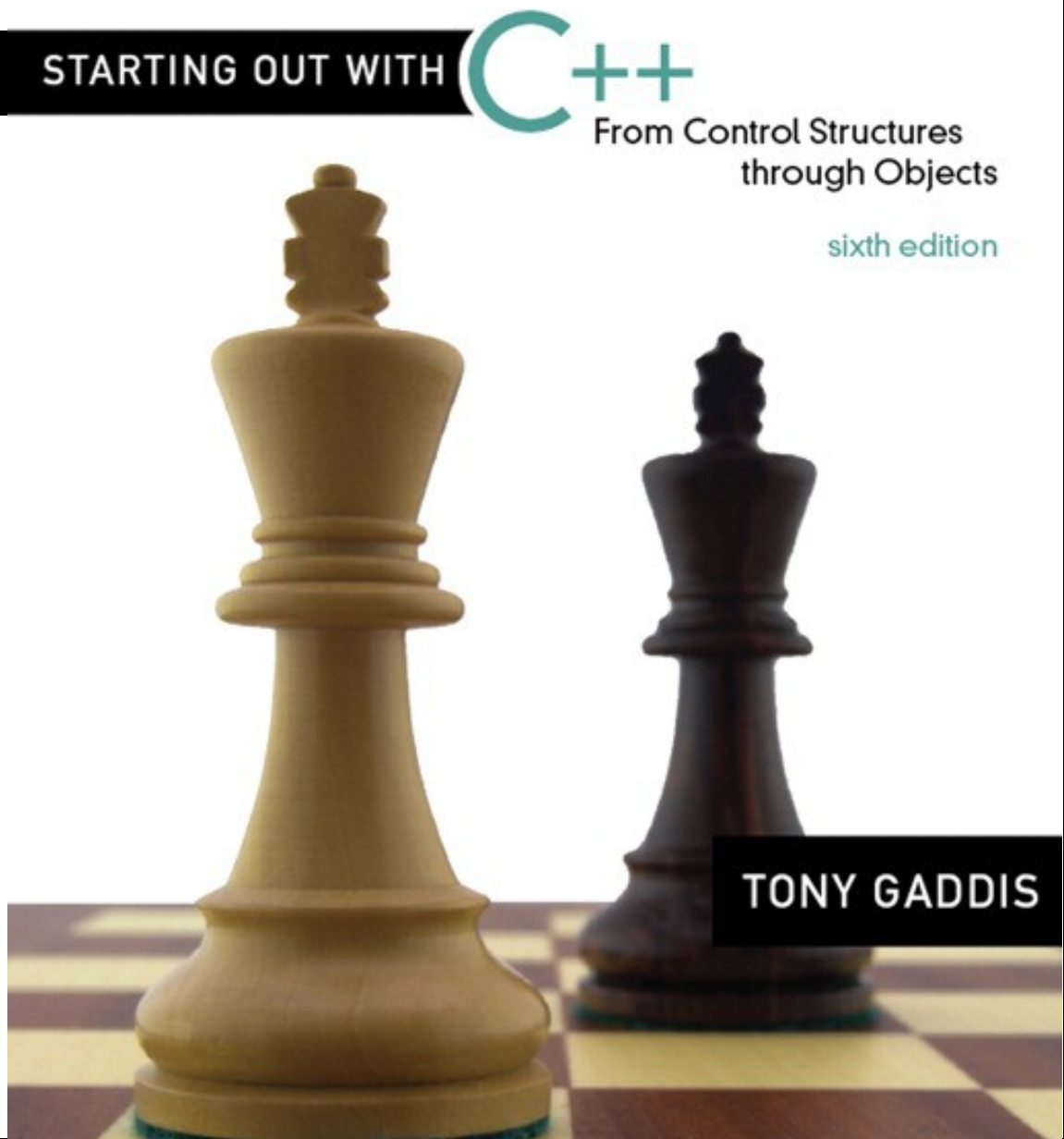
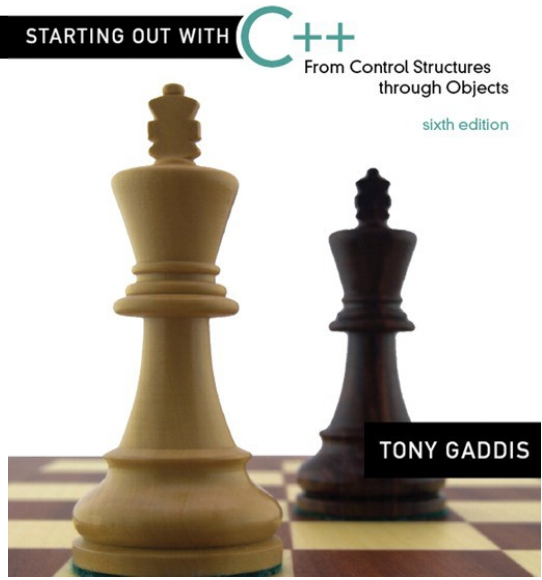


Chapter 4:

Making Decisions

Copyright © 2009 Pearson Education, Inc.
Publishing as Pearson Addison-Wesley



4.1

Relational Operators

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

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

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

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`

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

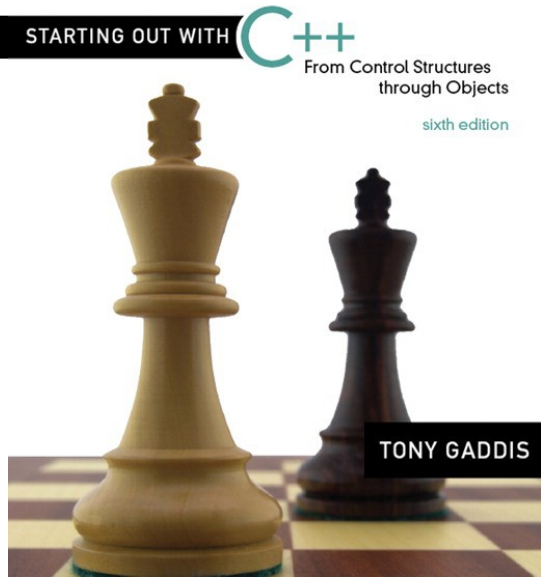
Relational Expressions

- Can be assigned to a variable:

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

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

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.2

The `if` Statement

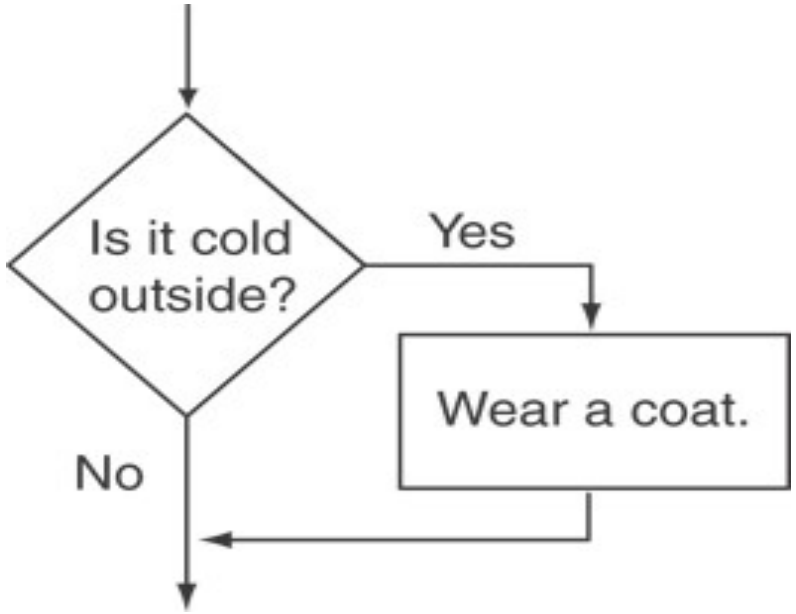
Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

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."

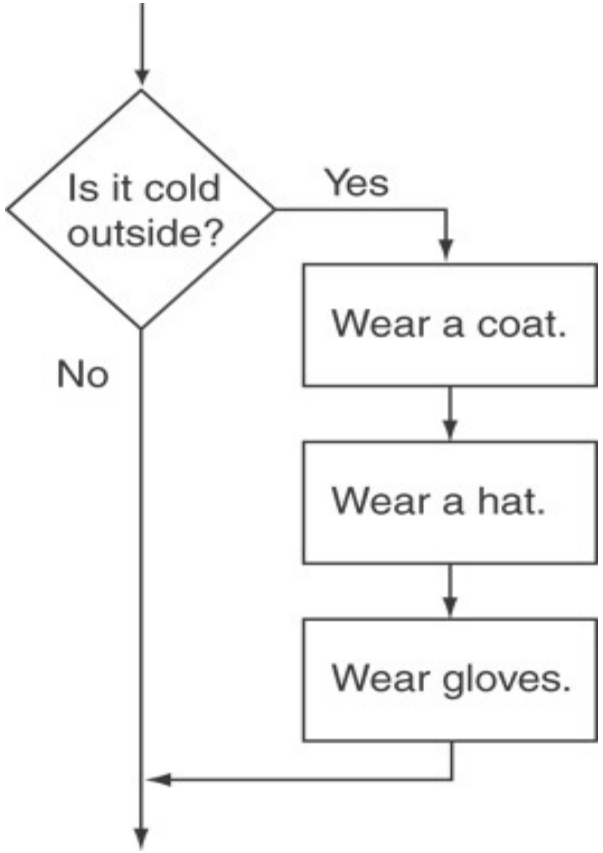
Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Flowchart for Evaluating a Decision



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Flowchart for Evaluating a Decision



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The `if` Statement

- General Format:

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

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

`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.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-2

```
1  // This program averages three test scores
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  int main()
7  {
8     int score1, score2, score3;  // To hold three test scores
9     double average;              // To hold the average score
10
```

(Program Continues)

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-2 *(continued)*

```
11      // Get the three test scores.
12      cout << "Enter 3 test scores and I will average them: ";
13      cin >> score1 >> score2 >> score3;
14
15      // Calculate and display the average score.
16      average = (score1 + score2 + score3) / 3.0;
17      cout << fixed << showpoint << setprecision(1);
18      cout << "Your average is " << average << endl;
19
20      // If the average is greater than 95, congratulate the user.
21      if (average > 95)
22         cout << "Congratulations! That's a high score!\n";
23      return 0;
24  }
```

Program Output with Example Input Shown in Bold

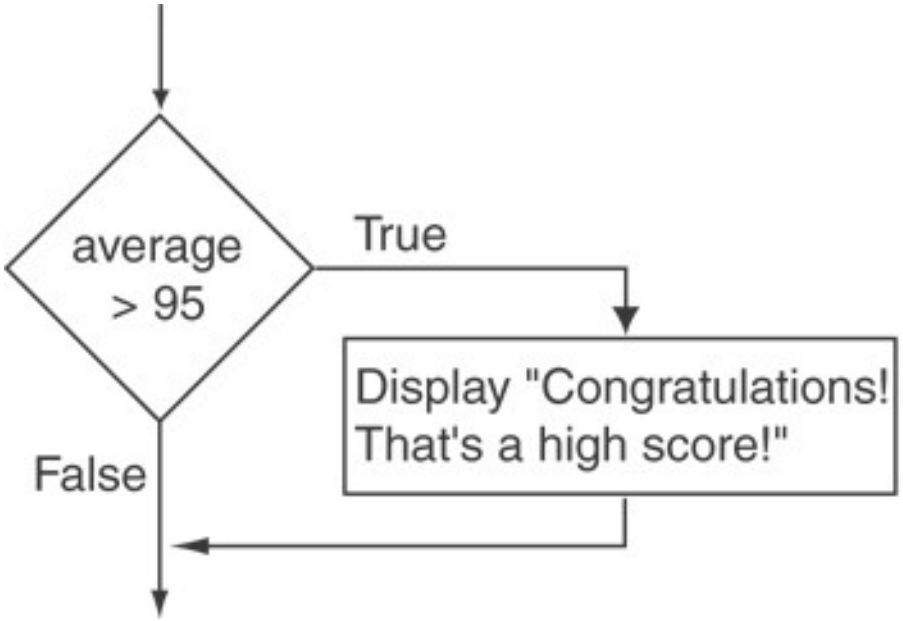
```
Enter 3 test scores and I will average them: 80 90 70 [Enter]
Your average is 80.0
```

