# Lecture 6: Control Statements – Part III

Class page: https://github.com/tsung-wei-huang/cs1410-40

Dr. Tsung-Wei Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

# Learning Objectives

- The essentials of counter-controlled repetition
- Use for and do…while to execute statements
- How to break and continue within the control flow

# Counter-based Control Flow

- **Counter-controlled repetition requires**
  - the name of a control variable (or loop counter)
  - the initial value of the control variable
  - the loop-continuation condition that tests for the final value of the control variable (i.e., whether looping should continue)
  - the increment (or decrement) by which the control variable is modified each time through the loop.

# Revisit the Counter-based While Loop

```
1
2  // Counter-controlled repetition.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int counter = 1; // declare and initialize control variable
9
10    while ( counter <= 10 ) // loop-continuation condition
11    {
12       cout << counter << " ";
13       counter++; // increment control variable by 1
14    } // end while
15
16    cout << endl; // output a newline
17 } // end main
```

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

# for Repetition Statement

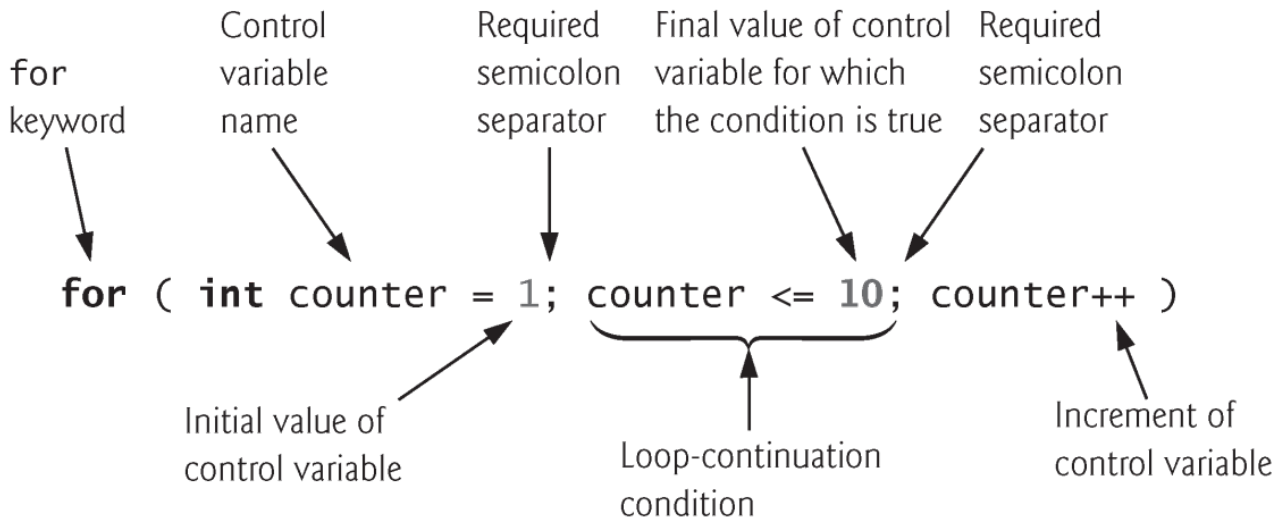
- The for repetition statement specifies the counter-controlled repetition details in a single line of code.
- The initialization occurs once when the loop is encountered.
- The condition is tested next and each time the body completes.
- The body executes if the condition is true.
- The increment occurs after the body executes.
- Then, the condition is tested again.
- If there is more than one statement in the body of the `for`, braces are required to enclose the body of the loop.

