4

Decisions

4.1 Relational and Logical Operators

In Chapter 1, we discussed the two logical programming constructs *decision* and *loop*. In order to make a decision or control a loop, you need to specify a condition that determines the course of action.

A **condition** is an expression involving relational operators (such as < and =) that is either true or false. Conditions also may incorporate logical operators (such as And, Or, and Not). ANSI values determine the order used to compare strings with the relational operators. Boolean variables and literals can assume the values True or False.

ANSI Values

Each of the 47 different keys in the center typewriter portion of the keyboard can produce two characters, for a total of 94 characters. Adding 1 for the character produced by the space bar makes 95 characters. Associated with these characters are numbers ranging from 32 to 126. These values, called the ANSI (or ASCII) values of the characters, are given in Appendix A. Table 4.1 shows a few of them.

TABLE 4.1 A few ANSI values.

32 (space)	48 0	66 B	122 z
33 !	49 1	90 Z	123 {
34 "	57 9	97 a	125 }
35 #	65 A	98 b	126 ~

Most of the best-known fonts, such as Courier New, Microsoft San Serif, and Times New Roman, adhere to the ANSI standard, which assigns characters to the numbers from 0 to 255. Table 4.2 shows a few of the higher ANSI values.

TABLE 4.2 A few higher ANSI values.

162 ¢	177 ±	181 μ	190 ¾
169 ©	178 ²	188 ¼	247 ÷
176 °	179 ³	189 ½	248 ϕ

If *n* is a number between 0 and 255, then

```
Chr(n)
```

is the string consisting of the character with ANSI value *n*. If *str* is any string, then

```
Asc(str)
```

is the ANSI value of the first character of *str*. For instance, the statement

```
txtBox.Text = Chr(65)
```

displays the letter A in the text box, and the statement

```
lstBox.Items.Add(Asc("Apple"))
```

displays the number 65 in the list box.

Concatenation can be used with Chr to obtain strings using the higher ANSI characters. For instance, with one of the fonts that conforms to the ANSI standard, the statement

```
txtBox.Text = 32 & Chr(176) & " Fahrenheit"
```

displays 32° Fahrenheit in the text box.

The quotation-mark character (") can be placed into a string by using Chr(34). For example, after the statement

```
txtBox.Text = "George " & Chr(34) & "Babe" & Chr(34) & " Ruth"
```

is executed, the text box contains

```
George "Babe" Ruth
```

Relational Operators

VideoNote
Relational and logical operators

The relational operator *less than* (<) can be applied to numbers, strings, and dates. The number a is said to be less than the number b if a lies to the left of b on the number line. For instance, $2 < 5$, $-5 < -2$, and $0 < 3.5$.

The string a is said to be less than the string b if a precedes b alphabetically when using the ANSI table to alphabetize their values. For instance, "cat" < "dog", "cart" < "cat", and "cat" < "catalog". Digits precede uppercase letters, which precede lowercase letters. Two strings are compared working from left to right, character by character, to determine which one should precede the other. Therefore, "9W" < "bat", "Dog" < "cat", and "Sales-99" < "Sales-retail".

The date *d1* is said to be less than the date *d2* if *d1* precedes *d2* chronologically. For instance, #12/7/1941# < #6/6/1944#.

Table 4.3 shows the different relational operators and their meanings.

TABLE 4.3 Relational operators.

Visual Basic Notation	Numeric Meaning	String Meaning	Date Meaning
=	equal to	identical to	same as
<>	not equal to	different from	different than
<	less than	precedes alphabetically	precedes chronologically
>	greater than	follows alphabetically	follows chronologically
<=	less than or equal to	precedes alphabetically or is identical to	precedes chronologically or is the same as
>=	greater than or equal to	follows alphabetically or is identical to	follows chronologically or is the same as

Example 1 Determine whether each of the following conditions is true or false.

(a) 1 <= 1

(b) 1 < 1

(c) "car" < "cat"

(d) "Dog" < "dog"

(e) Today < Today.AddDays(1)

SOLUTION

(a) True. The notation <= means "less than *or* equal to." That is, the condition is true provided either of the two circumstances holds. The second one (equal to) holds.

