Computer Science I

CS 135

5. Selection: If and Switch Controls

René Doursat

Department of Computer Science & Engineering
University of Nevada, Reno

Fall 2005

Computer Science I

CS 135

0. Course Presentation
1. Introduction to Programming
2. Functions I: Passing by Value
3. File Input/Output
4. Predefined Functions
5. If and Switch Controls
6. While and For Loops
7. Functions II: Passing by Reference
8. 1-D and 2-D Arrays

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. Logical Expressions

c. If / Else Selection Structures

d. Switch Selection Structures

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. **Control Structures**

- ✓ What are control structures?
- ✓ Selection structures
- ✓ Repetition structures (next week)

b. **Logical Expressions**

c. **If / Else Selection Structures**

d. **Switch Selection Structures**

5.a Control Structures

What are control structures?

- Reminder: there are six basic computer operations

1. a computer can receive information (Get, Read, etc.)
2. a computer can put out information (Display, etc.)
3. a computer can perform arithmetic (Add, Divide, etc.)
4. a computer can assign a value to a variable or memory location (Set, Initialize, etc.)
5. a computer can compare variables and select one of two alternate actions → *selection structures*
6. a computer can repeat a group of actions → *repetition structures*

5.a Control Structures

What are control structures?

Structure theorem

- ✓ it is possible to write any computer program by using only three basic control structures that are easily represented in pseudocode:
  - sequence structures
  - selection structures
  - repetition structures

introduce branching ("jumps") in the sequential logic

Sequence structures

- ✓ straightforward execution of one processing step after another
- ✓ sequence of pseudocode statements: do this, do that, then this, then that, etc.

5.a Control Structures

What are control structures?

- ➢ Selection structures
  - ✓ condition and choice between two actions, depending on whether the condition is true or false
  - ✓ represented by the pseudocode keywords IF, THEN, ELSE, and ENDIF
- ➢ Repetition structures
  - ✓ block of statements to be executed repeatedly, as long as a condition is true
  - ✓ represented by the pseudocode keywords WHILE and ENDWHILE

5.a Control Structures

What are control structures?

- Sequence, selection and repetition structures

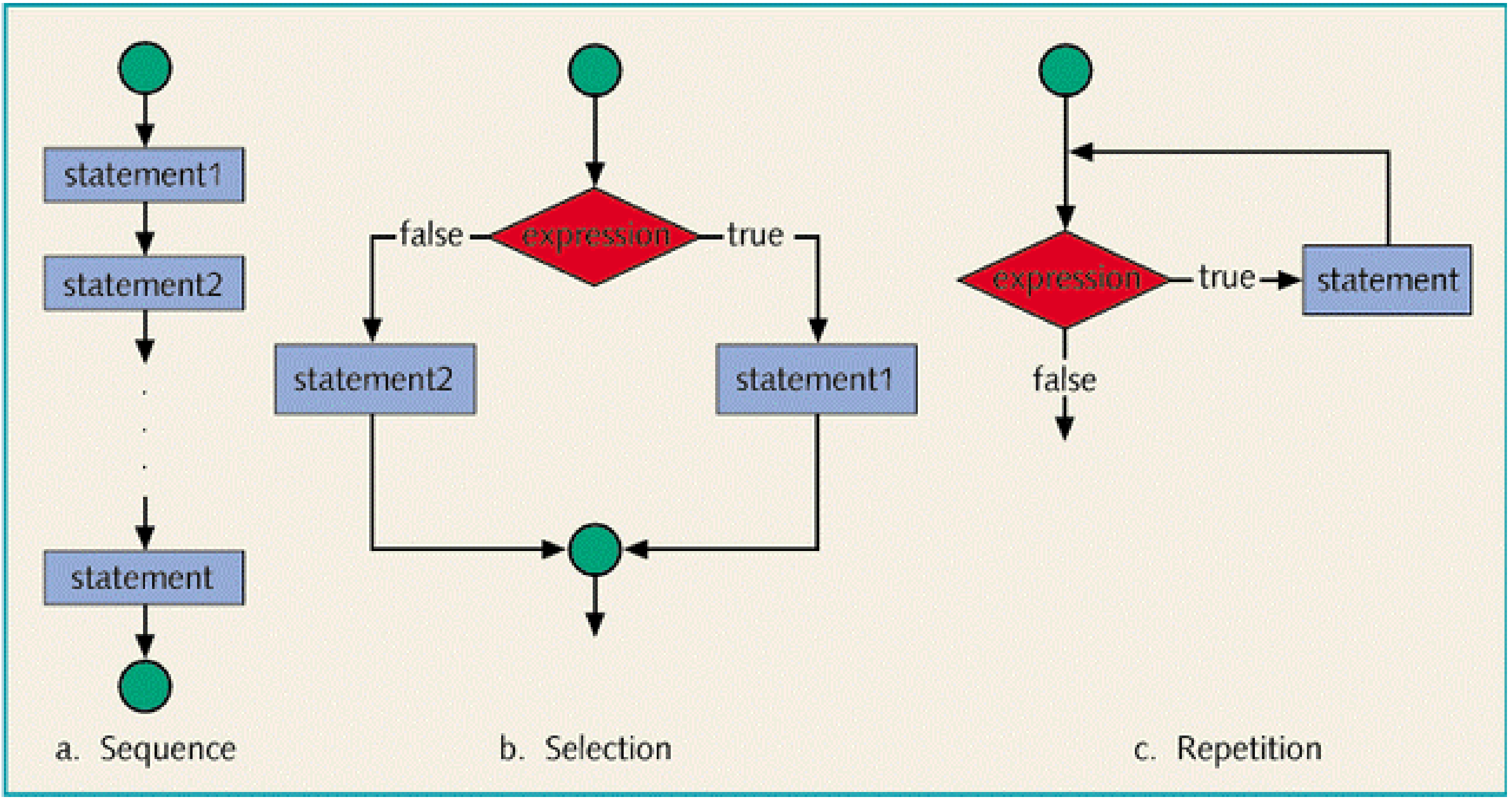


Figure 4-1 Flow of execution

5.a Control Structures

Selection structures

- A computer can compare variables and select one of two alternate actions → *selection structures*
  - ✓ examples:
    - one-way – if it starts to rain, go inside the building
    - two-way – if the car starts, drive; otherwise, take the bus
  - ✓ pseudocode examples:

```
IF age >= 12 THEN
    Prompt for entrance fee
ENDIF
```