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!
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Flowchart for Lines 21 and 22



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

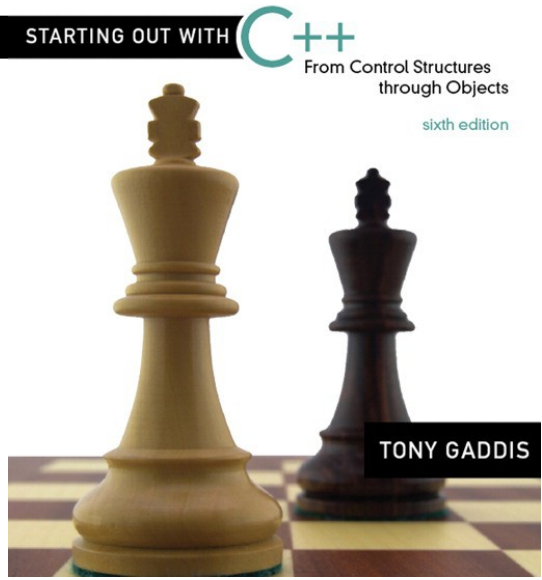
`if` statement notes

- 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`

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.3

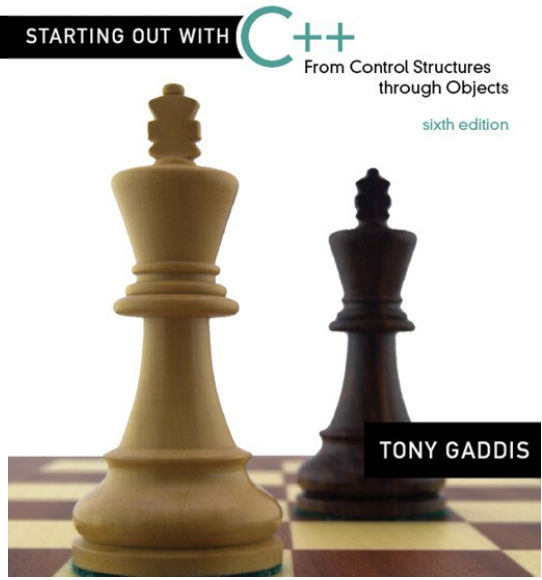
Flags

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Flags

- Variable that signals a condition
- Usually implemented as a `bool` variable
- As with other variables in functions, must be assigned an initial value before it is used

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.4

Expanding the `if` Statement

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

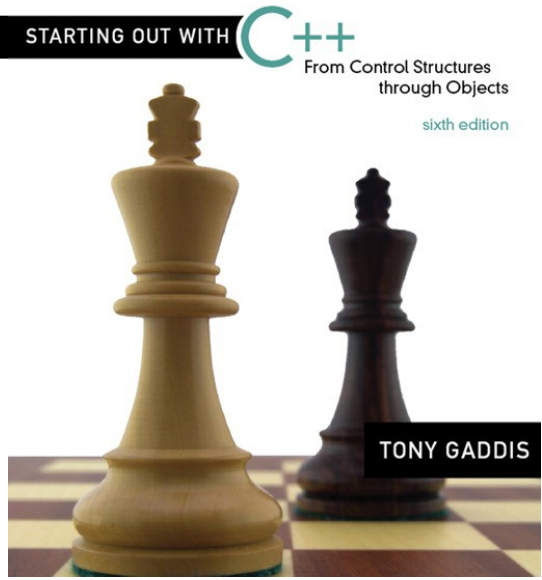
Expanding the `if` Statement

- 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 block of code

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.5

The `if/else` Statement

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

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.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The `if/else` Statement

- General Format:

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

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

`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.

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-8

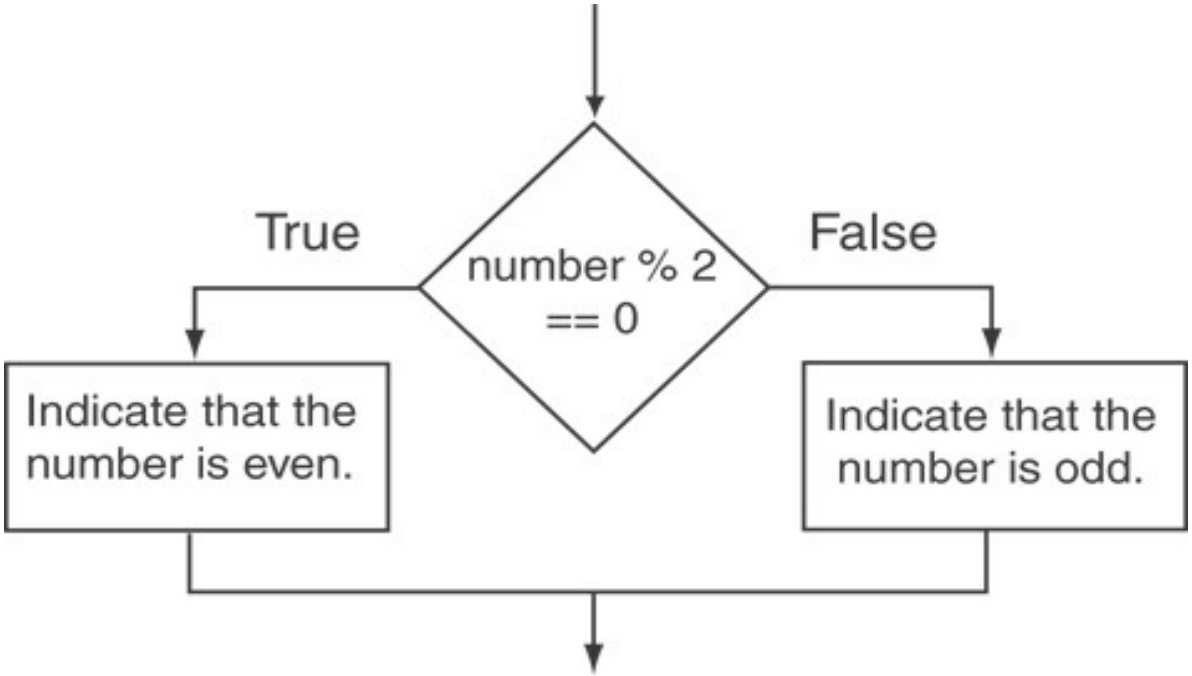
```
1  // This program uses the modulus operator to determine
2  // if a number is odd or even. If the number is evenly divisible
3  // by 2, it is an even number. A remainder indicates it is odd.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int number;
10
11    cout << "Enter an integer and I will tell you if it\n";
12    cout << "is odd or even. ";
13    cin >> number;
14    if (number % 2 == 0)
15       cout << number << " is even.\n";
16    else
17       cout << number << " is odd.\n";
18    return 0;
19 }
```

