SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

COURSE NAME : 23ITT101 PROGRAMMING IN C & DATA STRUCTURES I YEAR/

II SEMESTER

UNIT-II C DECISION STATEMENTS & FUNCTIONS

Topic: Looping Statements

M.Mohana Priya
Assistant Professor
Department of Artificial Intelligence and Machine Learning

C Looping Statements

In any programming language including C, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.

The looping statements are used to execute a single statement or block of statements repeatedly until the given condition is FALSE.

Why use loops in C language?

The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times. For example, if we need to print the first 10 natural numbers then, instead of using the printf statement 10 times, we can print inside a loop which runs up to 10 iterations.

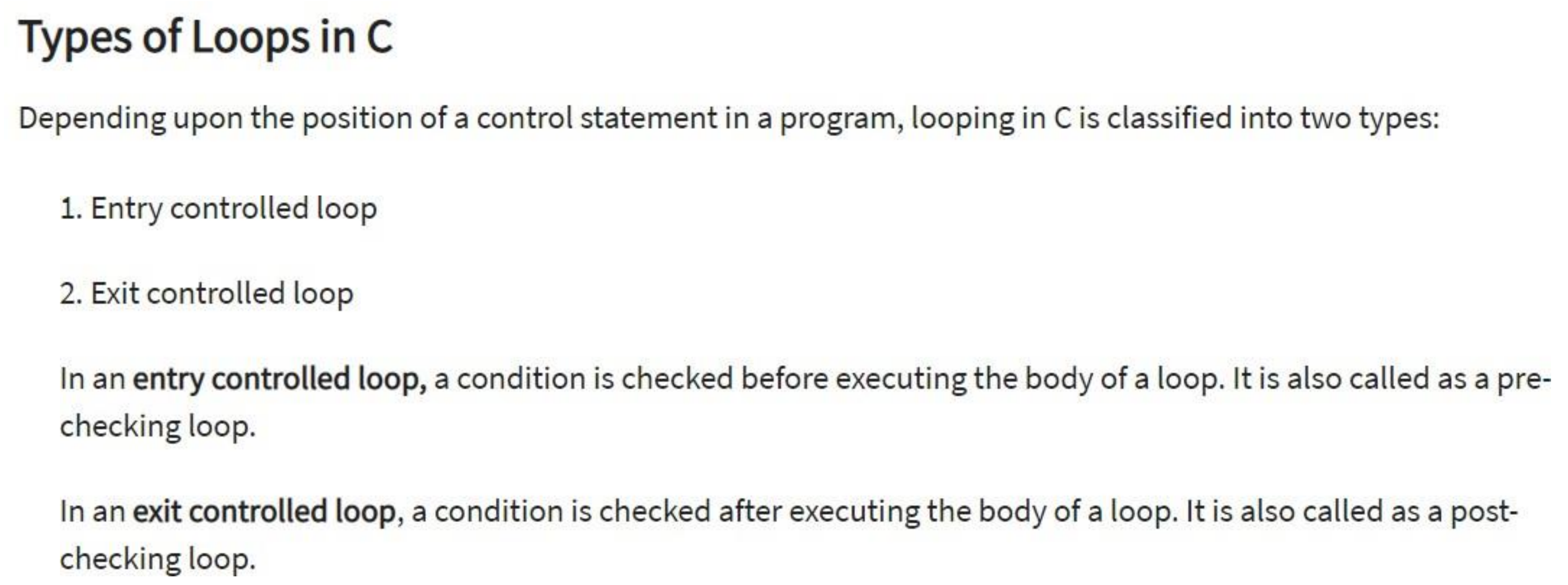
Advantage of loops in C

1) It provides code reusability.

2) Using loops, we do not need to write the same code again and again.

3) Using loops, we can traverse over the elements of data structures (array or linked lists).

C Looping Statements



Types of Loops in C

Depending upon the position of a control statement in a program, looping in C is classified into two types:

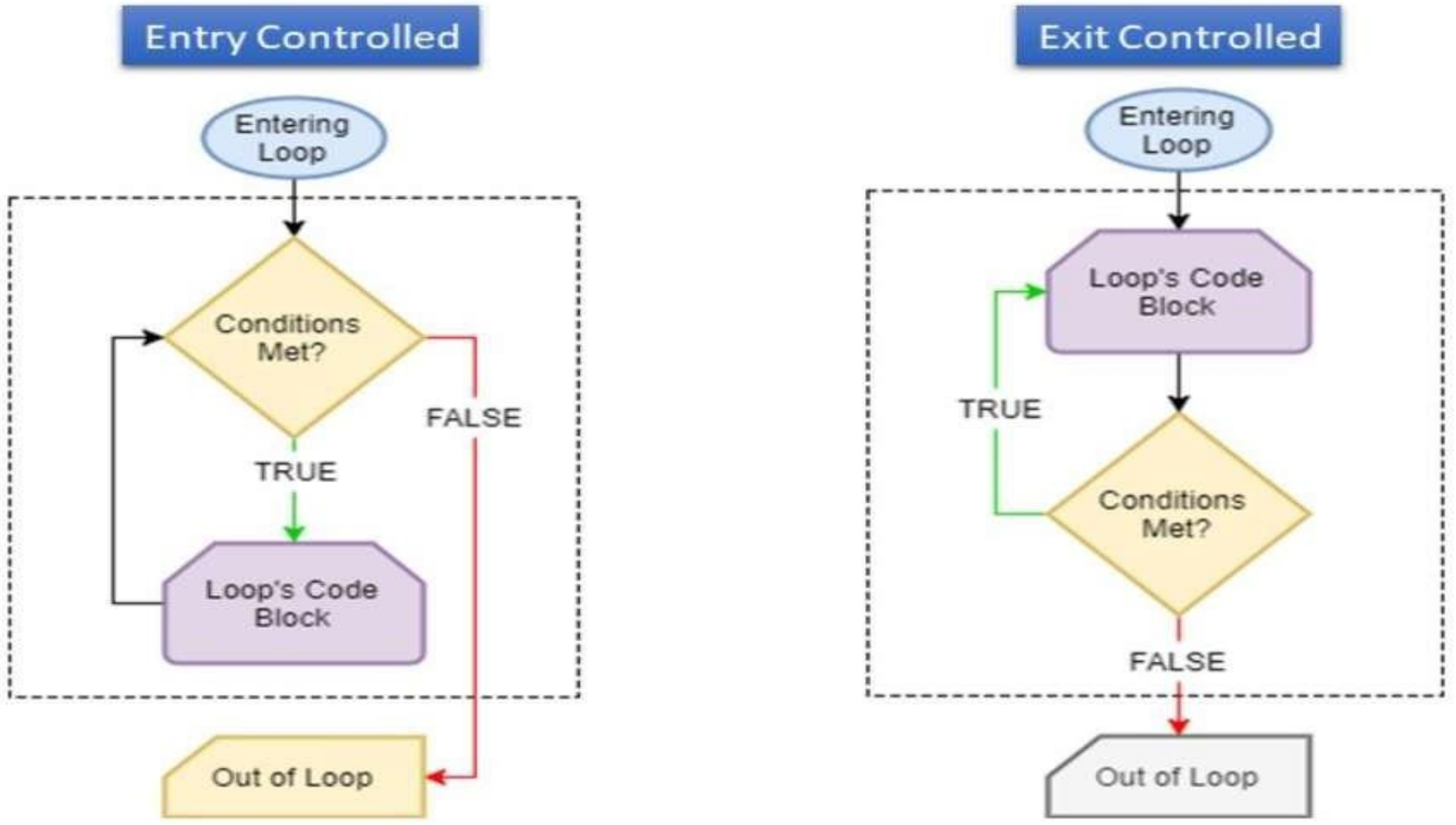
1. Entry controlled loop

2. Exit controlled loop

In an **entry controlled loop,** a condition is checked before executing the body of a loop. It is also called as a pre-checking loop.

