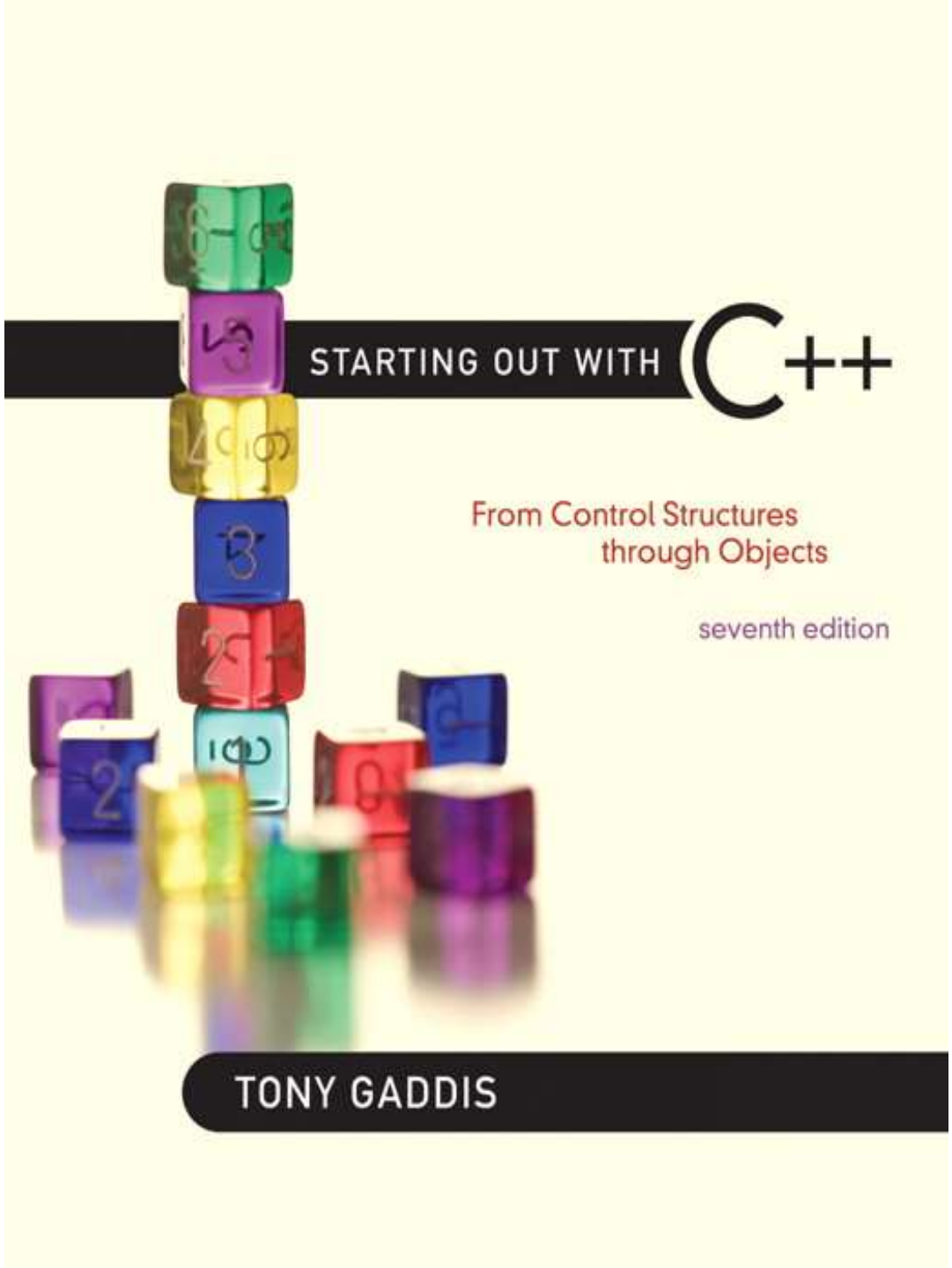
Chapter 4:

Making Decisions



Addison-Wesley
is an imprint of
PEARSON

“He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever” Chinese Proverb

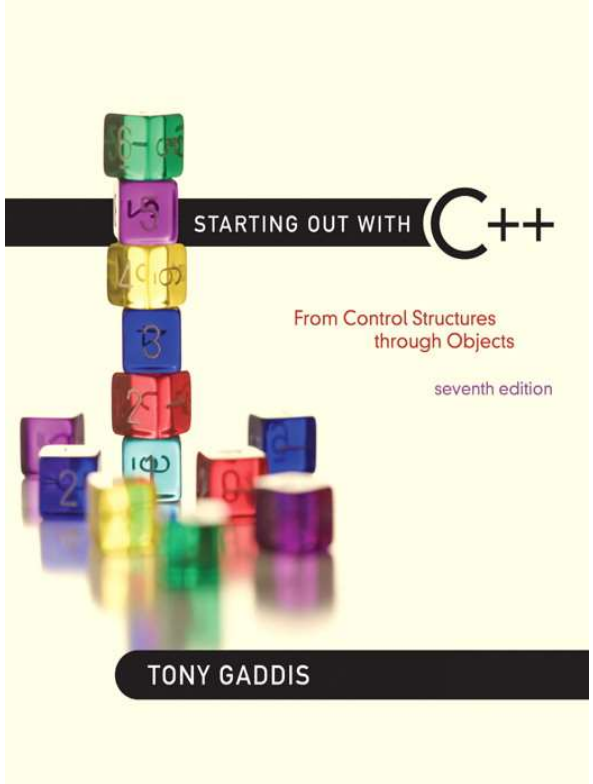
Control Structures

- Before writing a program
  - Have a thorough understanding of problem
  - Carefully plan your approach for solving it
- While writing a program
  - Know what “building blocks” are available
  - Use good programming principles

Control Structures

- Sequential execution
  - Statements executed in order
- Transfer of control
  - Next statement executed *not* next one in sequence
- 3 control structures
  - Sequence structure
    - Programs executed sequentially by default
  - Selection structures
    - **`if`**, **`if`**…**`else`**, **`if`**…**`else`**…**`if`**, **`switch`**
  - Repetition structures
    - **`while`**, **`do/while`**, **`for`**

Review



Relational Operators

Relational Operators

- **Relational operators allow you to compare *numeric (integer and floating point)* and *char* values and determine whether one is greater than, less than, equal to, or not equal to another.**

Relational Operators

- Used to compare numbers to determine relative order
- Operators:

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Relational Expressions

- Boolean expressions – `true` or `false`
- Examples:

  `12 > 5` is `true`

  `7 <= 5` is `false`

  if `x` is 10, then

  `x == 10` is `true`,

  `x != 8` is `true`, and

  `x == 8` is `false`

Relational Expressions

- Can be assigned to a variable:

```
result = x <= y;
```

- Assigns `0` for `false`, `1` for `true`
- Do not confuse `=` and `==`

Confusing Equality (==) and Assignment (=) Operators

- Common error
  - Does not typically cause syntax errors
- Aspects of problem
  - Expressions that have a value can be used for decision
    - Zero = false, nonzero = true
  - Assignment statements produce a value (the value to be assigned)

Confusing Equality (==) and Assignment (=) Operators

- Example

```
if ( payCode == 4 )
   cout << "You get a bonus!" << endl;
```

  – If paycode is 4, bonus given

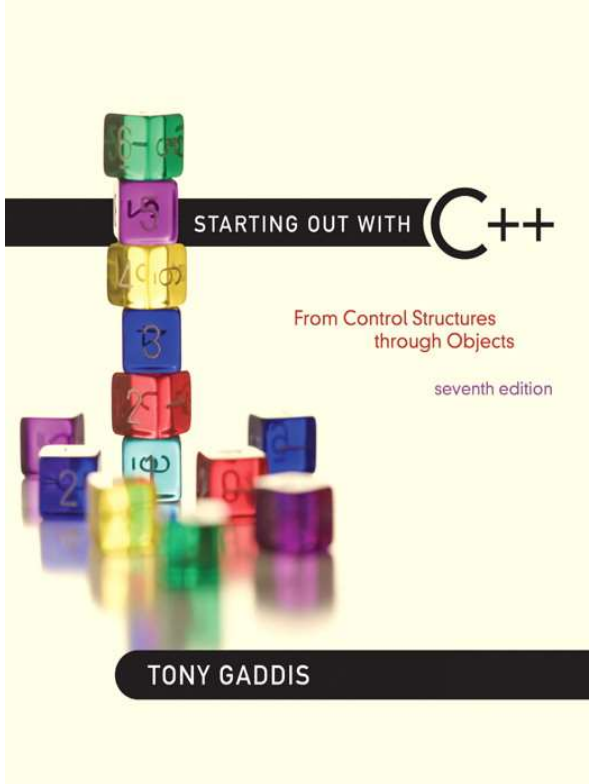
- If **==** was replaced with **=**

```
if ( payCode = 4 )
   cout << "You get a bonus!" << endl;
```

