- Office Location: room 339
- Email: xidong@pvamu.edu
- Office Hour:

Monday to Wednesday

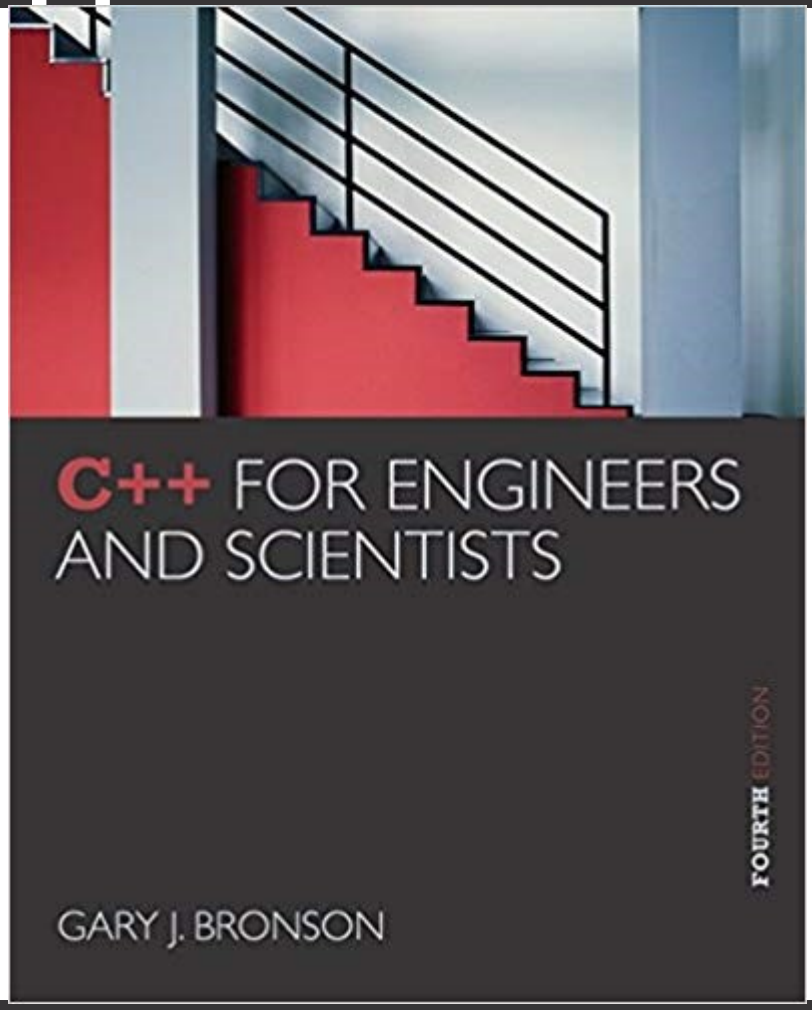
12:00 pm to 15:00 pm

Tuesday

10:00 am to 11:00am

Recommendation: Meeting Appointment with email

ELEG 1043
Computer Applications in Engineering



Chapter 4:
Selection Structures

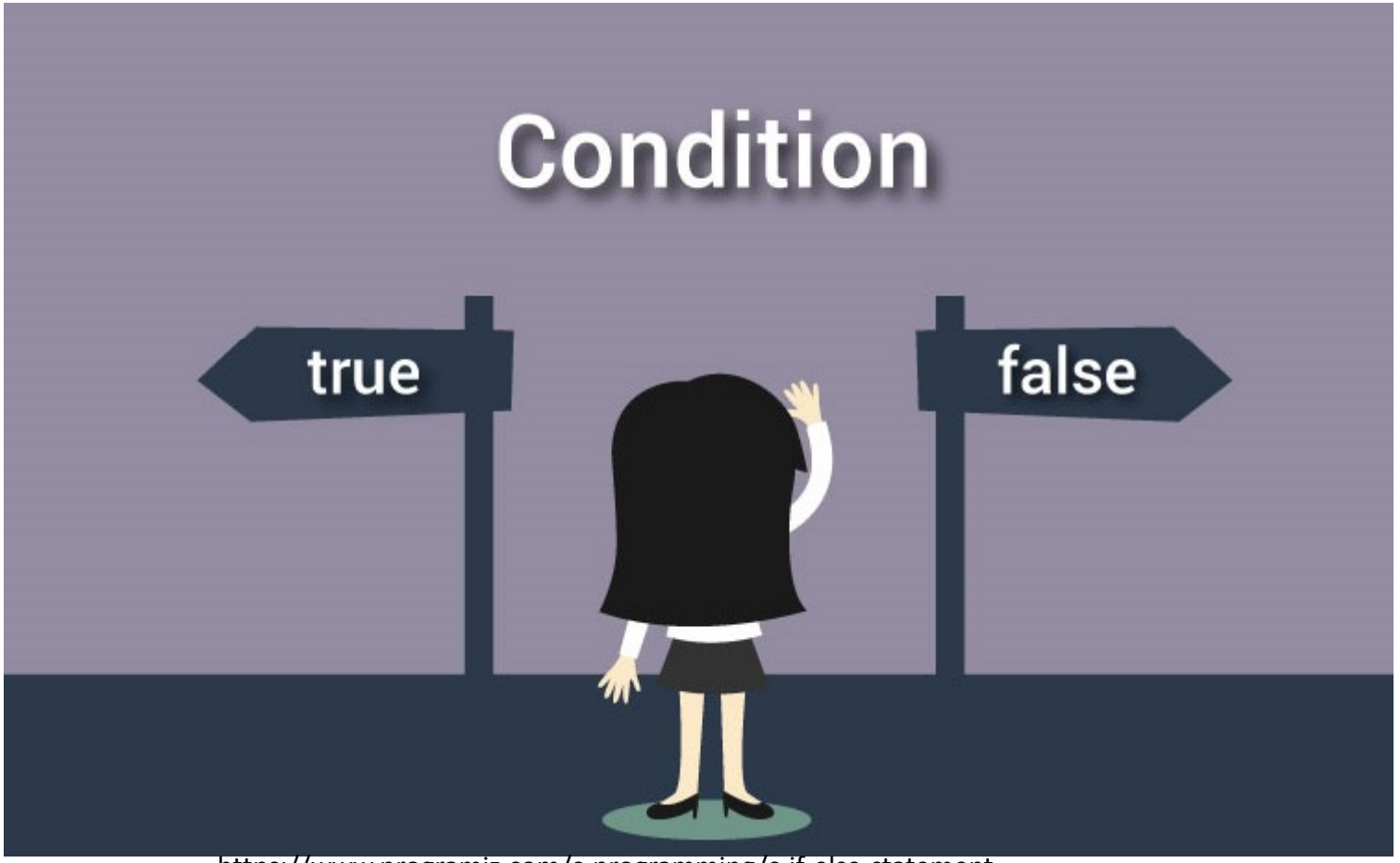
Acknowledgement

- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

Objectives

- In this chapter, you will learn about:
  - Selection criteria
  - The **`if-else`** statement
  - Nested **`if`** statements
  - The **`switch`** statement
  - Program testing
  - Common programming errors

Selection Criteria



https://www.programiz.com/c-programming/c-if-else-statement

Selection Criteria

- **`if-else`** statement: Implements a decision structure for two alternatives

  Syntax:

