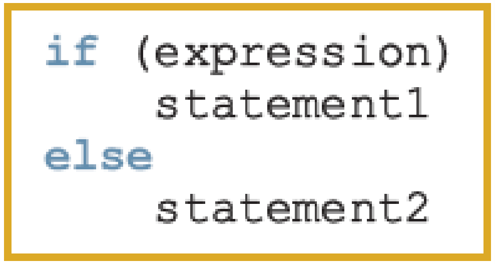
CS118 – Programming Fundamentals

Lecture # 11
Monday, September 30, 2019
FALL 2019
FAST – NUCES, Faisalabad Campus

Zain Iqbal

Two-Way Selection

- Two-way selection takes the form:



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

- If expression is true, statement1 is executed; otherwise, statement2 is executed
- statement1 and statement2 are any C++ statements
- **else** is a reserved word

Two-Way Selection (cont'd.)

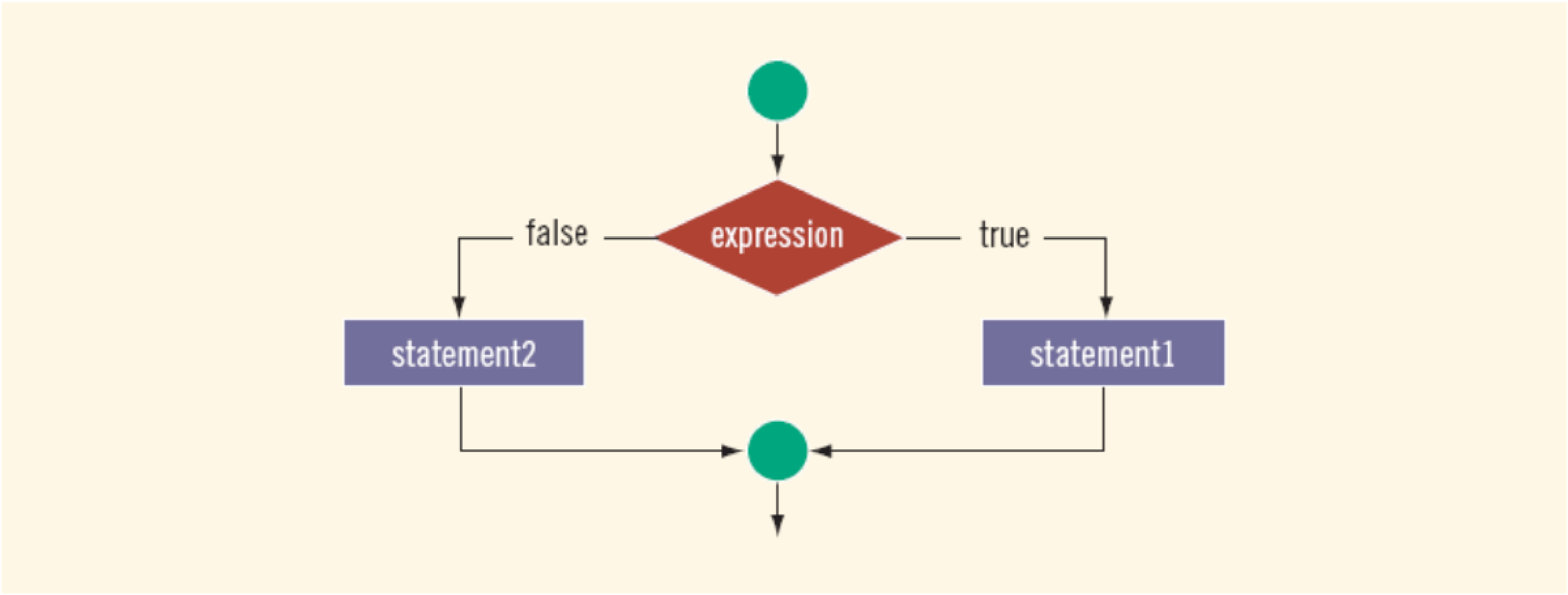
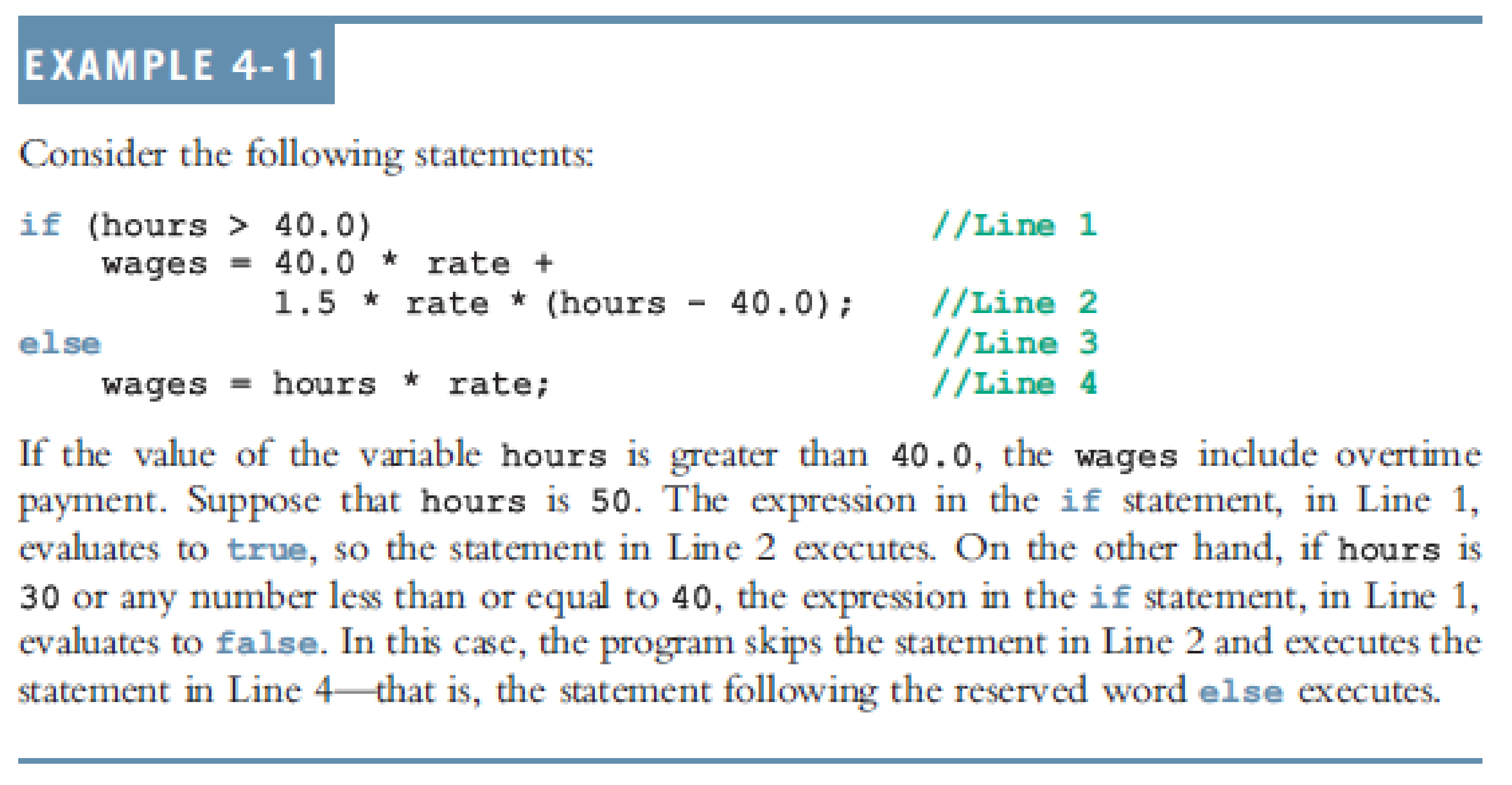