  – Paycode set to 4 (no matter what it was before)
  – Statement is true (since 4 is non-zero)
  – Bonus given in every case

Confusing Equality (==) and Assignment (=) Operators

- Lvalues
  - Expressions that can appear on left side of equation
  - Can be changed (I.e., variables)
    - `x = 4;`
- Rvalues
  - Only appear on right side of equation
  - Constants, such as numbers (i.e. cannot write `4 = x;`)
- Lvalues can be used as Rvalues, but not vice versa



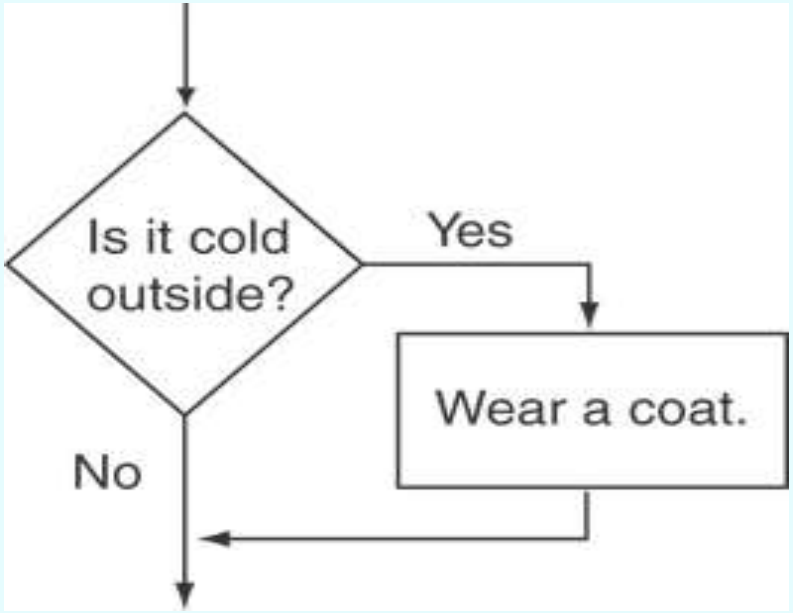
The `if` Statement

One-way or single selector

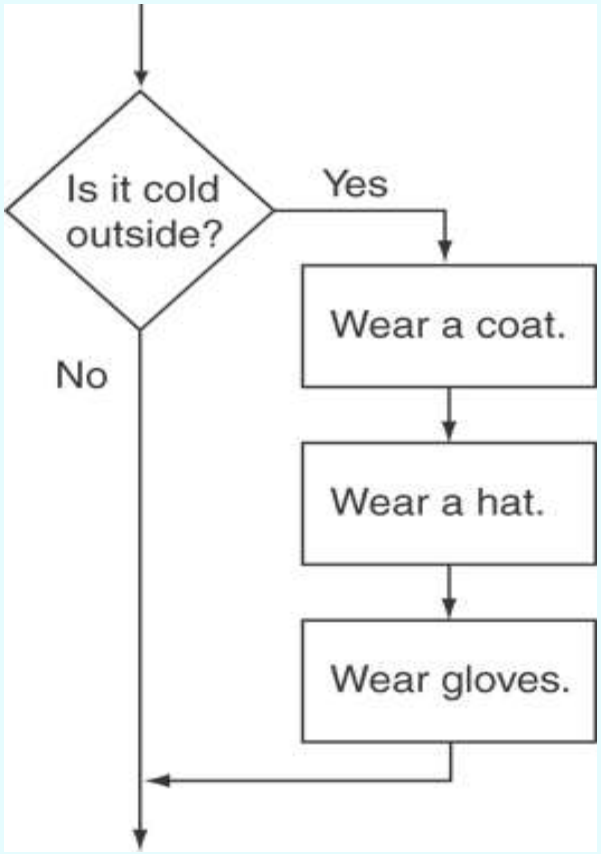
The `if` Statement

- Allows statements to be conditionally executed or skipped over
- Models the way we mentally evaluate situations:
  - "If it is raining, take an umbrella."
  - "If it is cold outside, wear a coat."