(b) False. The notation < means "strictly less than" and no number can be strictly less than itself.

(c) True. The characters of the strings are compared one at a time working from left to right. Because the first two match, the third character decides the order.

(d) True. Because uppercase letters precede lowercase letters in the ANSI table, the first character of "Dog" precedes the first character of "dog".

(e) True. Today precedes tomorrow chronologically.

Conditions also can involve variables, numeric operators, and functions. To determine whether a condition is true or false, first evaluate the numeric or string expressions and then decide if the resulting assertion is true or false.

Example 2 Suppose the numeric variables *a* and *b* have values 4 and 3, and the string variables *c* and *d* have values "hello" and "bye". Are the following conditions true or false?

(a) (a + b) < 2*a

(b) (c.Length − b) = (a/2)

(c) c < ("good" & d)

SOLUTION

(a) The value of a + b is 7 and the value of 2*a is 8. Because 7 < 8, the condition is true.

(b) True, because the value of c.Length − b is 2, the same as (a / 2).

(c) The condition "hello" < "goodbye" is false, because "h" follows "g" in the ANSI table.

Logical Operators

Programming situations often require more complex conditions than those considered so far. For instance, suppose we would like to state that the value of a numeric variable, *n*, is strictly between 2 and 5. The proper Visual Basic condition is

(2 < n) And (n < 5)

The condition (2 < n) And (n < 5) is a combination of the two conditions 2 < n and n < 5 with the logical operator And.

The three main logical operators are And, Or, and Not. If *cond1* and *cond2* are conditions, then the condition

cond1 **And** ***cond2***

is true if both *cond1* and *cond2* are true. Otherwise, it is false. The condition

cond1 **Or** ***cond2***

is true if either *cond1* or *cond2* (or both) is true. Otherwise, it is false. The condition

Not ***cond1***

is true if *cond1* is false, and is false if *cond1* is true.

Example 3 Suppose the numeric variable *n* has value 4 and the string variable *answ* has value "Y". Determine whether each of the following conditions is true or false.

(a) (2 < n) And (n < 6)
(b) (2 < n) Or (n = 6)
(c) Not (n < 6)
(d) (answ = "Y") Or (answ = "y")
(e) (answ = "Y") And (answ = "y")
(f) Not (answ = "y")
(g) ((2 < n) And (n = 5 + 1)) Or (answ = "No")
(h) ((n = 2) And (n = 7)) Or (answ = "Y")
(i) (n = 2) And ((n = 7) Or (answ = "Y"))

SOLUTION

(a) True, because the conditions (2 < 4) and (4 < 6) are both true.
(b) True, because the condition (2 < 4) is true. The fact that the condition (4 = 6) is false does not affect the conclusion. The only requirement is that at least one of the two conditions be true.
(c) False, because (4 < 6) is true.
(d) True, because the first condition becomes ("Y" = "Y") when the value of *answ* is substituted for *answ*.
(e) False, because the second condition is false. Actually, this compound condition is false for every value of *answ*.
(f) True, because ("Y" = "y") is false.
(g) False. In this logical expression, the compound condition ((2 < n) And (n = 5 + 1)) and the simple condition (answ = "No") are joined by the logical operator Or. Because both these conditions are false, the total condition is false.
(h) True, because the second Or clause is true.
(i) False. Comparing (h) and (i) shows the necessity of using parentheses to specify the intended grouping.

Boolean Data Type

A statement of the form

```
txtBox.Text = CStr(condition)
```

will display either True or False in the text box, depending on the condition. Any variable or expression that evaluates to either True or False is said to have a **Boolean data type**. The following lines of code display False in the text box.

```
Dim x As Integer = 5
txtBox.Text = CStr((3 + x) < 7)
```

A variable is declared to be of type Boolean with a statement of the form

```
Dim varName As Boolean
```

The following lines of code will display True in the text box.

```
Dim boolVar As Boolean
Dim x As Integer = 2
Dim y As Integer = 3
boolVar = x < y
txtBox.Text = CStr(boolVar)
```