FIGURE 4-3 Two-way selection

Two-Way Selection (cont'd.)



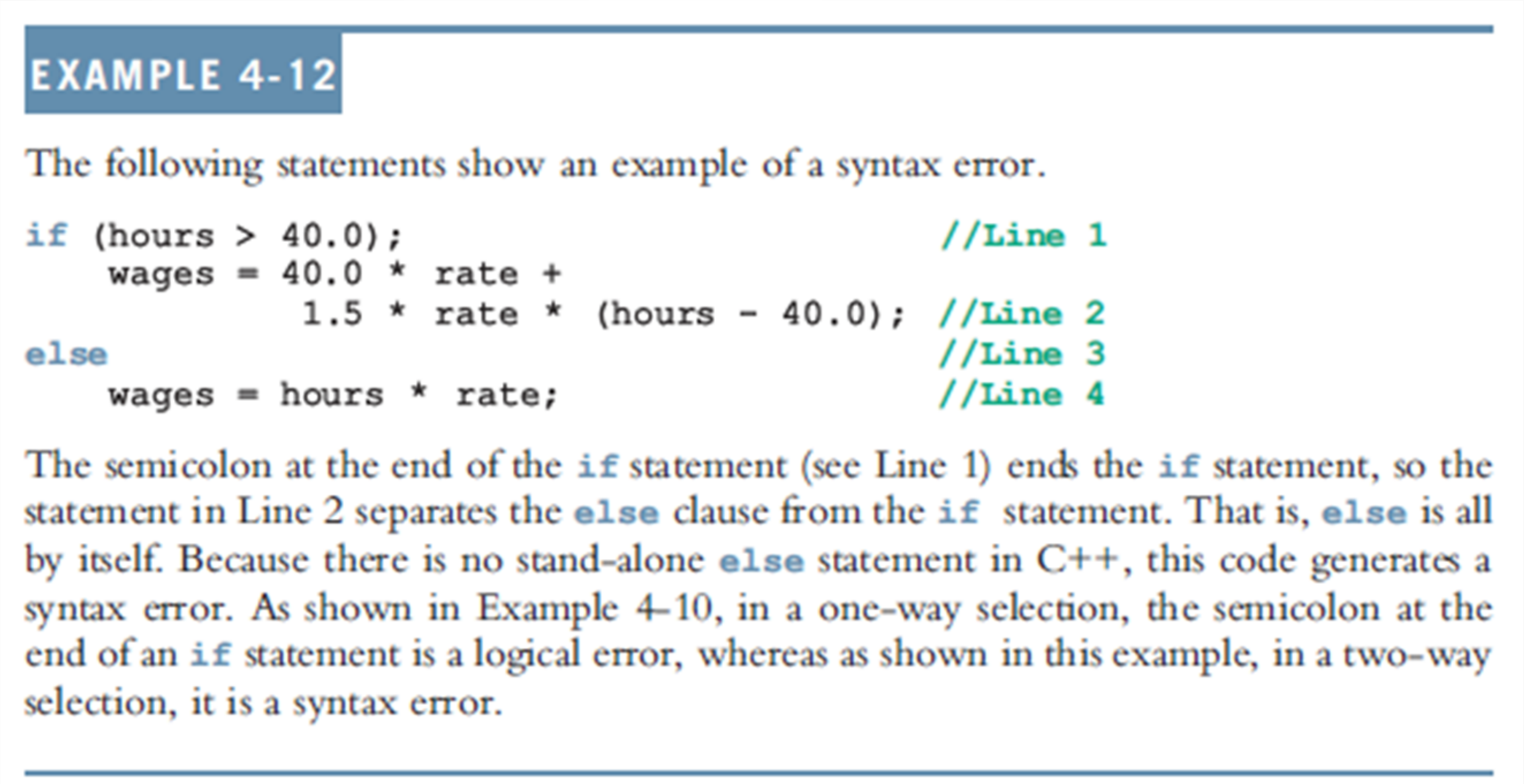
EXAMPLE 4-11

Consider the following statements:

```
if (hours > 40.0)                              //Line 1
    wages = 40.0 * rate +
            1.5 * rate * (hours - 40.0);       //Line 2
else                                           //Line 3
    wages = hours * rate;                      //Line 4
```

If the value of the variable hours is greater than 40.0, the wages include overtime payment. Suppose that hours is 50. The expression in the if statement, in Line 1, evaluates to true, so the statement in Line 2 executes. On the other hand, if hours is 30 or any number less than or equal to 40, the expression in the if statement, in Line 1, evaluates to false. In this case, the program skips the statement in Line 2 and executes the statement in Line 4—that is, the statement following the reserved word else executes.

Two-Way Selection (cont'd.)



EXAMPLE 4-12

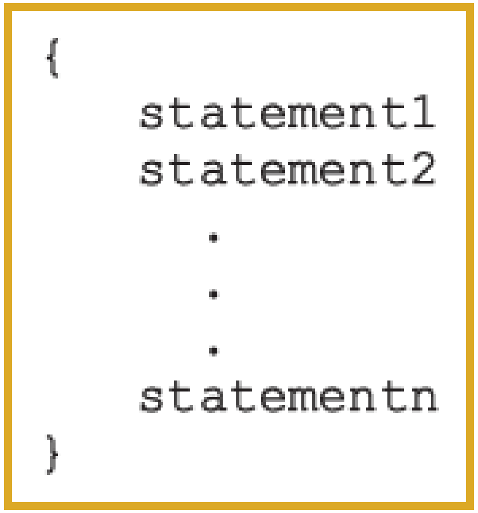
The following statements show an example of a syntax error.

```
if (hours > 40.0);                          //Line 1
    wages = 40.0 * rate +
             1.5 * rate * (hours - 40.0);   //Line 2
else                                        //Line 3
    wages = hours * rate;                   //Line 4
```

The semicolon at the end of the if statement (see Line 1) ends the if statement, so the statement in Line 2 separates the else clause from the if statement. That is, else is all by itself. Because there is no stand-alone else statement in C++, this code generates a syntax error. As shown in Example 4-10, in a one-way selection, the semicolon at the end of an if statement is a logical error, whereas as shown in this example, in a two-way selection, it is a syntax error.

