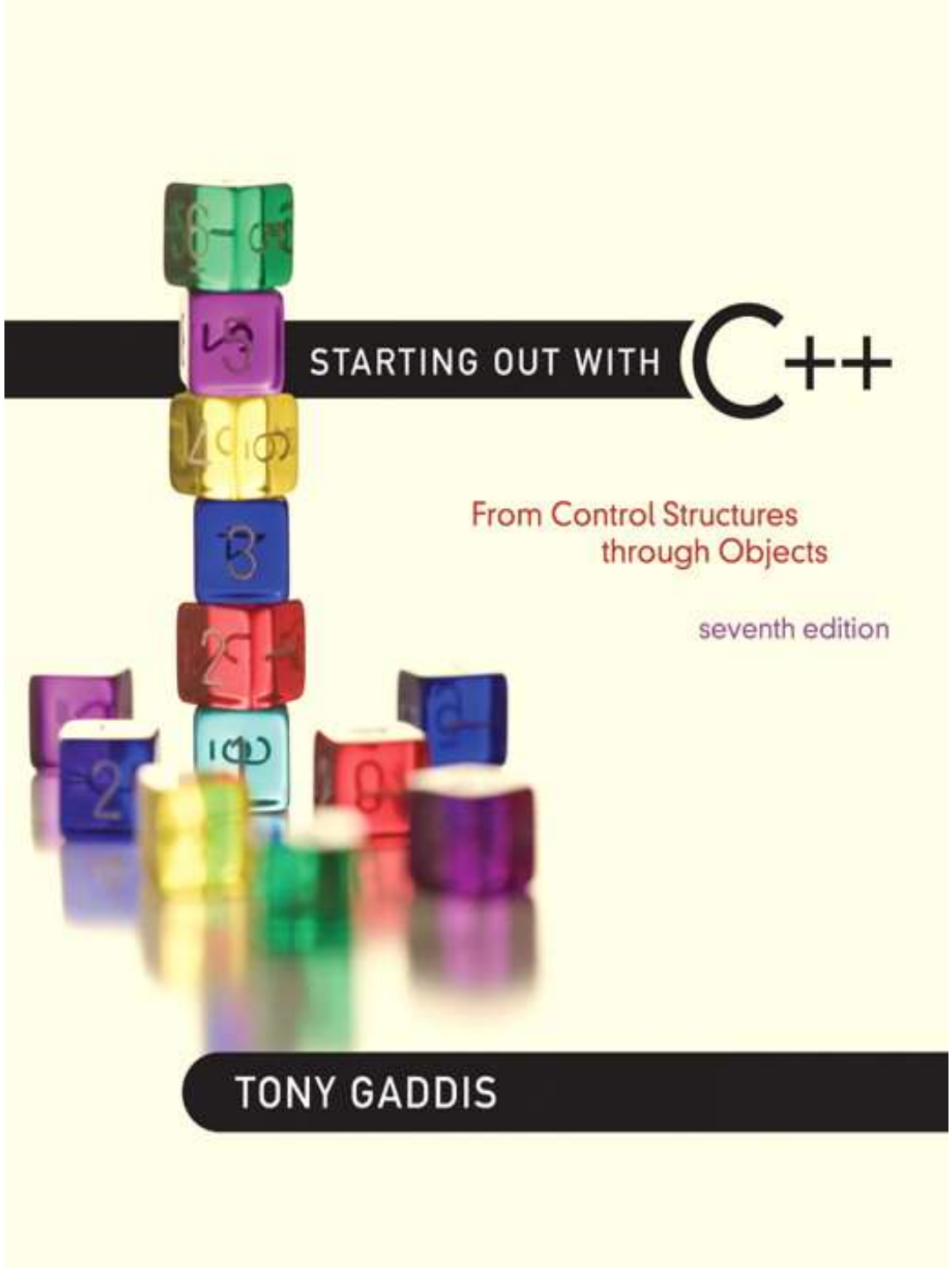
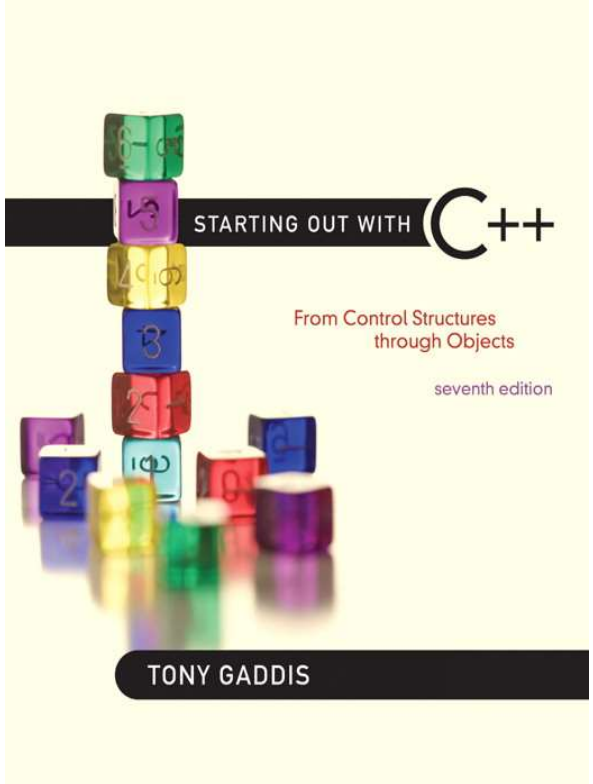
Chapter 4:

Making Decisions



Addison-Wesley
is an imprint of
PEARSON



4

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

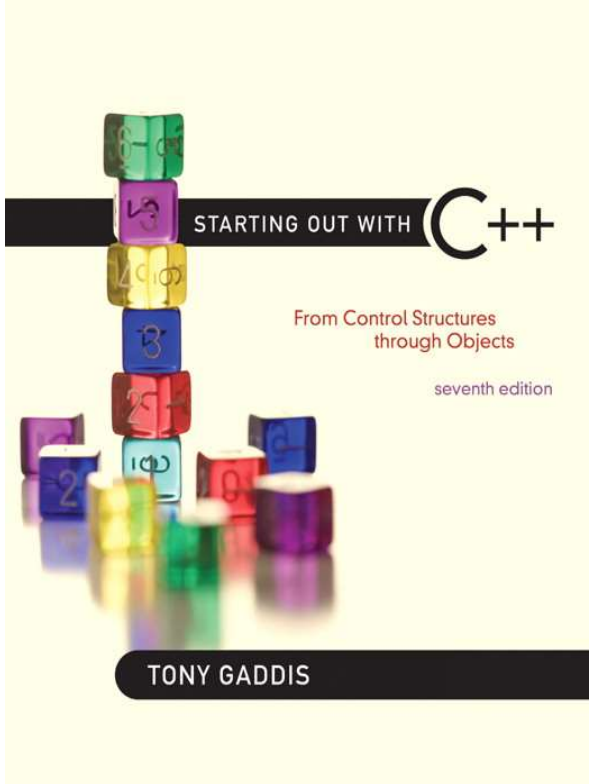
```
21    // Determine the letter grade.
22    if (testScore >= A_SCORE)
23       cout << "Your grade is A.\n";
24    else if (testScore >= B_SCORE)
25       cout << "Your grade is B.\n";
26    else if (testScore >= C_SCORE)
27       cout << "Your grade is C.\n";
28    else if (testScore >= D_SCORE)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

```
// Determine the letter grade.
if (testScore >= A_SCORE)
   cout << "Your grade is A.\n";
else if (testScore >= B_SCORE)
   cout << "Your grade is B.\n";
else if (testScore >= C_SCORE)
   cout << "Your grade is C.\n";
else if (testScore >= D_SCORE)
   cout << "Your grade is D.\n";
else if (testScore >= 0)
   cout << "Your grade is F.\n";
else
   cout << "Invalid test score.\n";
```

This trailing `else` catches invalid test scores

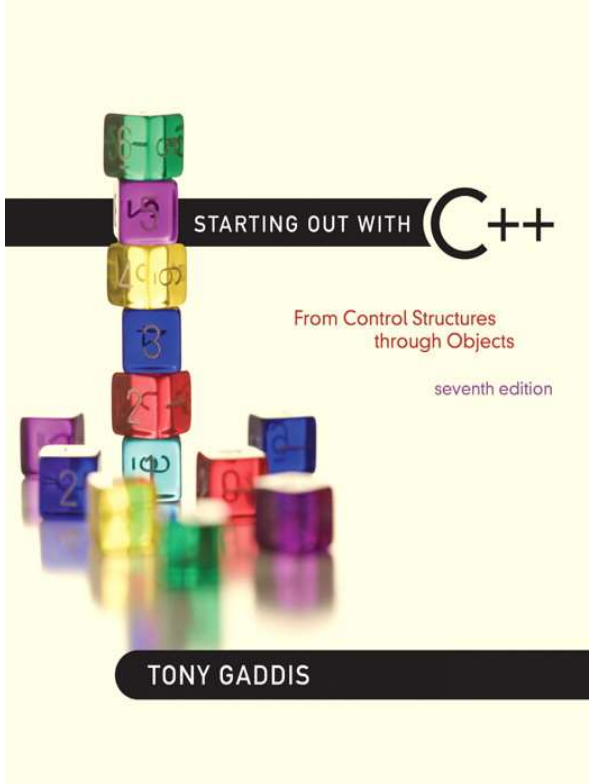


Flags

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

Review



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

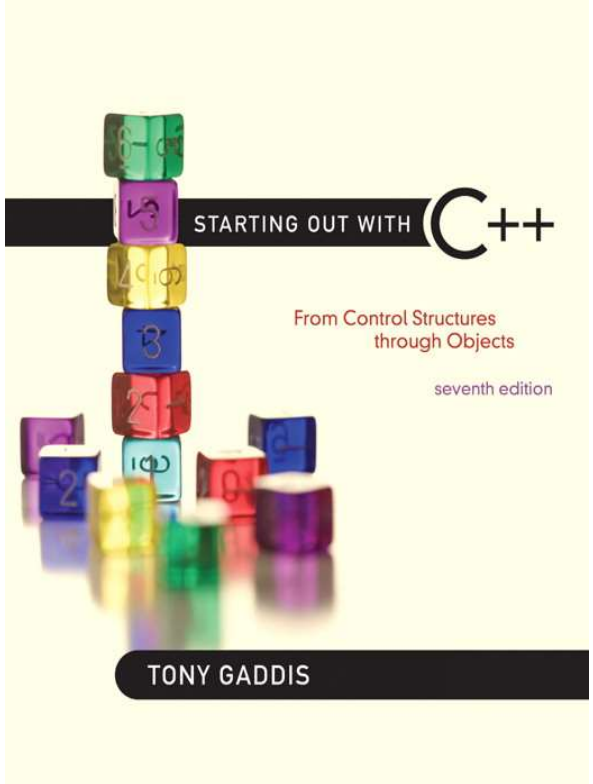
```
// Determine the user's loan qualifications.
if (income >= MIN_INCOME || years > MIN_YEARS)
   cout << "You qualify.\n";
else
{
   cout << "You must earn at least $"
        << MIN_INCOME << " or have been "
        << "employed more than " << MIN_YEARS
        << " years.\n";
}
```

The logical ! Operator in Program 4-17