One-way selection

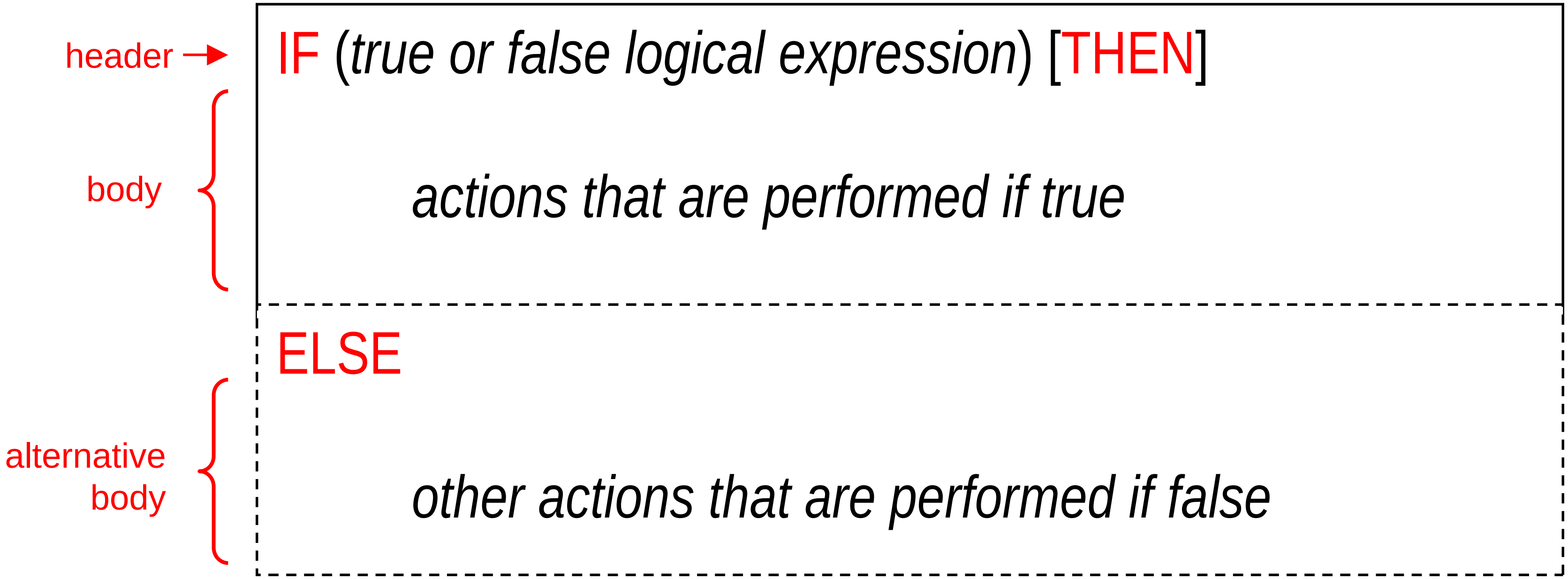
```
IF student is female THEN
    Add 1 to female count
ELSE
    Add 1 to male count
ENDIF
```

Two-way selection

5.a Control Structures

Selection structures

- Anatomy of an if /else selection structure (pseudocode)
  - ✓ the "header" is a logical expression
  - ✓ the "body" contains actions that are performed (or not) depending on the header



header →

IF (*true or false logical expression*) [THEN]

body

actions that are performed if true

ELSE

alternative body

other actions that are performed if false

Anatomy of an if / else selection structure

5.a Control Structures

Repetition structures (next week)

- A computer can repeat a group of actions

  → *repetition structures*

  - ✓ examples:
    - calculate 100 student grades
    - pour water in the saucepan until it is full
    - cook the pasta until it is “al dente”
  - ✓ pseudocode example:

```
WHILE water_level < pan_height
    Add 1 tablespoon to water_volume
    water_level = water_volume / pan_surface
ENDWHILE
```

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. **Control Structures**
   - ✓ What are control structures?
   - ✓ Selection structures
   - ✓ Repetition structures

b. **Logical Expressions**

c. **If / Else Selection Structures**

d. **Switch Selection Structures**

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. Logical Expressions

- ✓ Relational operators
- ✓ Logical (Boolean) operators
- ✓ Order of precedence

c. If / Else Selection Structures

d. Switch Selection Structures

5.b Logical Expressions

In this section, we look at the header of **`if / else`** selection structures

```
IF (true or false logical expression)
        actions performed if true
ELSE
        actions performed if false
```

5.b Logical Expressions

Relational operators

- Relational operators allow to make comparisons

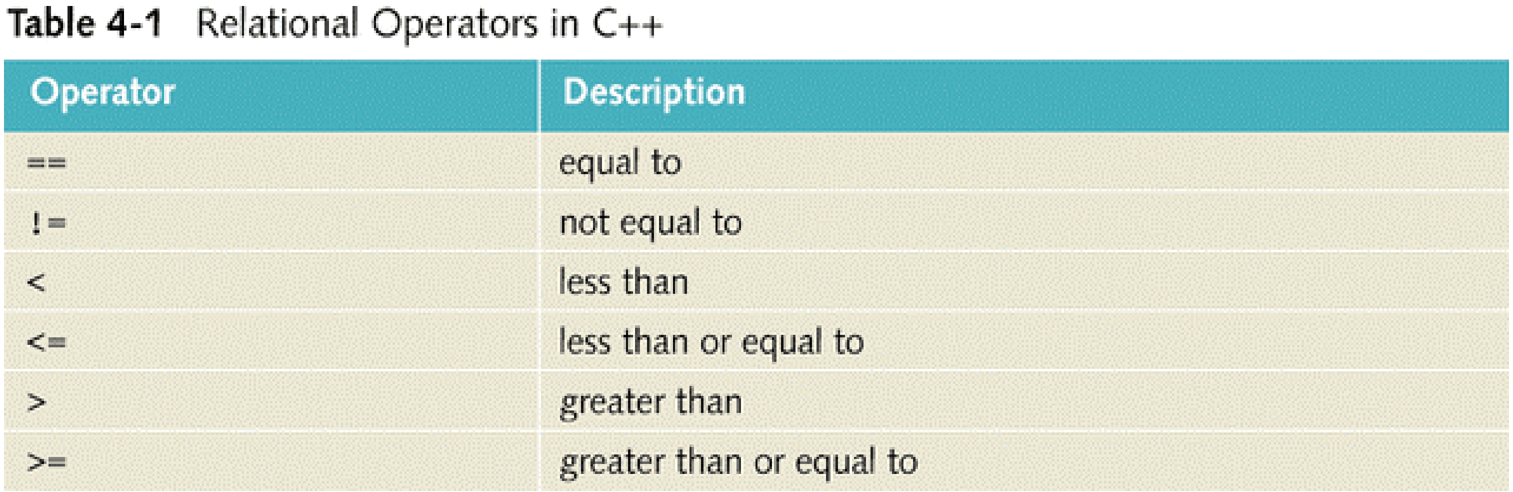


Table 4-1 Relational Operators in C++

Operator	Description
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

- ✓ a relational operator
  - is a binary operator: it takes two numeric operands
  - yields a boolean result, `true` or `false`
  - false also evaluates as 0 and true evaluates as nonzero

5.b Logical Expressions

Relational operators

- Relational operators allow to make comparisons
  - ✓ examples:
    - **`8 < 8.01`** 8 less than 8.01 **`true`**
    - **`6 != 6.0`** 6 not equal to 6.0 **`false`**
    - **`7 >= 7`** 7 greater than or equal to 7 **`true`**
    - **`'a' == 'b'`** char ‘a’ equal to ‘b’ **`false`**
  - ✓ unlike the assignment operator **`=`** it is ok to have expressions on *both* sides of a relational operator, for example if **`x`** is 6:
    - **`(7 + x/5.0) > (x + 2.1)`** is **`true`**
  - ✓ because of precision problems, use caution when equating floating-point expressions (using **`==`**)
    - **`2.0/7 + 5.0/7 == 1.0`** is likely to be **`false`**

5.b Logical Expressions

Relational operators

- Relational operators allow to make comparisons
  - ✓ relational operators are strictly **binary**
    - ~~`0 <= x < 10`~~ is illegal syntax
    - it must be written:

      `(0 <= x) && (x < 10)`
  - ✓ the operator **`&&`** (“and”) is a logical operator; we will look at logical operators in the next slides
  - ✓ caution when comparing different data types, such as numbers and characters: **`8 < '5'`** is **`true`**!
  - ✓ do not compare strings using relational operators

