School Of Mechanical & Manufacturing Engineering,

Department of Mechanical Engineering

# CS-114 - Fundamental of Programing

# Lab Manual # 06

**Lab Instructor:** Muhammad Affan

**Student Name:** ____________________________

**CMS ID** __________________________

**DATE:**

__________________________

School Of Mechanical & Manufacturing Engineering,

Department of Mechanical Engineering

# **LabManual # 6**

## Nested Loops

## Objective

To understand repetition structure and nested for, while and do..while loops in C++.

## Description

A nested loop means a loop statement inside another loop statement. That is why nested loops are also called as “loop inside loops”. The number of loops depends on the complexity of a problem. Suppose, a loop, outer loop, running n number of times consists of another loop inside it, inner loop, running m number of times.

### Syntax for Nested For loop:

Below is the syntax of a Nested for loop:

```
for ( initialization; condition; increment ) {
  for ( initialization; condition; increment ) {
   // statement of inside loop
 }
            // statement of outer loop
}
```

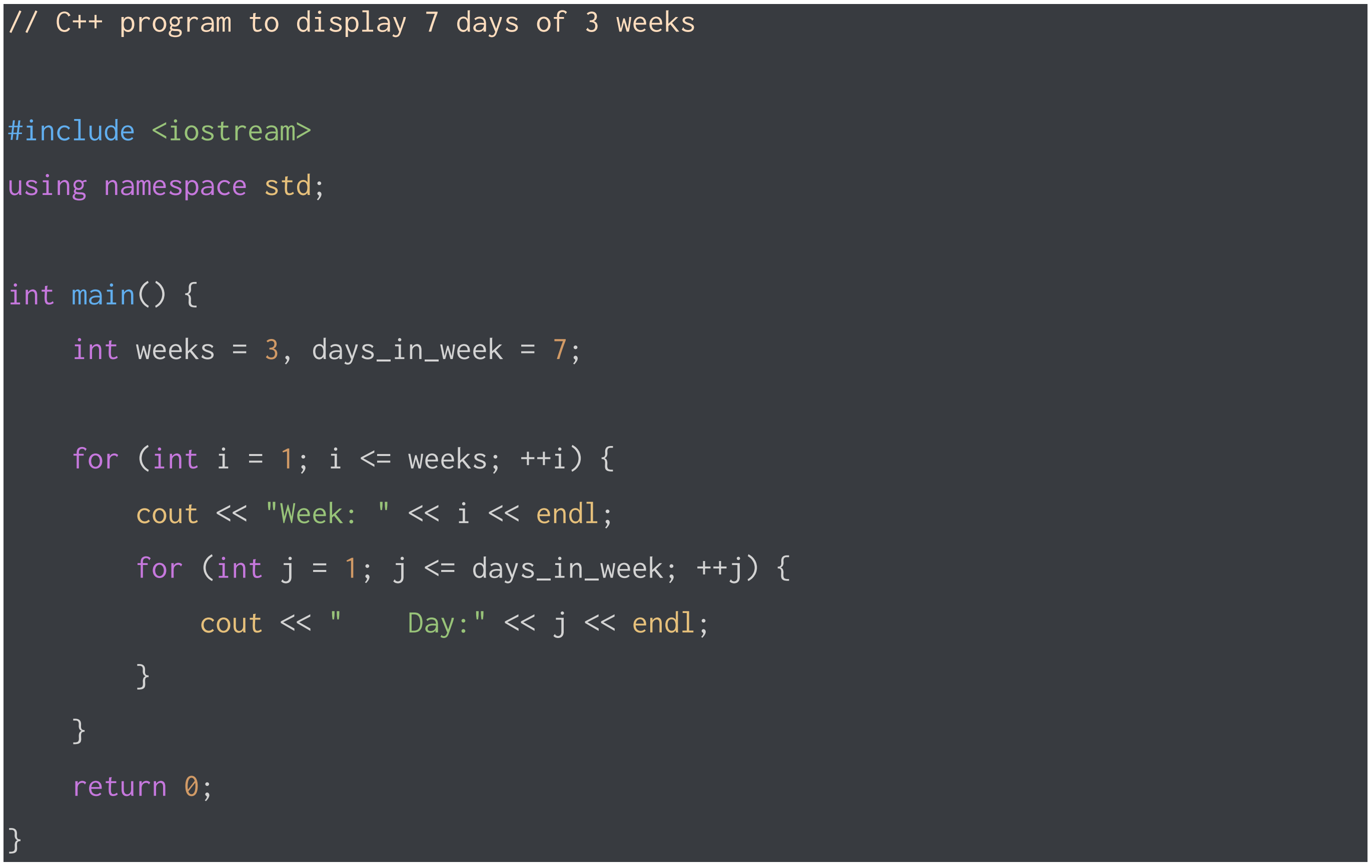
### Example:

A loop within another loop is called a nested loop. Let's take an example,

Suppose we want to loop through each day of a week for 3 weeks.

