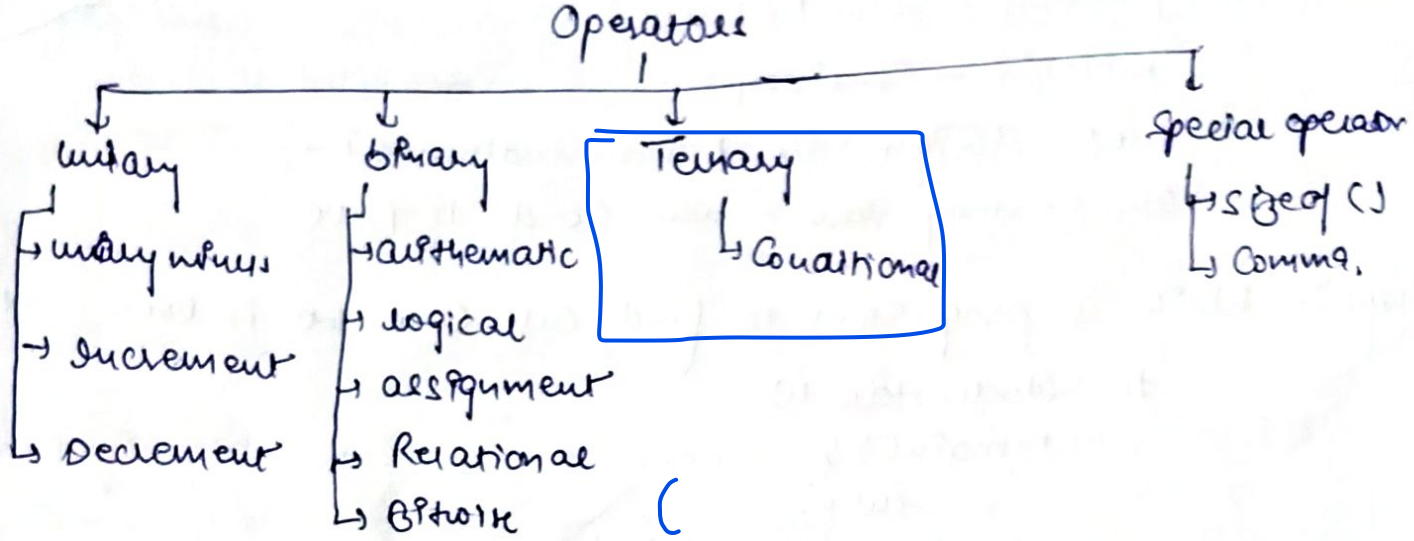
Unit-2

Arithmetic Expression & Conditional Branching



Operator :- Operator is a symbol that tells the Computer to perform certain mathematical and logical manipulations

Special Operators :-

1) size of (); size of operator returns the size (no. of byte) of operant and the operant may be a constant, variable or a Data type.

size of ( );

2) Comma operator :- Comma operator is used to link two or more related expression together and the expressions are evaluated in left to right manner.

We can obtain the value of combined Expression from the value of right most expression.

E.g.

Sum = (a=10, b=20, c=a+b);

Sum = 30.

Ternary Operator :- (Conditional)

Syntax :-

Expression ? Value1 : Value2

Expression - Condition

Value1 - Assign value 1 when Condition is True

Value2 - Assig value 2 when Cond is false

Ans :- Write a programm to find out Give no. is even or odd

```
#Include<stdio.h>
Int main() {
    Int n;
    Scanf("%d", &n)
    (n%2==0)? printf("even") : printf("odd");
    return 0;
}
```

Ans. Final a Operation to find out largest of a No. by using Ternary

```
#Include <stdio.h>
Int main() {
    Int a,b,c
    Scanf("% %d %d" &a, &b, &c);
(a>b && a>c)? Printf("a") : break;
(b>a && b>c)? Printf("b") ; break;
(c>a && c>b)? Printf("c") : break;
    return 0;
}
```

Another

```
Int a= (a>b)? (a>c? a:c) : (b>c?b:c);
Printf ("%d is largest", a);
```

Doubt

Binary operator:

1) arithematics Operator :-

Operators	meaning	Example.
+	addition	a+b = 15
-	Subtraction	a-b = 10
*	multiplication	a*b = 56
/	Division	a/b = 3
% (modulo)	Reminder	a%b = 2

2) logical operator :- logical operators used when we want to taste more than one condition and make a decision.

i) logical AND (&&)
ii) logical OR (||)
iii) logical NOT (!)

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

3) Relational operator :-

A = 10 B = 20

operator	Meaning	Example	
= =	equal	A = = B	⇒ 0
>	Greater	A > B	⇒ 0
<	less	A < B	⇒ 1
>=	Greater ~~than~~ equal	A >= B	⇒ 0
<=	less or equal	A <= B	⇒ 1
!=	Not equal	A != B	⇒ 1

④ Assignment Operator :- assignment operator is used to assign the result of an expression

Statement with simple assignment operator	Statement with Shorthand Operator
a = a+1	a += 1
a = a-1	a -= 1
a = a * (n-1)	a *= (n-1)
a = a/(n-1)	a /= (n-1)
a = a % (n-1)	a % = (n-1)

2) += → It add right hand operand to left operand and assign the result to left operand

Bitwise Operator :-

Bitwise Operator is used for manipulation of data at bit level, these operator is used for testing the bits or shifting them right or left.

Bitwise Operator are not applied to float or double number.

1) One's Complement (~)
2) bitwise AND (&)
3) bitwise OR (|)
4) << bitwise left shift
5) >> bitwise right shift
6) ^ bitwise exclusive OR (XOR)

Bitwise Operator :-

System variable_name Shift_operator no b

Shift oper << bitwise left
>> bitwise right

no- no. of shifts

uniary
That acts on single operands
1) uniary (-) :- when an operand is precedes by minus sign the uniary operator negate its value.

