Programming Logic and Design
Ninth Edition

Chapter 7

File Handling and Applications

Objectives

In this chapter, you will learn about:

- Computer files
- The data hierarchy
- Performing file operations
- Control break logic
- Merging files
- Master and transaction file processing
- Random access files

Understanding Computer Files

- **Computer file**
  - A collection of data stored on **permanent storage devices** such as your computer's hard drive, a hard drive on the cloud, DVDs, USB drives, and reels of magnetic tape
  - **Text files** (numbers, names, salaries) that can be read by a text editor
  - **Binary files** (images and music) not encoded as text

Understanding Computer Files (continued -1)

- Computer files have:
  - A **filename** **-** an identifying name given to a computer file that frequently describes the contents
    - JanuaryPayroll
    - PreviousMonthSales
  - A **filename extension** - a group of characters added to the end of a filename that describes the type of the file
    - .txt
    - .dat
    - .docx

Understanding Computer Files (continued -2)

- Computer files have:
  - A specific creation time and modification date
  - A file size measured in bytes
    - **byte** – one character
    - **kilobyte** – thousands of bytes
    - **megabyte** – millions of bytes
    - **gigabyte** – billions of bytes
    - terabyte – trillions of bytes

Understanding Computer Files (continued -3)

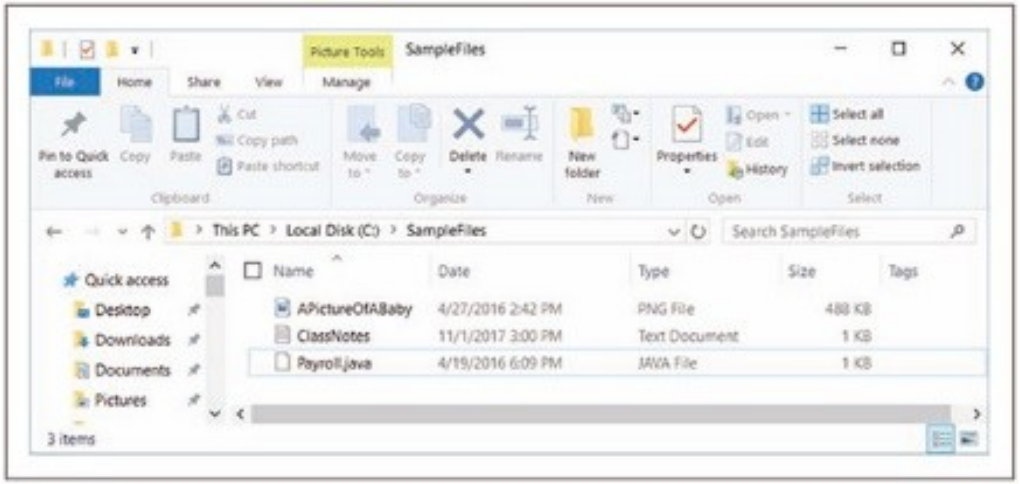


Figure 7-1 Three stored files showing their names, dates of modification, types, and sizes

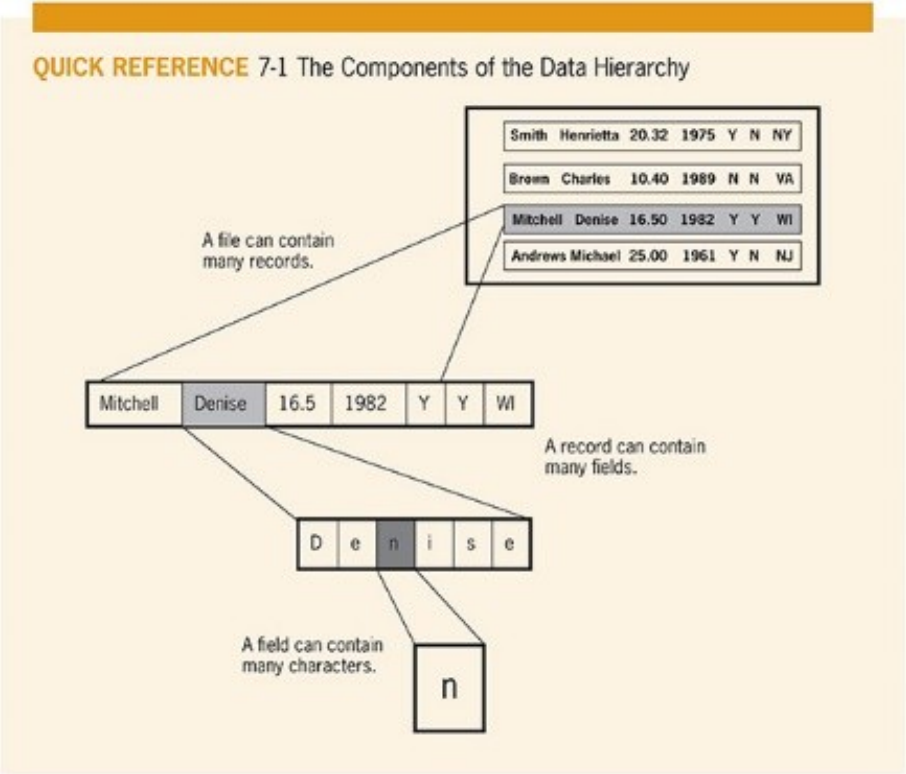
Understanding Computer Files (continued -4)

- **Organizing files**
  - **Directories and folders**
    - Organization units on storage devices
  - **Path**
    - Combination of disk drive plus the complete hierarchy of directories
    - Example: `C:\Logic\SampleFiles\PayrollData.dat`

Understanding the Data Hierarchy

- **Data hierarchy**
  - Describes the relationships between data components
  - Consists of:
    - **Characters** – letters number and special symbols
    - **Fields** – data items representing a single attribute of a record
    - **Records** – groups of fields that go together for some logical reason
    - **Files** – groups of related records
    - **Database** – holds related file data in tables

Understanding the Data Hierarchy (continued)



Performing File Operations

- File operations to use data files in your programs
  - Declare a file identifier

```
InputFile employeeData
OutputFile updatedData
```

  - **Open the file**

```
open employeeData "EmployeeData.dat"
```

  - **Reading from a file** and processing the data

```
input name from employeeData
input address from employeeData
input payRate from employeeData
```

Performing File Operations

(continued -1)

- **Reading from a file** and processing the data
  - Programming languages have different ways of determining how much data to input
  - In many languages, a **delimiter** such as a comma, semicolon, or tab character is stored between data fields

Performing File Operations

(continued -2)

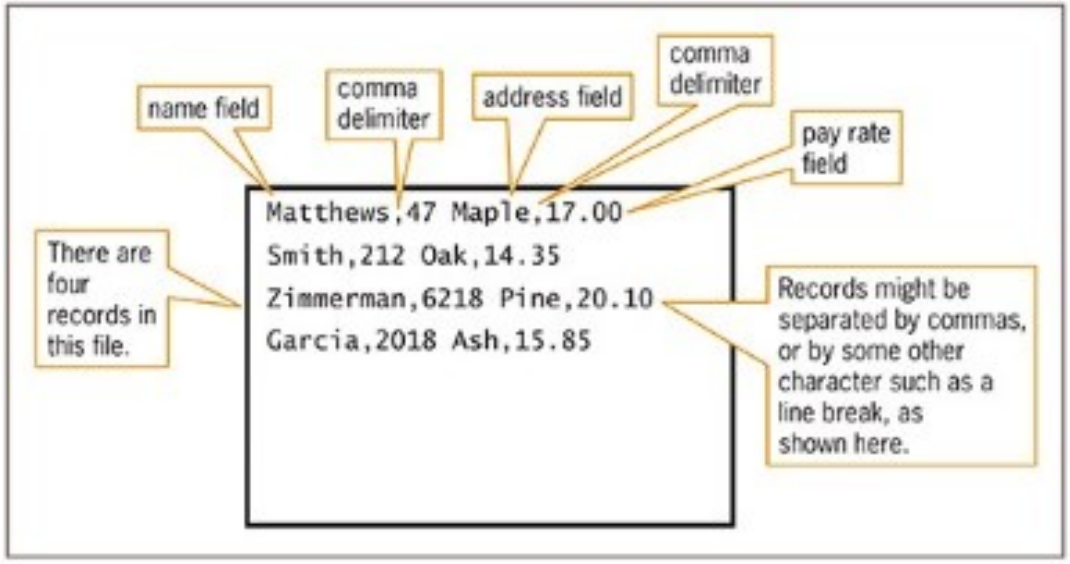


Figure 7-2 How employee data in a readable comma-delimited file might appear in a text reader

