VTC Academy
Think Ahead, Beyond Boundaries

BASIC PROGRAMMING LANGUAGE

LESSON 5

Loop Statements

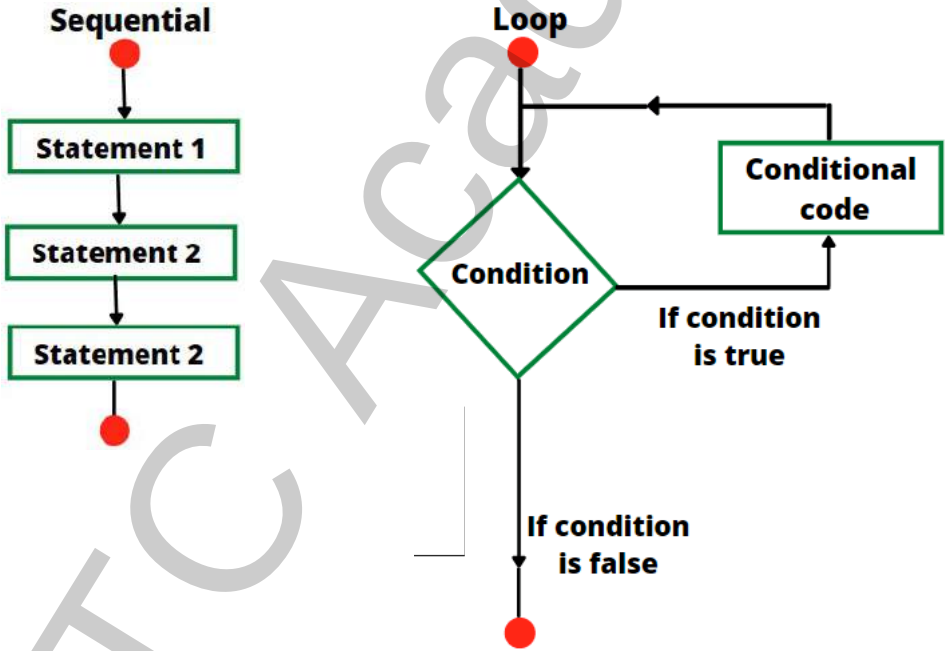
CONTENT

VTC Academy
Think Ahead, Beyond Boundaries

1. Concept of Loop Statements
2. `for` Statement
3. `while` Statement
4. `do…while` Statement
5. `break` and `continue` Statement
6. Nested Loop
7. Infinite Loop
8. Summary

Loop Statements

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



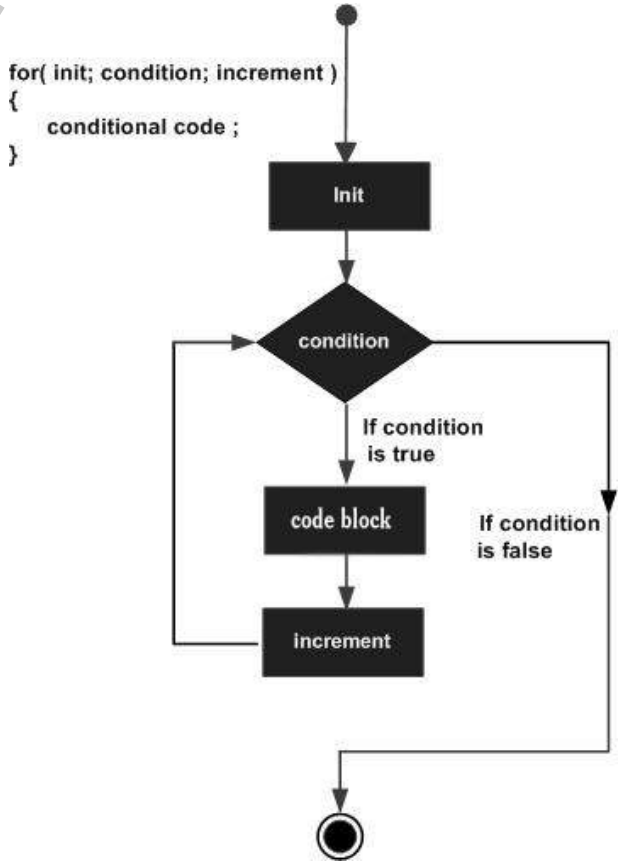
Loop Statements

- A loop statement consists of two parts, a body of a loop and a control statement.
- The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false.
- Loop statements available in C are:
  - `for` statement
  - `while` statement
  - `do...while` statement

for Statement

- A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- `for` statement syntax:

```
for (init; condition; increment)
{
    /* statement(s) */
}
```



How for Statement Work?