Compound (Block of) Statements

- Compound statement (block of statements):



```
{
      statement1
      statement2
         .
         .
         .
      statementn
}
```

- A compound statement is a single statement

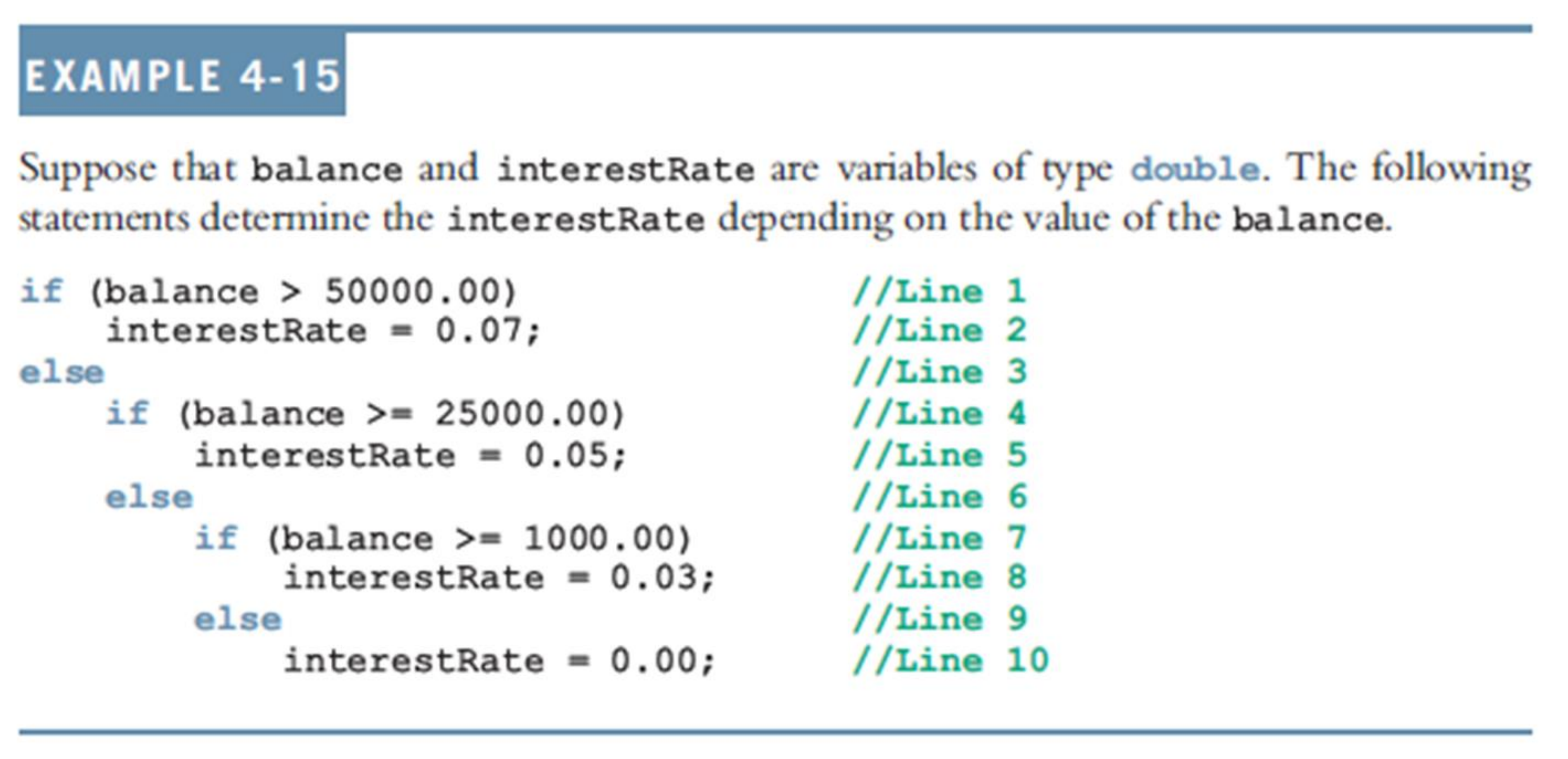
Compound (Block of) Statements (cont'd.)

```
if (age > 18)
{
    cout << "Eligible to vote." << endl;
    cout << "No longer a minor." << endl;
}
else
{
    cout << "Not eligible to vote." << endl;
    cout << "Still a minor." << endl;
}
```

Multiple Selections: Nested if

- **Nesting:** One control statement in another
- An **else** is associated with the most recent **if** that has not been paired with an **else**

Multiple Selections: Nested if (cont'd.)

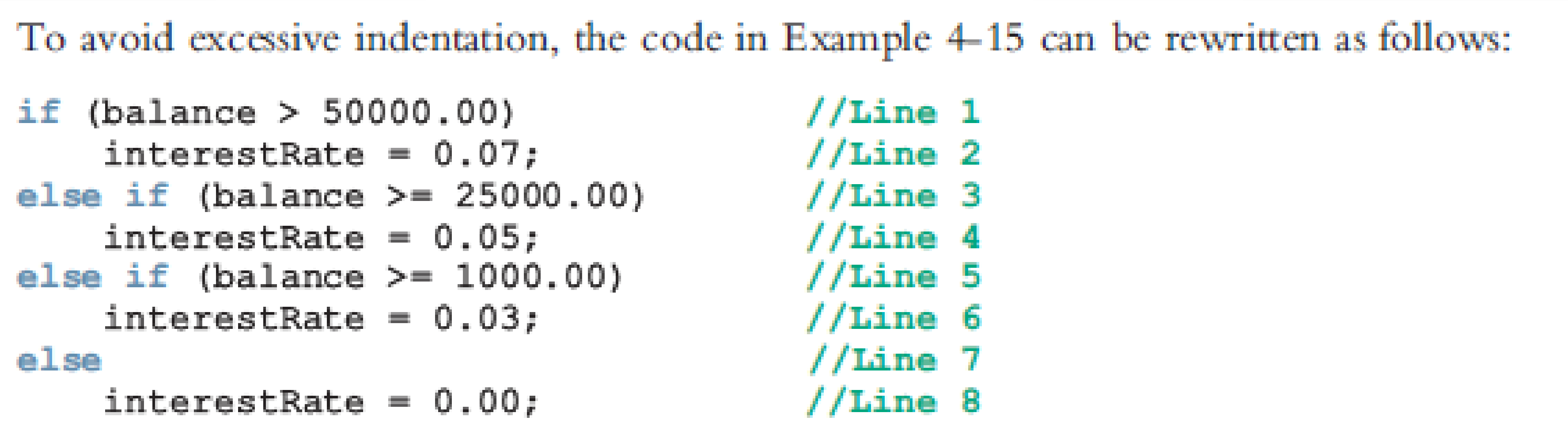


EXAMPLE 4-15

Suppose that balance and interestRate are variables of type double. The following statements determine the interestRate depending on the value of the balance.

```
if (balance > 50000.00)                    //Line 1
    interestRate = 0.07;                   //Line 2
else                                       //Line 3
    if (balance >= 25000.00)               //Line 4
        interestRate = 0.05;               //Line 5
    else                                   //Line 6
        if (balance >= 1000.00)            //Line 7
            interestRate = 0.03;           //Line 8
        else                               //Line 9
            interestRate = 0.00;           //Line 10
```

