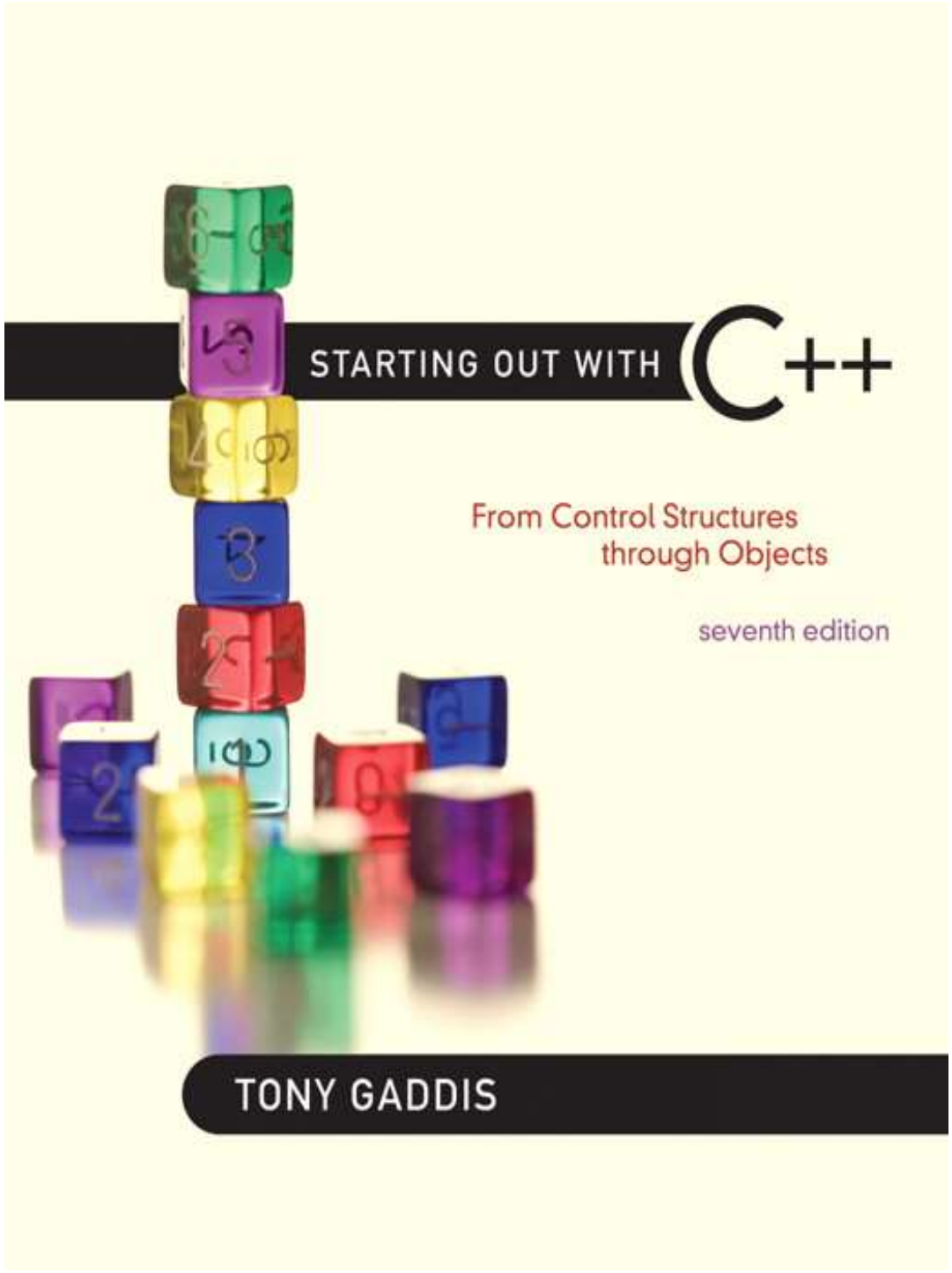
Chapter 4:

Making Decisions



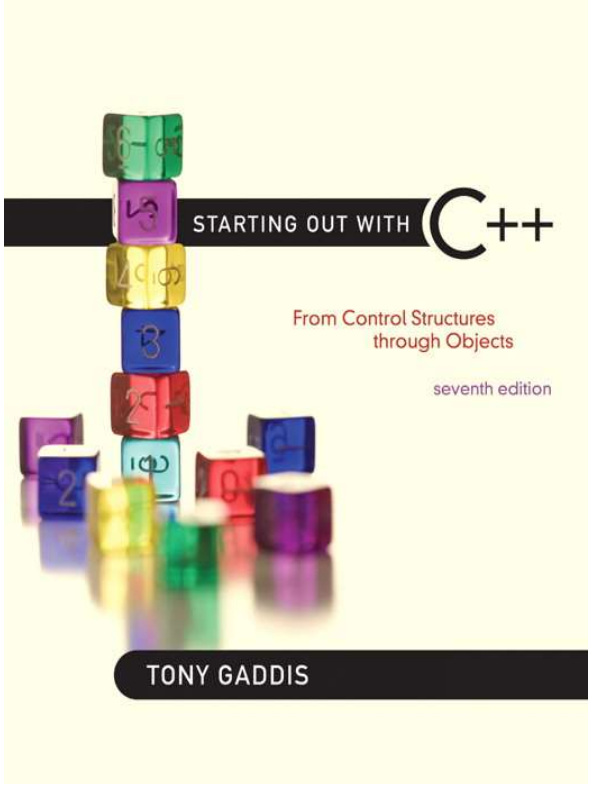
Addison-Wesley
is an imprint of
PEARSON

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

Precedence

Operators	Precedence
`()`	highest (applied first)
`* / %`	
`+ -`	
`< <= > >=`	
`== !=`	
`=`	lowest (applied last)

