

MNNIT COMPUTER CODING CLUB

**CLASS-4**

**BASICS OF C**



# 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.

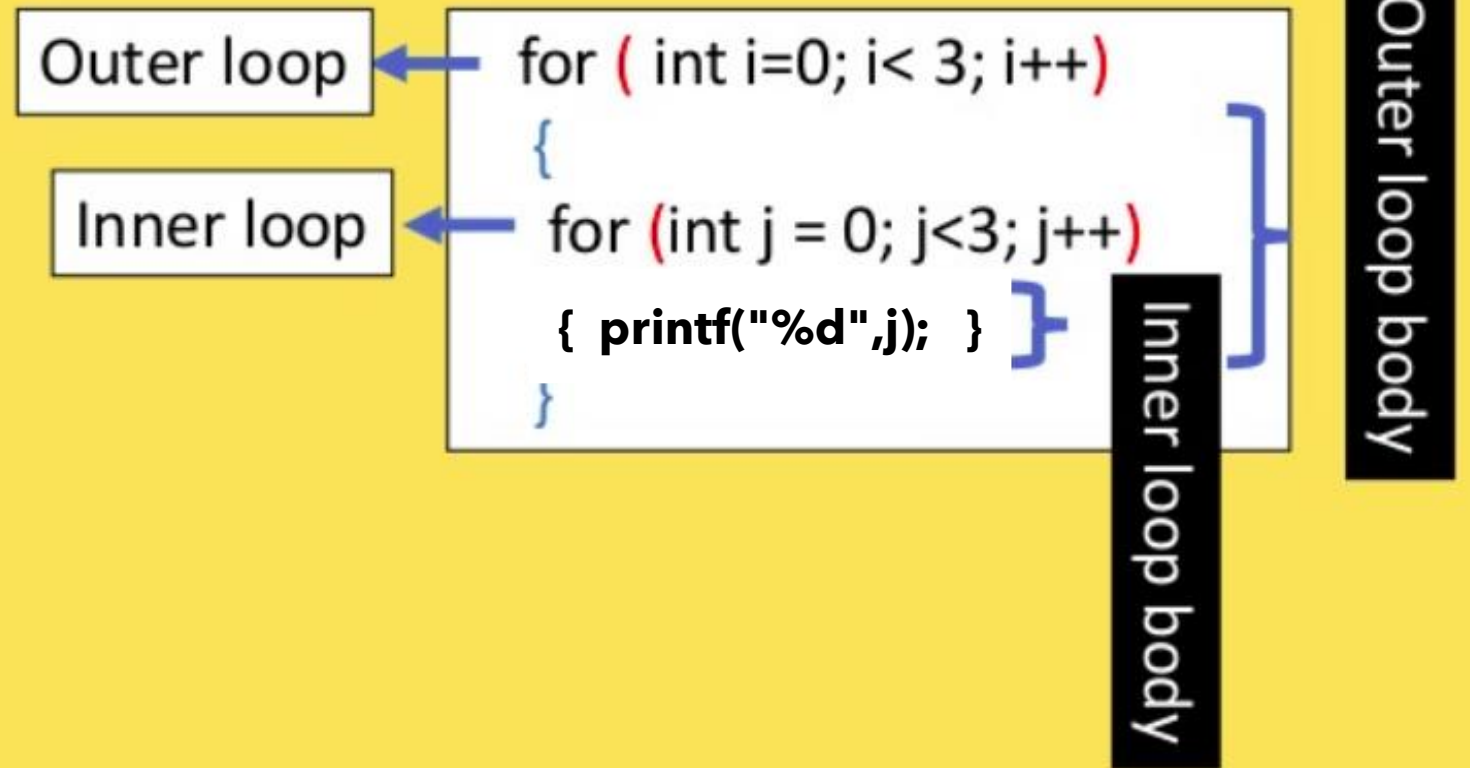
```
1  #include<stdio.h>
2  int main()
3  {
4      int n;
5      for(n=1;n<=5;)
6      {
7          printf("This is an infinite loop\n");
8      }
9  }
10
```