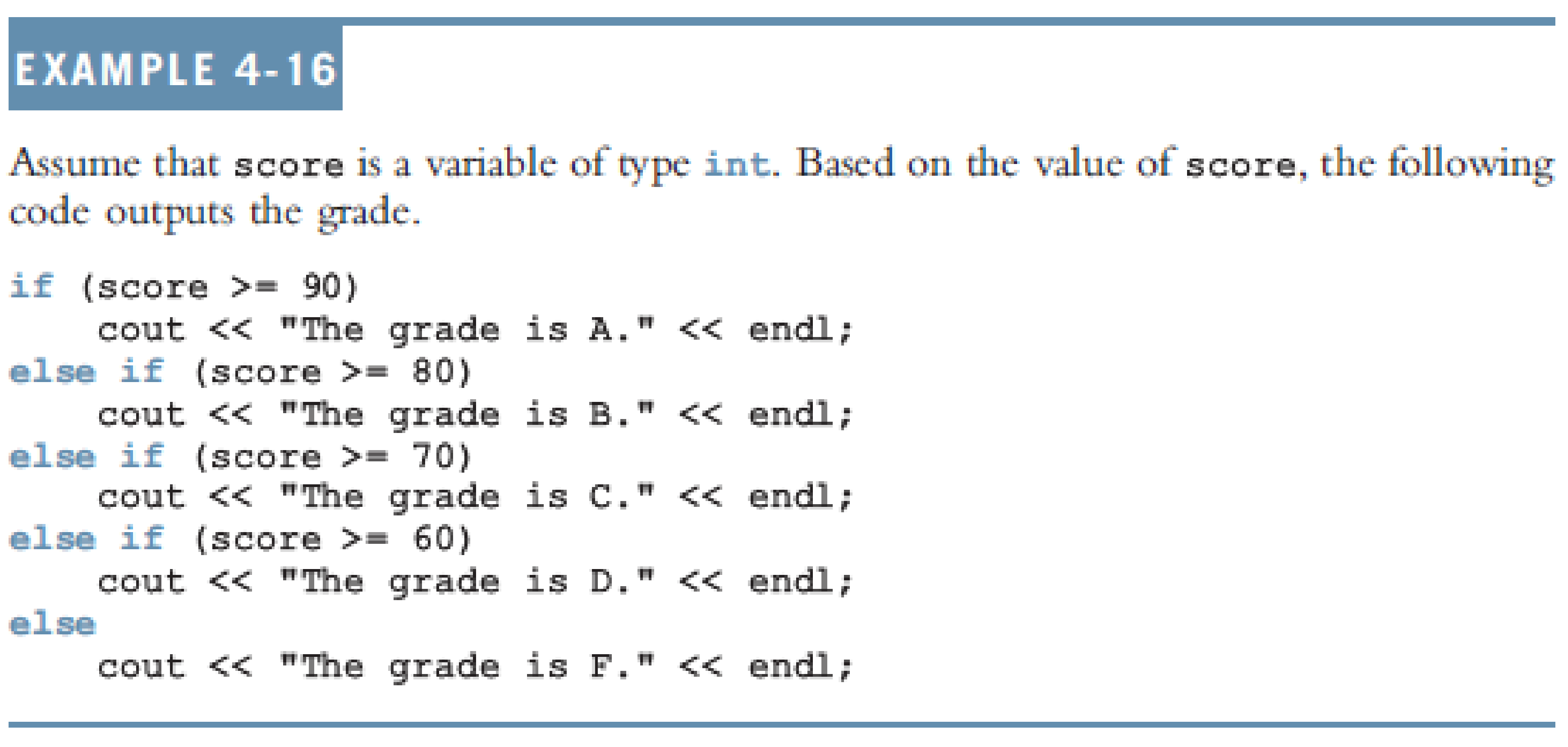
Multiple Selections: Nested if (cont'd.)



To avoid excessive indentation, the code in Example 4-15 can be rewritten as follows:

```
if (balance > 50000.00)                  //Line 1
    interestRate = 0.07;                 //Line 2
else if (balance >= 25000.00)            //Line 3
    interestRate = 0.05;                 //Line 4
else if (balance >= 1000.00)             //Line 5
    interestRate = 0.03;                 //Line 6
else                                     //Line 7
    interestRate = 0.00;                 //Line 8
```

Multiple Selections: Nested if (cont'd.)



EXAMPLE 4-16

Assume that score is a variable of type int. Based on the value of score, the following code outputs the grade.

```
if (score >= 90)
    cout << "The grade is A." << endl;
else if (score >= 80)
    cout << "The grade is B." << endl;
else if (score >= 70)
    cout << "The grade is C." << endl;
else if (score >= 60)
    cout << "The grade is D." << endl;
else
    cout << "The grade is F." << endl;
```

