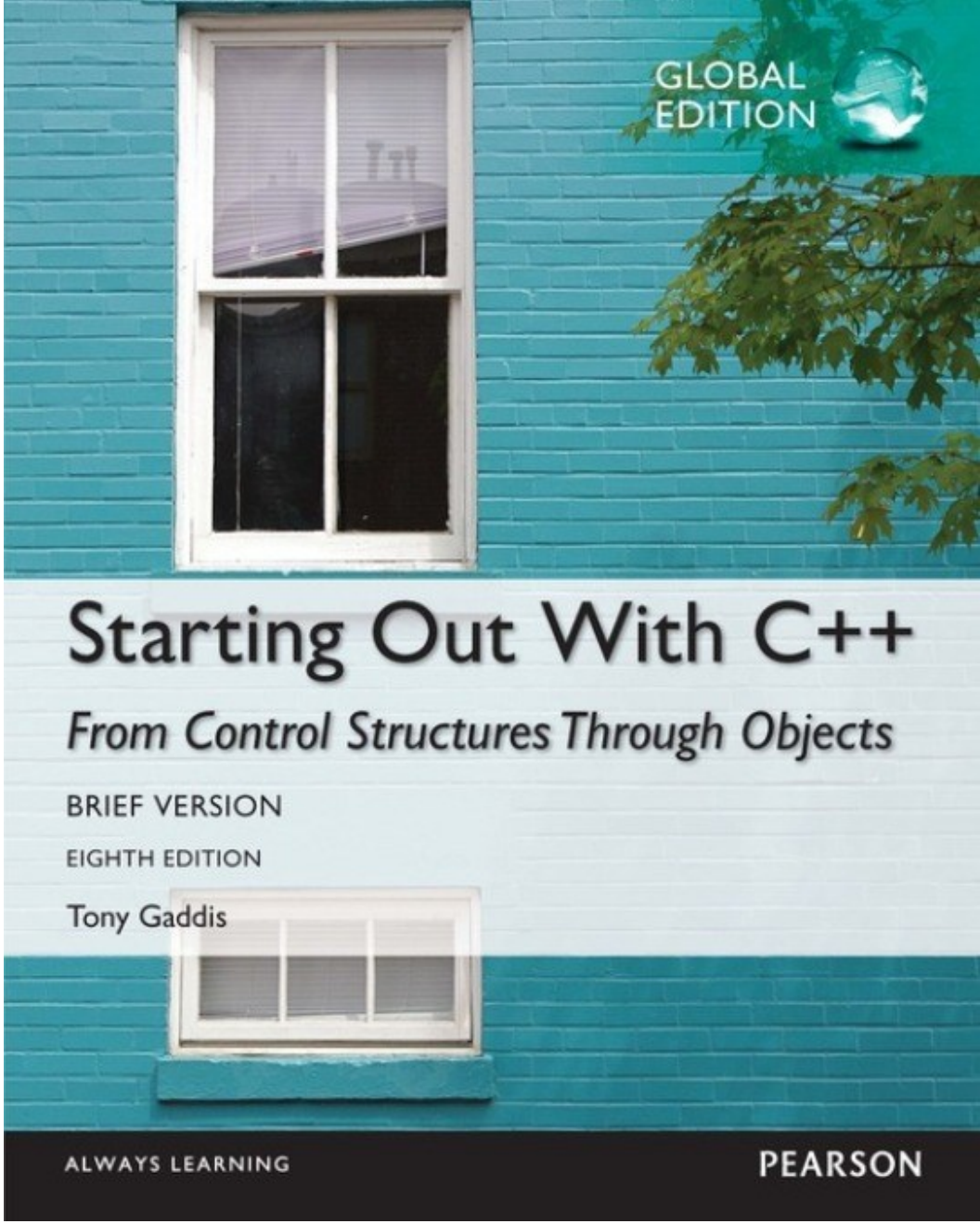
Chapter 4:

Making Decisions



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:

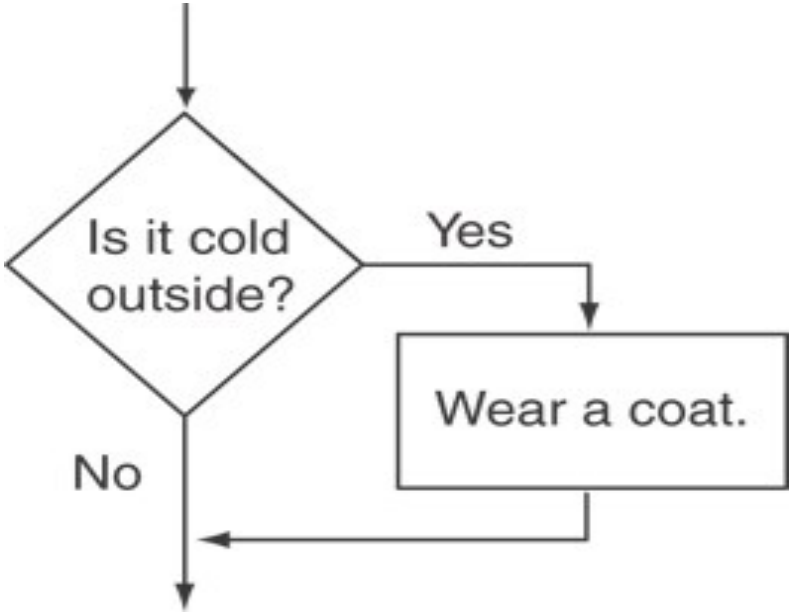
```
result = x <= y;
```

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

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



The `if` Statement

- General Format:

```
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

```
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…

`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
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  }
```

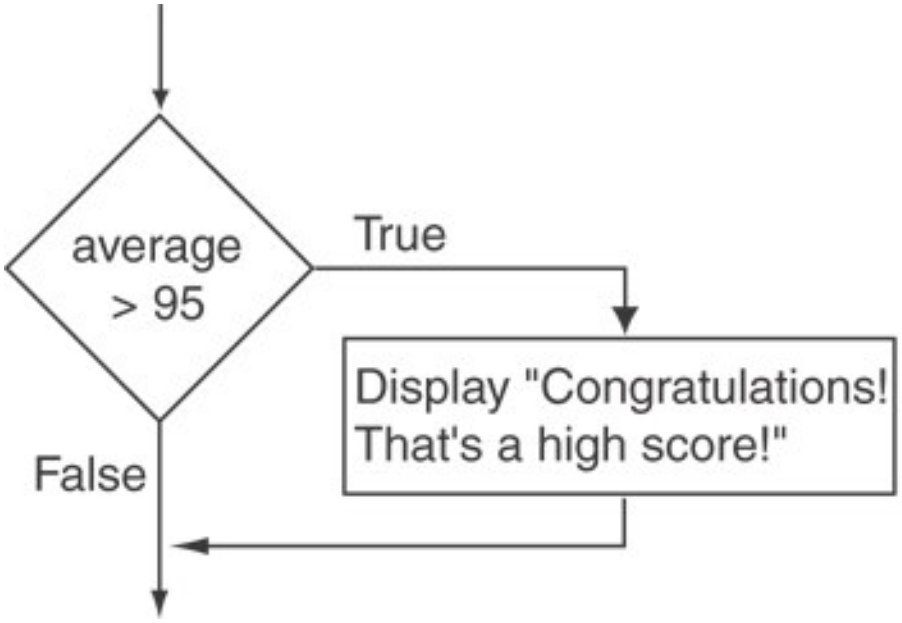
Program Output with Example Input Shown in Bold

```
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!
```

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';
```

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

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 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

