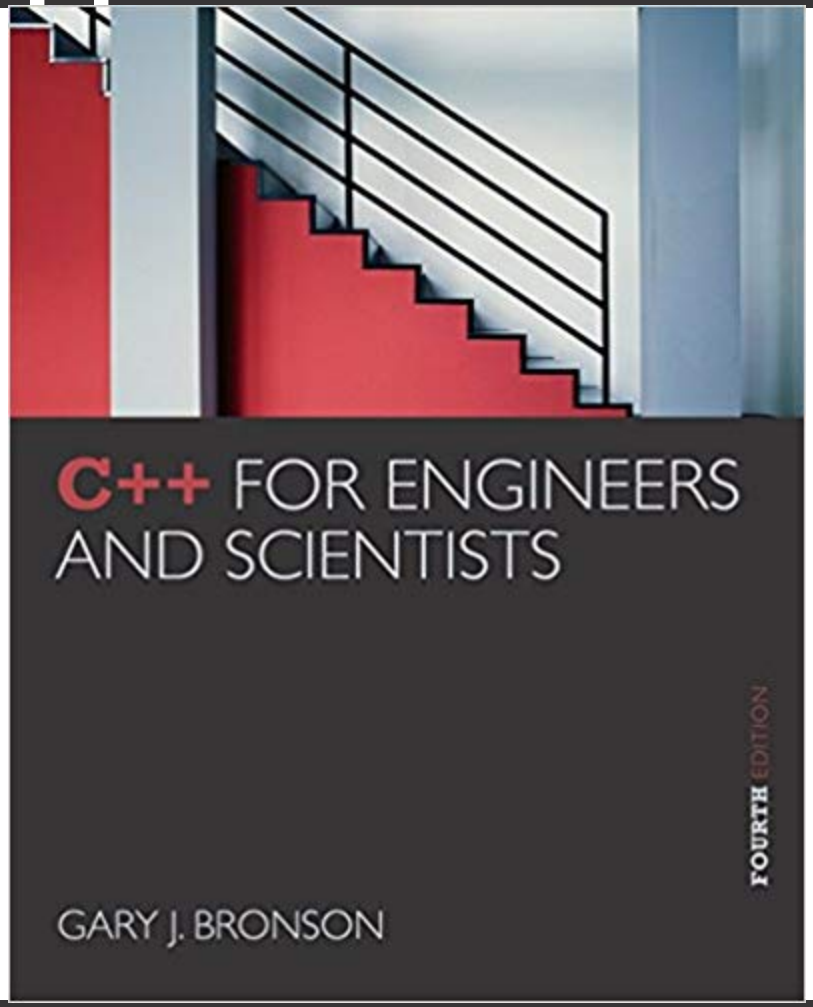
ELEG 1043
Computer Applications in Engineering



