```
  If clickFlag Then
    MessageBox.Show(msg & "clicking on an item of the list box.")
  Else
    MessageBox.Show(msg & "pressing an arrow key.")
  End If
  clickFlag = False
End Sub
```

2. The ordering in the list box is determined by the ANSI table (where the items are treated as strings), not the numerical value. Therefore the last item in the list box might not have the greatest numerical value.

CHAPTER 6 SUMMARY

1. A *Do loop* repeatedly executes a block of statements either as long as or until a certain condition is true. The condition can be checked either at the top of the loop or at the bottom.
2. A *For . . . Next loop* repeats a block of statements a fixed number of times. The *counter variable* assumes an initial value and increases it by one after each pass through the loop until it reaches the terminating value. Alternative increment values can be specified with the *Step* keyword.
3. Visual Basic uses *local type inference* to infer the data types of local variables declared without an As clause by looking at the data type of the initialization expression.
4. The items in a list box are assigned *index numbers* ranging from 0 to [number of items minus 1]. Loops can use the index numbers to extract information from list boxes.
5. A *flag* is a Boolean variable used to indicate whether a certain event has occurred or a certain situation exists.

CHAPTER 6 PROGRAMMING PROJECTS

1. *Caffeine Absorption*. After caffeine is absorbed into the body, 13% is eliminated from the body each hour. Assume a person drinks an 8-oz cup of brewed coffee containing 130 mg of caffeine, and the caffeine is absorbed immediately into the body. Write a program to compute the following values. See Fig. 6.12.
   (a) The number of hours required until 65 mg (one-half the original amount) remain in the body.

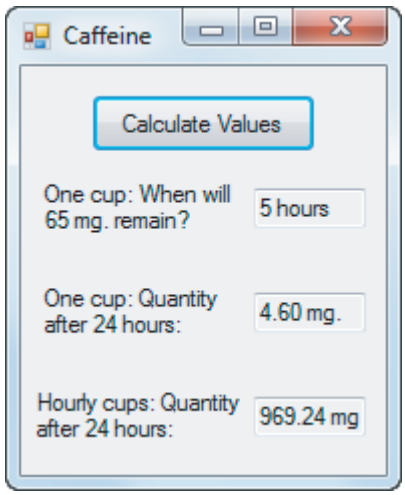

FIGURE 6.12 Output of Programming Project 1.

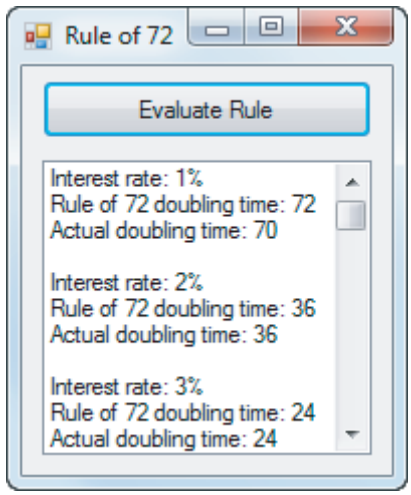

FIGURE 6.13 Output of Programming Project 2.

(b) The amount of caffeine in the body 24 hours after the person drinks the coffee.

(c) Suppose the person drinks a cup of coffee at 7 a.m. and then drinks a cup of coffee at the end of each hour until 7 a.m. the next day. How much caffeine will be in the body at the end of the 24 hours?

2. The *Rule of 72* is used to approximate the time required for prices to double due to inflation. If the inflation rate is r%, then the Rule of 72 estimates that prices will double in $72/r$ years. For instance, at an inflation rate of 6%, prices double in about 72/6 or 12 years. Write a program to test the accuracy of this rule. For each interest rate from 1% to 20%, the program should display the rounded value of $72/r$ and the actual number of years required for prices to double at an r% inflation rate. (Assume prices increase at the end of each year.) See Fig. 6.13.

