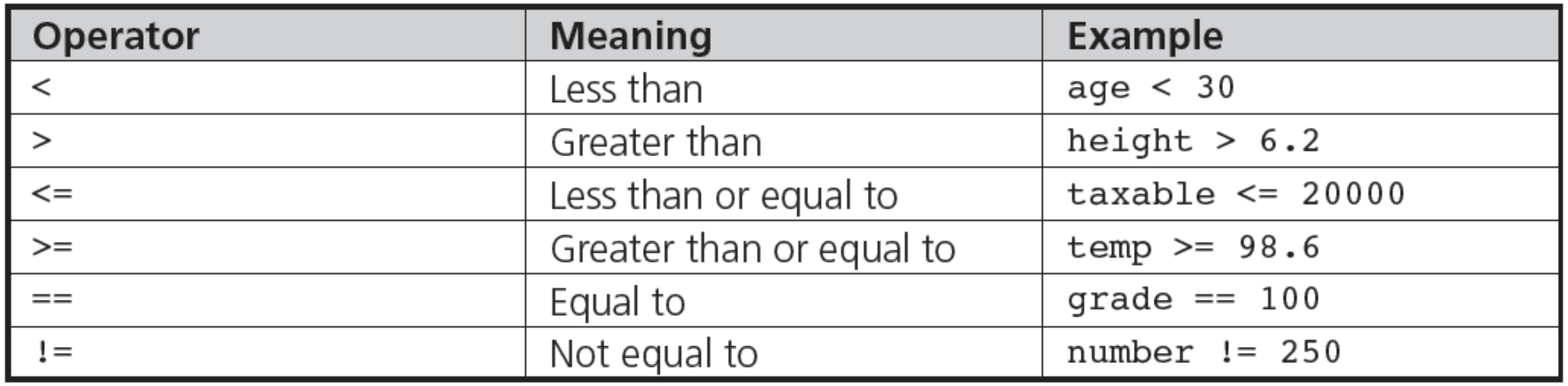
Relational Expressions

- All computers are able to compare numbers
  - Can be used to create an intelligence-like facility
- Relational expressions: expressions used to compare operands
  - Format: a relational operator connecting two variable and/or constant operands
  - Examples of valid relational expressions:

```
Age > 40  length <= 50  flag == done
```

Relational Expressions (cont'd.)

Table 4.1 C++'s Relational Operators



Operator	Meaning	Example
`<`	Less than	`age < 30`
`>`	Greater than	`height > 6.2`
`<=`	Less than or equal to	`taxable <= 20000`
`>=`	Greater than or equal to	`temp >= 98.6`
`==`	Equal to	`grade == 100`
`!=`	Not equal to	`number != 250`

Relational Expressions (cont'd.)

- Relational expressions (conditions):
  - Are evaluated to yield a numerical result
  - Condition that is true evaluates to 1
  - Condition that is false evaluates to 0
- Example:
  - The relationship `2.0 > 3.3` is always false; therefore, the expression has a value of 0

Logical Operators

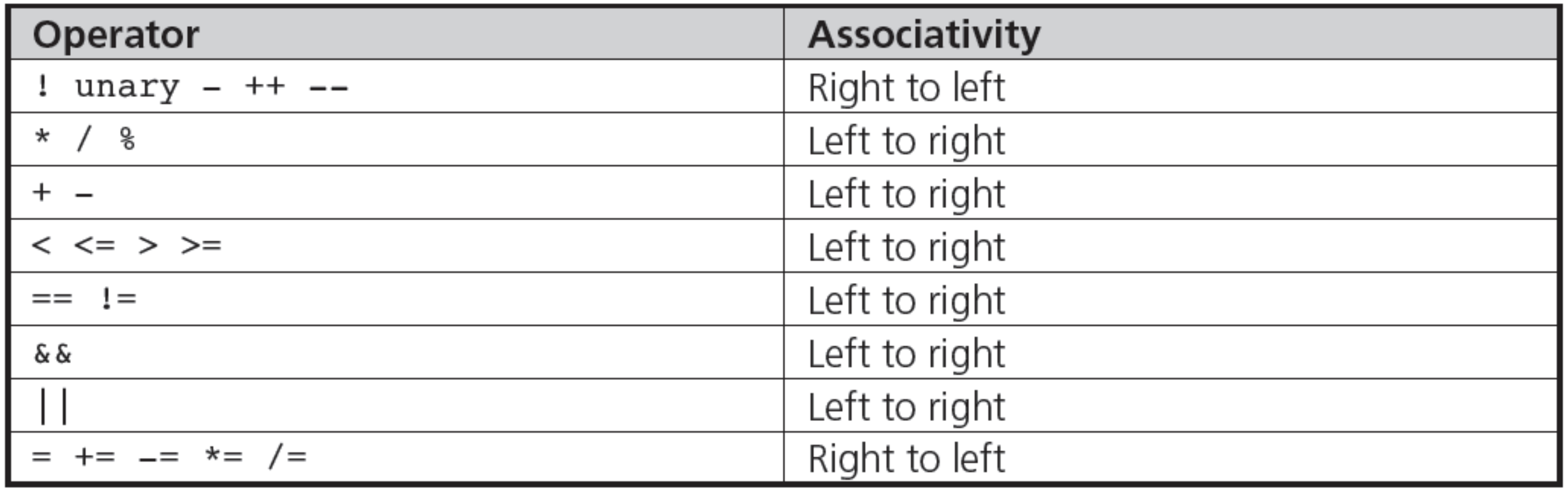
- More complex conditions can be created using logical operators AND, OR, and NOT
  - Represented by the symbols: `&&`, `||`, `!`
- AND operator, `&&`:
  - Used with two simple expressions
  - Example: `(age > 40) && (term < 10)`
  - Compound condition is true (has value of 1) only if `age` > 40 and `term` < 10

Logical Operators (cont'd.)

- OR operator, `||`:
  - Used with two simple expressions
  - Example: `(age > 40) || (term < 10)`
  - Compound condition is true if `age` > 40 or if `term` < 10 or if both conditions are true
- NOT operator, `!`:
  - Changes an expression to its opposite state
  - If `expression` is true, then `!expression` is false

Logical Operators (cont'd.)

Table 4.2 Operator Precedence and Associativity



Operator	Associativity
`! unary - ++ --`	Right to left
`* / %`	Left to right
`+ -`	Left to right
`< <= > >=`	Left to right
`== !=`	Left to right
`&&`	Left to right
`||`	Left to right
`= += -= *= /=`	Right to left

A Numerical Accuracy Problem

- Avoid testing equality of single- and double-precision values and variables using `==` operator
  - Tests fail because many decimals cannot be represented accurately in binary
- For real operands:
  - The expression `operand_1 == operand_2`

    should be replaced by:

    `abs(operand_1 - operand_2) < EPSILON`
  - If this expression is true for very small `EPSILON`, then the two operands are considered equal