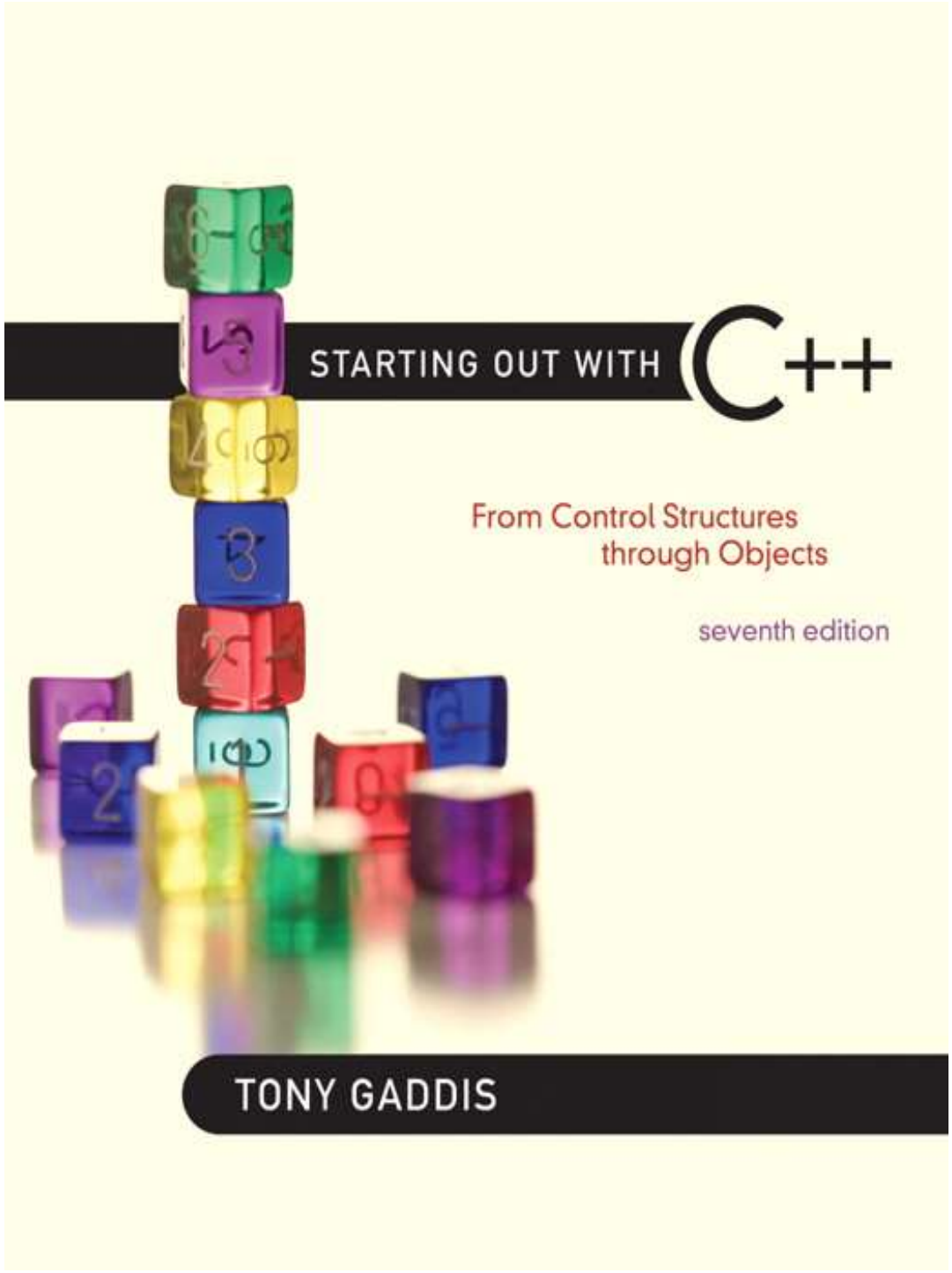
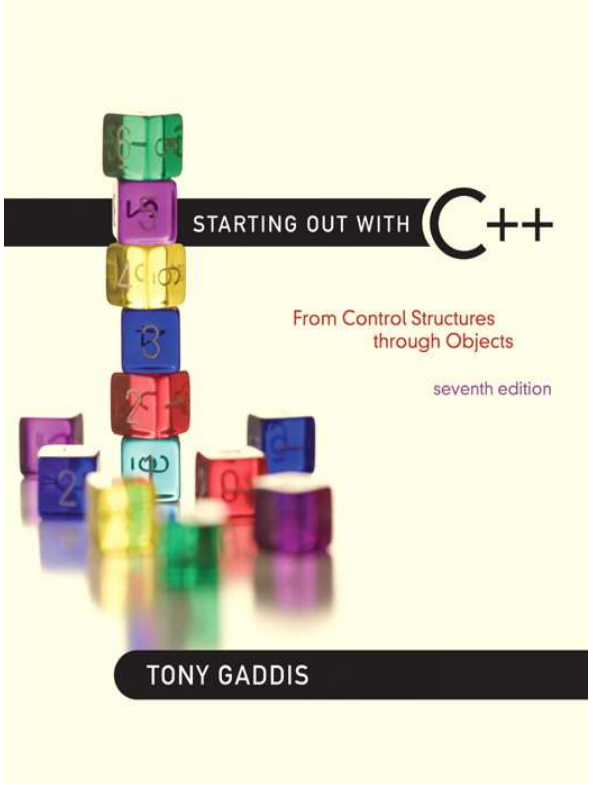
Chapter 4:

Relational and logicals operators



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



Relational Operators

Relational Operators

- **Relational operators allow you to compare *numeric (integer and floating point)* and *char* values and determine whether one is greater than, less than, equal to, or not equal to another.**

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 `==`

Confusing Equality (==) and Assignment (=) Operators

- Common error
  - Does not typically cause syntax errors
- Aspects of problem
  - Expressions that have a value can be used for decision
    - Zero = false, nonzero = true
  - Assignment statements produce a value (the value to be assigned)

Confusing Equality (==) and Assignment (=) Operators

- Example

```
if ( payCode == 4 )
   cout << "You get a bonus!" << endl;
```

  - If paycode is 4, bonus given

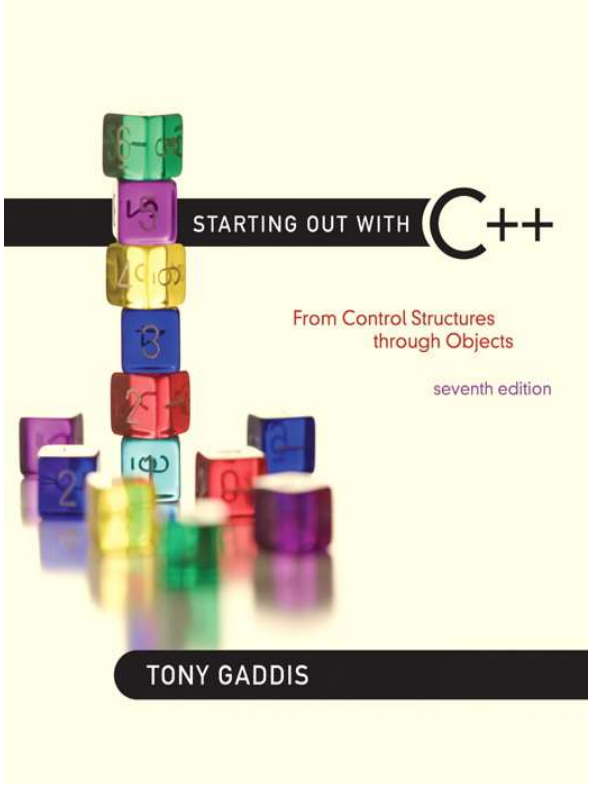
- If == was replaced with =

```
if ( payCode = 4 )
   cout << "You get a bonus!" << endl;
```

  - Paycode set to 4 (no matter what it was before)
  - Statement is true (since 4 is non-zero)
  - Bonus given in every case

Confusing Equality (==) and Assignment (=) Operators

- Lvalues
  - Expressions that can appear on left side of equation
  - Can be changed (I.e., variables)
    - `x = 4;`
- Rvalues
  - Only appear on right side of equation
  - Constants, such as numbers (i.e. cannot write `4 = x;`)
- Lvalues can be used as Rvalues, but not vice versa



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`