2. Both arrays of structures and two-dimensional arrays are used to hold related data. If some of the data are numeric and some are string, then structures must be used, because all entries of a two-dimensional array must be of the same type. Arrays of structures should also be used if the data will be sorted. Two-dimensional arrays are best suited to tabular data.

7.5 A Case Study: Analyze a Loan

This case study develops a program to analyze a loan. Assume the loan is to be repaid in equal monthly payments and interest is compounded monthly. The program should request the amount (principal) of the loan, the annual rate of interest, and the number of years over which the loan is to be repaid. The five options to be provided by buttons are as follows:

1. Calculate the monthly payment. The formula is

$$[\text{monthly payment}] = \frac{p \cdot r}{1 - (1 + r)^{-n}}$$

   where p is the principal of the loan, r is the monthly interest rate (annual rate divided by 12) given as a number between 0 (for 0 percent) and 1 (for 100 percent), and n is the number of months over which the loan is to be repaid. When a payment computed in this manner results in fractions of a cent, the value should be rounded up to the next nearest cent. This corrected payment can be achieved using the formula

   [corrected payment] = **`Math.Round(`*`originalPayment`*` + 0.005, 2)`**

2. Display an amortization schedule—that is, a table showing for each month the amount of interest paid, the amount of principal repaid, and the balance on the loan at the end of the month. At any time, the balance of the loan is the amount of money that must be paid in order to retire the loan. The monthly payment consists of two parts—interest on the balance and repayment of part of the principal. Each month

$$[\text{interest payment}] = r * [\text{balance at beginning of month}]$$

$$[\text{amount of principal repaid}] = [\text{monthly payment}] - [\text{interest payment}]$$

$$[\text{new balance}] = [\text{balance at beginning of month}] - [\text{amount of principal repaid}]$$

3. Calculate the interest paid during a calendar year. (This amount is deductible when itemizing deductions on a Federal income tax return.) The user should specify the number of the payment made in January of that year. For instance, if the first payment was made in September 2009, then the payment made in January 2010 would be payment number 5.
4. Show the effect of changes in the interest rate. Display a table giving the monthly payment for each interest rate from 1% below to 1% above the specified annual rate in steps of one-eighth of 1%.
5. Quit.

The User Interface

Figure 7.31 shows a possible form design and Table 7.15 gives the initial settings for the form and its controls. Figures 7.32, 7.33, 7.34, and 7.35 show possible outputs of the program for each task available through the buttons.

Designing the Analyze-a-Loan Program

Every routine uses data from the three text boxes. Therefore, we create a Sub procedure to read the contents of the text boxes and convert the contents into a usable form. Two of the routines

