

# Ch13-FileIO

August 7, 2020

## 1 13 File Input/Output (IO)

### 1.1 Topics

- input/output streams
- file input stream
- file output stream
- reading unstructured and structured text files
- formatting file output

### 1.2 13.1 Streams

- a **stream** is an abstract object that represents the flow of data from a source like keyboard or a file to a destination like the screen or a file
- we've learned earlier about standard io streams
- `iostream` is used to read the data from standard input (keyboard)
  - data is then stored in computer memory to be manipulated to solve problems
  - data result is written to the standard output (monitor) from computer memory
- C++ uses various streams to reading data from and write data to
  - `stringstream` is another stream that creates stream of strings
- often programs need to read data, process it and write the result back to secondary devices for permanent storage
- file stream is used to read data from secondary storage like disk and write result and data back to it for permanent

### 1.3 13.2 File stream

- we use `<fstream>` header to create input and output file streams

### 1.4 13.3 File input

- **ifstream** object is created to read data from file
- it creates a stream that flows from the file into the program (memory)

#### 1.4.1 steps for file input

1. open file to read data from
  - file must exist; run-time error otherwise
2. read file contents
3. close the file

### 1.4.2 open file

- to open the file you need to create ifstream object
- syntax to create ifstream object:

```
// create stream without opening the file
ifstream objectName;
// create object and open the given file
ifstream objectName("file_name");
```

- objectName is any identifier you want to use it for this particular ifstream
- file name is passed as an argument; we'll learn how to read text files
- file name must be present to read data from
- let's open and read this sample text file called `inputfile.txt`

```
[1]: #include <fstream> // ifstream
      #include <iostream>
      #include <string>

      using namespace std;
```

```
[2]: // declare ifstream object
      ifstream fin;
      // i prefer fin as stream object name; rhymes with cin
```

```
[3]: // open the file using open method
      fin.open("inputfile.txt");
```

```
[4]: // declare stream object and open the given file
      ifstream fin1("inputfile.txt");
```

### 1.4.3 read data

- once the ifstream object is created, reading data is similar to reading from istream
- we use >> (input extraction) and getline functions to read the data
- syntax:

```
ifstreamObject >> variable1 >> variable2 >> ...;
```

- >> - extracts one value of variable type and stops at a whitespace or mismatch type

```
getline(ifstreamObject, strVariable);
```

- getline( ) - reads the whole line as string into strVariable

```
[5]: // let's read couple of words from inputfile.txt
      string word1, word2;
```

```
[6]: fin >> word1 >> word2;
```

```
[7]: cout << word1 << " " << word2;
```

this is

```
[8]: // let's read the rest of the line  
string line
```

```
[9]: getline(fin, line);
```

```
[10]: cout << line;
```

first sentence.

```
[11]: // let's read the next line  
getline(fin, line);  
cout << line;
```

this is 2nd sentence

```
[12]: // let's read the next line  
getline(fin, line);  
cout << line;
```

some numbers are below

```
[13]: // let's read the 3 numbers  
int nums[3];
```

```
[14]: fin >> nums[0] >> nums[1] >> nums[2];
```

```
[15]: cout << nums[0] << " " << nums[1] << " " << nums[2];  
// done reading all the contents of the file
```

10 20 30

```
[15]: @0x10d2b5ec0
```

#### 1.4.4 close file

- use close() method on ifstream objects

```
[16]: fin.close();
```

```
[17]: // can check if file is open  
fin.is_open()
```

```
[17]: false
```

```
[18]: fin1.close();
```

#### 1.4.5 ifstream member functions

- there are a whole bunch of methods available in ifstream objects
- all the methods can be found here with examples:  
[https://en.cppreference.com/w/cpp/io/basic\\_ifstream](https://en.cppreference.com/w/cpp/io/basic_ifstream)

### 1.5 13.4 Reading structured data

- one must know the contents of the data in order to properly read them into program
- reading unstructured data is difficult
  - best way is to read line by line and process each line
- reading structured data is a bit easier
- let's read the structured data provided in `studentgrades.txt` file
  - there are 3 rows or records and 5 columns (values) for each record
  - first 2 columns are string (names) and the rest 3 columns are integers (grades)

```
[19]: #include <iostream>
#include <fstream>
#include <string>
#include <functional>
#include <algorithm>

using namespace std;
```

```
[20]: // struct type is a perfect way to read these student's grades
struct Student {
    string firstName;
    string lastName;
    int grades[3];
    float averageGrade;
    char letterGrade;
};
```

```
[21]: // let's create a vector of Student type to store all the records
#include <vector>
```

```
[22]: vector<Student> gradebook;
```

```
[23]: //ifstream fin; // declare it if not declared before
```

```
[24]: // let's read the data
// fin is ifstream object declared above
fin.open("studentgrades.txt");
```

```
[25]: // let's compute average grade
float average(const Student & s) {
    float sum = s.grades[0] + s.grades[1] + s.grades[2];
    return sum/3.0;
}
```

```
[26]: while(!fin.eof()) { // eof() checks if end-of-file has been reached
    // create Student object to hold the data temporarily
    Student temp;
    fin >> temp.firstName >> temp.lastName >> temp.grades[0] >> temp.grades[1]
    ↪>> temp.grades[2];
    if (!fin.good()) break;
    temp.averageGrade = average(temp);
    // add the temp to gradebook
    gradebook.push_back(temp);
}
```

```
[27]: // close file
fin.close();
```

```
[28]: // let's write a function to print Student's info
void printStudent(const Student & s) {
    cout << s.firstName << " " << s.lastName << " " << s.grades[0] << " "
        << s.grades[1] << " " << s.grades[2] << " avg: " << s.averageGrade;
}
```

```
[29]: // let's print the first student's info
printStudent(gradebook[0]);
```