4.5

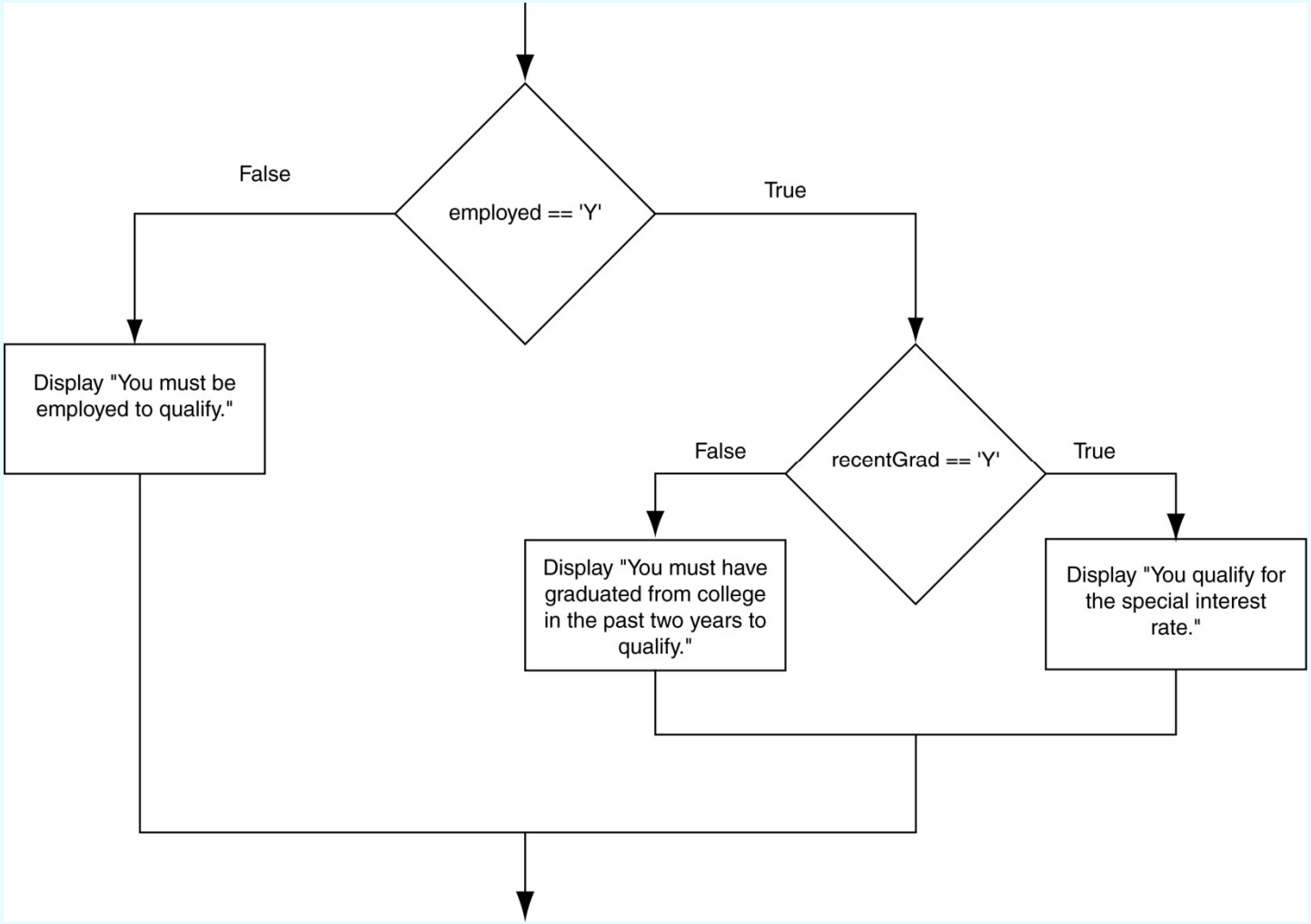


Nested `if` Statements

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

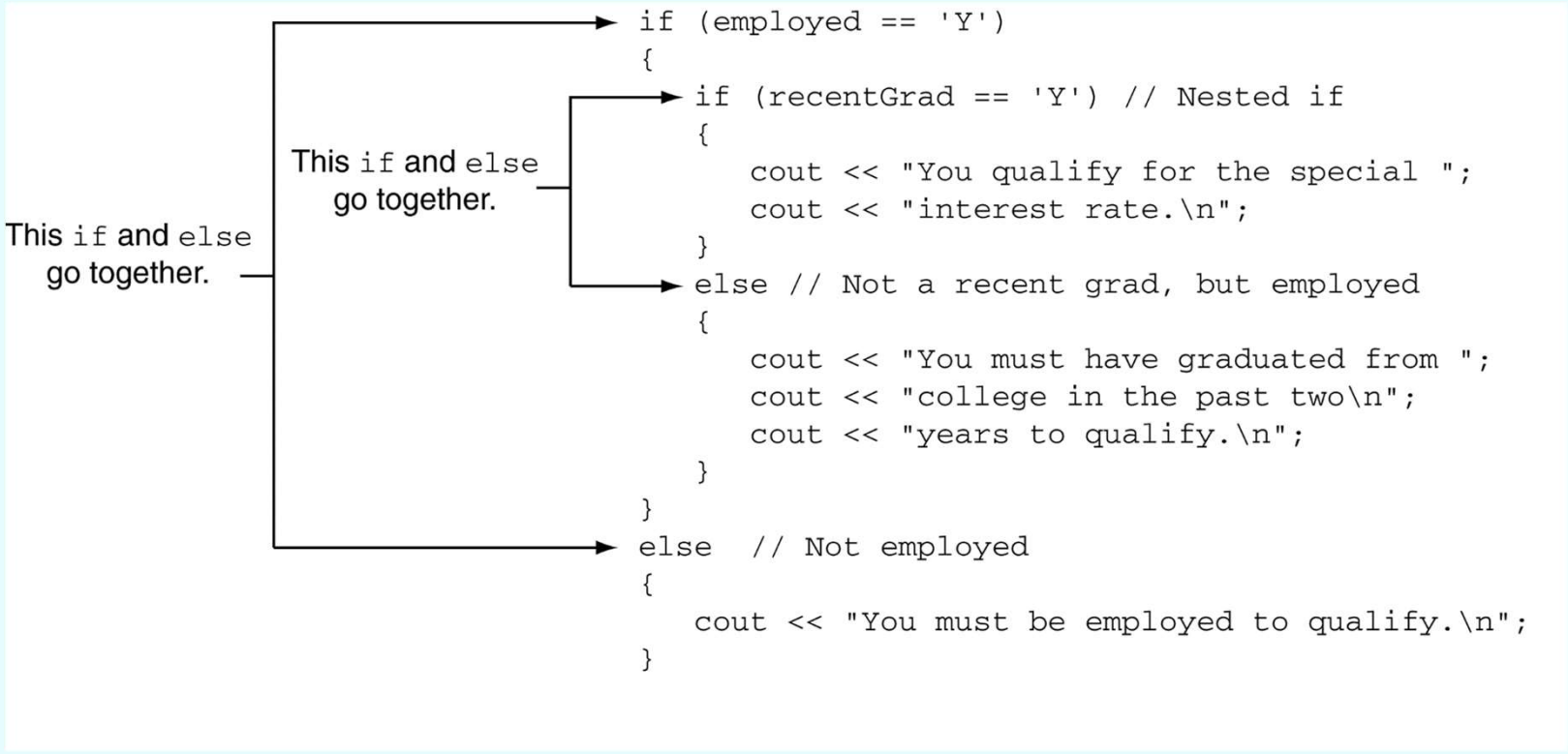
```
20      // Determine the user's loan qualifications.
21      if (employed == 'Y')
22      {
23         if (recentGrad == 'Y') //Nested if
24         {
25            cout << "You qualify for the special ";
26            cout << "interest rate.\n";
27         }
28      }
```

