen("Filename1.txt", "r");
        if(ftr1 == NULL)
        {
            printf("ERROr");
            exit();
        }
        ftr2 = fopen("Filename2.txt", "w");
        if(ftr2 == NULL)
        {
            printf("ERROr");
            exit(1);
        }
        while((c=getc(ftr1)) != EOF)
        {
            putc(c, ftr2);
        }
        fclose(ftr1);
        fclose(ftr2);
        getch();
    }
```