Tony Gaddis 5th Ed
Starting Out with C++

COMPUTER SCIENCE

CHAPTER 5
LOOPING

Mohammad Al-Husseini

Increment and Decrement

- Increment means to increase a value by one
  - num = num + 1;
    num += 1;
  - There is a simple short hand for this, known as the increment operator
    - num++;
- Decrementation is the same, except you reduce the value by 1
  - num = num – 1;
    num -= 1;
    - num--;
- Note: These change the value of the variable

Prefix and Postfix Modes

- num++ is using the increment operator in postfix mode
  - This means that the incrementation is the last operation in the statement
  - num = 4;
    cout << num++;
    - This displays 4, but num becomes 5
- ++num is prefix mode
  - So the incrementation happens first
  - num = 4;
    cout << ++num;
    - This displays 5 and num becomes 5

Prefix and Postfix Modes

```
1  // This program demonstrates the prefix and postfix
2  // modes of the increment and decrement operators.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int num = 4;
9
10    cout << num << endl;    // Displays 4
11    cout << num++ << endl; // Displays 4, then adds 1 to num
12    cout << num << endl;    // Displays 5
13    cout << ++num << endl; // Adds 1 to num, then displays 6
14    cout << endl;           // Displays a blank line
15
16    cout << num << endl;    // Displays 6
17    cout << num-- << endl; // Displays 6, then subtracts 1 from num
18    cout << num << endl;    // Displays 5
19    cout << --num << endl; // Subtracts 1 from num, then displays 4
20
21    return 0;
22 }
```

++ AND -- In

- You can use these in any expression with a single variable
  - Mathematical Expressions
    - a = 2, b = 5, c = a * b++;
    - Beware: c = ++(a * b);
  - Relational Expressions
    - x = 10;
      if (x++ > 10)
        cout << “x is greater than 10.”;