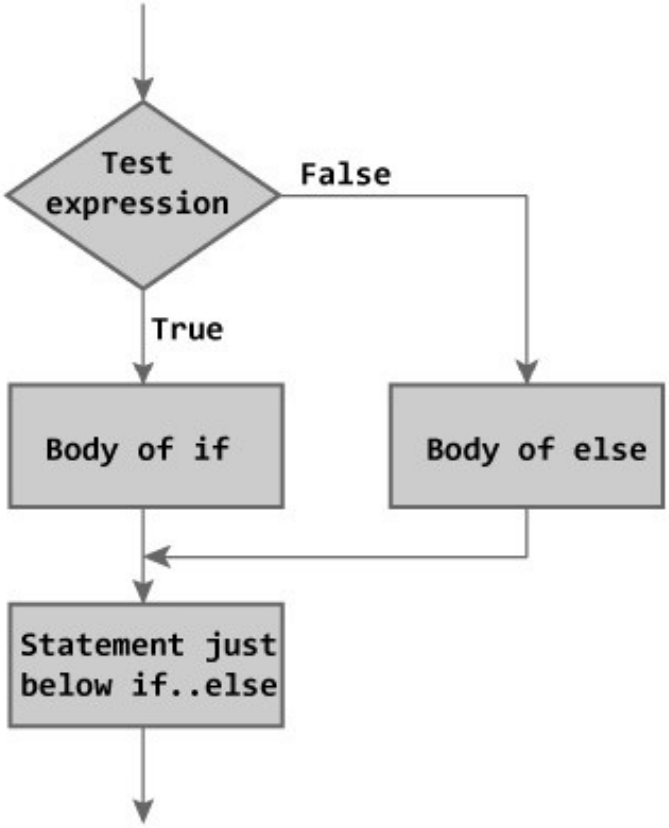
  if (condition)

  statement executed if condition is true;

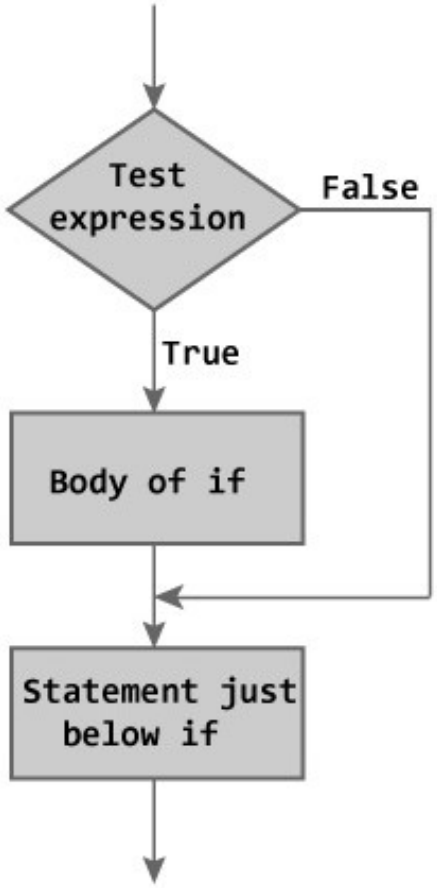
  else

  statement executed if condition is false;

Flowchart of if statement



Flowchart of if statement



Selection Criteria (continued)

- The condition is evaluated to its numerical value:
  - A non-zero value is considered to be **true**
  - A zero value is considered to be **false**
- The **`else`** portion is optional
  - **Executed only if the condition is false**
- The condition may be any valid C++ expression

Relational Operators

- **Relational expression:** Compares two operands or expressions using **relational operators**

Relational 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`

Table 4.1 C++'s Relational Operators

Example 1

```
#include <iostream>
using namespace std;

int main()
{
    int number;
    cout<<"Enter an integer: \n";
    cin>>number;

    // Test expression is true if number is less than 0
    if (number < 0)
    {
      cout<<“You entered "<<number<<“\n”;
    }

    cout<<“The if statement is easy.";
    return 0;
}
```

Example 2

```
#include <iostream>
using namespace std;

int main()
{
    int number;
    cout<<"Enter an integer: \n";
    cin>>number;

    // True if remainder is 0
    if( number%2 == 0 )
        cout<<number<<“  is an even integer.\n";
    else
        cout<<number<<“  is an odd integer.\n";

    return 0;

}
```

Relational Operators (continued)

- Relational expressions are evaluated to a numerical value of 1 or 0 only:
  - If the value is **1**, the expression is **true**
  - If the value is **0**, the expression is **false**
- `char` values are automatically coerced to `int` values for comparison purposes
- Strings are compared on a character by character basis
  - The string with the first lower character is considered smaller

Relational Operators (continued)

- Examples of string comparisons

Expression	Value	Interpretation	Comment
`"Hello"> "Good-bye"`	1	true	The first H in Hello is greater than the first G in Good-bye.
`"SMITH" > "JONES"`	1	true	The first S in SMITH is greater than the first J in JONES.
`"123" > "1227"`	1	true	The third character in 123, the 3, is greater than the third character in 1227, the 2.
`"Behop" > "Beehive"`	1	true	The third character in Behop, the h, is greater than the third character in Beehive, the second e.