John Smith 100 95 85 avg: 93.3333

```
[30]: printStudent(gradebook[0]);
```

John Smith 100 95 85 avg: 93.3333

```
[31]: // print all the students' info
for(Student s: gradebook) {
    printStudent(s);
    cout << endl;
}
```

John Smith 100 95 85 avg: 93.3333

Jane Doe 85 89 99 avg: 91

Jill Jones 56 89 99 avg: 81.3333

```
[32]: // sort the student records based on average score?
// need to define a comparison function and pass it to sort
```

```
// compares two students' average grades in ascending order
bool compareSmaller(const Student & s1, const Student & s2) {
    return (s1.averageGrade < s2.averageGrade);
}
```

```
[33]: // now we can sort the gradebook
sort(gradebook.begin(), gradebook.end(), compareSmaller);
```

```
[34]: // print all the students' info
for(Student s: gradebook) {
    printStudent(s);
    cout << endl;
}
```

```
Jill Jones 56 89 99 avg: 81.3333
Jane Doe 85 89 99 avg: 91
John Smith 100 95 85 avg: 93.3333
```

```
[35]: // let's write a compare function for descending order
bool compareGreater(const Student & s1, const Student & s2) {
    return (s1.averageGrade > s2.averageGrade);
}
```

```
[36]: // now we can sort the gradebook in descending order using our own compare_
→function
sort(gradebook.begin(), gradebook.end(), compareGreater);
```

```
[37]: // print all the students' info
// looks like this could go into a function...
for(Student s: gradebook) {
    printStudent(s);
    cout << endl;
}
```

```
John Smith 100 95 85 avg: 93.3333
Jane Doe 85 89 99 avg: 91
Jill Jones 56 89 99 avg: 81.3333
```

## 1.6 13.5 File output

- sending output data to a file is similar to reading data
- 3 steps:
  1. Create a new file
  2. Write data to the file
  3. Close the file

### 1.6.1 create file

- to write data to a file, first create ofstream object
- syntax:

```
// create ofstream object without creating a file
ofstream fout;
// create ofstream object with a given file
ofstream fout("output-filename");
```

- if the file exists, the file will be truncated
  - existing data will be lost

```
[38]: #include <fstream> // ifstream and ofstream
#include <iostream>
#include <string>
#include <iomanip>
#include <vector>
#include <algorithm>

using namespace std;
```

```
[39]: // create output file stream object
ofstream fout;
```

```
[40]: // create file
fout.open("outputfile.txt");
// you should see a new text file created in the same folder as this notebook
```

```
[41]: ofstream fout1("outputfile1.txt");
// you should see a new text file created in the same folder as this notebook
```

### 1.6.2 write data

- writing data to file is similar to writing data to std output stream
- use << (output insertion operator) with the stream object

```
[42]: // write data to output file stream
fout << "Hello World!" << endl;
fout1 << 2 << " + " << 2 << " = " << (2+2) << endl;
```

### 1.6.3 close file

- closing file is important specially that was opened to write
- file will be locked if it's not explicitly closed until the program ends

```
[43]: fout.close();
fout1.close();
```

## 1.7 13.6 Formatting output

- iomanip manipulators work exactly the same way for file output
- fixed, setw(), setprecision(), left, right, ws, setfill(), etc. all can be used to format the file output