Checkpoint

1. What will the following display?

   A.
   ```
   x = 2;
   y = x++;
   cout << x << y;
   ```

   B.
   ```
   x = 2;
   y = ++x;
   cout << x << y;
   ```

   C.
   ```
   x = 2;
   y = 4;
   cout << x++ << --y;
   ```

   D.
   ```
   x = 2;
   y = 2 * x++;
   cout << x << y;
   ```

Checkpoint

1. What will the following display?

   E.
   ```
   x = 99;
   if (x++ < 100)
      cout << “It is true!”;
   else
      cout << “It is false!”;
   ```

   F.
   ```
   x = 0;
   if (++x)
      cout << “It is true!”;
   else
      cout << “It is false!”;
   ```

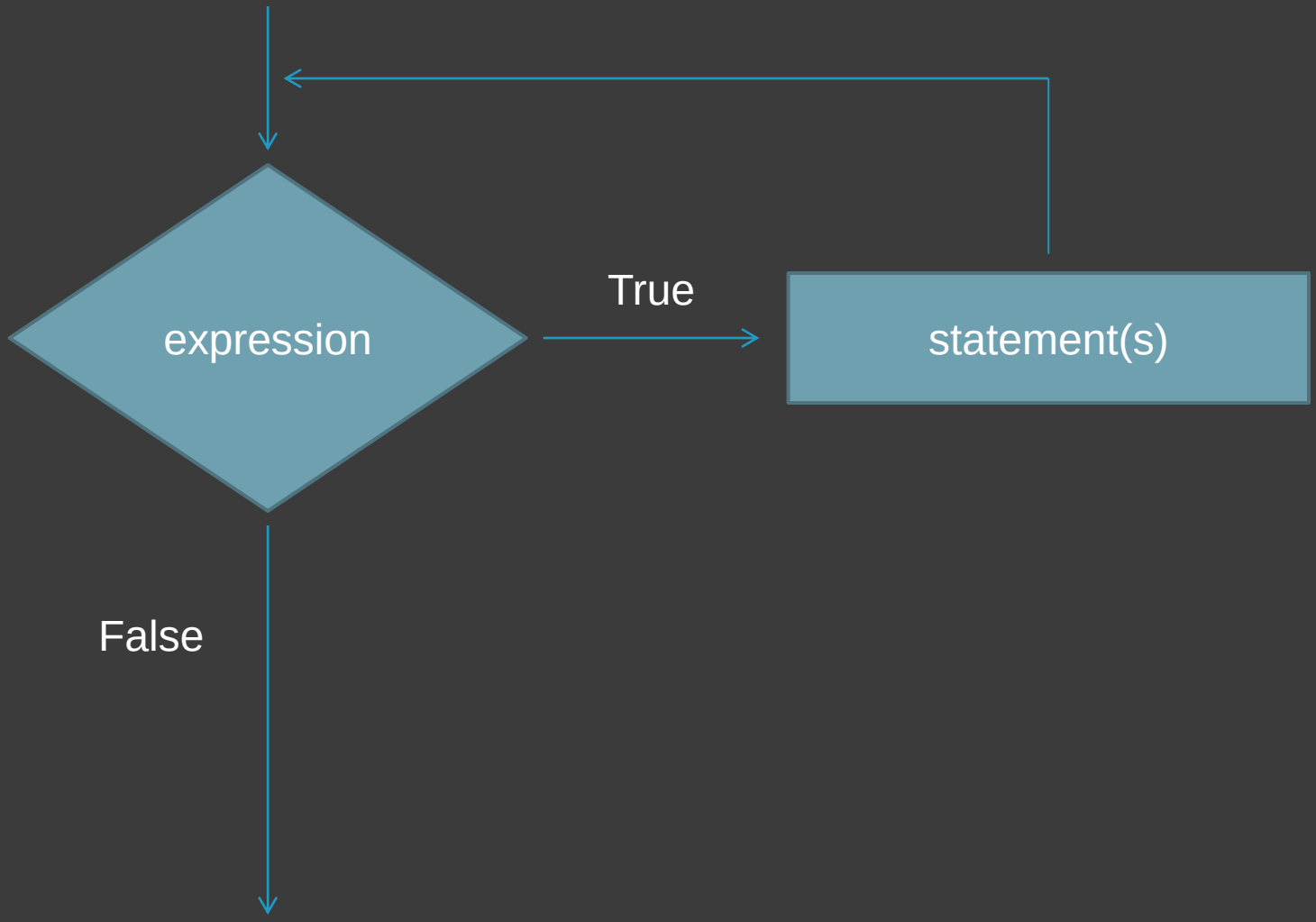
Loops

- Essentially an if statement that continues repeating until false
- Three varieties that vary in the way they continue the repetition
  - while
  - do-while
  - for

The While Loop

- Continues looping while the expression is true
- while (expression)
  statement;
- while (expression)
  {
  statement;
  …
  statement;
  }
- Will not execute (or will stop executing) when the expression is/becomes false

The While Loop



The While Loop

```
1  // This program demonstrates a simple while loop.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int number = 1;
8
9     while (number <= 5)
10    {
11       cout << "Hello\n";
12       number++;
13    }
14    cout << "That's all!\n";
15    return 0;
16 }
```

- “Hello\n” five times
  “That’s all!\n”

The While Loop

- Is a pretest loop, meaning it checks if the expression is true before executing
  - int number = 6;
    while (number <= 5)
    {
    cout << “Hello\n”;
    number++;
    }
  - Displays nothing

Infinite Loops

- If the expression is never becomes false, you will be stuck in the loop forever
  - int number = 1;
    while (number <= 5)
      cout << “Hello\n”;
- If you place a semicolon after the loop, it will also become infinite
  - int number = 1;
    while (number <= 5);
    {
      cout << “Hello\n”;
      number++;
    }

Loop Braces

- Like if statements, you must include braces if you want statements past the first to be included

  - ```
    int number = 1;
    while(number <= 5)
        cout << “Hello\n”;
        number++;
    ```

  - This is an infinite loop, because the incrementation happens after the loop, which never ends
  - Remember good programming style: indentations and braces
    ```

Validating Input

- while loops are excellent for input validation
- The loop can repeat until the correct input is entered
  - cout << “Enter a number in the range 1 – 100: ”;
    cin >> number;

```
while (number < 1 || number > 100)
{
   cout << “ERROR: Enter a value in the
                  range 1-100: “;
   cin >> number;
}
```

Validating Input