# for Repetition Statement Example

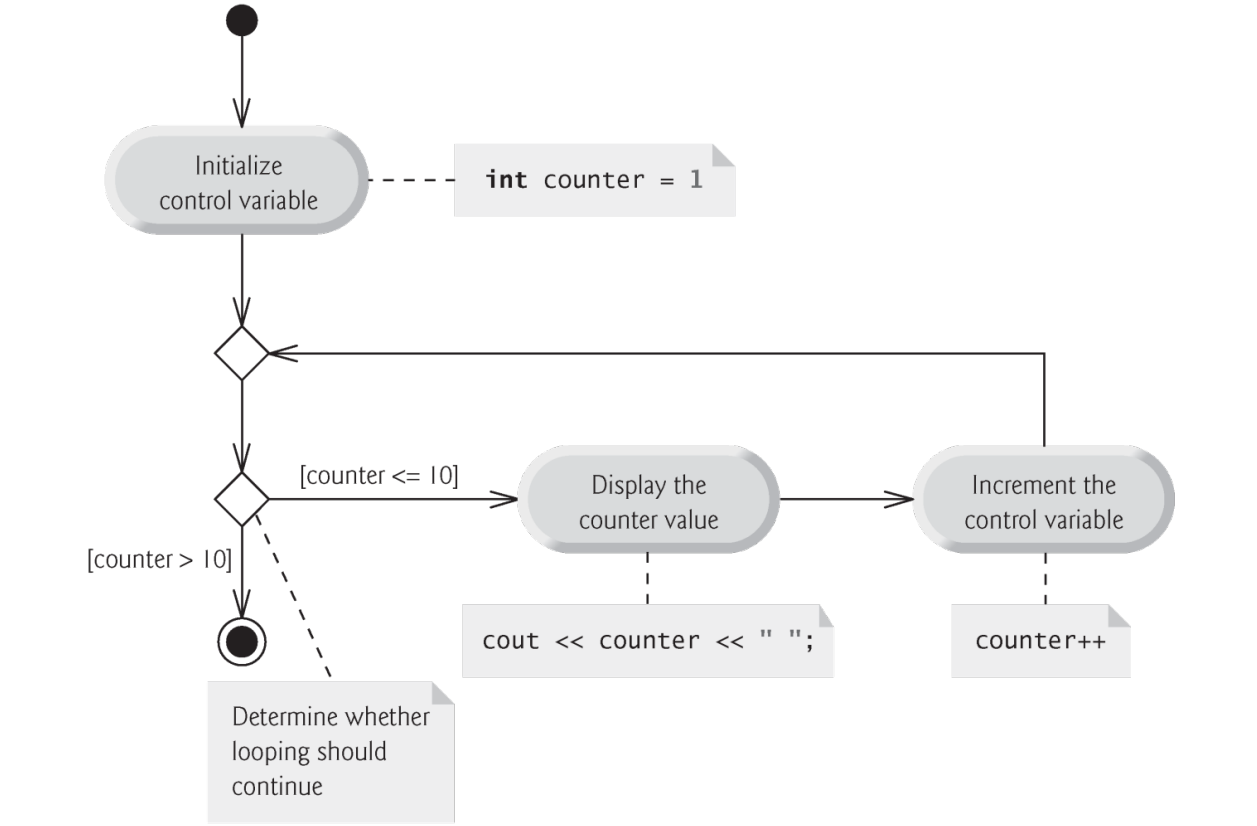
```
1
2  // Counter-controlled repetition with the for statement.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     // for statement header includes initialization,
9     // loop-continuation condition and increment.
10    for ( int counter = 1; counter <= 10; counter++ )
11       cout << counter << " ";
12
13    cout << endl; // output a newline
14 } // end main
```

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

# for Repetition Statement



# for Repetition Statement



# for vs while

- **The general form of the `for` statement is**
  - `for` ( *initialization*; *loopContinuationCondition*; *increment* )
    *statement*
- **where the *initialization* expression initializes the loop's control variable, *loopContinuationCondition* determines whether the loop should continue executing and *increment* increments the control variable.**
- **In most cases, the `for` statement can be represented by an equivalent `while` statement, as follows:**
  - *initialization*;

```
while ( loopContinuationCondition )
{
    statement
    increment;
}
```

# Example: Rewrite while to for

```
int a = 2;

while ( a < 10000 )
{
    a += (a % 2) ? 1 : 2;
}
```

# Increment a Counter

- **The initialization, loop-continuation condition and increment expressions of a `for` statement can contain arithmetic expressions.**
- **The expressions**

```
counter = counter + 1
counter += 1
++counter
counter++
```

