Nested Loop

In this lesson, you will be introduced to the nested loops in C++.

We'll cover the following

Introduction

Suppose you want to print the table of 6, 7, and 8 in a program. First, we have to choose a number whose table we want to print. Then, we will print the table for that number. How can we do this task?

In C++, we can use nested loops to accomplish such tasks.

---

*A loop inside the body of another loop is called a **nested loop**.*

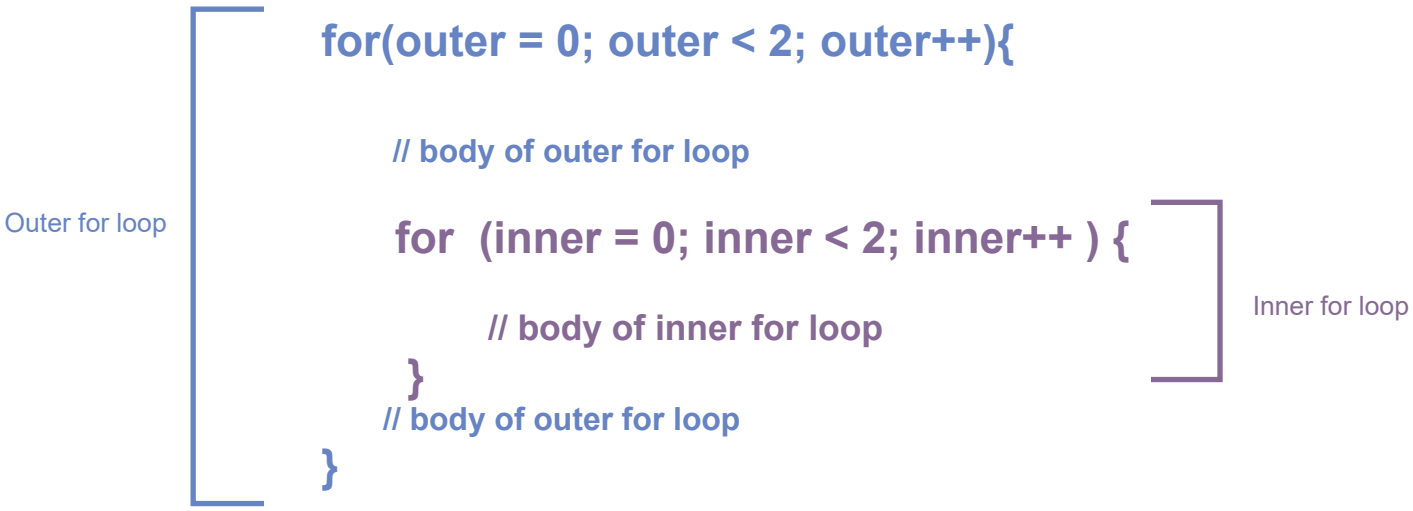
Types

In C++, we have:

- Nested while loop
- Nested do-while loop
- Nested for loop

Syntax

Let's go over the syntax of the nested `for` loop.



In the figure above, we have two `for` loops:

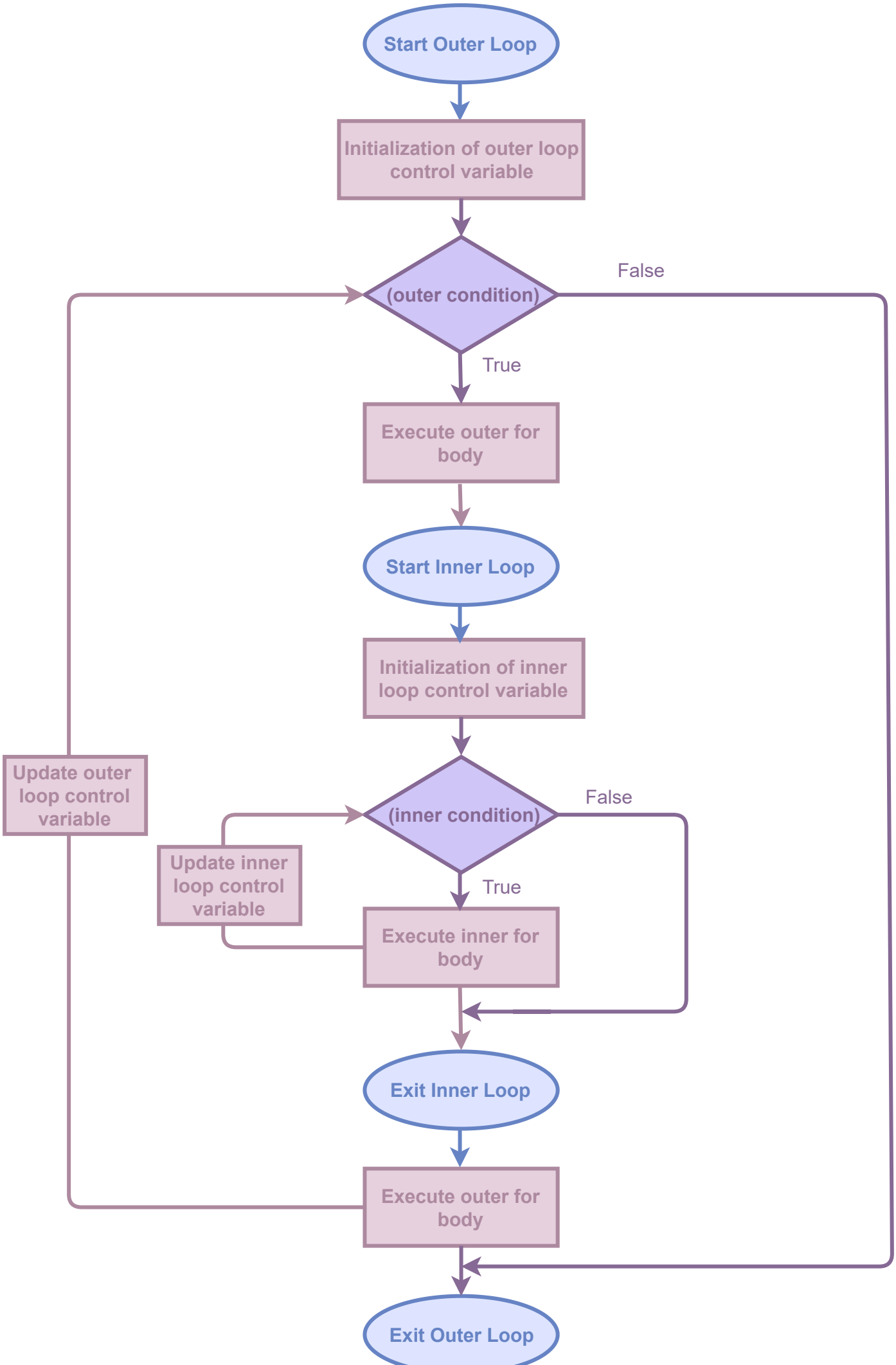
- Outer `for` loop
- Inner `for` loop

The outer `for` loop contains an inner `for` loop inside its body. We can do the same for the `while` and `do-while` loops.

> 📝 We can have multiple loops inside the body of a loop.

Flowchart

Let's look at the flowchart of the `for` loop.


Start Outer Loop
Initialization of outer loop control variable
(outer condition)
False
True
Execute outer for body
Start Inner Loop
Initialization of inner loop control variable
(inner condition)
False
True
Update outer loop control variable
Update inner loop control variable
Execute inner for body
Exit Inner Loop
Execute outer for body
Exit Outer Loop

Example program

Let's translate the example given above into a C++ program.

Press the **RUN** button and see the output!

```
#include <iostream>

using namespace std;

int main() {
  // Declares variable inner and outer
  int inner, outer;
  // Outer for loop
  for (outer = 6; outer <= 8; outer++) {
    // Outer for loop body
    cout << "Table of " << outer << " is:" << endl;
    // Inner for loop
    for (inner = 1; inner <= 5; inner++) {
      // Inner for loop body
      cout << outer << " * " << inner << " = " << (outer * inner) << endl;
    }
    // Exit inner for loop
  }
  // Exit outer for loop
  return 0;
}
```

Explanation

In the nested `for` loop, for the single value of the `outer` variable, the inner loop iterates over all its values. For example, for `outer = 6` the inner loop runs from `inner = 1` to `inner = 5`. After this is done, `outer` is incremented to `7`, and the inner loop iterates over all its values again. This process continues until the value of the `outer` is less than or equal to `8`.

Line No. 7: Declares `inner` and `outer` variables

Line No. 9: Defines an outer `for` loop that takes the values from `6 to 8`

- `outer = 6:` The initial value of the `outer` is set to `6`
- `outer <= 8:` If the loop condition evaluates to true, it executes the statements from **Lines No. 10 to 18**.

- `outer++:` After executing the loop block, it will jump back to **line No. 9**. At this point, it will increment the value of the `outer` by `1`, and again evaluate the condition.

Line No. 11: Prints the value of `outer` to the console.

Line No. 13: Defines an inner `for` loop that takes the values from `1 to 5`

- `inner = 1:` The initial value of the `inner` is set to `1`.
- `inner <= 5:` If the loop condition evaluates to true, it executes the statements from **lines No. 14 to 16**.
- `inner++:` After executing the loop block, it jumps back to **Line No. 13**. At this point, it increments the value of the `inner` by `1` and evaluates the condition again.

Line No 15: Multiplies the value of `outer` by `inner` and display it on the screen

We cannot nest `while` and `do-while` loops.

Retake Quiz

That’s all you needed to know about the workings of nested loops in C++. Let’s discuss the `break` statement in the upcoming lesson.