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.
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Flowchart for Lines 14 through 18



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-9

```
1  // This program asks the user for two numbers, num1 and num2.
2  // num1 is divided by num2 and the result is displayed.
3  // Before the division operation, however, num2 is tested
4  // for the value 0. If it contains 0, the division does not
5  // take place.
6  #include <iostream>
7  using namespace std;
8
9  int main()
10 {
11    double num1, num2, quotient;
12
```

(Program Continues)

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-9 *(continued)*

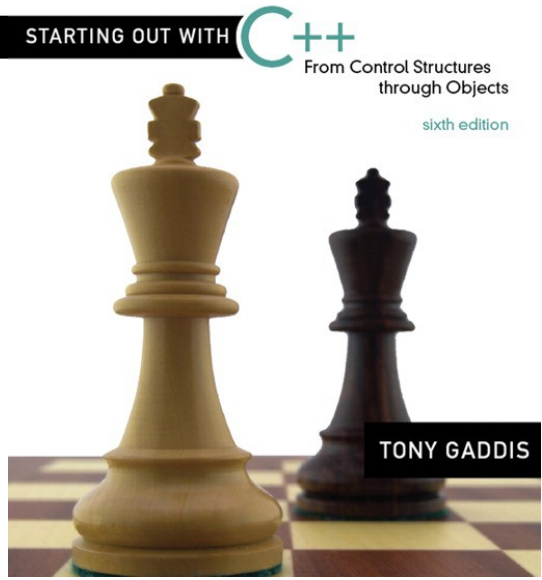
```
13      // Get the first number.
14      cout << "Enter a number: ";
15      cin >> num1;
16
17      // Get the second number.
18      cout << "Enter another number: ";
19      cin >> num2;
20
21      // If num2 is not zero, perform the division.
22      if (num2 == 0)
23      {
24         cout << "Division by zero is not possible.\n";
25         cout << "Please run the program again and enter\n";
26         cout << "a number other than zero.\n";
27      }
28      else
29      {
30         quotient = num1 / num2;
31         cout << "The quotient of " << num1 << " divided by ";
32         cout<< num2 << " is " << quotient << ".\n";
33      }
34      return 0;
35  }
```

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.
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.6

Nested `if` Statements

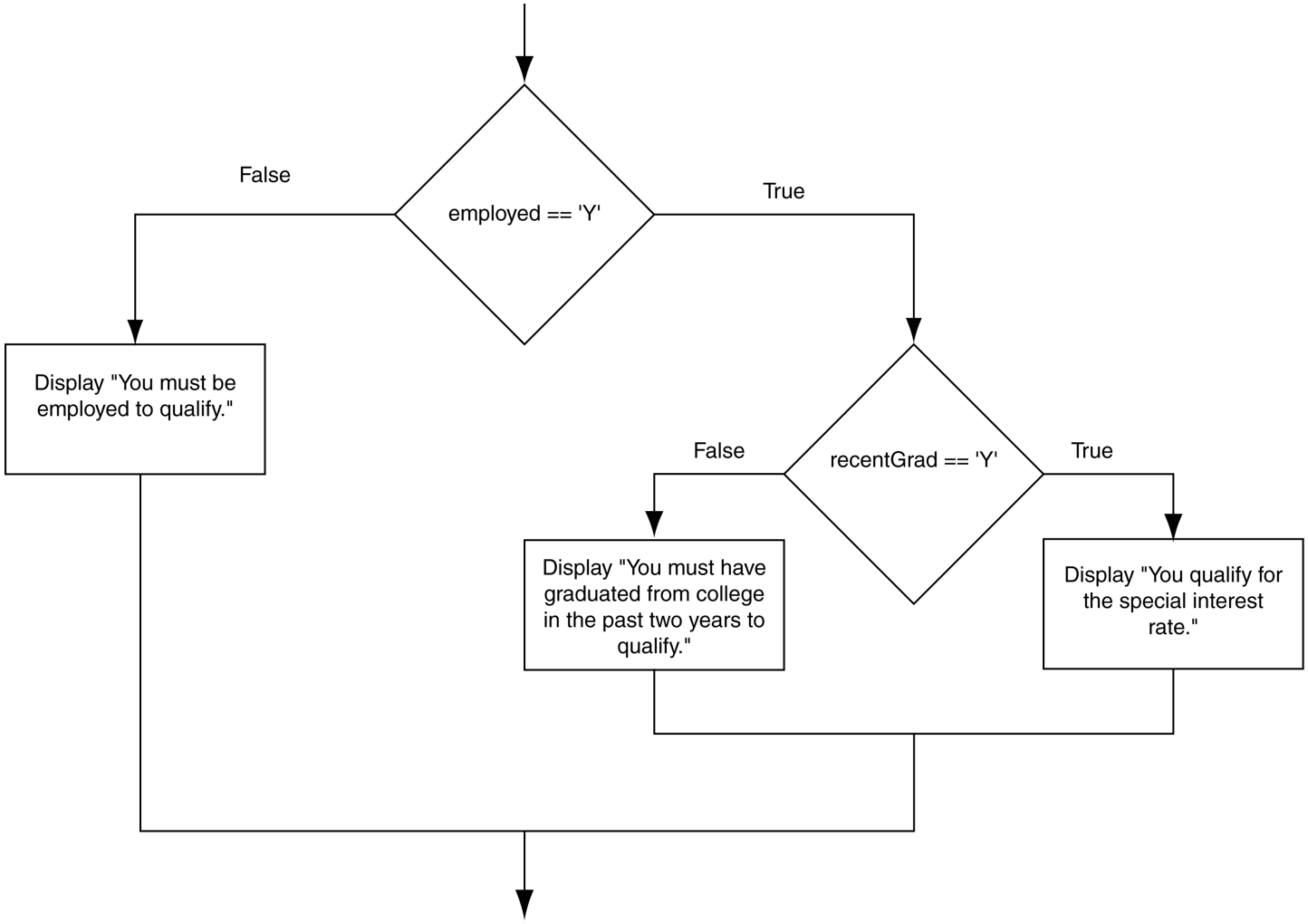
Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