To achieve this, we can create a loop to iterate three times (3 weeks). And inside the loop, we can create another loop to iterate 7 times (7 days). This is how we can use nested loops.

Example: Nested for Loop



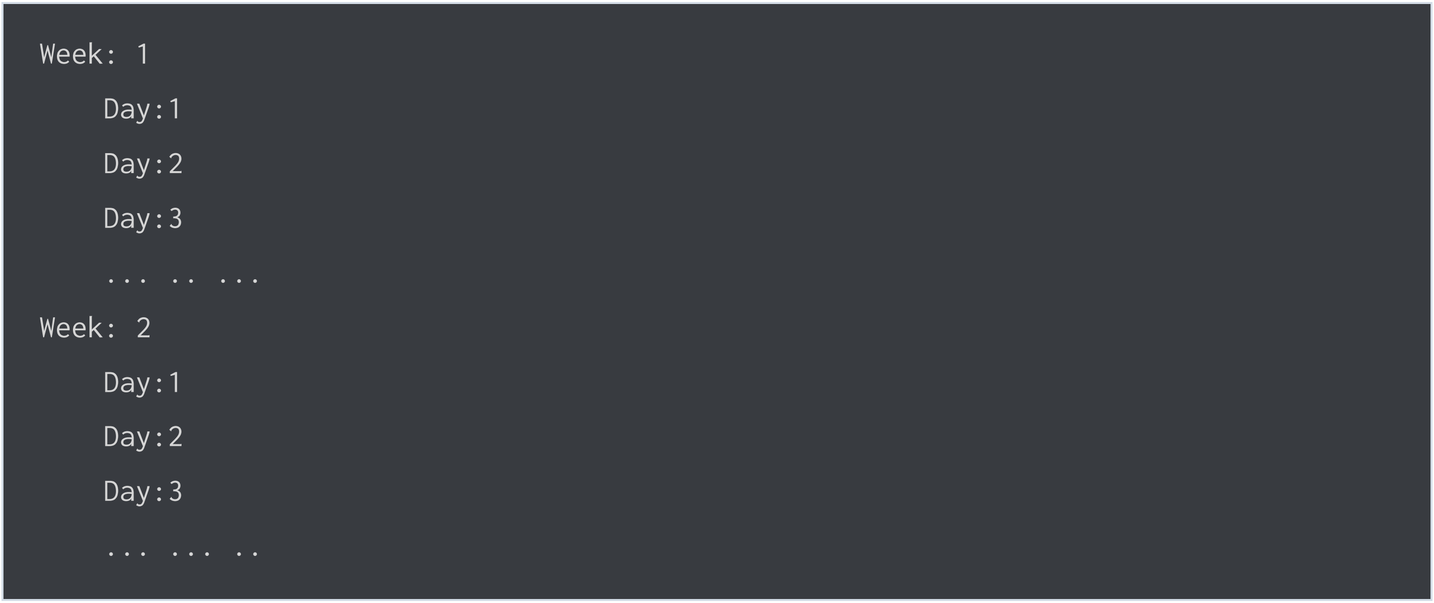
```cpp
// C++ program to display 7 days of 3 weeks

#include <iostream>
using namespace std;

int main() {
    int weeks = 3, days_in_week = 7;

    for (int i = 1; i <= weeks; ++i) {
        cout << "Week: " << i << endl;
        for (int j = 1; j <= days_in_week; ++j) {
            cout << "    Day:" << j << endl;
        }
    }
    return 0;
}
```

## Output



```
Week: 1
    Day:1
    Day:2
    Day:3
    ... .. ...
Week: 2
    Day:1
    Day:2
    Day:3
    ... ... ..
```

## Break statement

Suppose we are writing a program to search for a particular number among 1000 numbers. In the 10th iteration, we have found the desired number. At this point, we don't want to transverse the remaining 990 numbers instead we want the loop to terminate and continue with the execution of the statement following the loop. This is where the break statement comes into play.