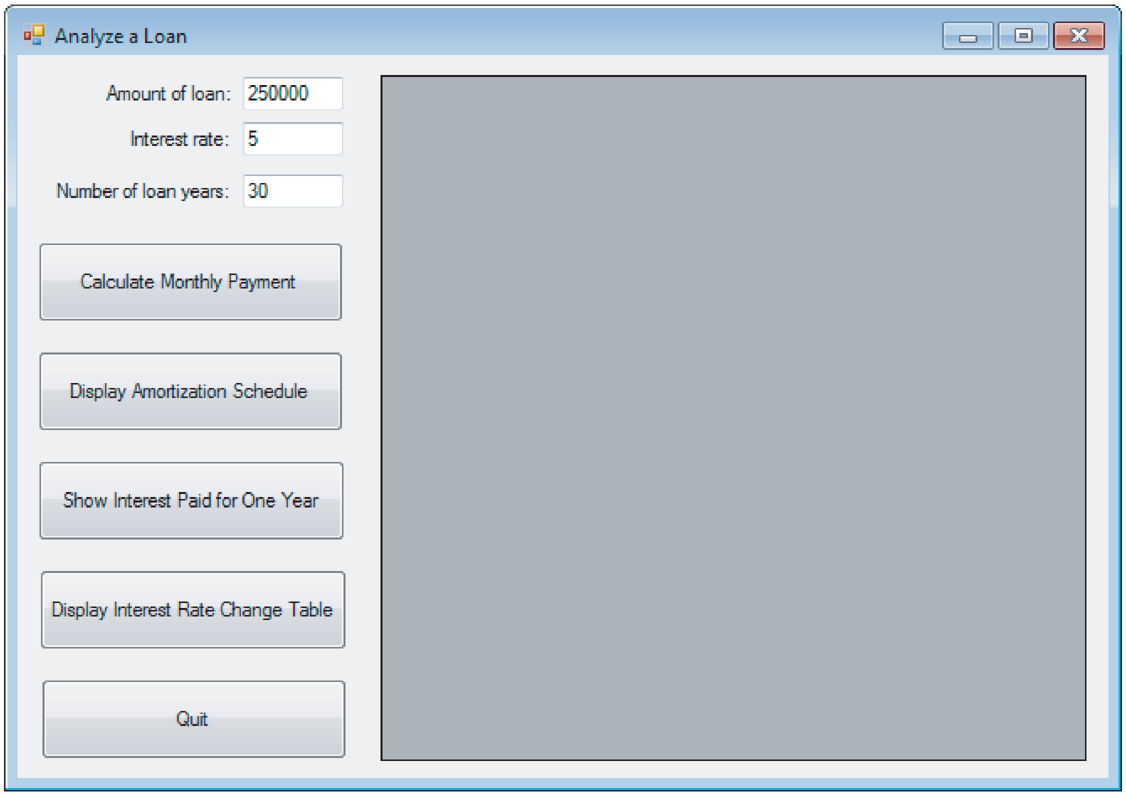
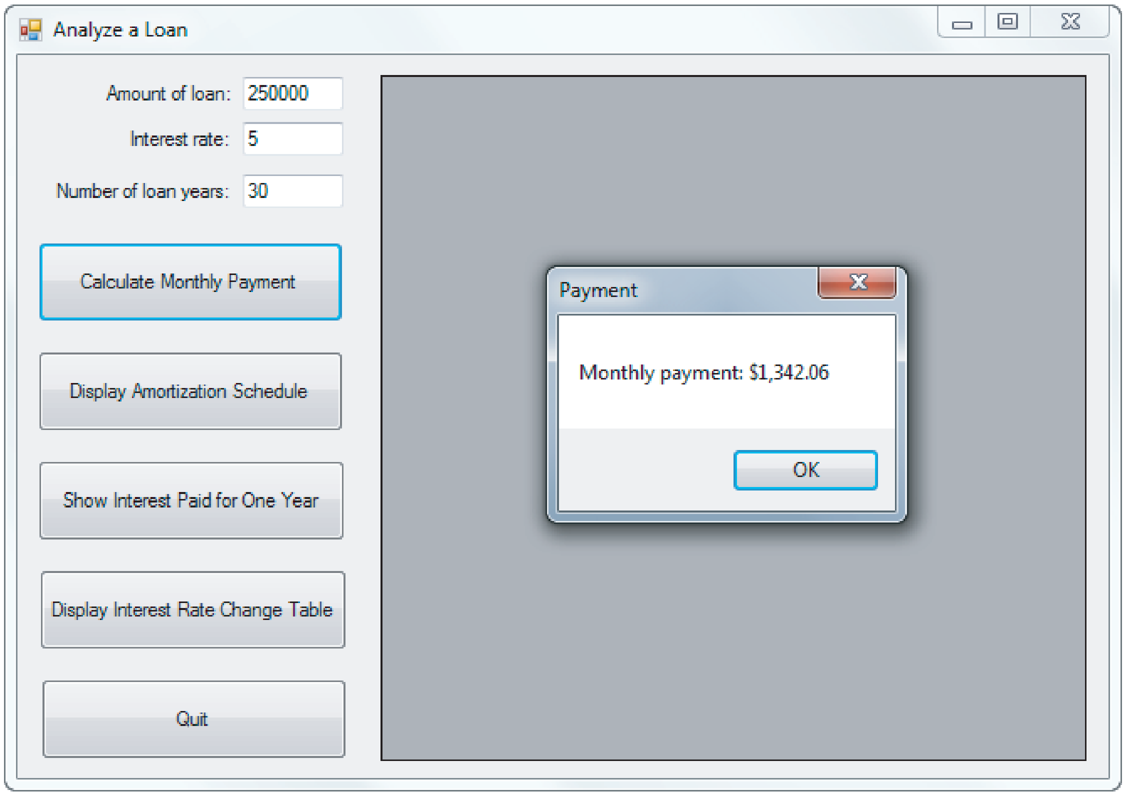

Analyze a Loan
Amount of loan: 250000
Interest rate: 5
Number of loan years: 30
Calculate Monthly Payment
Display Amortization Schedule
Show Interest Paid for One Year
Display Interest Rate Change Table
Quit

FIGURE 7.31 Template for the Analyze-a-Loan program.

