Chapter 5 - Control Flow - Loops

Why Is Repetition Needed?

Print the numbers 1 through 100

```
cout  << "1" <<  endl  ;
cout  << "2" <<  endl  ;

...
cout  << "100" <<   endl ;
```

- Tedious, subject to errors
- We need a concise way to do statements over and over

Repetition - Loop

- Used when a program needs to repeatedly process one or more instructions until some condition is met, at which time the loop ends.

- The process of performing the same task over and over again is called iteration, and C++ provides built-in iteration functionality. A loop executes the same section of program code over and over again, as long as a loop condition of some sort is met with each iteration.

- This section of code can be a single statement or a block of statements (a compound statement)

- Repetition allows efficient use of variables
- Allows to input, add, and average multiple numbers using a limited number of variables

For example, to add five numbers:

- Declare a variable for each number, input the numbers and add the variables together **or**
- Create a loop that reads a number into a variable and adds it to a variable that contains the sum of the numbers

Types of Repetition Structures

Two types of repetition structures: ***pretest*** *and* ***posttest*** *loops*

Pretest:

1. Loop condition appears at beginning of pretest loop
2. Determines number of times instructions w/in loop body are processed

Types of pretest loop:

1. while
2. for

Posttest:

1. Loop condition appears at end of posttest loop
2. Determines number of times instructions w/in loop body are processed
3. HOWEVER, instructions processed at least once--the first time!

Types of posttest loop:

1. do...while

Counter-Controlled Repetition Requires

1. the **name of a control variable** (or loop counter)
2. the **initial value** of the control variable
3. the **loop-continuation condition** that tests for the final value of the control variable to determine when to exit
4. the control variable to be **incremented** (or **decremented**) each time through the loop

***** ALL LOOPS : if loop body contains more than one statement, statements must be entered as a statement block--that is,**

in a set of braces {}.

Using Shortcut Arithmetic Operators

To increase a variable's value by exactly one :

- prefix ++
  - Used before the variable name
    - ++someValue;
- postfix ++
  - Used after the variable name
    - anotherValue++;

Example :

```
int num = 10;

cout << num++;
//equivalent to:     cout << num;        num = num + 1;

cout << ++num;
//equivalent to:     num = num + 1;      cout << num;
```

```
/*
 * For Loop.cpp
 *
 *  Author: Husain Gholoom
 */
#include<iostream>
using namespace std;
int main()
{
        int v=4;
        int abc= ++v; // before execution, v is incremented by one.
        int xyz= v++; // after execution , v is incremented by one.

        cout<<"++v is "<<abc<<endl;
        cout<<"v++ is "<<xyz<<endl;
        cout<<"v is "<<v<<endl;

       return 0;         }
```

Using a for Loop

- for loop - A special loop that is used when a definite number of loop iterations is required
- **for** is reserved word
- Set of parentheses
- Three sections within parentheses
    - Initializing the loop control variable
    - Testing the loop control variable
    - Updating the loop control variable

for loop syntax:

- `for (initial expr; logical expr; update expr)`

  one statement ;

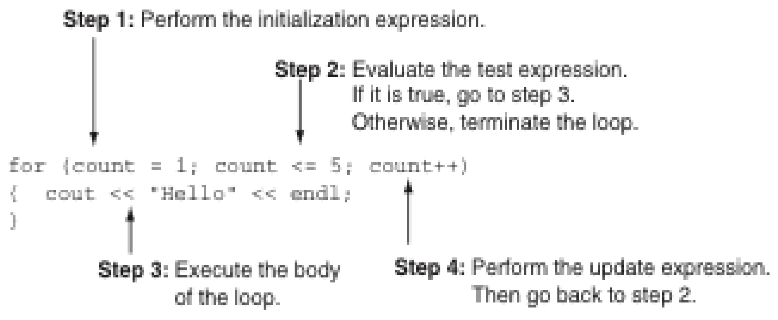
  or

- `for (initial expr; logical expr; update expr)`

```
{       statement;
        statement;
        // Place as many statements
        // here as necessary.        }
```