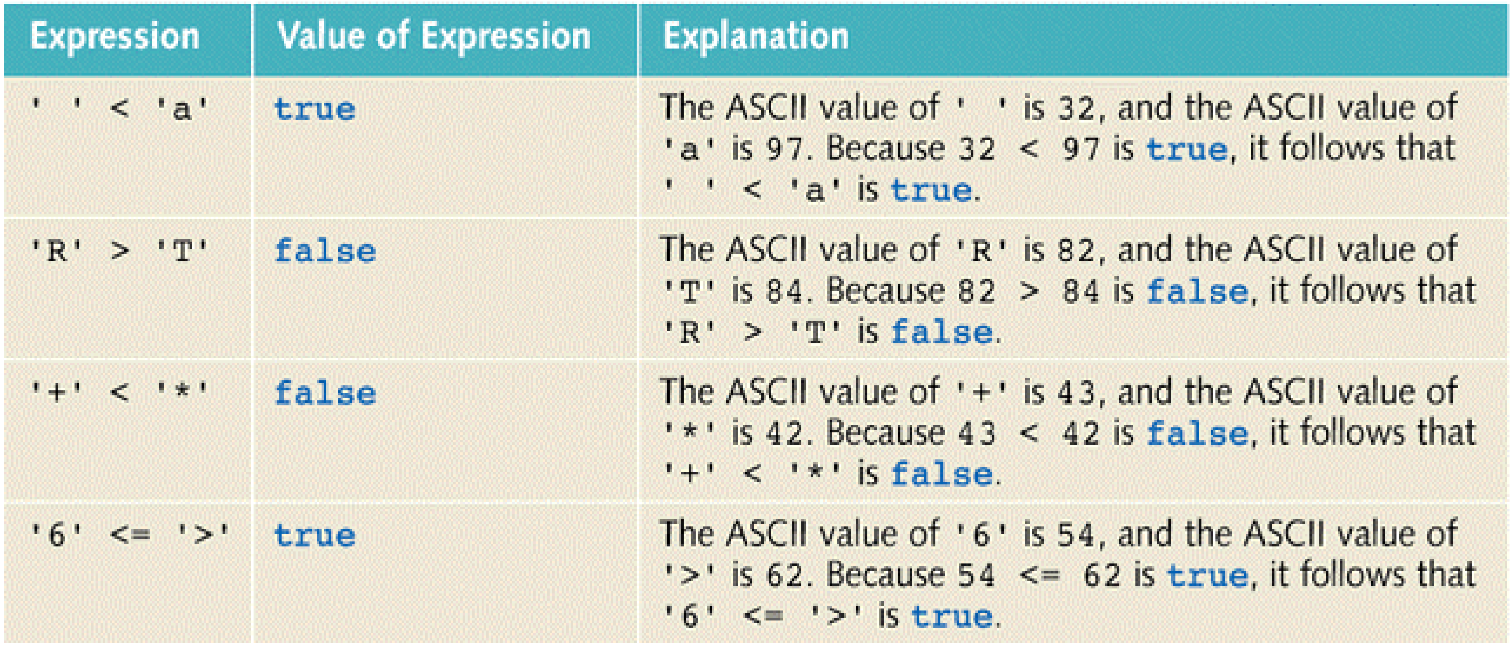
    ~~`"apple" <= "orange"`~~ is illegal syntax

5.b Logical Expressions

Relational operators

- Comparing characters
  - ✓ relational operators compare the characters' ASCII codes

Table 4-2 Evaluating Expressions Using Relational Operators and the ASCII Collating Sequence



Expression	Value of Expression	Explanation
`' ' < 'a'`	`true`	The ASCII value of `' '` is 32, and the ASCII value of `'a'` is 97. Because `32 < 97` is `true`, it follows that `' ' < 'a'` is `true`.
`'R' > 'T'`	`false`	The ASCII value of `'R'` is 82, and the ASCII value of `'T'` is 84. Because `82 > 84` is `false`, it follows that `'R' > 'T'` is `false`.
`'+' < '*'`	`false`	The ASCII value of `'+'` is 43, and the ASCII value of `'*'` is 42. Because `43 < 42` is `false`, it follows that `'+' < '*'` is `false`.
`'6' <= '>'`	`true`	The ASCII value of `'6'` is 54, and the ASCII value of `'>'` is 62. Because `54 <= 62` is `true`, it follows that `'6' <= '>'` is `true`.

5.b Logical Expressions

Logical (Boolean) operators

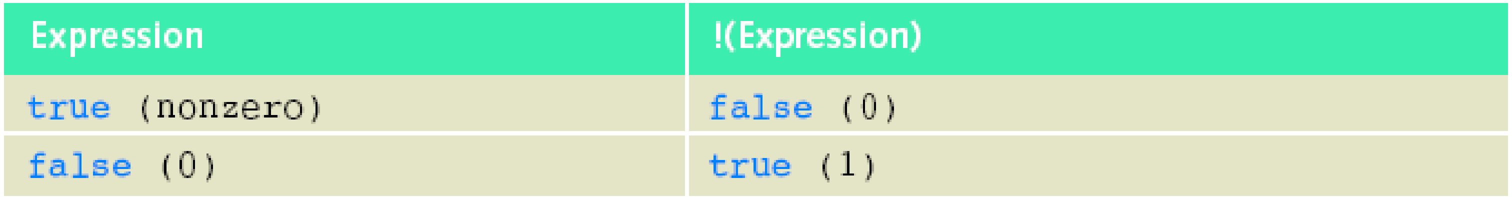
- Logical operators allow to combine logical expressions
  - there are 3 main logical operators in C++
    - **&&**   the binary ‘and’ operator
    - **||**   the binary ‘or’ operator
    - **!**   the unary ‘not’ operator
  - each logical operator
    - takes only logical values, **true** and **false**, as operands
    - yields only a logical value, **true** and **false**, as a result

5.b Logical Expressions

Logical (Boolean) operators

- The **!** ('not') operator
  - ✓ reverses the value of its logical operand

Table 4-5 The ! (not) Operator



Expression	!(Expression)
`true (nonzero)`	`false (0)`
`false (0)`	`true (1)`

- ✓ examples:

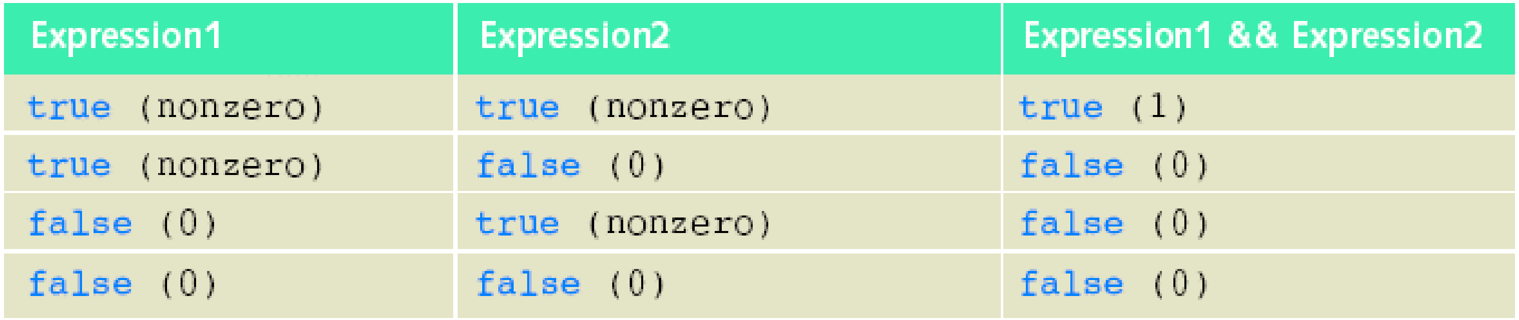
`!(8 > 15)`	not (8 is greater than 15)	`true`
`!(6 == 6)`	not (6 is equal to 6)	`false`
`!('a'>'b')`	not ('a' is greater than 'b')	`true`

5.b Logical Expressions

Logical (Boolean) operators

- The && ('and') operator
  - ✓ is true if and only if both of its logical operands are true