```
1  // This program calculates the number of soccer teams
2  // that a youth league may create from the number of
3  // available players. Input validation is demonstrated
4  // with while loops.
5  #include <iostream>
6  using namespace std;
7
8  int main()
9  {
10    int players,       // Number of available players
11        teamPlayers,   // Number of desired players per team
12        numTeams,      // Number of teams
13        leftOver;      // Number of players left over
14
15    // Get the number of players per team.
16    cout << "How many players do you wish per team?\n";
17    cout << "(Enter a value in the range 9 - 15): ";
18    cin >> teamPlayers;
19
20    // Validate the input.
21    while (teamPlayers < 9 || teamPlayers > 15)
22    {
23       cout << "You should have at least 9 but no\n";
24       cout << "more than 15 per team.\n";
25       cout << "How many players do you wish per team? ";
26       cin >> teamPlayers;
27    }
```

Validating Input

```
25          cout << "How many players do you wish per team? ";
26          cin >> teamPlayers;
27      }
28
29      // Get the number of players available.
30      cout << "How many players are available? ";
31      cin >> players;
32
33      // Validate the input.
34      while (players <= 0)
35      {
36          cout << "Please enter a positive number: ";
37          cin >> players;
38      }
39
40      // Calculate the number of teams.
41      numTeams = players / teamPlayers;
42
43      // Calculate the number of leftover players.
44      leftOver = players % teamPlayers;
45
46      // Display the results.
47      cout << "There will be " << numTeams << " teams with ";
48      cout << leftOver << " players left over.\n";
49      return 0;
50  }
```

Counters

- Just a variable that increments or decrements each time a loop iterates
- Counts the number of times a loop iterates
- Must be initialized

- ```
  int num = 0; // counter
  while (num++ < 10)
      cout << num << " " << num * num << endl;
  ```
  ```

Checkpoint

2. Write an input validation loop that asks the user to enter a number in the range of 10 through 25.

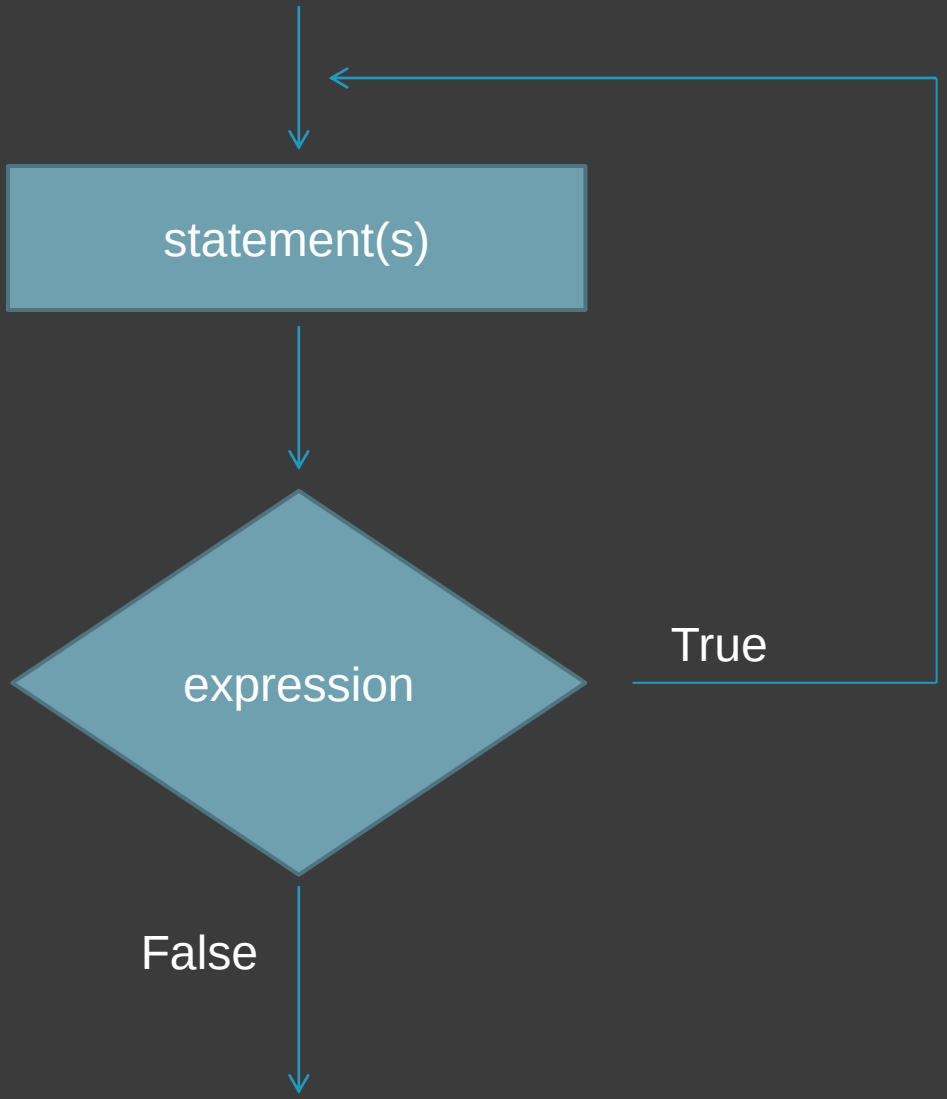
3. Write an input validation loop that asks the user to enter ‘Y’, ‘y’, ‘N’, or ‘n’.

