1) What are Iterative Statements?

Iterative statements will help in repeated execution of some statements, depending on the value of some expression.

In C programming Language there are 3 iterative statements

i) While loop ii) do-while loop iii) for loop.

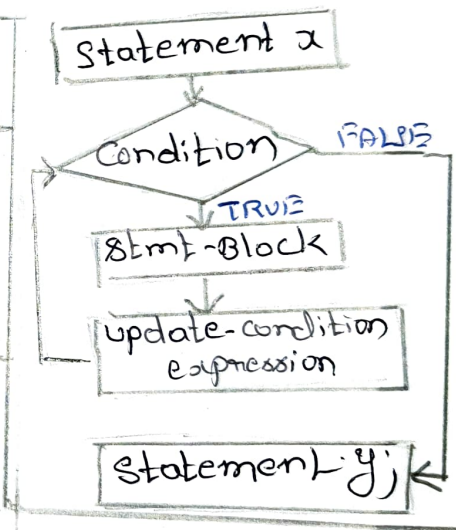
2) Explain While loop:

Repeat one-or-more statements depending on condition.

Syntax:

```
statement x;
While(condition)
{
    statement-Block;
    update condition expression;
}
statement y;
```

Flow Chart



Remember

1) Initialize condition variable before loop.
2) Update condition variable with in loop.

Example

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

Initialize condition variable.
condition.
update condition variable.

i<=10 (TRUE) / i=10 (FALSE)

i: 1 2 3 4 5 6 7 8 9 10 11

Display

```
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
```

While Loop Example Programs

1) Calculate Sum of numbers from m to n

I/O

Step1: Enter value of m:....

Step2: Enter value of n:....

Calculate Sum (Logic)

Step3: Sum = ...

Example I/O

Enter value of m: 5

Enter value of n: 10

Sum = 45

5+6+7+8+9+10

PROGRAM

```
#include<stdio.h>
main()
{
    int m, n, sum=0;
    Printf("\n Enter value of m: ");
    Scanf("%d", &m);                    -> Step-1
    Printf("\n Enter value of n: ");
    scanf("%d", &n);                    -> Step2

    While (m<=n)
    {
        sum= sum+m;
        m=m+1;                          -> Logic
    }
    Printf("\n Sum=%d", sum);           -> Step 3
}
```

Example

m	~~5~~	~~6~~	~~7~~	~~8~~	~~9~~	10			
n	10								
sum	~~0~~	~~5~~	~~11~~	~~18~~	~~26~~	~~35~~	45		

Write a program to read the no's until -1 is encountered. Also count the no. of negatives, Positives & zeros entered.

I/O Steps

1) Enter -1 to exit
2) Enter any numbers: ....
3) Until entered no ... ≠ -1 do the following
   Find whether the no. is +ve/-ve/0 (Logic)
4) Enter any numbers: ...
5) Positive numbers entered = ...
6) Negive numbers entered = ...
7) Zeros entered = ...

(5, 6, 7: After coming out of Loop (when user enters -1))

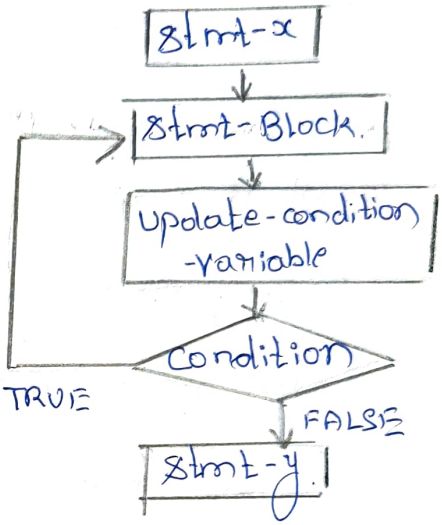
Program

```
#include<stdio.h>
main()
{
    int num;
    int neg=0, pos=0, zr=0;
1)  Printf("\n Enter -1 to exit");
2)  {printf("\n Enter any no: ");
    scanf("%d", &num);}
3)  While(num ≠ -1)
    {
        if(num>0)
            pos++;
        else if(num<0)
            neg++;
        else
            zr++;
4)  {printf("\n Enter any no: ");
    scanf("%d", &num);}
    }
5)  Printf("\n Positive nos entered
        = %d \n", pos);
6)  Printf("\n Negative nos entered
        = %d \n", neg);
7)  Printf("\n Zeros entered = %d",
        zr);
}
```

Explain do-while loop.