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

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Flowchart for a Nested `if` Statement



Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

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      }
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Nested `if` Statements

- Another example, from Program 4-11

```
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      }
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

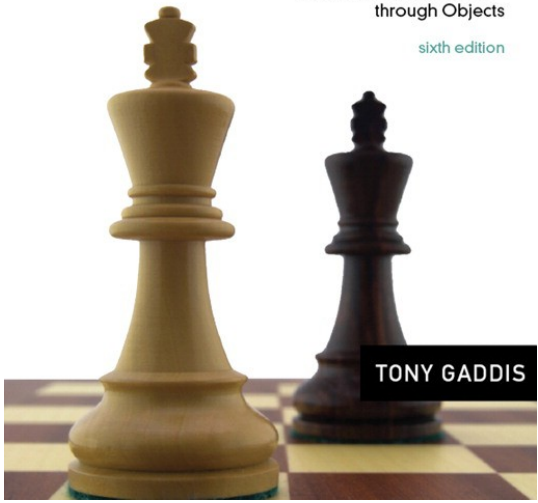
Use Proper Indentation!

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";
}
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.7

The `if/else if` Statement

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The `if/else if` Statement

- Tests a series of conditions until one is found to be true
- Often simpler than using nested `if/else` statements
- Can be used to model thought processes such as:

  "If it is raining, take an umbrella,
  else, if it is windy, take a hat,
  else, take sunglasses”

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

if/else if format

```
if (expression)
      statement1;  // or block
else if (expression)
      statement2;  // or block
    .
    . // other else ifs
    .
else if (expression)
      statementn;  // or block
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

From Program 4-13

```
15      // Determine the letter grade.
16      if (testScore < 60)
17         cout << "Your grade is F.\n";
18      else if (testScore < 70)
19         cout << "Your grade is D.\n";
20      else if (testScore < 80)
21         cout << "Your grade is C.\n";
22      else if (testScore < 90)
23         cout << "Your grade is B.\n";
24      else
25         cout << "Your grade is A.\n";
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Using a Trailing `else` to Catch Errors

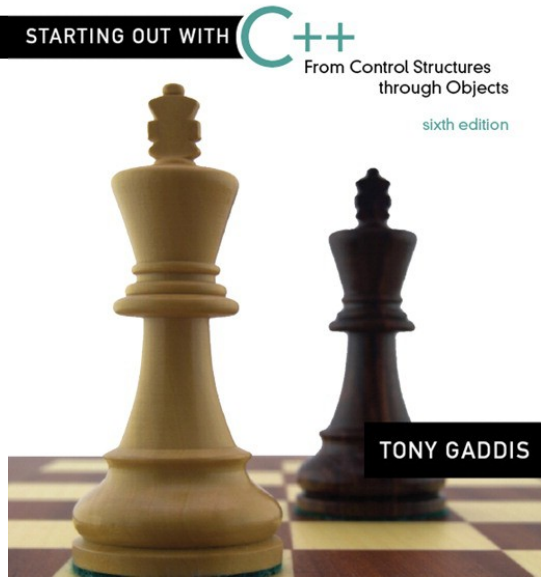
- The trailing else clause is optional, but is best used to catch errors

```
15    // Determine the letter grade.
16    if (testScore < 60)
17       cout << "Your grade is F.\n";
18    else if (testScore < 70)
19       cout << "Your grade is D.\n";
20    else if (testScore < 80)
21       cout << "Your grade is C.\n";
22    else if (testScore < 90)
23       cout << "Your grade is B.\n";
24    else if (testScore <= 100)
25       cout << "Your grade is A.\n";
26    else
27        cout << "We do not give scores higher than 100.\n";
```

From Program 4-14:

This trailing `else` catches invalid test scores

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.8

Menus

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Menus

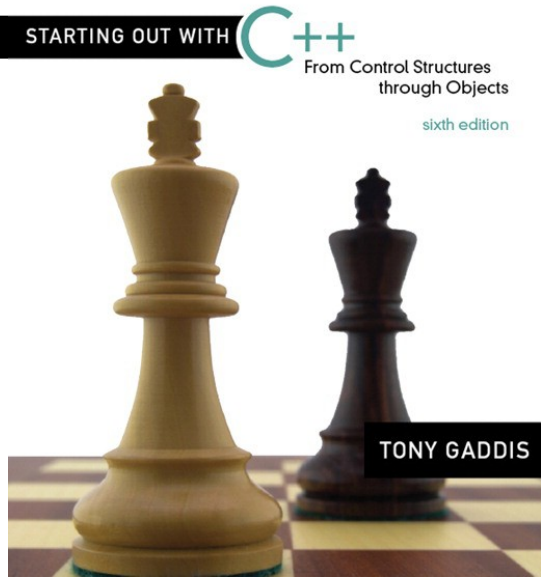
- Menu-driven program: program execution controlled by user selecting from a list of actions
- Menu: list of choices on the screen
- Menus can be implemented using `if/else if` statements

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Menu-driven program organization

- Display list of numbered or lettered choices for actions
- Prompt user to make selection
- Test user selection in *expression*
  - if a match, then execute code for action
  - if not, then go on to next *expression*

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.9

Logical Operators

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Logical Operators

- Used to create relational expressions from other relational expressions
- Operators, meaning, and explanation:

&&	AND	New relational expression is true if both expressions are true
||	OR	New relational expression is true if either expression is true
!	NOT	Reverses the value of an expression – true expression becomes false, and false becomes true

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Logical Operators - examples