TABLE 7.15 Objects and initial properties for the Analyze-a-Loan program.

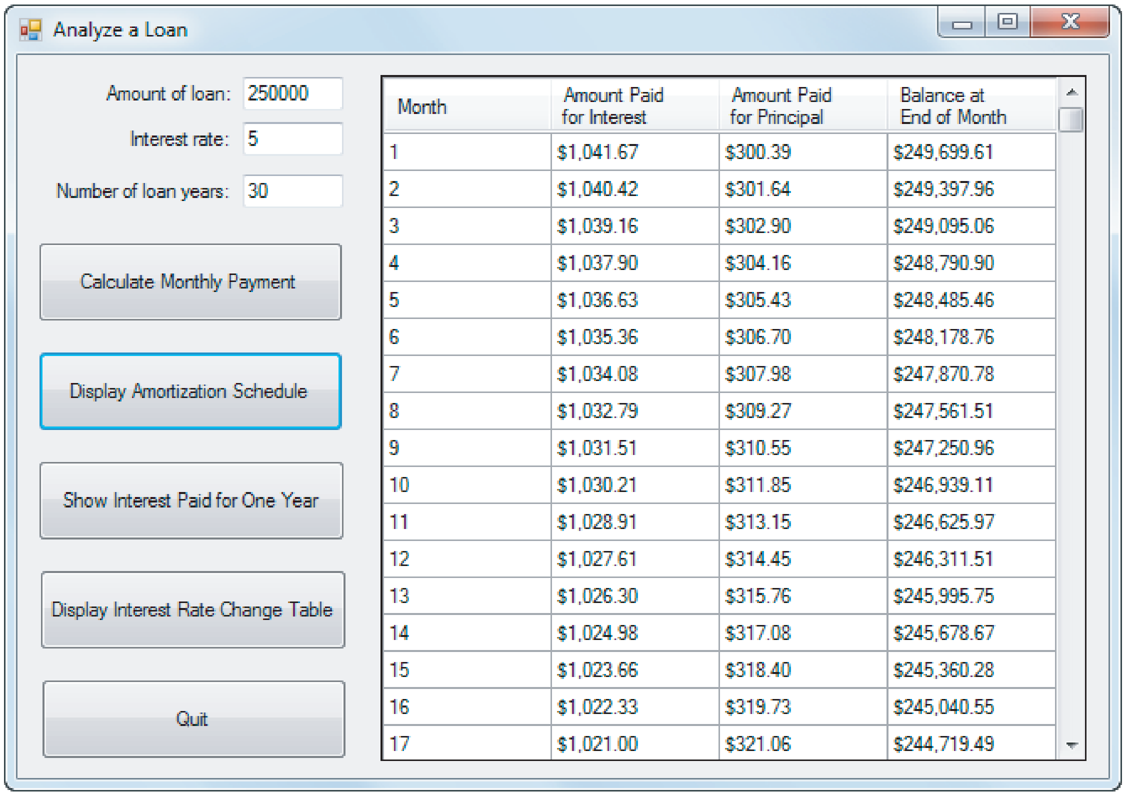
Object	Property	Setting
frmLoan	Text	Analysis of a Loan
lblPrincipal	Text	Amount of loan:
txtPrincipal		
lblYearlyRate	Text	Interest rate:
txtYearlyRate		
lblNumYears	Text	Number of loan years:
txtNumYears		
btnPayment	Text	Calculate Monthly Payment
btnAmort	Text	Display Amortization Schedule
btnShow	Text	Show Interest Paid for One Year
btnRateTable	Text	Display Interest Rate Change Table
btnQuit	Text	Quit
dgvOutput	RowHeaderVisible	False

display extensive tables in a DataGridView control. The simplest way to fill a table is to use an array of structures and a LINQ query. We will need two types of structures—one (named Month) to hold the amortization data for a month and the other (named EffectOfRate) to hold monthly payments for different interest rates. Since the array of Month structures is needed by two routines, we use a Function procedure to fill it so we won't have to duplicate the lengthy code. Since the value of the monthly payment is needed several times, we include a Function procedure called Payment.