```
// Determine the user's loan qualifications.
if (!(income >= MIN_INCOME || years > MIN_YEARS))
{
   cout << "You must earn at least $"
        << MIN_INCOME << " or have been "
        << "employed more than " << MIN_YEARS
        << " years.\n";
}
else
   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*)



Checking Numeric Ranges with Logical Operators

Checking Numeric Ranges with Logical Operators

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

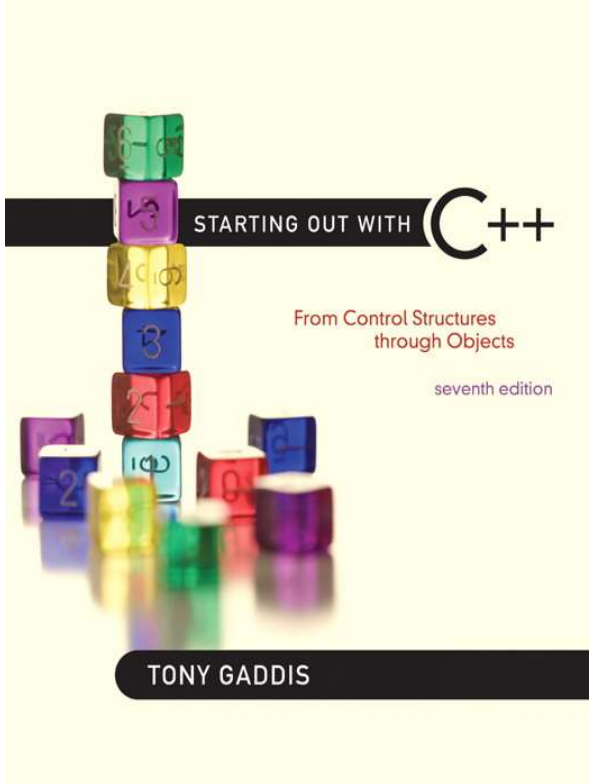
```
if (grade >= 0 && grade <= 100)
   cout << "Valid grade";
```

- 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!
```



The Conditional Operator

Two Way Selector

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

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

The Conditional Operator in Program 4-22

```
// This program calculates a consultant's charges at $50
// per hour, for a minimum of 5 hours. The ?: operator
// adjusts hours to 5 if less than 5 hours were worked.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   const double PAY_RATE = 50.0;  // Hourly pay rate
   const int MIN_HOURS = 5;       // Minimum billable hours
   double hours,                  // Hours worked
          charges;                // Total charges

   // Get the hours worked.
   cout << "How many hours were worked? ";
   cin >> hours;

   // Determine the hours to charge for.
   hours = hours < MIN_HOURS ? MIN_HOURS : hours;

   // Calculate and display the charges.
   charges = PAY_RATE * hours;
   cout << fixed << showpoint << setprecision(2)
        << "The charges are $" << charges << endl;
   return 0;
}
```

Two Variables with the Same Name in Program 4-30

Program 4-30

```
// This program uses two variables with the name number.
#include <iostream>
using namespace std;

int main()
{
   // Define a variable named number.
   int number;

   cout << "Enter a number greater than 0: ";
   cin >> number;
   if (number > 0)
   {
      int number;  // Another variable named number.
      cout << "Now enter another number: ";
      cin >> number;
      cout << "The second number you entered was "
           << number << endl;
   }
   cout << "Your first number was " << number << endl;
   return 0;
}
```

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
```