

CC-211L

Object Oriented Programming

Laboratory 13

File Processing

Version: 1.0.0

Release Date: 16-5-2024

Department of Information Technology

University of the Punjab

Lahore, Pakistan

Contents:

- Learning Objectives
- Required Resources
- General Instructions
- Background and Overview
 - Files and Streams
 - Sequential File
 - Random-Access File
- Activities
 - Pre-Lab Activity
 - Files and Streams
 - Creating a Sequential File
 - Opening a file
 - Closing a file
 - Task 01
 - Task 02
 - Task 03
 - In-Lab Activity
 - File Position Pointers
 - Member functions `tellp()` and `tellg()`
 - Updating a Sequential File
 - Creating a Random File
 - Writing Data to a Random File
 - Reading Data from a Random File
 - Task 01
 - Task 02
 - Task 03
 - Post-Lab Activity
 - Task 01
- Submissions
- Evaluations Metric
- References and Additional Material
- Lab Time and Activity Simulation Log

Learning Objectives:

- Files and Streams
- Create a Sequential File
- Read a Sequential File
- Update a Sequential File
- Random Access File
- Create a Random-Access File
- Read a Random-Access File

Resources Required:

- Desktop Computer or Laptop
- Microsoft ® Visual Studio 2022

General Instructions:

- In this Lab, you are **NOT** allowed to discuss your solution with your colleagues, even not allowed to ask how is s/he doing, this may result in negative marking. You can **ONLY** discuss with your Teaching Assistants (TAs) or Lab Instructor.
- Your TAs will be available in the Lab for your help. Alternatively, you can send your queries via email to one of the followings.

Background and Overview:

Files and Streams:

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.

A stream is an abstraction that represents a device on which operations of input and output are performed. A stream can be represented as a source or destination of characters of indefinite length depending on its usage.

Sequential File:

A sequential file is simply a file where the data is stored one item after another. A more technical definition would be that a sequential file is a collection of data stored on a disk in a sequential, non-indexed, manner. C++ treats all such files simply as a sequence of bytes. Each file ends with a special end of file marker.

Random-Access File:

Random file access enables us to read or write any data in our disk file without having to read or write every piece of data before it while in Sequential files to reach data in the middle of the file you must go through all the data that precedes it. We can quickly search for data, modify data, delete data in a random-access file.

Activities:

Pre-Lab Activities:

Files and Streams:

C++ views each file simply as a sequence of bytes. Each file ends either with an end-of-file marker or at a specific byte number recorded in an operating-system-maintained administrative data structure. When a file is opened, an object is created, and a stream is associated with the object.

To perform file processing in C++, headers `<iostream>` and `<fstream>` must be included.

Header `<fstream>` includes the definitions for the stream class templates

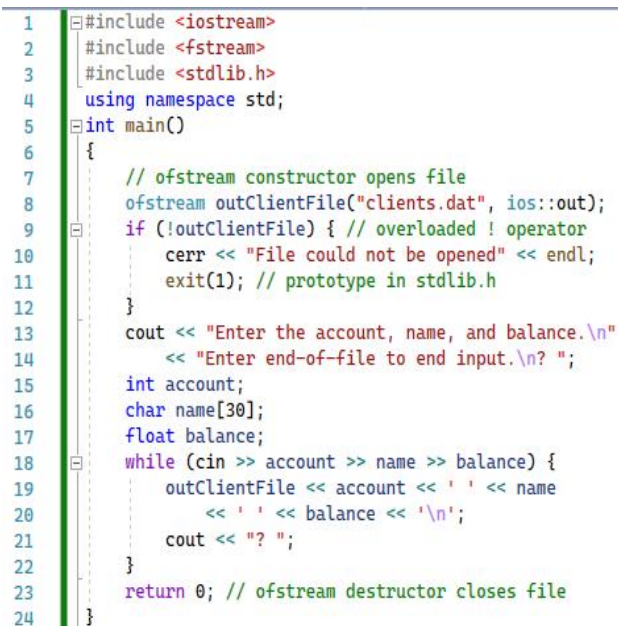
- `basic_ifstream`—a subclass of `basic_istream` for file input
- `basic_ofstream`—a subclass of `basic_ostream` for file output
- `basic_fstream`—a subclass of `basic_iostream` for file input and output.

In addition, the `<fstream>` library provides typedef aliases for these template specializations:

- `ifstream` is an alias for `basic_ifstream<char>`
- `ofstream` is an alias for `basic_ofstream<char>`
- `fstream` is an alias for `basic_fstream<char>` .

Creating a Sequential File:

Example:



```
1 #include <iostream>
2 #include <fstream>
3 #include <stdlib.h>
4 using namespace std;
5 int main()
6 {
7     // ofstream constructor opens file
8     ofstream outClientFile("clients.dat", ios::out);
9     if (!outClientFile) { // overloaded ! operator
10         cerr << "File could not be opened" << endl;
11         exit(1); // prototype in stdlib.h
12     }
13     cout << "Enter the account, name, and balance.\n"
14          << "Enter end-of-file to end input.\n? ";
15     int account;
16     char name[30];
17     float balance;
18     while (cin >> account >> name >> balance) {
19         outClientFile << account << ' ' << name
20                      << ' ' << balance << '\n';
21         cout << "? ";
22     }
23     return 0; // ofstream destructor closes file
24 }
```

Fig. 01

(Sequential File)

Opening a file:

Following syntax is followed to open a file:

Ofstream outClientFile;

outClientFile.open(Filename, Mode)

Following are the different modes to open a file:

Mode	Description
ios::app	Append all output to the end of the file.
ios::ate	Open a file for output and move to the end of the file. Data can be written anywhere in the file.
ios::in	Open file for input.
ios::out	Open file for output.
ios::trunc	Discard's the file content.
ios::binary	Open a file for binary.

Closing a file:

File is closed implicitly when a destructor for the corresponding object is called. It can also be called explicitly by using member function close:

outClientFile.close();

Reading a Sequential file:

Example:

```
1  #include <iostream>
2  #include <fstream>
3  #include <iomanip>
4  #include <stdlib.h>
5  using namespace std;
6  void outputLine(int, const char*, double);
7  int main(){
8      // ifstream constructor opens the file
9      ifstream inClientFile("clients.dat", ios::in);
10     if (!inClientFile) {
11         cerr << "File could not be opened\n";
12         exit(1);
13     }
14     int account;
15     char name[30];
16     double balance;
17     cout << setiosflags(ios::left) << setw(10) << "Account"
18          << setw(13) << "Name" << "Balance\n";
19     while (inClientFile >> account >> name >> balance)
20         outputLine(account, name, balance);
21     return 0; // ifstream destructor closes the file
22 }
23 void outputLine(int acct, const char* name, double bal){
24     cout << setiosflags(ios::left) << setw(10) << acct
25          << setw(13) << name << setw(7) << setprecision(2)
26          << resetiosflags(ios::left)
27          << setiosflags(ios::fixed | ios::showpoint)
28          << bal << '\n';
29 }
```

Fig. 02 (Reading Sequential File)

File Content:



```
clients.dat - Notepad
File Edit Format View Help
1 Ahmad 100000
2 Ali 1520000
3 Usman 259790
```

Fig. 03 (Reading

Sequential File)

Output:



```
Microsoft Visual Studio Debug Console
Account Name Balance
1 Ahmad 100000.00
2 Ali 1520000.00
3 Usman 259790.00
```

Fig. 04 (Reading

Sequential File)

**Task 01: Word Occurrence
marks]**


[Estimated time 15 minutes / 15

- Create a text file and write some text in it
- Read the file contents using Sequential Access
- Count the words occurrence and display the most repetitive word/s on the Console

**Task 02: Map 2D Array
marks]**

[Estimated time 15 minutes / 15

- Create a text file and write 2D array contents as shown in the figure



```
3 4
1 2 3 4
5 6 7 8
1 2 3 4
```

Fig. 05 (Pre-

Lab Task)

- Read the file contents using Sequential Access
- Map the contents to a 2D array and display it on the Console.
- On the first line, there are number of rows and columns being printed.

**Task03: Running Calculator
marks]**

[Estimated time 20 minutes / 20

Create a program in cpp in which user will keep on inserting new numbers to get the total sum until user inputs -1. Get each number from user, write it in a file “Numbers.txt” and get the sum of all the numbers written in it. Numbers will be stored with space between them and comma. Like for example enters 5, 10, 40, 50, 28, 27. You will display the sum of all the numbers in file when ever a new number is inserted in file.