Repeat one-or-more statements depending on condition. condition is checked at the exit part.

FLOW-CHART



Syntax:

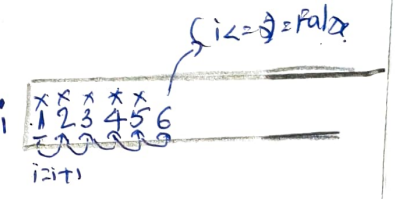
```
Stmt x;
do
{
    Stmt-Block;
}while(condition);
Stmty;
```

↑ don't forget ;

Example-Program

```
#include<Stdio.h>
main()
{
    int i=1;
    do
    {
        printf("\n i=%d \n", i);
        i=i+1;
    }while(i<=5);
}
```

→ Initialize condition variable



→ update condition variable.

→ condition

```
i=1
i=2
i=3
i=4
i=5
```

Sample Programs for Do-While Loop.

1) Write a program using a do-while loop to display the square & cube of first n natural numbers.

Steps

Step 1) Enter the value of n: __

Step 2) ----------

Step 3) initialize loop

Step 4) Print i, i*i, i*i*i;

Step 5) until i<=n

Step 6) ----------

Sample i/p.

Enter value of n: 4

1	1	1
2	4	8
3	9	27
4	16	64

Sample Program

```
#include<stdio.h>
main()
{
    int i, n;
    Printf("\n Enter the value of n: ");        -> Step 1
    scanf("%d", &n);
    Printf("\n ----------\n");                  -> Step 2
    i=1;                                        -> Step 3
    do
    {
      Printf("\n %d \t %d \t %d \n", i, i*i, i*i*i);   -> Step 4
      i++;
    } while(i<=n);                              -> Step 5
    Printf("\n ----------\n");                  -> Step 6
}
```

2) Write a program to list all the leap years from 1900 to 1920:

Steps

S1) m=1900 , n=1920

~~Until m<=n~~

do

S2) → if m is divisible by 4 i.e, m%4==0
& m is not divisible by 100 i.e, m%100!=0
& m is divisible by 400 i.e, m%400==0

S3) Then print m is a Leap year

S4) m=m+1;

S5) continue the above 2 steps as long as m<=n.

Program

```
#include<stdio.h>
main()
{
    int m=1900, n=1920; — S1
    do
    {
        if((m%4==0) && (m%100!=0) && (m%400)==0))  → S2
            Printf("\n year %d is Leap year \n", m); → S3
        m++; → S4
    }while(m<=n); → S5

}
```

Write a program to read the characters until a * is encountered. Also count the number of upper case, lower case & numbers entered.

Steps

S1 → Enter any character: ---

Find whether entered character is upper case letter / Lower case Letter / Digit [LOGIC]

S2 → Continue above until entered character is not *

S3 Print uppercase=---, Lowercase=---- & Digits=1---

Program:

```
#include <stdio.h>
main()
{
    char ch;
    int up=0, low=0, dig=0;
    do
    {
        printf("\n Enter any character: ");   → S1
        scanf("%c", &ch);
        if(ch>='A' && ch<='Z')
            up++;
        else if(ch>='a' && ch<='z')
            low++;
        else if(ch>='0' && ch<='9')
            dig++;
    }while(ch!='*');   → S2
```

Logic

```
printf("\n Uppercase = %d \t Lowercase = %d \t
        Digits = %d", up, low, dig);
                                    ↪ S3.
}
```

Explain for loop.

