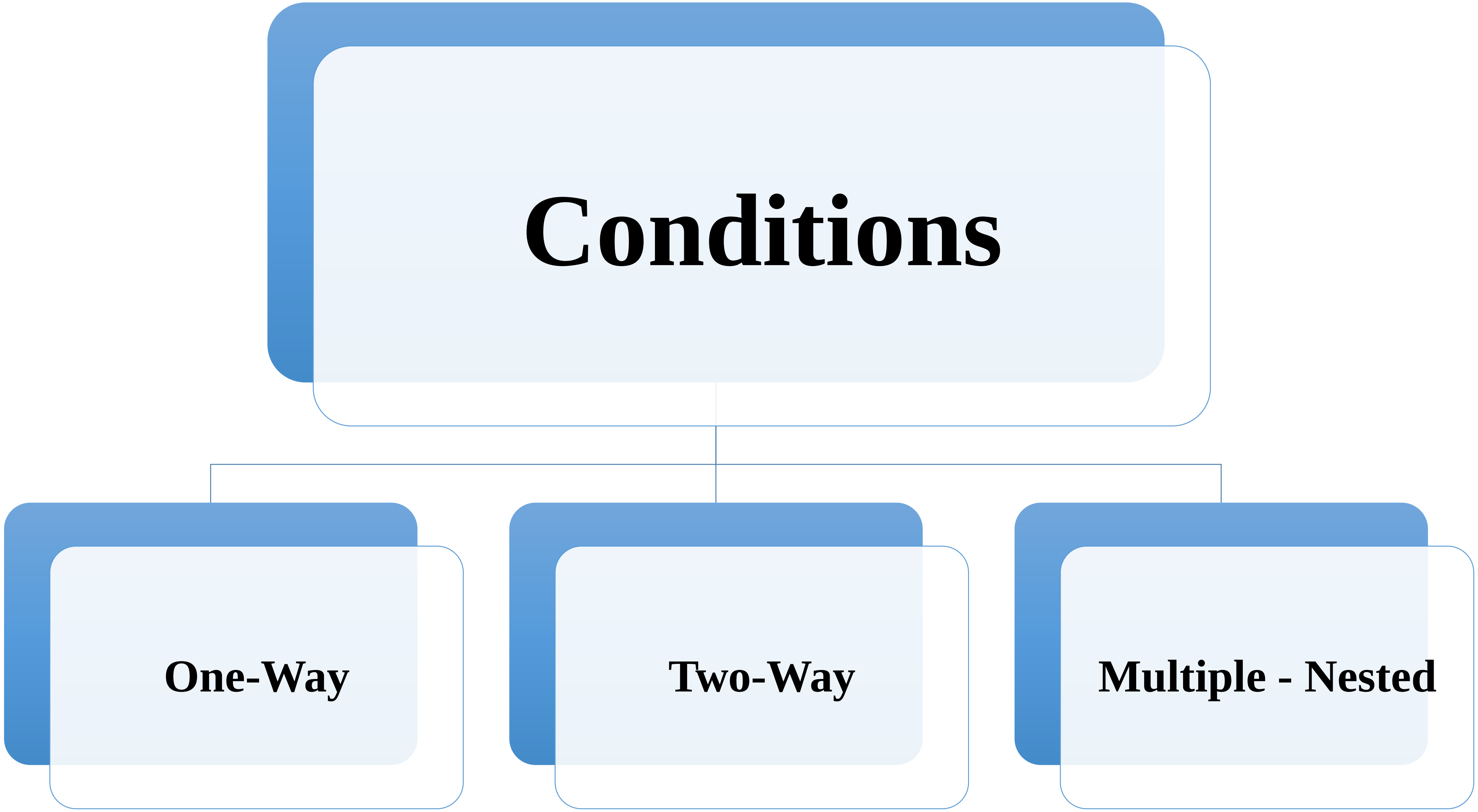
E1123 Computer Programming (a)

(Fall 2020)