The answer to part (i) of Example 3 can be confirmed to be False by executing the following lines of code.

```
Dim n as Integer = 4
Dim answ as String = "Y"
txtBox.Text = CStr((n = 2) And ((n = 7) Or (answ = "Y")))
```

Two Boolean-Valued Methods

If *strVar* is a string variable, then the expression

```
strVar.Substring(strVar.length – 3) = "ing"
```

is true if and only if the value of *strVar* ends with *ing*. To generalize, if *strVar2* is another string variable, then the expression

```
strVar.SubString(strVar.length - strVar2.Length) = strVar2          (1)
```

is true if and only if the value of *strVar* ends with the value of *strVar2*.

The EndsWith method provides an alternate way to test for the end of a string. The value of

```
strVar.EndsWith(strVar2)                                            (2)
```

is True if and only if the value of *strVar* ends with the value of *strVar2*.

Condition (2) is preferable to condition (1) since it is more concise and readable. Code such as condition (2) is called **declarative code** (or **self-evident code**), since it clearly declares *what* you want to accomplish. Expression (1) shows *how* to accomplish the task but requires some deciphering in order to figure out what is being achieved. One of our guiding programming principles will be to write declarative code whenever possible. We prefer *what* to *how*.

The counterpart of the EndsWith method is the StartsWith method. The value of

```
strVar.StartsWith(strVar2)
```

is True if and only if the value of *strVar* begins with the value of *strVar2*.

A Boolean-Valued Function

The IsNumeric function is used to determine if a value input by the user, say in a text box or input dialog box, can be used in numeric computations. The value of

```
IsNumeric(strVar)
```

is True if *strVar* can be converted to a number with CInt or CDbl, and is False otherwise. For instance, `IsNumeric(strVar)` will be True when the value of *strVar* is "2345", "$123", or "5,677,890". The function value will be False when the value of *strVar* is "five" or "4 – 2".

Comments

1. A condition involving numeric variables is different from an algebraic truth. The assertion (a + b) < 2*a, considered in Example 2, is not a valid algebraic truth because it isn't true for all values of *a* and *b*. When encountered in a Visual Basic program, however, it will be considered true if it is correct for the current values of the variables.
2. Conditions evaluate to either True or False. These two values often are called the possible **truth values** of the condition.
3. A condition such as 2 < n < 5 should never be used, because Visual Basic will not evaluate it as intended. The correct condition is (2 < n) And (n < 5).
4. A common error is to replace the condition Not (n < m) by the condition (n > m). The correct replacement is (n >= m).

Practice Problems 4.1

1. Is the condition `"Hello " = "Hello"` true or false?
2. Explain why (27 > 9) is true, whereas ("27" > "9") is false.
3. Complete Table 4.4.

TABLE 4.4 Truth values of logical operators.

cond1	cond2	cond1 And cond2	cond1 Or cond2	Not cond2
True	True	True		
True	False		True	
False	True			False
False	False			

EXERCISES 4.1

In Exercises 1 through 6, determine the output displayed in the text box.

1. `txtBox.Text = Chr(104) & Chr(105)`
2. `txtBox.Text = "C" & Chr(35)`
3. `txtBox.Text = "The letter before G is " & Chr(Asc("G") - 1)`
4. `txtBox.Text = Chr(Asc("B")) 'The ANSI value of B is 66`
5.
```
Dim quote, person, qMark As String
quote = "We're all in this alone."
person = "Lily Tomlin"
qMark = Chr(34)
txtBox.Text = qMark & quote & qMark & " - " & person
```
6.
```
Dim letter As String
letter = "D"
txtBox.Text = letter & " is the " & (Asc(letter) - Asc("A") + 1) &
              "th letter of the alphabet."
```

In Exercises 7 through 18, determine whether the condition is true or false. Assume a = 2 and b = 3.

7. 3*a = 2*b
8. (5 − a)*b < 7
9. b <= 3
10. a^b = b^a
11. a^(5 − 2) > 7
12. 3E−02 < .01*a
13. (a < b) Or (b < a)
14. (a*a < b) Or Not (a*a < a)
15. Not ((a < b) And (a < (b + a)))
16. Not (a < b) Or Not (a < (b + a))
17. ((a = b) And (a*a < b*b)) Or ((b < a) And (2*a < b))
18. ((a = b) Or Not (b < a)) And ((a < b) Or (b = a + 1))

In Exercises 19 through 30, determine whether the condition is true or false.