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int i, j;
5     for(i = 0; i < 2; i++)
6     {
7         printf("This is outer loop body\n");
8         for (j = 0; j < 2; j++)
9         {
10            printf("This is inner loop body\n");
11        }
12    }
13 }
14
15
16 /*
17 This is outer loop body
18 This is inner loop body
19 This is inner loop body
20 This is outer loop body
21 This is inner loop body
22 This is inner loop body
23 */
```

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)

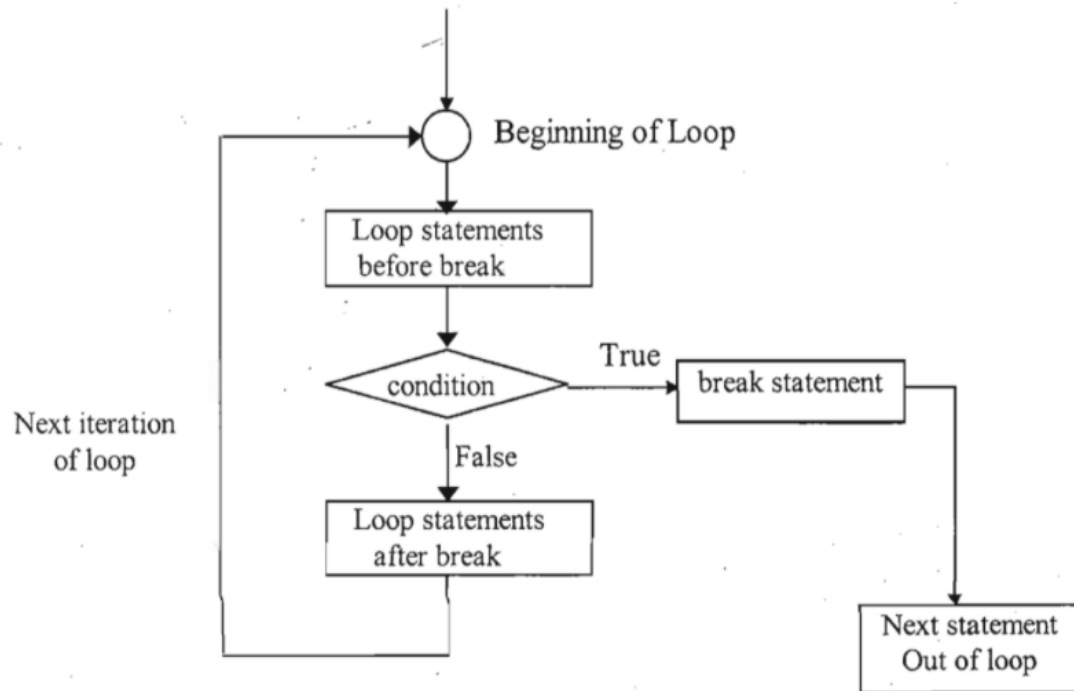
```
#include<stdio.h>

