

## 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

Addison-Wesley  
is an imprint of  
PEARSON  
Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 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

Addison-Wesley  
is an imprint of  
PEARSON  
Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Relational Expressions

- Can be assigned to a variable:
 

```
result = x <= y;
```
- Assigns 0 for false, 1 for true
- Do not confuse `=` and `==`

Addison-Wesley  
is an imprint of  
PEARSON  
Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

**4.2**  
**The if Statement**

Addison-Wesley  
is an imprint of  
PEARSON  
Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

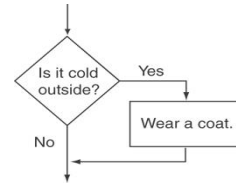
## The `if` Statement

- 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."

Addison-Wesley  
3 as imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## Flowchart for Evaluating a Decision



Addison-Wesley  
3 as imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## Flowchart for Evaluating a Decision



Addison-Wesley  
3 as imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## The `if` Statement

- General Format:

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

Addison-Wesley  
3 as imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## 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.

Addison-Wesley  
3 as imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## `if` Statement in Program 4-2

### 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 }
```

Continued...

Addison-Wesley  
3 as imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## if Statement in Program 4-2

### 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

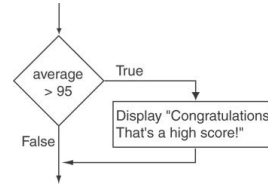
Enter 3 test scores and I will average them: **100 100 100** [Enter]  
Your average is 100.0  
Congratulations! That's a high score!

Addison-Wesley  
an imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Flowchart for Program 4-2 Lines 21 and 22



Addison-Wesley  
an imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## if Statement Notes

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

```

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

```
- ❸ Be careful testing floats and doubles for equality
- ❹ `0` is false; any other value is true

Addison-Wesley  
an imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 4.3

### Expanding the if Statement



Addison-Wesley  
an imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Expanding the if Statement

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

```

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

```

- ❷ `{ }` creates a block of code

Addison-Wesley  
an imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 4.4

### The if/else Statement



Addison-Wesley  
an imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 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.

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The if/else statement

- General Format:

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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 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.

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

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

### 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's\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

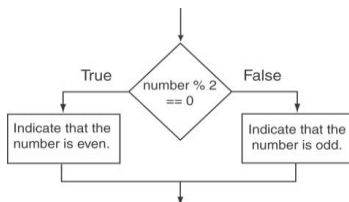
Enter an integer and I will tell you if it is odd or even. **17** [Enter]  
17 is odd.

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Flowchart for Program 4-8 Lines 14 through 18



Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Testing the Divisor in Program 4-9

### 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 }
```

Continued...

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Testing the Divisor in Program 4-9

```

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.

```

Addison-Wesley

Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.



# 4.5

## Nested if Statements

Addison-Wesley

Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 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

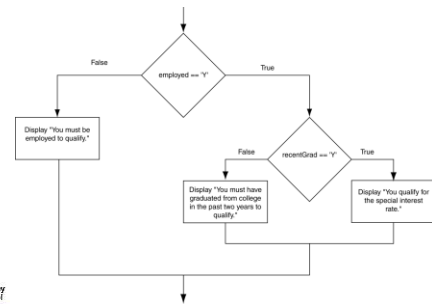
Addison-Wesley

Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Flowchart for a Nested if Statement



Addison-Wesley

Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 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 }

```

Addison-Wesley

Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 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 }

```

Addison-Wesley

Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Use Proper Indentation!

```

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";
}

```

This if and else go together.

This if and else go together.

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.



## 4.6

### The if/else if Statement

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## 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"

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## if/else if Format

```

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

```

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## The if/else if Statement in Program 4-13

```

21 // Determine the letter grade.
22 if (testScore >= A_SCORE)
23     cout << "Your grade is A.\n";
24 else if (testScore >= B_SCORE)
25     cout << "Your grade is B.\n";
26 else if (testScore >= C_SCORE)
27     cout << "Your grade is C.\n";
28 else if (testScore >= D_SCORE)
29     cout << "Your grade is D.\n";
30 else
31     cout << "Your grade is F.\n";

```

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## Using a Trailing else to Catch Errors in Program 4-14

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

```


21 // Determine the letter grade.
22 if (testScore >= A_SCORE)
23     cout << "Your grade is A.\n";
24 else if (testScore >= B_SCORE)
25     cout << "Your grade is B.\n";
26 else if (testScore >= C_SCORE)
27     cout << "Your grade is C.\n";
28 else if (testScore >= D_SCORE)
29     cout << "Your grade is D.\n";
30 else if (testScore >= 0)
31     cout << "Your grade is F.\n";
32 else
33     cout << "Invalid test score.\n";

```

This trailing else catches invalid test scores

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.