```
int x = 12, y = 5, z = -4;
```

`(x > y) && (y > z)`	`true`
`(x > y) && (z > y)`	`false`
`(x <= z) || (y == z)`	`false`
`(x <= z) || (y != z)`	`true`
`!(x >= z)`	`false`

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The && Operator in Program 4-16

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

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The || Operator in Program 4-17

```
23      // Determine the user's loan qualifications.
24      if (income >= 35000 || years > 5)
25         cout << "You qualify.\n";
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The ! Operator in Program 4-18

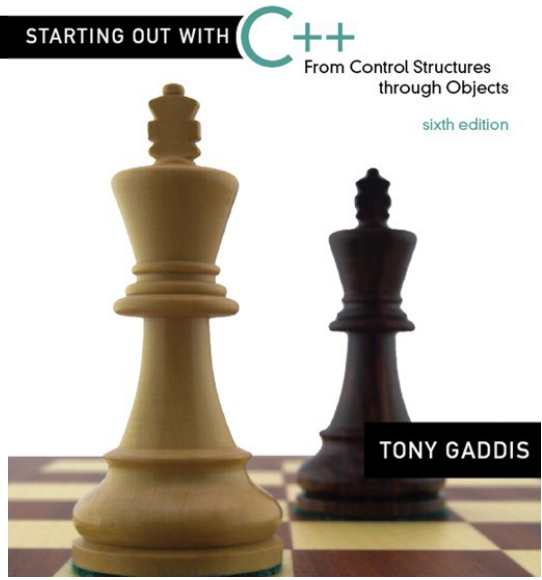
```
22      // Determine the user's loan qualifications.
23      if (!(income >= 35000 || years > 5))
24      {
25         cout << "You must earn at least $35,000 or have\n";
26         cout << "been employed for more than 5 years.\n";
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Logical Operators - notes

- ! has highest precedence, followed by &&, then ||
- If the value of an expression can be determined by evaluating just the sub-expression on left side of a logical operator, then the sub-expression on the right side will not be evaluated (*short circuit evaluation*)

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.10

Checking Numeric Ranges with Logical Operators

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Checking Numeric Ranges with Logical Operators

- Used to test to see if a value falls **inside** a range:

```
if (grade >= 0 && grade <= 100)
   cout << "Valid grade";
```

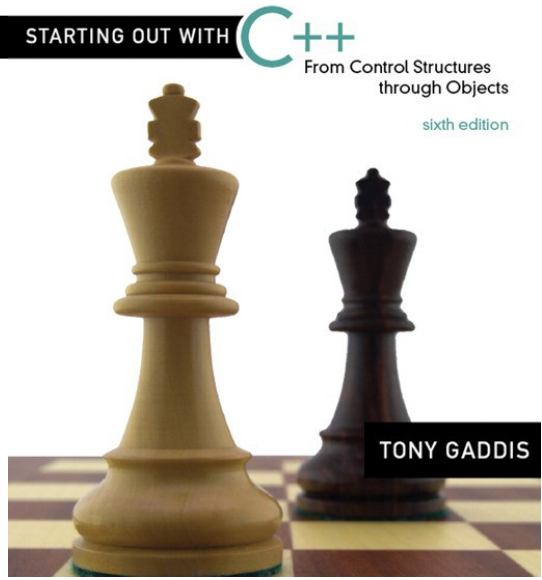
- Can also test to see if value falls **outside** of range:

```
if (grade <= 0 || grade >= 100)
   cout << "Invalid grade";
```

- Cannot use mathematical notation:

```
if (0 <= grade <= 100) //doesn't work!
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.11

Validating User Input

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Validating User Input

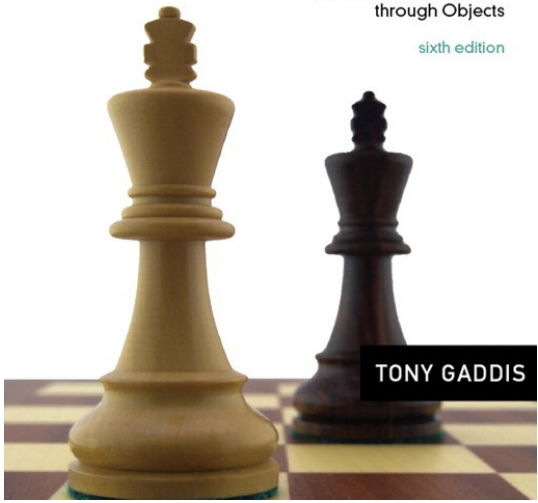
- Input validation: inspecting input data to determine whether it is acceptable
- Bad output will be produced from bad input
- Can perform various tests:
  - Range
  - Reasonableness
  - Valid menu choice
  - Divide by zero