Table 4-6 The && (and) Operator



Expression1	Expression2	Expression1 && Expression2
true (nonzero)	true (nonzero)	true (1)
true (nonzero)	false (0)	false (0)
false (0)	true (nonzero)	false (0)
false (0)	false (0)	false (0)

- ✓ examples: *one <u>false</u> operand is enough to yield <u>false</u>*
  - `(14 >= 5) && ('A' == 'B')` false
  - `(14 != 5) && ('A' < 'B')` true
  - `(14 < 5) && !('$' >= '*')` false
  - `!(14 < 5) && 3` *?? never mind* true

5.b Logical Expressions

Logical (Boolean) operators

- Boolean algebra with && ('and')
  - ✓ important equivalences among expressions containing &&

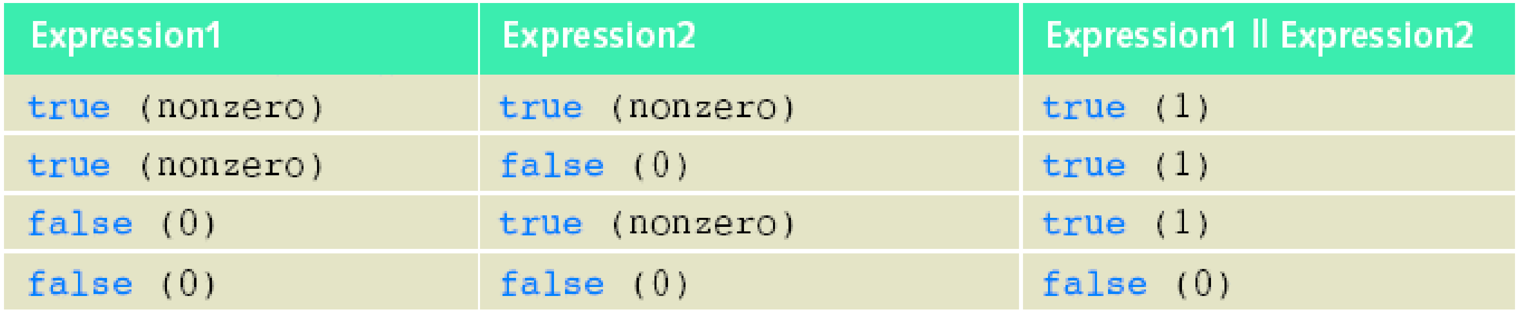
Logical expression	Equivalent expression
`x && true`	`x`
`x && false`	`false`
`x && x`	`x`
`x && !x`	`false`
`x && y`	`y && x`
`x && (y && z)`	`(x && y) && z`

5.b Logical Expressions

Logical (Boolean) operators

- The || ('or') operator
  - ✓ is true if at least one of its logical operand is true

Table 4-7 The || (or) Operator



Expression1	Expression2	Expression1 || Expression2
`true (nonzero)`	`true (nonzero)`	`true (1)`
`true (nonzero)`	`false (0)`	`true (1)`
`false (0)`	`true (nonzero)`	`true (1)`
`false (0)`	`false (0)`	`false (0)`

- ✓ examples: *one true operand is enough to yield true*
  - `(14 == 5) || ('A' <= 'B')` **true**
  - `!(14 != 5) || ('A' == 'B')` **false**
  - `(14 > 5) || !('$' >= '*')` **true** *?? never mind*
  - `!(14 > 5) || 0` **false**

5.b Logical Expressions

Logical (Boolean) operators

- Boolean algebra with || ('or')
  - ✓ important equivalences among expressions containing ||

Logical expression	Equivalent expression
`x || true`	`true`
`x || false`	`x`
`x || x`	`x`
`x || !x`	`true`
`x || y`	`y || x`
`x || (y || z)`	`(x || y) || z`

5.b Logical Expressions

Logical (Boolean) operators

- Boolean algebra with && and ||
  - ✓ important equivalences among expressions with && and ||

Logical expression	Equivalent expression
`x || (y && z)`	`(x || y) && (x || z)`
`x || (x && z)`	`x`
`x && (y || z)`	`(x && y) || (x && z)`
`x && (x || z)`	`x`
`!(x || y)`	`!x && !y`
`!(x && y)`	`!x || !y`

- ✓ *De Morgan's laws*

5.b Logical Expressions

Order of precedence

- How to evaluate complex logical expressions
  - ✓ expressions can mix arithmetic, relational and logical operators:
    - `!(5 + 3 <= 9) || 6 < 15 && 7 != 8`
  - ✓ evaluation follows a priority scheme

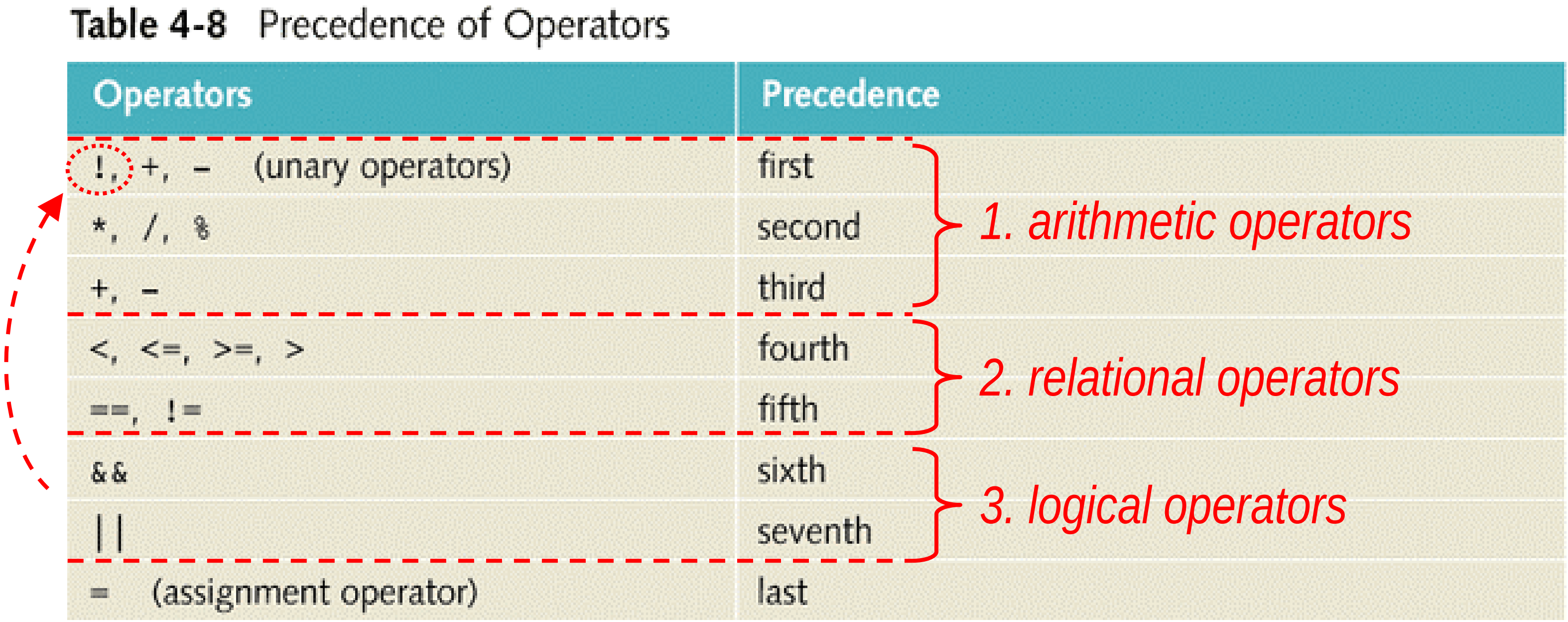


Table 4-8 Precedence of Operators

Operators	Precedence
!, +, - (unary operators)	first
*, /, %	second
+, -	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
||	seventh
= (assignment operator)	last