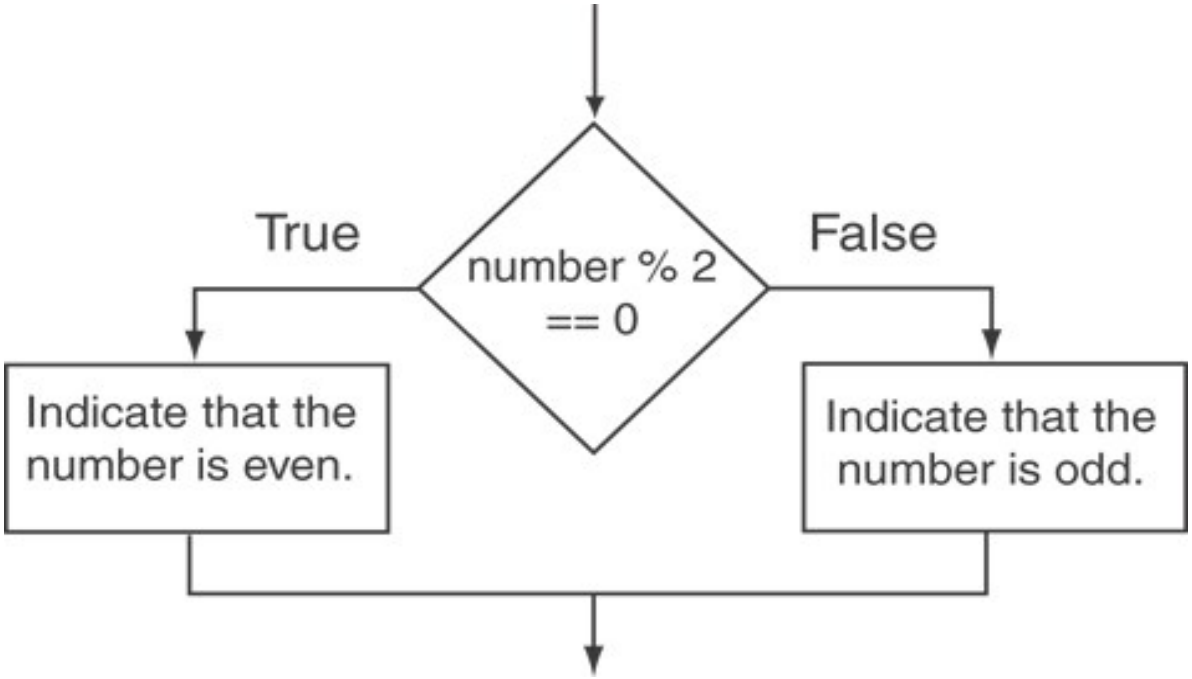
Program 4-8

```
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.
```

Flowchart for Program 4-8 Lines 14 through 18



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…

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

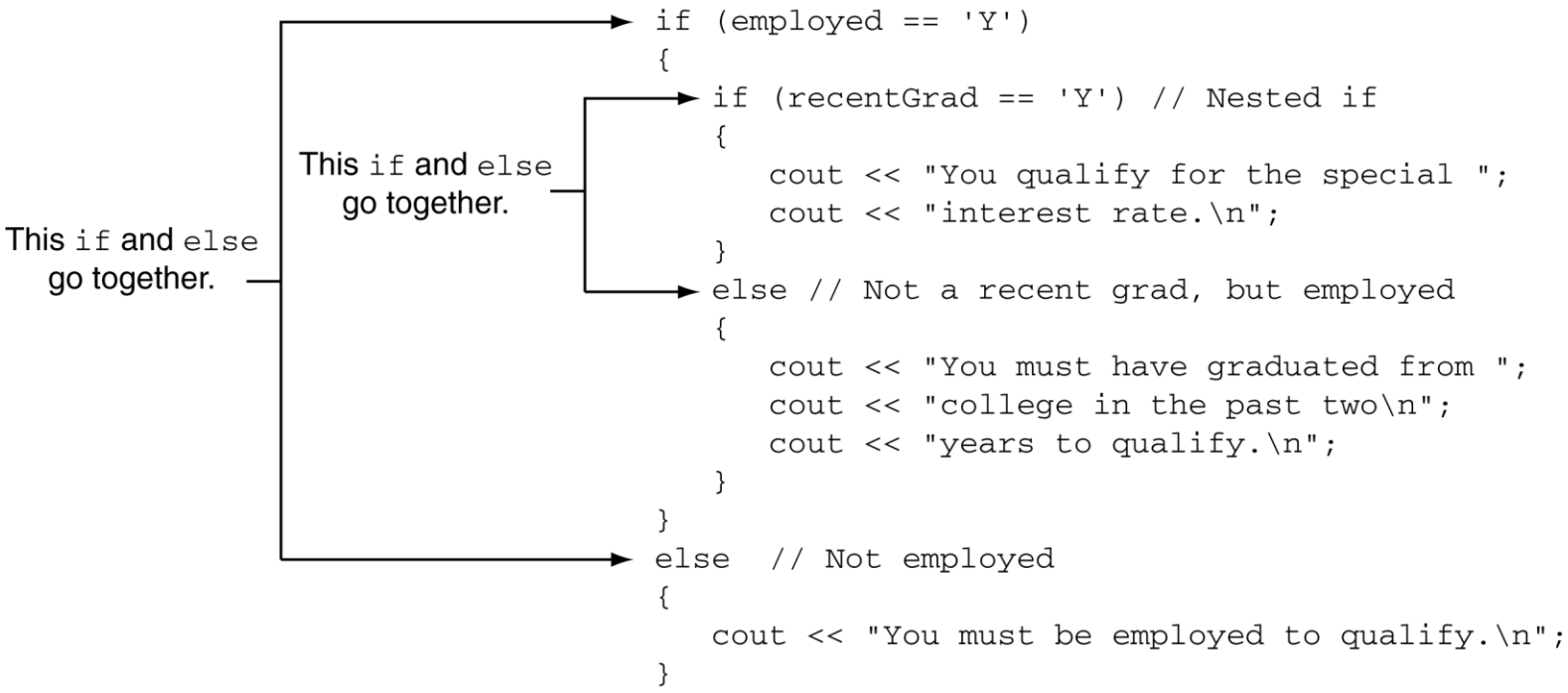
(When the user enters 0 for num2)

```
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.
```

