The `for` Statement

- Same function as the `while` statement but in different form

```
for (initializing list; expression; altering list)
   statement;
```

- Function: statement executed while expression has non-zero (true) value
- Components:
  - Initializing list: initial value of expression
  - Expression: a valid C++ expression
  - Altering list: statements executed at end of each `for` loop to alter value of expression

The `for` Statement (cont'd.)

- Components of `for` statement correspond to operations performed in `while` statement
  - Initialization
  - Expression evaluation
  - Altering of expression values
- Components of `for` statement are optional, but semicolons must always be present

The `for` Statement (cont'd.)

Program 5.9

```
#include <iostream>
using namespace std;
int main()
{
  int count;
  for (count = 2; count <= 20; count = count + 2)
    cout << count << " ";

  return 0;
}
```

This is the output of Program 5.9:

```
2 4 6 8 10 12 14 16 18 20
```

The `for` Statement (cont'd.)

- Program 5.9 modified: initializer outside `for` loop

Program 5.9a

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

int main()
{
  int count;

  count = 2;    // initialized outside the for statement
  for ( ; count <= 20; count = count + 2)
    cout << count << " ";

  return 0;
}
```

The `for` Statement (cont'd.)

Program 5.10

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

int main()
{
  const int MAXNUMS = 10;
  int num;
  cout << endl;      // print a blank line
  cout << "NUMBER   SQUARE   CUBE\n"
       << "------   ------   ----\n";
  for (num = 1; num <= MAXNUMS; num++)
    cout << setw(3) << num << "       "
         << setw(3) << num * num << "     "
         << setw(4) << num * num * num << endl;

  return 0;
}
```

The `for` Statement (cont'd.)

- When Program 5.10 is run, the display produced is:

```
 NUMBER          SQUARE          CUBE

_ _ _ _ _ _    _ _ _ _ _ _ _    _ _ _ _

      1               1              1
      2               4              8
      3               9             27
      4              16             64
      5              25            125
      6              36            216
      7              49            343
      8              64            512
      9              81            729
     10             100           1000
```

Interactive `for` Loops

- Same effect as using `cin` object within a `while` loop
- Provides interactive user input

Program 5.11

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

// This program calculates the average
// of MAXCOUNT user-entered numbers
int main()
{
  const int MAXCOUNT = 5;
  int count;
  double num, total, average;

  total = 0.0;

  for (count = 0; count < MAXCOUNT; count++)
  {
    cout << "Enter a number: ";
    cin  >> num;
    total = total + num;
  }

  average = total / count;
  cout << "The average of the data entered is " << average
       << endl;

  return 0;
}
```

Interactive `for` Loops (cont'd.)

- Program 5.11: `for` statement creates a loop
  - Loop executed five times
- Actions performed in each loop
  - User prompted to enter a number
  - Number added to the total

Interactive `for` Loops (cont'd.)

- Initialization variations:
  - Alternative 1: initialize `total` outside the loop and `count` inside the loop as in Program 5.11
  - Alternative 2: initialize both `total` and `count` inside loop

    ```
    for (total = 0.0, count = 0; count < MAXCOUNT;
    count++)
    ```

  - Alternative 3: initialize and declare both `total` and `count` inside loop

    ```
    for (double total = 0.0, int count = 0; count <
    MAXCOUNT; count++)
    ```

Nested Loops

- A loop contained within another loop
- Example:

```
for(i = 1; i <= 5; i++)          // start of outer loop
{
 cout << "\ni is now " << i << endl;

 for(j = 1; j <= 4; j++)     // start of inner loop
 {
    cout << " j = " << j;
 }                                // end of inner loop
}                                 // end of outer loop
```

Nested Loops (cont'd.)

- Outer (first) loop:
  - Controlled by value of `i`
- Inner (second) loop:
  - Controlled by value of `j`
- Rules:
  - For each single trip through outer loop, inner loop runs through its entire sequence
  - Different variable to control each loop
  - Inner loop statements contained within outer loop

Nested Loops (cont'd.)

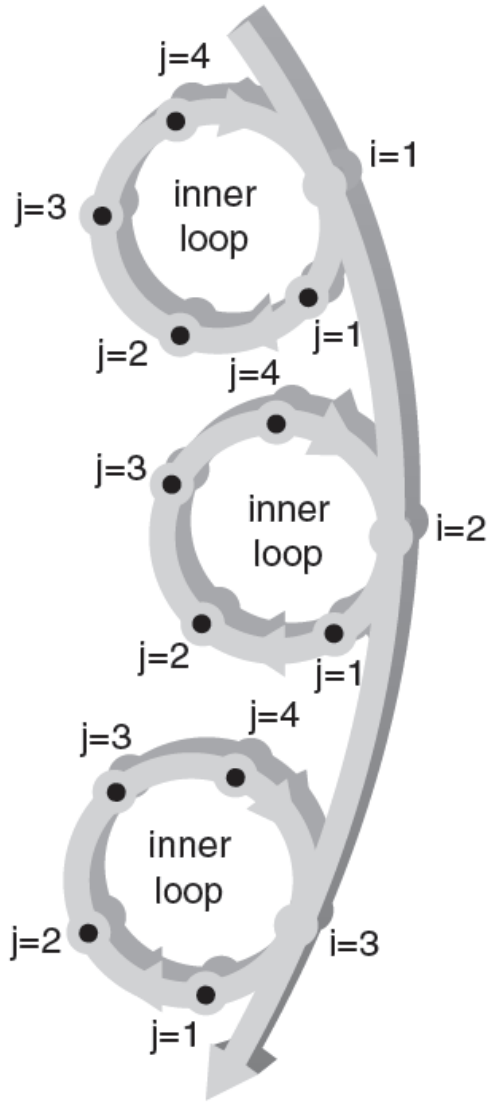

Figure 5.7 For each value of `i`, `j` loops four times