VTC Academy
Think Ahead, Beyond Boundaries

- The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables.
- Next, the condition is evaluated. If it is true, the body of the loop is executed. Else, the loop will stop and run next statement after loop.
- After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables.
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself. If condition is false, loop terminates.

for Statement Example

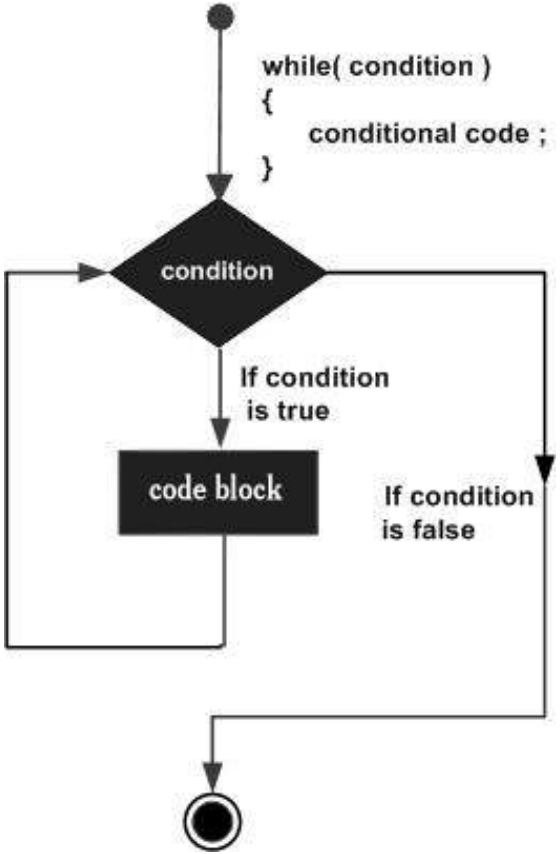
VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>

int main()
{
    int a;
    for (a = 1; a < 10; a = a + 1)
    {
        printf("Value of a: %d\n", a);
    }
    return 0;
}
```

while Statement

- The while loop is a pre-test loop; the condition is tested before the statements in the body of the loop are executed
- While the test condition is true, execute the statements in the body of the loop. The loop terminates when the condition becomes false
- Syntax:

```
while (condition) {
    /* statement(s) */
}
```



while Statement

VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>

int main()
{
    int a = 1;
    while (a < 10)
    {
        printf("Value of a: %d\n", a);
        a++;
    }
    return 0;
}
```