Nested `if` Statements

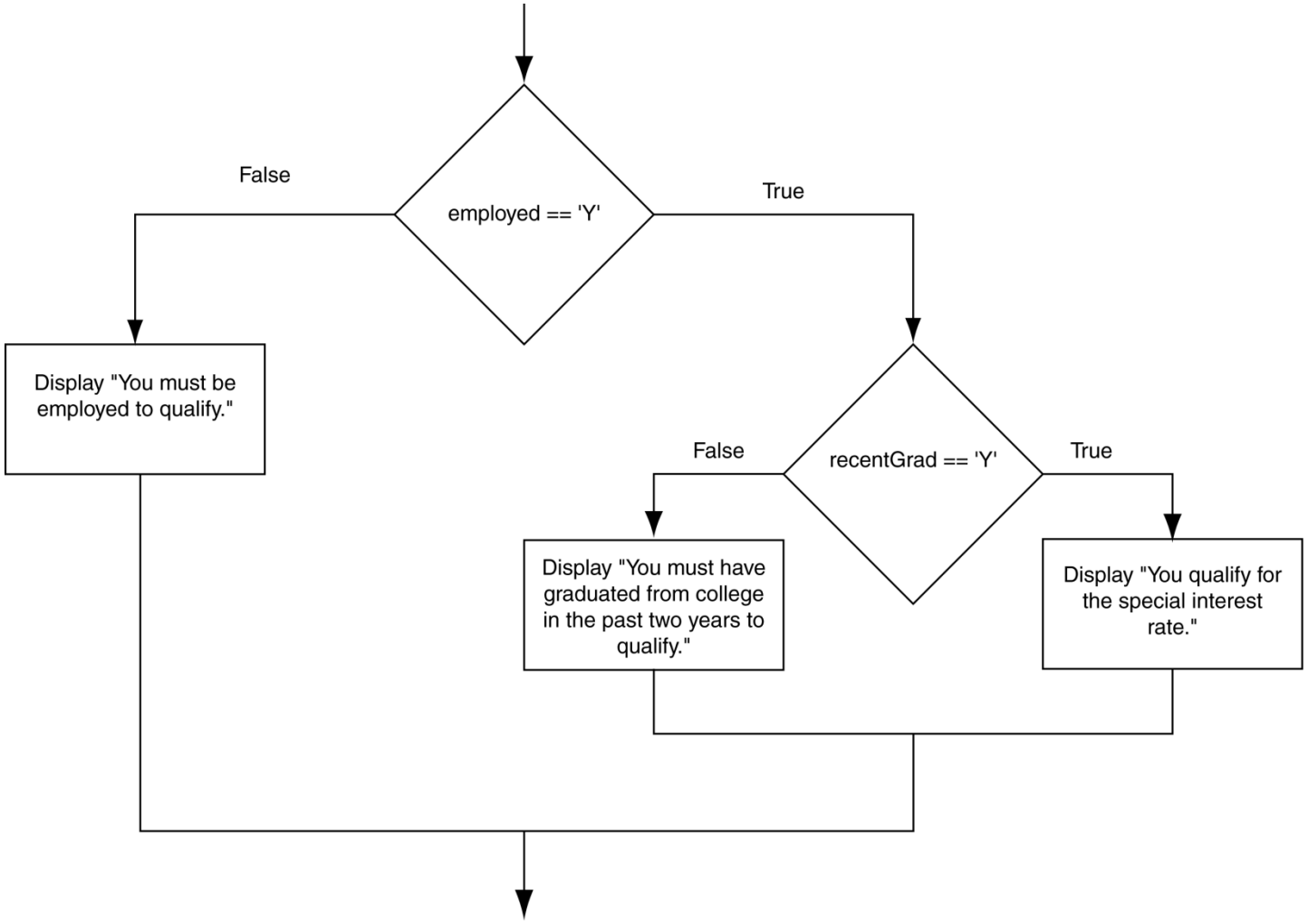
- An `if` statement that is nested inside another `if` statement
- Nested `if` statements can be used to test more than one condition