Logical Operators

- AND (`&&`): Condition is true only if both expressions are true
- OR (`||`): Condition is true if either one or both of the expressions is true
- NOT (`!`): Changes an expression to its opposite state; true becomes false, false becomes true

A Numerical Accuracy Problem

- Comparing single and double precision values for equality (==) can lead to errors because values are stored in different binary manner.
- Instead, test that the absolute value of the difference is within an acceptable range
  - Example:

```
abs(operandOne - operandTwo) < 0.000001
```

The `if-else` Statement

- **`if-else`** performs instructions based on the result of a comparison
- Place statements on separate lines for readability
- Syntax:

```
if (expression)          <-- no semicolon here
  statement1;
else                     <-- no semicolon here
  statement2;
```

The `if-else` Statement (cont'd)

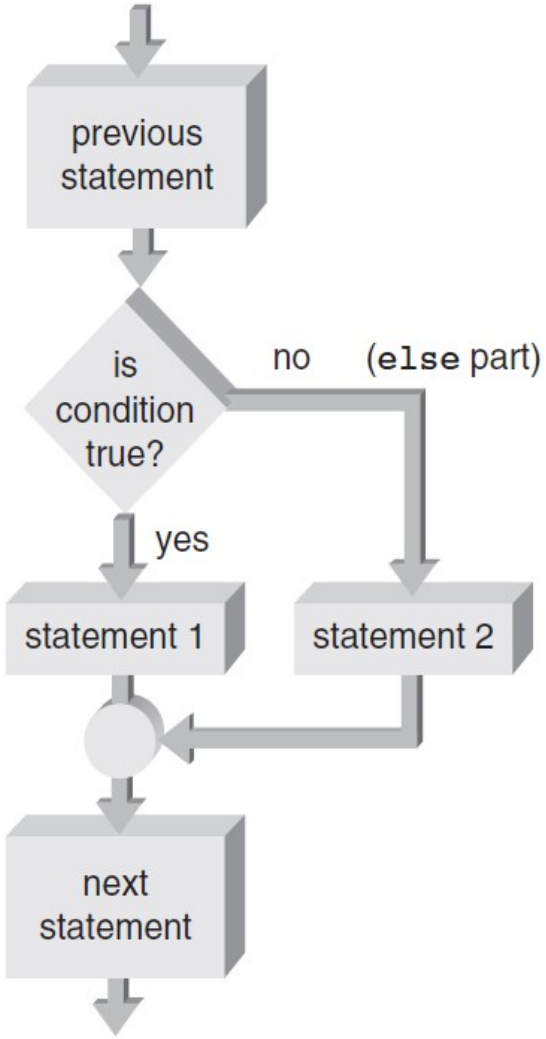


Figure 4.2
The `if-else` flowchart

The `if-else` Statement (continued)

Program 4.1

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
  double radius;

  cout << "Please type in the radius: ";
  cin  >> radius;

  if (radius < 0.0)
    cout << "A negative radius is invalid" << endl;
  else
    cout << "The area of this circle is " << 3.1416 * pow(radius,2) << endl;

  return 0;
}
```

Compound Statements

- **Compound statement:** A sequence of single statements contained between braces
  - Creates a block of statements
  - A block of statements can be used anywhere that a single statement is legal
  - Any variable declared within a block is usable only within that block
- **Scope:** The area within a program where a variable can be used
  - A variable's scope is based on where the variable is declared

Block Scope (continued)

```
{    // start of outer block
   int a = 25;
   int b = 17;

   cout << "The value of a is " << a
        <<" and b is " << b << endl;

   {    // start of inner block
     double a = 46.25;

     int c = 10;
     cout << "a is now " << a
          << " b is now " << b
          << " and c is " << c << endl;
   }   // end of inner block

   cout << "a is now " << a
        << " and b is " << b << endl;
}   // end of outer block
```

Block Scope (continued)