Performing File Operations

(continued -3)

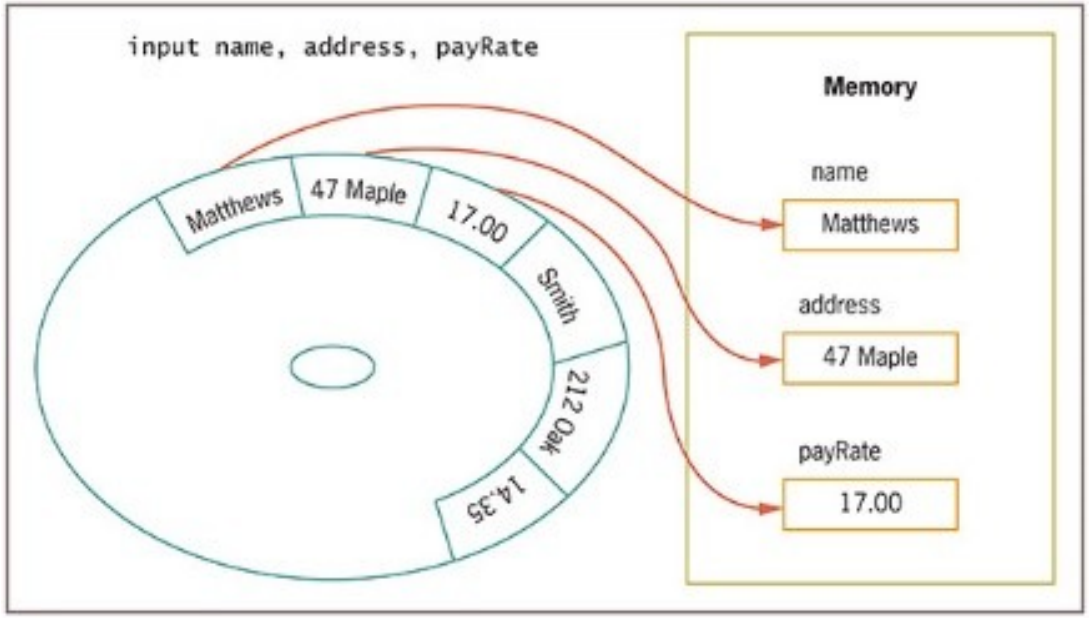


Figure 7-3 Reading three data items from a storage device into memory

Performing File Operations

(continued -4)

- **Sequential file**
  - Program reads all the records in this file from beginning to end, processing them one t a time
- **Sorting**
  - The process of placing records in order by the value in a specific field or fields
    - **Ascending order** – records sorted in order from lowest to highest values
    - **Descending order** – records sorted in order from highest to lowest values

Performing File Operations

(continued -5)

- Writing data to a file
  - When you store data in a computer file on a persistent storage device, you **write to the file**
    - `output name, address, payRate to employeeData`
- **Closing a file**
  - When you finish using a file, the program should **close the file**
  - Always close every file you open
- **Default input and output devices** (keyboard and monitor) do not require opening or closing

A Program that Performs File Operations

- **Backup file**
  - a copy kept in case values need to be restored to their original state
  - The backup copy is called a **parent file**
  - The newly revised copy is a **child file**

A Program that Performs File Operations

(continued -1)

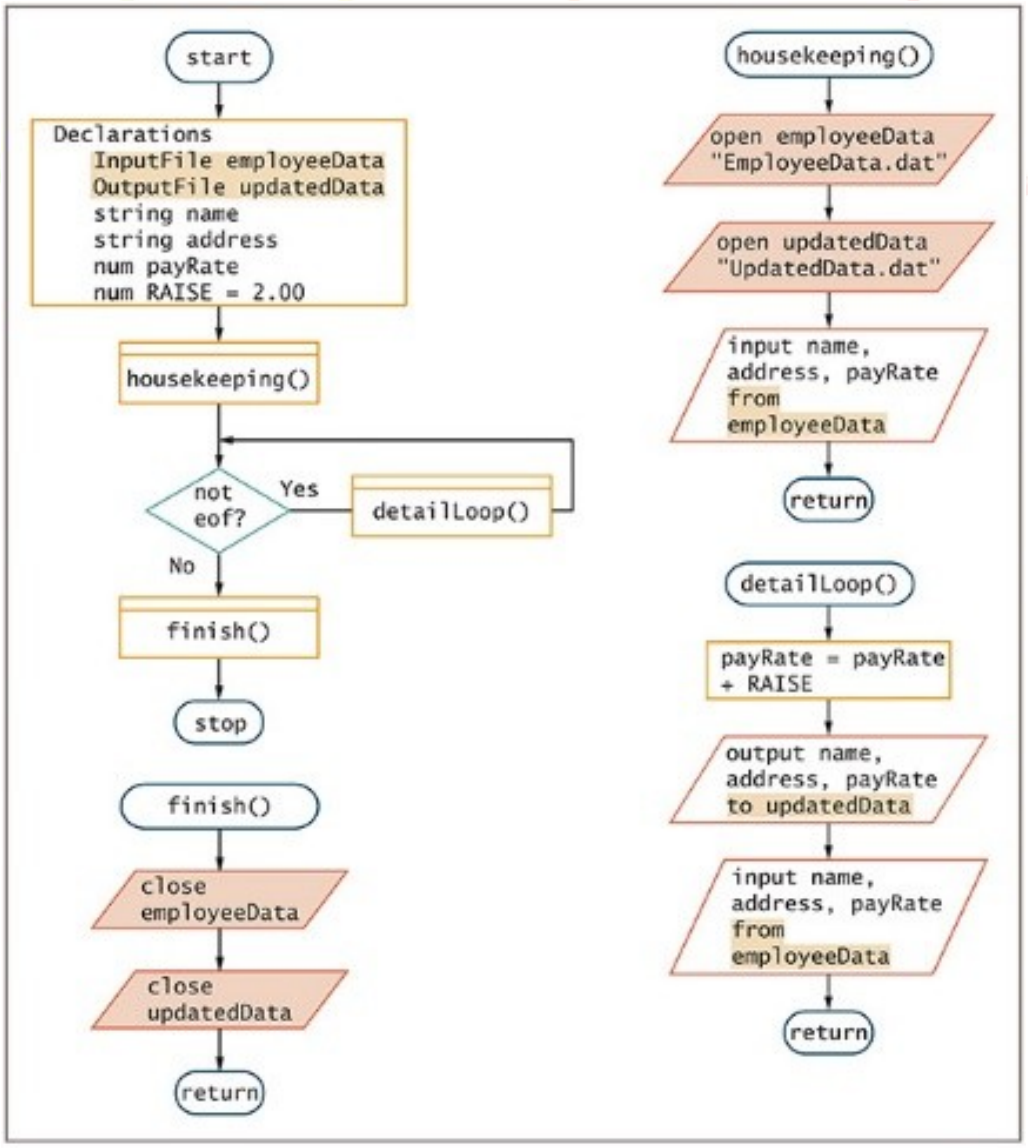


Figure 7-4 Flowchart and pseudocode for program that uses files *(continues)*

A Program that Performs File Operations

(continued -2)