3. Write a program to provide information on the height of a ball thrown straight up into the air. The program should request as input the initial height, h feet, and the initial velocity, v feet per second. The height of the ball (in feet) after t seconds is given by the formula $h + vt - 16t^2$ feet. The four options to be provided by buttons are as follows:

(a) Determine the maximum height of the ball. ***Note:*** The ball will reach its maximum height after $v/32$ seconds.

(b) Determine approximately when the ball will hit the ground. ***Hint:*** Calculate the height after every .1 second and determine when the height is no longer a positive number.

(c) Display a table showing the height of the ball every quarter second for five seconds or until it hits the ground. See Fig. 6.14.

(d) Quit.

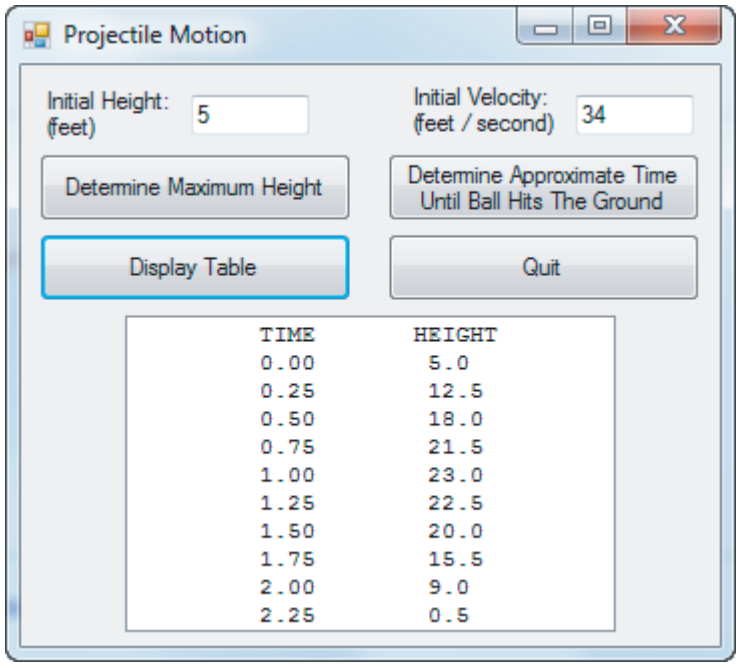

FIGURE 6.14 Sample Output for Programming Project 3.

4. A *palindrome* is a word or phrase that reads the same forwards and backwards, character for character, disregarding punctuation, case, and spaces. Some examples are "racecar", "Madam, I'm Adam.", and "Was it a cat I saw?". Write a program that allows the user to input a word or phrase and then determines if it is a palindrome. The program should use a Boolean-valued Function procedure named IsPalindrome that returns the value True when the word or phrase is a palindrome and the value False otherwise.

5. The following words have three consecutive letters that are also consecutive letters in the alphabet: THI**RST**Y, A**FGH**ANISTAN, **STU**DENT. Write a program that accepts a word as input and determines whether or not it has three consecutive letters that are consecutive letters in the alphabet. The program should use a Boolean-valued function named IsTripleConsecutive that accepts an entire word as input. ***Hint:*** Use the Asc function.

6. Write a program that uses a flag and does the following:

(a) Ask the user to input a sentence containing parentheses. ***Note:*** The closing parenthesis should not directly precede the period.

(b) Display the sentence with the parentheses and their contents removed. Test the program with the following sentence as input: BASIC (Beginner's All-purpose Symbolic Instruction Code) was once the world's most widely known computer language.

7. *Depreciation to a Salvage Value of 0*. For tax purposes an item may be depreciated over a period of several years, *n*. With the *straight-line* method of depreciation, each year the item depreciates by 1/*n*th of its original value. With the *double-declining-balance* method of depreciation, each year the item depreciates by 2/*n*ths of its value at the beginning of that year. (In the final year it is depreciated by its value at the beginning of the year.) Write a program that performs the following tasks:

(a) Request a description of the item, the year of purchase, the cost of the item, the number of years to be depreciated (estimated life), and the method of depreciation. The method of depreciation should be chosen by clicking on one of two buttons.

