Lab 3: For Loops

```
#include <iostream>
using namespace std;

int main()
{


// --------------------------- For Loops --------------------------------------

   // The syntax of a basic for loop
   //
   // for ( <initialize> ; <condition> ; <update>)
   // {
   //     // Statements inside the loop
   // }

   // How many times will this run? 4? 5? 6? Try it.

   const int SIZE = 5;
   cout << "Entering FOR Loop" << endl;

   // The variable i can be referred to as the count control variable.
   for(int i = 0; i < SIZE; i++)
   {
      cout << "value of 'i' : " << i << endl;
   }

   return 0;
}
```

For Loops

A **for loop** is similar to an IF statement in that it will only execute a statement(s), if the condition is true. Unlike an if statement, however, a for loop will repeatedly execute the statement(s) inside the loop if the condition continues to be true. What happens inside a **for loop:**

1. Each statement is completed in order,
2. Upon completion of the last statement, the program cycles back to the **for loop statement,**
    a. the count control variable is updated,
    b. the condition is tested again, and if it is still true, step 1 repeats, otherwise the loop breaks

As the name describes, this ends up forming a loop that will repeat until the condition is false.

Basic For Loop to Iterate N Times

```
#include <iostream>
using namespace std;

int main()
{

   int n = 3;

   for(int i = 0; i < n; i++)
   {
      cout << "I will print " << n << " times" << endl;
   }

   return 0;
}
```

Infinite Loops

This is an example of a **for loop** setup to iterate n number of times. This is the most common use of a **for loop** as they are most useful when we know how many times we want our code to loop.

```
#include <iostream>
using namespace std;

int main()
{
   // Entering a number less than 0 will cause an infinite loop. Why?

   int choice;
   cout << "\nEntering a number less than 0 will cause an infinite loop.\n";
   cin >> choice;

   for (int i = 0; i != choice; i++)
   {
      cout << "Infinite Loop" << endl;
   }

   return 0;
}
```

You must be careful when dealing with **for loops**. It is possible to create infinite loops that will never stop. This occurs when the condition is always true. This could happen for multiple reasons.

How For Loop Fields Work

```
#include <iostream>
using namespace std;

int main()
{
   // This is to help demonstrate when each field executes. Note that:
   // 1. The first field only executes once at the very beginning.
   // 2. The third field executes after the body.

   int x = 0;

   for(cout << "First field\n" ; x < 5 ; cout << " Third field\n")
   {
      cout << "  Body of Loop\n";
      x++;
   }

   cout << endl;

   return 0;
}
```

Yes, this will actually run! Go ahead and run it. This code demonstrates when **for loop** fields will execute. The first and third fields will take any valid C++ statement.

Nested For Loops

```
#include <iostream>
using namespace std;

int main()
{
   // For loops can be nested. How many times will the 'cout' execute?

   const int SIZE2 = 3;
   int count_1 = 0;
   cout << "\n Nested For Loops" << endl;

   for (int i = 0; i < SIZE2; i++)
   {
      for (int j = 0; j < SIZE2; j++)
      {
         count_1++;
         cout << "FOR loop 3 count : " << count_1 << endl;
      }
   }

   return 0;
}
```

For loops can be nested. Each time the outer loop completes once, the inner loop will execute. How many times will the "cout" statement execute?

Variable Scope

```
#include <iostream>
using namespace std;

int main()
{
   if (1 == 1)
   {
      int myint = 10;
      cout << "\nmyint exists in this IF statement, but not outside of it.\n";
   }

   // The myint variable is not in this scope and will cause a compiler error.
   myint = 12;

  return 0;
}
```

The scope of a variable defines where a variable exists and where it does not exist. For new programmers, a simple rule you can use is: the scope of a variable is constrained to the area between the set of curly braces it was declared in. Therefore, the scope of a variable declared inside an IF statement would only exist in that IF statement.

Variable Scope In For Loops

```
#include <iostream>
using namespace std;

int main()
{
   // If the cout below the for loop is uncommented you will get an error
   // because 'i's scope is limited to the the for loop it was created in.

   // Could you declare 'count_2' inside a for loop and still get
   // the same output? Why or why not?

   int count_2 = 0;
   int SIZE3 = 3;
   cout << "\nVariable Scope In For Loops\n";

   for (int i = 0; i < SIZE3; i++)
   {
      for (int j = 0; j < SIZE3; j++)
      {
         count_2++;
         cout << "For loop count : " << count_2 << endl;
      }
   }

   //cout << i << endl;

   return 0;
}
```

Advanced For Loop Example

```
#include <iostream>
using namespace std;

int main()
{
   const int SIZE4 = 15;
   int buffer = SIZE4;

   for (int start = 0; start < SIZE4; start++)
   {

      for (int i = 0; i < start; i++)
      {
         cout << " ";
      }

      for (int j = 0; j < buffer - start; j++)
      {
         cout << "*";
      }
      cout << endl;
      buffer--;
   }

   return 0;
}
```

Here is an example of two nested for loops used to create ACSII art. Copy and Paste into your IDE to see the outcome. Can you guess the output?