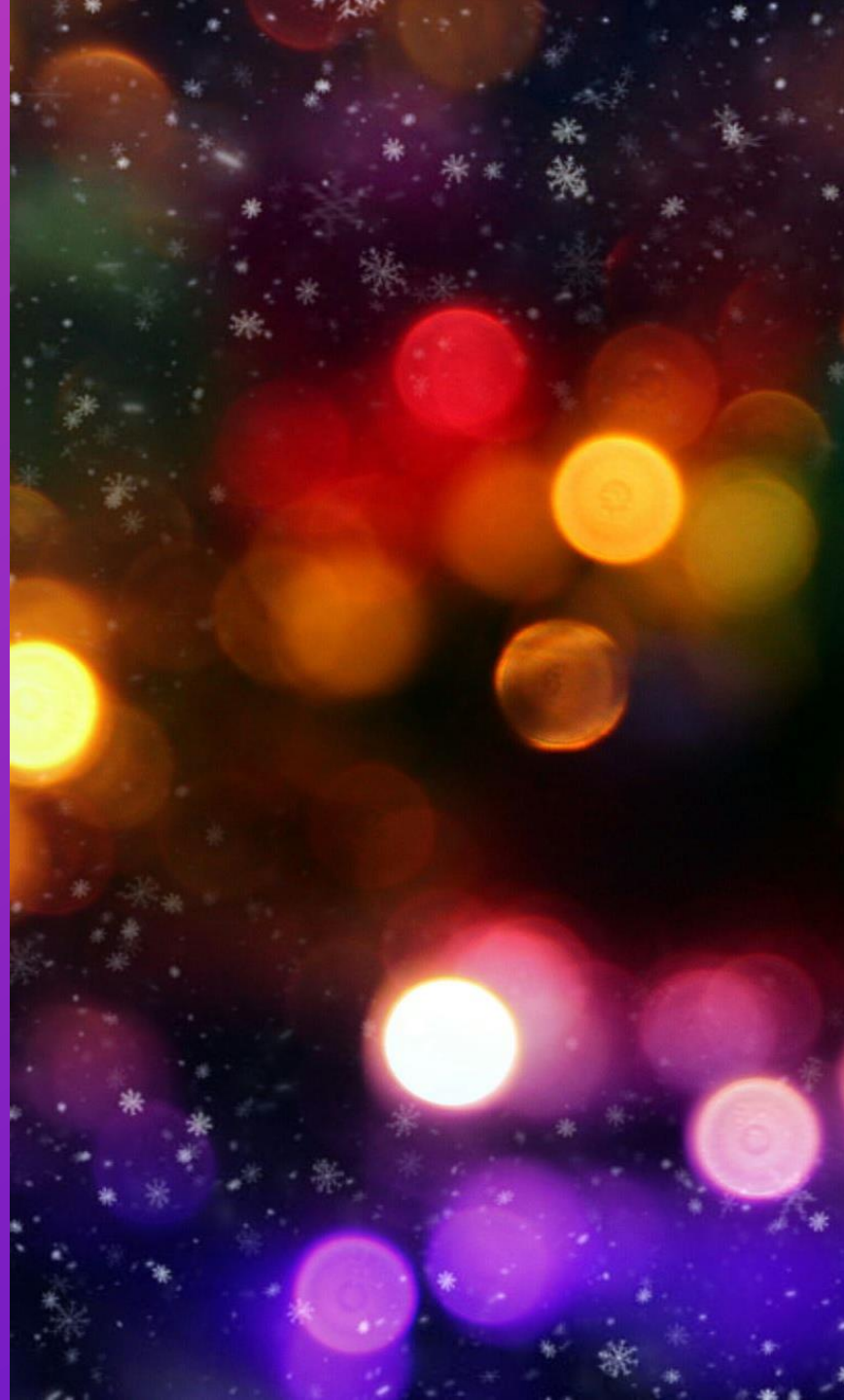
MNNIT COMPUTER CODING CLUB

CLASS-4

BASICS OF C



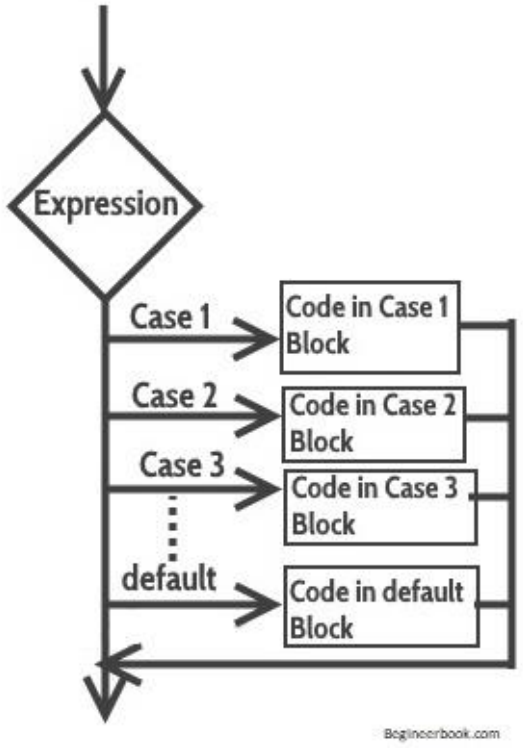
SWITCH CASE

The switch statement allows us to execute one code block among many alternatives.
You can do the same thing with the `if...else..if` ladder. However, the syntax of the `switch` statement is much easier to read and write.

Syntax:

```
switch (expression)
{
    case constant1:
      // statements
      break;

    case constant2:
      // statements
      break;
    .
    .
    default:
      // default
statements
}
```



Note:
The break statement is optional. If omitted, execution will continue on into the next case. The flow of control will fall through to subsequent cases until a break is reached.

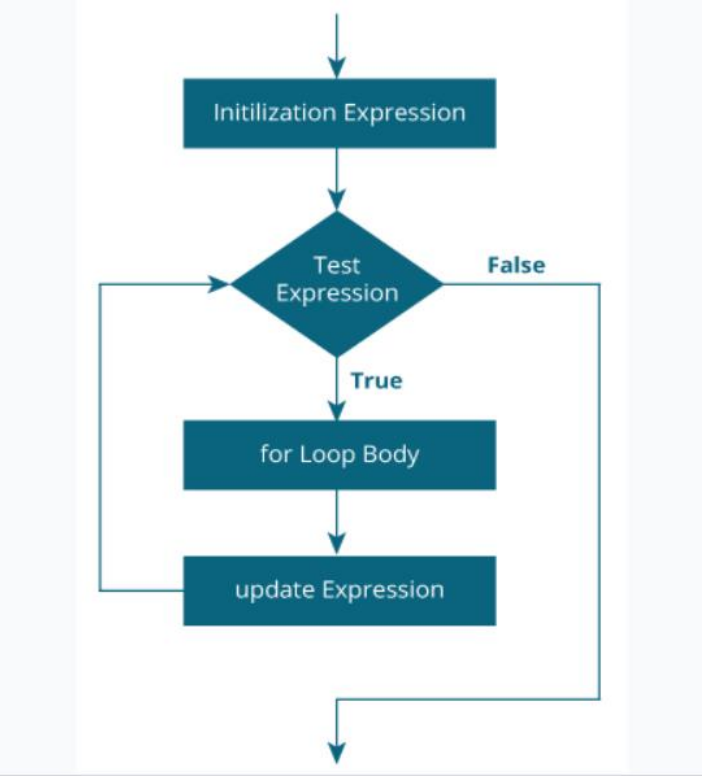
LOOPS IN C

In programming, a loop is used to repeat a block of code until the specified condition is met.

C programming has three types of loops:

- for loop
- while loop
- do...while loop

FOR LOOP



How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of for loop are executed, and the update expression is updated.
- Again, the test expression is evaluated.
- This process goes on until the test expression is false. When the test expression is false, the loop terminates.