In an **exit controlled loop**, a condition is checked after executing the body of a loop. It is also called as a post-checking loop.

C Looping Statements

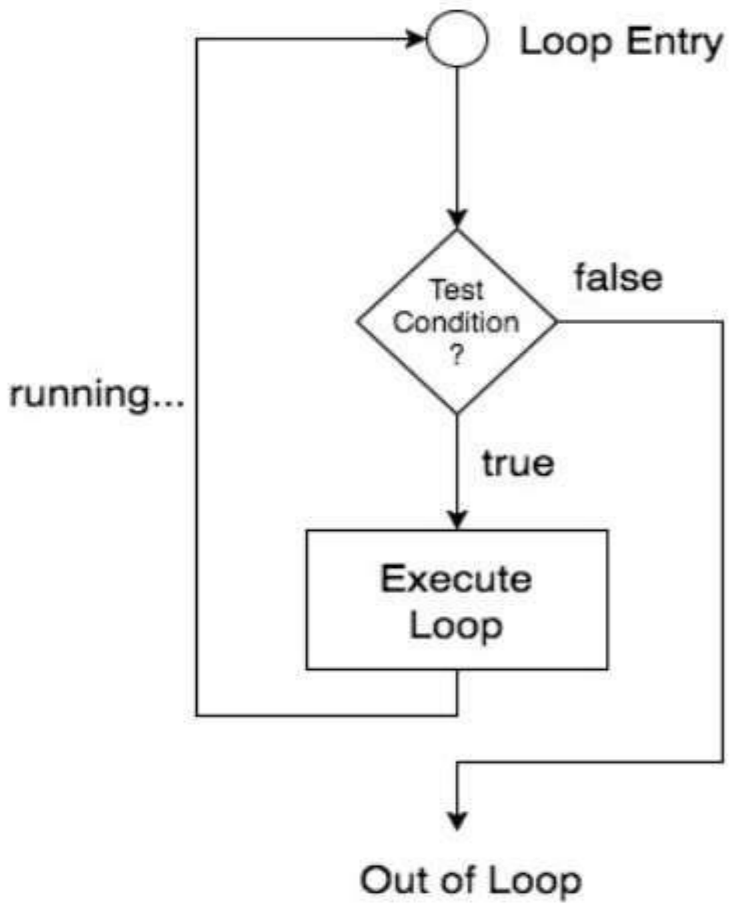


Entry and Exit Controlled Loops

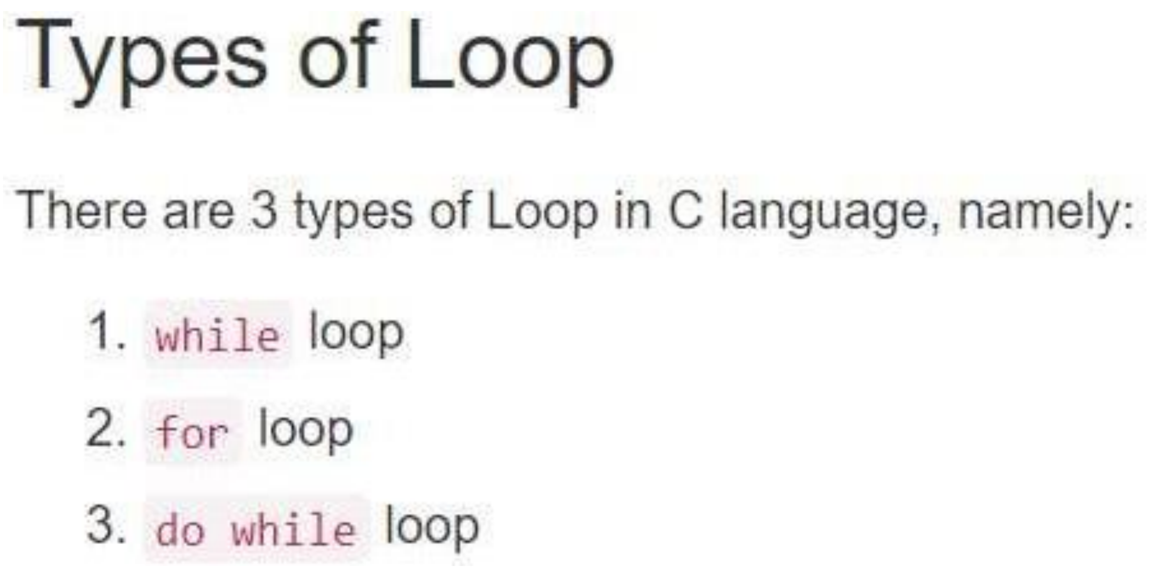
C Looping Statements

How it Works

The below diagram depicts a loop execution,



C Looping Statements

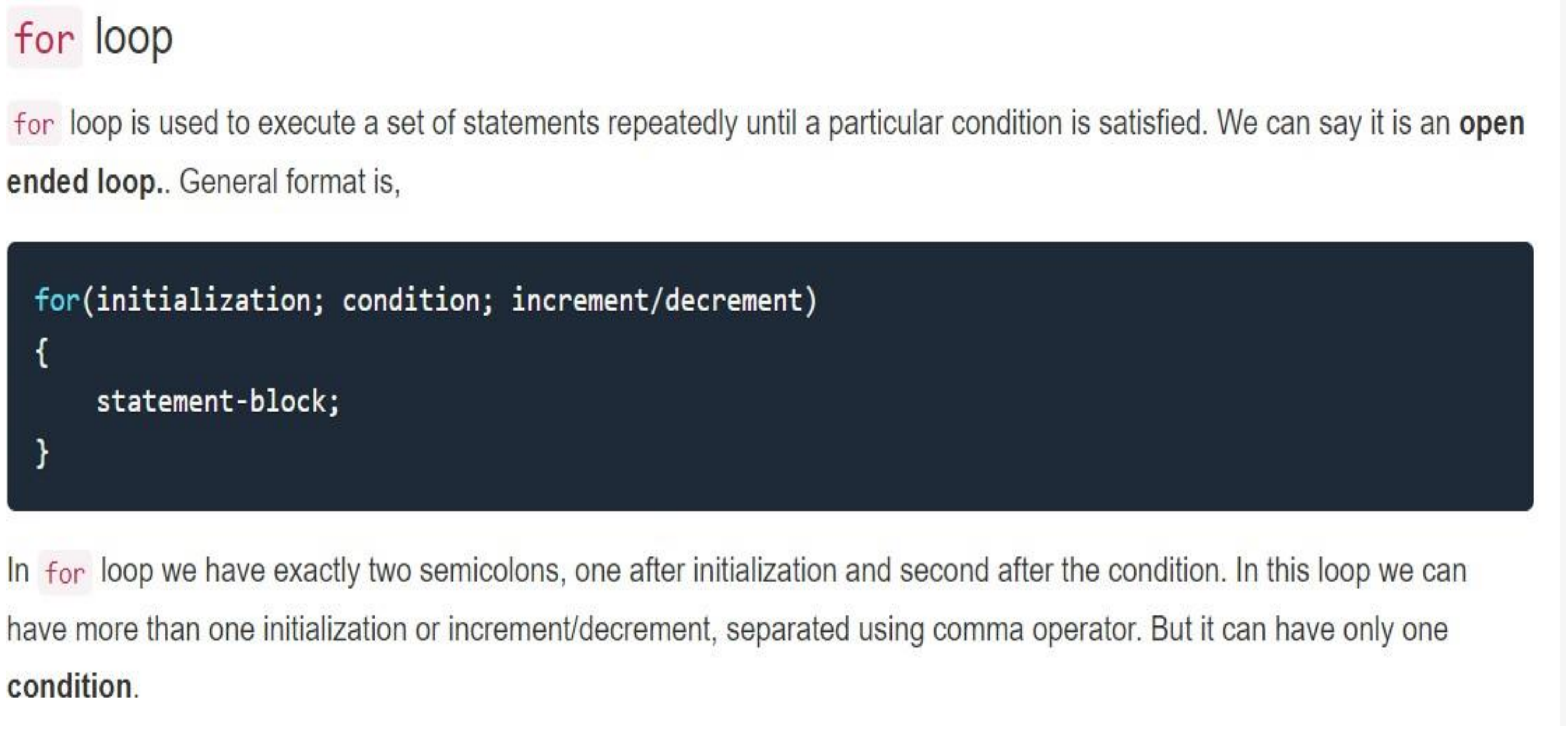


Types of Loop

There are 3 types of Loop in C language, namely:

1. `while` loop
2. `for` loop
3. `do while` loop

for Loop



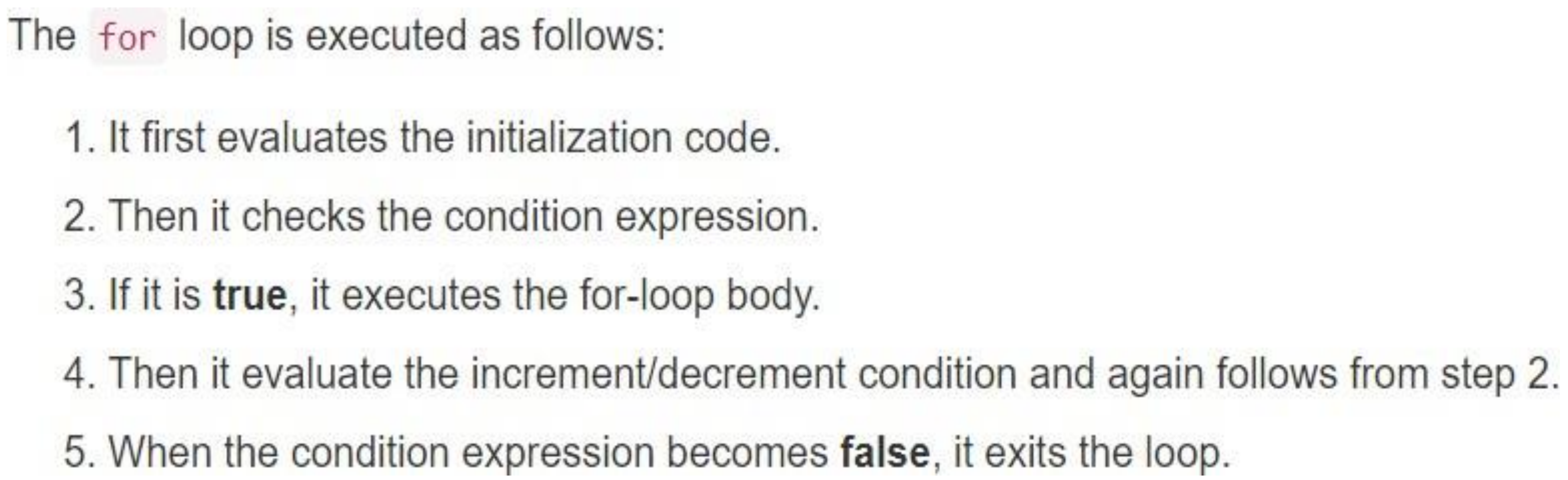
for loop

for loop is used to execute a set of statements repeatedly until a particular condition is satisfied. We can say it is an **open ended loop.**. General format is,

```
for(initialization; condition; increment/decrement)
{
    statement-block;
}
```

In for loop we have exactly two semicolons, one after initialization and second after the condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. But it can have only one **condition**.

for Loop

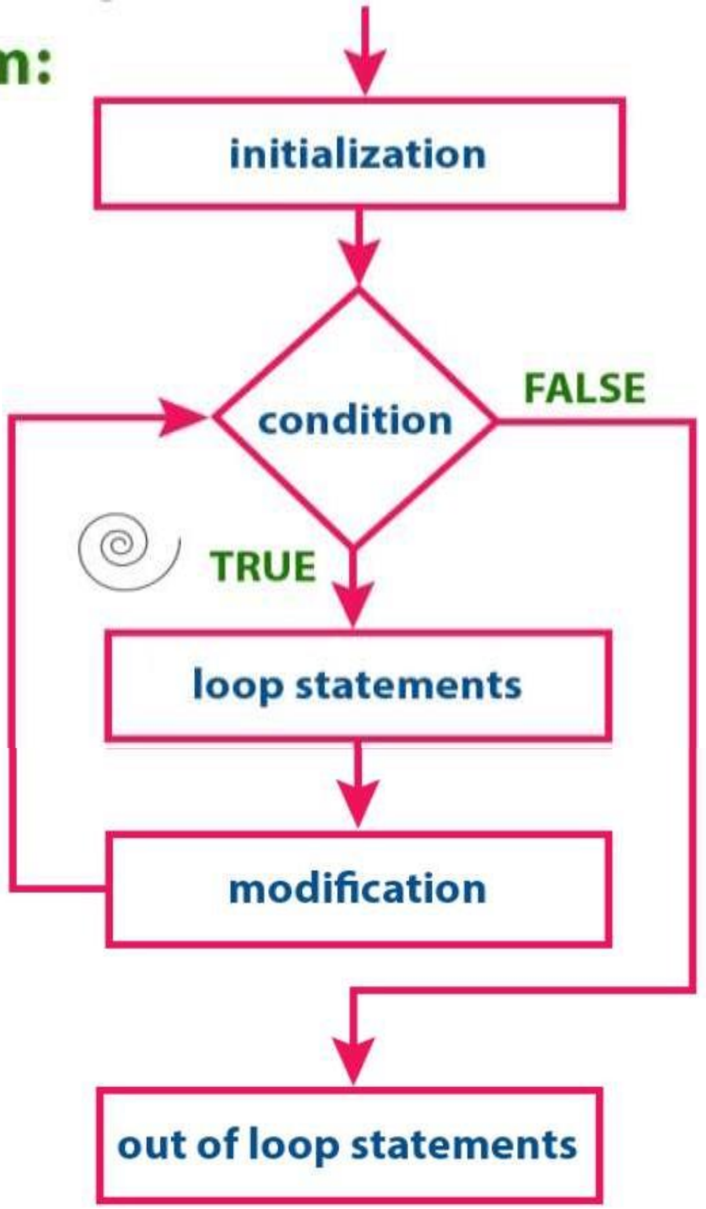


The `for` loop is executed as follows:

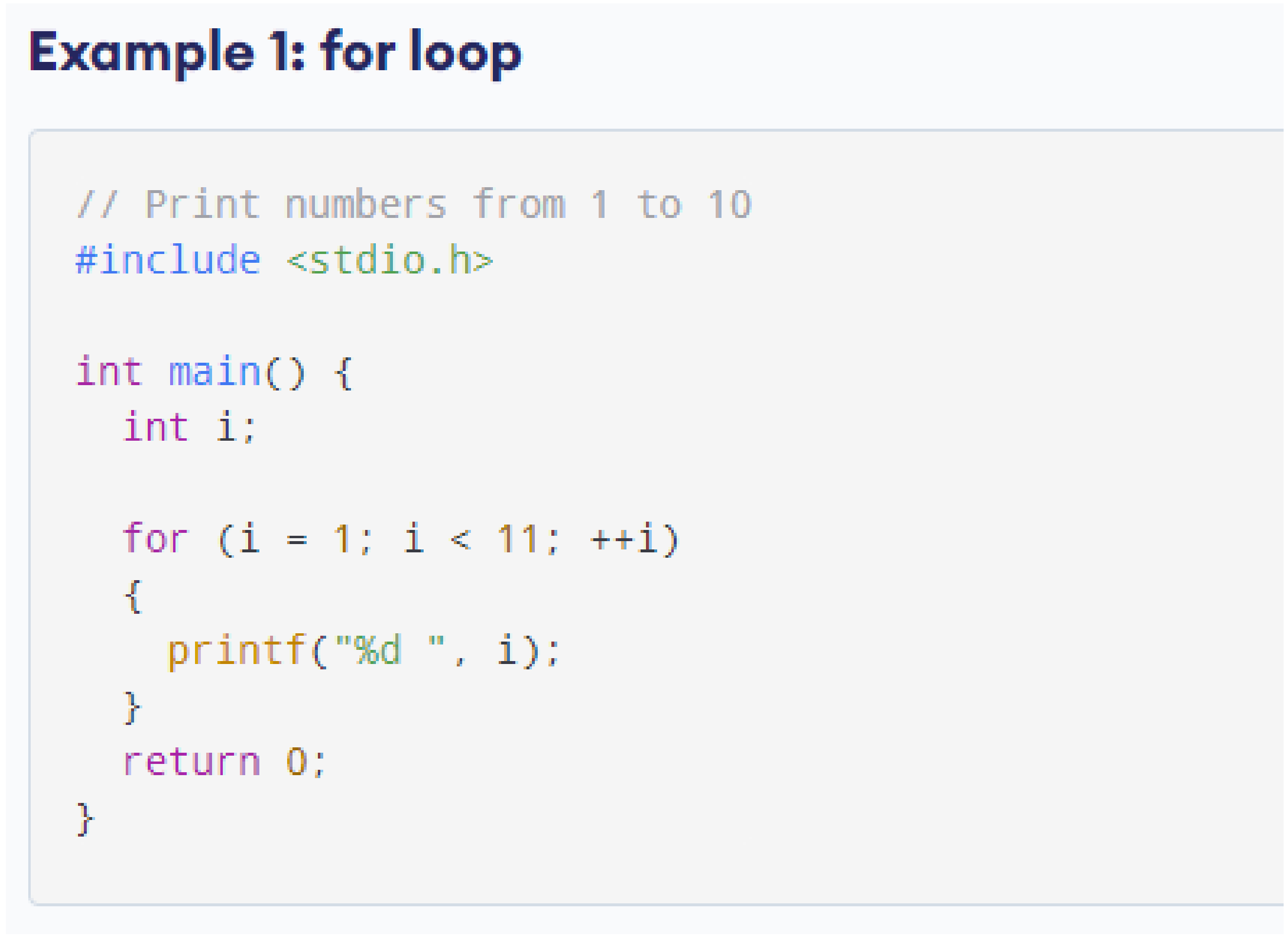
1. It first evaluates the initialization code.
2. Then it checks the condition expression.
3. If it is **true**, it executes the for-loop body.
4. Then it evaluate the increment/decrement condition and again follows from step 2.
5. When the condition expression becomes **false**, it exits the loop.

for Loop

Execution flow diagram:



for Loop



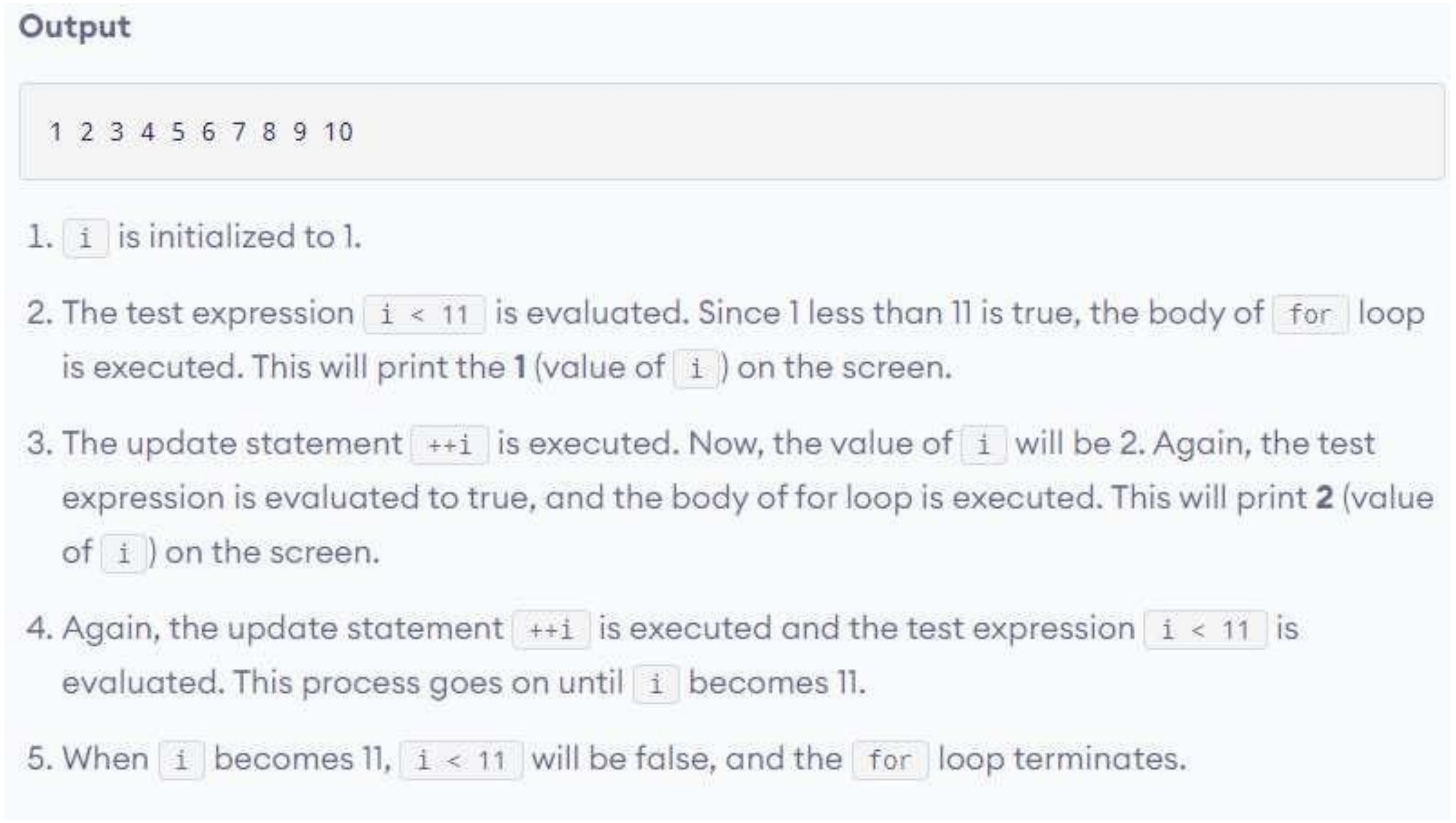
Example 1: for loop

```c
// Print numbers from 1 to 10
#include <stdio.h>

int main() {
  int i;

  for (i = 1; i < 11; ++i)
  {
    printf("%d ", i);
  }
  return 0;
}
```