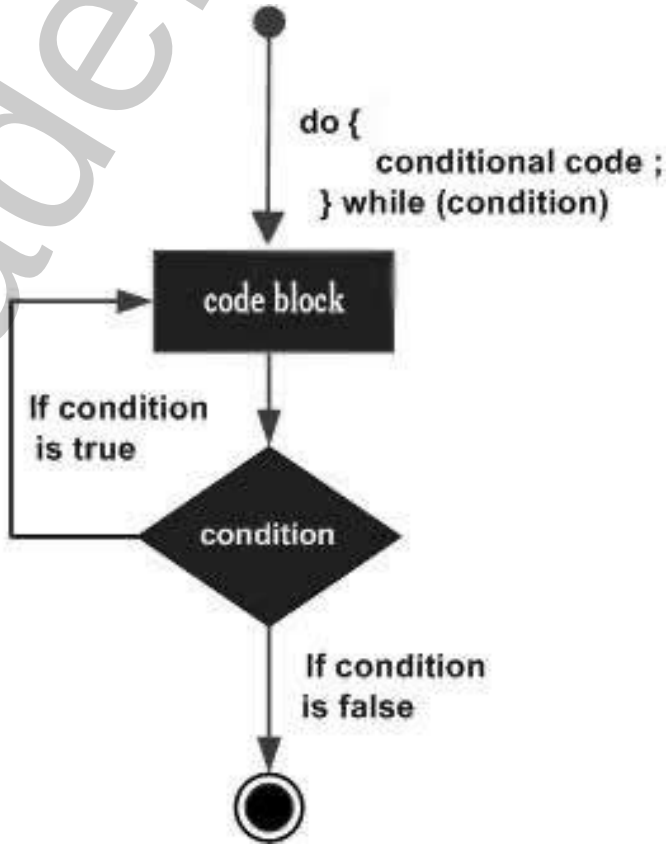
do...while Statement

- The do-while loop is a post-test loop; the condition is tested after the statements in the body of the loop have been executed.
- As long as the test condition is true, the program will go back and repeat the body of the loop. The loop terminates when this condition becomes false.
- Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming checks its condition at the bottom of the loop.
- The body of do...while loop is executed at least once.

do...while Statement

- Syntax:

```
do {
    /* statement(s) */
} while (condition);
```



do...while Statement Example

VTC Academy
Think Ahead, Beyond Boundaries

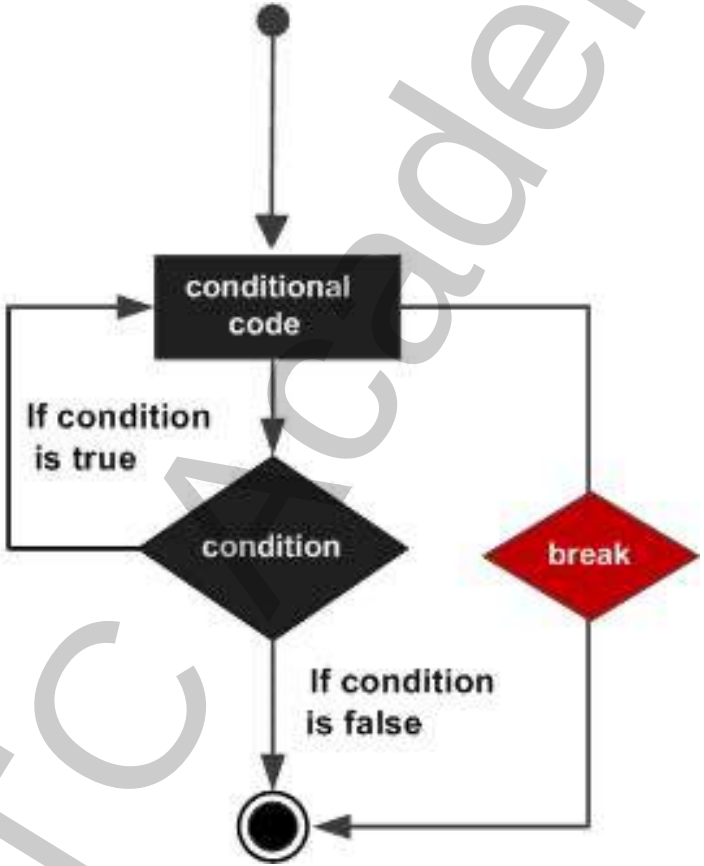
```c
#include <stdio.h>

int main()
{
    int a = 1;
    do {
        printf("Value of a: %d\n", a);
        a++;
    } while (a < 10);
    return 0;
}
```

break Statement

VTC Academy
Think Ahead, Beyond Boundaries

- The `break` statement is used to terminate a case in a `switch` statement.
- It can also be used for abrupt termination of a loop.
- When the `break` statement is encountered in a loop, the loop is terminated immediately and control is passed to the statement following the loop.
- If you are using nested loops, the `break` statement will stop the execution of the innermost loop and start executing the next line of code after the block.