(b) Display a year-by-year description of the depreciation. See Fig. 6.15.

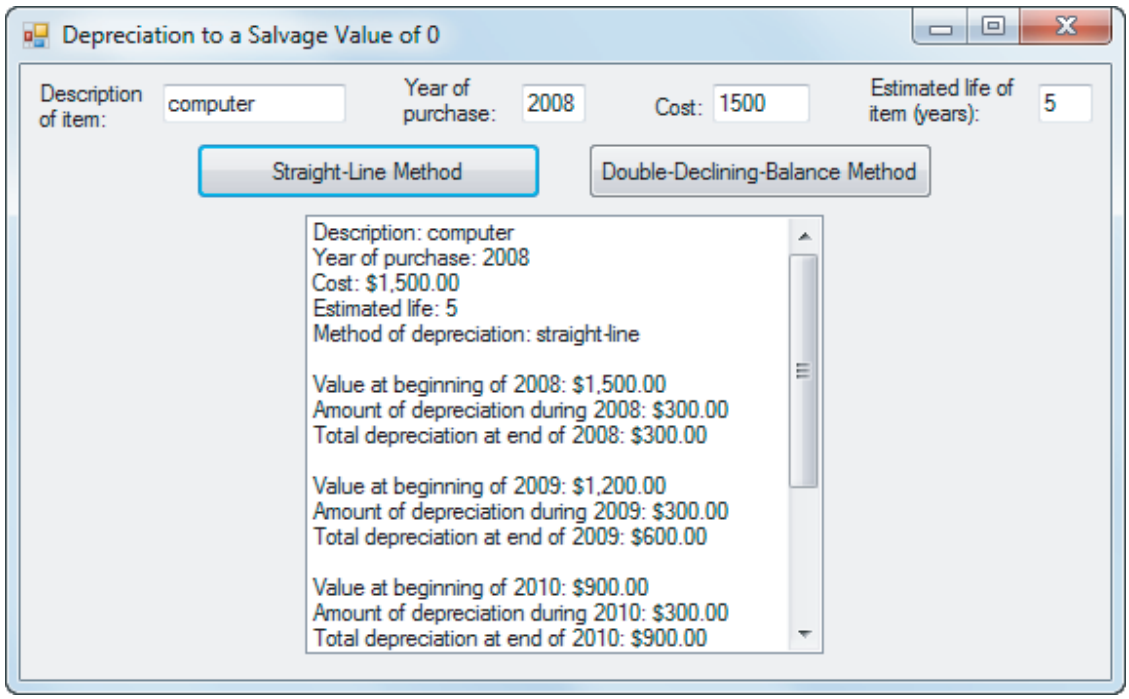

FIGURE 6.15 **Sample output of Programming Project 7.**

8. An especially efficient technique for searching an ordered list of items is called a **binary search**. A binary search looks for a value by first determining in which half of the list it resides. The other half of the list is then ignored, and the retained half is temporarily regarded as the entire list. The process is repeated until the item is found or the entire list

has been considered. Use the algorithm and flowchart for a binary search shown below to rewrite the btnSearch_Click event procedure from Example 4 of Section 6.3.

Figure 6.16 shows a partial flowchart for a binary search. (The sought-after value is denoted by *quarry*. The Boolean variable *flag* keeps track of whether or not *quarry* has been found.) The algorithm for a binary search of the items in an ordered list box is as follows:

(i) At each stage, denote the index of the first item in the retained list by *first* and the index of the last item by *last*. Initially, set the value of *first* to 0, set the value of *last* to one less than the number of items in the list, and set the value of *flag* to False.