Conditions

Instructor

Dr / Ayman Soliman



Conditions

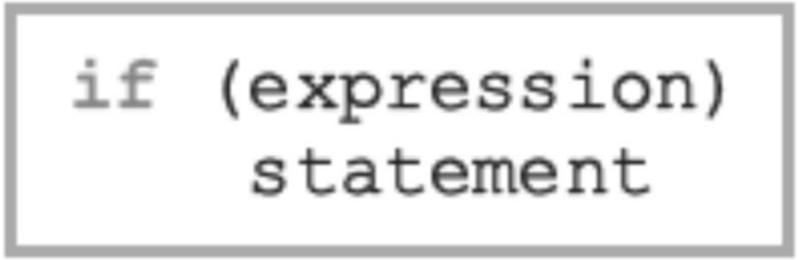
One-Way

Two-Way

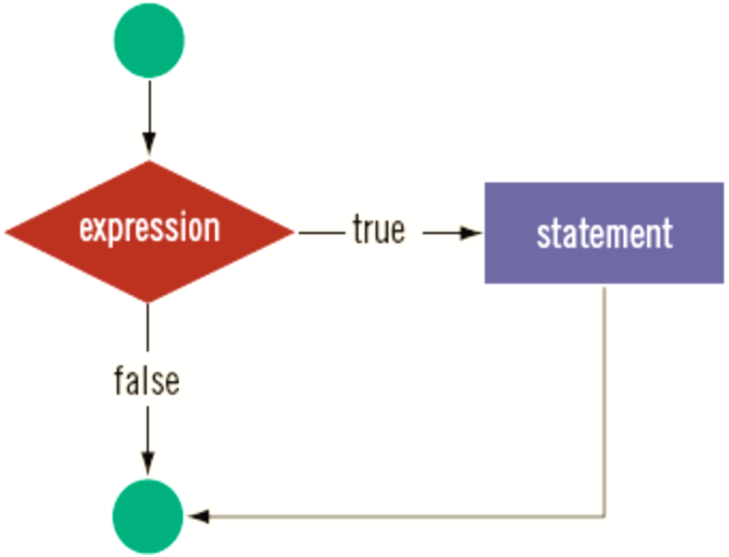
Multiple - Nested

➢ One-Way Selection

The syntax of one-way selection is:



```
if (expression)
    statement
```

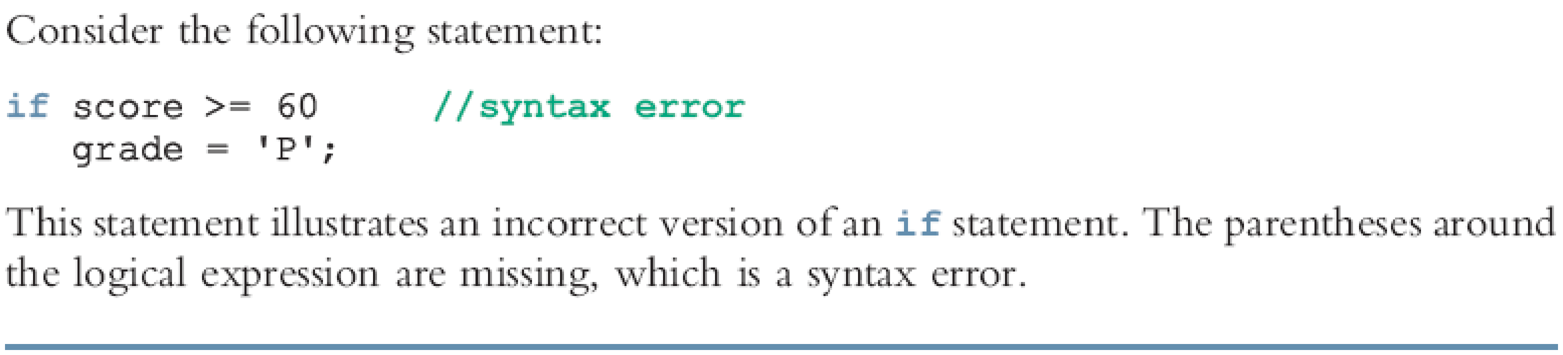


❑ The statement is executed if the value of the expression is true

❑ The statement is bypassed if the value is false; program goes to the next statement

❑ if is a reserved word

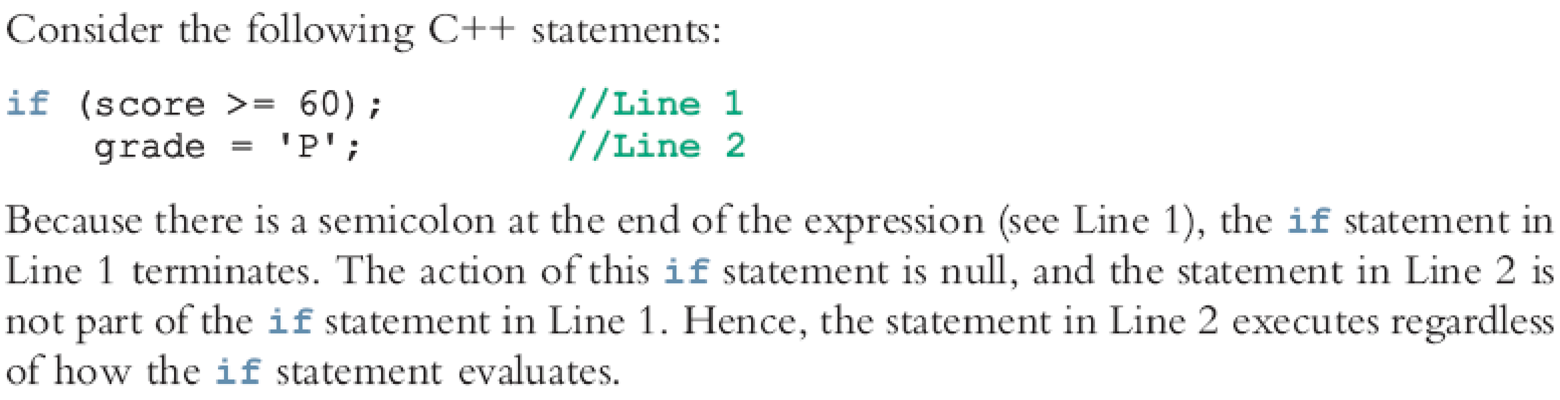
➢ One-Way Selection (syntax error)



Consider the following statement:

```
if score >= 60      //syntax error
   grade = 'P';
```

This statement illustrates an incorrect version of an if statement. The parentheses around the logical expression are missing, which is a syntax error.

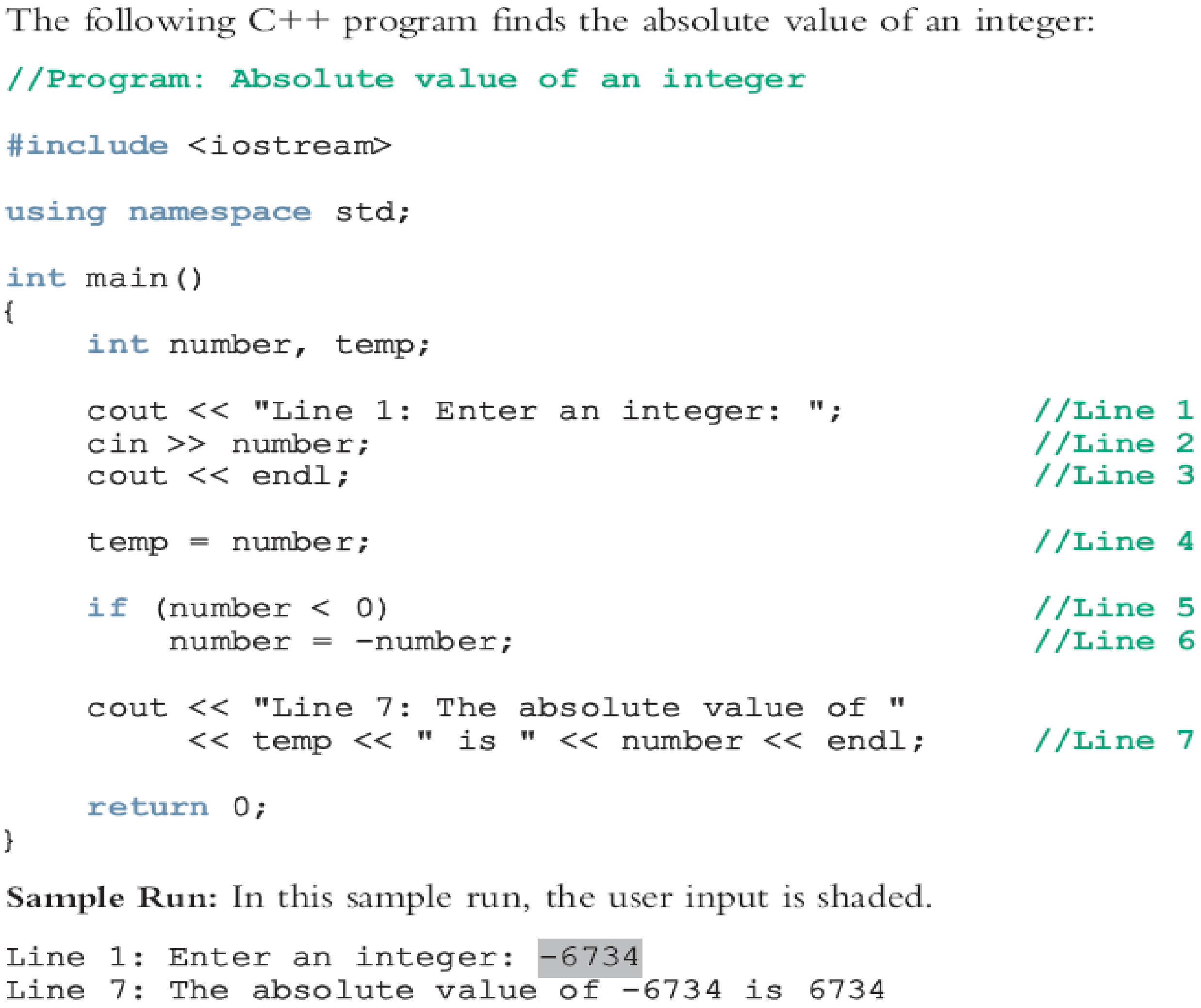


Consider the following C++ statements:

```
if (score >= 60);           //Line 1
    grade = 'P';            //Line 2
```