break Statement



break Statement Example

VTC Academy
Think Ahead, Beyond Boundaries

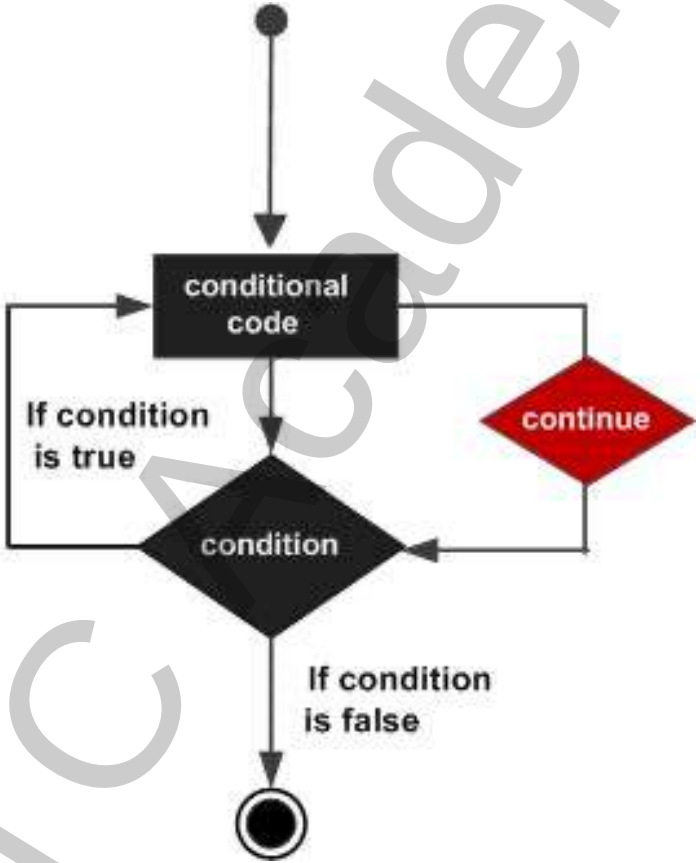
```c
#include <stdio.h>
int main()
{
    int a = 10;
    while (a < 20)
    {
        printf("Value of a: %d\n", a);
        a++;
        if (a > 15)
        {
            break;
        }
    }
    return 0;
}
```

continue Statement

VTC Academy
Think Ahead, Beyond Boundaries

- The `continue` statement causes the next iteration of the enclosing loop to begin.
- When this statement is encountered, the remaining statements in the body of the loop are skipped and the control is passed on to the re-initialization step.
- For the `for` loop, continue statement causes the conditional test and increment portions of the loop to execute.
- For the `while` and `do...while` loops, continue statement causes the program control to pass to the conditional tests.

continue Statement



continue Statement Example

```c
#include <stdio.h>
int main()
{
    int a = 10;
    do
    {
        if (a == 15)
        {
            a = a + 1;
            continue;
        }
        printf("Value of a: %d\n", a);
        a++;
    } while (a < 20);
    return 0;
}
```

Nested Loop

- C programming allows to use one loop inside another loop. It is called nester loop.
- For each iteration of the outer loop, the inner loop repeats its entire cycle.
- Syntax of `for` nested loop:

```
for (init; condition; increment) {  // outer loop
    for (init; condition; increment) {  // inner loop
        statement(s);
    }
    statement(s);
}
```

Nested Loop

- Syntax of `while` nested loop:

```
while (condition) {  // outer loop
    while (condition) {  // inner loop
        statement(s);
    }
    statement(s);
}
```

Nested Loop Example

VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>
int main() {
    int i, j;
    for(i = 2; i < 100; i++) {
        for (j = 2; j <= (i/j); j++) {
            if (!(i%j)) {
                break;
            }
        }
        if (j > (i/j)) {
            printf("%d is prime\n", i);
        }
    }
    return 0;
}
```

Infinite Loop

- A loop becomes an infinite loop if a condition never becomes false.
- The for loop is traditionally used for this purpose. Since none of the three expressions that form the 'for' loop are required, you can make an endless loop by leaving the conditional expression empty.

Infinite Loop Example

VTC Academy Think Ahead, Beyond Boundaries

- An example of infinite loop using `for` statement:

```c
#include <stdio.h>

int main()
{
    for ( ; ; )
    {
        printf("This loop will run forever.\n");
    }
    return 0;
}
```

Infinite Loop Example

VTC Academy
Think Ahead, Beyond Boundaries

- An example of infinite loop using `while` statement:

```c
#include <stdio.h>

int main()
{
    while (1)
    {
        printf("This loop will run forever.\n");
    }
    return 0;
}
```

Summary

- In C programming, a loop statement is used to repeat a block of code until the specified condition is met.
- Loop statements available in C are:
  - `for` statement
  - `while` statement
  - `do...while` statement
- The `break` statement will terminate the loop and control is passed to the statement following the loop.
- The `continue` statement causes the next iteration of the enclosing loop to begin.

VTC Academy
Think Ahead, Beyond Boundaries