Chapter 5:
Repetition Statements

C++ FOR ENGINEERS
AND SCIENTISTS

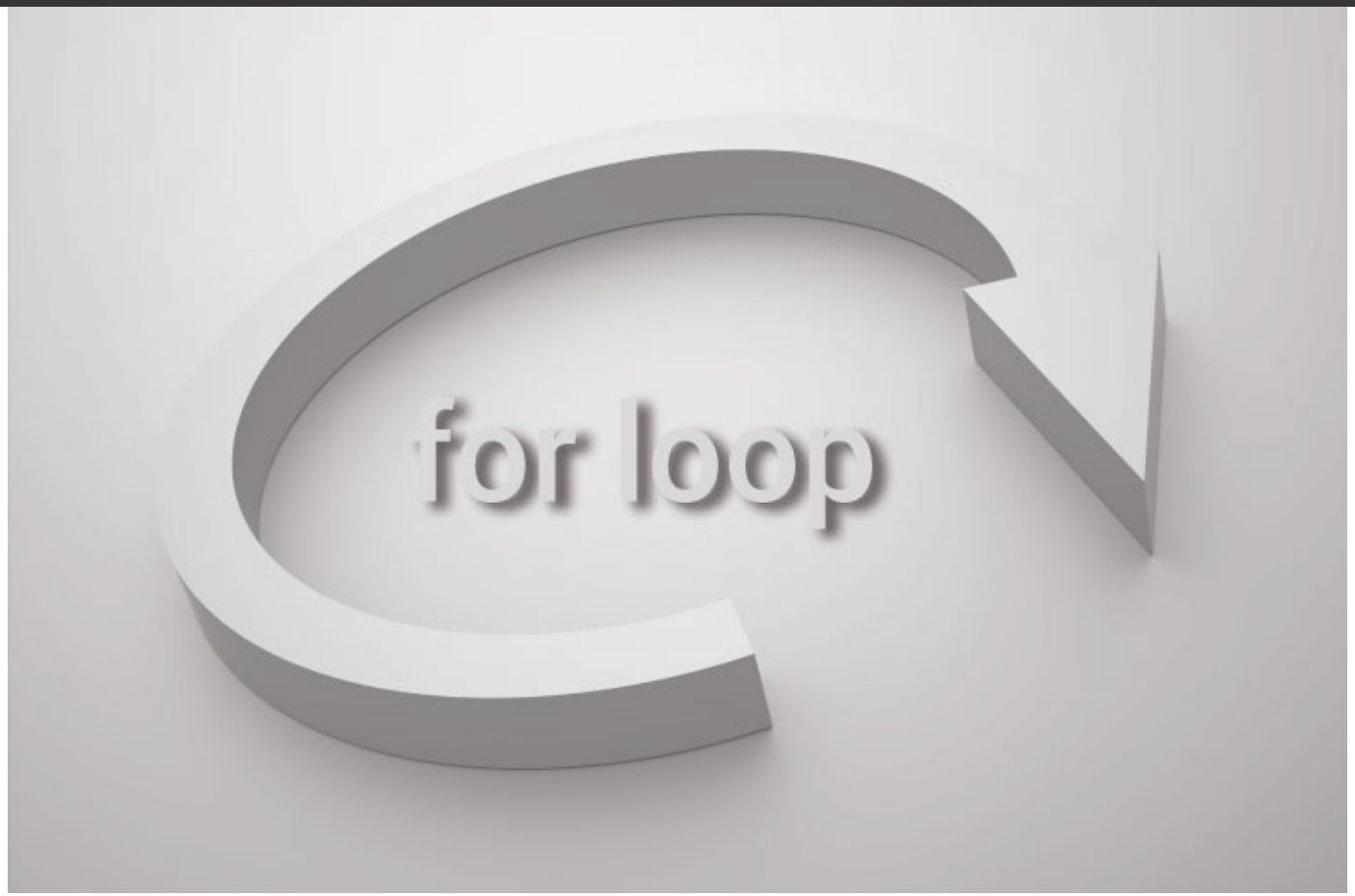
Acknowledgement

- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

The Null Statement

- **Null statement**
  - Semicolon with nothing preceding it
    - `;`
  - Do-nothing statement required for syntax purposes only

for Loops



https://cdn.programiz.com/sites/tutorial2program/files/C%2B%2Bfor-loop.jpg

`for` Loops

- `for` statement: A loop with a fixed count condition that handles alteration of the condition
  - Syntax:

    for (initializing list; expression; altering list)

    statement;
- **Initializing list:** Sets the starting value of a counter
- **Expression:** Contains the maximum or minimum value the counter can have; determines when the loop is finished

`for` Loops (continued)

- **Altering list:** Provides the increment value that is added or subtracted from the counter in each iteration of the loop
- If initializing list is missing, the counter initial value must be provided prior to entering the `for` loop
- If altering list is missing, the counter must be altered in the loop body
- Omitting the expression will result in an infinite loop

for Loops (continued)

Program 5.9

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

int main()
{
  const int MAXCOUNT = 5;
  int count;

  cout << "NUMBER  SQUARE ROOT\n";
  cout << "------    -----------\n";

  for (count = 1; count <= MAXCOUNT; count++)
    cout << setw(4) << count
         << setw(15) << sqrt(double(count)) << endl;