(continued)

```
start
   Declarations
      InputFile employeeData
      OutputFile updatedData
      string name
      string address
      num payRate
      num RAISE = 2.00
   housekeeping()
   while not eof
      detailLoop()
   endwhile
   finish()
stop

housekeeping()
   open employeeData "EmployeeData.dat"
   open updatedData "UpdatedData.dat"
   input name, address, payRate from employeeData
return

detailLoop()
   payRate = payRate + RAISE
   output name, address, payRate to updatedData
   input name, address, payRate from employeeData
return

finish()
    close employeeData
    close updatedData
return
```

Figure 7-4 Flowchart and pseudocode for program that uses files

Understanding Control Break Logic

- A **control break** is a temporary detour in the logic of a program
  - A **control break program** uses a change in a value to initiate special actions or processing
  - A **control break report** groups similar data together
    - Input records must be in sequential order

Understanding Control Break Logic (continued -1)

Company Clients by State of Residence

Name	City	State	
Albertson	Birmingham	Alabama	
Davis	Birmingham	Alabama	
Lawrence	Montgomery	Alabama	
		Count for Alabama	3
Smith	Anchorage	Alaska	
Young	Anchorage	Alaska	
Davis	Fairbanks	Alaska	
Mitchell	Juneau	Alaska	
Zimmer	Juneau	Alaska	
		Count for Alaska	5
Edwards	Phoenix	Arizona	
		Count for Arizona	1

Figure 7-5 A control break report with totals after each state

Understanding Control Break Logic (continued -2)

- Examples of control break reports
  - All employees listed in order by department number, with a new page started for each department
  - All books for sale in a bookstore listed in order by category (such as reference or self-help), with a count following each category of book
  - All items sold in order by date of sale, with a different ink color for each new month

Understanding Control Break Logic (continued -3)

- **Single-level control break**
  - A detour based on the value of a single variable
  - Uses a **control break field** to hold the previous value

Understanding Control Break Logic

(continued -4)

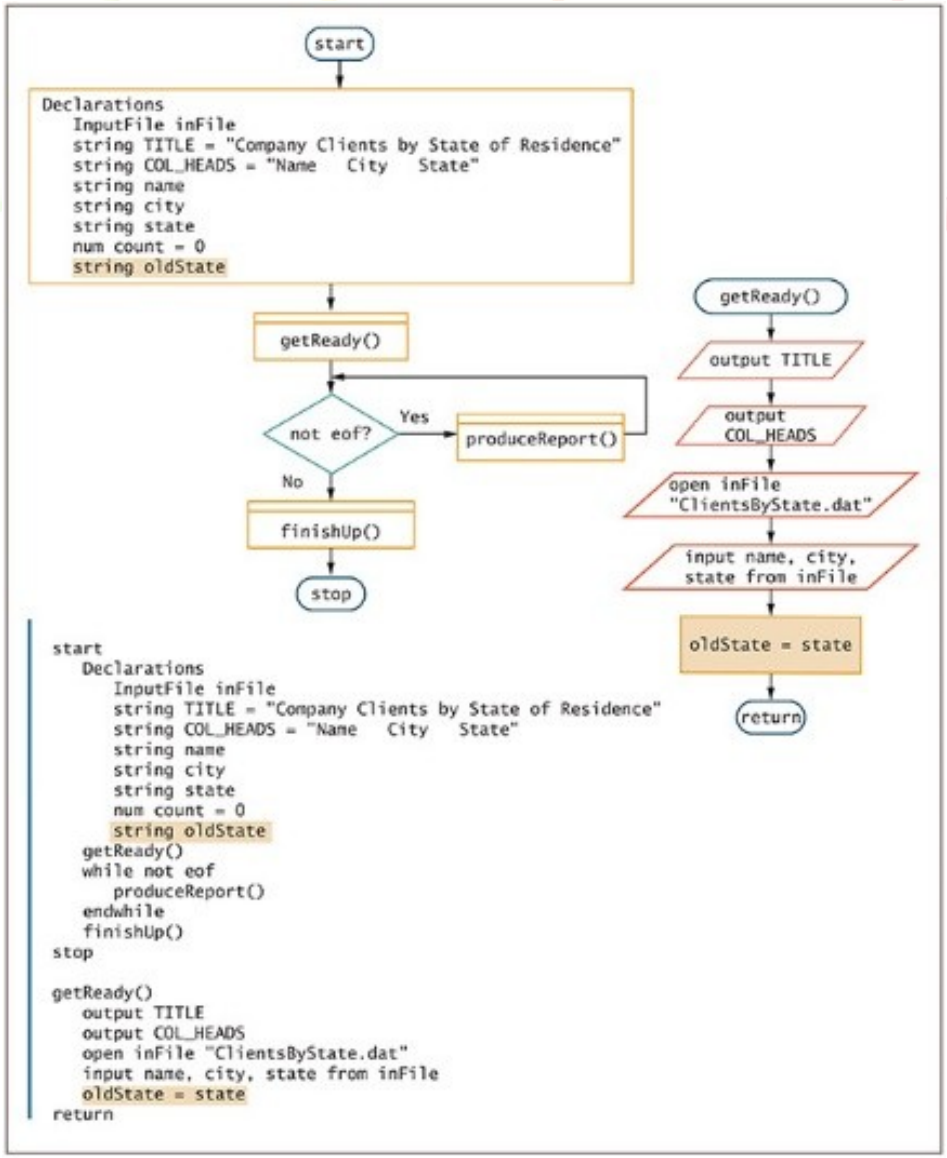


Figure 7-6 Mainline logic and getReady() module for the program that produces clients by state report

Understanding Control Break Logic

(continued -5)