Analyze a Loan
Amount of loan: 250000
Interest rate: 5
Number of loan years: 30
Calculate Monthly Payment
Display Amortization Schedule
Show Interest Paid for One Year
Display Interest Rate Change Table
Quit
Payment
OK

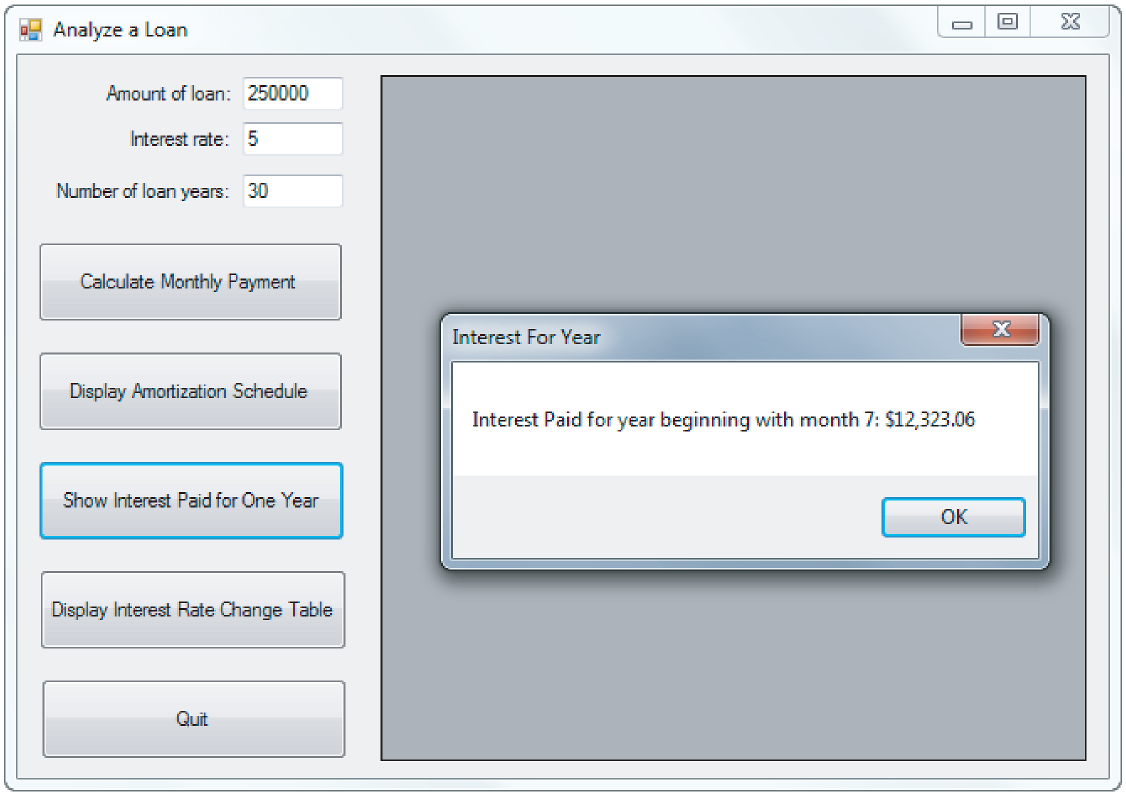
Monthly payment: $1,342.06

FIGURE 7.32 Monthly payment for a loan.


Analyze a Loan
Amount of loan: 250000
Interest rate: 5
Number of loan years: 30
Calculate Monthly Payment
Display Amortization Schedule
Show Interest Paid for One Year
Display Interest Rate Change Table
Quit

Month	Amount Paid for Interest	Amount Paid for Principal	Balance at End of Month
1	$1,041.67	$300.39	$249,699.61
2	$1,040.42	$301.64	$249,397.96
3	$1,039.16	$302.90	$249,095.06
4	$1,037.90	$304.16	$248,790.90
5	$1,036.63	$305.43	$248,485.46
6	$1,035.36	$306.70	$248,178.76
7	$1,034.08	$307.98	$247,870.78
8	$1,032.79	$309.27	$247,561.51
9	$1,031.51	$310.55	$247,250.96
10	$1,030.21	$311.85	$246,939.11
11	$1,028.91	$313.15	$246,625.97
12	$1,027.61	$314.45	$246,311.51
13	$1,026.30	$315.76	$245,995.75
14	$1,024.98	$317.08	$245,678.67
15	$1,023.66	$318.40	$245,360.28
16	$1,022.33	$319.73	$245,040.55
17	$1,021.00	$321.06	$244,719.49