1. arithmetic operators

2. relational operators

3. logical operators

5.b Logical Expressions

Order of precedence

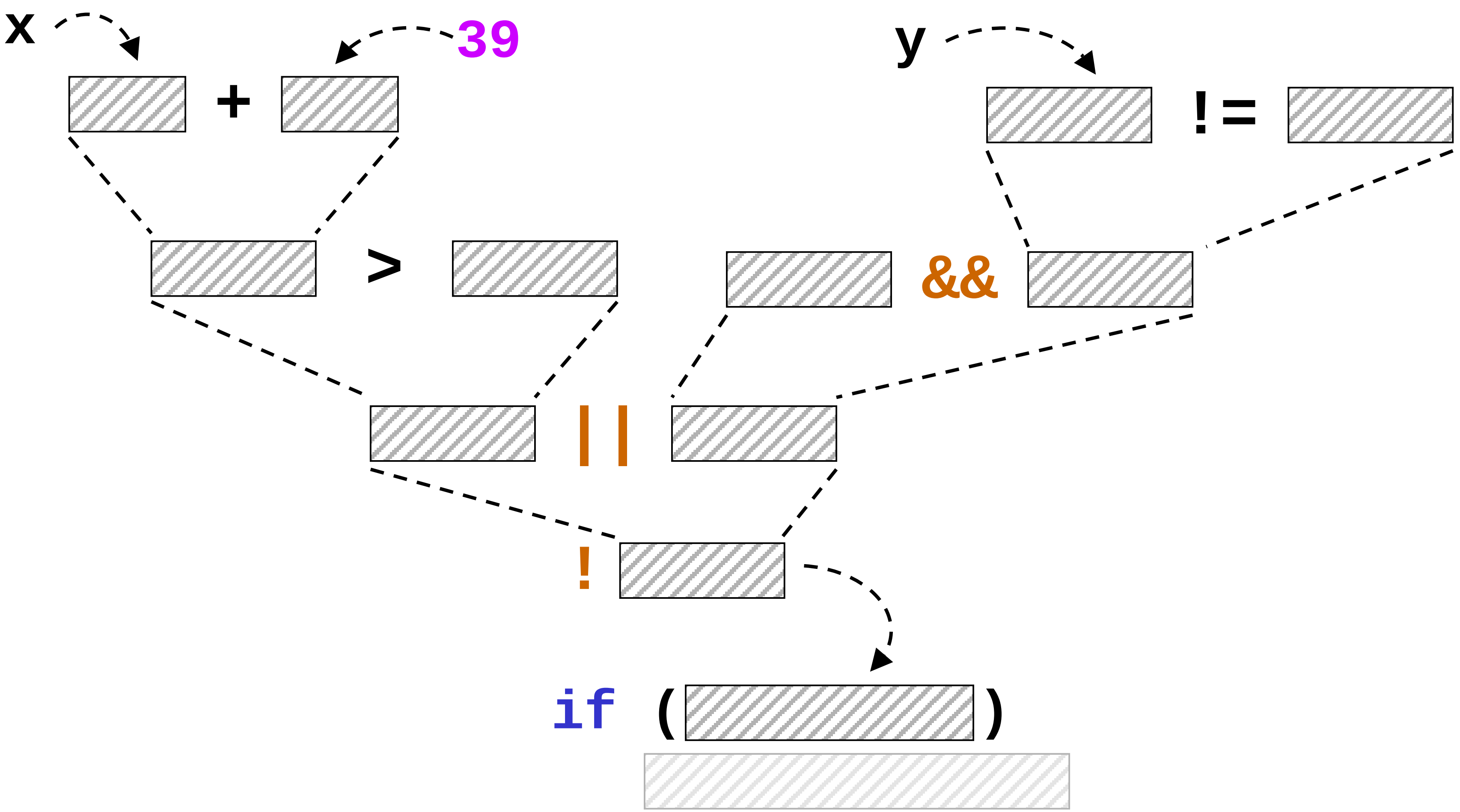
- Example of precedence in logical expressions
  - ✓ `bool found = true;`
  - ✓ `double x = 5.2, y = 3.4;`
  - ✓ `int a = 5, b = 8, n = 0;`
  - ✓ `char ch = '$';`

Logical expression	Value
`!found && x >= 0`	`false`
`!(found || x < 0)`	`false`
`x + y <= 20.5`	`true`
`n < 1 || n > 100`	`true`
`'A' <= ch && ch <= 'Z' || 'a' <= ch && ch <= 'z'`	`false`
`a + 2 <= b && found`	`true`

5.b Logical Expressions

Order of precedence

- Building expressions is like a construction game



Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. **Logical Expressions**

- ✓ Relational operators
- ✓ Logical (Boolean) operators
- ✓ Order of precedence

c. **If / Else Selection Structures**

d. **Switch Selection Structures**

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. Logical Expressions

c. If / Else Selection Structures

- ✓ One-way selection structure: `if`
- ✓ Two-way selection structure: `if … else`
- ✓ Compound statement selection structures
- ✓ Nested selection structures
- ✓ Conditional operator

d. Switch Selection Structures

5.c If / Else Selection Structures

One-way selection structure: `if`

- A one-way selection decides whether to execute a statement or not

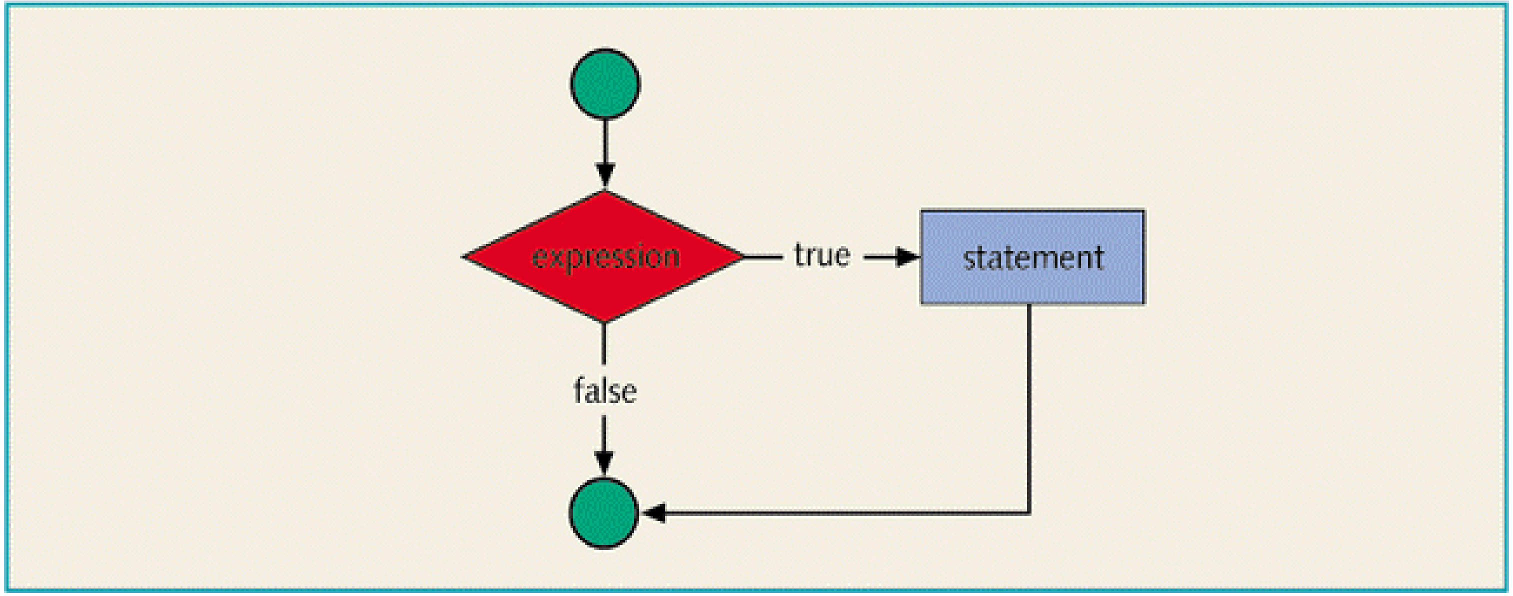


Figure 4-2 One-way selection

5.c If / Else Selection Structures

One-way selection structure: `if`

- Syntax of a one-way selection

```
    if (expression)
         statement
```

```
if (age >= 12)
    pay_entrance();
```