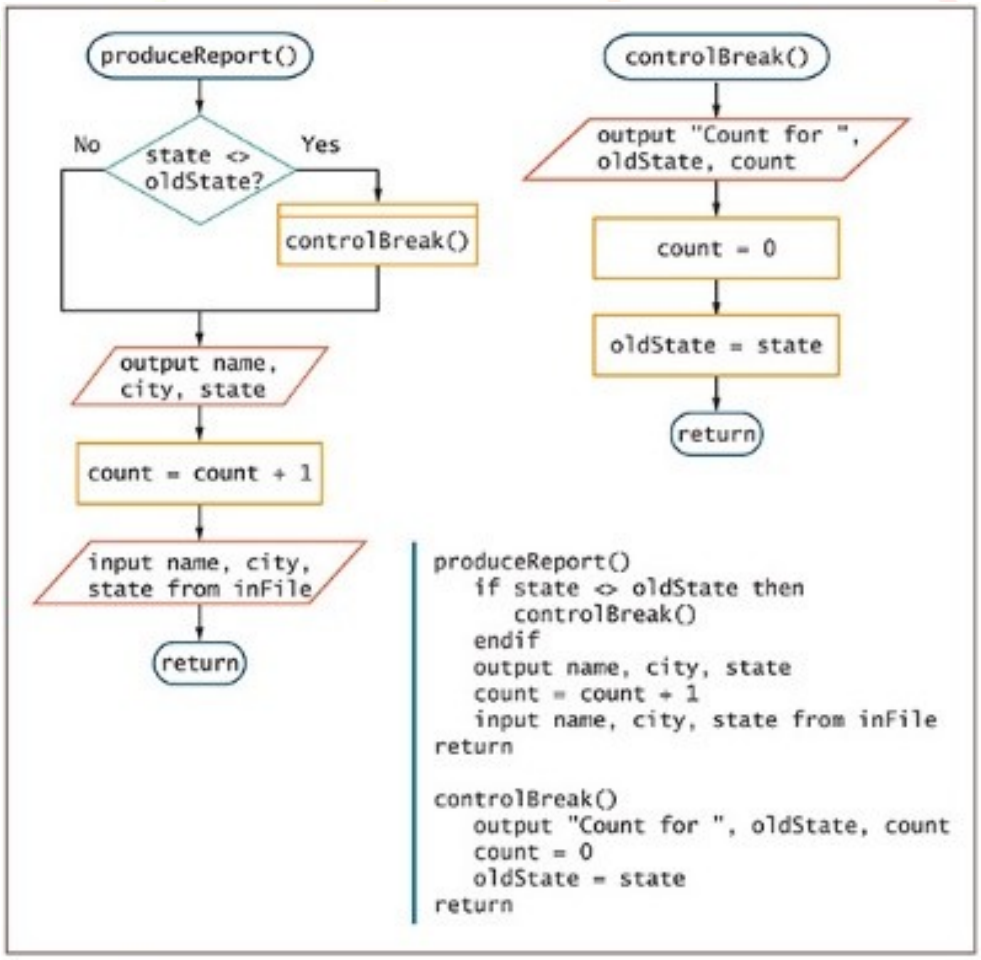


Figure 7-7 The `produceReport()` and `controlBreak()` modules for the program that produces a list of clients by state

Understanding Control Break Logic (continued -6)

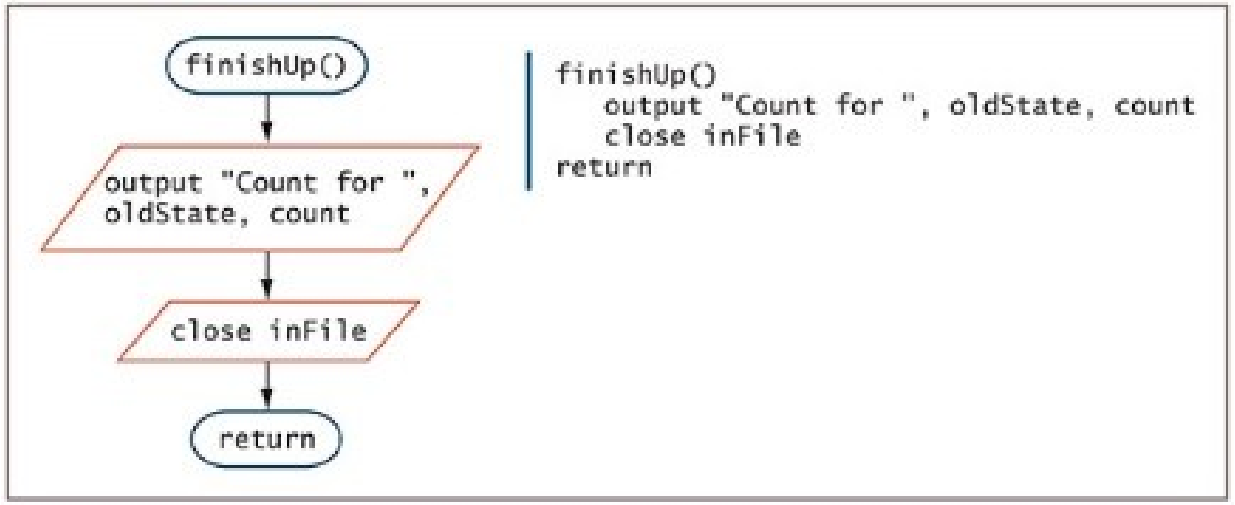


Figure 7-8 The finishUp() module for the program that produces clients by state report

Merging Sequential Files

- **Merging files**
  - Combining two or more files while maintaining the sequential order or the records
- Examples
  - A file of current employees in ID number order, and a file of newly hired employees also in ID number order
  - A file of parts manufactured in the Northside factory in part-number order, and a file of parts manufactured in the Southside factory also in part-number order

Merging Sequential Files (continued - 1)

- Two conditions required for merging files
  - Each file has the same record layout
  - Sorted in the same order based on the same field

Merging Sequential Files (continued - 2)

East Coast File		West Coast File	
eastName	**eastBalance**	**westName**	**westBalance**
Able	100.00	Chen	200.00
Brown	50.00	Edgar	125.00
Dougherty	25.00	Fell	75.00
Hanson	300.00	Grand	100.00
Ingram	400.00		
Johnson	30.00		

Figure 7-9 Sample data contained in two customer files

Merged File	
mergedName	**mergedBalance**
Able	100.00
Brown	50.00
Chen	200.00
Dougherty	25.00
Edgar	125.00
Fell	75.00
Grand	100.00
Hanson	300.00
Ingram	400.00
Johnson	30.00

Figure 7-10 Merged customer file

Merging Sequential Files (continued - 3)

- Mainline logic similar to other file-processing programs, except for handling two files
- With two input files, must determine when both files are at eof
  - Define a flag variable to indicate that both files have reached eof
  - Must define two input files
  - Read one record from each input file

Merging Sequential Files (continued - 4)

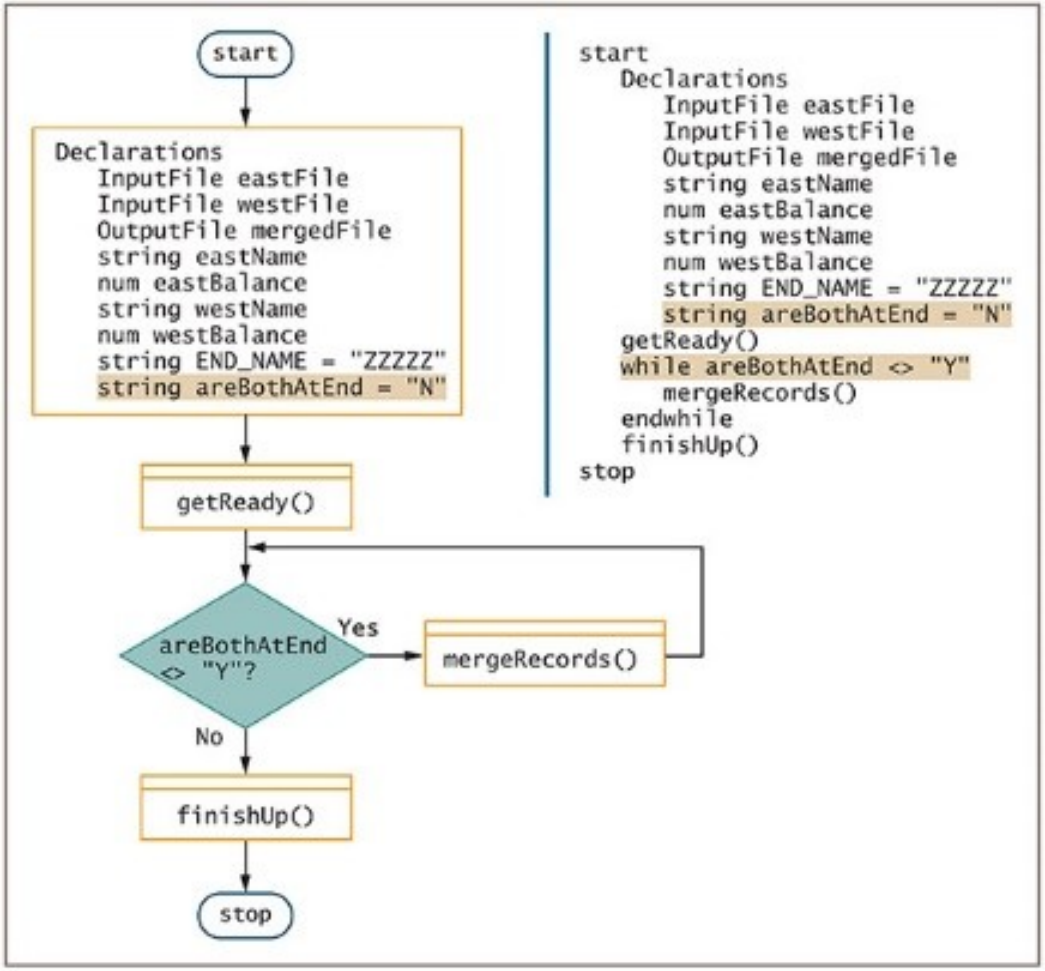


Figure 7-11 Mainline logic of a program that merges files

Merging Sequential Files (continued -5)