Nested `if` Statements

- Another example, from Program 4-1

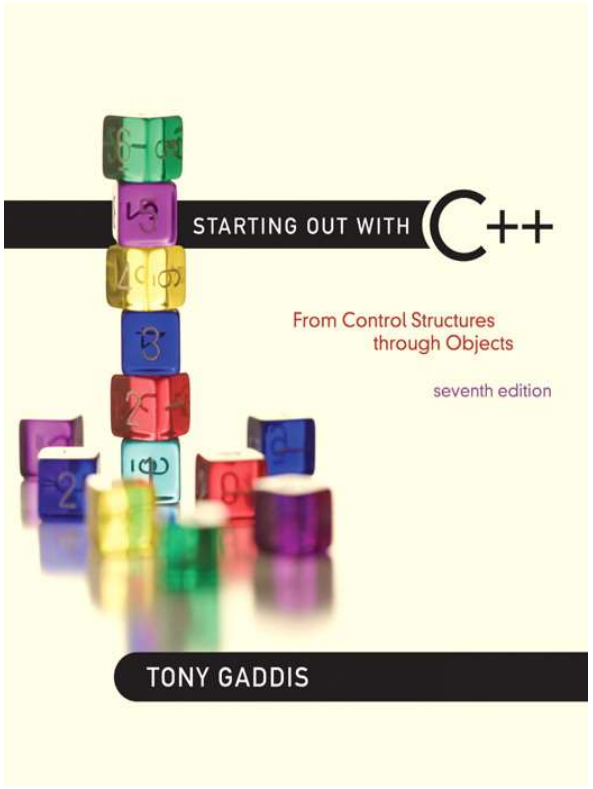
```
20      // Determine the user's loan qualifications.
21      if (employed == 'Y')
22      {
23         if (recentGrad == 'Y') // Nested if
24         {
25            cout << "You qualify for the special ";
26            cout << "interest rate.\n";
27         }
28         else // Not a recent grad, 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!




```
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";
}
```

4.6



The `if/else if` Statement

Multi-selector

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

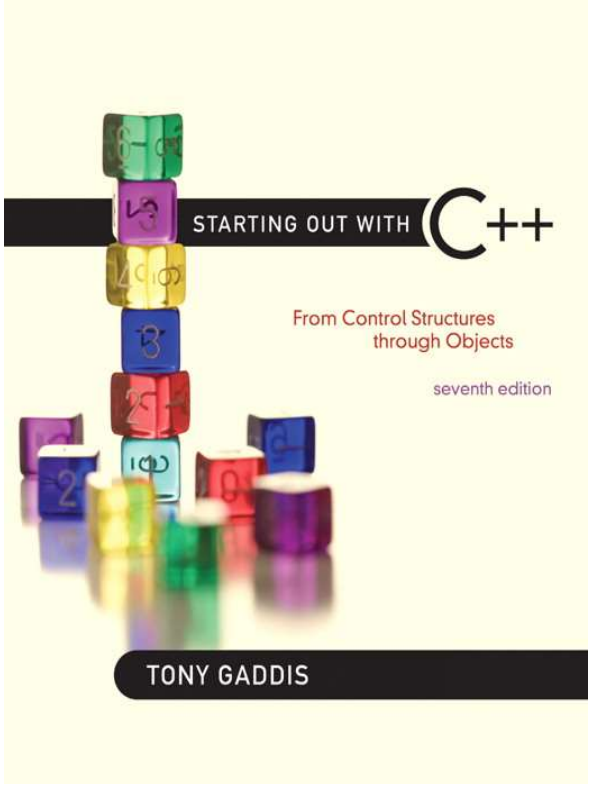
```
21      // Determine the letter grade.
22      if (testScore >= A_SCORE)
23         cout << "Your grade is A.\n";
24      else if (testScore >= B_SCORE)
25         cout << "Your grade is B.\n";
26      else if (testScore >= C_SCORE)
27         cout << "Your grade is C.\n";
28      else if (testScore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