Because there is a semicolon at the end of the expression (see Line 1), the if statement in Line 1 terminates. The action of this if statement is null, and the statement in Line 2 is not part of the if statement in Line 1. Hence, the statement in Line 2 executes regardless of how the if statement evaluates.

Example



The following C++ program finds the absolute value of an integer:

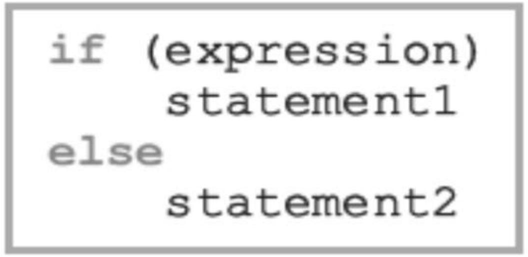
```
//Program: Absolute value of an integer

#include <iostream>

using namespace std;

int main()
{
    int number, temp;

    cout << "Line 1: Enter an integer: ";         //Line 1
    cin >> number;                                //Line 2
    cout << endl;                                 //Line 3

    temp = number;                                //Line 4

    if (number < 0)                               //Line 5
        number = -number;                         //Line 6

    cout << "Line 7: The absolute value of "
         << temp << " is " << number << endl;     //Line 7

    return 0;
}
```

Sample Run: In this sample run, the user input is shaded.

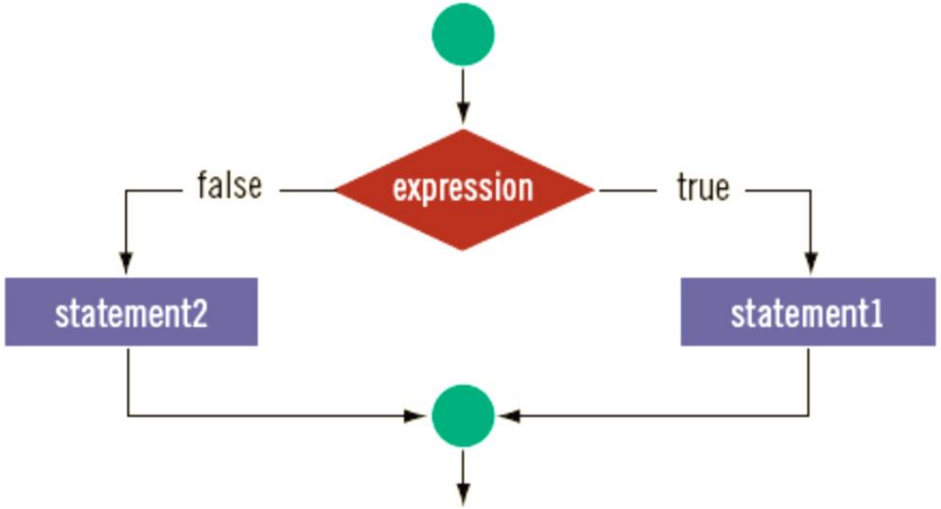
```
Line 1: Enter an integer: -6734
Line 7: The absolute value of -6734 is 6734
```

➢ Two-Way Selection

Two-way selection takes the form:

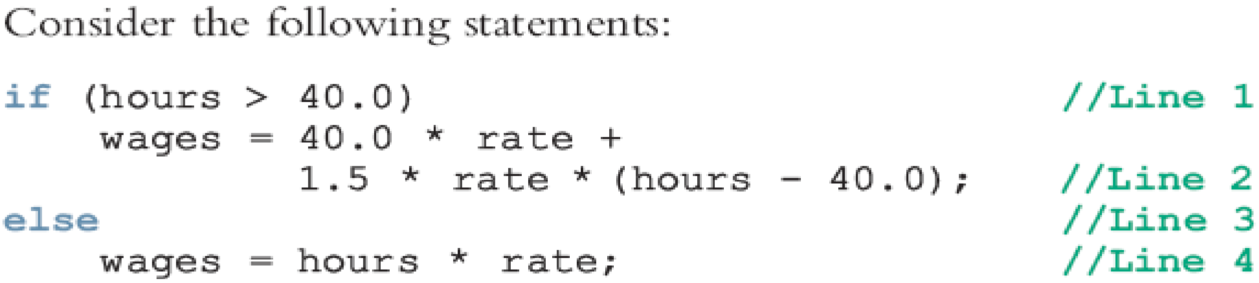


```
if (expression)
     statement1
else
     statement2
```



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

Example→

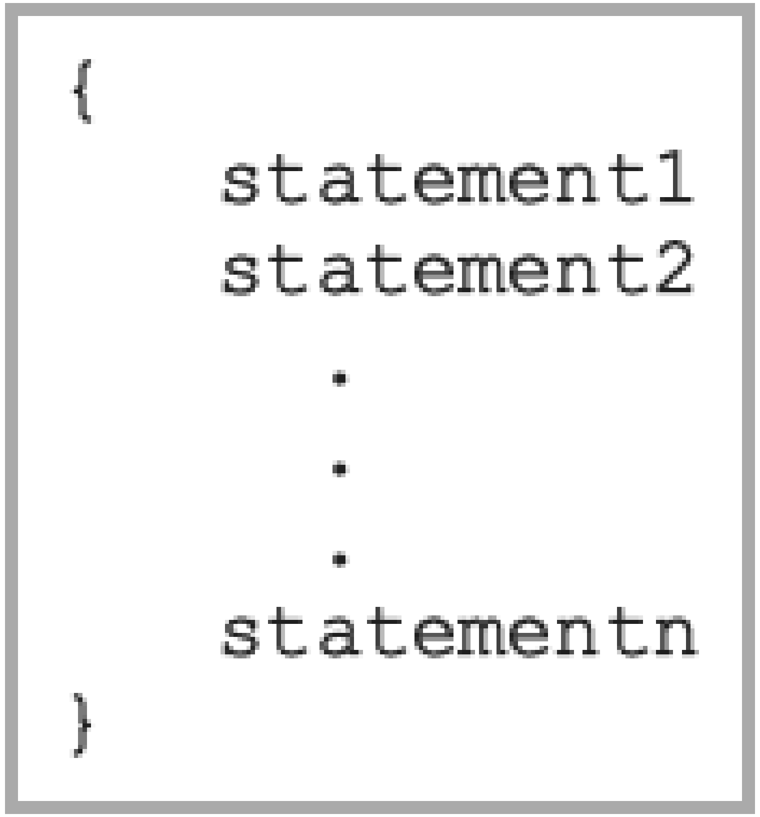


Consider the following statements:

```
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
```