4. Write an input validation loop that asks the user to enter “Yes” or “No”.

The do-while Loop

- Like the while loop, but it is posttest
  - It does the statements first, then tests the expression to loop
- do
  {
  statement;
  …
  statement;
  } while (expression);
- Don’t forget the semicolon after the while (but only for do-while)

The do-while Loop



The do-while Loop

```
1  // This program averages 3 test scores. It repeats as
2  // many times as the user wishes.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int score1, score2, score3; // Three scores
9     double average;             // Average score
10    char again;                 // To hold Y or N input
11
12    do
13    {
14       // Get three scores.
15       cout << "Enter 3 scores and I will average them: ";
16       cin >> score1 >> score2 >> score3;
17
18       // Calculate and display the average.
19       average = (score1 + score2 + score3) / 3.0;
20       cout << "The average is " << average << ".\n";
21
22       // Does the user want to average another set?
23       cout << "Do you want to average another set? (Y/N) ";
24       cin >> again;
25    } while (again == 'Y' || again == 'y');
26    return 0;
27 }
```

The do-while Loop

- Essentially a while loop, but you want it to run the code one time first (post test)
- Don’t forget the semicolon after the while portion (but not for a regular while loop)
- Particularly useful with menu driven programs

do-while Loop Menu

```
1  // This program displays a menu and asks the user to make a
2  // selection. A do-while loop repeats the program until the
3  // user selects item 4 from the menu.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  int main()
9  {
10    int choice;        // Menu choice
11    int months;        // Number of months
12    double charges;    // Monthly charges
13
14    // Constants for membership rates
15    const double ADULT = 40.0;
16    const double SENIOR = 30.0;
17    const double CHILD = 20.0;
18
19    // Set up numeric output formatting.
20    cout << fixed << showpoint << setprecision(2);
21
22    do
23    {
24       // Display the menu.
25       cout << "\n\t\tHealth Club Membership Menu\n\n";
26       cout << "1. Standard Adult Membership\n";
27       cout << "2. Child Membership\n";
28       cout << "3. Senior Citizen Membership\n";
29       cout << "4. Quit the Program\n\n";
30       cout << "Enter your choice: ";
31       cin >> choice;
```

do-while Loop Menu

```
33          // Validate the menu selection.
34          while (choice < 1 || choice > 4)
35          {
36             cout << "Please enter 1, 2, 3, or 4: ";
37             cin >> choice;
38          }
39
40          // Validate and process the user's choice.
41          if (choice != 4)
42          {
43             // Get the number of months.
44             cout << "For how many months? ";
45             cin >> months;
46
47             // Respond to the user's menu selection.
48             switch (choice)
49             {
50                case 1:  charges = months * ADULT;
51                         break;
52                case 2:  charges = months * CHILD;
53                         break;
54                case 3:  charges = months * SENIOR;
55             }
56
57             // Display the monthly charges.
58             cout << "The total charges are $";
59             cout << charges << endl;
60          }
61       } while (choice != 4);
62       return 0;
63    }
```

Checkpoint

5. What will the following display?

   A.
```
int count = 10;
do
   cout << “Hello World” << endl;
while (count++ < 1);
```

   B.
```
int v = 0;
do
   cout << v++;
while (v < 5);
```

Checkpoint

C.
```
int count = 0, funny = 1, serious = 0, limit = 4;
do
{
   funny++:
   serious += 2;
} while (count++ < limit);
cout << funny << " " << serious << " " << count;
```

6. Write a program segment with a do-while loop that asks the user to enter a number. The loop should keep a running total of the numbers entered, and stop when the total is greater than 300.