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

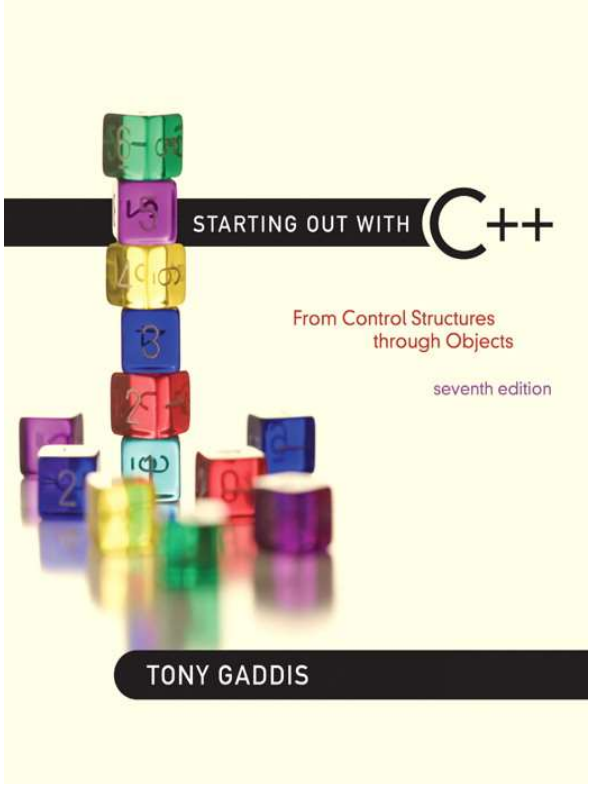


4.7

Flags

Flags

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



4.8

Logical Operators

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 && 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    }
```

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    }
```

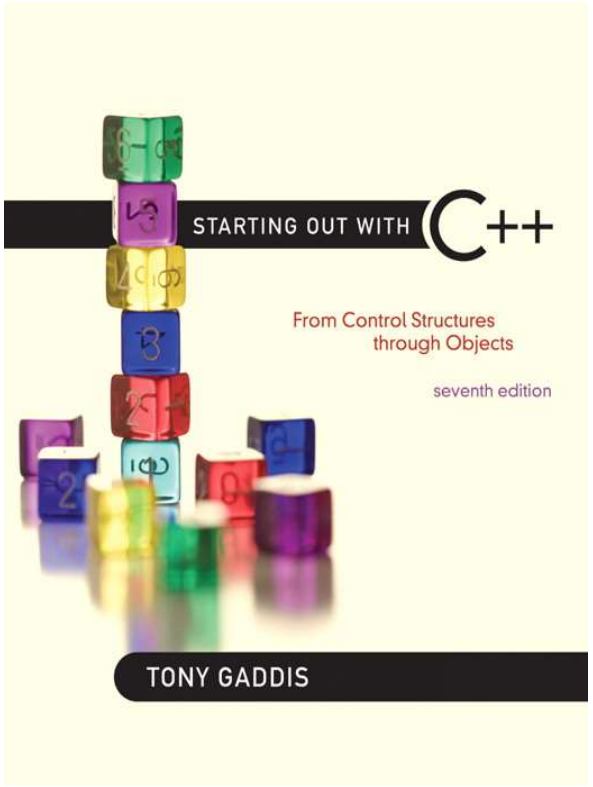
The logical ! Operator in Program 4-17

```
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";
```

Logical Operator-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*)