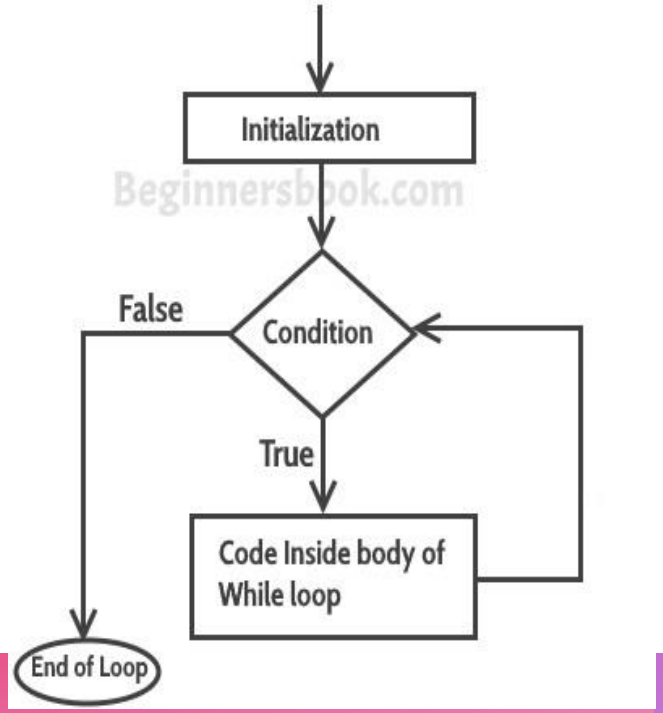
```
for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}
```

```c
#include<stdio.h>
int main()
{
    int i;
    for(i=0; i<5; i++)
    {
        printf("Value of i: %d\n", i);
    }
}
```

WHILE LOOP

How while loop works?

- The initialization statement is kept out and executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the wile loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of while loop are executed. These also contains update statements if any.
- Again, the test expression is evaluated.
- This process goes on until the test expression is false. When the test expression is false, the loop terminates.

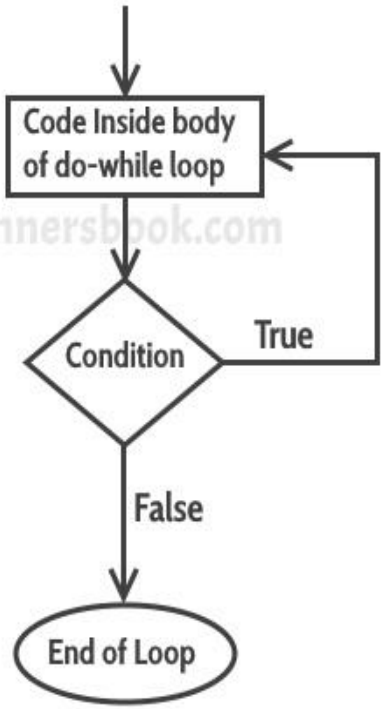



```c
#include<stdio.h>
int main()
{
    int i; //Initialisation
    while(i<5) //Test Condition
    {
        printf("Value of i: %d", i); //Body
        i++; //Updation
    }
}
```

DO-WHILE LOOP

How DO while loop works?

- The initialization statement is kept out and executed only once.
- Then, the body of the loop gets executed first including any updation specified inside body
- Then, the body exits, and test expression specified in while() is executed
- If the test expression is true, again the statements inside body are executed and testing is done untill condition becomes false
- If the test expression is false, then the statements after while(); are executed.




```c
#include<stdio.h>
int main()
{
    int i; //Initialisation
    do
    {
        printf("Value of i: %d", i); //Body
        i++; //Updation
    }
    while (i<5);
}
```

Difference between do while and while Loop

do-while	while
It is exit controlled loop	It is entry controlled loop
The loop executes the statement at least once	loop executes the statement only after testing condition
The condition is tested before execution.	The loop terminates if the condition becomes false.
There is semicolon at the end of while statement.	There is no semicolon at the end of while statement

INFINITE LOOP

The loops that go on executing infinitely and never terminate are called infinite loops. Sometimes we write these loops by mistake while sometimes we deliberately make use of these loops in our programs.