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



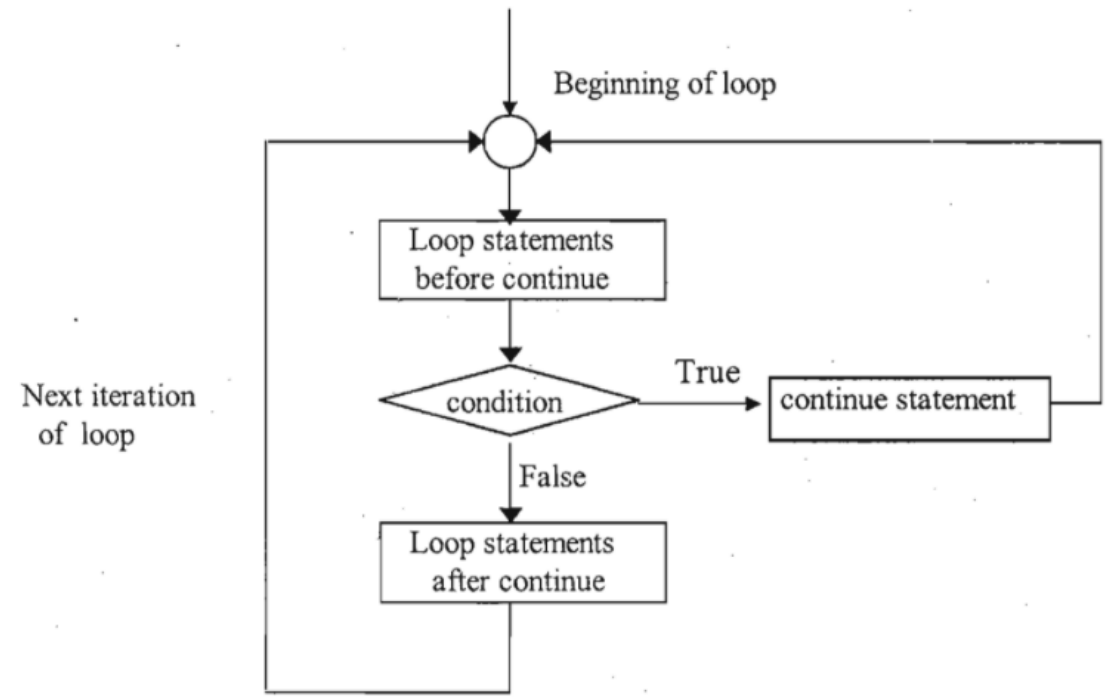
# BREAK AND CONTINUE

**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.

```
1  #include<stdio.h>
2  int main()
3  {
4      int n;
5      for(n=1;n<=5;n++)
6      {
7          if (n==3)
8          {
9              printf ("I understand the use of break\n");
10             break;
11         }
12         printf ("Number %d\n", n);
13     }
14     printf ("Out of for loop\n");
15 }
16
17 /*
18 Output:
19 Number 1
20 Number 2
21 I understand the use of break
22 Out of for loop
23 */
```

```
1  #include<stdio.h>
2  int main()
3  {
4      int n;
5      for(n=1;n<=5;n++)
6      {
7          if (n==3)
8          {
9              printf ("I understand the use of continue\n");
10             continue;
11         }
12         printf ("Number %d\n", n);
13     }
14     printf ("Out of for loop\n");
15 }
16
17 /*
18 Output:
19 Number = 1
20 Number" = 2
21 I understand the use of continue
22 Number = 4
23 Number = 5
24 Out of for loop
25 */
```



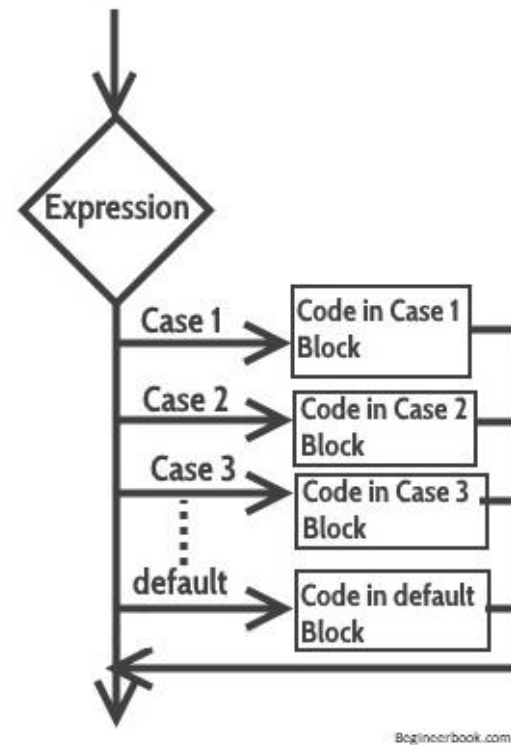
# SWITCH CASE

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.