```
{    // start of outer block
   int a = 25;
   int b = 17;

   cout << "The value of a is " << a
        <<" and b is " << b << endl;

   {    // start of inner block
     double a = 46.25;

     int c = 10;
     cout << "a is now " << a
          << " b is now " << b
          << " and c is " << c << endl;
   }   // end of inner block

   cout << "a is now " << a
        << " and b is " << b << endl;
}   // end of outer block
```

One-Way Selection

- **One-way selection:** An **`if`** statement without the optional **`else`** portion

```
int a = 1;
if(a > 0)
{
    cout<<a;
}
```

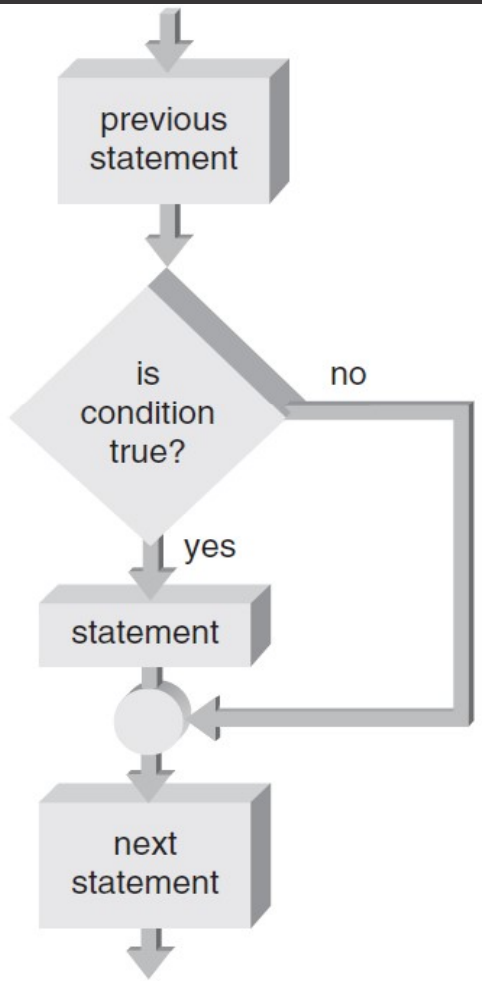


Figure 4.3 A one-way selection **`if`** statement

Problems Associated with the `if-else` Statement

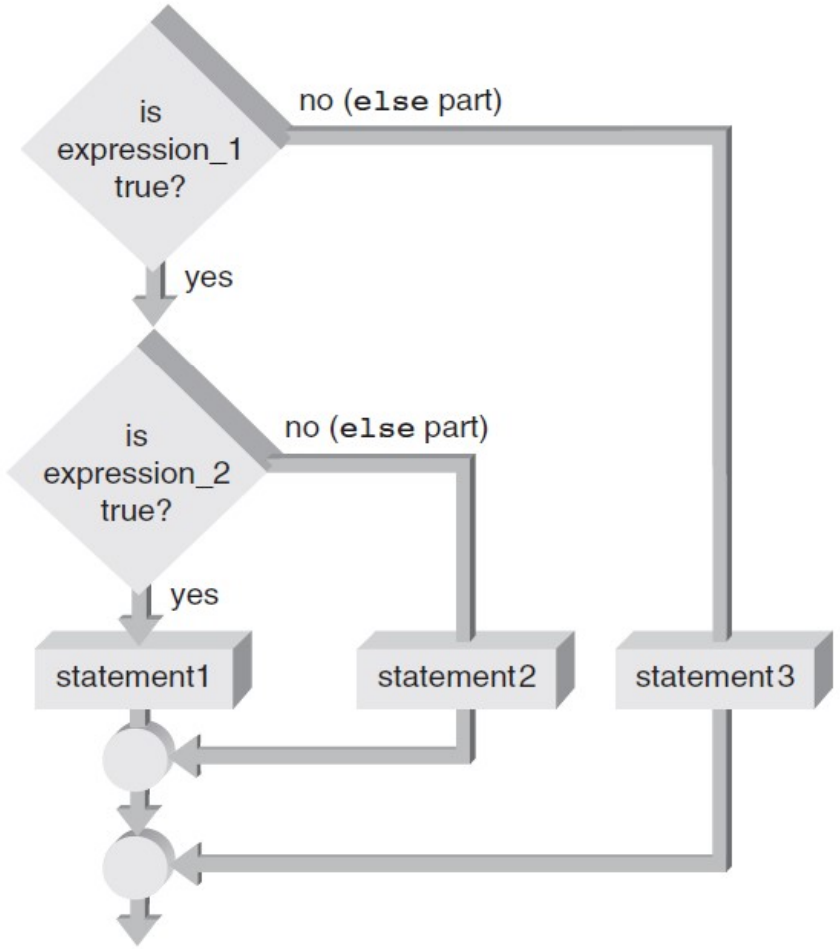
- Common problems with **`if-else`** statements:
  - Misunderstanding what an expression is