  **are all equivalent in the incrementing portion of the `for` statement's header (when no other code appears there).**
- **The "increment" of a `for` statement can be negative, in which case the loop actually counts downward.**
- **If the loop-continuation condition is initially false, the body of the `for` statement is not performed.**

# for Repetition Statement Pitfall!

```
for ( counter=1; counter<=10; counter++ ) ;
```

→ Do nothing for 10 times

```
for ( counter=1; counter<=10; counter++ ) {

  → Enclose your for loop statement with {}

}
```

# Examples using for Repetition Statement

- Vary the control variable from 1 to 100 in increments of 1.
  - `for ( int i = 1; i <= 100; i++ )`
- Vary the control variable from 100 down to 1 in decrements of 1.
  - `for ( int i = 100; i >= 1; i-- )`
- Vary the control variable from 7 to 77 in steps of 7.
  - `for ( int i = 7; i <= 77; i += 7 )`
- Vary the control variable from 20 down to 2 in steps of -2.
  - `for ( int i = 20; i >= 2; i -= 2 )`
- Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17.
  - `for ( int i = 2; i <= 17; i += 3 )`
- Vary the control variable over the following sequence of values: 99, 88, 77, 66, 55.
  - `for ( int i = 99; i >= 55; i -= 11 )`

# Example

- **What is the difference between the two statements?**

```
for ( int c=10; c>=0; c-- ) ;
```

```
for ( unsigned c=10; c>=0; c-- ) ;
```

# Example using for statement

- **A person invests $1000.00 in a savings account yielding 5 percent interest.**
- **Use the following formula to calculate and print the amount of money in the account at the end of each year for 10 years:**

  $a = p\,(1 + r)^n$

  ***where***

  ***p* is the original amount invested (i.e., the principal),**
  ***r* is the annual interest rate,**
  ***n* is the number of years and**
  ***a* is the amount on deposit at the end of the *n*th year.**
- **This problem involves a loop that performs the indicated calculation for each of the 10 years.**

# Example using for statement

```
1
2   // Compound interest calculations with for.
3   #include <iostream>
4   #include <iomanip>
5   #include <cmath> // standard C++ math library
6   using namespace std;
7
8   int main()
9   {
10     double amount; // amount on deposit at end of each year
11     double principal = 1000.0; // initial amount before interest
12     double rate = .05; // interest rate
13
14     // display headers
15     cout << "Year" << setw( 21 ) << "Amount on deposit" << endl;
16
17     // set floating-point number format
18     cout << fixed << setprecision( 2 );
19
```