The for Loop

- Two categories of loops
  - conditional loop = executes as long as a particular condition exists, such as while and do-while
  - count-controlled loop = iterates an exact number of times (such as 12 months in a year), this is the “for”
- There are three steps to count-controlled loops
  1. Initialize a counter variable to a starting value
  2. Test the counter variable compared to the maximum value and terminate the loop if reached
  3. Update the counter variable each iteration
- Count-controlled loops are so common that C++ made the for loop

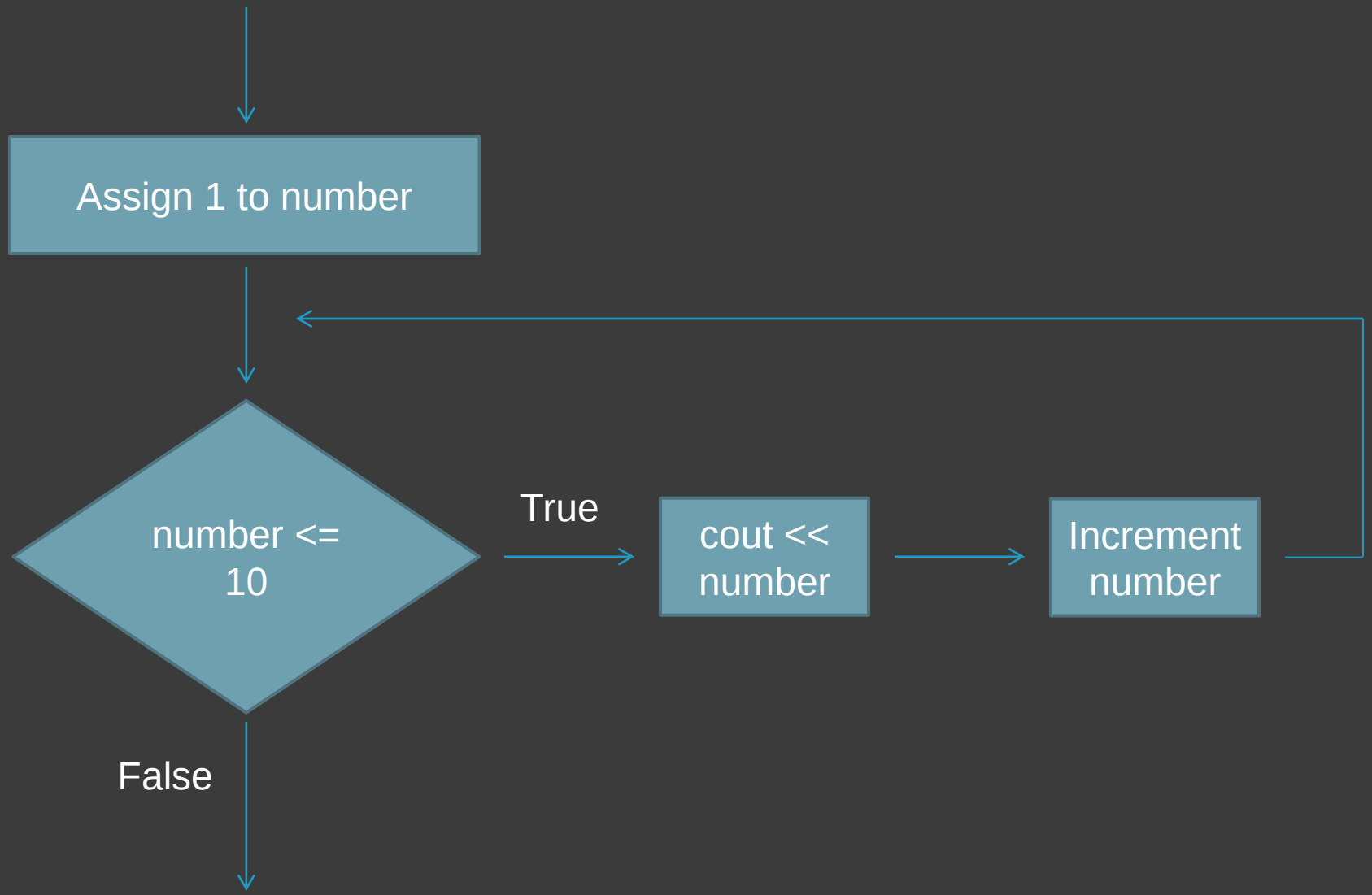
The for Loop

- for (initialization; test; update)
  {
  statement;
  …
  statement;
  }

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

- Note the semicolons

The for Loop


Assign 1 to number
number <= 10
True
cout << number
Increment number
False

The for Loop

```
1  // This program displays the numbers 1 through 10 and
2  // their squares.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int num;
9
10    cout << "Number    Number Squared\n";
11    cout << "-------------------------\n";
12
13    for (num = 1; num <= 10; num++)
14       cout << num << "\t\t" << (num * num) << endl;
15    return 0;
16 }
```