2) Increment & Decrement [++a, --a]
The Increment/Decrement operator are uniary operate that increase/decrease it operand value by 1

prefix Expression :- This indicates pre-increment/pre-decrement that means the value of operand must be increment/decrement before it is used

postfix increment (a++, a--) :- this indecates post increment/decrement which means use the value of operands first than increment/decrement

Operator preference and precedence & associativity :-
Operator precedence :- when an expression have more than one operator than which operator execute or evaluate first is decided by operator precedence.

Associativity :- ~~here~~ in an expression when we have more than one operator with same precedence then which operator is evaluated first is decided by associativity, it can be either left to right OR right to left

Operator Category	Operator	Operator Precedence	Associativity
1) parenthesis/Braces	(), [], {}	1	L to R
2) Unary	-, ++, --, !, ~	2	R to L
3) Multiplicative	*, /, %	3	L to R
4) Additive	+ -	4	L to R
5) Bitwise shift	<<, >>	5	L to R
6) Relational	<, >, >=, <=	6	L to R
7) Equality	==, !=	7	L to R
8) Bitwise	&, |	8	L to R
9) Logical	&&, ||	9	L to R
10) Conditional	? :	10	R to R
11) Assignment	=, +=, -=, *=	11	R to L
12) Comma	,	12	L to R

① $x = 3*4+5*6$

$= 42$

② $x = 3*(4+5)*6$

$x = 3*9*6$

$= 162$

③ $x = 5*6+7-2/4\%3$

$= 30+7-0\%3$

$= 37$ ~~36.87~~

④ $y = 5\&13\,\&\&\,6\|\|23$

$y = 5\&13\,\&\&\,1$

5 - 00000101

00001100

0100 = 5

$y = 5\&\&6\|1$

1||1 = 1

Q a && b || c && (!b) = ? a = 2 b = 4 c = 3

2 && 4 || 3 && (!4)

1 || 3 && (0)

1 && (0)

0

1

Q. (10<12 && 45<18) || (10>65 && 45>72)

(1 && 1) || (0 && 0

1 || 0

1

Q. Void main ()

{

int i = -3, j = -2, k = 0, m

m = ++i && ++j || ++k;

printf ("%d.%d %d%d", i, j, k, m);

}

-2, 3, 0, 1

m = -2 && 3 || ++k

m = 1 || k = 0

1

Type Conversion & Type Casting

Type Conversion & Typecasting of variable refers to changing of variable of an data type into another.

Implicit Conversion :- Type Conversion is automatically done by the Compiler, in this Conversion the data type is promoted from lower level to higher level

1) Within Expression

```
void main ()
{
int a;
float b;
char c;
double d;
= a * b + c - d;
```

↓

double - because size is high

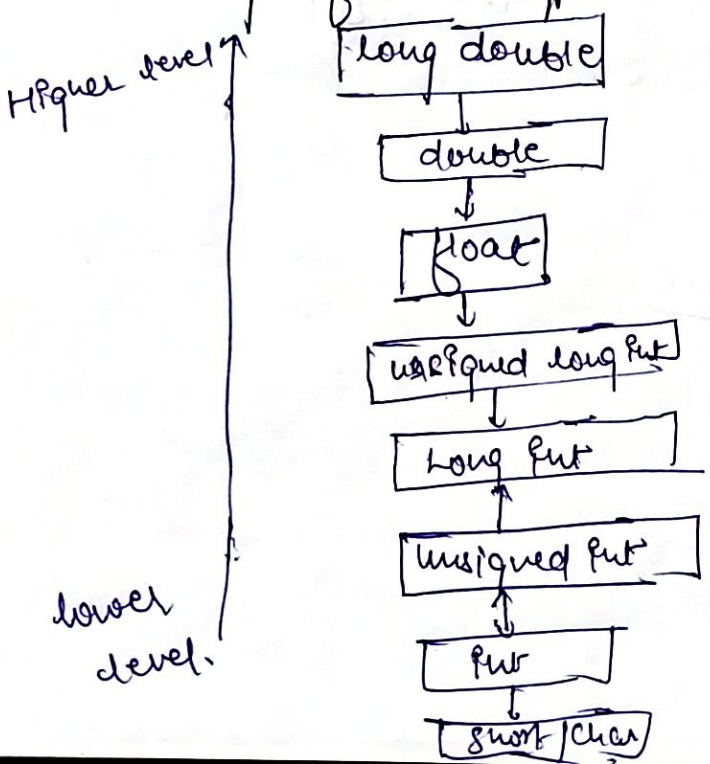
2) Assignment

```
void main (){
int a = 10;
float b
  b = a;
printf("%f", b);
}
```

o/p 10.000000

Promotion :- To evaluate the expression, the data-Type is promoted from lower level to higher level.

Conversion hierchy of Data Type



Demotion

In the case of demotion higher level data-type is converted into a lower level data-type and some data is lost.

No compile time warning message is generated when information is lost.

```
void main(){
    int a;
    float b = 10.25;
    a = b;
    printf("%d", a);
}
```

O/p ⇒ loss of data.

Type Casting (Explicit Conversion)

Type casting is done by the programmer, in this data is demoted to lower data level

```
void main(){
    int a, b=8
    float c;
    c = (a+b)/2;
    printf("%f", c)
}
```

→ Now = 5.

Q Write a program to read float point number and convert it into corresponding integer number.