19. "9W" <> "9w"
20. "Inspector" < "gadget"
21. "Car" < "Train"
22. "J" >= "J"
23. "99" > "ninety-nine"
24. "B" > "?"
25. ("Duck" < "pig") And ("pig" < "big")
26. "Duck" < "Duck" & "Duck"
27. Not (("B" = "b") Or ("Big" < "big"))
28. #7/4/1776# >= #7/4/1776#
29. #6/17/1775# <= #7/4/1776#
30. (7 < 34) And ("7" > "34")

In Exercises 31 through 40, determine whether or not the two conditions are equivalent—that is, whether they will be true or false for exactly the same values of the variables appearing in them.

31. a <= b; (a < b) Or (a = b)
32. Not (a < b); a > b
33. (a = b) And (a < b); a <> b
34. Not ((a = b) Or (a = c)); (a <> b) And (a <> c)
35. (a < b) And ((a > d) Or (a > e));
    ((a < b) And (a > d)) Or ((a < b) And (a > e))
36. Not ((a = b + c) Or (a = b)); (a <> b) Or (a <> b + c)
37. (a < b + c) Or (a = b + c); Not ((a > b) Or (a > c))
38. Not (a >= b); (a <= b) Or Not (a = b)
39. Not (a >= b); (a<= b) And Not (a = b)
40. (a = b) And ((b = c) Or (a = c)); (a = b) Or ((b = c) And (a = c))

In Exercises 41 through 45, write a condition equivalent to the negation of the given condition. (For example, a <> b is equivalent to the negation of a = b.)

41. a > b
42. (a = b) Or (a = d)
43. (a < b) And (c <> d)
44. Not ((a = b) Or (a > b))
45. (a <> "") And (a < b) And (a.Length < 5)
46. Rework Exercise 20 by evaluating the Boolean expression in a program.
47. Rework Exercise 21 by evaluating the Boolean expression in a program.
48. Rework Exercise 22 by evaluating the Boolean expression in a program.
49. Rework Exercise 23 by evaluating the Boolean expression in a program.

In Exercises 50 through 59, determine whether True or False is displayed in the text box.

50.
```
Dim str As String = "target"
txtBox.Text = CStr(str.StartsWith("t") And str.EndsWith("t"))
```

51.
```
Dim str As String = "ticket"
txtBox.Text = CStr(str.StartsWith("T") Or str.EndsWith("T"))
```

52.
```
Dim str1 As String = "target"
Dim str2 As String = "get"
txtBox.Text = CStr(str1.EndsWith(str2))
```

53.
```
Dim str1 As String = "Teapot"
Dim str2 As String = "Tea"
txtBox.Text = CStr(str1.StartsWith(str2))
```

54.
```
Dim str As String = "$1,234.56"
txtBox.Text = CStr(IsNumeric(str))
```

55.
```
Dim str As String = "10,000,000"
txtBox.Text = CStr(IsNumeric(str))
```

56.
```
Dim str As String = "10 million"
txtBox.Text = CStr(IsNumeric(str))
```

57.
```
Dim str As String = "2 + 3"
txtBox.Text = CStr(IsNumeric(str))
```

58.
```
Dim str As String = "10E+06"
txtBox.Text = CStr(IsNumeric(str))
```

59.
```
Dim str As String = "5E-12"
txtBox.Text = CStr(IsNumeric(str))
```

Solutions to Practice Problems 4.1

1. False. The first string has six characters, whereas the second has five. Two strings must be 100% identical to be called equal.
2. When 27 and 9 are compared as strings, their first characters, *2* and 9, determine their order. Since *2* precedes 9 in the ANSI table, "27" < "9".
3.

cond1	cond2	cond1 And cond2	cond1 Or cond2	Not cond2
True	True	True	True	False
True	False	False	True	True
False	True	False	True	False
False	False	False	False	True