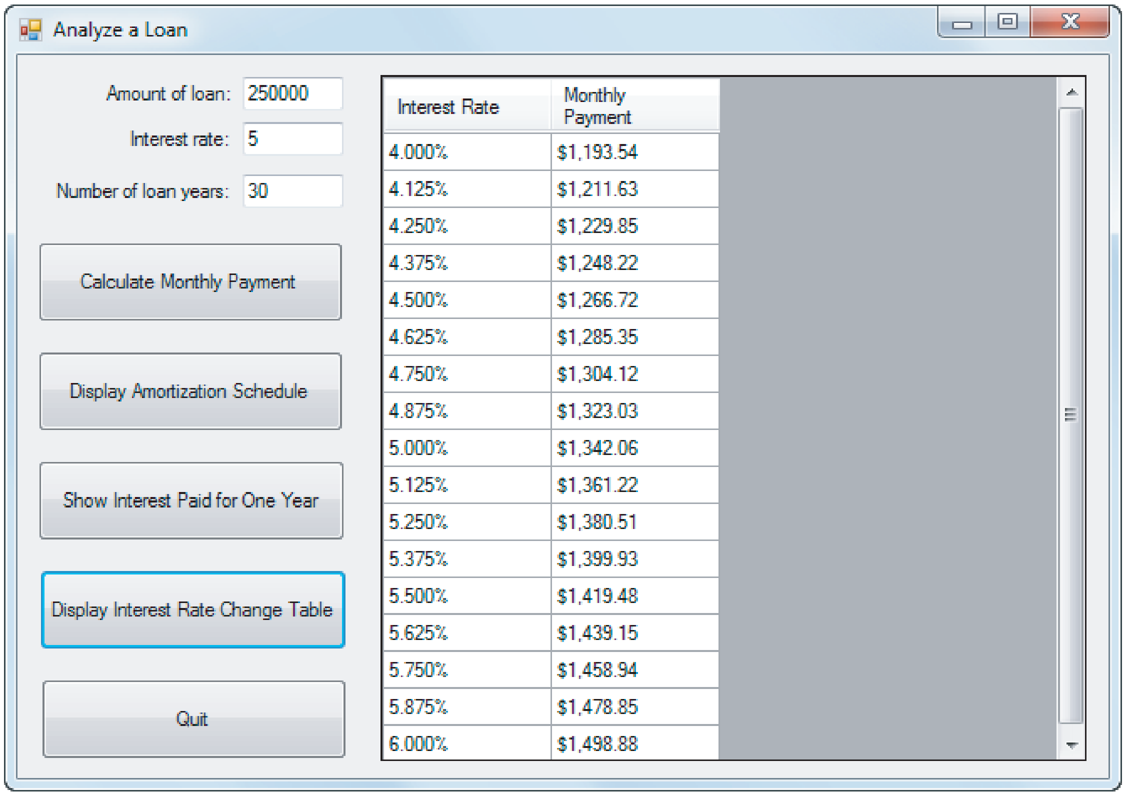
FIGURE 7.33 Amortization schedule for a loan.


Analyze a Loan
Calculate Monthly Payment
Display Amortization Schedule
Show Interest Paid for One Year
Display Interest Rate Change Table
Quit
Interest For Year
OK

Amount of loan: 250000
Interest rate: 5
Number of loan years: 30

Interest Paid for year beginning with month 7: $12,323.06

FIGURE 7.34 Interest paid during one year.


Analyze a Loan
Calculate Monthly Payment
Display Amortization Schedule
Show Interest Paid for One Year
Display Interest Rate Change Table
Quit

Amount of loan: 250000
Interest rate: 5
Number of loan years: 30

Interest Rate	Monthly Payment
4.000%	$1,193.54
4.125%	$1,211.63
4.250%	$1,229.85
4.375%	$1,248.22
4.500%	$1,266.72
4.625%	$1,285.35
4.750%	$1,304.12
4.875%	$1,323.03
5.000%	$1,342.06
5.125%	$1,361.22
5.250%	$1,380.51
5.375%	$1,399.93
5.500%	$1,419.48
5.625%	$1,439.15
5.750%	$1,458.94
5.875%	$1,478.85
6.000%	$1,498.88

FIGURE 7.35 Consequences of interest rate change.

The program is divided into the following tasks:

1. Calculate and display the monthly payment.
2. Calculate and display a complete amortization schedule.
3. Calculate and display the interest paid during a specified one-year period.
4. Calculate and display a table showing the effect of different interest rates on the monthly payment.
5. Quit.

Let's consider these tasks one at a time.