for Loop

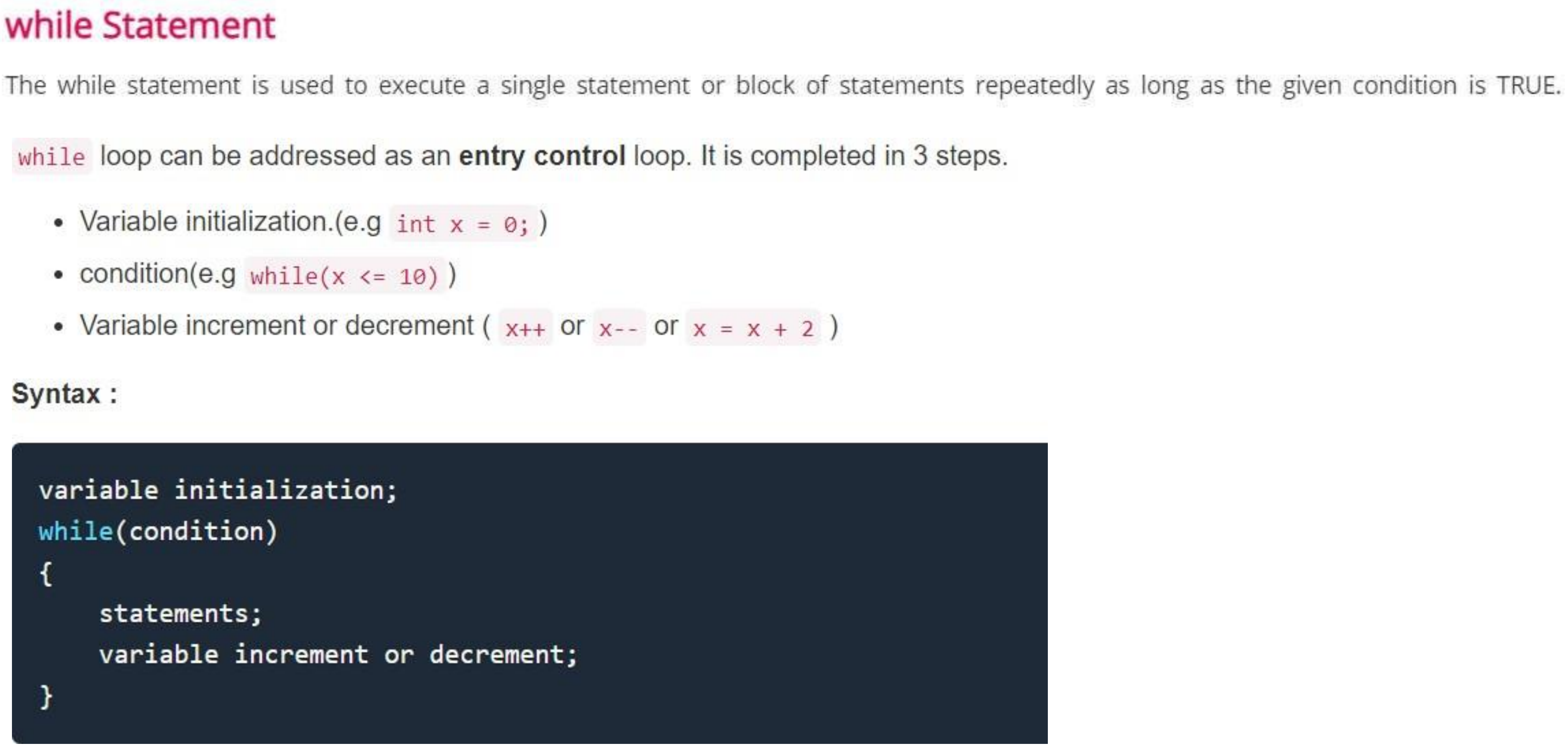


Output

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

1. `i` is initialized to 1.
2. The test expression `i < 11` is evaluated. Since 1 less than 11 is true, the body of `for` loop is executed. This will print the **1** (value of `i`) on the screen.
3. The update statement `++i` is executed. Now, the value of `i` will be 2. Again, the test expression is evaluated to true, and the body of for loop is executed. This will print **2** (value of `i`) on the screen.
4. Again, the update statement `++i` is executed and the test expression `i < 11` is evaluated. This process goes on until `i` becomes 11.
5. When `i` becomes 11, `i < 11` will be false, and the `for` loop terminates.

while Loop



while Statement

The while statement is used to execute a single statement or block of statements repeatedly as long as the given condition is TRUE.

`while` loop can be addressed as an **entry control** loop. It is completed in 3 steps.

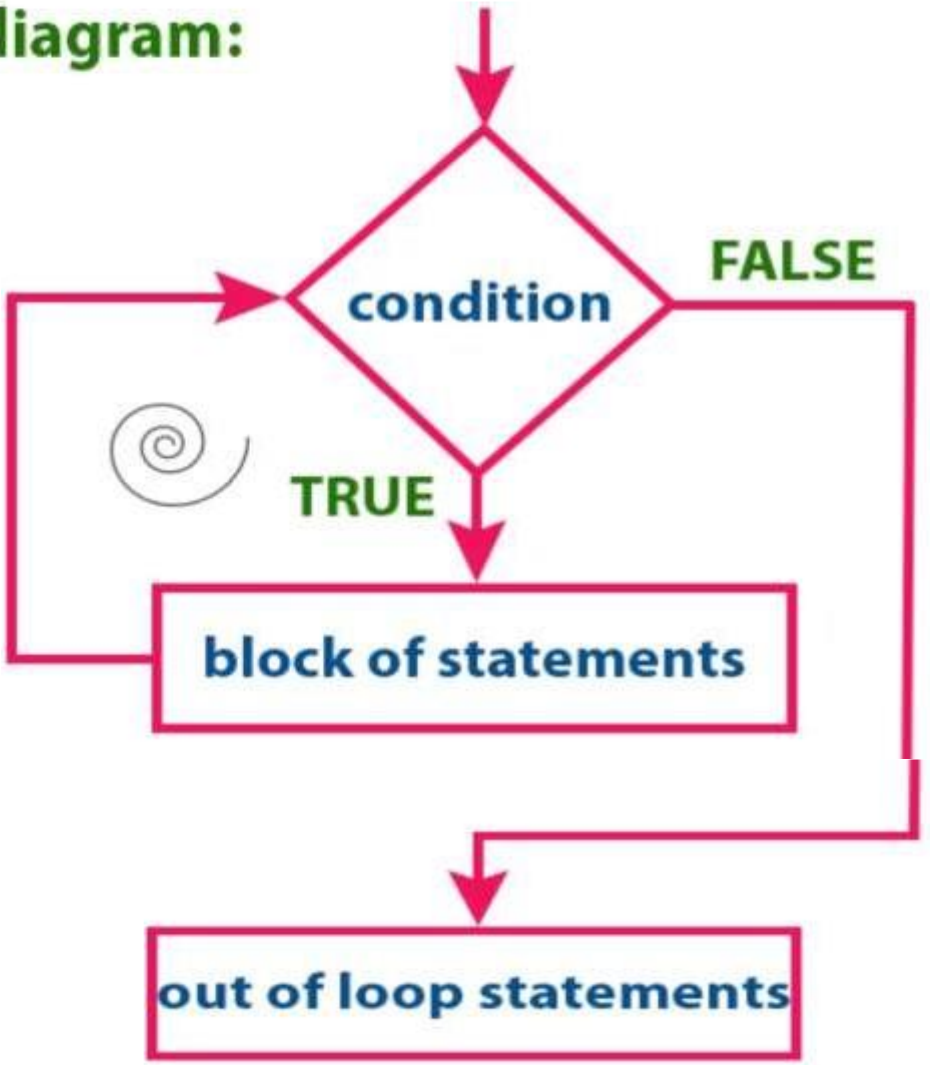
- Variable initialization.(e.g `int x = 0;`)
- condition(e.g `while(x <= 10)`)
- Variable increment or decrement (`x++` or `x--` or `x = x + 2`)

Syntax :