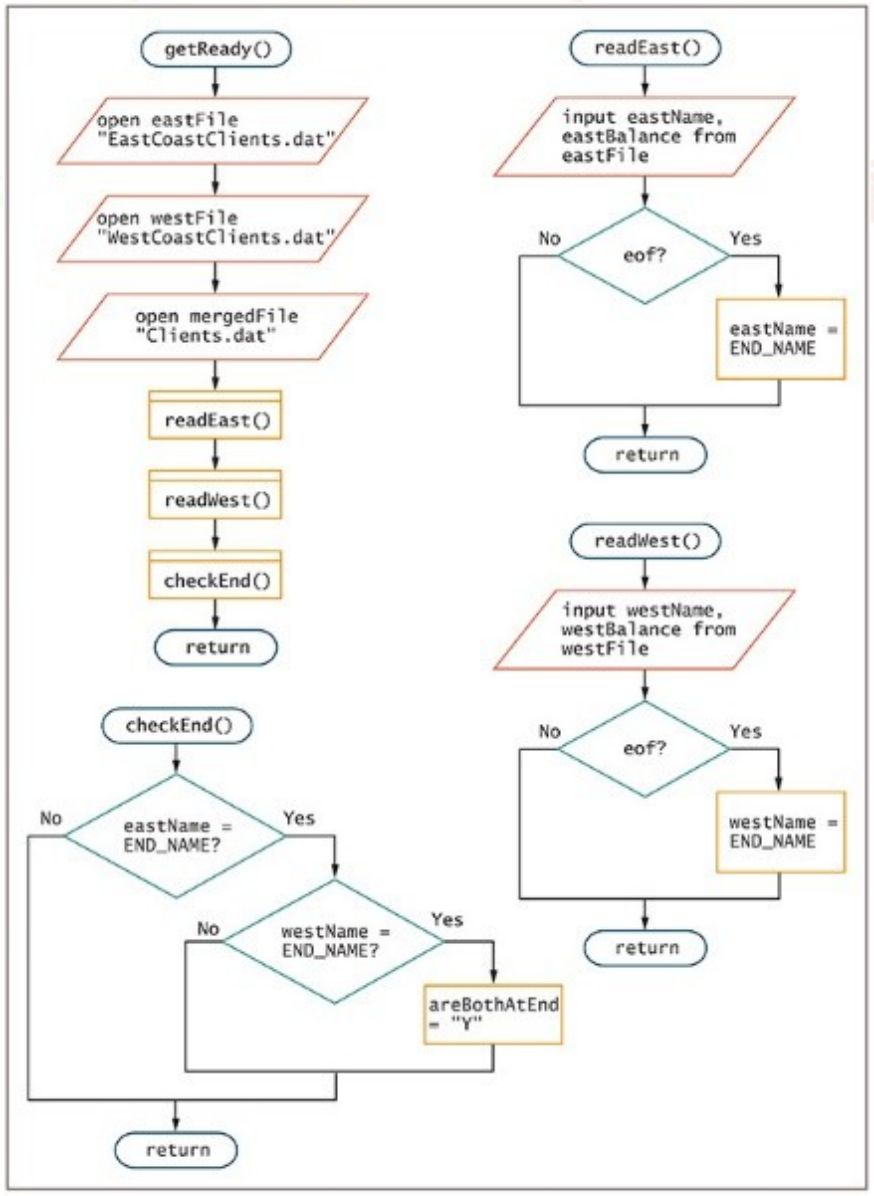


Figure 7-12 The getReady() method for a program that merges files, and the methods it calls (continues)

Merging Sequential Files (continued -6)

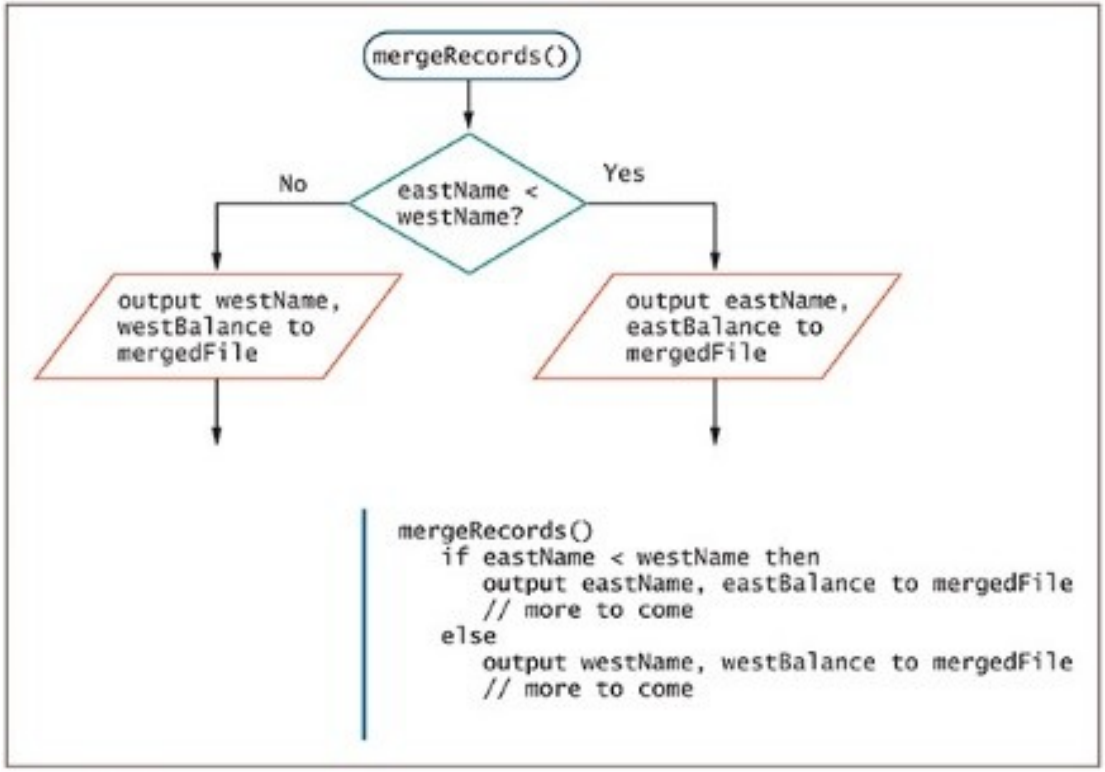
(continued)

```
getReady()
   open eastFile "EastCoastClients.dat"
   open westFile "WestCoastClients.dat"
   open mergedFile "Clients.dat"
   readEast()
   readWest()
   checkEnd()
return

readEast()
   input eastName, eastBalance from eastFile
   if eof then
      eastName = END_NAME
   endif
return

readWest()
   input westName, westBalance from westFile
   if eof then
      westName = END_NAME
   endif
return

checkEnd()
   if eastName = END_NAME then
      if westName = END_NAME then
         areBothAtEnd = "Y"
      endif
   endif
return
```