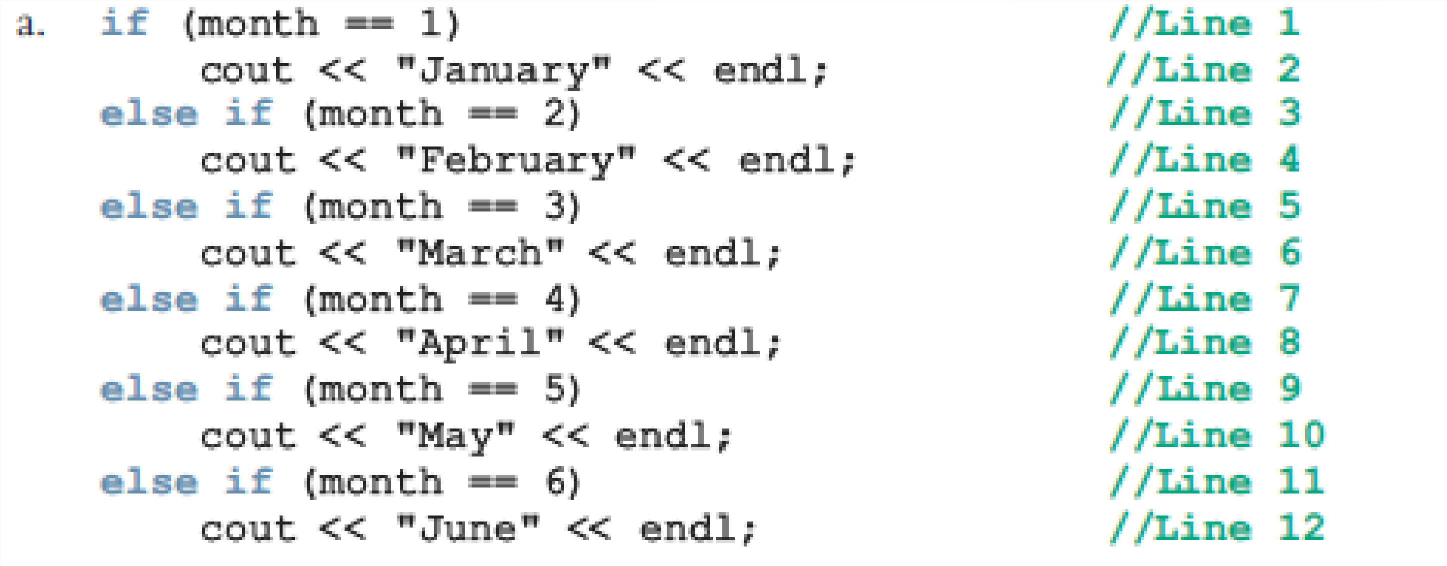
if-else **Pairing**

Assume that all the variables are properly declared and consider the following statements:

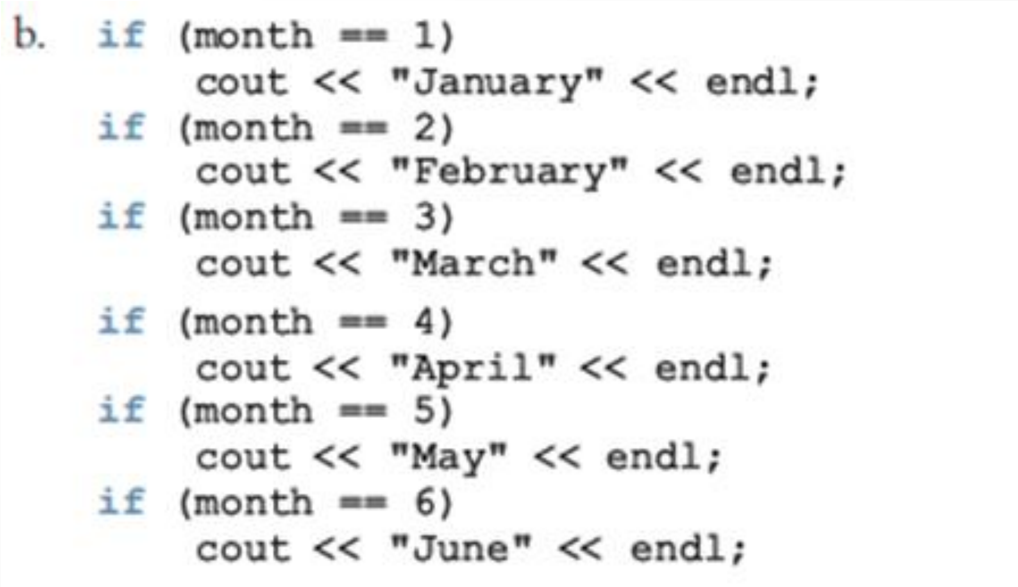
```
if(gender == 'M')                 //Line 1
     if(age < 21)                 //Line 2
          policyRate = 0.05;      //Line 3
     else             //Line 4
          policyRate = 0.035;     //Line 5
else if (gender = 'F')       //Line 6
     if(age < 21)                 //Line 7
          policyRate = 0.04;      //Line 8
     else             //Line 9
          policyRate = 0.03;      //Line 10
```

In this code, the else in Line 4 is paired with the if in Line 2. Note that for the else in Line 4, the most recent incomplete if is the if in Line 2. The else in Line 6 is paired with the if in Line 1. The else in Line 9 is paired with the if in Line 7. Once again the indentation does not determine the pairing, but it communicates the pairing

Comparing if…else Statements with a Series of if Statements



```
a.  if (month == 1)                             //Line 1
        cout << "January" << endl;              //Line 2
    else if (month == 2)                        //Line 3
        cout << "February" << endl;             //Line 4
    else if (month == 3)                        //Line 5
        cout << "March" << endl;                //Line 6
    else if (month == 4)                        //Line 7
        cout << "April" << endl;                //Line 8
    else if (month == 5)                        //Line 9
        cout << "May" << endl;                  //Line 10
    else if (month == 6)                        //Line 11
        cout << "June" << endl;                 //Line 12
```



```
b.  if (month == 1)
        cout << "January" << endl;
    if (month == 2)
        cout << "February" << endl;
    if (month == 3)
        cout << "March" << endl;
    if (month == 4)
        cout << "April" << endl;
    if (month == 5)
        cout << "May" << endl;
    if (month == 6)
        cout << "June" << endl;
```

Short-Circuit Evaluation

- **Short-circuit evaluation:** evaluation of a logical expression stops as soon as the value of the expression is known
- **Example:**

Assume x = 21, y=5, z = 3, ch = ‘B’

```
(x >= 20)  | |  ( y == 10) //Line 1
(ch == 'A') && (z < 7)   //Line 2
```

Comparing Floating-Point Numbers for Equality: A Precaution

- Comparison of floating-point numbers for equality may not behave as you would expect
- **Example:**
  - 1.0 == 3.0/7.0 + 2.0/7.0 + 2.0/7.0 evaluates to false
  - Why? 3.0/7.0 + 2.0/7.0 + 2.0/7.0 = 0.99999999999999989
- **Solution:** use a tolerance value
  - Example: fabs(x – y) < 0.000001

```
#include<iostream>
#include <iomanip>
#include<cmath>

using namespace std;

int main()
{
        double x =1.0;
        double y = 3.0 / 7.0 + 2.0 / 7.0 + 2.0 / 7.0 ;
        cout << fixed << showpoint << setprecision(17);
        cout << "3.0 / 7.0 + 2.0 / 7.0 + 2.0 / 7.0 = "
                << 3.0 / 7.0 + 2.0 / 7.0 + 2.0 / 7.0 << endl;
        cout << "x = " << x << endl << "y = " << y << endl;
        if(x == y)
                cout << "x and y are same" <<endl;
        else
                cout << "x and y are not same" << endl;
        if (fabs(x-y)<0.000001)
                cout << "x and y are same within the tolerance 0.000001" << endl;
        else
                cout <<  "x and y are not same within the tolerance 0.000001" << endl;
        return 0;
}
```

```
Sample Run:
3.0 / 7.0 + 2.0 / 7.0 + 2.0 / 7.0 = 0.99999999999999989
x = 1.00000000000000000
y = 0.99999999999999989
x and y are not the same.
x and y are the same within the tolerance 0.000001.
```

Associativity of Relational Operators: A Precaution

```
#include<iostream>

using namespace std;

int main()
{
        int x;

        cout << "Enter and integer =  " ;
        cin >> x ;
        cout << endl ;

        if (0 <= x <= 10)
                cout << x << " is within 0 and 10" << endl;
        else
                cout << x << " is not within 0 and 10" << endl;

        return 0;
}
```

Associativity of Relational Operators: A Precaution (cont'd.)