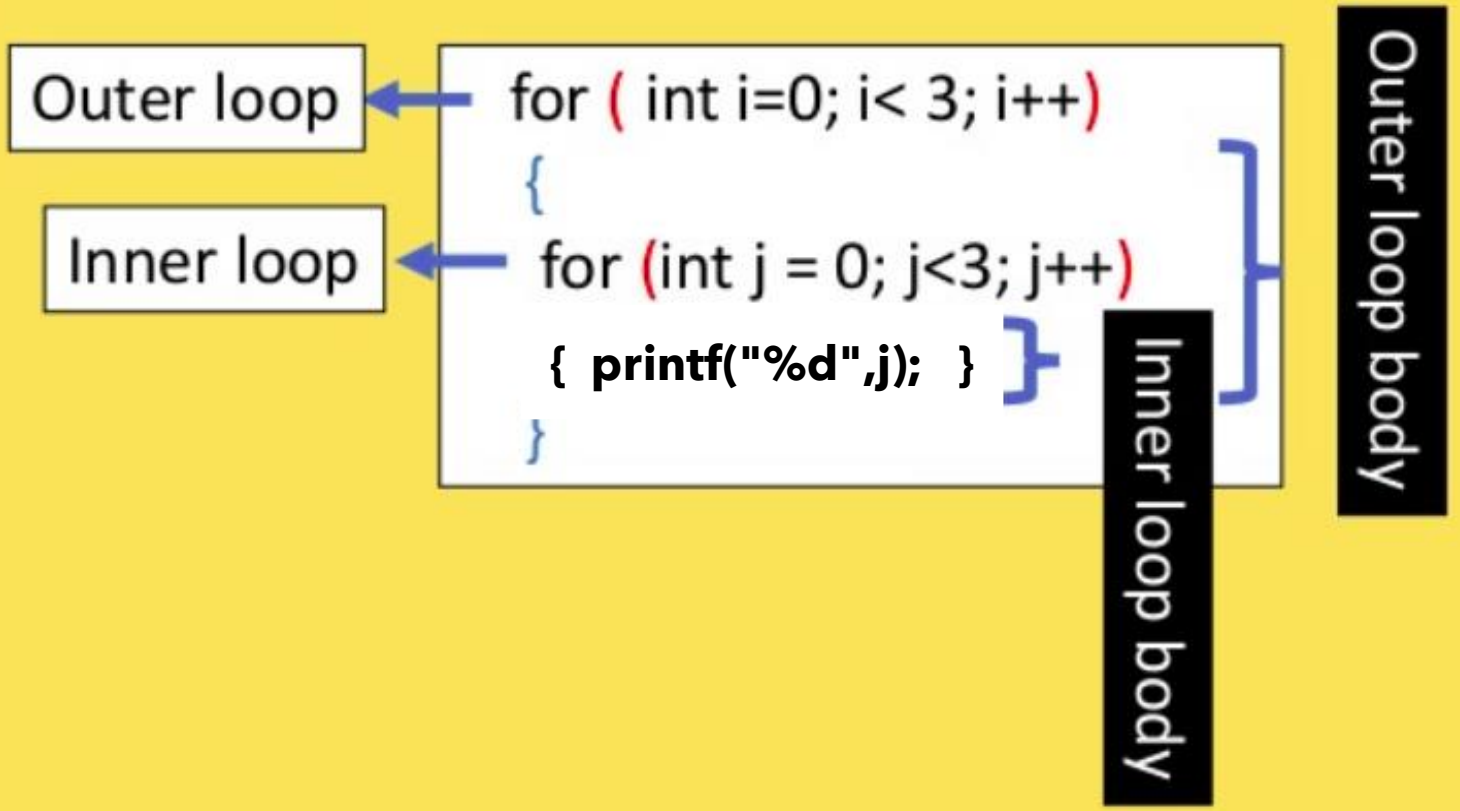
```c
#include<stdio.h>
int main()
{
    int n;
    for(n=1;n<=5;)
    {
        printf("This is an infinite loop\n");
    }
}
```

To stop the running of Infinite loop use: **ctrl+C**

It is important to stop the running of Infinite loop ASAP otherwise your system might get hanged.

