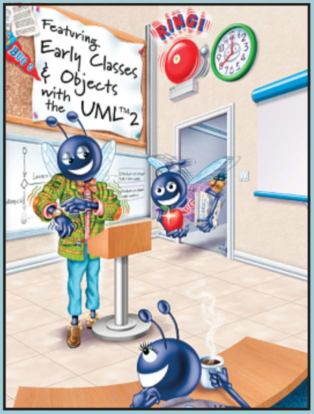

17

# File Processing

**OBJECTIVES**

In this chapter you will learn:

- To create, read, write and update files.
- Sequential file processing.
- Random-access file processing.
- To use high-performance unformatted I/O operations.
- The differences between formatted-data and raw-data file processing.
- To build a transaction-processing program using random-access file processing.

## Assignment Checklist

Name: ____________________ Date: ____________________

Section: ____________________

| Exercises | Assigned: Circle assignments | Date Due |
|---|---|---|
| **Prelab Activities** | | |
| Matching | YES NO | |
| Fill in the Blank | 9, 10, 11, 12, 13, 14, 15, 16 | |
| Short Answer | 17, 18 | |
| Correct the Code | 19, 20, 21 | |
| **Lab Exercise** | | |
| Lab Exercise — Telephone Words | YES NO | |
| **Labs Provided by Instructor** | | |
| 1. | | |
| 2. | | |
| 3. | | |
| **Postlab Activities** | | |
| Coding Exercises | 1, 2 | |
| Programming Challenge | 1 | |

## Prelab Activities

### Matching

**Name:** ______________________ **Date:** ______________________

**Section:** ______________________

After reading Chapter 17 of *C++ How to Program: Fifth Edition*, answer the given questions. These questions are intended to test and reinforce your understanding of key concepts and may be done either before the lab or during the lab.

For each term in the column on the left, write the corresponding letter for the description that best matches it from the column on the right.

| | Term | Description |
|---|---|---|
| ___ | 1. Record | a) Provides rapid, direct access to data in a file. |
| ___ | 2. Sequential-access file | b) The smallest data item that computers support. |
| ___ | 3. Byte | c) Group of 8 bits. |
| ___ | 4. Field | d) File that must be read or written from beginning to end. |
| ___ | 5. Data hierarchy | e) At least one field that is distinct from all other records in the file. |
| ___ | 6. Random-access file | f) Data items ranging from bits to databases. |
| ___ | 7. Record key | g) Group of related character fields. |
| ___ | 8. Bit | h) Group of characters that conveys meaning. |

## Prelab Activities

Name:

### Fill in the Blank

**Name:** ____________________________ **Date:** ____________________________

**Section:** ____________________________

Fill in the blank for each of the following statements:

9. A group of related files is stored in a(n) _____________.

10. A collection of programs designed to create and manage databases is called a(n) _____________.

11. The header `<fstream>` provides the definitions for stream-class templates _____________, _____________ and _____________.

12. _____________ indicates the position in the file from which the next input is to occur.

13. Repositioning the read location in a file requires a call to function _____________.

14. Repositioning the write location in a file requires a call to function _____________.

15. Unary operator _____________ returns a type's size in bytes.

16. Using _____________ records is a convenient way to implement random-access files.

## Prelab Activities

Name:

### Short Answer

**Name:** ____________________________ **Date:** ____________________________

**Section:** ____________________________

In the space provided, answer each of the given questions. Your answers should be as concise as possible; aim for two or three sentences.

17. List the computer data hierarchy from bit to database.

18. Why are random-access files preferable to sequential-access files in performance-oriented situations?

## Prelab Activities

Name:

### Correct the Code

**Name:** ______________________ **Date:** ______________________

**Section:** ______________________

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic, syntax or compilation error, circle the error in the program, and write the corrected code in the space provided after each problem. If the code does not contain an error, write “no error.” [*Note:* It is possible that a program segment may contain multiple errors.]

19. The following code attempts to open `temp.dat` for writing, assume that all the necessary header files and `using` statements have been provided.

```
Stream outfile( "temp.dat", ios::out );

if ( outfile )
{
   cerr << "operation failed";
   exit( 1 );
} // end if
```

*Your answer:*

20. The following code should write 100 empty `ClientData` objects to the `ofstream` object `outCredit`, which has already been successfully opened.

```
ClientData blankClient; // constructor zeros out each data member

// output 100 blank records to file
for ( int i = 0; i < 100; i++ )
   outCredit.write( ( blankClient ), sizeof( ClientData ) );
```

*Your answer:*

## Prelab Activities

Name:

### Correct the Code

21. The following line of code should create `fstream` object `outCredit` attached to file `"credit.dat"` for input and output of fixed-length records.

```
fstream outCredit( "credit.dat", ios::in & ios::out & ios::binary );
```

*Your answer:*

# Lab Exercise

## Lab Exercise — Telephone Words

**Name:** ____________________ **Date:** ____________________

**Section:** ____________________

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into five parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 17.1)
5. Problem-Solving Tips

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. The source code for the template is available at `www.deitel.com` and `www.prenhall.com./deitel`.

### Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 17 of *C++ How To Program: Fifth Edition*. In this lab, you will practice:

- Opening output files.
- Writing to a sequential-access file.

### Description of the Problem

Standard telephone keypads contain the digits 0 through 9. The numbers 2 through 9 each have three letters associated with them, as is indicated by the following table:

| Digit | Letter |
|---|---|
| 2 | A B C |
| 3 | D E F |
| 4 | G H I |
| 5 | J K L |
| 6 | M N O |
| 7 | P R S |
| 8 | T U V |
| 9 | W X Y |

Many people find it difficult to memorize phone numbers, so they use the correspondence between digits and letters to develop seven-letter words that correspond to their phone numbers. For example, a person whose telephone number is 686-2377 might use the correspondence indicated in the above table to develop the seven-letter word “NUMBERS.”

Businesses frequently attempt to get telephone numbers that are easy for their clients to remember. If a business can advertise a simple word for its customers to dial, then no doubt the business will receive a few more calls.

## Lab Exercise

Name:

## Lab Exercise — Telephone Words

Each seven-letter word corresponds to exactly one seven-digit telephone number. The restaurant wishing to increase its take-home business could surely do so with the number 825-3688 (i.e., “TAKEOUT”).

Each seven-digit phone number corresponds to many separate seven-letter words. Unfortunately, most of these represent unrecognizable juxtapositions of letters. It is possible, however, that the owner of a barber shop would be pleased to know that the shop’s telephone number, 424-7288, corresponds to “HAIRCUT.” The owner of a liquor store would, no doubt, be delighted to find that the store’s telephone number, 233-7226, corresponds to “BEERCAN.” A veterinarian with the phone number 738-2273 would be pleased to know that the number corresponds to the letters “PETCARE.”

Write a C++ program that, given a seven-digit number, writes to a file every possible seven-letter word corresponding to that number. There are 2187 (3 to the seventh power) such words. Avoid phone numbers with the digits 0 and 1.

### Sample Output

```
Enter a phone number (digits 2 through 9) in the form: xxx-xxxx
568-9876
```

Contents of `phone .dat`

```
JMTWTPM JMTWTPN JMTWTPO JMTWTRM JMTWTRN JMTWTRO JMTWTSM JMTWTSN JMTWTSO
JMTWUPM JMTWUPN JMTWUPO JMTWURM JMTWURN JMTWURO JMTWUSM JMTWUSN JMTWUSO
JMTWVPM JMTWVPN JMTWVPO JMTWVRM JMTWVRN JMTWVRO JMTWVSM JMTWVSN JMTWVSO
JMTXTPM JMTXTPN JMTXTPO JMTXTRM JMTXTRN JMTXTRO JMTXTSM JMTXTSN JMTXTSO
JMTXUPM JMTXUPN JMTXUPO JMTXURM JMTXURN JMTXURO JMTXUSM JMTXUSN JMTXUSO
JMTXVPM JMTXVPN JMTXVPO JMTXVRM JMTXVRN JMTXVRO JMTXVSM JMTXVSN JMTXVSO
...

LOVXVPM LOVXVPN LOVXVPO LOVXVRM LOVXVRN LOVXVRO LOVXVSM LOVXVSN LOVXVSO
LOVYTPM LOVYTPN LOVYTPO LOVYTRM LOVYTRN LOVYTRO LOVYTSM LOVYTSN LOVYTSO
LOVYUPM LOVYUPN LOVYUPO LOVYURM LOVYURN LOVYURO LOVYUSM LOVYUSN LOVYUSO
LOVYVPM LOVYVPN LOVYVPO LOVYVRM LOVYVRN LOVYVRO LOVYVSM LOVYVSN LOVYVSO

 Phone number is 568-9876
```

### Template

```cpp
// Lab 1: telephoneWords.cpp
#include <iostream>
using std::cerr;
using std::cin;
using std::cout;

#include <fstream>
using std::ofstream;

#include <cstdlib>
using std::exit;

void wordGenerator( const int * const );

int main()
{
```

**Fig. L 17.1** | Contents of `telephoneWords.cpp`. (Part 1 of 3.)

## Lab Exercise

Name:

## Lab Exercise — Telephone Words

```
   int phoneNumber[ 7 ] = { 0 }; // holds phone number

   // prompt user to enter phone number
   cout << "Enter a phone number (digits 2 through 9) "
      << "in the form: xxx-xxxx\n";

   // loop 8 times: 7 digits plus hyphen;
   // hyphen is not placed in phoneNumber
   for ( int u = 0, v = 0; u < 8; u++ )
   {
      int i = cin.get();

      // test if i is between 0 and 9
      if ( i >= '0' && i <= '9' )
         phoneNumber[ v++ ] = i - '0';
   } // end for

   wordGenerator( phoneNumber ); // form words from phone number
   return 0;
} // end main

// function to form words based on phone number
void wordGenerator( const int * const n )
{
   // set output stream and open output file
   /* Write a declaration for an ofstream object called
      outFile to open the file "phone.dat" */

   // letters corresponding to each number
   /* Write a declaration for an array of 10 const char *'s
      called phoneLetters. Use an initializer list to assign
      each element of the array the corresponding string of
      three letters. Use dummy characters for 0 and 1 */

   // terminate if file could not be opened
   /* Write code to check if the file was opened successfully,
      and terminate if not */

   int count = 0; // number of words found

   // output all possible combinations
   for ( int i1 = 0; i1 <= 2; i1++ )
   {
      for ( int i2 = 0; i2 <= 2; i2++ )
      {
         for ( int i3 = 0; i3 <= 2; i3++ )
         {
            for ( int i4 = 0; i4 <= 2; i4++ )
            {
               for ( int i5 = 0; i5 <= 2; i5++ )
               {
                  for ( int i6 = 0; i6 <= 2; i6++ )
                  {
                     for ( int i7 = 0; i7 <= 2; i7++ )
                     {
```

**Fig. L 17.1** | Contents of `telephoneWords.cpp`. (Part 2 of 3.)

## Lab Exercise

Name:

### Lab Exercise — Telephone Words

```
                              /* Write a series of cascaded stream insertion
                                 operations to output a set of seven letters
                                 to outFile, followed by a space */

                              if ( ++count % 9 == 0 ) // form rows
                                 outFile << '\n';
                           } // end for
                        } // end for
                     } // end for
                  } // end for
               } // end for
            } // end for
         } // end for

         // output phone number
         outFile << "\nPhone number is ";

         for ( int i = 0; i < 7; i++ )
         {
            if ( i == 3 )
               outFile << '-';

            outFile << n[ i ];
         } // end for

         /* Write a statement to close the ouput file */
      } // end function wordGenerator
```

**Fig. L 17.1** | Contents of `telephoneWords.cpp`. (Part 3 of 3.)

### Problem-Solving Tips

1. To determine every possible seven-letter word, you will need to use seven nested `for` loops and an array of `char *`, which can be thought of as a two-dimensional array of `char`s, containing the three letters that correspond to each digit.
2. To write a particular seven-letter word, output one character at a time. Use the array of `char *`'s, the first subscript of which will be the digit from the phone number and the second subscript of which will be the counter variable from the `for` loop that corresponds to this digit (i.e., the first letter will use the counter variable from the first, outermost `for` loop).

## Postlab Activities

### Coding Exercises

**Name:** ____________________________ **Date:** ____________________________

**Section:** ____________________________

These coding exercises reinforce the lessons learned in the lab and provide additional programming experience outside the classroom and laboratory environment. They serve as a review after you have completed the *Prelab Activities* and *Lab Exercises* successfully.

For each of the following problems, write a program or a program segment that performs the specified action.

1. Create a simple sequential-access file-processing program that might be used by professors to help manage their student records. For each student, the program should obtain an ID number, the student's first name, the student's last name and the student's grade. The data obtained for each student constitutes a record for the student and should be stored in an object of a class called `Student`. The program should save the records in a sequential file specified by the user.

2. Create a simple sequential-access file-processing program to complement the program in *Coding Exercise 1*. This program should open the file created by the *Coding Exercise 1* program and read and display the grade information for each student. The program should also display the class average.

## Postlab Activities

Name:

### Programming Challenge

**Name:** ____________________________ **Date:** ____________________________

**Section:** ____________________________

The *Programming Challenges* are more involved than the *Coding Exercises* and may require a significant amount of time to complete. Write a C++ program for each of the problems in this section. The answers to these problems are available at `www.deitel.com` and `www.prenhall.com/deitel`. Pseudocode, hints and/or sample outputs are provided to aid you in your programming.

1. Suppose we wish to process survey results that are stored in a file. This exercise requires two separate programs. First, create a program that prompts the user for survey responses and outputs each response to a file. Use an `ofstream` to create a file called `"numbers.txt"`. Then create a program to read the survey responses from `"numbers.txt"`. The responses should be read from the file by using an `ifstream`. Input one integer at a time from the file. The program should continue to read responses until it reaches the end of file. The results should be output to the text file `"output.txt"`.

**Hint:**

- The second program will use both `ifstream` and `ofstream` objects, the first for reading responses from `numbers.txt` and the second for writing frequency counts to `output.txt`.