~~void main ()~~
~~{~~
~~float a; int~~

```
void main ()
{
    float b;
    printf("enter a number");
    scanf("%f", &b);
    int a = b;
    printf("%d", a);
}
```

Q. Write a program to Read a float point no. & find the right most digit of integral part of that no.

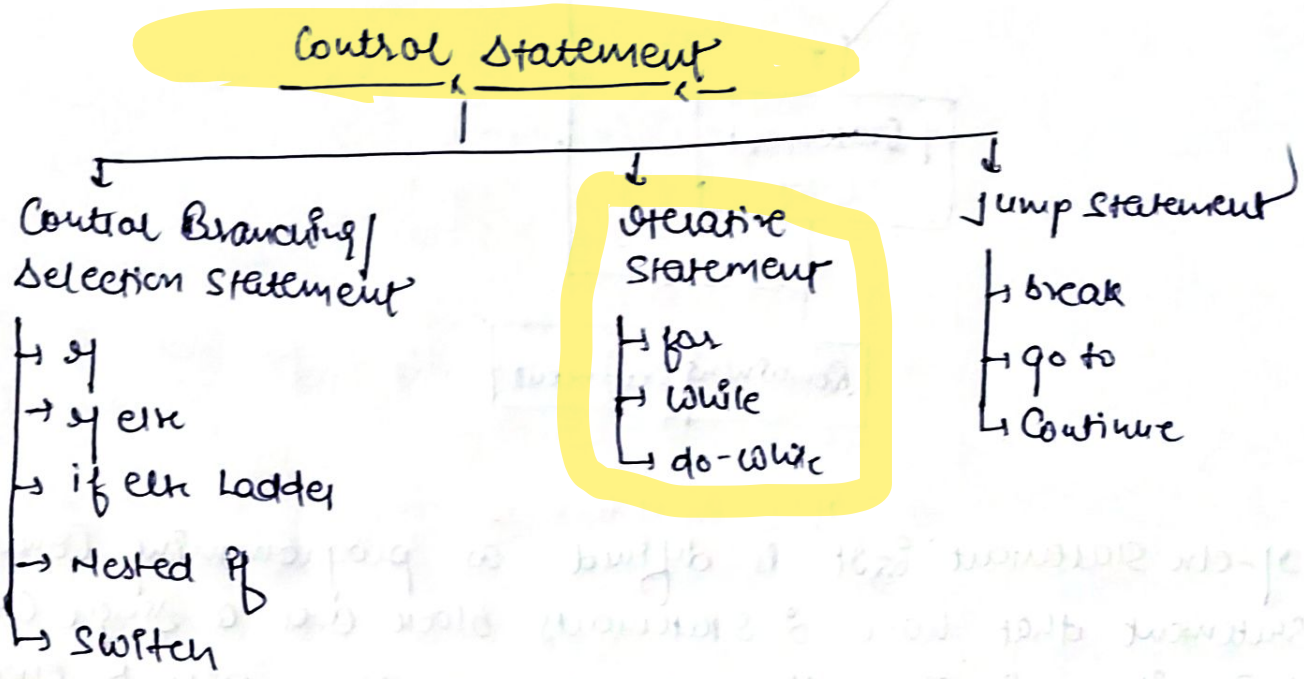
```
void main ()
{
    int a = float b;
    float scanf("%f", &b);
    a = b;
    int right = a%10;
    printf("%d", right);
}
```

Mixed mode Operation

Expression that involves element of type real and integer are known as mixed mode operation on the expression.

* If an expression contains integer and floating point element than the result will be of float type

Operators that are used in forming mixed mode expression are basically arithmetic operators & modulo operator



Control statement enables us to specify the flow of program control, they specify the order in which instruction in the program, must be excuted

1) if :-

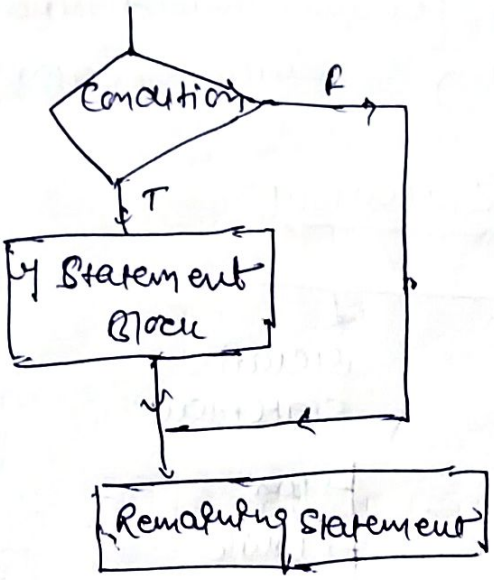
Syntax :-
```
if (condition)
  {
  statements;
  }
remaining statements;
```

E.g
```
void main ()
  {
  int a;
  if (a>10)
  printf ("Hello");
  printf ("world");
  }
```

O/P

if a>10 then Helloworld

if a<10 then world.



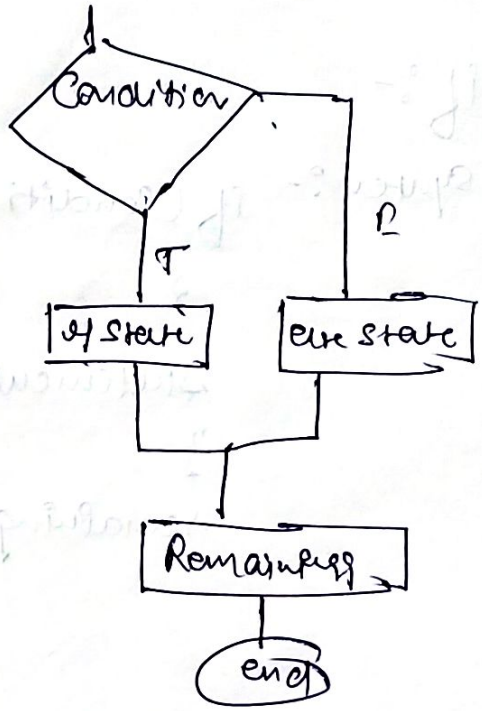
iii] If-else statement :- It is defined as programming Conditional statement that have 2 statements block over a single condition.