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

From Program 4-19

```
11      // Get the numeric test score.
12      cout << "Enter your numeric test score and I will\n";
13      cout << "tell you the letter grade you earned: ";
14      cin >> testScore;
15
16      if (testScore < 0 || testScore > 100) //Input validation
17      {
18         // An invalid score was entered.
19         cout << testScore << " is an invalid score.\n";
20         cout << "Run the program again and enter a value\n";
21         cout << "in the range of 0 to 100.\n";
22      }
23      else
24      {
25         // Determine the letter grade.
26         if (testScore < 60)
27            grade = 'F';
28         else if (testScore < 70)
29            grade = 'D';
30         else if (testScore < 80)
31            grade = 'C';
32         else if (testScore < 90)
33            grade = 'B';
34         else if (testScore <= 100)
35            grade = 'A';
36
37         // Display the letter grade.
38         cout << "Your grade is " << grade << endl;
39      }
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.12

More About Variable Definitions and Scope

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

More About Variable Definitions and Scope

- Scope of a variable is the block in which it is defined, from the point of definition to the end of the block
- Usually defined at beginning of function
- May be defined close to first use

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

From Program 4-21

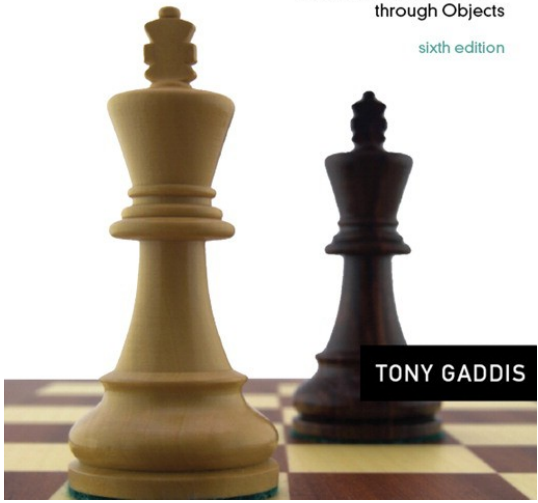
```
5  int main()
6  {
7     // Get the annual income.
8     cout << "What is your annual income? ";
9     double income;    //variable definition
10    cin >> income;
11
12    if (income >= 35000)
13    {
14       // Get the number of years at the current job.
15       cout << "How many years have you worked at "
16            << "your current job? ";
17       int years;     //variable definition
18       cin >> years;
19
20       if (years > 5)
21          cout << "You qualify.\n";
22       else
23       {
24          cout << "You must have been employed for\n";
25          cout << "more than 5 years to qualify.\n";
26       }
27    }
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Still More About Variable Definitions and Scope

- Variables defined inside { } have local or block scope
- When inside a block within another block, can define variables with the same name as in the outer block.
  - When in inner block, outer definition is not available
  - Not a good idea

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.13

Comparing Strings

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Comparing Strings

- You cannot use relational operators with C-strings
- Must use the `strcmp` function to compare C-strings
- `strcmp` compares the ASCII codes of the characters in the C-strings. Comparison is character-by-character

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Comparing Strings

The expression

```
strcmp(str1, str2)
```

compares the strings `str1` and `str2`

- It returns `0` if the strings are the same
- It returns a negative number if `str1 < str2`
- It returns a positive number if `str1 > str2`

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-24

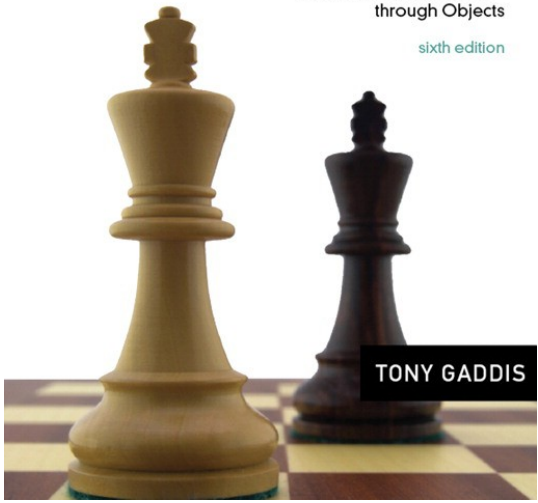
```
1  // This program correctly tests two C-strings for equality
2  // with the strcmp function.
3  #include <iostream>
4  #include <cstring>
5  using namespace std;
6
7  int main()
8  {
9     const int SIZE = 40;
10    char firstString[SIZE], secondString[SIZE];
11
12    // Get two strings
13    cout << "Enter a string: ";
14    cin.getline(firstString, SIZE);
15    cout << "Enter another string: ";
16    cin.getline(secondString, SIZE);
17
18    // Compare them with strcmp.
19    if (strcmp(firstString, secondString) == 0)
20       cout << "You entered the same string twice.\n";
21    else
22       cout << "The strings are not the same.\n";
23    return 0;
24 }
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program Output with Example Input Shown in Bold

```
Enter a string: Alfonso [Enter]
Enter another string: Alfonso [Enter]
You entered the same string twice.
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.14

The Conditional Operator

PEARSON Addison Wesley

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The Conditional Operator

- Can use to create short `if/else` statements
- Format: `expr ? expr : expr;`

```
x<0 ? y=10 : z=20;
```

First Expression: Expression to be tested

2nd Expression: Executes if first expression is true

3rd Expression: Executes if the first expression is false

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The Conditional Operator

- The value of a conditional expression is
  - The value of the second expression if the first expression is true
  - The value of the third expression if the first expression is false
- Parentheses （） may be needed in an expression due to precedence of conditional operator

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-27

```
1  // This program calculates a consultant's charges at $50
2  // per hour, for a minimum of 5 hours. The ?: operator
3  // adjusts hours to 5 if less than 5 hours were worked.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  int main()
9  {
10    const double PAY_RATE = 50.0;
11    double hours, charges;
12
13    cout << "How many hours were worked? ";
14    cin >> hours;
15    hours = hours < 5 ? 5 : hours; //conditional operator
16    charges = PAY_RATE * hours;
17    cout << fixed << showpoint << setprecision(2);
18    cout << "The charges are $" << charges << endl;
19    return 0;
20 }
```