Here is an example of a simple for loop that prints "Hello" five times:

```
for (int count = 1; count <= 5; count++)
        cout << "Hello" << endl;
```

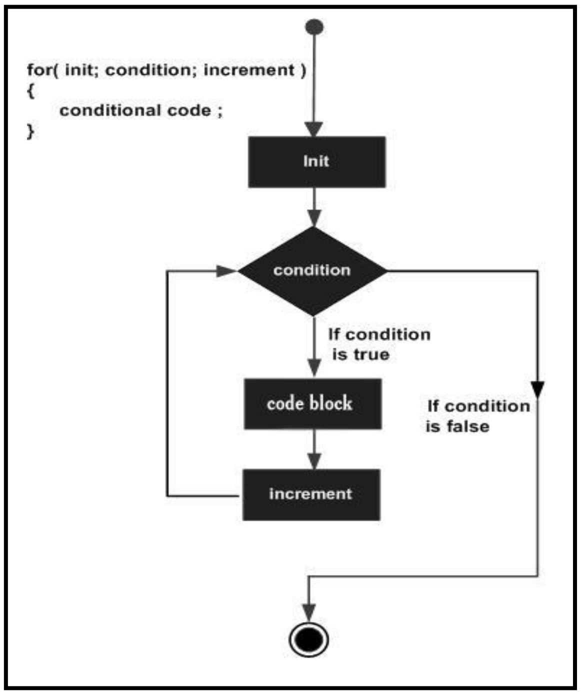

Step 1: Perform the initialization expression.
Step 2: Evaluate the test expression. If it is true, go to step 3. Otherwise, terminate the loop.
for (count = 1; count <= 5; count++)
{ cout << "Hello" << endl;
}
Step 3: Execute the body of the loop.
Step 4: Perform the update expression. Then go back to step 2.

Execution of for loop:

- It must initialize a counter variable (**initial expr**) to a starting value.
- `logical expr is` evaluated. It must test the counter variable by comparing it to a final value.
    - If loop condition evaluates to `true`, execute `statement / s`
    - Execute `update statement.` This is usually done by incrementing or decrementing the loop control variable.

- Repeat until loop condition evaluates to `false`
- Primarily used to implement counter-controlled loops Called <u>counted</u> or <u>indexed</u>
- update expr should change value of loop control variable
- If logical expr omitted, it is assumed true.

Flow Diagram:



Examples :

```
#include<iostream>
using namespace std;
int main()
{
       int i;
         for (  i = 0 ; i<=3 ; i--)  {
              cout<<"  i - >>  "<< i << endl;        }

         cout<<"\nYes : "<<i;

       return 0;

}
```

```
#include<iostream>
using namespace std;
int main()
{
       int i;
         for (  i = 6 ; i>=3 ; --i)  {
              cout<<"  i - >>  "<< i << endl;        }

         cout<<"\nYes : "<<i;

       return 0;

}
```

```
#include<iostream>
using namespace std;
int main()
{
       int i,j;
         for ( i=0 ,j=10; (  i<=10 && j > 5  ) ; i++ , j--)     {
           cout<<"i->>  "<<i<<"  J-->  "<<j<<endl;         }
           cout<<"Yes:"<<i;

       return 0;

}
```

Nested control structures

- Problem:

  A college has a list of test results (1 = pass, 2 = fail) for 10 students. Write a program that analyzes the results. If more than 8 students pass, print "Raise Tuition".

- We can see that

  - The program must process 10 test results. A counter-controlled loop will be used.
  - Two counters can be used : one to count the number of students who passed the exam and one to count the number of students who failed the exam.
  - Each test result is a number : either a 1 or a 2. If the number is not a 1, we assume that it is a 2.