  return 0;
}
```

for Loops (cont'd)

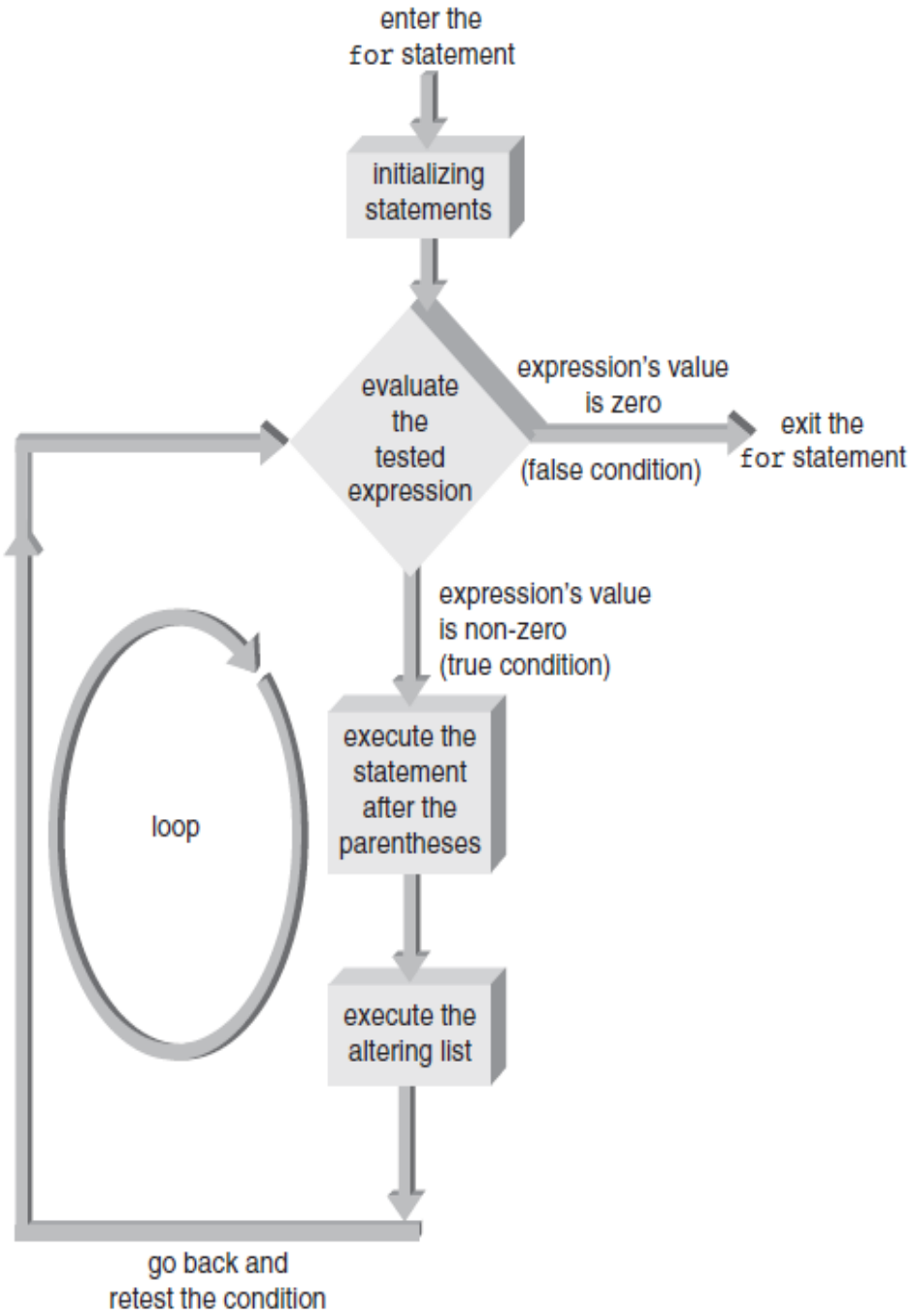


Figure 5.10 `for` loop flowchart.

A Closer Look: Loop Programming Techniques

- These techniques are suitable for pretest loops (**`for`** and **`while`**):
  - **Interactive input within a loop**
    - Includes a **`cin`** statement within a **`while`** or **`for`** loop
  - **Selection within a loop**
    - Using a **`for`** or **`while`** loop to cycle through a set of values to select those values that meet some criteria

A Closer Look: Loop Programming Techniques (continued)

Program 5.13

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

// This program computes the positive and negative sums of a set
// of MAXNUMS user-entered numbers
int main()
{
  const int MAXNUMS = 5;
  int i;
  double usenum, positiveSum, negativeSum;
```

A Closer Look: Loop Programming Techniques (continued)

```
  positiveSum = 0; // this initialization can be done in the declaration
  negativeSum = 0; // this initialization can be done in the declaration
  for (i = 1; i <= MAXNUMS; i++)
  {
    cout << "Enter a number (positive or negative) : ";
    cin  >> usenum;
    if (usenum > 0)
      positiveSum = positiveSum + usenum;
    else
      negativeSum = negativeSum + usenum;
  }
  cout << "The positive total is " << positiveSum << endl;
  cout << "The negative total is " << negativeSum << endl;

  return 0;
}
```

A Closer Look: Loop Programming Techniques (continued)