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";
}
```

Flowchart for a Nested `if` Statement



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”

if/else if Format

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

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";
```

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

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`

The logical operators in Coding

- The logical **&&** Operator :

```
if (x > y && y > z)
   cout << "x is greater than y and z";
```

- The logical **||** Operator :

```
if (z < x || z == y)
   cout << "z is less than x OR z equals y";
```

- The logical ! Operator :

```
if (!(y >= z))
   cout << "y is less than z";
else
   cout << "y is greater than z ";
```

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!
```

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

Menu-Driven Program Example

```
#include <iostream>

using namespace std;

int main()
{
        char choice;

        cout << "Good day! Welcome to The Bakery! What would you like today?\n";
        cout << "A: Earl Gray Tea and Biscuits\n";
        cout << "B: Coffee and a blueberry scone\n";
        cout << "C: Espresso and a tea biscuit\n";
        cout << "D: Coffee and a Muffin\n";
        cout << "E: The Assorted Tea, Scones, and Biscuits Platter\n";

        cout << "\n Select your choice: ";
        cin >> choice;

        if (choice == 'A' || choice == 'a')
                cout << "A: Earl Gray Tea and Biscuits\n";

        else if (choice == 'B' || choice == 'b')
                cout << "B: Coffee and a blueberry scone\n";

        else if (choice == 'C' || choice == 'c')
                cout << "A: Earl Gray Tea and Biscuits\n";

        else if (choice == 'D' || choice == 'd')
                cout << "D: Coffee and a Muffin\n";

        else if (choice == 'E' || choice == 'e')
                cout << "E: The Assorted Tea, Scones, and Biscuits Platter\n";

        else //Displaying error message
                cout << "Invalid input\n";

        return  0;
}
```

Validating User Input

- Input validation: inspecting input data to determine whether it is acceptable
- Bad output will be produced from bad input

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    }
```

Comparing Characters

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

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";
```

Comparing `string` Objects

- Like characters, strings are compared using their ASCII values

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

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

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

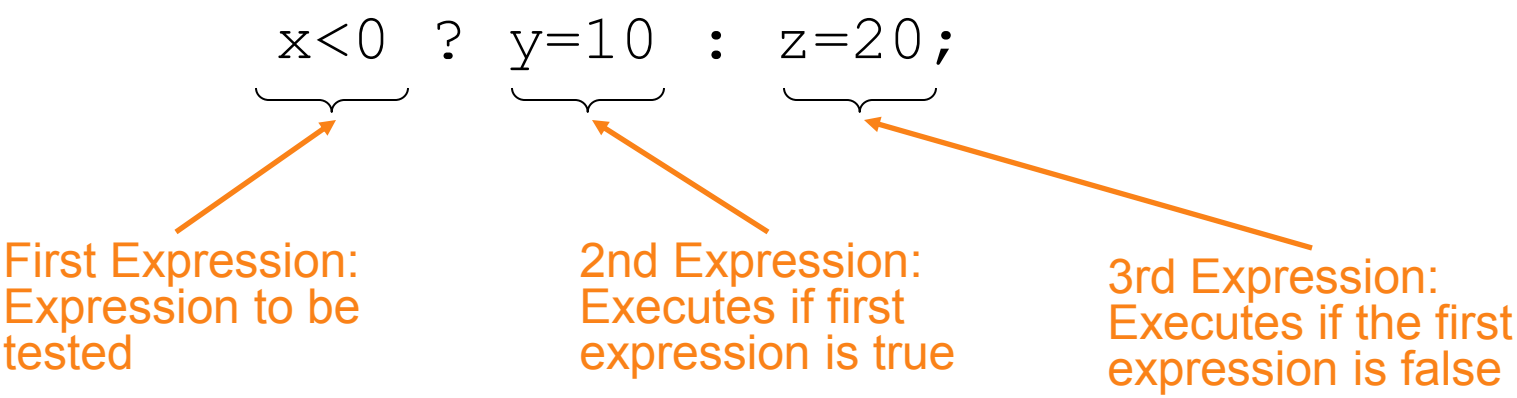
The characters in each string must match before they are equal

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";
```

The Conditional Operator

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



The `switch` Statement

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

`switch` Statement Format

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

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!
```

`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

`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:`

`break` Statement

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

`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…

`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.
```

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

Variables with the Same Name

- 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

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
```