- ✓ **`if`** is a reserved keyword
- ✓ *expression* is a <u>logical expression</u>
  - sometimes called a “decision maker” because it decides whether to execute the statement that follows it or not
- ✓ *statement* follows *expression* and can be <u>any C++ statement</u>
  - sometimes called the “action statement”
  - *statement* is executed if the value of *expression* is true
  - *statement* is bypassed if the value is false: the program goes to the next statement directly

5.c If / Else Selection Structures

Two-way selection structure: `if` … `else`

- A two-way selection decides whether to execute one statement or another

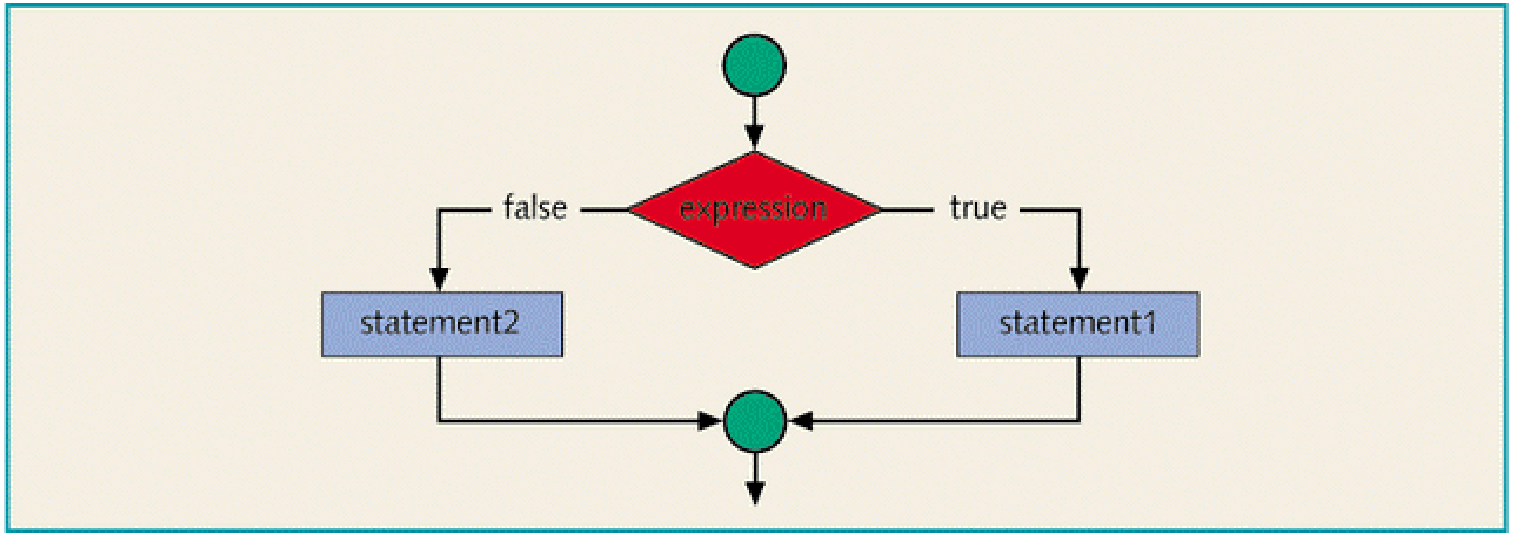


Figure 4-3 Two-way selection

5.c If / Else Selection Structures

Two-way selection structure: **if … else**

- Syntax of a two-way selection

```
if (expression)
    statement1
else
    statement2
```

```
if (age >= 12)
    pay(8.00);
else
    pay(3.50);
```

- ✓ **else** is also a reserved keyword
- ✓ *expression* is a logical expression
- ✓ *statement1* and *statement2* and can be any C++ statements
  - *statement1* is executed if the value of *expression* is true
  - *statement2* is executed if the value is false
  - after that, if there was not failure or early exit, the program goes to the next statement after the if / else structure

5.c If / Else Selection Structures

Compound statement selection structures

- Compound statement
  - ✓ the body of an if/else structure can contain multiple C statements
  - ✓ a block of statements is called a “compound statement” and must be surrounded with curly braces { }

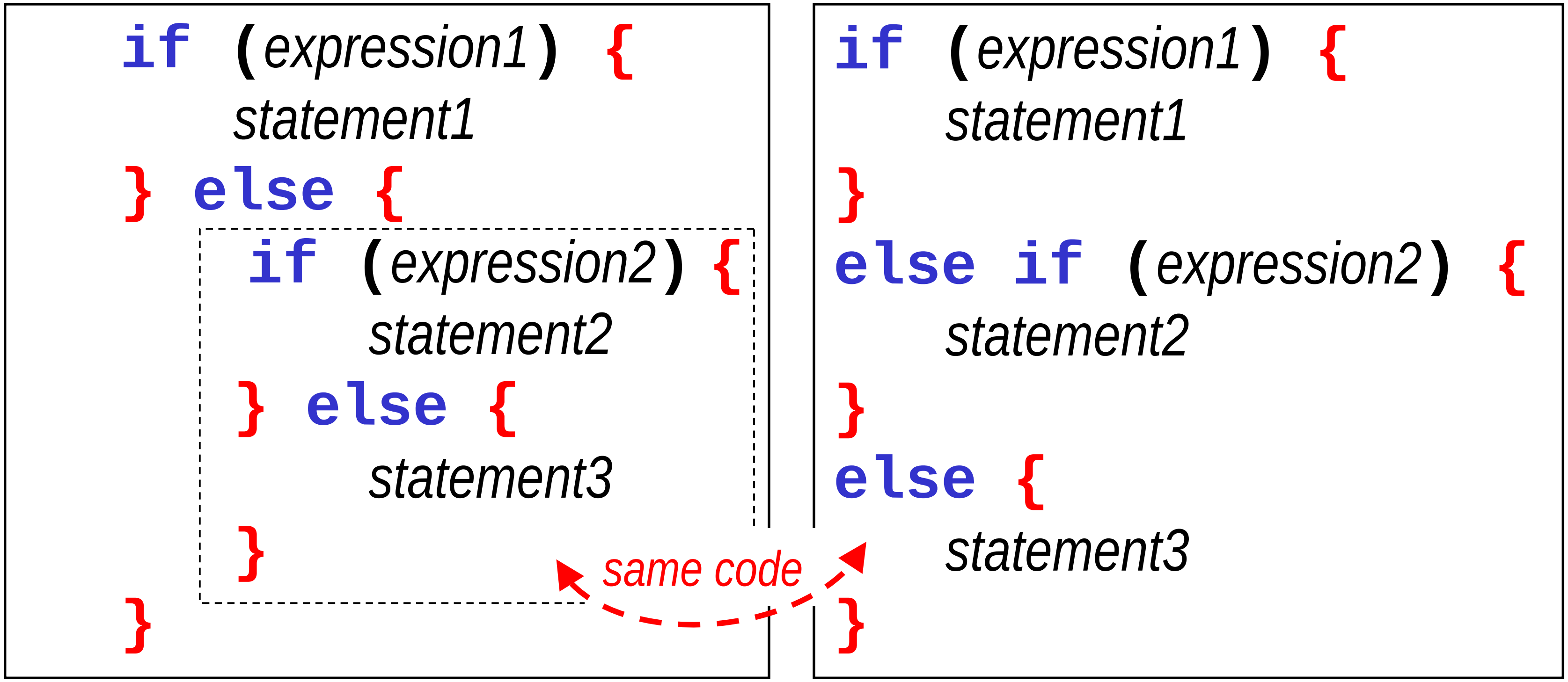
```
if (expression) {
    statement1
    statement2
    statement3
}
else {
    statement4
    statement5
}
```

```
if (age >= 12) {
    cout << "adult";
    pay(8.00);
    . . .
}
else {
    cout << "child";
    pay(3.50);
}
```