What are nested loops?

- A nested loop is a loop inside the body of another loop.
- The nested loop is known as the **inner loop** and the loop in which it is nested is known as the **outer loop**



Working of Nested Loops

```c
#include<stdio.h>
int main()
{
    int i, j;
    for(i = 0; i < 2; i++)
    {
        printf("This is outer loop body\n");
        for (j = 0; j < 2; j++)
        {
            printf("This is inner loop body\n");
        }

    }
}

/*
This is outer loop body
This is inner loop body
This is inner loop body
This is outer loop body
This is inner loop body
This is inner loop body
*/
```

1. STEP 1:The outer loop is initialized with value of i as 0
2. STEP 2: Value of i is tested, since the condition is true(i<2), the loop is entered
3. STEP 3:A newline is displayed This is outer loop body This is part of outer loop.
4. STEP 4 :The control goes to inner loop, where j is initialized with 0
5. STEP 5: Value of j is tested, j<2 is true, inner loop is entered
6. STEP 6: The statement This is innerloop body is executed, value of j is displayed
7. STEP 7: The value of j is incremented.
8. Now STEPs 5, 6 and 7 are repeated till the condition ,j<2, becomes false.
9. When value of j is 2, control comes out of inner loop.
8. STEP 8: Now the control goes to outer loop update statement, i is incremented.
9. STEPs 2 – 7 are repeated. The steps are repeated for value of i = 1 This continues till value of i becomes 2. Then the outer loop is terminated.

PYRAMIDS

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

(a)

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

(b)

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15

(c)

1
1 1
1 1 1
1 1 1 1
1 1 1 1 1

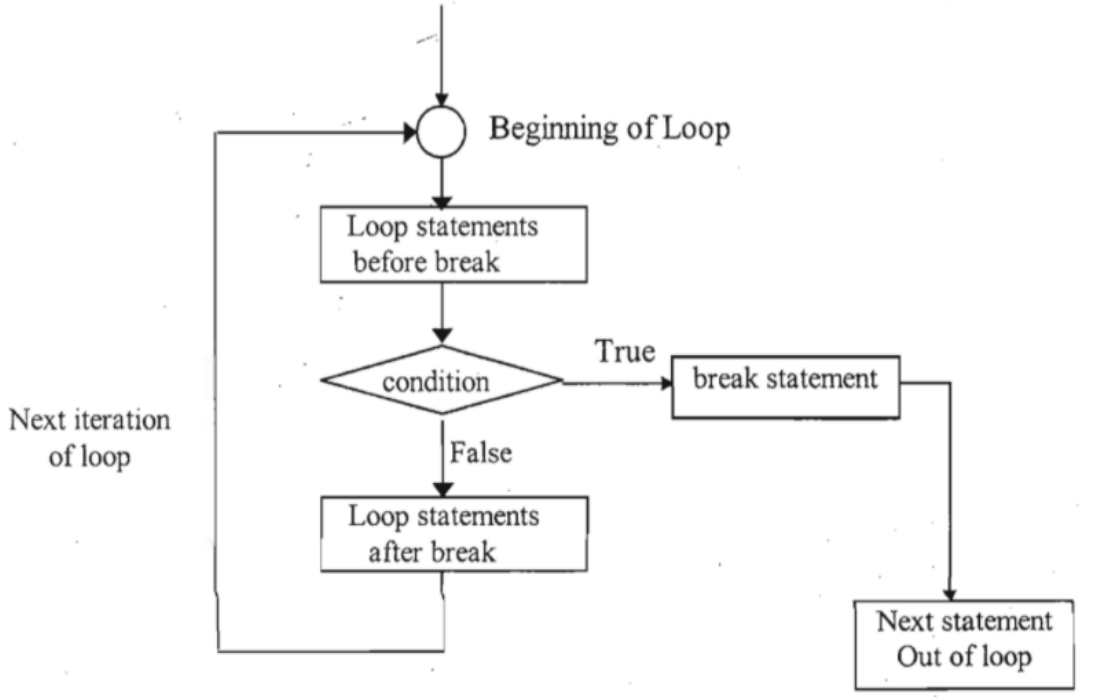
(d)