1. **Calculate and display the monthly payment.**
   - 1.1 Input the principal, interest rate, and duration of the loan.
   - 1.2 Apply the function Payment and display its value.
2. **Display a complete amortization schedule.**
   - 2.1 Input the principal, interest rate, and duration of the loan.
   - 2.2 Assign values to each element of the array of elements with type Month.
     - 2.2.1 Determine the monthly payment.
     - 2.2.2 For each month, determine the apportionment of the payment into interest payment and amount of principal repaid, and then determine the balance at the end of the month. The balance at the beginning of each month is the same as the balance at the end of the previous month, except for the first month where the beginning balance is the principal. These values are calculated with the three formulas presented earlier, with the exception of the last month. Due to rounding, the last monthly payment will be slightly less than the previous payments. The interest for the last month is calculated the same way as for the previous months. However, the amount of principal repaid will equal the balance at the beginning of the month. That way, the balance at the end of the last month will be 0.
   - 2.3 Declare a LINQ query to hold the values in the array of Month elements.
   - 2.4 Use the query to fill the DataGridView control.
   - 2.5 Specify headers for the table.
3. **Calculate and display the interest paid during a specified one-year period.**
   - 3.1 Input the principal, interest rate, and duration of the loan.
   - 3.2 Assign values to each element of the array of elements with type Month. (See the details in 2.2 above.)
   - 3.3 Request the number of the beginning month.
   - 3.4 Declare a LINQ query to limit consideration to the interest payments for the twelve months beginning with the requested month.
   - 3.5 Use the query's Sum method to compute the total of the interest payments for the year.
4. **Calculate and display a table showing the effect of different interest rates on the monthly payment.** First, the interest rate is reduced by one percentage point and the new monthly payment is computed. Then the interest rate is increased by regular increments until it reaches one percentage point above the original rate, with new monthly payment amounts computed for each intermediate interest rate. The subtasks for this task are then as follows:
   - 4.1 Input the principal, interest rate, and duration of the loan.

4.2 Assign values to each element of the array of elements with type EffectOfRate.
    4.2.1 Reduce the interest rate from the text box by 1%.
    4.2.2 Calculate the new monthly payment and place the interest rate and payment into an element of the array of type EffectOfRate.
    4.2.3 Increase the interest rate by 1/8 of 1%.
    4.2.4 Repeat until the interest rate is 1% percent above the interest rate in the text box.
4.3 Declare a LINQ query to hold the values in the array of EffectOfRate elements.
4.4 Use the query to fill the DataGridView control.
4.5 Specify headers for the table.

5. Quit. End the program.

The hierarchy chart in Figure 7.36 shows the stepwise refinement for second, third, and fourth tasks.

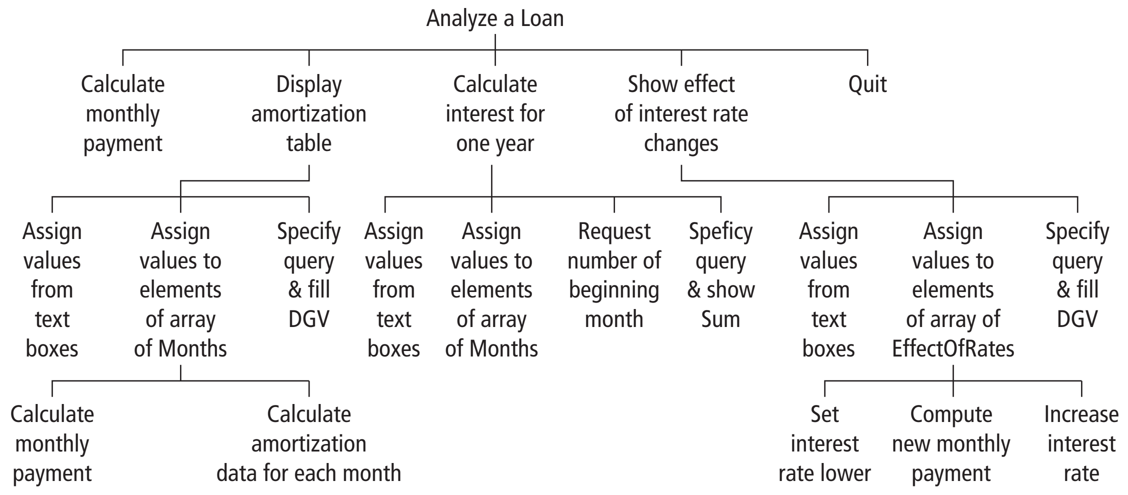

Analyze a Loan
Calculate monthly payment
Display amortization table
Calculate interest for one year
Show effect of interest rate changes
Quit
Assign values from text boxes
Assign values to elements of array of Months
Specify query & fill DGV
Assign values from text boxes
Assign values to elements of array of Months
Request number of beginning month
Speficy query & show Sum
Assign values from text boxes
Assign values to elements of array of EffectOfRates
Specify query & fill DGV
Calculate monthly payment
Calculate amortization data for each month
Set interest rate lower
Compute new monthly payment
Increase interest rate