5.c If / Else Selection Structures

Nested selection structures

- If/else structures can be inserted inside other if/else structures
  - ✓ some statements inside the body of a selection structure can themselves be if/else selection structures



```
if (expression1) {
      statement1
} else {
    if (expression2) {
          statement2
    } else {
          statement3
    }
}
```

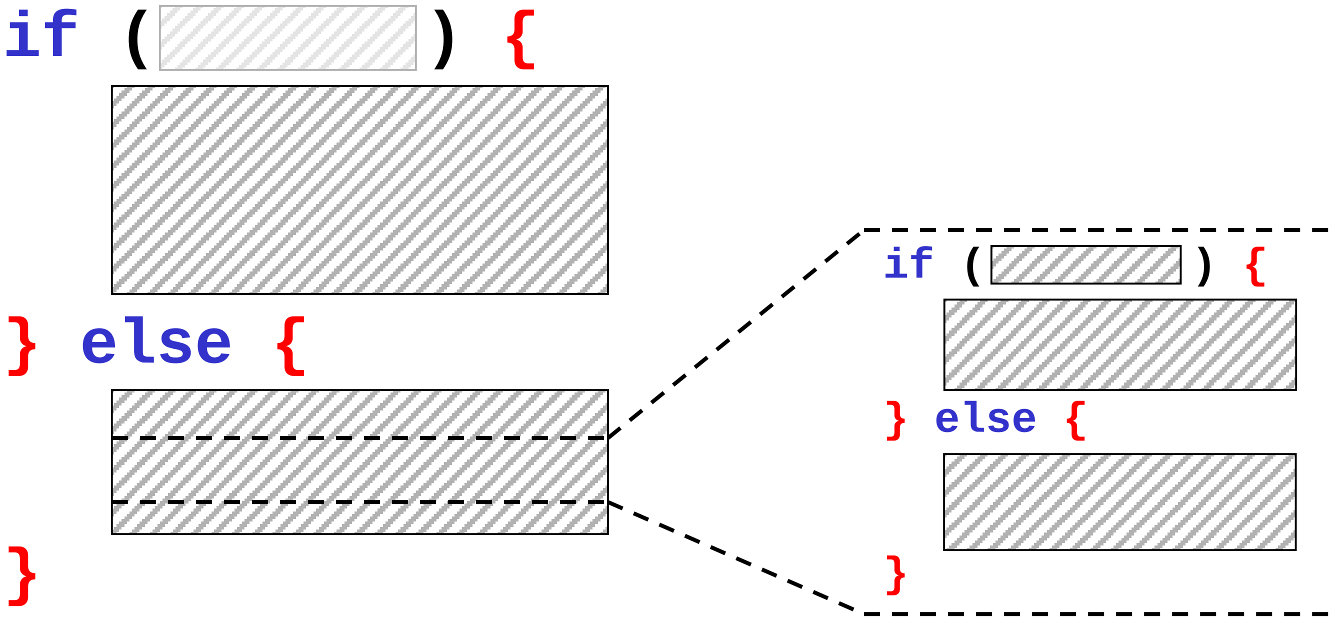
same code

```
if (expression1) {
      statement1
}
else if (expression2) {
      statement2
}
else {
      statement3
}
```

5.c If / Else Selection Structures

Nested selection structures

- ➢ Programming is like a construction game
  - ✓ control structures can be nested in other control structures



```
if (  ) {

} else {

}
```

```
if (  ) {

} else {

}
```

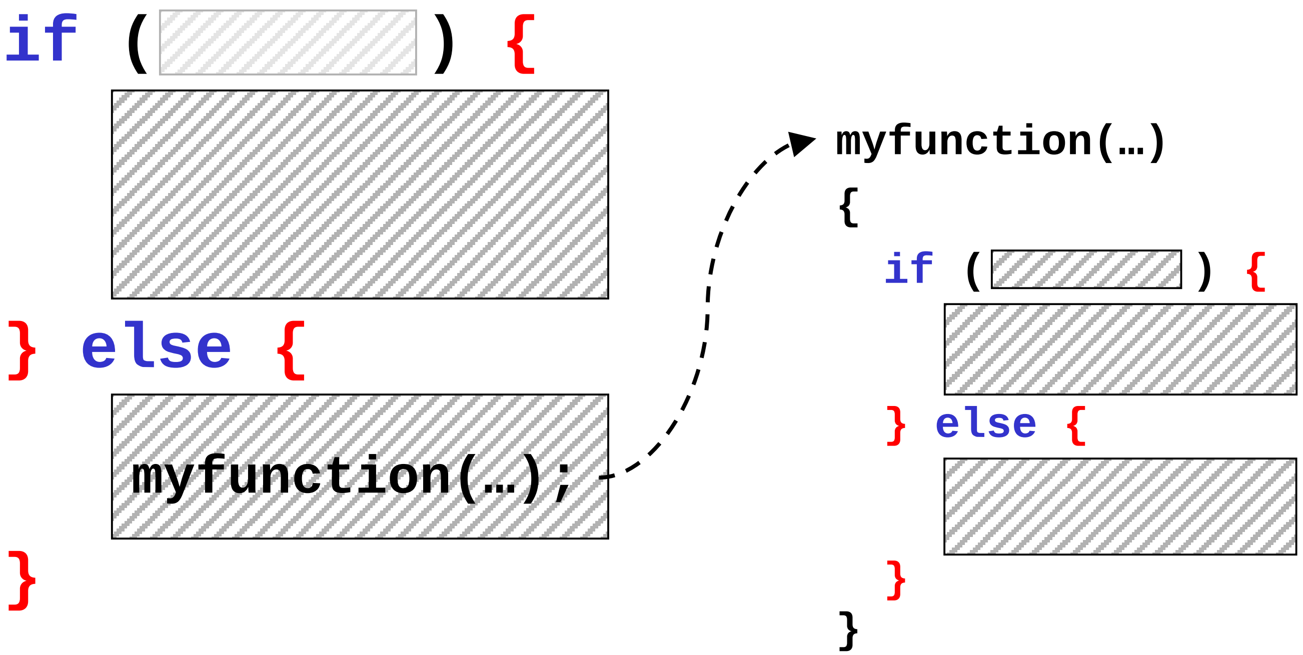
→ *however, too much nesting inside the same area of code is not good programming practice*

5.c If / Else Selection Structures

Nested selection structures

- Programming is like a construction game

→ *breaking up into FUNCTIONS is better practice*



```
if (   ) {

} else {
    myfunction(…);
}
```

```
myfunction(…)
{
   if (   ) {

   } else {

   }
}
```

✓ it will generally depend on the size of a compound statement: if it gets too big, cut it out and put it in a function

5.c If / Else Selection Structures

Conditional operator

- The conditional operator is a one-line shortcut for `if`
  - ✓ the conditional operator is used exclusively for conditional assignment statements involving the same variable
  - ✓ instead of

```
if (expression) {
    x = value1;
} else {
    x = value2;
}
```