The for Loop

- Pretest loop
  - int count = 0;
    for (count = 11; count <= 10; count++)
      cout << “Hello” << endl;
    - Will not activate
  - Also note that you can use an existing variable instead of defining it within the for header
  - Beware adjusting count inside the body
    - int count;
      for (count = 0; count <= 10; count++)
      {
        cout << “Hello” << endl;
        count++;
      }

The for Loop

- You can use various forms of update expressions, provided they eventually cause the loop to terminate

  - for (num = 2; num <= 100; num += 2)
    cout << num << endl;

  - for (num = 10; num >= 0; num--)
    cout << num << endl;

The for Loop

- If you define the counter variable in the for header, it only have scope within the loop

  - ```
    for (int count = 1; count <= 10; count++)
        cout << count << endl;
    cout << “count is now “ << count;          // Error
    ```

  - ```
    int count;
    for(count = 1; count <= 10; count++)
        cout << count << endl;
    cout << “count is now “ << count;          // Valid
    ```

User Controlled for Loop

```
1  // This program demonstrates a user controlled for loop.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int num;        // Loop counter variable
8     int maxValue;   // Maximum value to display
9
10    // Get the maximum value to display.
11    cout << "I will display a table of numbers and\n";
12    cout << "their squares. How high should I go? ";
13    cin >> maxValue;
14
15    cout << "\nNumber    Number Squared\n";
16    cout << "-------------------------\n";
17
18    for (num = 1; num <= maxValue; num++)
19       cout << num << "\t\t" << (num * num) << endl;
20    return 0;
21 }
```

The for Loop

- You can have more than one statement in the initialization and test expressions
  - int x, y;
    for (x = 1, y = 1; x <= 5; x++, y++)
    {
       cout << x << " plus " << y
            << " equals " << x + y << endl;
    }
- Do not try this with the test expression, however
  - You must use && or ||

The for Loop

- You can omit any or all of a for loop's expressions
  - ```
    int num = 1;
    for ( ; num <= maxValue; num++)
       cout << num << endl;
    ```
  - ```
    int num = 1;
    for ( ; num <= maxValue; )
    {
       cout << num << endl;
       num ++;
    }
    ```
  - ```
    for ( ; ; )
       cout << "Hello World!" << endl;     // Infinite Loop
    ```
    ```

Keeping a Running Total

- Sometimes you need to calculate the total of a series of numbers that are provided as input
- This can be done with a variable, known as an accumulator

Keeping a Running Total

```
1  // This program takes daily sales figures over a period of time
2  // and calculates their total.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6
7  int main()
8  {
9     int days;             // Number of days
10    double total = 0.0;   // Accumulator, initialized with 0
11
12    // Get the number of days.
13    cout << "For how many days do you have sales figures? ";
14    cin >> days;
15
16    // Get the sales for each day and accumulate a total.
17    for (int count = 1; count <= days; count++)
18    {
19       double sales;
20       cout << "Enter the sales for day " << count << ": ";
21       cin >> sales;
22       total += sales;    // Accumulate the running total.
23    }
24
25    // Display the total sales.
26    cout << fixed << showpoint << setprecision(2);
27    cout << "The total sales are $" << total << endl;
28    return 0;
29 }
```

Which Loop?

- While
  - Pretest
  - Good for validation
  - When you don’t know the number of loops
- Do-While
  - Posttest
  - Good for menus
  - When you want the code to run at least once
- For
  - Pretest
  - You know how many times you want to loop

Checkpoint

7. Name the three expressions in the for header.

8. You want to write a for loop that displays “I love to program” 50 times. Your counter variable is declared as count.
   A. What initialization expression will you use?
   B. What test expression will you use?
   C. What update expression will you use?
   D. Write the loop.

Checkpoint

9. What will the following display?
   A.
```
for (int count = 0; count < 6; count++)
   count << (count + count);
```
   B.
```
for (int value = -5; value < 5; value++)
   cout << value;
```
   C.
```
int x;
for (x = 5; x <= 14; x += 3)
   cout << x << endl;
cout << x;
```

10. Write a for loop that displays your name 10 times.

Checkpoint

11. Write a for loop that displays all the odd numbers, 1 through 49.
12. Write a for loop that displays every fifth number, 0 through 100.
13. Write a for loop that repeats seven times, asking the user to enter a number. The loop should also calculate the sumo f the numbers entered.
14. Write a for loop that calculates the following

$$\frac{1}{30}+\frac{2}{29}+\frac{3}{28}+\frac{4}{27}+\ldots+\frac{30}{1}$$

Sentinels

- A special value that marks the end of a list
- Cannot be a value that could appear in the list
- You choose the value
- Useful for when you don’t know how long the list is

Sentinels