### 1.7.1 print students' grades report in a tabular format

```
[44]: #include <iomanip>
```

```
[45]: fout.open("studentgradereport.txt");
```

```
[46]: int colWidth;
```

```
[47]: colWidth = 20;
```

```
[48]: // print all the students' info to the fout stream

// write column headers
fout << setw(90) << setfill('=') << " " << setfill(' ') << endl;
fout << setw(colWidth) << left << "First Name"
    << setw(colWidth) << left << "Last Name";
// students grades
for(int i=0; i<3; i++) {
    string testHeader = "test" + to_string(i+1);
    fout << setw(10) << right << testHeader;
}

fout << setw(15) << right << "Avgerage" << endl;
fout << setw(90) << right << setfill('=') << " " << endl;

// write records
fout << setfill(' ') << fixed << setprecision(1);
for(Student s: gradebook) {
    fout << setw(colWidth) << left << s.firstName
        << setw(colWidth) << left << s.lastName;
    for(int i=0; i<3; i++)
        fout << setw(10) << right << s.grades[i];
    fout << setw(15) << right << s.averageGrade << endl;
}
fout << setw(90) << setfill('*') << " " << endl;
```

```
[49]: // convert the above code to a function!
// all the stream objects must be passed-by reference!
// out is a generic ostream parameter (can be cout or fout)
void writeResults(ostream & out) {
    // print all the students' info to the fout stream
```



```

// write column headers
out << setw(90) << setfill('=') << " " << setfill(' ') << endl;
out << setw(colWidth) << left << "First Name"
    << setw(colWidth) << left << "Last Name";
// students grades
for(int i=0; i<3; i++) {
    string testHeader = "test" + to_string(i+1);
    out << setw(10) << right << testHeader;
}
out << setw(15) << right << "Avgerage" << endl;
out << setw(90) << setfill('=') << " " << endl;

// write records
out << setfill(' ') << fixed << setprecision(1);
for(Student s: gradebook) {
    out << setw(colWidth) << left << s.firstName
        << setw(colWidth) << left << s.lastName;
    for(int i=0; i<3; i++)
        out << setw(10) << right << s.grades[i];
    out << setw(15) << right << s.averageGrade << endl;
}
out << setw(90) << setfill('*') << " " << endl;
}

```

```

[50]: // write to standard output/console
writeResults(cout);

```

```

=====
=====
First Name          Last Name          test1    test2    test3
Avgerage
=====
=====
John                Smith                100      95      85
93.3
Jane                Doe                  85       89      99
91.0
Jill                Jones                56       89      99
81.3
*****
*****

```

```

[51]: // write to file output
writeResults(fout);

```

```

[52]: // close the file
fout.close();

```

## 1.8 13.7 Exercises

1. Write a program that computes distance between two points in Cartesian coordinates.
  - use struct to represent Point
  - prompt user to enter name of the input file that contains a bunch of points
    - using a text editor manually create a file with two coordinate points (x, y) per line
  - use as many function(s) as possible
  - write at least 3 test cases for each computing functions
  - program continues to run until user wants to quit
  - most of the part is done in Jupyter Notebook demo
2. Write a program to compute area and circumference of a circle using struct.
  - use struct to represent Circle
  - prompt user to enter name of the input text file that contains a bunch of radii of several circles
    - using a text editor manually create a file that contains an arbitrary number of radii
  - use as many function(s) as possible
  - write at least 3 test cases for each computing functions
  - program continues to run until user wants to quit
3. Write a program to compute area and perimeter of a rectangle using struct.
  - use struct to represent Rectangle
  - prompt user to enter name of the input text file that contains lengths and widths of several rectangles
    - using a text editor manually create a file with length and width of a rectangle per line
  - use as many function(s) as possible
  - write at least 3 test cases for each computing functions
  - program continues to run until user wants to quit
4. Write a program to compute area and perimeter of a triangle given 3 sides.
  - use struct to represent Triangle
  - prompt user to enter name of the file that contains 3 sides of several triangles
    - using a text editor manually create a file that contains 3 sides of a triangle per line
  - use as many function(s) as possible
  - write at least 3 test cases for each computing functions
  - program continues to run until user wants to quit

see a sample solution for exercise 4 at [demo\\_programs/Ch13/triangle.cpp](#)

5. A Grade Book:
  - Write a C++ menu-driven program that let's professors keep track of students grades with the following requirements:
  - program must use struct to keep track of students grades
  - program prompts user to enter name of the input text file that contains students information in the following format
    - first name, last name, test1, test2, test3, test4, test5
  - program calculates average grade and the letter grade (A-F) based on the average grade
  - program sorts the student records based on grade in non-increasing order (highest to lowest)
  - program lets user add a new student

- program lets user update existing student's information
  - program lets user delete existing student
  - program saves the data back into the same input file as a database
  - program creates a cleanly formatted report of students' grades
6. Airline Reservation System:
- Write a C++ menu-driven CLI-based program that let's an airline company manage airline reservation on a single aircraft they own with the following requirements:
  - aircraft has 10 rows with 2 seat on each row
  - program provides menu option to display all the available seats
  - program provides menu option to let user pick any available seat
  - program provides menu option to create total sales report
  - program provides menu option to update price of any seat
  - program saves the data into a file

## 1.9 13.8 Kattis problems

- typically Kattis problems don't require File IO

## 1.10 13.9 Summary

- the notebook covered file streams (input and output)
- learned how to read structured and unstructured data
- write and format output to a output file
- exercises and sample solution(s)

[ ]: