14

File Processing

Objectives

In this chapter you'll:

- Create, read, write and update files.
- Perform sequential file processing.
- Perform random-access file processing.
- Use high-performance unformatted I/O operations.
- Understand the differences between formatted-data and raw-data file processing.
- Build a transaction-processing program using random-access file processing.
- Understand the concept of object serialization.

Self-Review Exercises

14.1 *(Fill in the Blanks)* Fill in the blanks in each of the following:

a) Member function _____ of the file streams `fstream`, `ifstream` and `ofstream` closes a file.

ANS: `close`.

b) The `ostream` member function _____ is normally used when writing data to a file in random-access applications.

ANS: `write`.

c) Member function _____ of the file streams `fstream`, `ifstream` and `ofstream` opens a file.

ANS: `open`.

d) The `istream` member function _____ is normally used when reading data from a file in random-access applications.

ANS: `read`.

e) Member functions _____ and _____ of `istream` and `ostream` set the file-position pointer to a specific location in an input or output stream, respectively.

ANS: `seekg`, `seekp`.

14.2 *(True or False)* State which of the following are *true* and which are *false*. If *false*, explain why.

a) Member function `read` cannot be used to read data from the input object `cin`.

ANS: False. Function `read` can read from any input stream object derived from `istream`.

b) You must create the `cin`, `cout`, `cerr` and `clog` objects explicitly.

ANS: False. These four streams are created automatically for you. The `<iostream>` header must be included in a file to use them. This header includes declarations for each predefined stream object.

c) A program must call function `close` explicitly to close a file associated with an `ifstream`, `ofstream` or `fstream` object.

ANS: False. The files will be closed when destructors for `ifstream`, `ofstream` or `fstream` objects execute when the stream objects go out of scope or before program execution terminates, but it's a good programming practice to close all files explicitly with `close` once they're no longer needed.

d) If the file-position pointer points to a location in a sequential file other than the beginning of the file, the file must be closed and reopened to read from the beginning of the file.

ANS: False. Member functions `seekp` and `seekg` can be used to reposition the "put" or "get" file-position pointers, respectively, to the beginning of the file.

e) The `ostream` member function `write` can write to standard-output stream `cout`.

ANS: True.

f) Data in sequential files always is updated without overwriting nearby data.

ANS: False. In most cases, sequential file records are not of uniform length. Therefore, it's possible that updating a record will cause other data to be overwritten.

g) Searching all records in a random-access file to find a specific record is unnecessary.

ANS: True.

h) Records in random-access files must be of uniform length.

ANS: False. Records in a random-access file normally are of uniform length.

i) Member functions `seekp` and `seekg` must seek relative to the beginning of a file.

ANS: False. It's possible to seek from the beginning of the file, from the end of the file and from the current position in the file.

14.3 Assume that each of the following statements applies to the same program.

a) Write a statement that opens file `oldmast.dat` for input; use an `ifstream` object called `inOldMaster`.

ANS: `ifstream inOldMaster{"oldmast.dat", ios::in};`

b) Write a statement that opens file `trans.dat` for input; use an `ifstream` object called `inTransaction`.

ANS: `ifstream inTransaction{"trans.dat", ios::in};`

c) Write a statement that opens file `newmast.dat` for output (and creation); use `ofstream` object `outNewMaster`.

ANS: `ofstream outNewMaster{"newmast.dat", ios::out};`

d) Write a statement that reads a record from the file `oldmast.dat`. The record consists of integer `accountNumber`, string `name` (containing spaces) and floating-point `currentBalance`. Use `ifstream` object `inOldMaster`.

ANS: `inOldMaster >> accountNumber >> quoted(name) >> currentBalance;`

e) Write a statement that reads a record from the file `trans.dat`. The record consists of integer `accountNum` and floating-point `dollarAmount`. Use `ifstream` object `inTransaction`.

ANS: `inTransaction >> accountNum >> dollarAmount;`

f) Write a statement that writes a record to the file `newmast.dat`. The record consists of integer `accountNum`, string `name`, and floating-point `currentBalance`. Use `ofstream` object `outNewMaster`.

ANS: `outNewMaster << accountNum << " " << name << " " << currentBalance;`

14.4 Find the error(s) and show how to correct it (them) in each of the following.

a) File `payables.dat` referred to by `ofstream` object `outPayable` has not been opened.

```
outPayable << account << company << amount << endl;
```

ANS: *Error:* The file `payables.dat` has not been opened before the attempt is made to output data to the stream.
Correction: Use `ofstream` function `open` to open `payables.dat` for output.

b) The following statement should read a record from the file `payables.dat`. The `ifstream` object `inPayable` refers to this file, and `ifstream` object `inReceivable` refers to the file `receivables.dat`.

```
inReceivable >> account >> company >> amount;
```

ANS: *Error:* The incorrect `ifstream` object is being used to read a record from the file named `payables.dat`.
Correction: Use `ifstream` object `inPayable` to refer to `payables.dat`.

c) The file `tools.dat` should be opened to add data to the file without discarding the current data.

```
ofstream outTools("tools.dat", ios::out);
```

ANS: *Error:* The file's contents are discarded because the file is opened for output (`ios::out`).
Correction: To add data to the file, open the file either for updating (`ios::ate`) or for appending (`ios::app`).

Exercises

NOTE: Solutions to the programming exercises are located in the **`ch14solutions`** *folder.*

14.5 ***(Fill in the Blanks)*** Fill in the blanks in each of the following:

a) Computers store large amounts of data on secondary storage devices as _____.

ANS: files.

b) The standard stream objects declared by header `<iostream>` are _____, _____, _____ and _____.

ANS: `cin`, `cout`, `cerr`, `clog`.

c) `ostream` member function _____ repositions the file-position pointer in a file.

ANS: `put`.

d) _____ is the default file-open mode for an `ofstream`.

ANS: `write`.

e) `istream` member function _____ repositions the file-position pointer in a file.

ANS: `seekg`.

14.10 Write a series of statements that accomplish each of the following. Assume that we've defined class `Person` that contains the `private` data members

```
char lastName[15];
char firstName[10];
int age;
int id;
```

and `public` member functions

```
// accessor functions for id
void setId(int);
int getId() const;

// accessor functions for lastName
void setLastName(const string&);
string getLastName() const;

// accessor functions for firstName
void setFirstName(const string&);
string getFirstName() const;

// accessor functions for age
void setAge(int);
int getAge() const;
```

Also assume that any random-access files have been opened properly.

a) Initialize `nameage.dat` with 100 records that store values `lastName ="unassigned"`, `firstName = ""` and `age = 0`.

ANS:

```
// fstream object "fileObject" corresponds to file nameage.dat
Person blankPerson;
blankPerson.setLastName("unassigned");
blankPerson.setFirstName("");
blankPerson.setAge("0");
blankPerson.id = 0;

for (size_t r{0}; r < 100; r++) {
   fileObject.write(reinterpret_cast<const char *>(&blankPerson),
      sizeof(Person) );
}
```

b) Input 10 last names, first names and ages, and write them to the file.

ANS:

```
string last;
string first;
string age;
int id;
unsigned int counter{1};

while (counter <= 10) {
```

```
   do { // obtain id-number value
      cout << "Enter id number for new record (1 - 100): "
      cin << id;
   } while ((id < 1) || (id > 100));

   // move file-position pointer to correct record in file
   fileObject.seekg((id - 1) * sizeof(Person));

   // read record from file to determine if one already exists
   Person person;
   fileObject.read(reinterpret_cast<char *>(&person), sizeof(Person));

   // create record, if record does not previously exist
   if (person.getId() == 0) {
      cout << "Enter last name, first name, and age: ";
      cin >> setw(15) last;
      cin >> setw(15) first;
      cin >> setw(4) >> age;
      person.setLastName(last);
      person.setFirstName(first);
      person.setAge(age);
      person.setId(id);

      // move file-position pointer to correct record in file
      fileObject.seekp((id - 1) * sizeof(Person));

      // insert new record
      fileObject.write(reinterpret_cast<const char *>(&person),
         sizeof(Person));

      ++counter; // record added, increase counter
   }
   else { // display error if record previously exists
      cerr << "Record #" << id << " already contains data." << endl;
   }
}
```

c) Update a record that already contains information. If the record does not contain information, inform the user "`No info`".

ANS:

```
char lastName[15];
char firstName[10];
int age;
int id;

do { // obtain id-number value
   cout << "Enter id number for new record (1 - 100): "
   cin << id;
} while ((id < 1) || (id > 100));

// move file-position pointer to correct record in file
fileObject.seekg((id - 1) * sizeof(Person));

// read record from file to determine if one already exists
Person person;
fileObject.read(reinterpret_cast<char *>(&person), sizeof(Person));

// update record, if no record currently exists
if (person.getId() != 0) {
   cout << "Enter new last name, first name, and age: ";
   cin >> setw(15) last;
```

```
      cin >> setw(10) >> first;
      cin >> age;
      person.setLastName(last);
      person.setFirstName(first);
      person.setAge(age);
      person.setId(id);

      // move file-position pointer to correct record in file
      fileObject.seekp((id - 1) * sizeof(Person));

      // insert new record
      fileObject.write(reinterpret_cast<const char *>(&person),
         sizeof(Person));
   }
   else { // display error if record did not previously exists
      cerr << "No info." << endl;
   }
```

d) Delete a record that contains information by reinitializing that particular record.

ANS:

```
   do { // obtain id-number value
      cout << "Enter id number for new record (1 - 100): "
      cin << id;
   } while ((id < 1) || (id > 100));

   // move file-position pointer to correct record in file
   fileObject.seekg((id - 1) * sizeof(Person));

   // read record from file
   Person person;
   fileObject.read(reinterpret_cast<char *>(&person), sizeof(Person));

   if (person.id != 0) { // delete record, if record exists in file
       // create blank record
      Person blankPerson;
      blankPerson.setLastName("unassigned");
      blankPerson.setFirstName("");
      blankPerson.setAge("0");
      blankPerson.id = 0;

      // move file-position pointer to correct record in file
      fileObject.seekp((id - 1) * sizeof(Person));

      // replace existing record with blank record
      fileObject.write(reinterpret_cast<const char *>(&blankPerson),
         sizeof(Person));

      cout << "Record #" << id << " deleted." << endl;
   }
   else { // display error if record does not exist
      cerr << "Record #" << id << " is empty." << endl;
   }
```