```
1  // This program calculates the total number of points a
2  // soccer team has earned over a series of games. The user
3  // enters a series of point values, then -1 when finished.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int game = 1,    // Game counter
10        points,     // To hold a number of points
11        total = 0;  // Accumulator
12
13    cout << "Enter the number of points your team has earned\n";
14    cout << "so far in the season, then enter -1 when finished.\n\n";
15    cout << "Enter the points for game " << game << ": ";
16    cin >> points;
17
18    while (points != -1)
19    {
20       total += points;
21       game++;
22       cout << "Enter the points for game " << game << ": ";
23       cin >> points;
24    }
25    cout << "\nThe total points are " << total << endl;
26    return 0;
27 }
```

Loops to Read from Files

```
1  // This program displays five numbers in a file.
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5
6  int main()
7  {
8     ifstream inputFile;  // File stream object
9     int number;          // To hold a value from the file
10    int count = 1;       // Loop counter, initialized with 1
11
12    inputFile.open("numbers.txt");   // Open the file.
13    if (!inputFile)                  // Test for errors.
14       cout << "Error opening file.\n";
15    else
16    {
17       while (count <= 5)
18       {
19          inputFile >> number;       // Read a number.
20          cout << number << endl;    // Display the number.
21          count++;                   // Increment the counter.
22       }
23       inputFile.close();            // Close the file.
24    }
25    return 0;
26 }
```

Loops to Read from Files

- The problem with this program is that it assumes the file has exactly 5 numbers
- Recall that opening a file sets the object to true or false, if it opened successfully
- A similar behavior exists when reading data from a file
  - inputFile >> number;
  - Returns true if there was a number to read false if there wasn't
  - We can utilize this to read any number of files

Loops to Read from Files

```
1  // This program displays all of the numbers in a file.
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5
6  int main()
7  {
8     ifstream inputFile;  // File stream object
9     int number;          // To hold a value from the file
10
11    inputFile.open("numbers.txt");     // Open the file.
12    if (!inputFile)                    // Test for errors.
13       cout << "Error opening file.\n";
14    else
15    {
16       while (inputFile >> number)     // Read a number
17       {
18          cout << number << endl;      // Display the number.
19       }
20       inputFile.close();              // Close the file.
21    }
22    return 0;
23 }
```

Checkpoint

15. Which variable below is the accumulator

```
int a, x, y = 0;
for (x = 0; x < 10; x++)
{
   cout << “Enter a number: “;
   cin >> a;
   y += a;
}
```

16. Why do you need to be careful when choosing a sentinel?

Checkpoint

17. How would you modify the soccer program to accept any negative number as a sentinel?
18. Assume a file named values.txt exists and contains a series of numbers, one per line in the file. Also assume that the program successfully executes the following statements to open the file:
    ifstream inputFile;
    inputFile.open("values.txt");

    Write a loop that reads and displays each number.

Nested Loops

- When you need to perform a very repetitious task, such as a clock
  - 60 seconds, 60 minutes, 24 hours

Nested Loops