Flowchart for Evaluating a Decision



Flowchart for Evaluating a Decision



The `if` Statement

- General Format:

```
if (expression)
      statement;
```

The if Statement-What Happens

To evaluate:

```
if (expression)
    statement;
```

- If the *expression* is `true`, then *statement* is executed.
- If the *expression* is `false`, then *statement* is skipped.

`if` Statement in Program 4-2

Program 4-2

```
// This program averages three test scores
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int score1, score2, score3;  // To hold three test scores
   double average;              // To hold the average score

```

Continued…

`if` Statement in Program 4-2

Program 4-2 *(continued)*

```
    // Get the three test scores.
    cout << "Enter 3 test scores and I will average them: ";
    cin >> score1 >> score2 >> score3;

    // Calculate and display the average score.
    average = (score1 + score2 + score3) / 3.0;
    cout << fixed << showpoint << setprecision(1);
    cout << "Your average is " << average << endl;

    // If the average is greater than 95, congratulate the user.
    if (average > 95)
       cout << "Congratulations! That's a high score!\n";
    return 0;
}
```

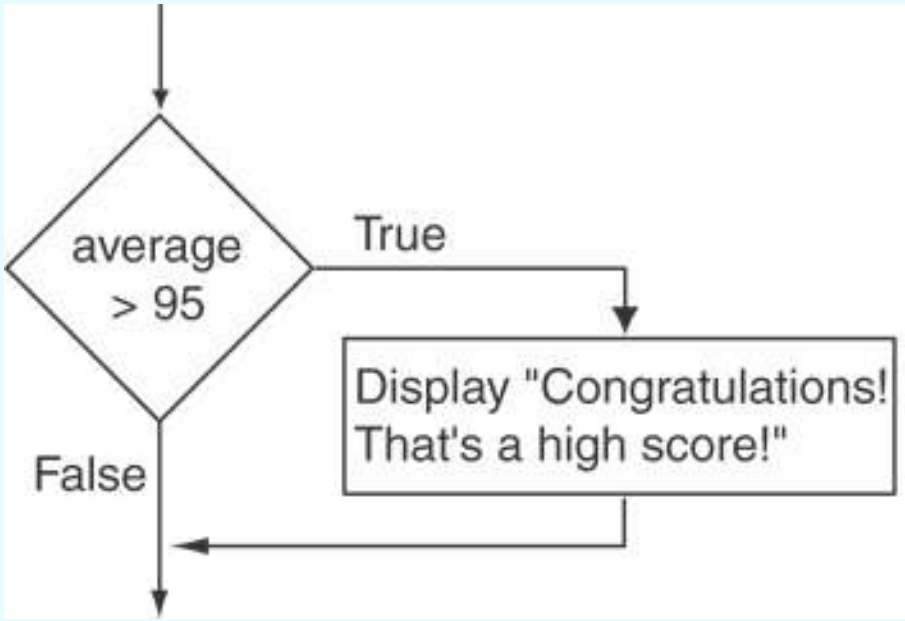
Program Output with Example Input Shown in Bold

```
Enter 3 test scores and I will average them: 80 90 70 [Enter]
Your average is 80.0
```

Program Output with Other Example Input Shown in Bold

```
Enter 3 test scores and I will average them: 100 100 100 [Enter]
Your average is 100.0
Congratulations! That's a high score!
```

Flowchart for Program 4-2 Lines 21 and 22



`if` Statement Notes

- Do not place `;` after `(`*`expression`*`)`
- Place *`statement`*`;` on a separate line after `(`*`expression`*`)`, indented:

```
if (score > 90)
    grade = 'A';
```

- Be careful testing `float`s and `double`s for equality
- `0` is `false`; any other value is `true`

```
1   // code5.cpp
2   // Using if statements, relational
3   // operators, and equality operators.
4   #include <iostream>
5
6   using std::cout;  // program uses cout
7   using std::cin;   // program uses cin
8   using std::endl;  // program uses endl
9
10  // function main begins progr
11  int main()
12  {
13     int num1;  // first n
14     int num2;  // second
15
16     cout << "Enter two integer
17          << "the relationships
18     cin >> num1 >> num2;    //
19
20     if ( num1 == num2 )
21        cout << num1 << " is eq
22
23     if ( num1 != num2 )
24        cout << num1 << " is not equal to " << num2 << endl;
25
```

using statements eliminate need for **std::** prefix.