⇒ If Condition is True then if statement block will be executed

⇒ If condition is false then else statement block is executed

Syntax :-

```
if (Condition) {
        if statement ;
    }
else {
    else statement block;
  }
  Remaining statement;
```



Q. Write a program to find out Given No. is even or odd.

```
void main(){
        int n;
        printf("enter the number");
        scanf("%d",&n)
    if(n%2==0){
        printf("even");
      }
    else{
        printf("odd");
      }
    }
```

Q. Write a program to find out largest of 2 No.

```
void main(){
        int a,b;
        printf("ewer two no a&b);
        scanf("%d %d",&a,&b);
    if(a>b){
        printf("%a is Greater");
      }
    else{
        printf("b is Greater");
      }
    }
```

Wap to find a person is eligible for voting or not?

```
void main(){
    int age;
    printf("enter the age :");
    scanf("%d",&age);
if(age>=18)
  printf("you are eligible for voting");
else
  printf("you are not eligible for voting");
}
```

3) if-else Ladder

Syntax:-

```
if(condition1){
  if statement block1;
}
else if(condition2){
  else if statement block2;
}
else if(condition 3){
  else if statement block3;
}
else{
  else statement;
}
```

It has multiple else if block statement blocks, if anyone of the condition is True, then the control will exit the else if ladder and execute the next set of statement

Q. Wap to find ⊕, ⊖ or zero.

```
#include<stdio.h>
int main(){
    int n;
    printf("enter a number\n");
    scanf("%d",&n);
if(n==0){
    printf("the Given Number is zero");
  }
else if(n>0){
    printf("the number is positive");
  }
else{
    printf("the number is Negative");
  }
return 0;
}
```

4) Nested If

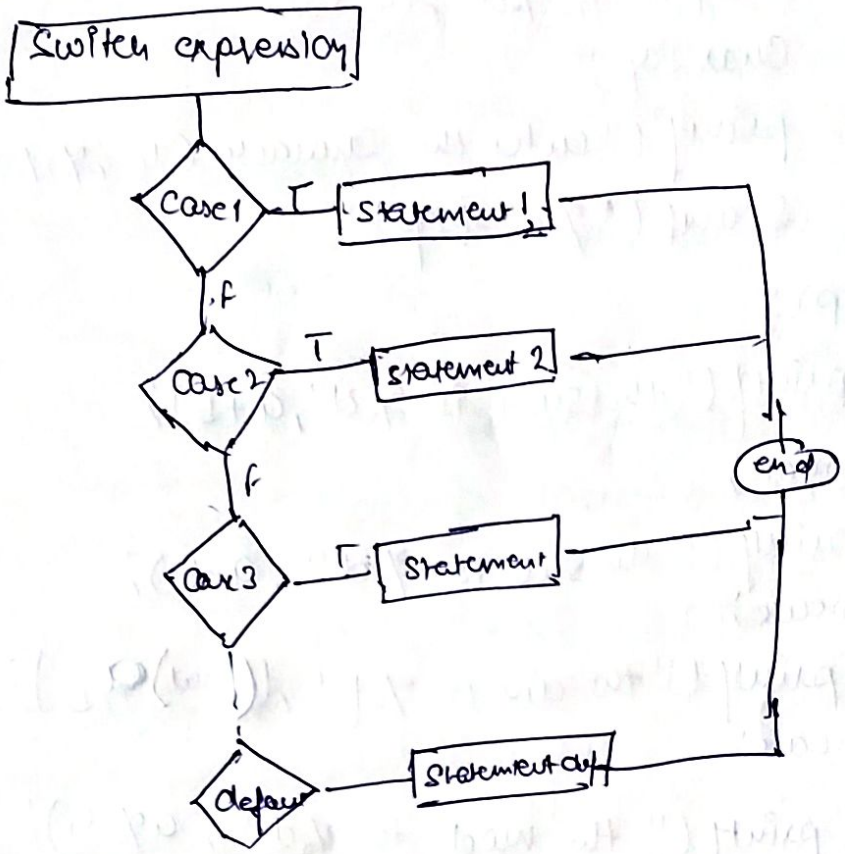
Q. Write a program largest among three No.

```
#include<stdio.h>
int main(){
    int a,b,c;
  printf("enter 3 No");
 scanf("%d %d %d",&a,&b,&c);
if(a>b){
  if(a>c){
    printf("largest No is %d",a);
  }
  else{
    printf("largest is No %d",c);
    }
else{
  if(b>c){
    printf("the largest No is %d",b);
    }
  else{
    printf("the largest No is %d",c);
    }
  }
return 0;
}
```

Switch

It is multiple branch selection statement which checks for a possible match to the provided condition against available cases

If non-of case matches then the control will encounter default value.



Limitation of switch case :-

1) Cases should always be followed by an integer constant, character or constant expression

2) All case should be distinct.

3) The block of statement under default is executed when non-of the cases match the value of expression.

4) Default case is optional

5) Cases & default can occur in any order.

6) The break statement causes an explicit exit from switch statement, if it is not present after executing the case which match value of the expression the following other cases are executed

Q. wap to read 2 values and perform basic function calculator

```
#include <stdio.h>
int main() {
    int a, b; ~~result~~;
    printf("enter two values a&b");
    scanf("%d %d", &a, &b);
    char op;
    printf("enter the character (+, -, %, /, *)");
    scanf("%c", &op);

  switch(op) {
case + : printf("the sum is %d", a+b);
         break;
case - : printf("the sub is %d", a-b);
         break;
case / :- printf("the div is %f", (float)a/b);
         break;
case %. : printf("the mod is %d", a%b);
         break;
case * : printf("the mul is %d", a*b);
         break;
default : printf("invalid operator selection");
  }
  return 0;
}
```

Loops

Looping statement :- Looping statement are Group of statement that are executed repeatedly until certain specified condition is satisfying

→ Looping is also known as repeatative or Iterative statements.

→ a loop in a programm consist of two part, one is called the body of the loop and other is known as Control statement (condition)

→ Control statement can be placed either before or after the body of the loop.

→ If the control statement is placed before the body of the loop. it is known as entry control loop

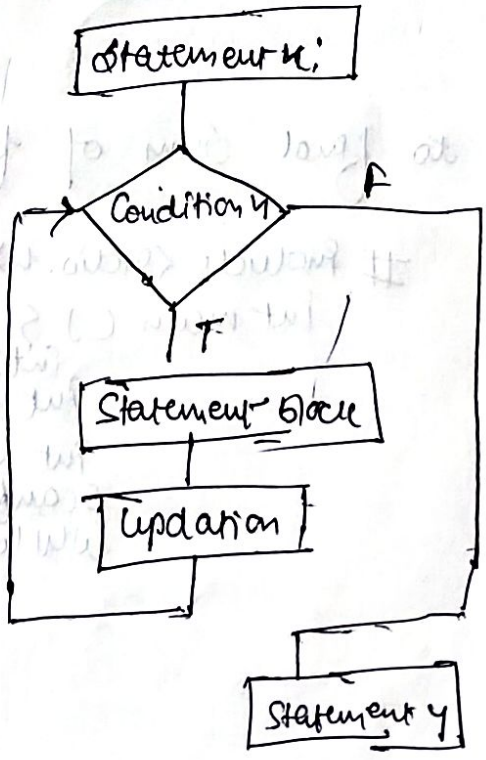
→ If the control statement is placed after the body of the loop. it is called as exit control loop.