Figure 7-12 The `getReady()` method for a program that merges files, and the methods it calls

Merging Sequential Files

(continued -7)




```
mergeRecords()
   if eastName < westName then
      output eastName, eastBalance to mergedFile
      // more to come
   else
      output westName, westBalance to mergedFile
      // more to come
```

Figure 7-13 Start of merging process

Merging Sequential Files

(continued -8)

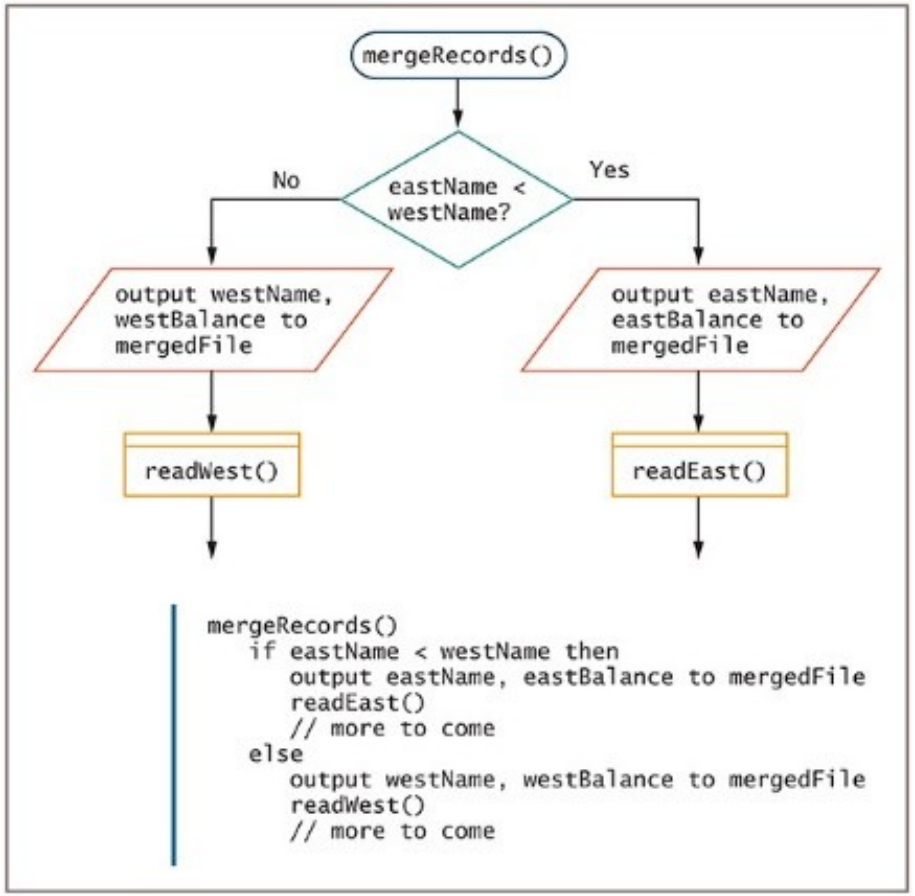


Figure 7-14 Continuation of merging process

Merging Sequential Files

(continued -9)

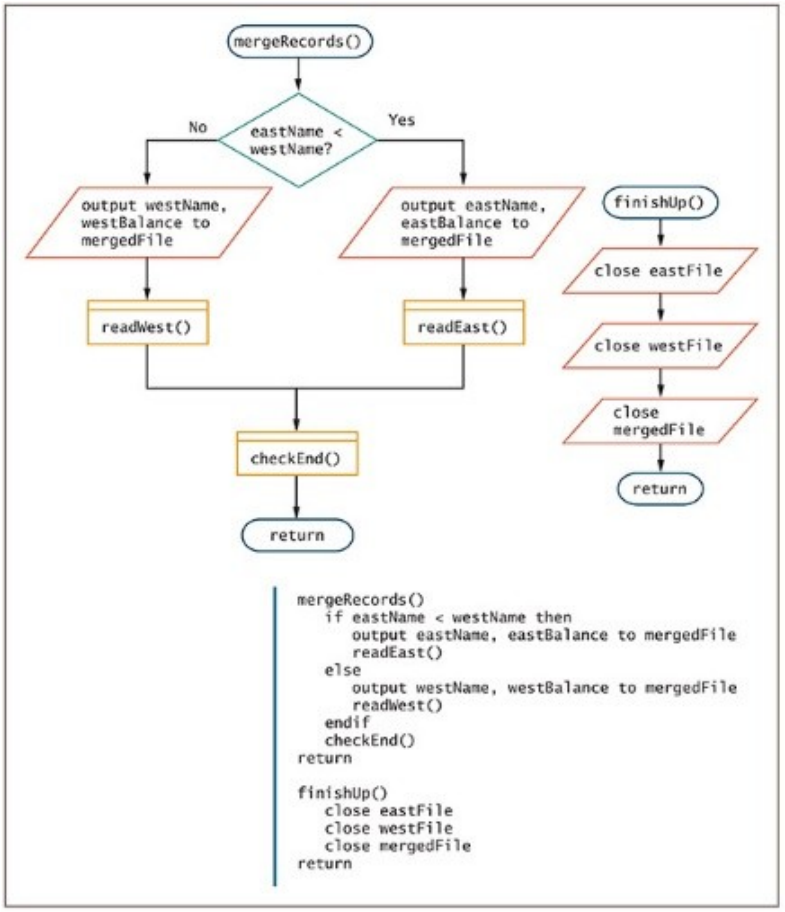


Figure 7-15 The `mergeRecords()` and `finishUp()` modules for the file-merging program

Master and Transaction File Processing

- Some related files have a master-transaction relationship
- **Master file**
  - Holds complete and relatively permanent data
- **Transaction file**
  - Contains temporary data to be used to update the master file
- **Update the master file**
  - Changes to values in its fields based on transactions

Master and Transaction File Processing (continued -1)

- Examples
  - A library maintains a master file of all patrons and a transaction file with information about each book or other items checked out
  - A college maintains a master file of all students and a transaction file for each course registration
  - A telephone company maintains a master file of every telephone line (number) and a transaction file with information about every call

Master and Transaction File Processing (continued -2)

- Updating approaches
  - Change information in master file
  - Copy master file and change new version
- Begin with both files sorted in the same order on the same field

Master and Transaction File Processing (continued -3)