Declare variables.

Can write **cout** and **cin** without **std::** prefix.

if structure compares values of **num1** and

If condition is true (i.e., , execute this

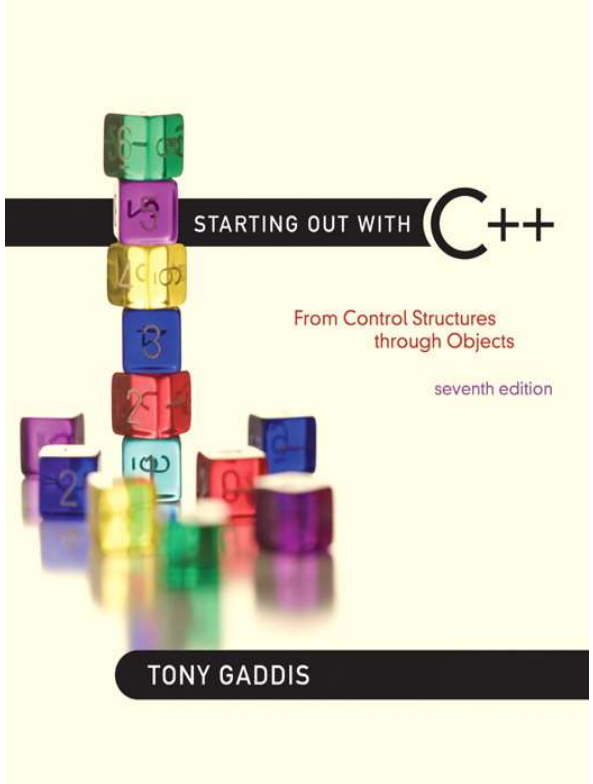
if structure compares values of **num1** and inequality.

If condition is true (i.e., values are not equal), execute this statement.

```
26     if ( num1 < num2 )
27        cout << num1 << " is less than " << num2 << endl;
28
29     if ( num1 > num2 )
30        cout << num1 << " is greater than " << num2 << endl;
31
32     if ( num1 <= num2 )
33        cout << num1 << " is less than or equal to "
34             << num2 << endl;
35
36     if ( num1 >= num2 )
37        cout << num1 << " is greater than or equal to "
38             << num2 << endl;
39
40     return 0;   // indicate that program ended successfully
41
42  } // end function main
```

Statements may be split over several lines.

```
1st RUN:
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12

2nd RUN:
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```



Expanding the `if` Statement

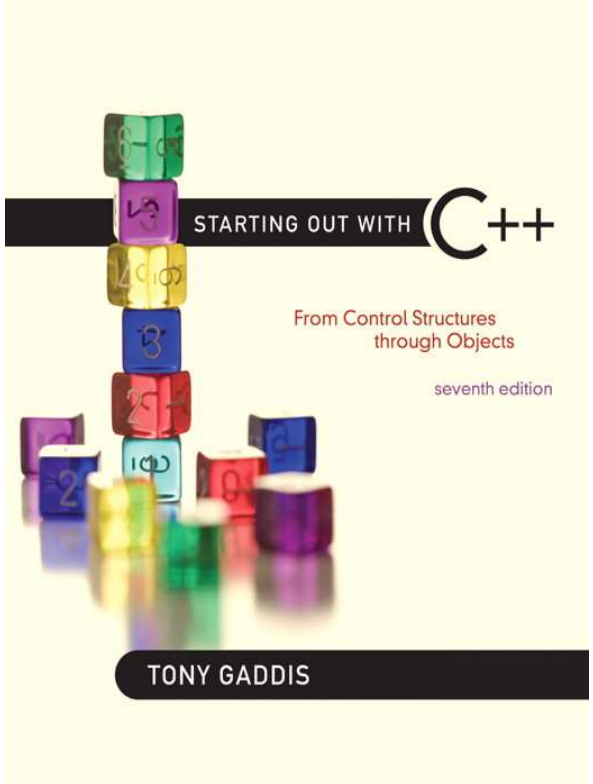
Expanding the `if` Statement

- To execute more than one statement as part of an `if` statement, enclose them in `{ }`:

```
if (score > 90)
{
    grade = 'A';
    cout << "Good Job!\n";
}
```

- `{ }` creates a <u>block</u> of code

4.2



The `if/else` Statement

Two way selector

The `if/else` statement

- Provides two possible paths of execution
- Performs one statement or block if the *`expression`* is true, otherwise performs another statement or block.

The `if/else` statement

- General Format:

```
if (expression)
        statement1;  // or block
else
        statement2;  // or block
```

`if/else`-What Happens

To evaluate:

```
if (expression)
   statement1;
else
   statement2;
```

- If the *`expression`* is `true`, then *`statement1`* is executed and *`statement2`* is skipped.
- If the *`expression`* is *`false`*, then *`statement1`* is skipped and *`statement2`* is executed.

The `if/else` statement and Modulus Operator in Program 4-8

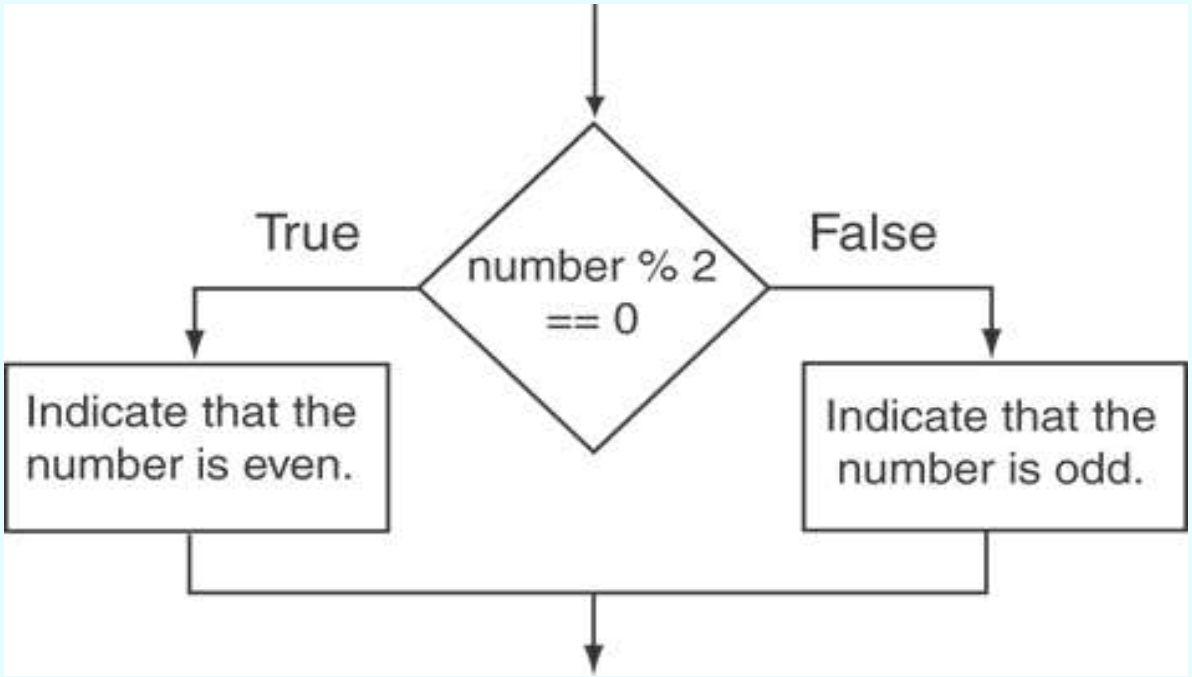
Program 4-8

```
// This program uses the modulus operator to determine
// if a number is odd or even. If the number is evenly divisible
// by 2, it is an even number. A remainder indicates it is odd.
#include <iostream>
using namespace std;

int main()
{
   int number;

   cout << "Enter an integer and I will tell you if it\n";
   cout << "is odd or even. ";
   cin >> number;
   if (number % 2 == 0)
      cout << number << " is even.\n";
   else
      cout << number << " is odd.\n";
   return 0;
}
```

Program Output with Example Input Shown in Bold