➢ Compound (Block of) Statement

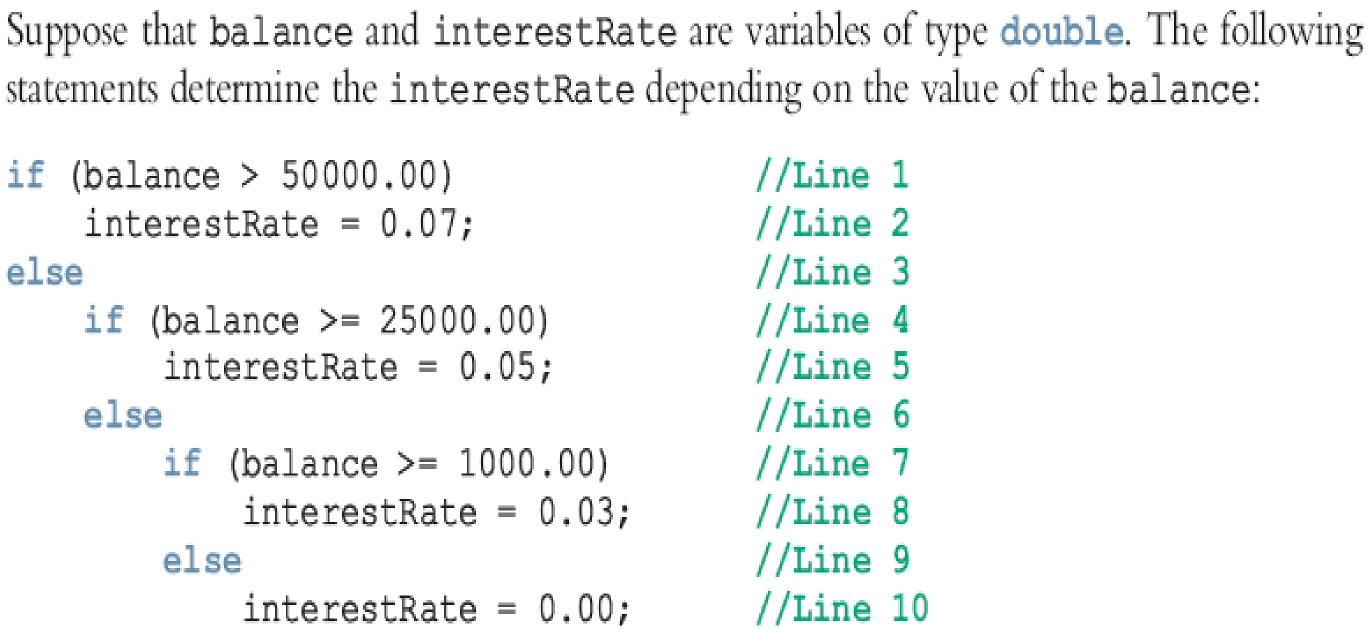
Compound statement (block of statements):



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

➢ Multiple Selections: Nested if

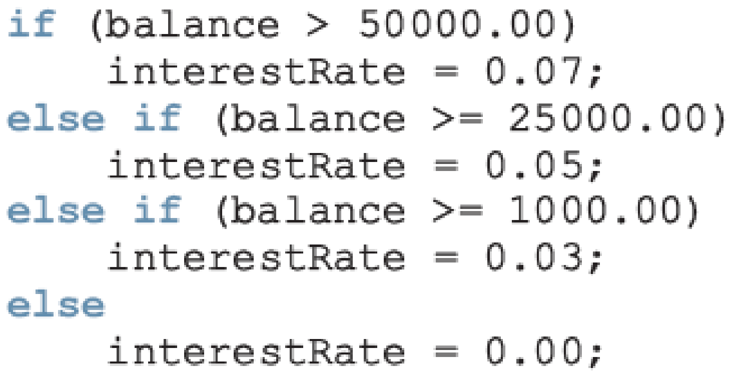
- ❑ Nesting: one control statement in another
- ❑ An else is associated with the most recent if that has not been paired with an else



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

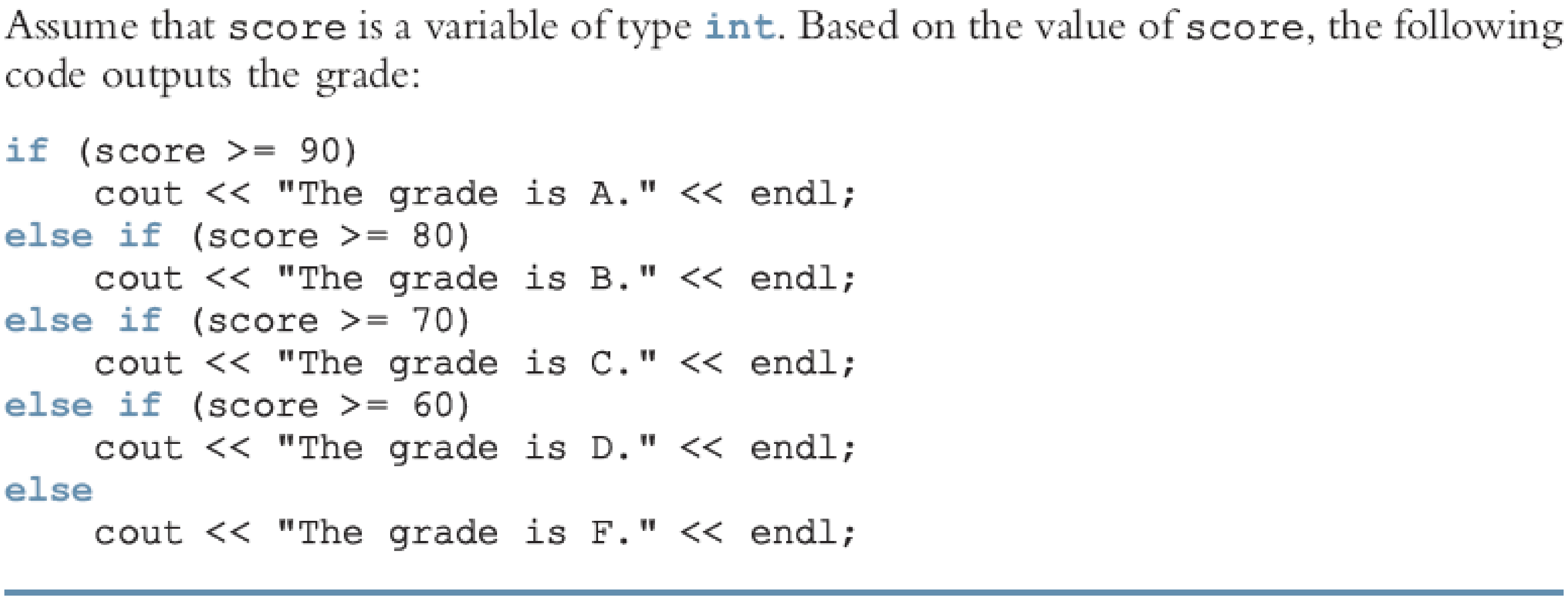
```
if (balance > 50000.00)                //Line 1
    interestRate = 0.07;               //Line 2
else                                   //Line 3
    if (balance >= 25000.00)           //Line 4
        interestRate = 0.05;           //Line 5
    else                               //Line 6
        if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

most efficient method