Steps :-

1) Initialisation
2) Termination condition
3) Modification/updation.

Syntax :-

```
while (condition) {
    statement block;
    updation;
}
```



Write a program to print 10 times using while loop.

```
#include<stdio.h>
int main(){
    int i=1;
    while(i<10){
        printf("*");
        i=i+1;
    }
return 0;
}
```

Write a programme to print first 10 no. using while loop

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

wap to find sum of first n integers.

```
#include<stdio.h>
int main(){
    int n;
    int i=1
    int sum=0;
    scanf("%d",&n)
    while(i<=n){
        sum=sum+i;
        i=i+1;
    }
    printf("%d",sum);
}
return 0;}
```

do-while :-

Exit Control :-

```
do{
    Statement Block;
      updation
  }while(condition);
  Statement y;
```

Difference b/w while & do-while iterative statement

while	do-while
1) while is entry control loop	1) do-while is exit control
2) Condition is placed before body of the loop.	2) Condition is placed after the body of the loop.
3) we don't use a semi-colon of the condition	3) we use semi-colon after the condition
4) If condition is true then only the body of loop is executed	4) It will be execute the body of the loop atleast one time irrespective of condition is True or false.

C.P.

```
Void main(){
      int i=10;
  do{
   printf("Hello");
      i=i++
   }while(i<10)
  printf("World");
   }
```

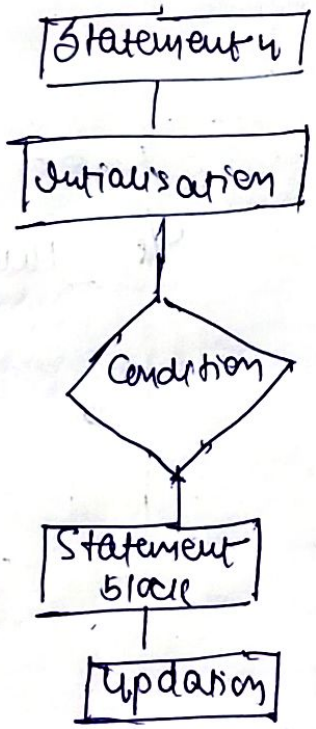
o/p Hellowond

E.g.

```
Put i = 10;
while (i<10) {
        Printf ("Hello");
    }
    Printf ("world");
}
```

③ For-loop (Entry Control).

```
Statement x;
for (Initialisation; Termination; updation) {
        Statement block;
    }
Statement y;
```



Wap to print 10 no.

```
void main(){
    int n=10;
  for(int i=1; i<=10; i++){
      Printf("%d", i);
    }
  }
```

Wap to find sum of first 50 Natural No.

```
void main(){
    int sum=0;
    int n=50;
  for(int i=1; i<=50; i++){
      sum = sum+i;
    }
    Printf("%d", sum);
  }
```

Wap to find factorial of Given Number n.

```
void main(){
    int mul=1;
    int n;
    Scanf("%d, &n);
  for(int i=1; i<=n; i++){
      mul = mul*i;
    }
    Printf("%d", mul);
  }
```

Wap to print the table of n.

```
#include <stdio.h>
int main() {
        int n;
        scanf("%d", &n);
    for (int i=1; i<=10; i++){
        printf("%d * %d = %d", n, i, n*i);

    }
    return 0;
}
```

Q:- Wap to check prime or not?

```
void main() {
        int i=2, n;
        scanf("%d", &n)
    while(i<n)
        {
        if(n%i==0) {
                printf("Not prime");
                break;
            }
            i++;
        }
    if(i==n) {
            printf("Prime");
        }
    }
```

Q:- Strong Number

```
# include <stdio.h>
  int main () {
        int n, rem, result=0;
        scanf("%d", &n)
              t = n;
  for (int i=1; i<=t; i++){
        rem = n%10;
        result += rem * rem * rem;
        n = n/10;
       if (n==0){
  }          break;
        }
     }
     if (sum==t) {
            printf ("no is Armstrong");
          }
       else {
            printf (" Not a armstrong");
          }
```

wap etc.