4.9



Checking Numeric Ranges with Logical Operators

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";
```

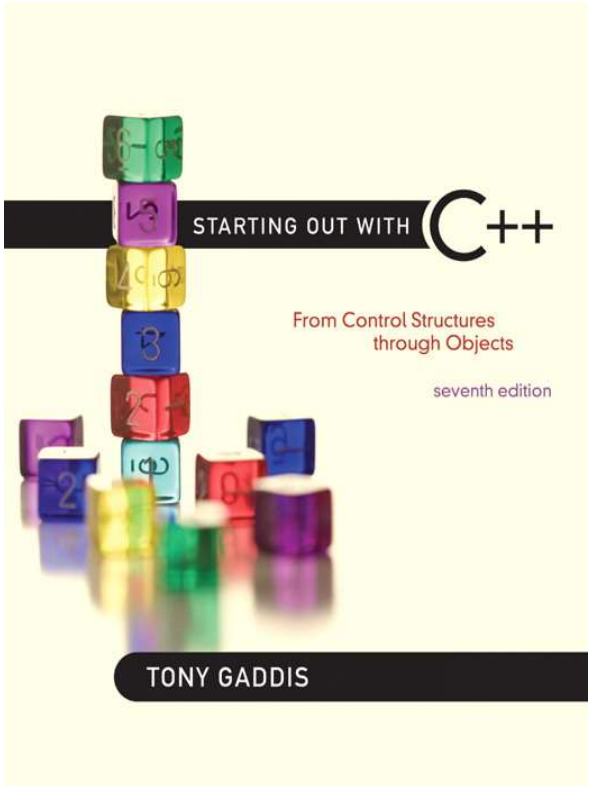
- Can also test to see if value falls **outside** of range:

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

- Cannot use mathematical notation:

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

4.10



Menus

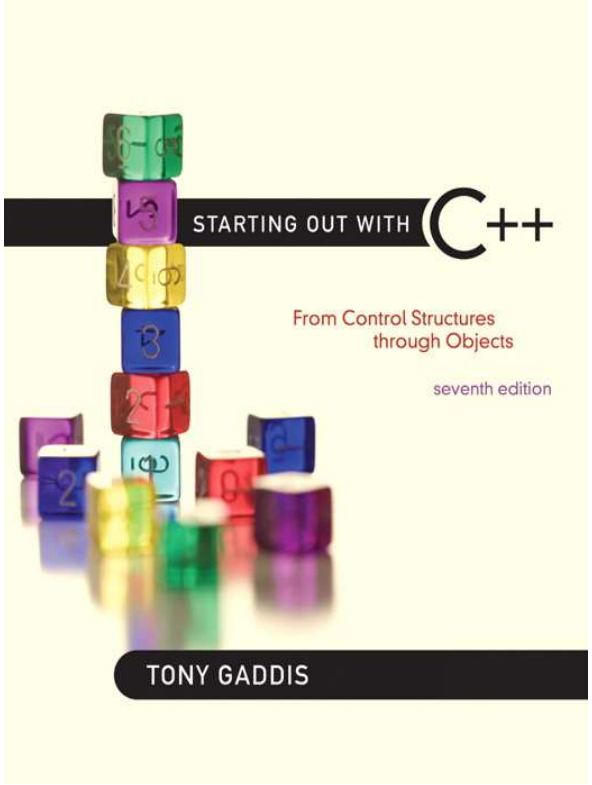
Menus

- 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 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`*

4.11



Validating User Input

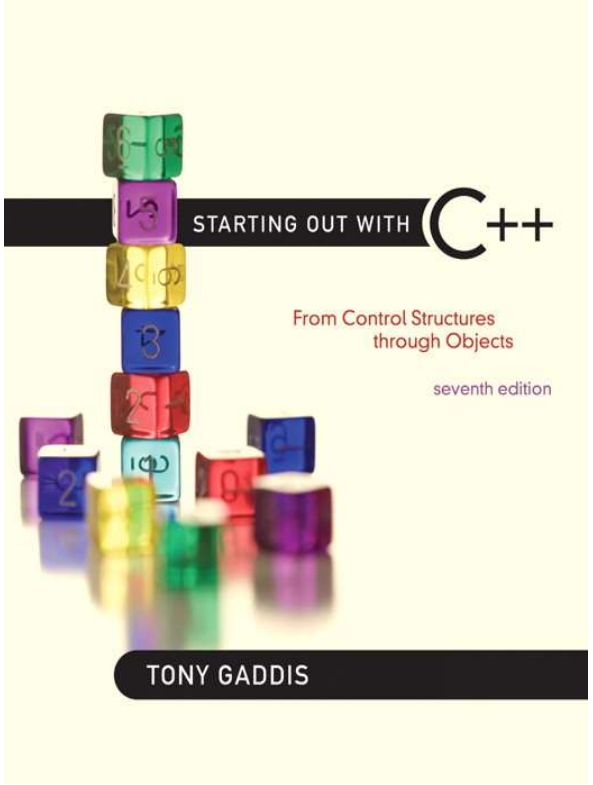
Validating User Input

- 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

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    }
```