FIGURE 7.36 **Hierarchy chart for the Analyze-a-Loan program.**

Pseudocode for the Analyze-a-Loan Program

Calculate Monthly Payment button:

```
    INPUT LOAN DATA (Sub procedure InputData)
    COMPUTE MONTHLY PAYMENT (Function Payment)
    DISPLAY MONTHLY PAYMENT
```

Display Amortization Schedule button:

```
    INPUT LOAN DATA (Sub procedure InputData)
    ASSIGN VALUES TO EACH MONTH IN ARRAY OF MONTHS (Sub procedure
    GenerateMonthsArray)
    DEFINE QUERY TO HOLD DATA FROM ARRAY OF MONTHS
    DISPLAY AMORTIZATION SCHEDULE
```

Show Interest Paid in One Year button:

```
    INPUT THE NUMBER OF THE BEGINNING MONTH
    INPUT LOAN DATA (Sub procedure InputData)
```

```
  ASSIGN VALUES TO EACH MONTH IN ARRAY OF MONTHS (Sub procedure GenerateMonthsArray)
  DEFINE QUERY TO HOLD RELEVANT DATA FROM ARRAY OF MONTHS
  EVALUATE SUM OF QUERY AND DISPLAY
Display Interest Rate Change Table button:
  INPUT LOAN DATA (Sub procedure InputData)
  ASSIGN VALUES TO EACH INTEREST RATE IN ARRAY OF EffectOfRates
  DEFINE QUERY TO HOLD RELEVANT DATA FROM ARRAY OF EffectOfRates
  DISPLAY INTEREST RATE CHANGE TABLE
```

