PEN203

C++ Program Control

C++ How to Program
Deitel & Deitel

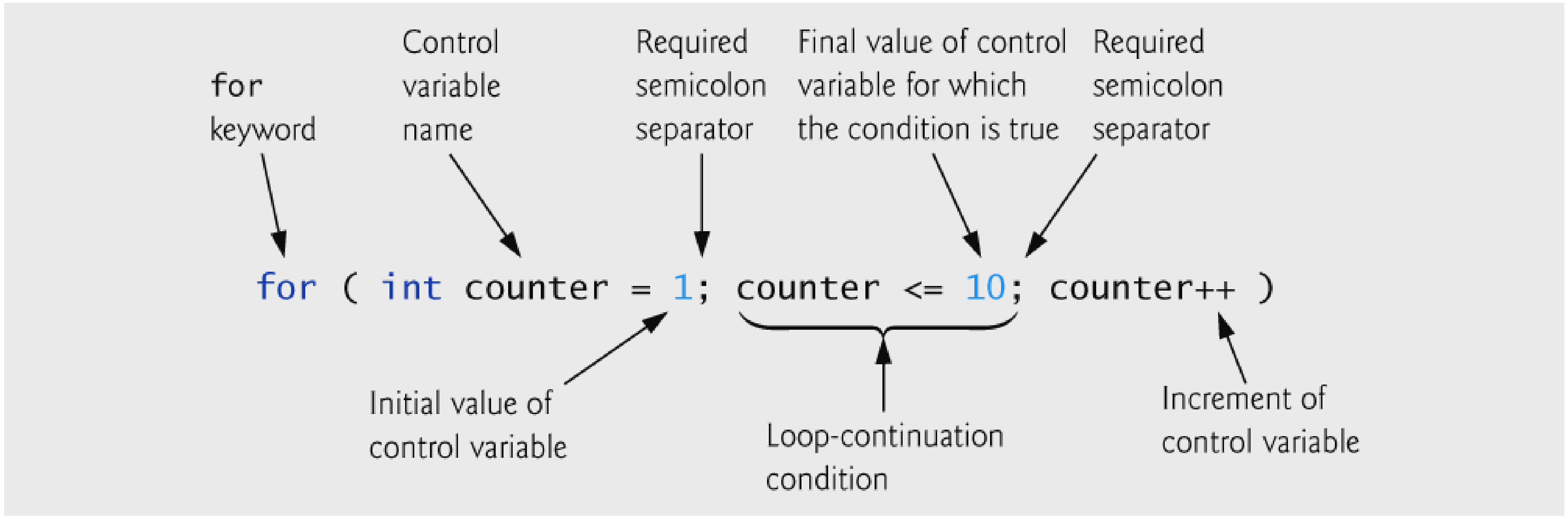
Outline

- for Repetition Statement
- switch Multiple-Selection Statement
- do-while Repetition Statement
- break and continue Statements
- Logical Operators

for Repetition Statement

```
// Fig. 2.17: fig02_17.cpp
// Counter-controlled repetition with the for structure.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
  // Initialization, repetition condition and incrementing
  // are all included in the for structure header.

  for ( int counter = 1; counter <= 10; counter++ )
     cout << counter << endl;

  return 0;   // indicate successful termination

} // end function main
```

for Repetition Statement


for keyword
Control variable name
Required semicolon separator
Final value of control variable for which the condition is true
Required semicolon separator
Initial value of control variable
Loop-continuation condition
Increment of control variable

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

for Repetition Statement

- for(initializaiton; loopContTest; increment or decrement)

  { loop statements; }
- To print integers between 1 and 10
  - for(int counter = 1; counter<=10; counter++)

    ```
            cout<<counter;
    ```
- for loops can usually be rewritten as while loops:

  ```
  initialization;
  while(loopContTest) {
          statement;
          increment;
  }
  ```

for Repetition Statement

- Initialization and increment
  - Can be comma-separated lists
  - Example:

```
for (int a= 0, b = 0;  a * b <= 20; a++, b++)
        cout<<a + b;
```

for Repetition Statement

- Arithmetic expressions can be placed in initialization, loop-continuation, and increment parts.
- Increment may be negative
- Loop variable often is printed or used inside for body. However it is not necessary.
- If the loop continuation condition is initially false, the body of the for statement is not performed.

for Repetition Statement

```
// Fig. 2.20: fig02_20.cpp
// Summation with for.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
  int sum = 0;                          // initialize sum

  // sum even integers from 2 through 100
  for ( int number = 2; number <= 100; number += 2 )
     sum += number;                     // add number to sum

  cout << "Sum is " << sum << endl;  // output sum
  return 0;                          // successful termination

} // end function main
```

switch Multiple-Selection Statement

- switch statement is useful when a variable or expression is tested for all possible values.
- switch statement can have a series of case labels and an optional default case

```
switch ( value ) {
        case 1:
                executable s.
                break;
        case 2:
                executable s.
                break;
        default:
                executable s.
                break;
        }
```

do-while Repetition Statement

- The do-while repetition statement
  - Similar to while structure
  - All actions placed in do-while executed at least once.

```
do {
        executable statements:
} while (condition);
```

  - To print the integers from 1 to 10

```
do {
        cout<<counter;
} while (++counter <= 10);
```

do-while Repetition Statement

```
// Fig. 2.24: fig02_24.cpp
// Using the do/while repetition structure.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
  int counter = 1;              // initialize counter

   do {
      cout << counter << "  ";   // display counter
   } while ( ++counter <= 10 );  // end do/while

   cout << endl;

   return 0;   // indicate successful termination

} // end function main
```

break and continue Statements

- break
  - Used to exit immediately from a while, for, do-while or switch statement.
  - Program execution continues with the first statement after the structure

break and continue Statements

```
// Fig. 2.26: fig02_26.cpp
// Using the break statement in a for structure.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{

  int x;  // x declared here so it can be used after the loop

  // loop 10 times
  for ( x = 1; x <= 10; x++ ) {

    // if x is 5, terminate loop
    if ( x == 5 )
      break;          // break loop only if x is 5

    cout << x << " ";   // display value of x

  } // end for

  cout << "\nBroke out of loop when x became " << x << endl;

  return 0;   // indicate successful termination

} // end function main
```

break and continue Statements

- continue
  - Skips the remaining statements in the body of a while, for or do-while

```
// Fig. 2.27: fig02_27.cpp
// Using the continue statement in a for structure.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
  // loop 10 times
  for ( int x = 1; x <= 10; x++ ) {

    // if x is 5, continue with next iteration of loop
    if ( x == 5 )
      continue;      // skip remaining code in loop body

    cout << x << " ";   // display value of x

  } // end for structure

  cout << "\nUsed continue to skip printing the value 5"
     << endl;

  return 0;            // indicate successful termination
```

Logical Operators

- && (logical AND)
  - Returns true if both conditions are true
- | | (logical OR)
  - Returns true if either of its conditions are true
- ! (logical NOT)
  - Reverse the truth of given condition