```
if  (balance > 50000.00)
     interestRate = 0.07;
else if  (balance >= 25000.00)
     interestRate = 0.05;
else if  (balance >= 1000.00)
     interestRate = 0.03;
else
     interestRate = 0.00;
```

➢ Example



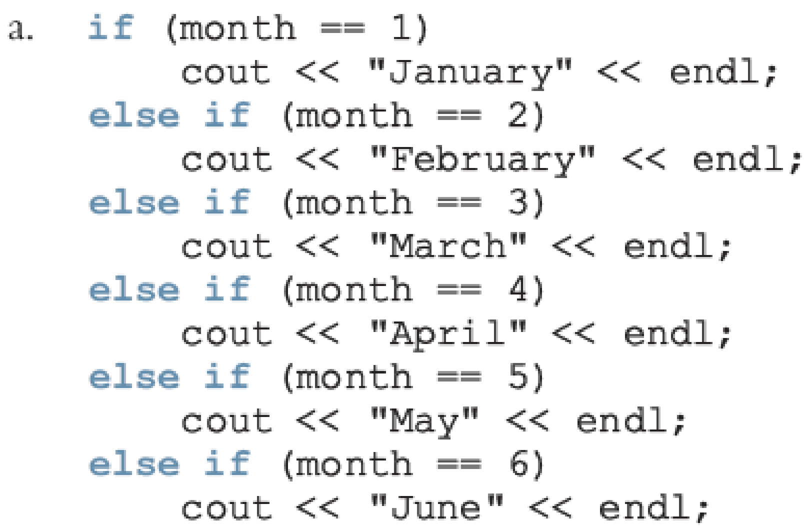
Assume that score is a variable of type int. Based on the value of score, the following code outputs the grade:

```
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;
```

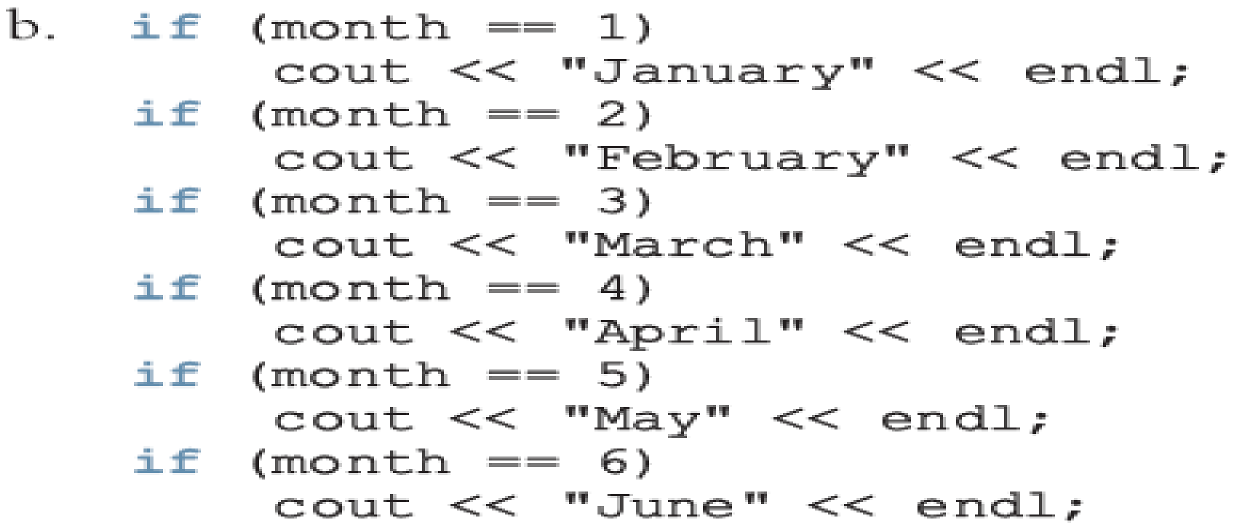
➢ Comparing if…else Statements with a Series of if Statements

First method

Second method



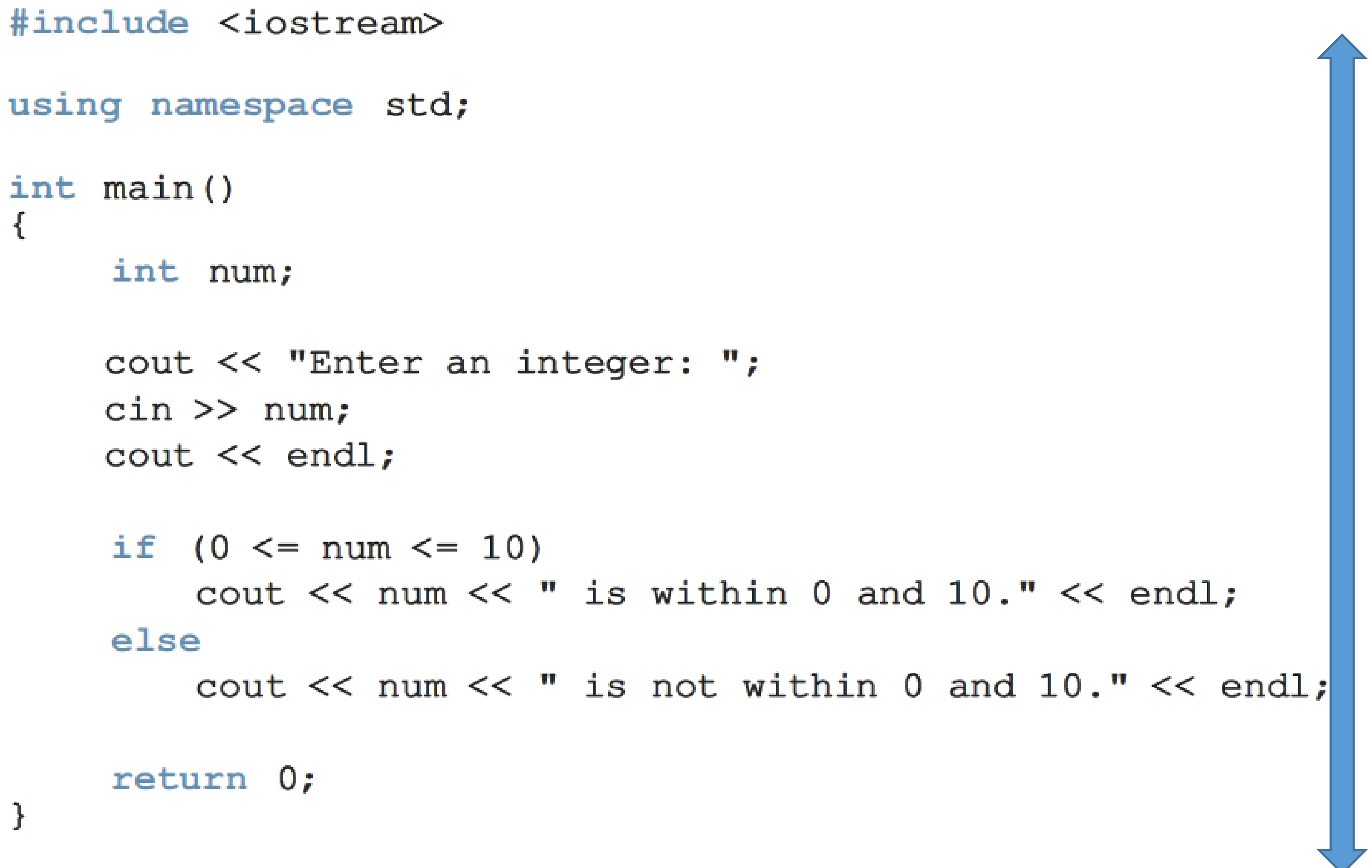
```
a.  if (month == 1)
        cout << "January" << endl;
    else if (month == 2)
        cout << "February" << endl;
    else if (month == 3)
        cout << "March" << endl;
    else if (month == 4)
        cout << "April" << endl;
    else if (month == 5)
        cout << "May" << endl;
    else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



```
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;
```