Q. find the reverse of a number

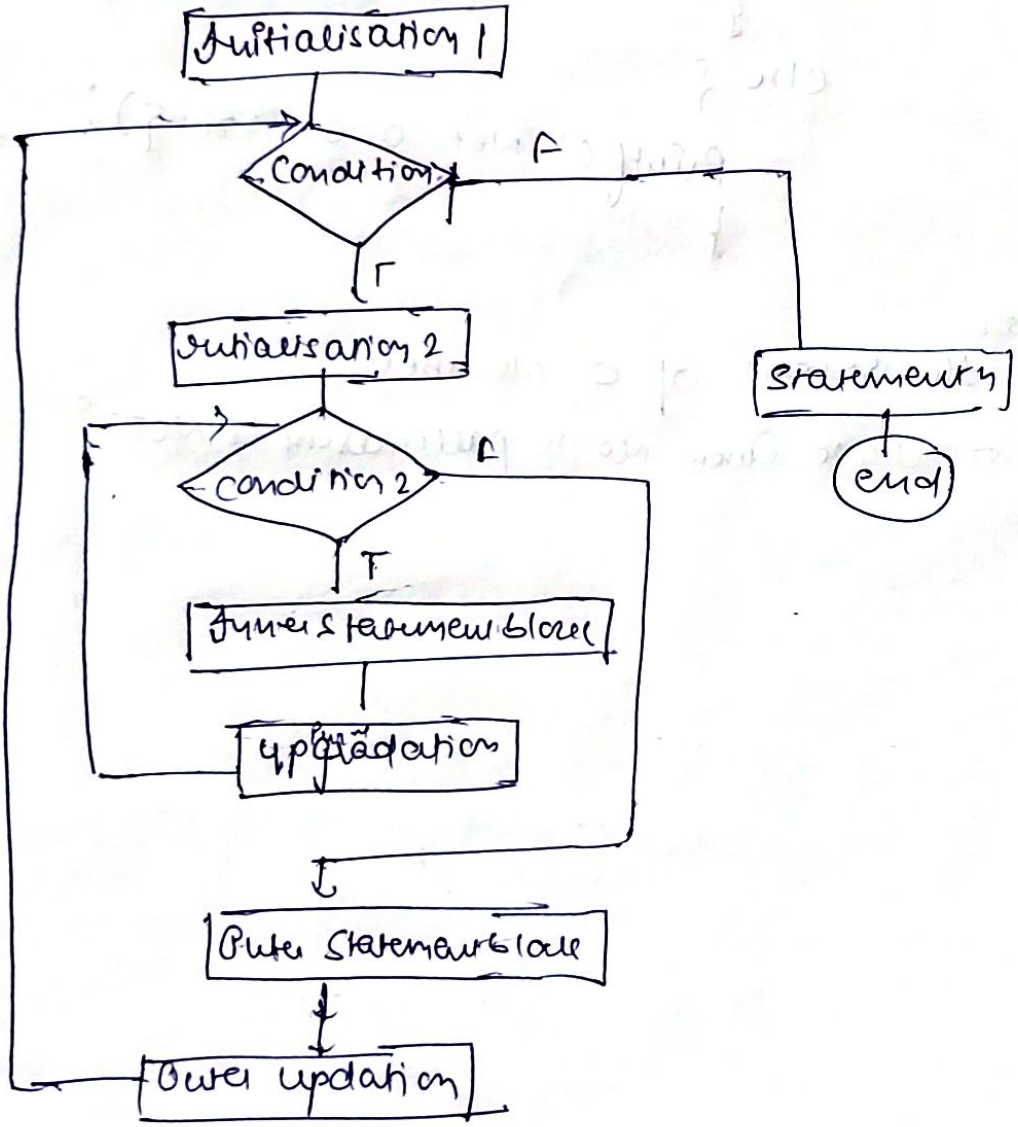
wap to check Given no is palindrom or NOT ?

Nesting of loop

In Nested of loop one loop is placed inside the other loop similarly the if statement can be Nested so, while and for loop can also be Nested.

Syntax:-

```
for(init1; C1, U1){
    for(I2, C2, U2){
      Inner block statement;
    }
    Outer block statement;
}
Statement n;
```



E.g.

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

```
#include <stdio.h>
int main() {
    int n=4, i, j;
    for (i=1; i<=n; i++) {
        for (j=1; j<=n; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

(2)

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

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

(3)

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

```
void main() {
    int i, j;
    for (i=1; i<=4; i++) {
        for (j=1; j<=i; j++) {
            printf("*");
        }
        printf("\n");
    }
}
```

④

```
void main ()
{
    int i, j;
for (i=1; i<=4; i++)
{
  for (j=4; j>=i; j--)
  {
    printf ("*");
  }
  printf ("\n");
}
```

break statement:

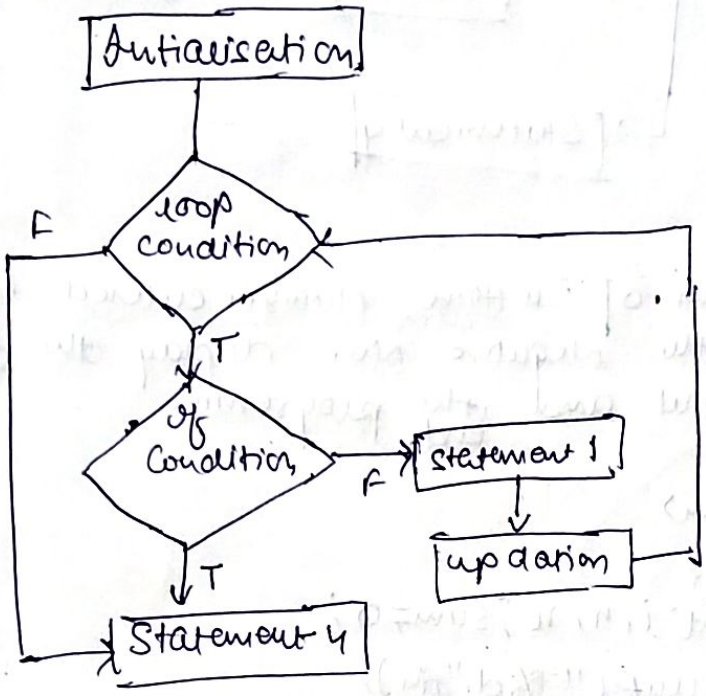
break statement is used in terminating the loop immediately after it is encountered

It can be used in terminating all three loops while do and for loop as soon as it gets executed programm control comes out of the loop. and first statement after the loop gets executed

Syntax:

```
break;
```

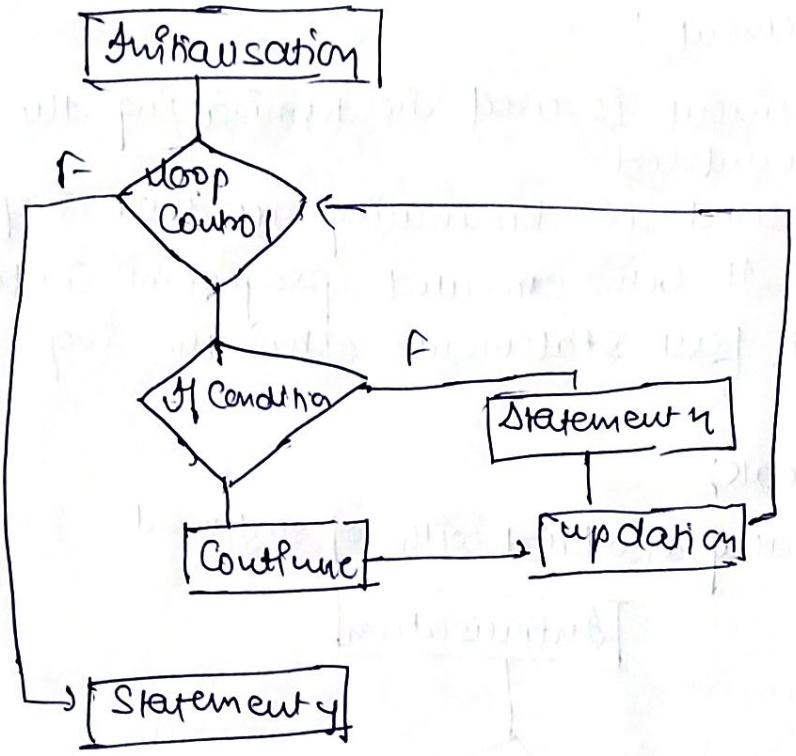
It is basically associated with if statement



Continue statement:

```
for(i;c;0)
 {
 if(condition){
     continue);
   }
   statement4;
}
```

Continue statement is used for next iteration of the loop. to take place skipping any code in b/w, Hence it is used to terminates the current iteration & continue with next iteration of the loop.



Wap to find the sum of n positive Number entered by user if the user enter the Negative no. display the sum excluding the Negative input and ~~the~~ end, program

```
#include <stdio.h>
int main () {
    int i, n, x, sum=0;
    scanf("%d",&n);
    for(i=1; i<=n; i++){
        scanf("%d",&x);
        ~~sum = sum + x~~
        if(x<0){
            break;
        }
        sum = sum+x;
    }
    printf("%d", sum);
    return 0;
}
```

① Write a program to find power of Given No.?
② Way to find Given No is perfect No. or NOT?
③ Way to find sum of all odd No.s & even No.s b/w 1 to N.

```
#include <stdio.h>
int main(){
    int n,m;
    scanf("%d %d",&n,&m);
    int ans = 1;
    for(int i=1; i<=m; i++){
        ans= n*ans;
    }
    printf("%d is power",ans);
    return 0;
}
```

#

Function is a Group of statement that is used to perform a specific task, a last programm can be broke up into too no. of segment or blocks.
this segment and block or known as function.

```
data-type fun ( );
 main ( )
   {
   fun ( );
   }
 fun ( )
  {
  function body;
  }
```

The moment compiler encounter the function call instead of executing the next statement in calling function, the control jumps to the statements that are part of the called function,
After executing called function, the control is returned back to the main function or calling function.

Advantages of fun :-
1) understanding, coding and testing multiple seperate fun is far easier than doing it for one big function.
2) programm will very lengthy if we don't use function.
3) fun is used to speed up program development
4) debugging is easier
5) Reusability :- this avoid the rewritting of the fun code.

① fun Declaration :-
Like the variable declaration we must declare our function

```
return type function-name (parameter list);
data-type function-name (data-type var 1, data-type var 2);
```

function calling :-

function can be called anywhere in the program the parameter list must not differ in the function calling and fun declaration.

We must pass the same No. of parameter as it is declared in the function declaration.

Syntax

function-name (var1, var2);

eg. add(a,b);

function Deffinition :-

It contains actual statement which are to be executed, after execution, it will return back to the calling function

Syntax

return-type function-name (data-type var1, data type var2)

e.g. ~~edit~~ int add (int a, int b)

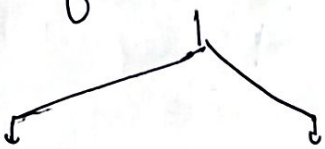
Wap to find sum of two Number using function

```
#include <stdio.h>
int sum (int a, int b);
int main () { int a,b;
              scanf("%d %d", &a, &b);
              sum(a,b);
              printf("%d", sum);
              return 0;
            }
int sum (int a, int b) {
              int add;
              add = a + b;
              return add;
            }
```

Wap to find factorial of Given No. using function

```
#include <stdio.h>
int fact(int a);
int main(){
    int m, z;
    scanf("%d", &m);
    z = fact(m);
    printf("factorial is %d", z);
}
int fact(int a){
    int a = 1;
    int i = 1
    for(i=1; i<=a; i++){
        a = a*i;
    }
    return a;
}
```

Types of function :-



pre-defined	user defined
→ printf()	Laadd()
→ scanf()	→ fact();
→ getch()	
→ clrscr()	
→ pow()	
→ sqrt()	

pre-defined are those function that are defined in header files
user defined function are defined by user itself ex. add, fact

Category of the function

1) function with argument and no return value
2) function with argument & no return value.
3) function without argument & having a return value.
4) function with argument & having a return value.

1] Function without argument & no return value :- the called function does not recieve any data from calling function and the called fun does not return any value back to calling fun.

2) Fun with argument & No-return value :

the called fun recieve the data from the calling function but does not return any value back to calling function. Instead it print data for its Scope Only.

```
~~int~~ void main (int a, int b);

void main ( ) {
        int x=2, y=5;
        add(x,y);
    }

void main (int a, int b) {
        int c
        c= a+b;
    printf("sum is %d", c);
    }
```

3) Function without argument & having a return value.

~~int sum (int a, int b);~~ The called fun doe not recieve any data from calling function it manages with its local data to carry out specified task. after process called fun return the computated data

~~void main ( ) {~~

~~sum ( );~~

```
int sum ( );
void main ( ) { int z;
        z= sum ( );
        printf("sum is %d", z);
    }
int sum ( ) { int x, y, sum;
        scanf("%d %d", &x, &y);
```

```
    sum = x+y;
    printf("the sum is %d", sum);
    return sum;
}
```

4) function with argument and having a return value.