# 4.7

## Flags

Addison-Wesley  
is an imprint of  
**PEARSON**


Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Flags

- ➊ Variable that signals a condition
- ➋ Usually implemented as a `bool` variable
- ➌ Can also be an integer
  - ➊ The value `0` is considered `false`
  - ➋ Any nonzero value is considered `true`
- ➍ As with other variables in functions, must be assigned an initial value before it is used

Addison-Wesley  
is an imprint of  
**PEARSON**

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.



# 4.8

## Logical Operators

Addison-Wesley  
is an imprint of  
**PEARSON**

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Logical Operators

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

<code>&amp;&amp;</code>	AND	New relational expression is true if both expressions are true
<code>  </code>	OR	New relational expression is true if either expression is true
<code>!</code>	NOT	Reverses the value of an expression – true expression becomes false, and false becomes true

Addison-Wesley  
is an imprint of  
**PEARSON**

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Logical Operators-Examples

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

<code>(x &gt; y) &amp;&amp; (y &gt; z)</code>	true
<code>(x &gt; y) &amp;&amp; (z &gt; y)</code>	false
<code>(x &lt;= z)    (y == z)</code>	false
<code>(x &lt;= z)    (y != z)</code>	true
<code>!(x &gt;= z)</code>	false

Addison-Wesley  
is an imprint of  
**PEARSON**

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The logical `&&` operator in Program 4-15

```

21 // Determine the user's loan qualifications.
22 if (employed == 'Y' && recentGrad == 'Y')
23 {
24     cout << "You qualify for the special "
25         << "interest rate.\n";
26 }
27 else
28 {
29     cout << "You must be employed and have\n"
30         << "graduated from college in the\n"
31         << "past two years to qualify.\n";
32 }

```

Addison-Wesley  
is an imprint of  
**PEARSON**

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The logical || Operator in Program 4-16

```

23 // Determine the user's loan qualifications.
24 if (income >= MIN_INCOME || years > MIN_YEARS)
25     cout << "You qualify.\n";
26 else
27 {
28     cout << "You must earn at least $"
29         << MIN_INCOME << " or have been "
30         << "employed more than " << MIN_YEARS
31         << " years.\n";
32 }

```

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## The logical ! Operator in Program 4-17

```

23 // Determine the user's loan qualifications.
24 if (!(income >= MIN_INCOME || years > MIN_YEARS))
25 {
26     cout << "You must earn at least $"
27         << MIN_INCOME << " or have been "
28         << "employed more than " << MIN_YEARS
29         << " years.\n";
30 }
31 else
32     cout << "You qualify.\n";

```

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## Logical Operator-Notes

- ❶ ! has highest precedence, followed by &&, then ||
- ❷ 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*)

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## 4.9

### Checking Numeric Ranges with Logical Operators



Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## Checking Numeric Ranges with Logical Operators

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

```
if (grade >= 0 && grade <= 100)
    cout << "Valid grade";
```
- ❷ Can also test to see if value falls **outside** of range:
 

```
if (grade <= 0 || grade >= 100)
    cout << "Invalid grade";
```
- ❸ Cannot use mathematical notation:
 

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

Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## 4.10

### Menus



Addison-Wesley  
is an imprint of  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.



## Menus

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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Menu-Driven Program Organization

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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 4.11

### Validating User Input

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Validating User Input

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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Input Validation in Program 4-19

```

16 int testScore; // To hold a numeric test score
17
18 // Get the numeric test score.
19 cout << "Enter your numeric test score and I will\n";
20 << "tell you the letter grade you earned: ";
21 cin >> testScore;
22
23 // Validate the input and determine the grade.
24 if (testScore >= MIN_SCORE && testScore <= MAX_SCORE)
25 {
26     // Determine the letter grade.
27     if (testScore >= A_SCORE)
28         cout << "Your grade is A.\n";
29     else if (testScore >= B_SCORE)
30         cout << "Your grade is B.\n";
31     else if (testScore >= C_SCORE)
32         cout << "Your grade is C.\n";
33     else if (testScore >= D_SCORE)
34         cout << "Your grade is D.\n";
35     else
36         cout << "Your grade is F.\n";
37 }
38 else
39 {
40     // An invalid score was entered.
41     cout << "That is an invalid score. Run the program\n";
42     << "again and enter a value in the range of\n";
43     << MIN_SCORE << " through " << MAX_SCORE << ".\n";
44 }

```

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 4.12

### Comparing Characters and Strings

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Comparing Characters

- ❶ Characters are compared using their ASCII values
- ❷ 'A' < 'B'
  - ❶ The ASCII value of 'A' (65) is less than the ASCII value of 'B'(66)
- ❸ '1' < '2'
  - ❶ The ASCII value of '1' (49) is less than the ASCII value of '2' (50)
- ❹ Lowercase letters have higher ASCII codes than uppercase letters, so 'a' > 'Z'