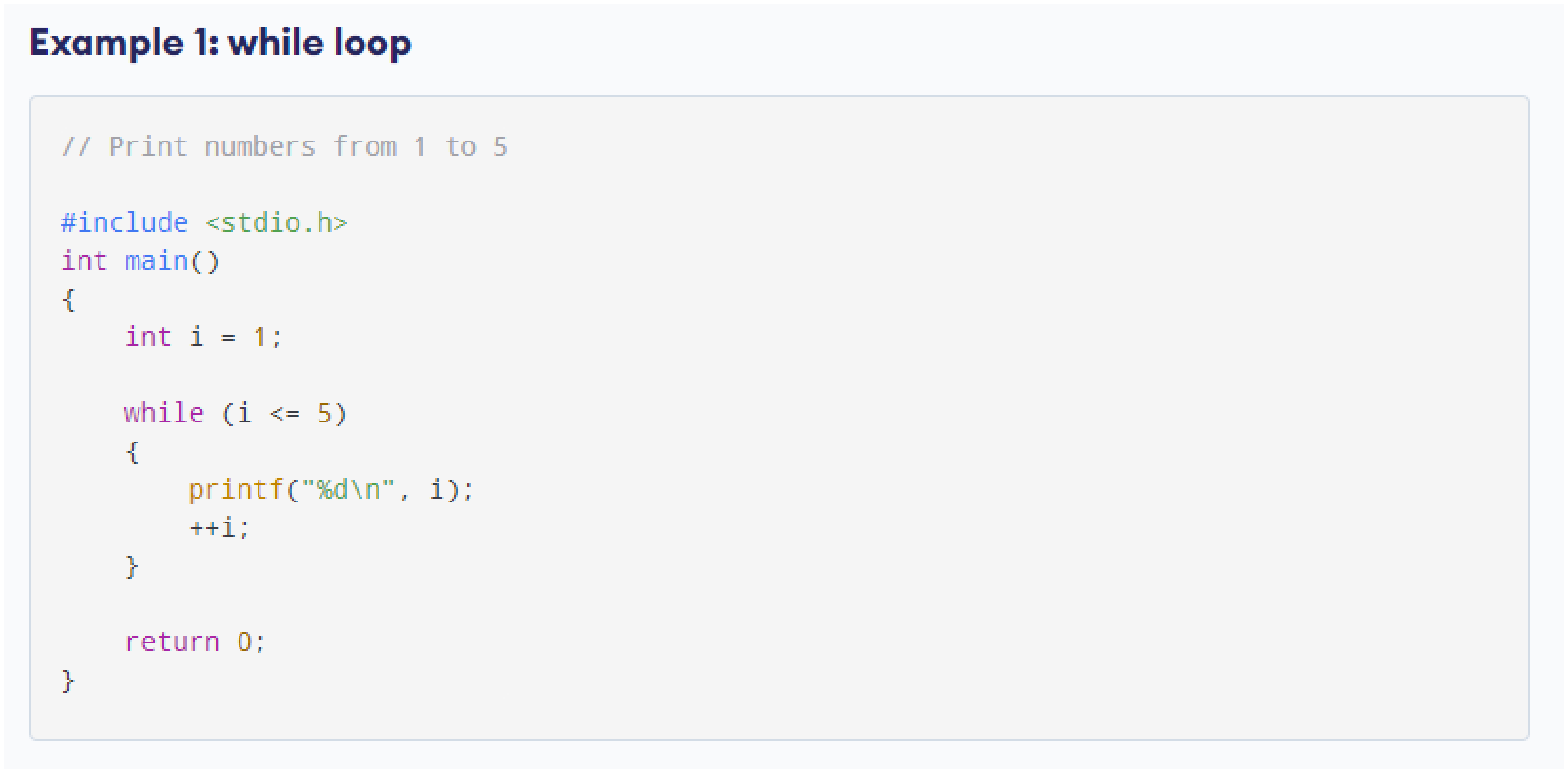
```
variable initialization;
while(condition)
{
    statements;
    variable increment or decrement;
}
```

while Loop

Execution flow diagram:



while Loop



Example 1: while loop

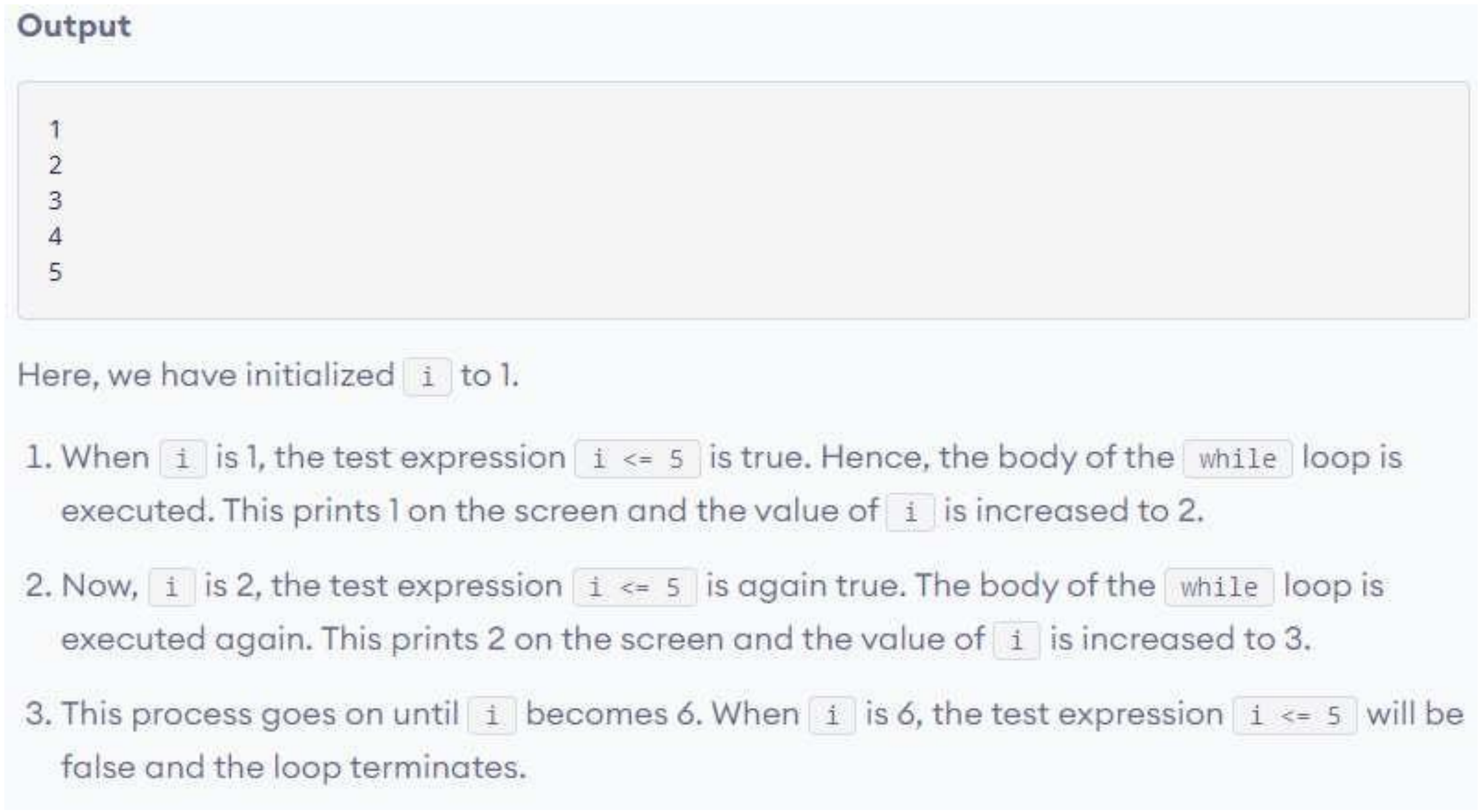
```c
// Print numbers from 1 to 5

#include <stdio.h>
int main()
{
    int i = 1;

    while (i <= 5)
    {
        printf("%d\n", i);
        ++i;
    }

    return 0;
}
```