# Example using for statement

```
20       // calculate amount on deposit for each of ten years
21       for ( int year = 1; year <= 10; year++ )
22       {
23          // calculate new amount for specified year
24          amount = principal * pow( 1.0 + rate, year );
25
26          // display the year and the amount
27          cout << setw( 4 ) << year << setw( 21 ) << amount << endl;
28       } // end for
29    } // end main
```

```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

# Example using for statement

- **C++ does not include an exponentiation operator, so we use the standard library function pow.**
  - `pow(x, y)` calculates the value of `x` raised to the $y^{th}$ power.
  - Takes two arguments of type `double` and returns a `double` value.
- **This program will not compile without including header file `<cmath>`.**
  - Includes information that tells the compiler to convert the value of `year` to a temporary `double` representation before calling the function.
  - Contained in `pow`'s function prototype.

# Example using for statement

- **The stream manipulator `setw(4)` specifies that the next value output should appear in a field width of 4.**
  - If less than 4 character positions wide, the value is right justified in the field by default.
  - If more than 4 character positions wide, the field width is extended to accommodate the entire value.
- **To indicate that values should be output left justified, simply output nonparameterized stream manipulator left .**
- **Right justification can be restored by outputting nonparameterized stream manipulator right.**

# Example using for statement

- Stream manipulator `fixed` indicates that floating-point values should be output as fixed-point values with decimal points.
- Stream manipulator `setprecision` specifies the number of digits to the right of the decimal point.
- Stream manipulators `fixed` and `setprecision` remain in effect until they're changed—such settings are called sticky settings.
- The field width specified with `setw` applies only to the next value output.

# do…while Repetition Statement

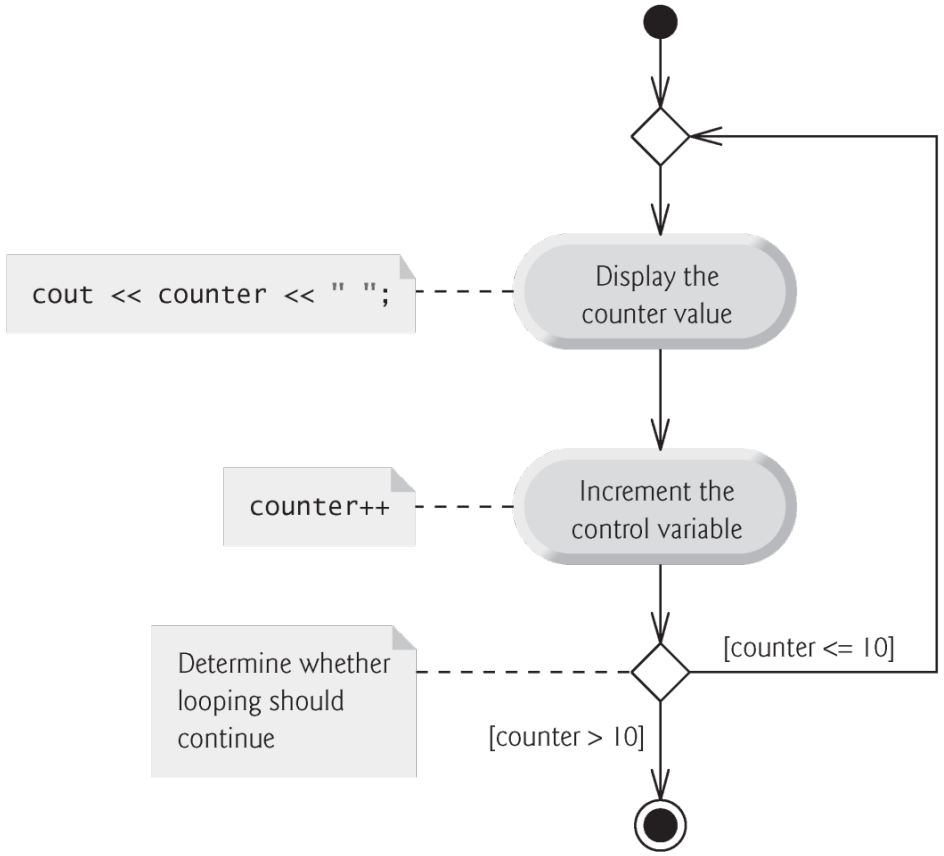
- Similar to the while statement.
- The do…while statement tests the loop-continuation condition after the loop body executes
- The loop body always executes at least once.
- It's not necessary to use braces in the do…while statement if there is only one statement in the body.
- Most programmers include the braces to avoid confusion between the while and do…while statements.
- Must end a do…while statement with a semicolon.

# do…while Repetition Statement

```
1
2  // do...while repetition statement.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int counter = 1; // initialize counter
9
10    do
11    {
12       cout << counter << " "; // display counter
13       counter++; // increment counter
14    } while ( counter <= 10 ); // end do...while
15
16    cout << endl; // output a newline
17 } // end main
```

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

# do…while Repetition Statement



# do…while Repetition Statement

## ❑ while vs. do…while

```
total = 0; grade = 0;
cout << "Enter grade, ";
cout << "-1 to end: ";
cin >> grade;
while (grade != -1) {
  total = total + grade;
  cout << "Enter grade, ";
  cout << "-1 to end: ";
  cin >> grade;
}
```

- ◆Duplicate input statements !!
- ◆Initial values are all zero.

```
total = 0; grade = 0;
do {
  total = total + grade;
  cout << "Enter grade, ";
  cout << "-1 to end: ";
  cin >> grade;
} while (grade != -1);
```

- No duplicate input statements !!
- Initial counter starts from -1.

# Summary

- while loop statement
- do...while loop statement
- for loop statement