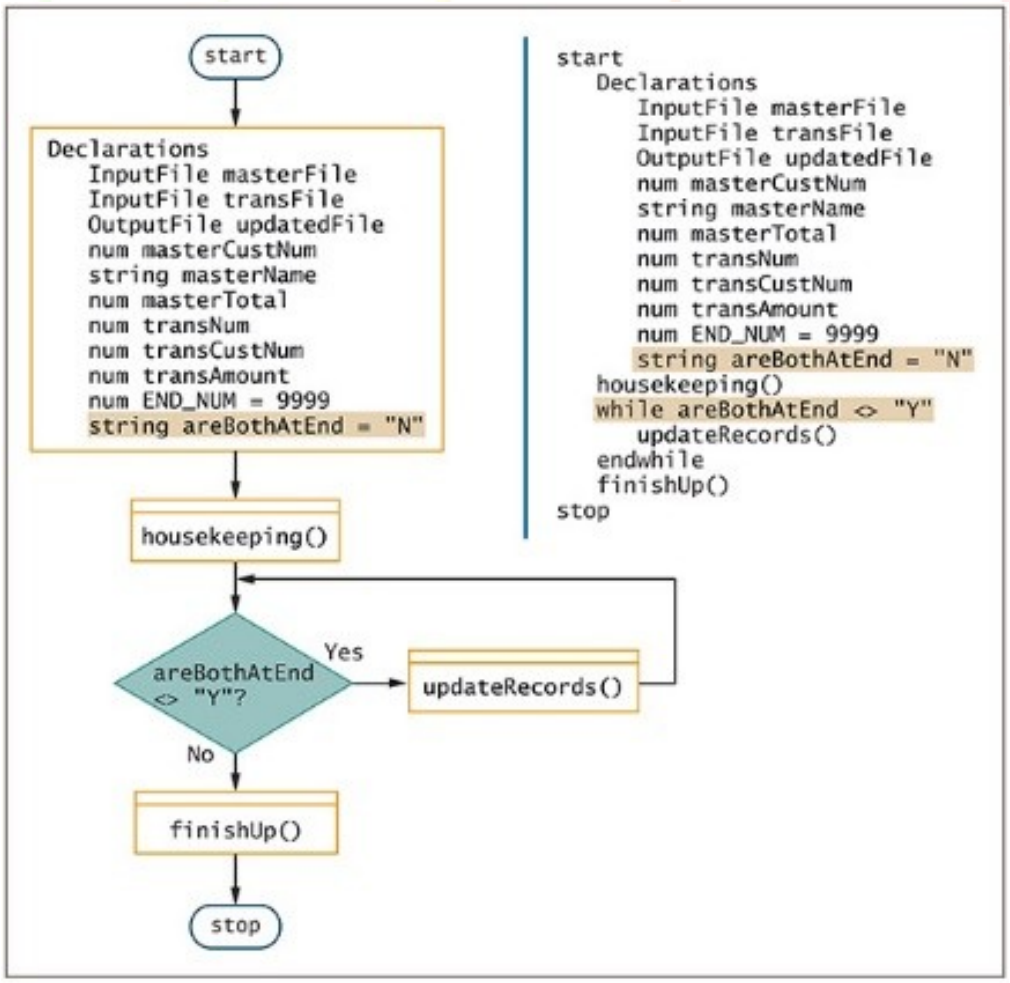


Figure 7-16 Mainline logic for the master-transaction program

Master and Transaction File Processing (continued -4)

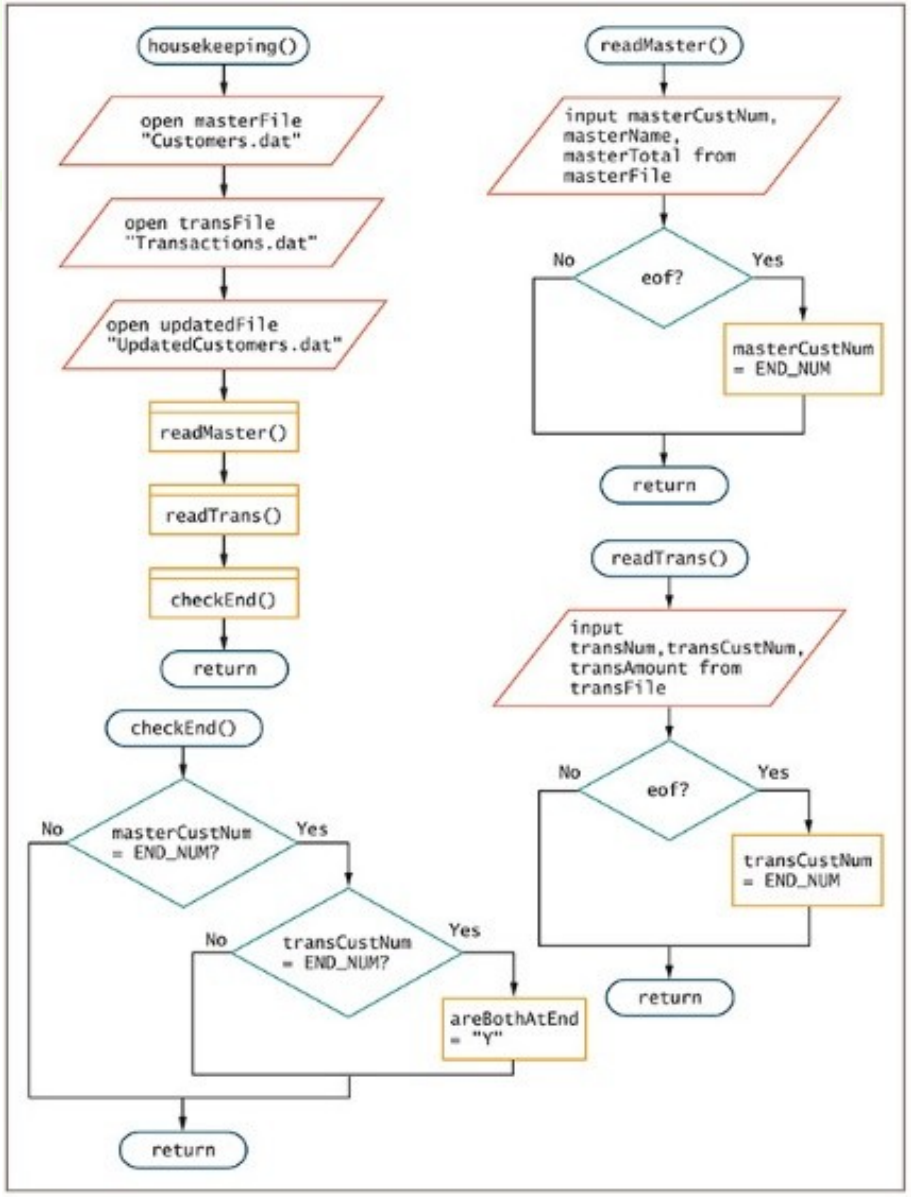


Figure 7-17 The housekeeping() module for the master-transaction program, and the modules it calls (continues)

Master and Transaction File Processing (continued -5)

(continued)

```
housekeeping()
   open masterFile "Customers.dat"
   open transFile "Transactions.dat"
   open updatedFile "UpdatedCustomers.dat"
   readMaster()
   readTrans()
   checkEnd()
return

readMaster()
   input masterCustNum, masterName, masterTotal from masterFile
   if eof then
      masterCustNum = END_NUM
   endif
return

readTrans()
   input transNum, transCustNum, transAmount from transFile
   if eof then
      transCustNum = END_NUM
   endif
return

checkEnd()
   if masterCustNum = END_NUM then
      if transCustNum = END_NUM then
         areBothAtEnd = "Y"
      endif
   endif
return
```

Figure 7-17 The housekeeping() module for the master-transaction program, and the modules it calls

Master and Transaction File Processing (continued -6)