- Top level outline:

  Analyze exam results and decide if tuition should be raised

```
/*
 *  for Statement.cpp
 *  Control structures
 *
 *  Author: Husain Gholoom
 */

#include<iostream>
using namespace std;
int main ()
{
             // initialize variables in declarations
      int    passes = 0,           // number of passes
             failures = 0,         // number of failures
             S_Counter = 1,    // student counter
             result;               // one exam result

             // process 10 students; counter-controlled loop
      for   ( S_Counter = 1; S_Counter <= 10 ; S_Counter ++ ) {
             cout << "Enter result (1=pass,2=fail): ";
             cin >> result;

             if ( result == 1 )        // if else nested in while
                 passes += 1;
             else
                 failures += 1;
      }
             // termination phase
             cout << "Passed " << passes << endl;
             cout << "Failed " << failures << endl;

             if ( passes > 8 )
                 cout << "Raise tuition " << endl;


      return 0;           // indicate the program ended successfully
}
```

Watch out !!!!!!!!!!!

What is output?

```
int x;
for (x=1; x <= 10; x++) {
       cout << "Repeat!" << endl;
       x++;
}

cout << "Done!" << endl;
```

- Do not update the loop variable in the body of a for loop.

What is the output?

Note: no semicolon

```
int x;
for (x = 10; x > 0; x = x-2)
     cout << x << endl;
```

Can define the loop variable inside the for:

```
for (int x = 10; x > 0; x=x-2)
     cout << x << endl;

cout << x << endl; //ERROR, can't use x here
```

Do NOT try to access x outside the loop (the scope of x is the for loop Only)

Non-deterministic count

How many rows are displayed ?

```
int num, Maximum;
  cout << "Enter How Many Times You Want the Number to be Squared" << endl;
  cin>>Maximum;
  cout << "Number Squared" << endl;
      cout << "------ --------------" << endl;
      for (num = 1; num <= Maximum; num++)
           cout << num << "         " << (num * num) << endl;
```

It depends . . .
It's still a count controlled loop, even though the count is not known until run-time.

The Expressions in the ***for*** are <u>**optional**</u>

You may omit any of the three exprs in the for loop header

```
      cout << "Hello!" << endl;
      int value , incr;
      cout << "Enter the starting value: ";
      cin >> value;
      for ( ; value <= 100; )
      {
            cout << "Please enter the increment amount: ";
            cin >> incr;
            value = value + incr;
            cout << value << endl;
      }     // technically it's a count controlled loop, but use a while
```

Can omit all three statements:

```
for (;;)
```

Nested for loops

A nested for loop is a loop in which one for loop resides inside another for loop and the inner for loop gets executed first.

The inner for loop of the nested loop gets declared, initialized and then incremented.

Once all the condition within the inner for loop gets satisfied and becomes true, it moves to the outer loop.

It is often called a "**loop within a loop**".

Example

```
for (int i = 1; i <= 5; i++) {
          for (int j = 1; j <= 10; j++) {
             cout << i * j << " ";
          }
   cout << "\n";
}
```

- All of the statements in the outer loop's body are executed 5 times.
- The inner loop runs 10 times for each of those 5 times, for a total of 50 numbers printed.

What is wrong with the following statement ?

```
int numPrint ;

for (   cin >> numPrint ; numPrint > 0; numPrint--) ;
          cout  << "* ";
          cout  <<endl;
```

What is the output if the user enters 10 ?

```
int numPrint ;

for (   cin >> numPrint ; numPrint > 0; numPrint--)
          cout  << "* ";
          cout  <<endl;
```

What is the output of the following :

```
int x = 0;
for(  ;  x < 10 ; ++x )
      cout << x << "   "  << endl;
```

What is the output of the following :

```
for (int count = 1; count <= 10; count++)

      cout << ++count << " ";     // This is a bad thing to do!
```

What is the output if the user enters 10 ?

```
cout << "Enter A number     " ;
int number;
cin >> number ;
for (  int i = 0 ; i < number ; )
{
     i = i + 2;
     if ( i <= number )
          cout << i << "  "  ;
}
```

What is the output of the following :

```
const int SIZE = 5;

for (int r = 1; r <= SIZE; r++) {
     for (int c = 1; c <= SIZE; c++) {
        cout << "*";
     }
     cout << endl;
 }
```

What is the output of the following :

```
const int MAXCOUNT = 50;

for (int i = 1; i <= MAXCOUNT; i++)

   if (  (i % 5  ) )
      cout << "The number "<< i << endl;
```

Using a for loop , write a C++ program that will print the following pattern

```
*******
******
*****
****
***
**
*
```