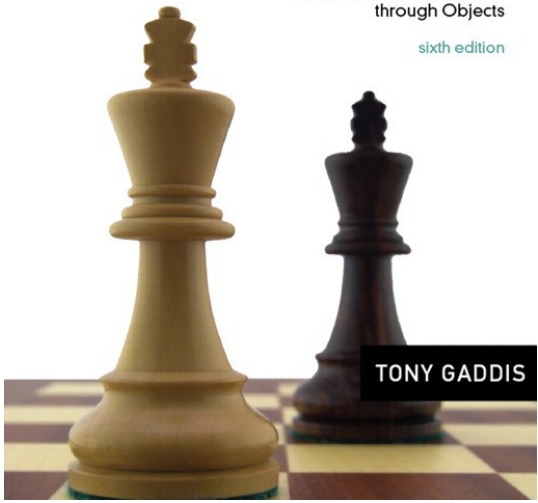
Program Output with Example Input Shown in Bold

```
How many hours were worked? 10 [Enter]
The charges are $500.00
```

Program Output with Example Input Shown in Bold

```
How many hours were worked? 2 [Enter]
The charges are $250.00
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.15

The `switch` Statement

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The `switch` Statement

- Used to select among statements from several alternatives
- In some cases, can be used instead of `if/else if` statements

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

`switch` statement format

```
switch (expression) //integer
{
  case exp1: statement1;
  case exp2: statement2;
  ...
  case expn: statementn;
  default:   statementn+1;
}
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-28

```
1  // The switch statement in this program tells the user something
2  // he or she already knows: what they just entered!
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char choice;
9
10    cout << "Enter A, B, or C: ";
11    cin >> choice;
12    switch (choice)
13    {
14       case 'A': cout << "You entered A.\n";
15                 break;
16       case 'B': cout << "You entered B.\n";
17                 break;
18       case 'C': cout << "You entered C.\n";
19                 break;
20       default:  cout << "You did not enter A, B, or C!\n";
21    }
22    return 0;
23 }
```

Program Output with Example Input Shown in Bold

```
Enter A, B, or C: B [Enter]
You entered B.
```

Program Output with Example Input Shown in Bold

```
Enter A, B, or C: F [Enter]
You did not enter A, B, or C!
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

`switch` statement requirements

1) *`expression`* must be an integer variable or an expression that evaluates to an integer value
2) *`exp`**`1`*** through *`exp`**`n`*** must be constant integer expressions or literals, and must be unique in the `switch` statement
3) `default` is optional but recommended

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

`switch` statement – how it works

1) *`expression`* is evaluated
2) The value of *`expression`* is compared against *`exp`**`1`* through *`exp`**`n`*.
3) If *`expression`* matches value *`exp`**`i`*, the program branches to the statement following *`exp`**`i`* and continues to the end of the `switch`
4) If no matching value is found, the program branches to the statement after `default:`

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

`break` statement

- Used to exit a `switch` statement
- If it is left out, the program "falls through" the remaining statements in the `switch` statement

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-30

```
1  // This program is carefully constructed to use the "fallthrough"
2  // feature of the switch statement.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int modelNum;  // Model number
9
10    // Get a model number from the user.
11    cout << "Our TVs come in three models:\n";
12    cout << "The 100, 200, and 300. Which do you want? ";
13    cin >> modelNum;
14
15    // Display the model's features.
16    cout << "That model has the following features:\n";
17    switch (modelNum)
18    {
19       case 300: cout << "\tPicture-in-a-picture.\n";
20       case 200: cout << "\tStereo sound.\n";
21       case 100: cout << "\tRemote control.\n";
22                 break;
23       default:  cout << "You can only choose the 100,";
24                 cout << "200, or 300.\n";
25    }
26    return 0;
27 }
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program Output with Example Input Shown in Bold

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 100 [Enter]
That model has the following features:
   Remote control.
```

Program Output with Example Input Shown in Bold

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 200 [Enter]
That model has the following features:
   Stereo sound.
   Remote control.
```

Program Output with Example Input Shown in Bold

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 300 [Enter]
That model has the following features:
   Picture-in-a-picture.
   Stereo sound.
   Remote control.
```

Program Output with Example Input Shown in Bold

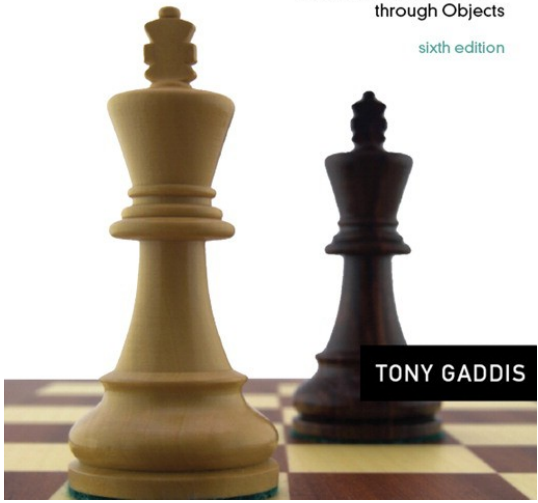
```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 500 [Enter]
That model has the following features:
You can only choose the 100, 200, or 300.
```

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Using `switch` with a menu

- `switch` statement is a natural choice for menu-driven program:
  - display the menu
  - then, get the user's menu selection
  - use user input as `expression` in `switch` statement
  - use menu choices as *`expr`* in `case` statements

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



4.16

Testing for File Open Errors

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Testing for File Open Errors

- Can test a file stream object to detect if an open operation failed:

```
infile.open("test.txt");
if (!infile)
{
  cout << "File open failure!";
}
```

- Can also use the `fail` member function

Copyright © 2009 Pearson Education, Inc. Publishing as Pearson Addison-Wesley