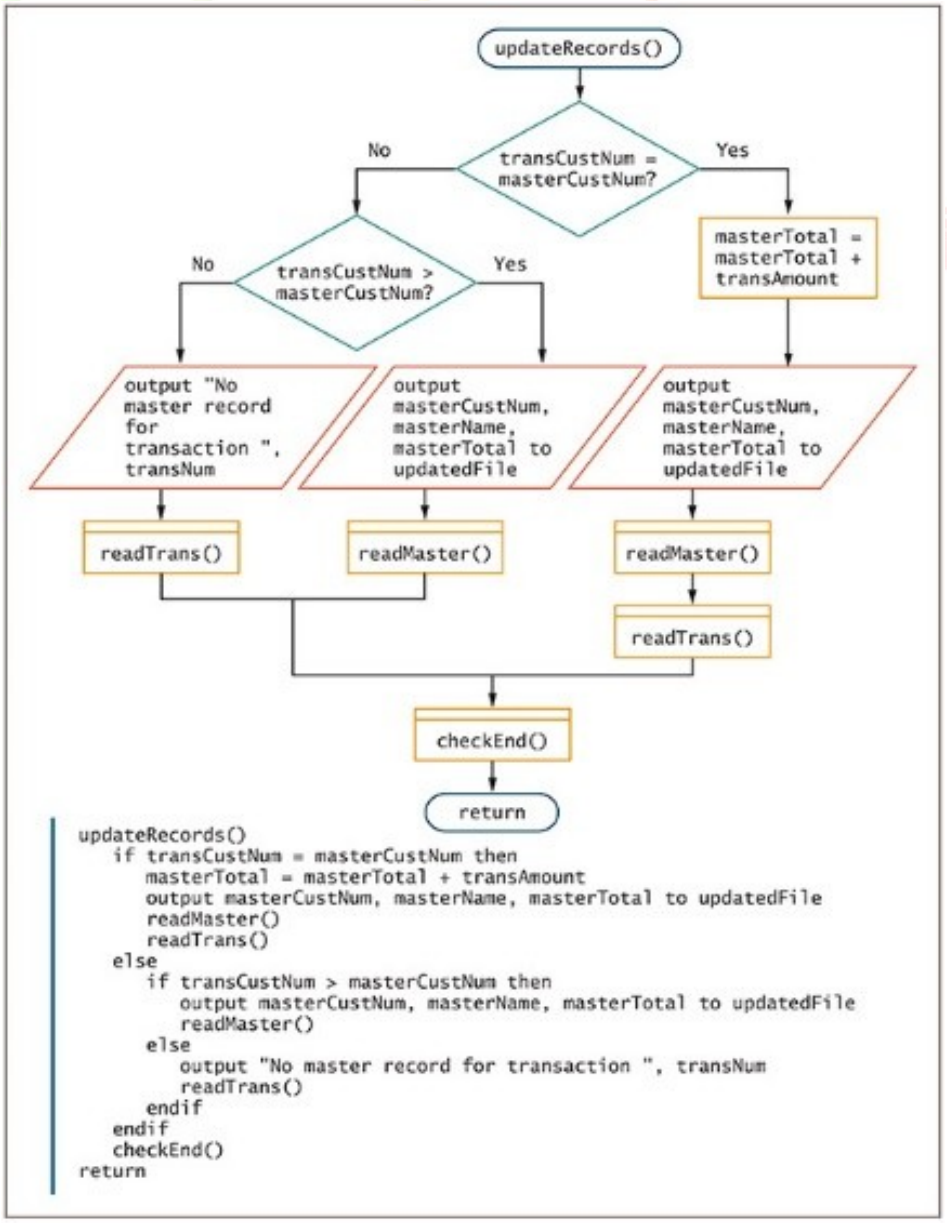


Figure 7-18 The updateRecords() module for the master-transaction program

Master and Transaction File Processing (continued -7)

Master File		Transaction File	
masterCustNum	masterTotal	transCustNum	transAmount
100	1000.00	100	400.00
102	50.00	105	700.00
103	500.00	108	100.00
105	75.00	110	400.00
106	5000.00		
109	4000.00		
110	500.00		

Figure 7-19 Sample data for the file-matching program

Master and Transaction File Processing (continued -8)

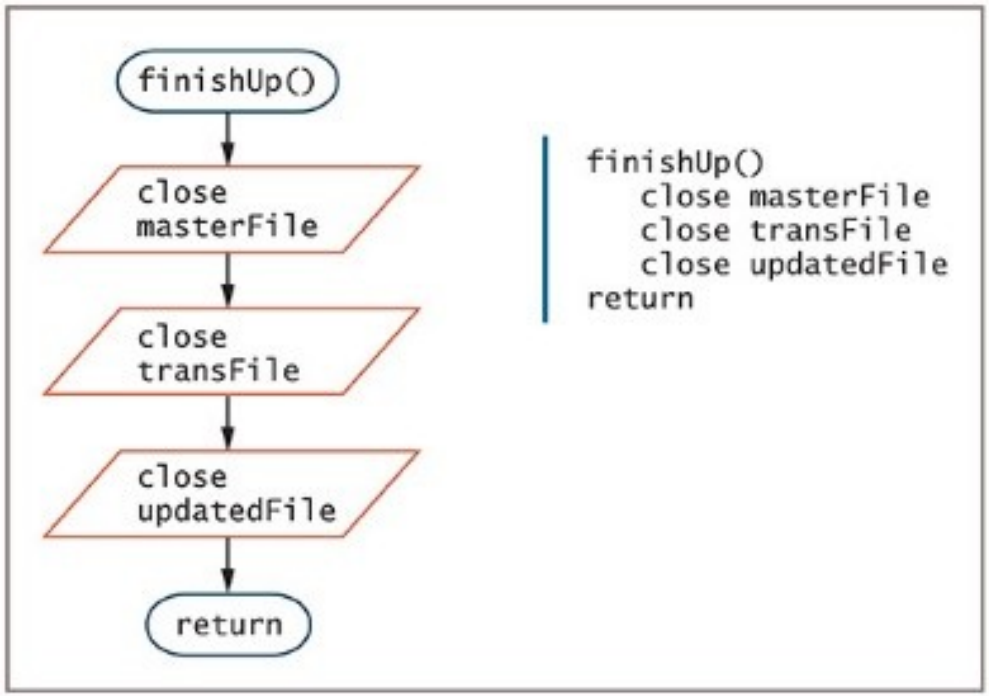


Figure 7-20 The `finishUp()` module for the master-transaction program

Random Access Files

- **Batch processing**
  - Involves performing the same tasks with many records, one after the other
  - Uses sequential files
- **Real-time** applications
  - Require that a record be accessed immediately while a client is waiting
- **Interactive program**
  - A program in which the user makes direct requests

Random Access Files (continued -1)

- **Random access files**
  - Records can be physically located in any order
  - **Instant access files**
    - Files in which records must be accessed immediately
  - Also known as **direct access files**

Random Access Files (continued -2)

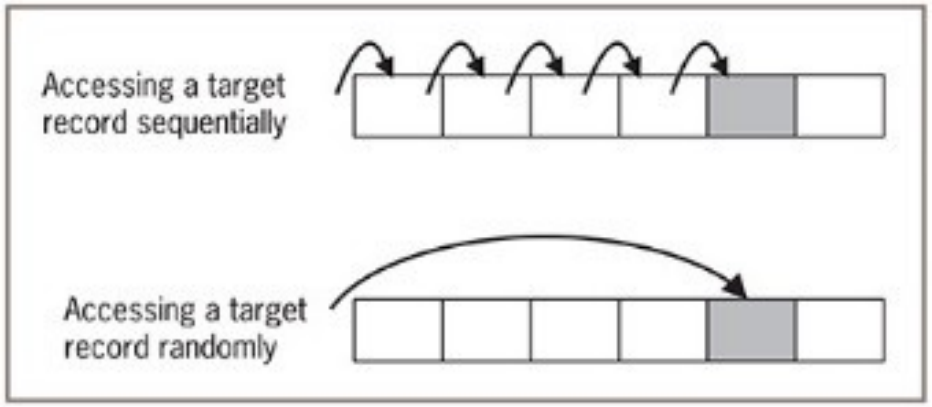


Figure 7-21 Accessing a record in a sequential file and in a random access file

Summary

- Computer file
  - A collection of data stored on a nonvolatile device in a computer system
- Data items are stored in a hierarchy
- To use a data file you must declare, open, read, write, and close the file
- Sequential file: records stored one after another in some order

Summary (continued -1)

- Control break program reads a sorted sequential file and performs special processing based on a change in one or more fields in each record in the file
- Merging files combines two or more files
  - Maintains the same sequential order
- Master files
  - Hold permanent data
  - Updated by transaction files

Summary (continued -2)

- Master files
  - Hold relatively permanent data
  - Updated by transaction files
- Real-time applications
  - Require random access files where records can be located in any order
- Instant access files and direct access files are files in which records must be accessed immediately