SOLUTION(B)

```
#include<stdio.h>
int main() {
    int i, j;
    for(i=1; i<=5; i++) {
        for(j=1; j<=i; j++) {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```

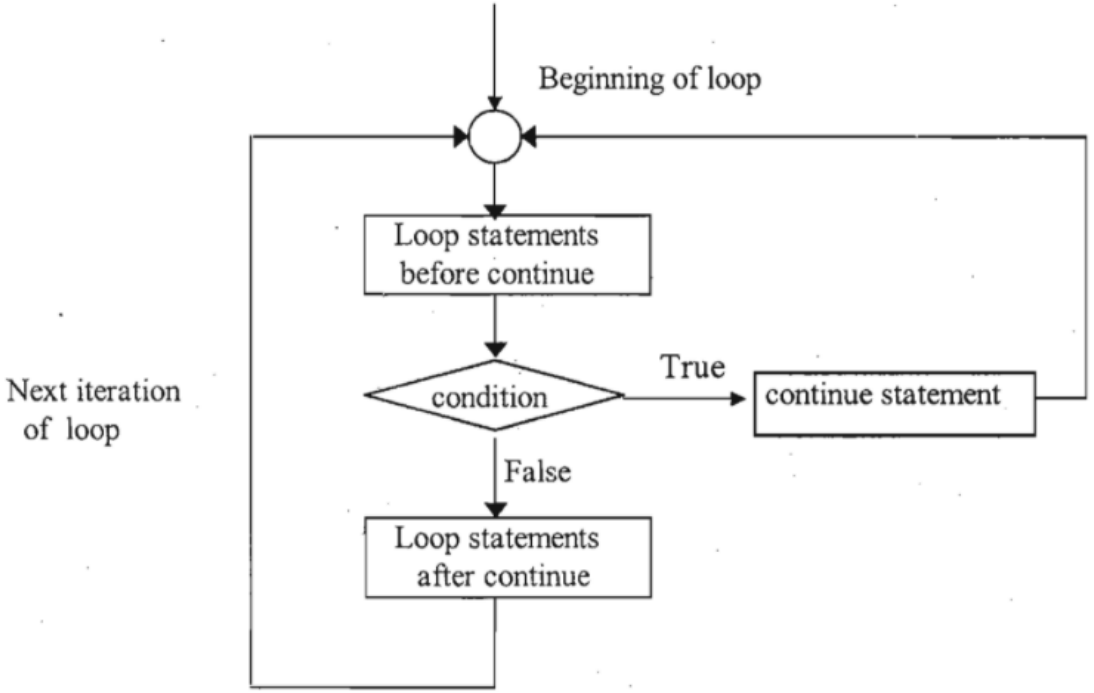
BREAK AND CONTINUE

Break: It is used to terminate the loop. This statement causes an immediate exit from that loop in which this statement appears, and the control is transferred to the statement immediately. after the loop.



Break (control statement)

Continue: It is used when we want to go to the next iteration of the loop after skipping some statements of the loop.



continue (control statement)

NOTE:

- In while and do-while loops, after continue statement the control is transferred to the test condition and then the loop continues
- Whereas in **for** loop after continue statement the control is transferred to update expression and then the condition is tested.

```c
#include<stdio.h>
int main()
{
    int n;
    for(n=1;n<=5;n++)
    {
        if (n==3)
        {
            printf ("I understand the use of break\n");
            break;
        }
        printf ("Number %d\n", n);
    }
    printf ("Out of for loop\n");
}

/*
Output:
Number 1
Number 2
I understand the use of break
Out of for loop
*/
```

```c
#include<stdio.h>
int main()
{
    int n;
    for(n=1;n<=5;n++)
    {
        if (n==3)
        {
            printf ("I understand the use of continue\n");
            continue;
        }
        printf ("Number %d\n", n);
    }
    printf ("Out of for loop\n");
}

/*
Output:
Number = 1
Number" = 2
I understand the use of continue
Number = 4
Number = 5
Out of for loop
*/
```