Addison-Wesley  
3 as imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Relational Operators Compare Characters in Program 4-20

```

10 // Get a character from the user.
11 cout << "Enter a digit or a letter: ";
12 ch = cin.get();
13
14 // Determine what the user entered.
15 if (ch >= '0' && ch <= '9')
16     cout << "You entered a digit.\n";
17 else if (ch >= 'A' && ch <= 'Z')
18     cout << "You entered an uppercase letter.\n";
19 else if (ch >= 'a' && ch <= 'z')
20     cout << "You entered a lowercase letter.\n";
21 else
22     cout << "That is not a digit or a letter.\n";

```

Addison-Wesley  
3 as imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Comparing string Objects

- ❶ Like characters, strings are compared using their ASCII values

```
string name1 = "Mary";
string name2 = "Mark";
```

The characters in each string must match before they are equal

```
name1 > name2 // true
name1 <= name2 // false
name1 != name2 // true
```

```
name1 < "Mary Jane" // true
```

Addison-Wesley  
3 as imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Relational Operators Compare Strings in Program 4-21

```

26 // Determine and display the correct price
27 if (partNum == "S-29A")
28     cout << "The price is $" << PRICE_A << endl;
29 else if (partNum == "S-29B")
30     cout << "The price is $" << PRICE_B << endl;
31 else
32     cout << partNum << " is not a valid part number.\n";

```

Addison-Wesley  
3 as imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

# 4.13

## The Conditional Operator

Addison-Wesley  
3 as imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The Conditional Operator

- ❶ Can use to create short if/else statements
- ❷ 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

Addison-Wesley  
3 as imprint of

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The Conditional Operator

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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The Conditional Operator in Program 4-22

```

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; // Hourly pay rate
11     const int MIN_HOURS = 5; // Minimum billable hours
12     double hours; // Hours worked
13     charges; // Total charges
14
15     // Get the hours worked.
16     cout << "How many hours were worked? ";
17     cin >> hours;
18
19     // Determine the hours to charge for.
20     hours = hours < MIN_HOURS ? MIN_HOURS : hours;
21
22     // Calculate and display the charges.
23     charges = PAY_RATE * hours;
24     cout << fixed << showpoint << setprecision(2)
25         << "The charges are $" << charges << endl;
26     return 0;
27 }

```

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## 4.14

### The switch Statement

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The switch Statement

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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## switch Statement Format

```

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

```

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## The switch Statement in Program 4-23

### Program 4-23

```

1 // The switch statement in this program tells the user something
2 // he or she already knows: the data 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!

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## switch Statement Requirements

- 1) *expression* must be an integer variable or an expression that evaluates to an integer value
- 2) *exp1* through *expn* must be constant integer expressions or literals, and must be unique in the `switch` statement
- 3) `default` is optional but recommended

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## switch Statement-How it Works

- 1) *expression* is evaluated
- 2) The value of *expression* is compared against *exp1* through *expn*.
- 3) If *expression* matches value *exp1*, the program branches to the statement following *exp1* and continues to the end of the `switch`
- 4) If no matching value is found, the program branches to the statement after `default`:

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## break Statement

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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## break and default statements in Program 4-25

### Program 4-25

```

1 // This program is carefully constructed to use the 'fall through'
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 }
```

Continued...

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.

## break and default statements in Program 4-25

```

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.
```

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.


## Using switch in Menu Systems

- `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

Addison-Wesley  
Learning Technology

PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley All rights reserved.



# 4.15

## More About Blocks and Scope

Addison-Wesley  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## More About Blocks and Scope

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

Addison-Wesley  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Inner Block Variable Definition in Program 4-29

```

16  if (income >= MIN_INCOME)
17  {
18      // Get the number of years at the current job.
19      cout << "How many years have you worked at "
20          << "your current job? ";
21      int years; // Variable definition
22      cin >> years;
23
24      if (years > MIN_YEARS)
25          cout << "You qualify.\n";
26      else
27      {
28          cout << "You must have been employed for\n"
29              << "more than " << MIN_YEARS
30              << " years to qualify.\n";
31      }
32  }

```

Addison-Wesley  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Variables with the Same Name

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

Addison-Wesley  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.

## Two Variables with the Same Name in Program 4-30

```

Program 4-30
1 // This program uses two variables with the name number.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     // Define a variable named number.
8     int number;
9
10    cout << "Enter a number greater than 0: ";
11    cin >> number;
12    if (number > 0)
13    {
14        int number; // Another variable named number.
15        cout << "now enter another number: ";
16        cin >> number;
17        cout << "The second number you entered was "
18            << number << endl;
19    }
20    cout << "your first number was " << number << endl;
21    return 0;
22 }

```

**Program Output with Example Input Shown in Bold**

```

Enter a number greater than 0: 2 [Enter]
now enter another number: 7 [Enter]
The second number you entered was 7
Your first number was 2

```

Addison-Wesley  
PEARSON

Copyright © 2015, 2012, 2009 Pearson Education, Inc., Publishing as Addison-Wesley. All rights reserved.