4.12



Comparing Characters and Strings

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

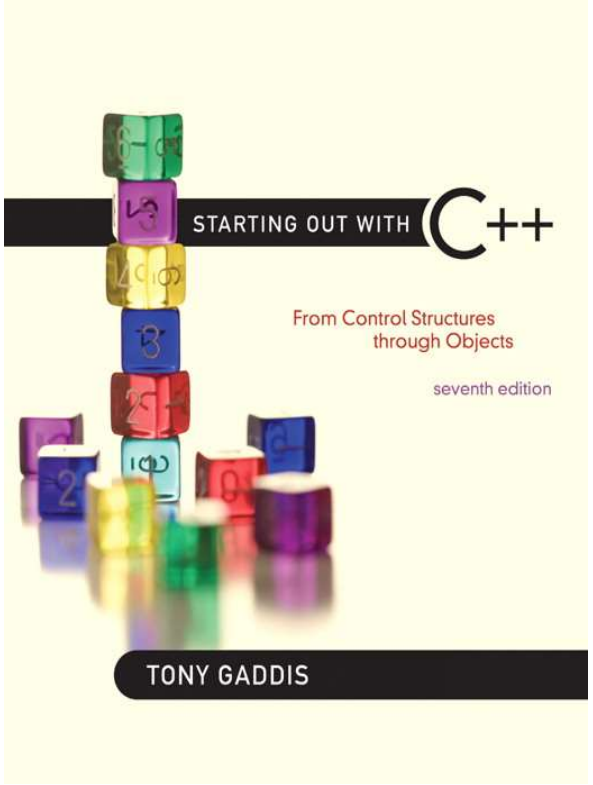
```
string name1 = "Mary";
string name2 = "Mark";

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";
```

4.14



The `switch` Statement

Multi-Selector

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.
```

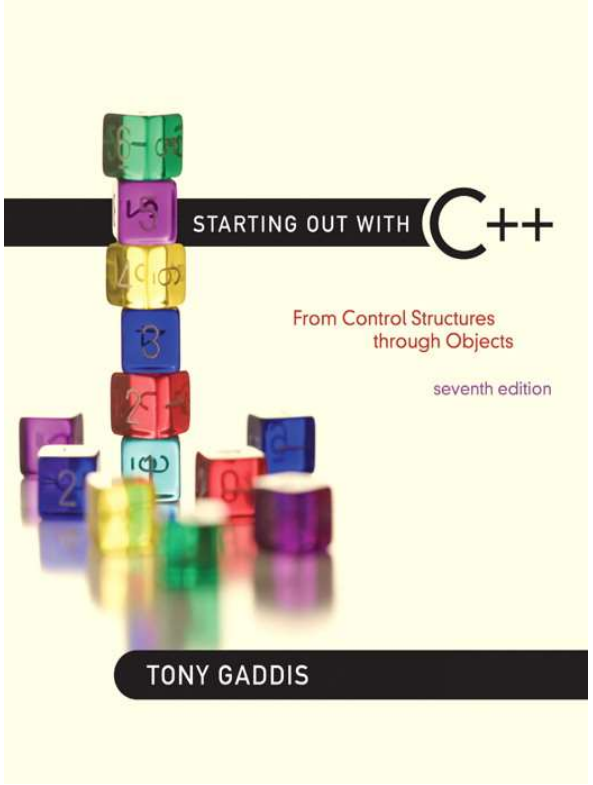
Program Output with Example Input Shown in Bold

```
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

4.15



More About Blocks and Scope

More About Blocks 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

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    }
```

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
```