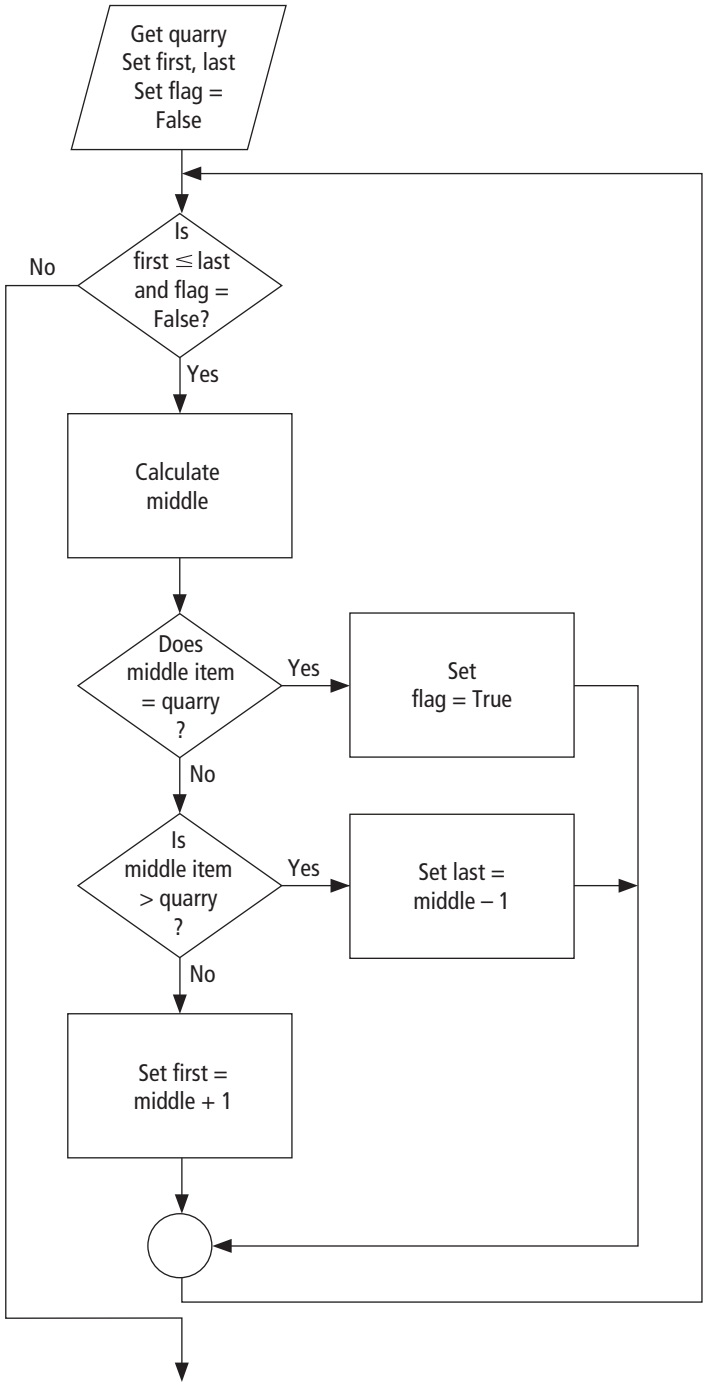

FIGURE 6.16 **Flowchart for the loop portion of a binary search.**

(ii) Look at the middle item of the current list—the item having index *middle* = CInt((*first* + *last*)/2).

(iii) If the middle item is *quarry*, then set *flag* to True and end the search.

(iv) If the middle item is greater than *quarry*, then *quarry* should be in the first half of the list. So the index of *quarry* must lie between *first* and *middle* − 1. Set *last* to *middle* − 1.

(v) If the middle item is less than *quarry*, then *quarry* should be in the second half of the list of possible items. So the index of *quarry* must lie between *middle*+ 1 and *last*. Set *first* to *middle*+ 1.

(vi) Repeat steps (ii) through (v) until *quarry* is found or until the halving process uses up the entire list. (When the entire list has been used up, *first* > *last*.) In the second case, *quarry* was not in the original list.