25. Describe the differences between retrieving data from a text file versus retrieving it from a binary file.

PROGRAMMING EXERCISES

1. Write a C# program that prints the current directory and the name and size of all files that are stored in the directory. Your display should be aesthetically pleasing. Numbers should be number aligned and formatted with a thousand separator. Provide headings over the column listings.

2. Place 10 to 20 integer values in a text file. Write a C# program to retrieve the values from the text file. Display the number of values processed, the average of the values, formatted with two decimal places, and the smallest and largest values. Include appropriate exception-handling techniques with your solution. *Hint:* To simplify the problem, the values can each be placed on separate lines in a Notepad file.

3. Write a program that enables the user to input name, address, and local phone number, including area code. Encourage the user to include dashes between the numbers (i.e. xxx-xxx-xxxx) when they type in the value. Once retrieved, store the values in a text file. For the phone, surround the phone numbers with asterisks in the file and store only the numbers for the phone number. Do not store the hyphens or dashes in the file. Include appropriate exception-handling techniques with your solution. After storing the name, address, and phone number, display a message indicating the data was stored properly. Use Notepad to view the contents.

4. Write a program that stores 50 random numbers in a file. The random numbers should be positive with the largest value being 1000. Store five numbers per line and 10 different lines. Use the `Random` **`class`** to generate the values. Include appropriate exception-handling techniques in your solution. When the application closes, locate the text file and verify its contents.

5. Write a program that retrieves numbers stored in a text file. Test your solution by retrieving data from a file that contains 10 different rows of data with five values per line. For your test, display the largest and smallest values from each row of data. Include appropriate exception-handling techniques in your solution. *Hint:* If you completed Programming Exercise #4, use the text file created by that exercise.

6. Write a program that displays a graphical user interface (Windows form) that allows multiple names, e-mail addresses, and local phone numbers to be entered. Allow only numbers to be entered for the phone number. Retrieve and store the values entered by the user in a text file and then ready the GUI for the next set of input values. Store each person's data on separate lines. Include appropriate exception-handling techniques in your solution. When the application closes, locate the text file and verify its contents.

7. Write an application that retrieves both string data and numbers from a text file. Test your solution by retrieving names of students and three scores per line from a text file. Process the values by calculating the average of the scores per student. Write the name and average to a different text file. Display on the console screen what is being written to the new file. Test your application with a minimum of eight records in the original file. *Hint:* You might consider adding delimiters between the data values in the original text file to simplify retrieving and processing the data. Include appropriate exception-handling techniques in your solution. When the application closes, locate the text file and verify its contents.

8. Write a program that produces a report showing the number of students who can still enroll in given classes. Test your solution by retrieving the data from a text file that you create using a text editor, such as Notepad. Some sample data follows. Include the name of the class, current enrollment, and maximum enrollment.

Class name	Current enrollment	Maximum enrollment
CS150	18	20
CS250	11	20
CS270	23	25
CS300	4	20
CS350	32	20

Classes should not be oversubscribed. Define a custom exception `class` for this problem so that an exception is thrown if the current enrollment exceeds the maximum enrollment by more than three students. When this unexpected condition occurs, halt the program and display a message indicating which course is overenrolled.

9. Write a graphical user application that accepts employee data to include employee name, number, pay rate, and number of hours worked. Pay is

to be computed as follows: Hours over 40 receive time-and-a-half pay. Store the employee name, number, and the total amount of pay (prior to deductions) in a text file. Close the file and then, in the same application, retrieve the stored values and display the employee name and the formatted total pay. Your application should allow the user to browse to the file location for saving and retrieving the file.

10. Allow the user to enter multiple sets of five numbers. Store the numbers in a binary file. For each set of values, calculate and store the average of the numbers prior to retrieving the next set of values. For example, if the user entered 27 78 120 111 67 as the first set of values, the first values written to the binary file would be 27 78 120 111 67 80.6. For an extra challenge, close the file, reopen it, and display the values from the file in a listbox control.