while Loop



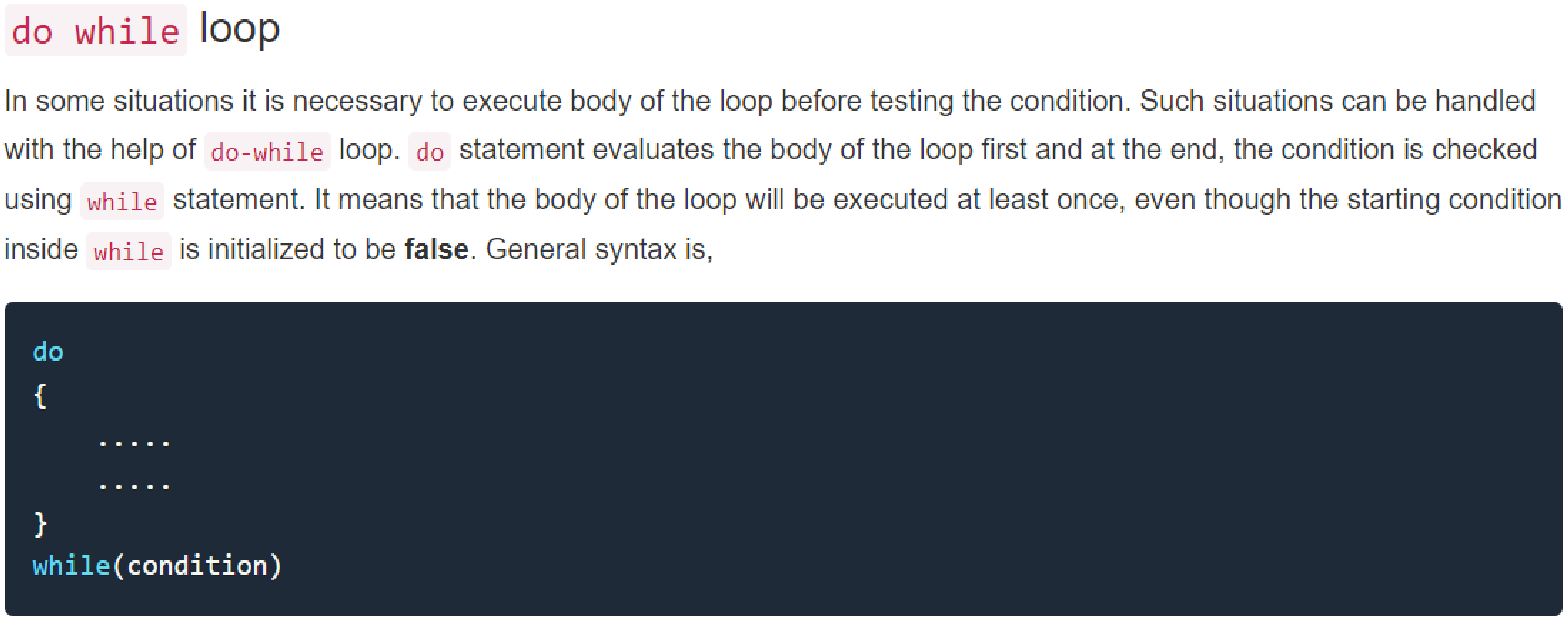
Output

```
1
2
3
4
5
```

Here, we have initialized `i` to 1.

1. When `i` is 1, the test expression `i <= 5` is true. Hence, the body of the `while` loop is executed. This prints 1 on the screen and the value of `i` is increased to 2.
2. Now, `i` is 2, the test expression `i <= 5` is again true. The body of the `while` loop is executed again. This prints 2 on the screen and the value of `i` is increased to 3.
3. This process goes on until `i` becomes 6. When `i` is 6, the test expression `i <= 5` will be false and the loop terminates.

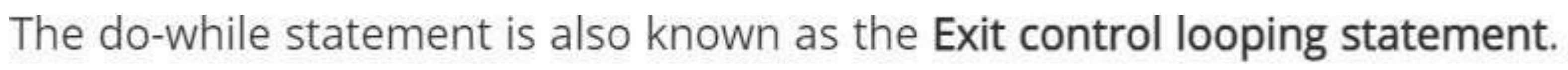
do….while Loop



`do while` loop

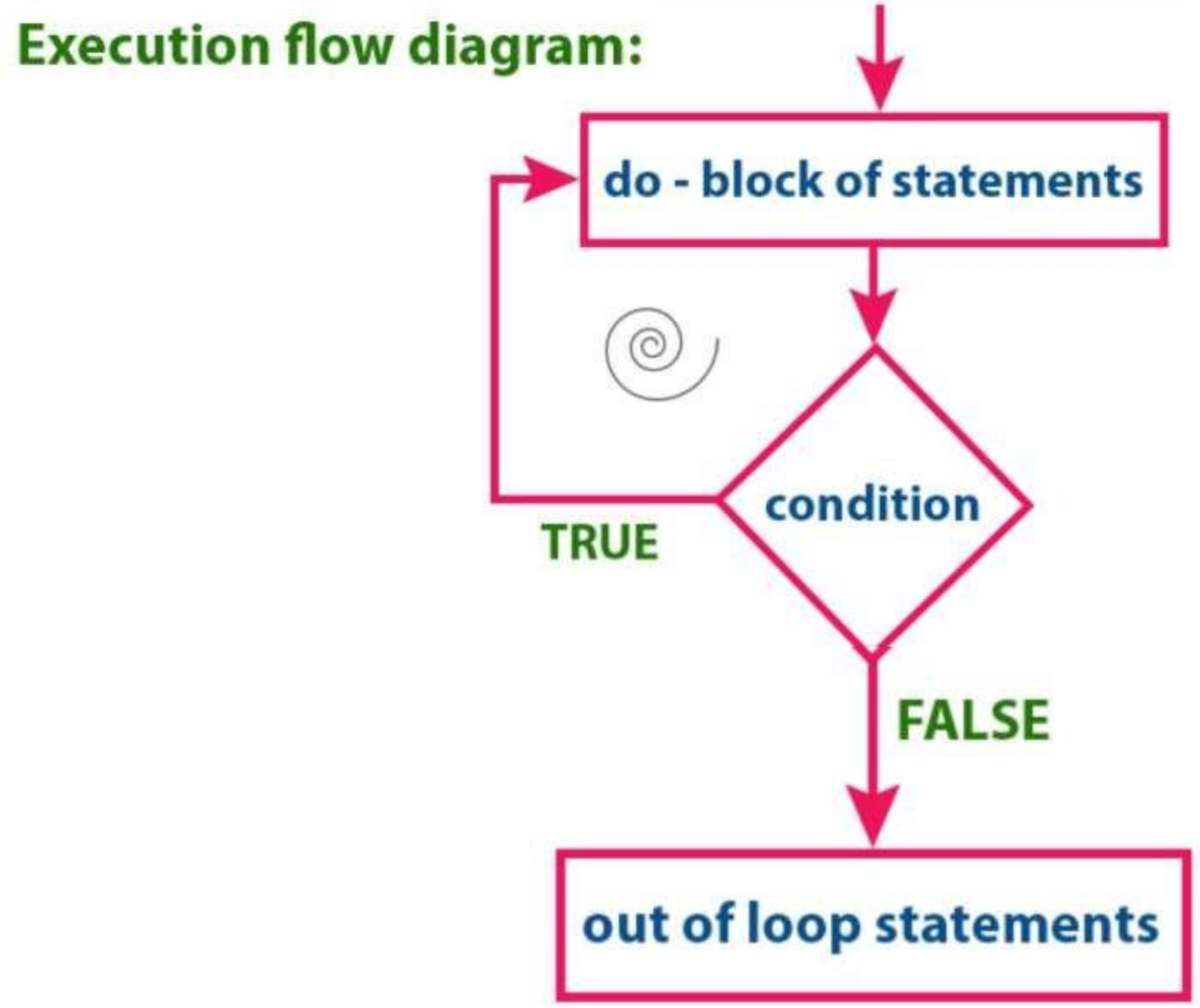
In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of `do-while` loop. `do` statement evaluates the body of the loop first and at the end, the condition is checked using `while` statement. It means that the body of the loop will be executed at least once, even though the starting condition inside `while` is initialized to be **false**. General syntax is,

```
do
{
    .....
    .....
}
while(condition)
```