```
Enter an integer and I will tell you if it
is odd or even. 17 [Enter]
17 is odd.
```

Flowchart for Program 4-8 Lines 14 through 18



Testing the Divisor in Program 4-9

Program 4-9

```
// This program asks the user for two numbers, num1 and num2.
// num1 is divided by num2 and the result is displayed.
// Before the division operation, however, num2 is tested
// for the value 0. If it contains 0, the division does not
// take place.
#include <iostream>
using namespace std;

int main()
{
   double num1, num2, quotient;

```

Continued…

Testing the Divisor in Program 4-9

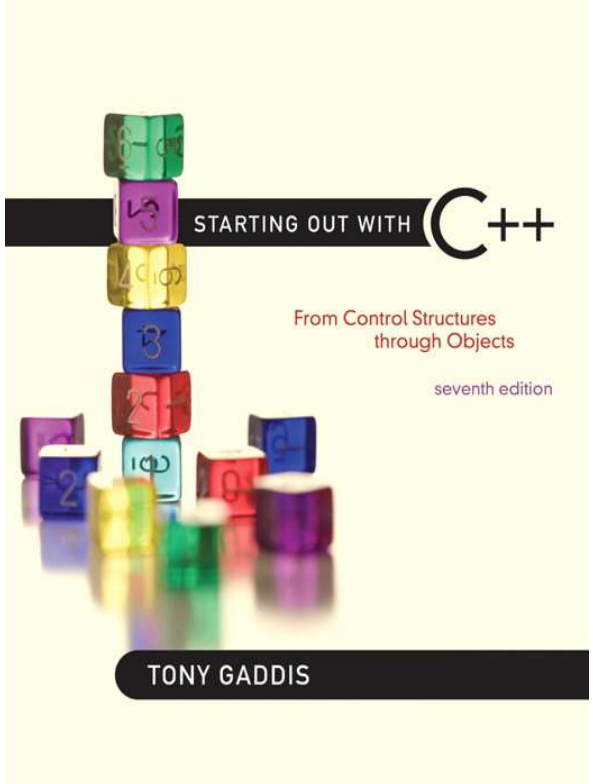
Program 4-9 *(continued)*

```
   // Get the first number.
   cout << "Enter a number: ";
   cin >> num1;

   // Get the second number.
   cout << "Enter another number: ";
   cin >> num2;

   // If num2 is not zero, perform the division.
   if (num2 == 0)
   {
      cout << "Division by zero is not possible.\n";
      cout << "Please run the program again and enter\n";
      cout << "a number other than zero.\n";
   }
   else
   {
      quotient = num1 / num2;
      cout << "The quotient of " << num1 << " divided by ";
      cout<< num2 << " is " << quotient << ".\n";
   }
   return 0;
}
```

Program Output with Example Input Shown in Bold

(When the user enters 0 for num2)