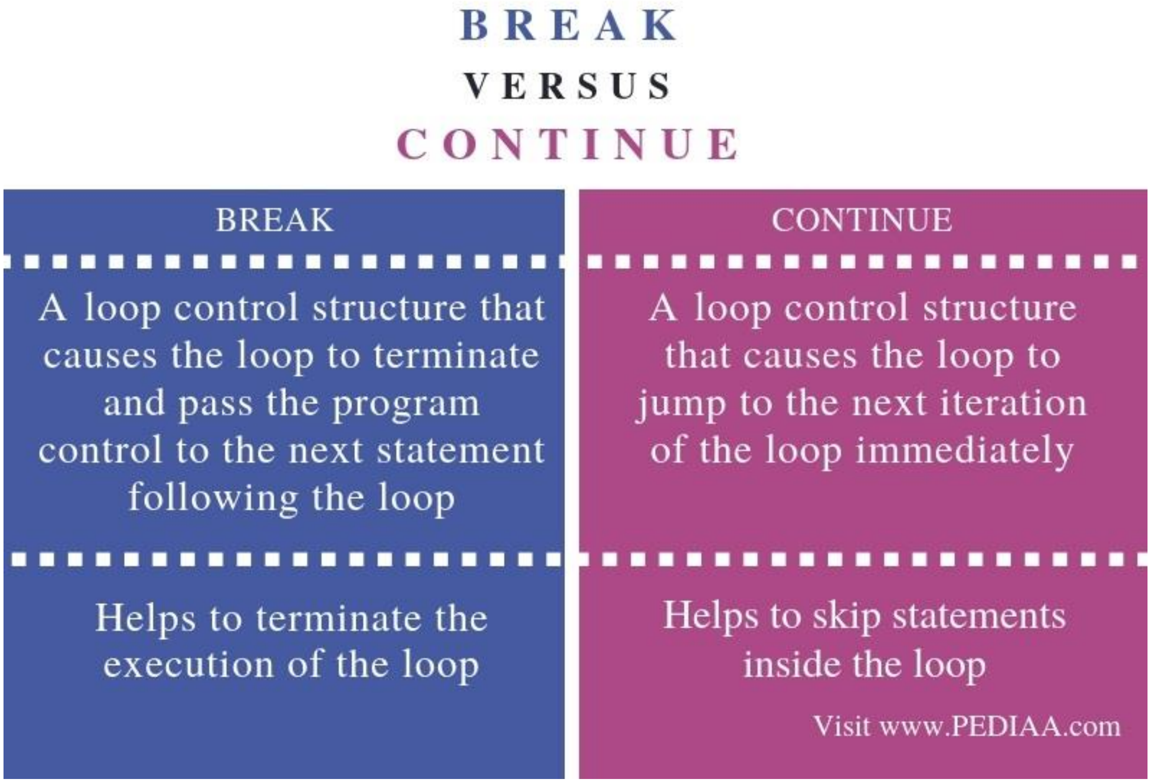
When break statement is encountered within the loop, the program control immediately breaks out of the loop and resumes execution with the statement following the loop. The break statement is commonly used in conjunction with a condition.

## Continue Statement:

The continue statement is used to prematurely end the current iteration and move on the to the next iteration. When the continue statement is encountered in a loop, all the statements after the continue statement are omitted and the loop continues with the next iteration. The continue statement is used in conjunction with a condition.

Sometimes people get confused with between the break and and continue statement. Always remember that the break statement when encountered breaks out of the loop, but when the continue statement is encountered, the loop is not terminated instead the control is passed to the beginning of the loop.

When the continue statement is encountered in the while and do while loop, the control is transferred to the test condition and then the loop continues. whereas in the for loop when continue statement is encountered the control is transferred to the update expression, then the condition is tested.



BREAK VERSUS CONTINUE

| BREAK | CONTINUE |
|---|---|
| A loop control structure that causes the loop to terminate and pass the program control to the next statement following the loop | A loop control structure that causes the loop to jump to the next iteration of the loop immediately |
| Helps to terminate the execution of the loop | Helps to skip statements inside the loop |

Visit www.PEDIAA.com

*Figure 1: Difference between break and continue in C++*

## Lab Task:

1. Generate the Fibonacci sequence using nested loops.
2. Create Pascal's triangle with nested loops.

## Home Task:

1. Write a program using break or continue statement that only adds prime numbers from 1 to 50 and display the sum on screen.

2. Write a program in C++ to create the following pattern.

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

3. Write a C++ program to print:

```
1
2 2
4 4 4 4
6 6 6 6 6 6
```