➢ **Which method is preferred?**

➢ Associativity of Relational Operators:



```
#include <iostream>

using namespace std;

int main()
{
     int num;

     cout << "Enter an integer: ";
     cin >> num;
     cout << endl;

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

     return 0;
}
```

Solution:

Sample Runs:

Sample Run 1:

Enter an integer: 5

5 is within 0 and 10. (correct)

Sample Run 2:

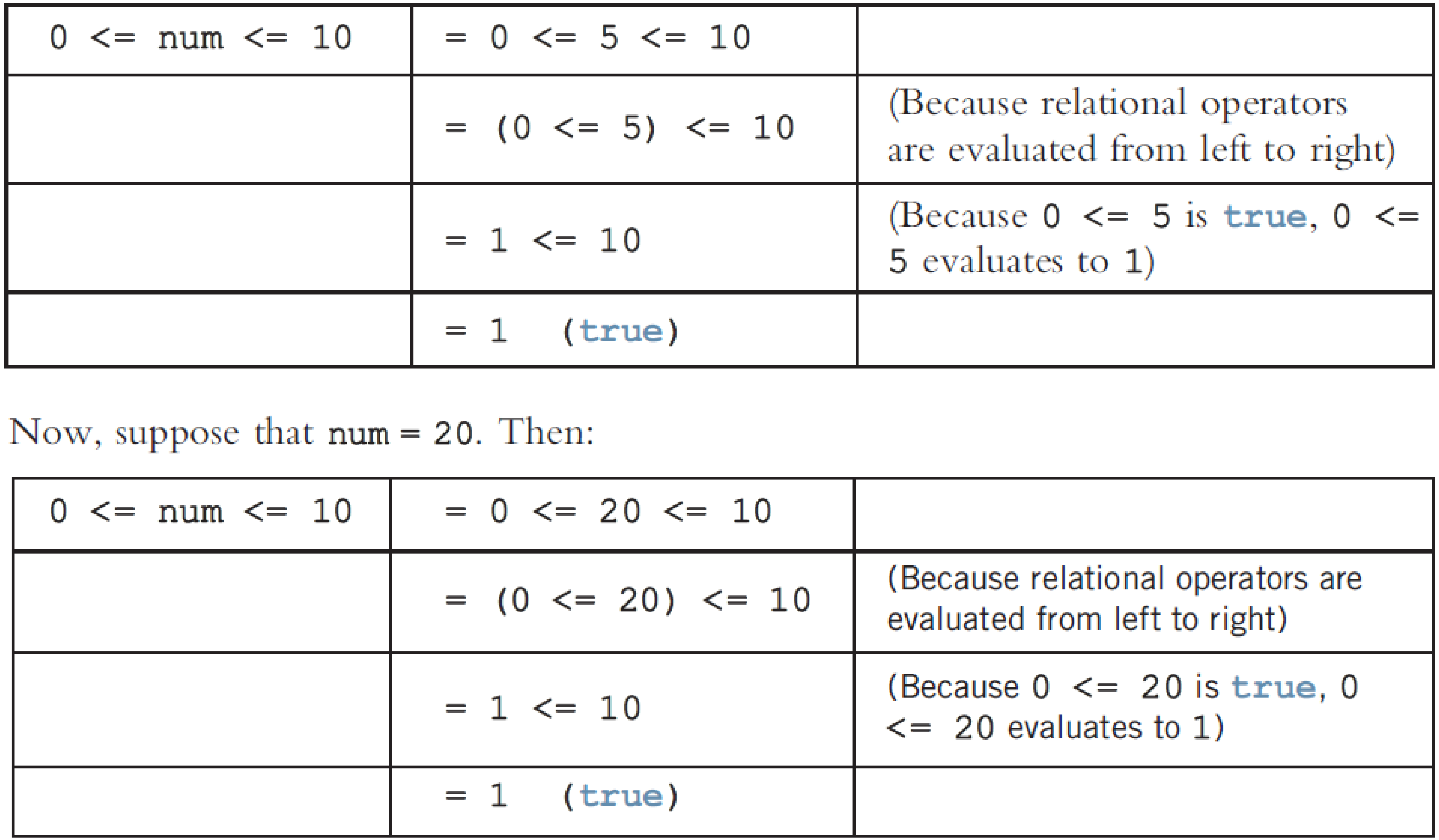
Enter an integer: 20

20 is within 0 and 10. (incorrect)