```
int sum (int a, int b);
    int main (){
        int x, y, z;
        scanf ("%d %d", &x, &y);
        z = sum (x, y);
        printf (" the sum is %d", sum);
        return 0;
    }
int sum (int a, int b) {
        int c;
        c = a+b;
        return c;
    }
```

Parameter passing Method in Function :-

1) Actual Parameter :-
Argument that are passed in the function call are known as actual parameter

2) Formal Parameter :-
formal argumen are the parameter that are declared in the bracket of the function declaration.
The data type of formal parameter should be same as that of actual parameter
They should also be same manner as in fx declaration.

∴ It is a process of passing the arguments from calling function to the called function

1) Call by Value :-
When the values of arguments are passed from calling function to the called fx, the values are copied into the called function.
If any changes are made to the value of called values then their is no change in the original value wether in the calling function.

Swapping of 2 No-

```
void swap(int a, int b);
void main()
{
int x,y
scanf("%d %d", &x, &y);
swap(x,y);
printf("after swapping %d is x & %d is y", x, y);
}
void swap(int a, int b){
        int t;
        t=a;
        a=b;
        b=t;
printf("in swap fx a=%d and b=%d ", a, b);
}
```

o/p

In Swap Fun'n a=20, b=10

After Swapping x=10, y=20

At the time of fn call, the actual parameter Get Copied into the memory location of formal parameter ~~and~~ a & b of the fn swap

ii) Call by Reference

Pointer :- pointer is a special variable that is used to store address of another variable and is represented as * - asterik

int a = 10

int *b = a

b ∋ 100

a = 10

*b = 10

This method passes the address and Reference of actual parameter to the Called function.

In the called fn, the formal parameter receives the address of actual parameter by using pointers to them

void

After the swap function is called address of the actual parameter x & y would be assign to the pointer a & b, after the swap function, original value x, y get changed.

Q. Wap to find sum of following series

① $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} - - -$

~~#include <stdio.h>~~

```
int main() {
    int n, sum=0, i=1, j, rem;
    scanf("%d", &n);
    for(j=1; j<=n; j++){
        sum = sum + rem;
    }
```

```
int main() {
    int n; i=1; j;
    float sum=0; rem;
    scanf("%d", &n);
for(j=1; j<=n; j++){
    rem = i/j;
    sum = sum + rem;
}
printf("%f is sum", sum);
return 0;
}
```

② $x + \frac{x^3}{3} + \frac{x^5}{4} + - - -$

```
#include<math.h>
int main(){
    int n,x,i;
    float sum=0,rem;
    scanf("%d",&n);
    for(i=1; i<=n; i=i+2){
        rem=(pow(x,i))/i;
        sum=sum+rem;
    }
    printf("the sum is %f",sum);
    return 0;
}
```

(3) $x+\frac{x^2}{2}+\frac{x^3}{3}+\frac{x^4}{4}$ ....

```
#include<stdio.h>
#include<math.h>
int main(){
    int n,x,i;
    float sum=0,rem;
    scanf("%d",&n);
    for(i=1; i<=n; i++){
        rem=(pow(x,i))/i;
        sum=sum+rem;
    }
    printf("%f is sum",sum);
    return 0;
}
```

(4) $x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+$ .....

```
#include<stdio.h>
int main(){
    int n,x,i;
    float sum=0,sem;
    scanf("%d",&n);
    for(i=1;i<=n;i++){
        sem=(pow(-1,i+1)*pow(x,i))/i;
        sum=sum+sem;
    }
    printf("%f",sum);
    return 0;
}
```

Storage class :-

Storage class gives the complete information about the variable and the informations are

1) Where the variable would be store
2) Initial value of the variable, if it is not initialize.
3) Scope of the variable
4) What is the life of variable

Types of storage class

1) Auto storage class :- This is the default storage class of variable.
   1) Storage - main memory
   2) Initial value - Garbage value
   3) Scope - local
   4) life - variable hold the value till the control remains within the block in which it is declared
   5) keyword - auto

ii) Register Storage Class :-

1) Storage - CPU Register
2) Initial value - Garbage value
3) Scope - local
4) Life → variable hold the value till the control remains within the block in which it is declared.
5) keyword - auto.

Register variable are used when quick access to the variable ~~are~~ is needed or variable required to access repeatedly time

iii) Static Class :-

1) Storage - Main memory
2) Initial value - zero.
3) Scope - Local
4) life - value of the variable persist b/w different fun call.
5) keyword - static

Difference b/w auto & static variable.

Difference b/w auto & static variable is that, static variable is not re-initialised when the fun is called again, it is initialised just once and further calls of the function, share the same value of static variable

iv) External Storage Class

1) Storage - Memory
2) Initial value - zero
3) Scope - Global
4) life - variable hold the value as long as program execution does not come to end
5) keyword - extern.

Imp

Recursive function (Recursion)

It is function that call itself to solve a smaller version of its task until a final call is made which does not require a call to it self

Every recursive fun has 2 cases

1) Base case
2) Recursive case.

1] Base Case :- In base case the problem is simple enough to be solve directly without making any further calls to same function

2] recursive Case :-

Step 1] :- In this case first the problem is divided into sub parts

Step 2] :- function call itself with the sub parts of the problem obtain in the first step.

Step 3] :- the result is obtain by combining the solution of simpler sub parts

Q. Write a program to find factorial of given number using recursive fun

```
int fact(int n);
void main() {
    int n, answer;
    scanf("%d", &n);
    int answer = fact(n);
    printf("factorial is %d", answer);
}

int fact(int n) {
    int factN1, factN2;
    factN1 = fact(n-1);
    factN2 = factN1 * n;
    if(n==0) {
        return 1;
    }
```

return factN_2;
}

Wap to find fibonacci series using recursive

```
int fibn(int n);
int main(){
        int n, ans;
        scanf("%d", &n);
        ans = fibn(n);
        printf("%d", n);
    return 0;
}
int fibn(int n){
        int sum, sumN1, sumN2
        if(n==0){
            return 0;
        }
    sumN1 = fibn(n-1);
    sumN2 = n + sumN1;
    sum = sumN1 + sumN2;
    return sum;
}
```