```
cout << fixed << right;
cout.fill('0');
for (int hours = 0; hours < 24; hours++)
{
   for (int minutes = 0; minutes < 60; minutes++)
   {
      for(int seconds = 0; seconds < 60; seconds++)
      {
         cout << setw(2) << hours << ":"
         cout << setw(2) << minutes << ":";
         cout << setw(2) << seconds << ":";
      }
   }
}
```

Nested Loops

- Displays the output as follows
  00:00:00
  00:00:01
  …
  23:59:59
- The inner loop goes through all its iterations for each iteration of the outer loop
- The hours loop iterates 24 times
  The minutes loop iterates 1,440 times
  The seconds loop iterates 86,400 times
  - Multiply to determine the number: 60 * 60 * 24

Nested Loops

```
1  // This program averages test scores. It asks the user for the
2  // number of students and the number of test scores per student.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int numStudents,   // Number of students
9         numTests,      // Number of test per student
10        total;         // Accumulator for total scores
11    double average;    // Average test score
12
13    // Get the number of students.
14    cout << "This program averages test scores.\n";
15    cout << "For how many students do you have scores? ";
16    cin >> numStudents;
17
18    // Get the number of test scores per student.
19    cout << "How many test scores does each student have? ";
20    cin >> numTests;
```

Nested Loops

```
22      // Determine each student's average score.
23      for (int student = 1; student <= numStudents; student++)
24      {
25          total = 0;          // Initialize the accumulator.
26          for (int test = 1; test <= numTests; test++)
27          {
28              int score;
29              cout << "Enter score " << test << " for ";
30              cout << "student " << student << ": ";
31              cin >> score;
32              total += score;
33          }
34          average = total / numTests;
35          cout << "The average score for student " << student;
36          cout << " is " << average << ".\n\n";
37      }
38      return 0;
39  }
```

Breaking Out of a Loop

- You can use the break keyword to terminate a loop early
- Be cautious however, as it makes it difficult to debug
- ```
  int count = 0;
  while (count++ < 10)
  {
      cout << count << endl;
      if (count == 5)
          break;
  }
  ```
  ```

Breaking Out of a Loop

```
1  // This program raises the user's number to the powers
2  // of 0 through 10.
3  #include <iostream>
4  #include <cmath>
5  using namespace std;
6
7  int main()
8  {
9     int value;
10    char choice;
11
12    cout << "Enter a number: ";
13    cin >> value;
14    cout << "This program will raise " << value;
15    cout << " to the powers of 0 through 10.\n";
16    for (int count = 0; count <= 10; count++)
17    {
18       cout << value << " raised to the power of ";
19       cout << count << " is " << pow(value, count);
20       cout << "\nEnter Q to quit or any other key ";
21       cout << "to continue. ";
22       cin >> choice;
23       if (choice == 'Q' || choice == 'q')
24          break;
25    }
26    return 0;
27 }
```

Breaking Out of a Loop

- If using break in a nested loop, it only terminates that loop, not the outer one
  - for (int row = 0; row < 5; row++)
    {
      for (int star = 0; star < 20; star++)
      {
        cout << '*';
        if (star == 10)
          break;
      }
    }
  - Results in 5 rows of 10 stars

The continue Statement

- Stops executing the current iteration and continues to the next
- Be cautious again, debugging and what not
- int testVal = 0;
  while (testVal++ < 10)
  {
  if (testVal == 4)
  continue;
  cout << testVal << " ";
  }
  - 1 2 3 5 6 7 8 9 10

The continue Statement

```
1  // This program calculates the charges for video rentals.
2  // Every third video is free.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6
7  int main()
8  {
9     int videoCount = 1;   // Video counter
10    int numVideos;        // Number of videos rented
11    double total = 0.0;   // Accumulator
12    char current;         // Current release, Y or N
13
14    // Get the number of videos.
15    cout << "How many videos are being rented? ";
16    cin >> numVideos;
```

The continue Statement

```
18      // Determine the charges.
19      do
20      {
21          if ((videoCount % 3) == 0)
22          {
23              cout << "Video #" << videoCount << " is free!\n";
24              continue;  // Immediately start the next iteration
25          }
26          cout << "Is video #" << videoCount;
27          cout << " a current release? (Y/N) ";
28          cin >> current;
29          if (current == 'Y' || current == 'y')
30              total += 3.50;
31          else
32              total += 2.50;
33      } while (videoCount++ < numVideos);
34      cout << fixed << showpoint << setprecision(2);
35      cout << "The total is $" << total;
36      return 0;
37  }
```

Checkpoint

19. How many times does a nested loop iterate?

20. Why should you be cautious with continue and break statements?