- x = 7

0 <= x <= 10	= 0 <= 7 <= 10	
	= (0 <=7) <= 10	Because relationship operators are evaluated from left to right
	= 1 <= 10	Because 0<=7 is true, 0<=7 evaluates to 1
	= 1 (true)	

- x = 30

0 <= x <= 10	= 0 <= 30 <= 10	
	= (0 <=30) <= 10	Because relationship operators are evaluated from left to right
	= 1 <= 10	Because 0<=30 is true, 0<=30 evaluates to 1
	= 1 (true)	

Solution:
0 <= x && x <= 10

Avoiding Bugs by Avoiding Partially Understood Concepts and Techniques

- Must use concepts and techniques correctly;
  - Otherwise solution will be either **incorrect** or **deficient**
- If you do not understand a concept or technique completely
  - Don’t use it
  - Save yourself an enormous amount of debugging time

Example

```
if (gpa >= 2.0)
if (gpa >= 3.9)
cout << "Dean\'s Honor List."
<< endl;
else
cout << "The GPA is below the graduation“
<<"requirement. \nSee your “
<< "academic advisor." << endl;
```

Input Failure and the if Statement

- If input stream enters a fail state
  - All subsequent input statements associated with that stream are **ignored**
  - Program continues to execute
  - May produce **erroneous results**
- Can use **if** statements to check status of input stream
- If stream enters the fail state, include instructions that **stop program execution**

```
if(cin)
```

Confusion Between the Equality (==) and Assignment (=) Operators

- C++ allows you to use any expression that can be evaluated to either true or false as an expression in the if statement:

```
 if (x = 5)
        cout << "The value is five." << endl;
```

- The appearance of = in place of == resembles a silent killer
  - It is not a syntax error
  - It is a logical error

Conditional Operator (?:)

- Conditional operator (?:) takes three arguments
  - Ternary operator
- Syntax for using the conditional operator:

  expression1 ? expression2 : expression3
- If **expression1** is true, the result of the conditional expression is **expression2**
- Otherwise, the result is **expression3**

Conditional Operator (?:)

Consider the following statement

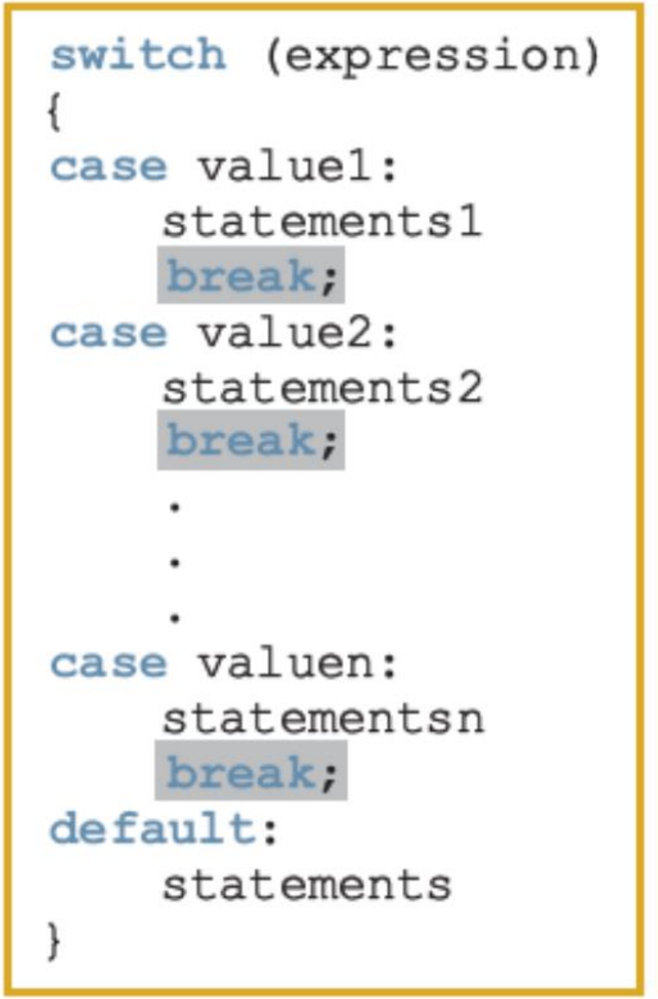
```
if(x >= y)
    large = x;
else
    large = y;
```

You can use the conditional operator to simplify the writing of this **if…else** statement as follows:

```
large = (x >= y)? x: y ;
```

switch **Structures**

- **switch structure:** Alternate to **if-else**
- switch **(integral)** expression is evaluated first
- Value of the expression determines which corresponding action is taken
- Expression is sometimes called the selector



```
switch (expression)
{
case value1:
     statements1
     break;
case value2:
     statements2
     break;
       .
       .
       .
case valuen:
     statementsn
     break;
default:
     statements
}
```