  - Using the assignment operator (**=**) instead of the relational operator (**==**)

Nested `if` Statements

- **`if-else`** statement can contain any valid C++ statement, including another **`if-else`**
- Nested **`if`** statement: an **`if-else`** statement completely contained within another **`if-else`**
- Use braces to block code, especially when inner **`if`** statement does not have its own **`else`**

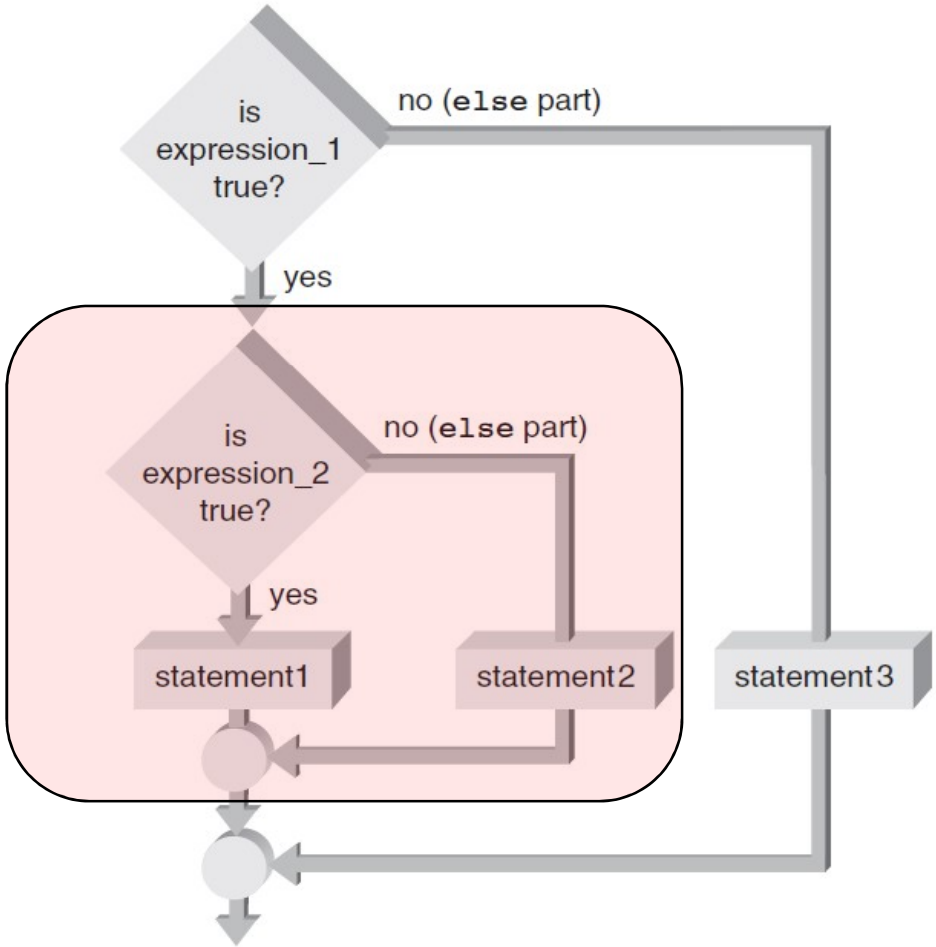
Nested `if` Statements (continued)

Figure 4.4a
Nested within the `if` part



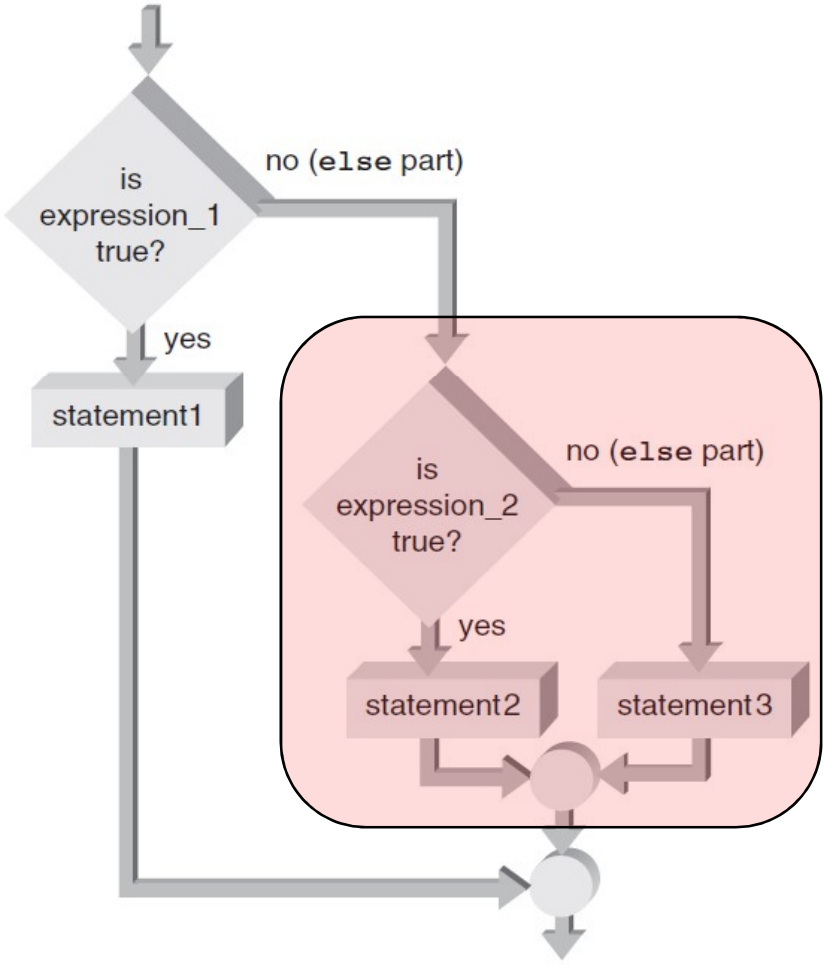
Nested `if` Statements (continued)

Figure 4.4a
Nested within the `if` part



Nested `if` Statements (continued)

Figure 4.4b
Nested within the **`else`** part



The `if-else` Chain

- If any condition is true, the corresponding statement is executed and the chain terminates
- Final **`else`** is only executed if no conditions were true
  - Serves as a catch-all case
- **`if-else`** chain provides one selection from many possible alternatives

The `if-else` Chain (continued)

- General form of an **`if-else`** chain

```
if (expression_1)
  statement1;
else if (expression_2)
  statement2;
else if (expression_3)
  statement3;
      .
      .
      .
else if (expression_n)
  statementn;
else
  last_statement;
```

Example 3

```
#include <iostream>
using namespace std;

int main()
{
    int number1, number2;
    cout<<"Enter two integers: \n";
    cin>>number1>>number2;

    if( number1 ==  number2)
          cout<<number1<<“  is equal to “<<number2;
    else if( number1 >  number2)
          cout<<number1<<“  is larger than“<<number2;
    else
          cout<<number1<<“  is smaller than“<<number2;

    return 0;

}
```