The Analyze-a-Loan Program

```
Structure Month
  Dim number As Integer
  Dim interestPaid As Double
  Dim principalPaid As Double
  Dim endBalance As Double
End Structure

Structure EffectOfRate
  Dim interestRate As Double
  Dim monthlyPayment As Double
End Structure

Private Sub btnPayment_Click(...) Handles btnPayment.Click
  Dim principal As Double  'Amount of the loan
  Dim yearlyRate As Double 'Annual rate of interest
  Dim numMonths As Integer 'Number of months to repay loan
  InputData(principal, yearlyRate, numMonths)
  Dim monthlyRate As Double = yearlyRate / 12
  Dim monthlyPayment As Double
  'Calculate monthly payment
  monthlyPayment = Payment(principal, monthlyRate, numMonths)
  'Display results
  MessageBox.Show("Monthly payment: " & FormatCurrency(monthlyPayment, 2),
                  "Payment")
End Sub

Private Sub btnAmort_Click(...) Handles btnAmort.Click
  Dim principal As Double   'Amount of the loan
  Dim yearlyRate As Double  'Annual rate of interest
  Dim numMonths As Integer  'Number of months to repay loan
  InputData(principal, yearlyRate, numMonths)
  Dim months(numMonths – 1) As Month
  Dim monthlyRate As Double = yearlyRate / 12
  Dim monthlyPayment As Double = Payment(principal, monthlyRate, numMonths)
  months = GenerateMonthsArray(principal, monthlyRate, numMonths)
  Dim query = From mnth In months
              Let num = mnth.number
              Let interest = FormatCurrency(mnth.interestPaid)
              Let prin = FormatCurrency(mnth.principalPaid)
              Let bal = FormatCurrency(mnth.endBalance)
              Select num, interest, prin, bal
  dgvResults.DataSource = query.ToList
  dgvResults.CurrentCell = Nothing
  dgvResults.Columns("num").HeaderText = "Month"
  dgvResults.Columns("interest").HeaderText = "Amount Paid for Interest"
  dgvResults.Columns("prin").HeaderText = "Amount Paid for Principal"
  dgvResults.Columns("bal").HeaderText = "Balance at End of Month"
End Sub

Private Sub btnShow_Click(...) Handles btnShow.Click
  Dim principal As Double  'Amount of loan
  Dim yearlyRate As Double 'Annual rate of interest
  Dim numMonths As Integer 'Number of months to repay loan
  InputData(principal, yearlyRate, numMonths)
  Dim months(numMonths - 1) As Month
  Dim monthlyRate As Double = yearlyRate / 12
  Dim monthlyPayment As Double = Payment(principal, monthlyRate, numMonths)
  months = GenerateMonthsArray(principal, monthlyRate, numMonths)
  Dim beginningMonth As Integer = CInt(InputBox("Enter beginning month: "))
  Dim query = From month In months
              Where (month.number >= beginningMonth) And
                    (month.number < beginningMonth + 12)
              Select month.interestPaid
  MessageBox.Show("Interest paid for year beginning with month " &
                  beginningMonth & ": " & FormatCurrency(query.Sum),
                  "Interest For Year")
End Sub

Private Sub btnRateTable_Click(...) Handles btnRateTable.Click
  Dim principal As Double  'Amount of loan
  Dim yearlyRate As Double 'Annual rate of interest
  Dim numMonths As Integer 'Number of months to repay loan
  InputData(principal, yearlyRate, numMonths)
  Dim rates(16) As EffectOfRate
  Dim monthlyRate As Double = yearlyRate / 12
  'Dim monthlyPayment As Double = Payment(principal, monthlyRate, numMonths)
  'Fill rates array
  For i As Integer = 0 To 16
    rates(i).interestRate = (yearlyRate - 0.01) + i * 0.00125
    rates(i).monthlyPayment = Payment(principal,
                              rates(i).interestRate / 12, numMonths)
  Next
  Dim query = From rate In rates
              Let annualRate = FormatPercent(rate.interestRate, 3)
              Let monthlyPayment = FormatCurrency(rate.monthlyPayment)
              Select annualRate, monthlyPayment
  dgvResults.DataSource = query.ToList
  dgvResults.CurrentCell = Nothing
  dgvResults.Columns("annualRate").HeaderText = "Interest Rate"
  dgvResults.Columns("monthlyPayment").HeaderText = "Monthly Payment"
End Sub

Private Sub btnQuit_Click(...) Handles btnQuit.Click
  Me.Close()
End Sub
```

```
Sub InputData(ByRef principal As Double,
              ByRef yearlyRate As Double, ByRef numMonths As Integer)
  'Input loan amount, yearly rate of interest, and duration
  Dim percentageRate As Double, numYears As Integer
  principal = CDbl(txtPrincipal.Text)
  percentageRate = CDbl(txtYearlyRate.Text)
  yearlyRate = percentageRate / 100    'Convert interest rate to decimal form
  numYears = CInt(txtNumYears.Text)
  numMonths = numYears * 12            'Duration of loan in months
End Sub

Function Payment(ByVal principal As Double, ByVal monthlyRate As Double,
                 ByVal numMonths As Double) As Double
  Dim estimate As Double           'Estimate of monthly payment
  estimate = principal * monthlyRate / (1 - (1 + monthlyRate) ^ (-numMonths))
  'Round the payment up if there are fractions of a cent
  If estimate = Math.Round(estimate, 2) Then
    Return estimate
  Else
    Return Math.Round(estimate + 0.005, 2)
  End If
End Function

Function GenerateMonthsArray(ByVal principal As Double,
                             ByVal monthlyRate As Double,
                             ByVal numMonths As Integer) As Month()
  Dim months(numMonths - 1) As Month
  'Fill the months array
  Dim monthlyPayment As Double = Payment(principal, monthlyRate, numMonths)
  'Assign values for first month
  months(0).number = 1
  months(0).interestPaid = monthlyRate * principal
  months(0).principalPaid = monthlyPayment - months(0).interestPaid
  months(0).endBalance = principal - months(0).principalPaid
  'Assign values for interior months
  For i As Integer = 1 To numMonths - 2
    months(i).number = i + 1
    months(i).interestPaid = monthlyRate * months(i - 1).endBalance
    months(i).principalPaid = monthlyPayment - months(i).interestPaid
    months(i).endBalance = months(i - 1).endBalance - months(i).principalPaid
  Next
  'Assign values for last month
  months(numMonths - 1).number = numMonths
  months(numMonths - 1).interestPaid = monthlyRate *
                                     months(numMonths - 2).endBalance
  months(numMonths - 1).principalPaid = months(numMonths - 2).endBalance
  months(numMonths - 1).endBalance = 0
  Return months
End Function
```

CHAPTER 7 SUMMARY

1. For programming purposes, lists of data are most efficiently processed if stored in an *array*. An array is declared with a *Dim* statement, which also can specify its *size* and initial values. The size of an already declared array can be specified or changed with a *ReDim* or *ReDim*