switch **Structures (cont'd.)**

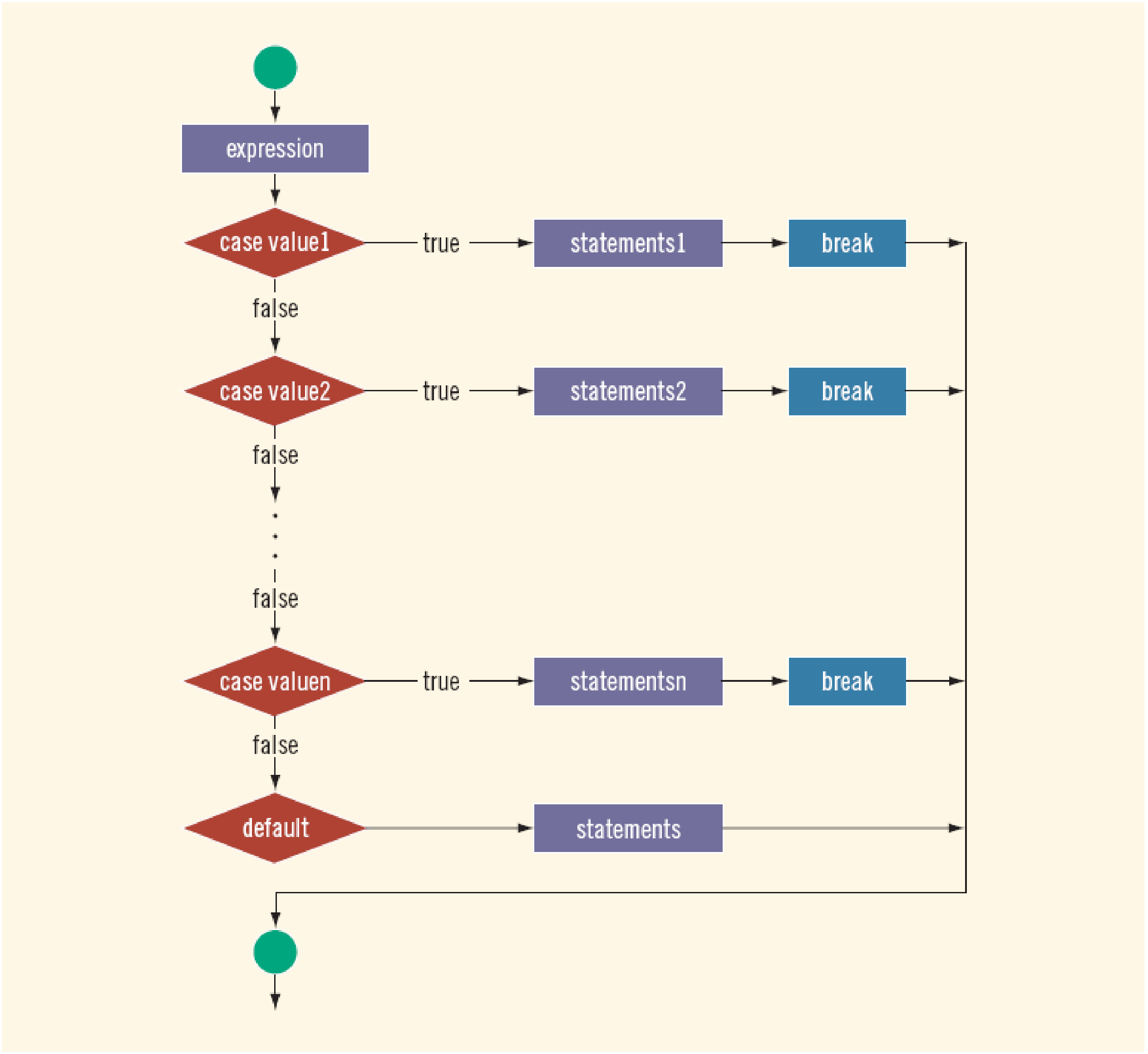
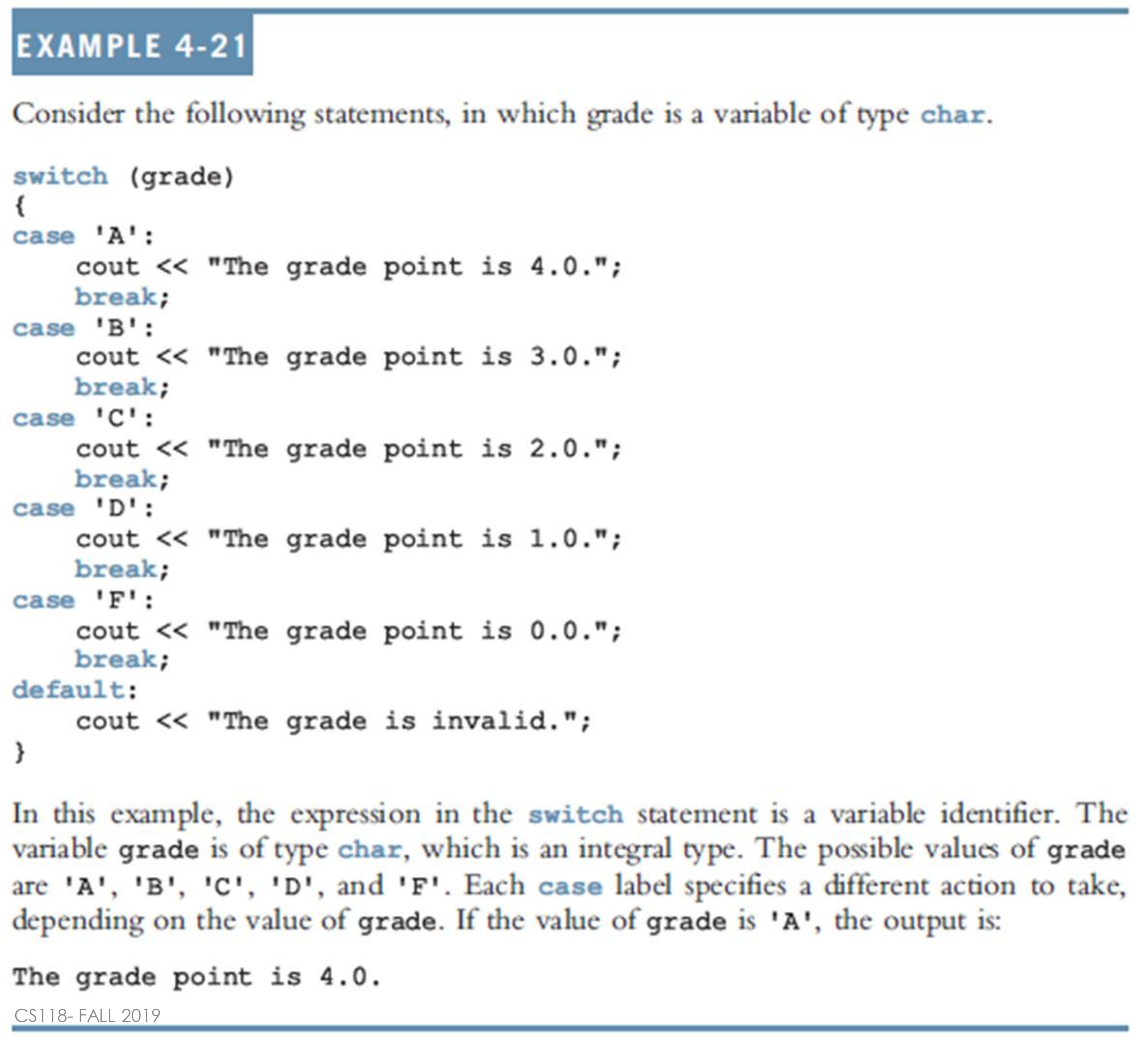