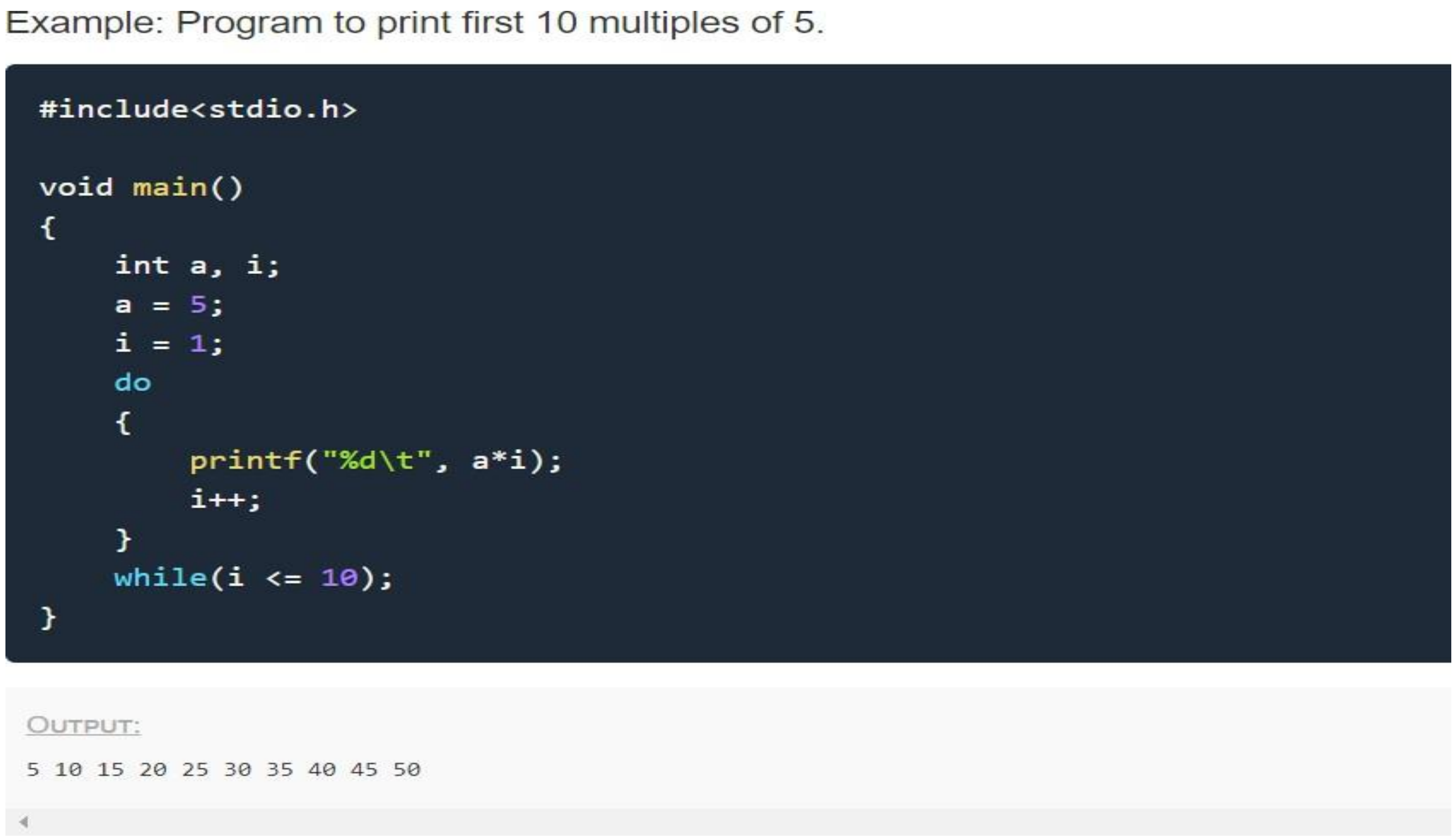


The do-while statement is also known as the **Exit control looping statement.**

do….while Loop



do….while Loop



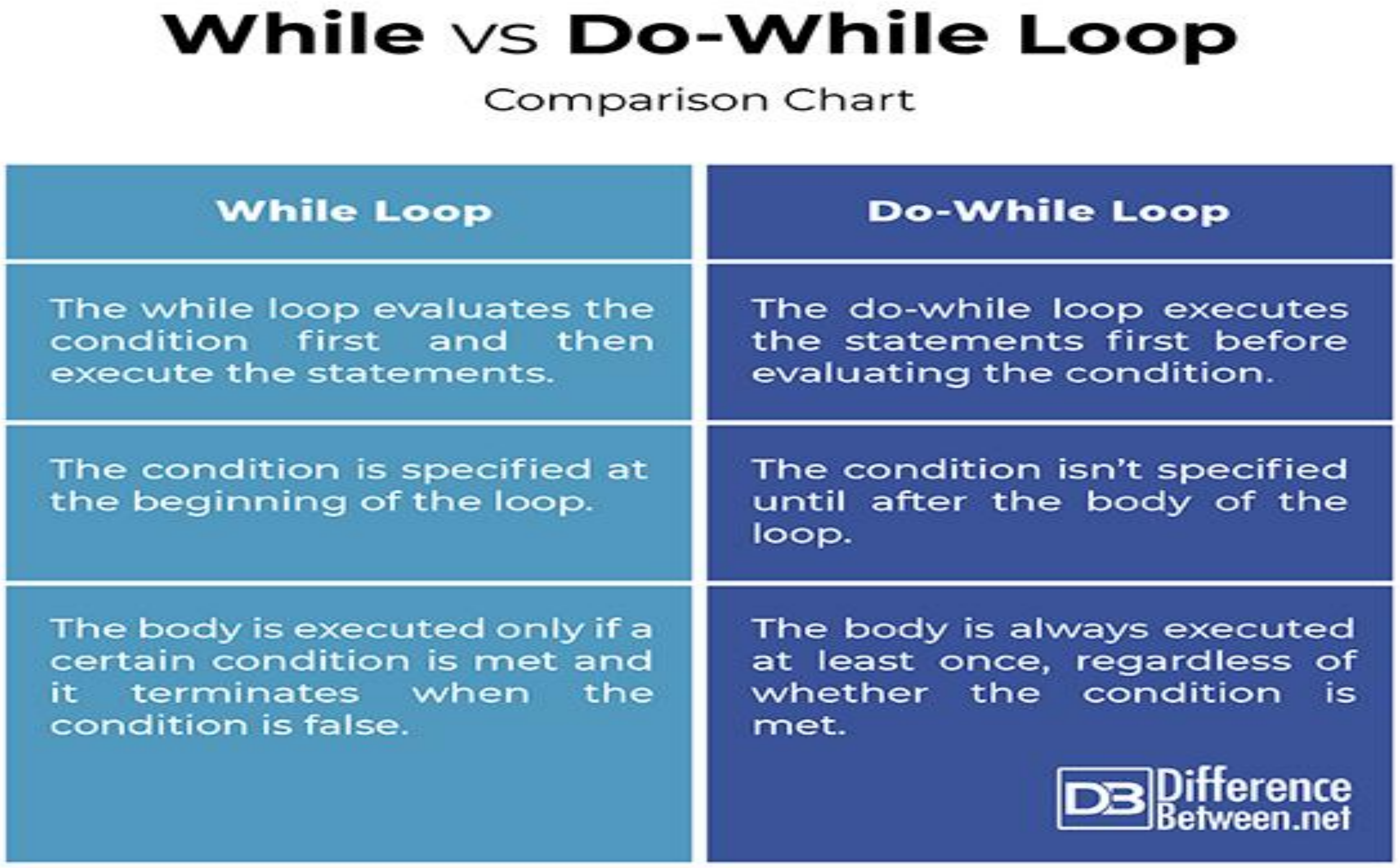
Example: Program to print first 10 multiples of 5.

```c
#include<stdio.h>

void main()
{
    int a, i;
    a = 5;
    i = 1;
    do
    {
        printf("%d\t", a*i);
        i++;
    }
    while(i <= 10);
}
```

OUTPUT:

```
5 10 15 20 25 30 35 40 45 50
```



While vs Do-While Loop

Comparison Chart

While Loop	Do-While Loop
The while loop evaluates the condition first and then execute the statements.	The do-while loop executes the statements first before evaluating the condition.
The condition is specified at the beginning of the loop.	The condition isn't specified until after the body of the loop.
The body is executed only if a certain condition is met and it terminates when the condition is false.	The body is always executed at least once, regardless of whether the condition is met.

DB Difference Between.net

Thank You!