  - ✓ you can write

```
x = (expression) ? value1 : value2;
```

using the question mark **?** and colon **:** symbols

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. Logical Expressions

c. If / Else Selection Structures

- ✓ One-way selection structure: `if`
- ✓ Two-way selection structure: `if / else`
- ✓ Compound statement selection structures
- ✓ Nested selection structures
- ✓ Conditional operator

d. Switch Selection Structures

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. Logical Expressions

c. If / Else Selection Structures

d. **Switch Selection Structures**

- ✓ Switch syntax and rules
- ✓ Typical switch examples

5.d Switch Selection Structures

Switch syntax and rules

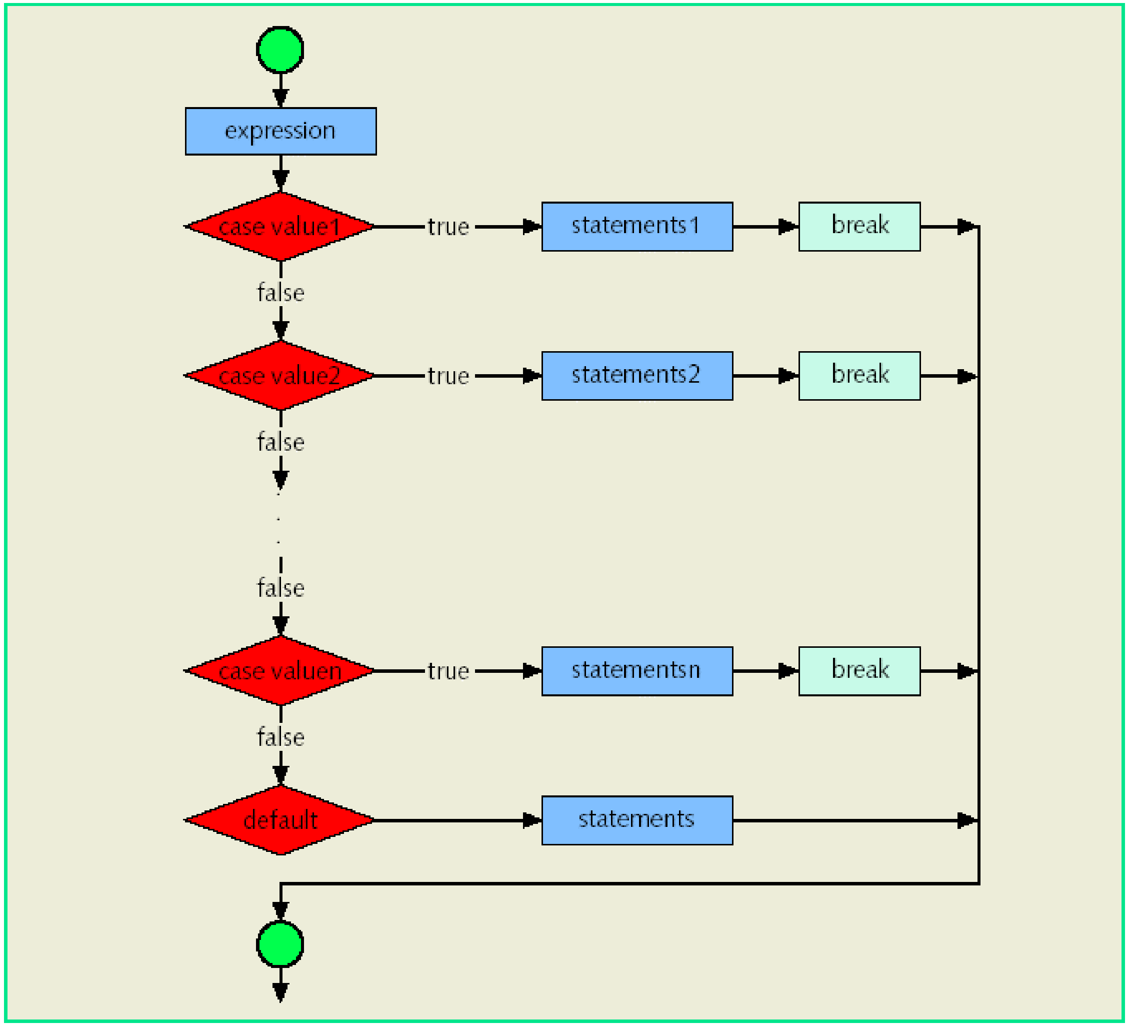


Figure 4-4 `switch` statement

5.d Switch Selection Structures

Switch syntax and rules

- A switch structure can replace multiple nested if/else
  - ✓ it is used exclusively in the case where the same <u>integral</u> expression or variable can evaluate to <u>multiple</u> constant values

```
if (expr == val1)
    statement1
else if (expr == val2)
    statement2
else if (expr == val3)
    statement3
else
    statement0
```

same code

```
switch (expr) {
case val1: statement 1
    break;
case val2: statement 2
    break;
case val3: statement 3
    break;
default: statement0
    break;
}
```

5.d Switch Selection Structures

Switch syntax and rules

- (Fun?) facts about switch selection structures
  - ✓ a switch selection structure uses four special keywords: **`switch`**, **`case`**, **`default`**, **`break`**
  - ✓ the expression in the header is evaluated first and can only yield an integer value
  - ✓ the value of the expression determines which corresponding statement is selected for execution
  - ✓ each constant case value must appear only once
  - ✓ each case label may be followed by one statement or a compound statement (here, curly braces are not necessary)
  - ✓ the break statement should appear after each statement; if it doesn’t, then the next statement will also be executed

5.d Switch Selection Structures

Switch syntax and rules

- Rules of switch selection structures
  - ✓ when value of the expression is matched against a case value:
    - → *statements execute at that point until a break statement is found or the end of switch structure is reached*
  - ✓ if value of the expression does not match any of the case values:
    - → *statements following the default label execute*
  - ✓ if there is neither a matching value, nor a default label:
    - → *the entire switch statement is skipped*
  - ✓ in any case, wherever it is found, a break statement causes an immediate exit from the switch structure

5.d Switch Selection Structures

Typical switch examples

- Switch example 1: conditional conversion scheme
  - ✓ a switch can convert when there is no simple one-line formula

```
switch (score/10) {
case 0: case 1: case 2: case 3: case 4: case 5:
    grade = 'F';
    break;
case 6:
    grade = 'D';
    break;
case 7:
    grade = 'C';
    break;
case 8:
    grade = 'B';
    break;
case 9: case 10:
    grade = 'A';
    break;
default:
    cout << "Invalid score: " << score;
}
```

5.d Switch Selection Structures

Typical switch examples

- Switch example 2: branching upon user input
  - ✓ a switch can perform different actions depending on user input

```
char answer;

cout << "Please select one option from ...";
cin >> answer;

switch (answer) {
case 'p':
    play_game();
    break;
case 'h':
    display_help();
    break;
case 'q':
    quit();
    break;
default:
    cout << "Invalid selection: " << answer;
}
```

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. Logical Expressions

c. If / Else Selection Structures

d. **Switch Selection Structures**

- ✓ Switch syntax and rules
- ✓ Typical switch examples

Computer Science I

CS 135

5. Selection: If and Switch Controls

a. Control Structures

b. Logical Expressions

c. If / Else Selection Structures

d. Switch Selection Structures

Computer Science I

CS 135

0. Course Presentation
1. Introduction to Programming
2. Functions I: Passing by Value
3. File Input/Output
4. Predefined Functions
5. If and Switch Controls
6. While and For Loops
7. Functions II: Passing by Reference
8. 1-D and 2-D Arrays