FIGURE 4-4 `switch` statement

switch **Structures (cont'd.)**

- One or more statements may follow a case label
- **Braces are not needed to turn multiple** statements into a single compound statement
- The break statement may or may not appear after each statement
- **switch**, **case**, **break**, and **default** are reserved words



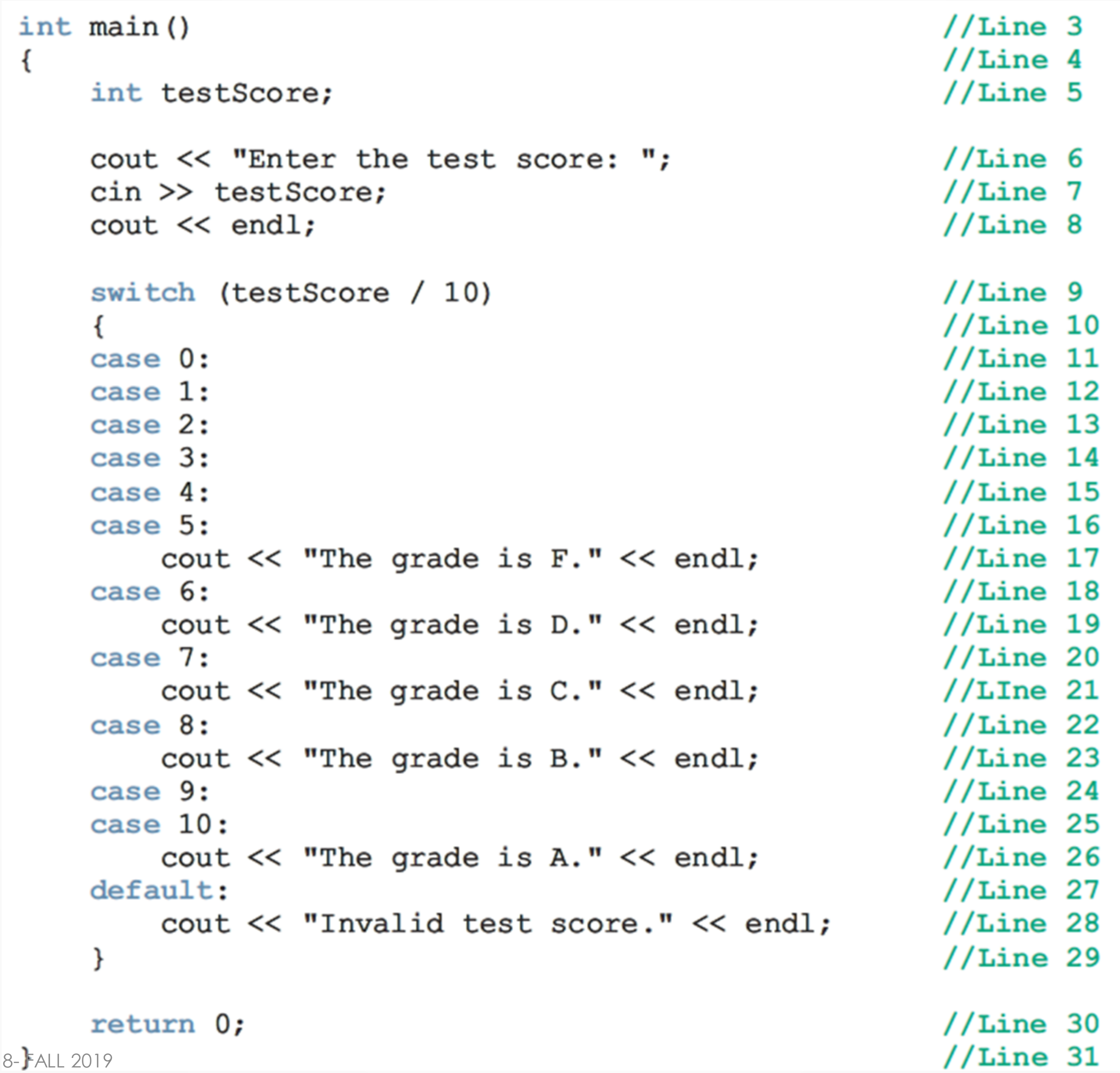
EXAMPLE 4-21

Consider the following statements, in which grade is a variable of type char.

```
switch (grade)
{
case 'A':
    cout << "The grade point is 4.0.";
    break;
case 'B':
    cout << "The grade point is 3.0.";
    break;
case 'C':
    cout << "The grade point is 2.0.";
    break;
case 'D':
    cout << "The grade point is 1.0.";
    break;
case 'F':
    cout << "The grade point is 0.0.";
    break;
default:
    cout << "The grade is invalid.";
}
```

In this example, the expression in the switch statement is a variable identifier. The variable grade is of type char, which is an integral type. The possible values of grade are 'A', 'B', 'C', 'D', and 'F'. Each case label specifies a different action to take, depending on the value of grade. If the value of grade is 'A', the output is:

```
The grade point is 4.0.
```



```
int main()                                              //Line 3
{                                                       //Line 4
    int testScore;                                      //Line 5

    cout << "Enter the test score: ";                   //Line 6
    cin >> testScore;                                   //Line 7
    cout << endl;                                       //Line 8

    switch (testScore / 10)                             //Line 9
    {                                                   //Line 10
    case 0:                                             //Line 11
    case 1:                                             //Line 12
    case 2:                                             //Line 13
    case 3:                                             //Line 14
    case 4:                                             //Line 15
    case 5:                                             //Line 16
        cout << "The grade is F." << endl;              //Line 17
    case 6:                                             //Line 18
        cout << "The grade is D." << endl;              //Line 19
    case 7:                                             //Line 20
        cout << "The grade is C." << endl;              //LIne 21
    case 8:                                             //Line 22
        cout << "The grade is B." << endl;              //Line 23
    case 9:                                             //Line 24
    case 10:                                            //Line 25
        cout << "The grade is A." << endl;              //Line 26
    default:                                            //Line 27
        cout << "Invalid test score." << endl;          //Line 28
    }                                                   //Line 29

    return 0;                                           //Line 30
}                                                       //Line 31
```

```
#include<iostream>
using namespace std;
int main(){
    int number;
    cout << "Enter a number in the range 0 - 7 : " ;
    cin >> number;
    cout << "The number you entered is = " << number << endl;
    switch(number){
        case 0:
        case 1:
            cout << "Learning to use " ;
        case 2:
            cout << "C++'s " ;
        case 3:
            cout <<"switch structure." << endl;
            break;
        case 4:
            break;
        case 5:
            cout << "This program shows the effect " ;
        case 6:
        case 7:
            cout << "of break statement." << endl;
            break;
        default:
            cout <<"The number is out of range." << endl;
    }
    cout << "Out of the switch structure" << endl;

    return 0;
}
```

0
2
4
5
7
8

Questions