Sample Run 3:

Enter an integer: -10

-10 is within 0 and 10. (incorrect)



0 <= num <= 10	= 0 <= 5 <= 10	
	= (0 <= 5) <= 10	(Because relational operators are evaluated from left to right)
	= 1 <= 10	(Because 0 <= 5 is true, 0 <= 5 evaluates to 1)
	= 1 (true)	

Now, suppose that num = 20. Then:

0 <= num <= 10	= 0 <= 20 <= 10	
	= (0 <= 20) <= 10	(Because relational operators are evaluated from left to right)
	= 1 <= 10	(Because 0 <= 20 is true, 0 <= 20 evaluates to 1)
	= 1 (true)	

(0 <= num && num <= 10)

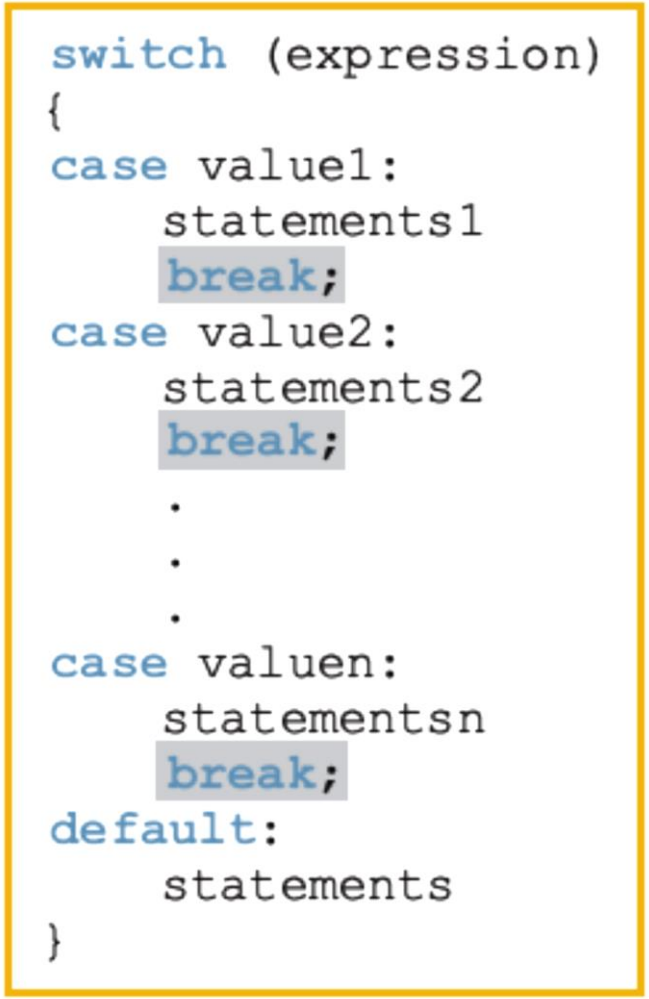
➢ switch Structures

- ❑ 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
}
```

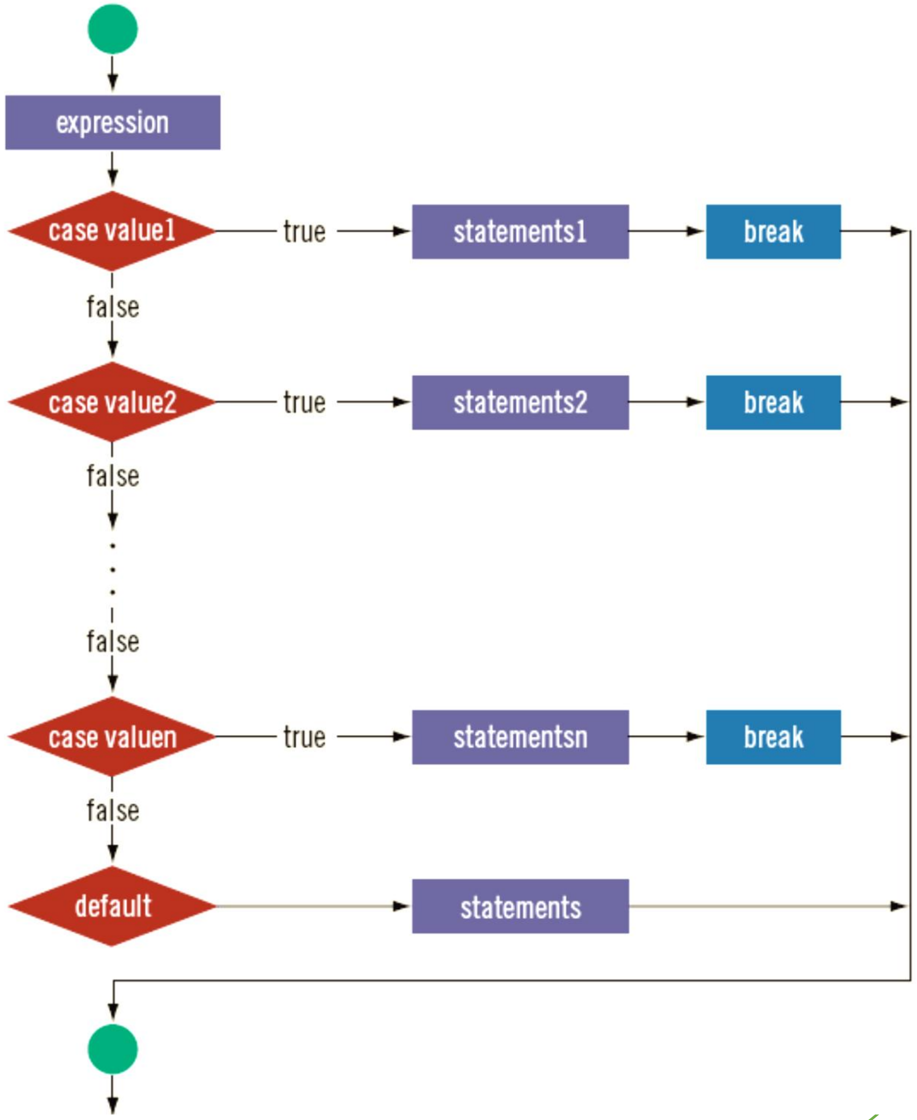
➢ switch Structures (cont.)

- ❑ One or more statements may follow a case label

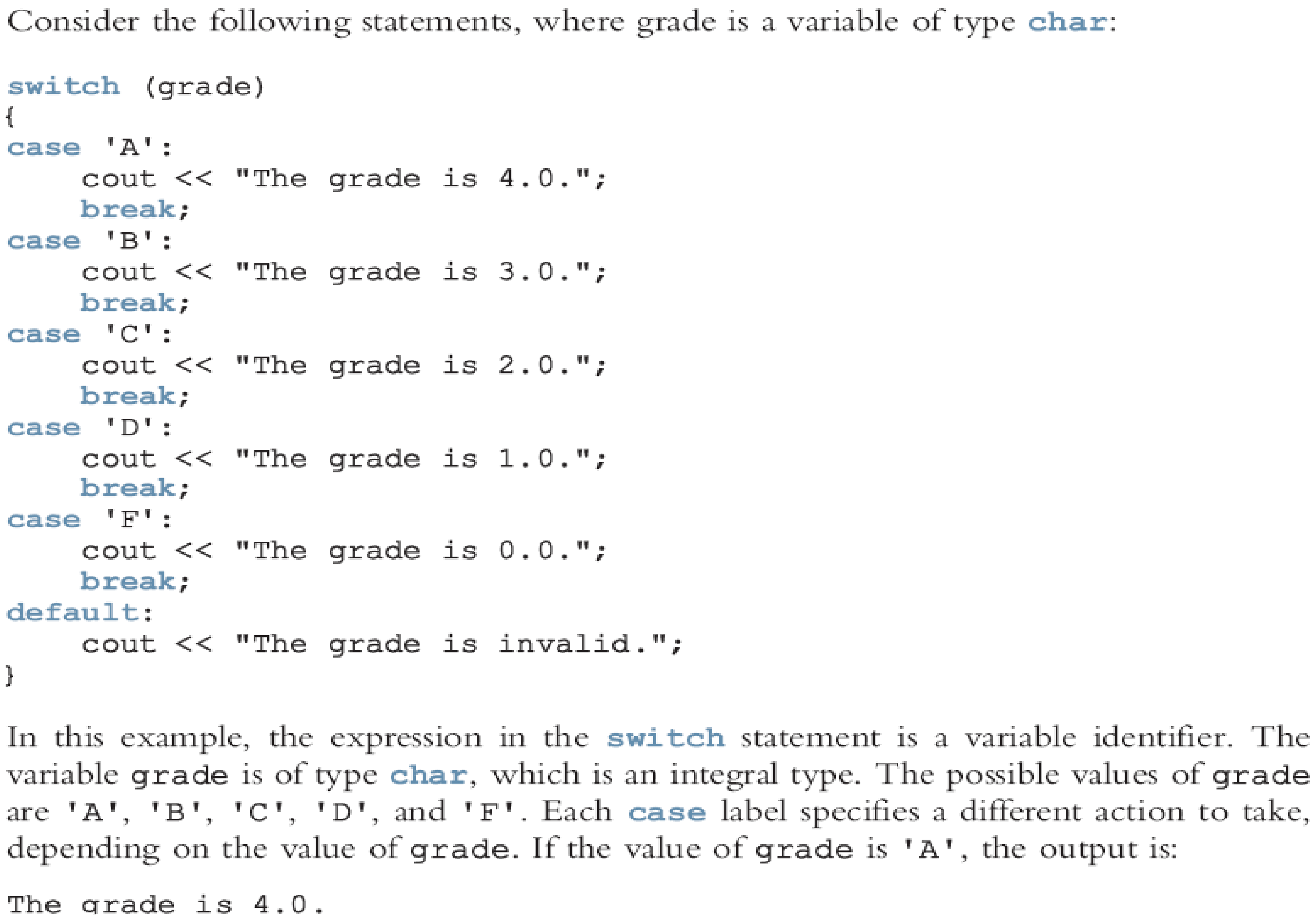
- ❑ Braces are not needed to turn multiple statements into a single compound statement

- ❑ The break statement may or may not appear after each statement

- ❑ switch, case, break, and default are reserved words



➢ Example



Consider the following statements, where grade is a variable of type char:

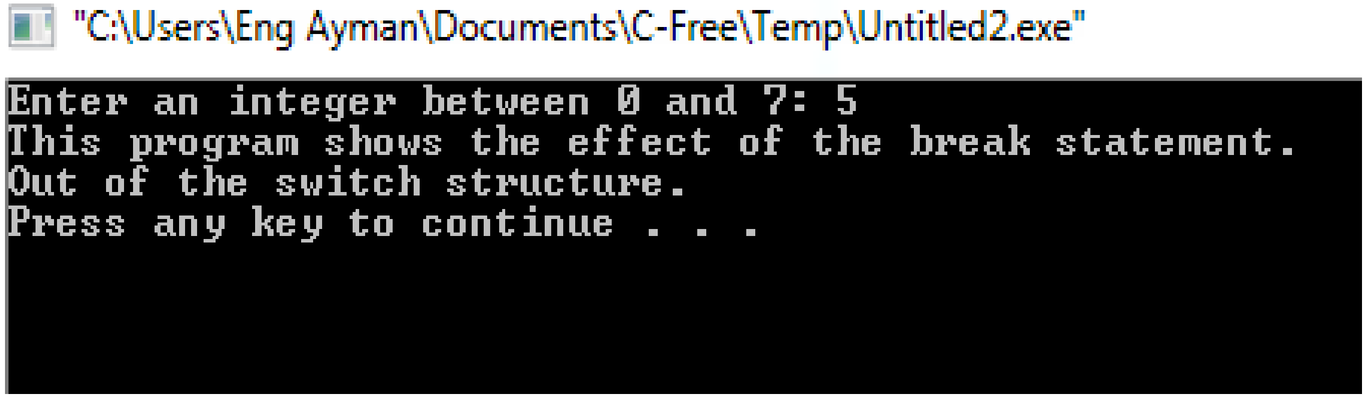
```
switch (grade)
{
case 'A':
    cout << "The grade is 4.0.";
    break;
case 'B':
    cout << "The grade is 3.0.";
    break;
case 'C':
    cout << "The grade is 2.0.";
    break;
case 'D':
    cout << "The grade is 1.0.";
    break;
case 'F':
    cout << "The grade is 0.0.";
    break;
default:
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

➤ Example (attention)

```
int main()
{                    int num;
                         cout << "Enter an integer between 0 and 7: ";
                         cin >> num;
            switch(num)
                         {
                         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 the break statement." << endl;
                         break;
                         default:
                         cout << "The number is out of range." << endl;
                         }
            cout << "Out of the switch structure." << endl;
            return 0;
            }
```



"C:\Users\Eng Ayman\Documents\C-Free\Temp\Untitled2.exe"

```
Enter an integer between 0 and 7: 5
This program shows the effect of the break statement.
Out of the switch structure.
Press any key to continue . . .
```

Thank
you