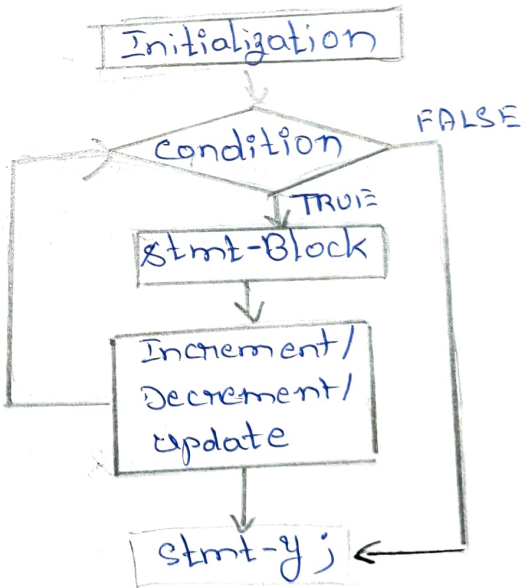
Used to repeatedly execute one-or-more statements according to the condition.

Note:→ Don't forget semicolons
→ According to logic you can skip any part of for stmt. Ex: for(;;) is ok.

Syntax

```
for(initialization; condition; increment/decrement/update)
{
    stmt-Block;
}
stmt y;
```

FLOW-CHART



Example program

```
#include<stdio.h>
main()
{
   int i;
   for(i=1; i<=5; i++)
   {   printf("\n i = %d \n", i);
   }
}
```

Note: Widely used for simplicity. Unlike while & do-while all the 3 phases of looping - Initialization, condition-check & update happens in one line.

Note about for Loops:

1)
```
int i=1;
for( ; i>=10; i++)
{

}
```
This is fine.

2) `for(i=1, j=1; i<10; i++)` ✓

3)
```
for(i=1; i<10; i++);
{
    printf("%d", i);
}
```
ok. But compiler will execute complete for loop. the printf will be executed once.
so only 10 will be printed

NOTE

1) `while(---)` → No semicolon

```
{

}
```

```
do
{

}while(---);
```
↓ semicolon

`for(i------);` → not required

2)
```
for(i=2; i<=50; i+=2)
{
    printf("\n i=%d", i);
}
```
✓

4)
```
i=0;
for( ; i<=10; )
{
    printf("%d", i);
    i++;
}
```
✓

4) `for(;;)`
`printf("Hi");`
→ Infinite Loop.
→ Program will not stop

5) If there is only 1 statement in stmt-block then there is no need of flower brackets { }

```
while(i<=10)
{
    printf("%d", i);
    i++;
}
```

```
for(i=0; i<=10; i++)
    printf("%d", i);
```

On what basis we select a particular Loop Statement

Loop Statements are differentiated according to various Parameters.

1) Entry Controlled Vs Exit Controlled.

Entry Controlled:
- 1) Condition is checked before entering Loop.
- 2) In the beginning itself if condition becomes false then Loop will not be executed at all
- → while() & for() Loops

Exit Controlled:
- → condition is checked after execution of loop
- Atleast once the Loop will be executed.
- → do-while Loop.

2) Counter-controlled Vs Sentinel value.

Counter-controlled:
- → We know in advance the number of iteration (Ex: n times)
- → Here we use for loop

Sentinel value:
- → We don't know in advance how many iterations
- → We use sentinel value Ex: Execute loop as long as user enters 'x'
- → Here, we use while() & do-while()

Explain - Nested Loops

→ Loops placed inside other Loops are called as nested Loops

→ This feature is there for all loops, but most widely used for "for" Loops.

→ Nesting to any level can be possible.

→ Example

```
#include<stdio.h>
main()
{
    int i,j;
    for(i=1; i<=5 ;i++)
    {
        printf("\n");
        for(j=1; i<=10; j++)
            printf("*");
    }
}
```

5 rows

10 stars in each row

output

Explain break statement

→ Used to terminate a ~~group~~ Loop.