```
Enter a number: 10 [Enter]
Enter another number: 0 [Enter]
Division by zero is not possible.
Please run the program again and enter
a number other than zero.
```

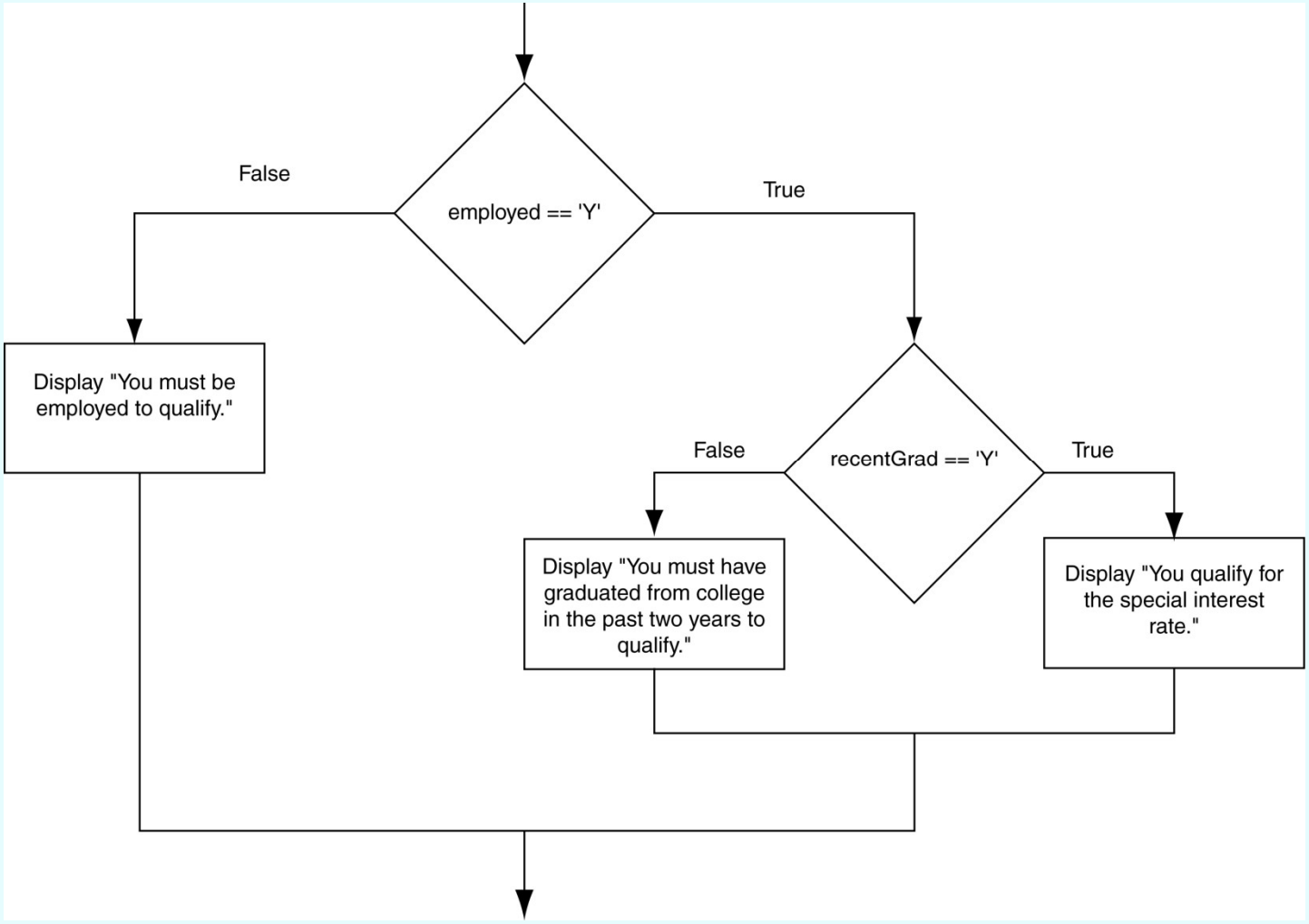


Nested `if` Statements

Nested `if` Statements

- An `if` statement that is nested inside another `if` statement
- Nested `if` statements can be used to test more than one condition

Flowchart for a Nested `if` Statement



Nested `if` Statements

- From Program 4-10

```
20      // Determine the user's loan qualifications.
21      if (employed == 'Y')
22      {
23         if (recentGrad == 'Y') //Nested if
24         {
25            cout << "You qualify for the special ";
26            cout << "interest rate.\n";
27         }
28      }
```

Nested `if` Statements

- Another example, from Program 4-1

```
20      // Determine the user's loan qualifications.
21      if (employed == 'Y')
22      {
23         if (recentGrad == 'Y') // Nested if
24         {
25            cout << "You qualify for the special ";
26            cout << "interest rate.\n";
27         }
28         else // Not a recent grad, but employed
29         {
30            cout << "You must have graduated from ";
31            cout << "college in the past two\n";
32            cout << "years to qualify.\n";
33         }
34      }
35      else // Not employed
36      {
37         cout << "You must be employed to qualify.\n";
38      }
```

Use Proper Indentation!

This `if` and `else` go together.

This `if` and `else` go together.

```
if (employed == 'Y')
{
   if (recentGrad == 'Y') // Nested if
   {
      cout << "You qualify for the special ";
      cout << "interest rate.\n";
   }
   else // Not a recent grad, but employed
   {
      cout << "You must have graduated from ";
      cout << "college in the past two\n";
      cout << "years to qualify.\n";
   }
}
else  // Not employed
{
   cout << "You must be employed to qualify.\n";
}
```