Program 5.14

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

int main()
{
  int x, y;

  cout << "x value    y value\n"
       << "-------    --------\n";

  for (x = 2; x <= 6; x++)
  {
    y = 10 * pow(x,2.0) + 3 * x - 2;
    cout << setw(4) << x
         << setw(11) << y << endl;
  }

  return 0;
}
```

A Closer Look: Loop Programming Techniques (continued)

- **Interactive loop control**
  - Variable is used to control the loop repetitions
  - Provides more flexibility at run-time
- **Random numbers and simulation**
  - Pseudorandom generator used for simulators
  - C++ functions: `rand(); srand()`

A Closer Look: Loop Programming Techniques (continued)

Program 5.16

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

// This program displays a table of numbers with their squares and
// cubes, starting from the number 1. The final number in the table
// is input by the user.

int main()
{
  int num, final;

  cout << "Enter the final number for the table: ";
  cin  >> final;
  cout << "NUMBER SQUARE CUBE\n";
  cout << "------ ------ ----\n";

  for (num = 1; num <= final; num++)
    cout << setw(3) << num
         << setw(8) << num * num
         << setw(7) << num * num * num << endl;

  return 0;
}
```

A Closer Look: Loop Programming Techniques (continued)

Program 5.17

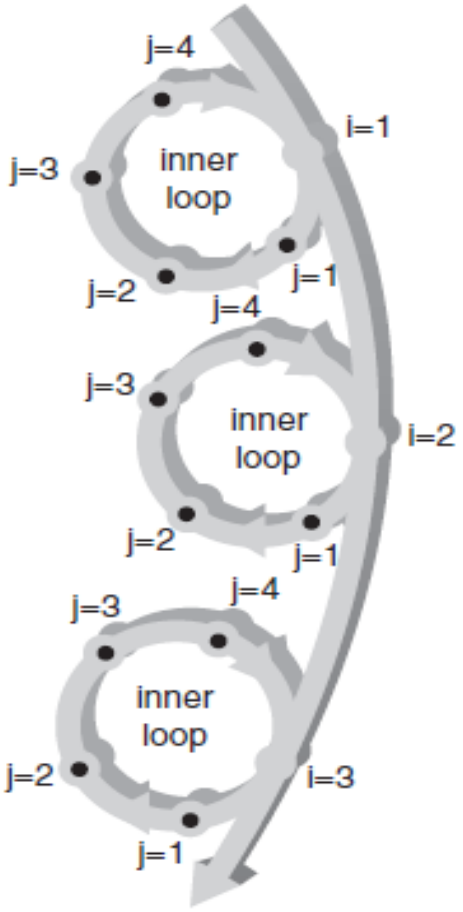
```
#include <iostream>
#include <cmath>
#include <ctime>
using namespace std;

// This program generates 10 pseudorandom numbers
// with C++'s rand() function

int main()
{
const int NUMBERS = 10;
  double randvalue;
  int i;

  srand(time(NULL));  // generates the first seed value
  for (i = 1; i <= NUMBERS; i++)
  {
    randvalue = rand();
    cout << randvalue << endl;
  }

  return 0;
}
```

Nested Loops

- **Nested loop:** A loop contained within another loop
  - All statements of the inner loop must be completely contained within the outer loop; no overlap allowed
  - Different variables must be used to control each loop
  - For each single iteration of the outer loop, the inner loop runs through all of its iterations

Nested Loops (continued)

Figure 5.12 For each `i, j` loops.



Nested Loops (continued)

Program 5.19

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

int main()
{
  const int MAXI = 5;
  const int MAXJ = 4;
  int i, j;

  for (i = 1; i <= MAXI; i++)     // start of outer loop <----+
  {                               //                          |
    cout << "\ni is now " << i << endl;  //                   |
                                  //                          |
    for (j = 1; j <= MAXJ; j++)   // start of inner loop      |
     cout << "   j = " << j;      // end of inner loop        |
  }                               // end of outer loop  <-----+
  cout << endl;

  return 0;
}
```

`do while` Loops

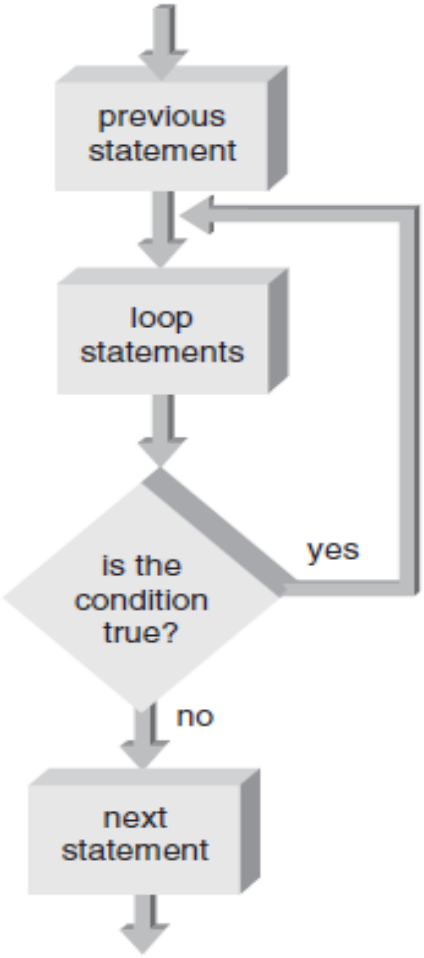
- **`do while`** loop is a posttest loop
  - Loop continues while the condition is true
  - Condition is tested at the end of the loop
  - Syntax:

```
do
        statement;
while (expression);
```

- All statements are executed at least once in a posttest loop

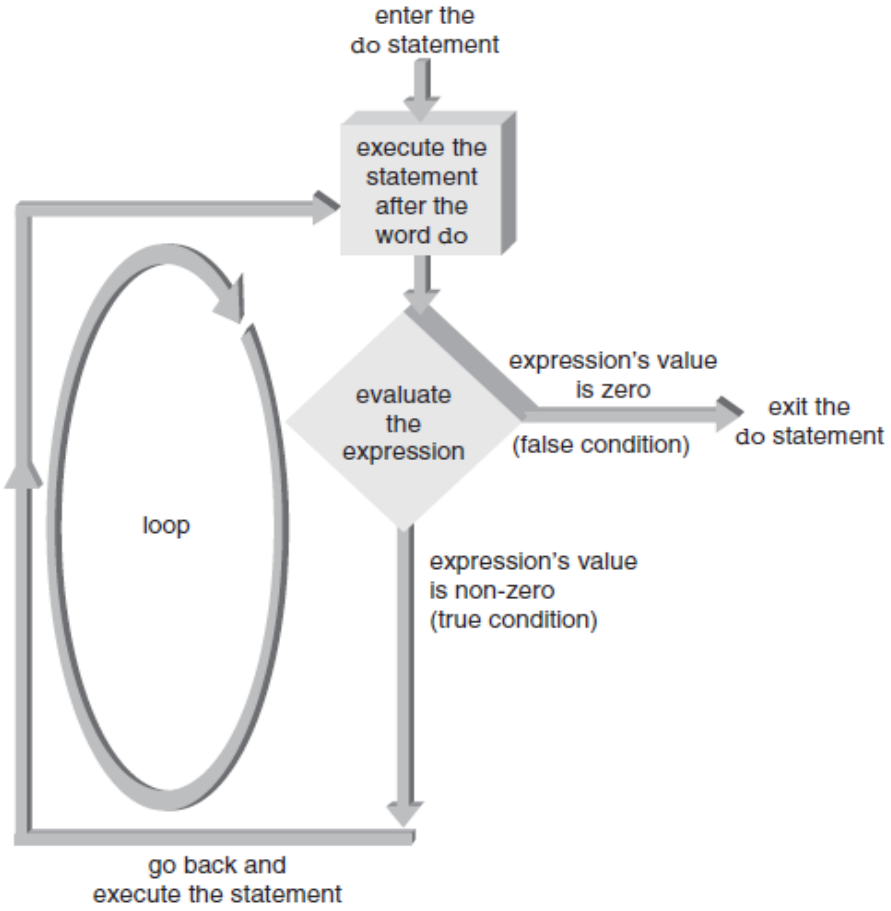
do while Loops

Figure 5.13 The `do while` loop structure.



do while Loops

Figure 5.14 The `do` statement's flow of control.



Validity Checks

- Useful in filtering user-entered input and providing data validation checks

```
do
{
  cout << "\nEnter an identification number: ";
  cin  >> id_num;
}
while (id_num < 1000 || id_num > 1999);
```

Common Programming Errors

- Using the assignment operator (=) instead of the equality comparison operator (==) in the condition expression
- Testing for equality with floating-point or double-precision operands; use an epsilon value instead

Common Programming Errors (continued)

- Placing a semicolon at the end of the **`for`** clause, which produces a null loop body
- Using commas (,) instead of semicolons (;) to separate items in the **`for`** statement
- Changing the value of the control variable
- Omitting the final semicolon in a **`do`** statement

Summary

- Loop: A section of repeating code, whose repetitions are controlled by testing a condition
- Three types of loops:
  - **`while`**
  - **`for`**
  - **`do while`**
- Pretest loop: Condition is tested at beginning of loop; loop body may not ever execute; ex., **`while`**, **`for`** loops

Summary (continued)

- Posttest loop: Condition is tested at end of loop; loop body executes at least once; ex., **`do while`**
- Fixed-count loop: Number of repetitions is set in the loop condition
- Variable-condition loop: Number of repetitions is controlled by the value of a variable