Note: only inside Loop;

→ Syntax: break;

→ When break statement is encountered Loop will be terminated abruptly & next-statement after Loop will be executed.

→ Example

```
#include<stdio.h>
main()
{
    int i=1;
    while(i<=10)
    {
        if(i==5)
            break;
        printf("\ni=%d", i);
        i=i+1;
    }
}
```

output

```
i=1
i=2
i=3
i=4
```

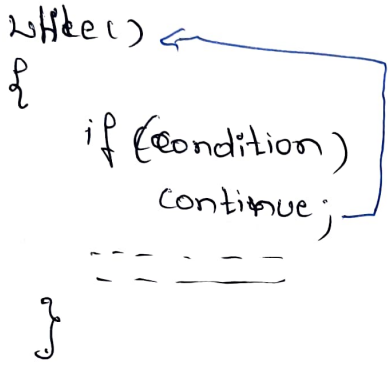
When i becomes 5 execution will jump out of loop.

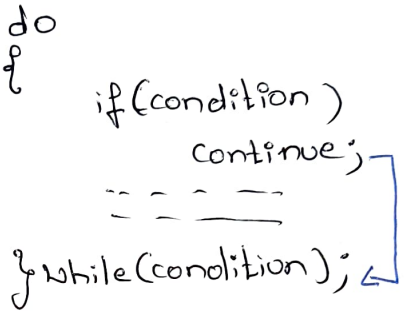
Explain continue Statement.

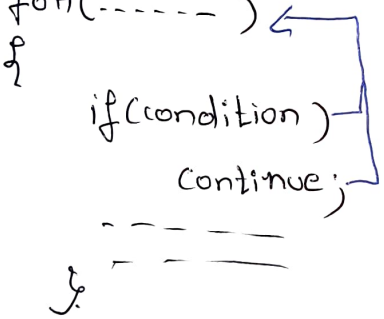
→ Used to skip the remaining part of the loop & goto next iteration.

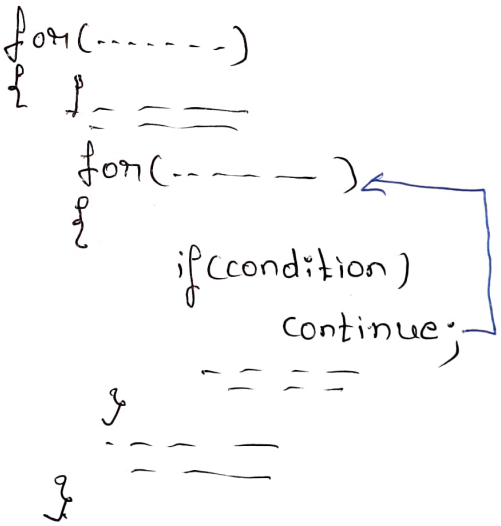
→ Note: Only inside Loop.

→ Syntax: continue;









→ Example

```
#include<stdio.h>
main()
{
    int i;
    for(i=1; i<=6; i++)
    {
        if(i==4)
            continue;
        printf("\n i=%d", i);
    }
}
```

output

```
i=1
i=2
i=3
i=5
i=6
```

When i = 4, control will go back to for loop & do i++ & check i<=6

Explain goto statement.

> Used to transfer control to a specified label.

>Syntax: goto Label;

Forward Jump

goto Label;

label:
statements;

Backward Jump

Label:
statements

goto Label;

Note: After label don't forget to put colon(:)
Often goto statement is mixed with if condition.
Ex: if(condition)
goto Label;

> Example

```
#include<stdio.h>
main()
{
    int num, sum=0;
    read:
        printf("Enter a no. Enter 999 to close \n");
        scanf("%d", &num);
        if(num != 999)
        {
            if(num <0)
                goto read;
            sum = sum+num;
            goto read;
        }
    printf("\n sum= %d", sum);
}
```