Activity #18: Writing to Files

Recorder's Report

Manager:	Reader:
Recorder:	Driver:
Date:	Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 1, Question #4

b) Model 2, Question #10

c) Model 3, Question #18

Activity #18: Writing to Files

In this activity, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to writing to files in C++.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain how to open a text file for writing
- Explain the difference between writing and appending to a file
- Explain how files can be used to save program configuration information

Process Skill Goals

During the activity, students should make progress toward:

- Write code that opens, writes to, and closes a file
- Redirect the output of a program to a text file

Model 1 A C++ Main Function

```cpp
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main() {
  ofstream fout;
  fout.open("studentInfo.txt", ofstream::app);

  string firstName, lastName, studentID;

  cout << "Enter first name: ";
  cin >> firstName;

  cout << "Enter last name: ";
  cin >> lastName;

  cout << "Enter student ID: ";
  cin >> studentID;

  fout << "Name: " << firstName << " " << lastName << endl;
  fout << "Student ID: " << studentID << endl;
  fout << endl;

  fout.close();

  cout << "Done! Data is saved in: studentInfo.txt" << endl;
}

```

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (15 min) Start time:

1. This code can be found in `activity18a.cpp`. Run it and record what output appears on the screen.

2. Locate the file `studentInfo.txt` and examine its contents.

a) What is in this file?

b) Did this file exist before the program ran?

c) On which line of code was the file created?

d) Which lines generated the file contents?

e) Why wasn't the message on line 28 in the file?

3. Run the program again using different input. Then open the `studentInfo.txt` file. What is in the file? Is the data from the first time you ran it still there?

4. Remove the `ofstream::app` argument in the `.open()` function call on line 9 and run the program again with different input.

a) What happened to the contents of `studentInfo.txt`?

b) What did the `ofstream::app` argument do?

5. You should have observed that lines 22-24 send output to the the file `studentInfo.txt`. How does C++ know to send the output to that file?

6. Rewrite the program so the user can keep entering names until they type in `DONE` for the first name.

Model 2 Redirecting Output to a File

```
for (int i = 1; i <= 5; i++) {
  cout << "Number: " << i << endl;
}
cerr << "Done!" << endl;

```

```
        terminal commands

g++ activity18c.cpp -o test.o
./test.o > output.txt 2> error.txt
```

Refer to Model 2 above as your team develops consensus answers to the questions below.

Questions (15 min) Start time:

7. The file `activity18b.cpp` contains the code shown on the left-hand side of the model. Run this program as you normally would and describe its output.

8. Now open a terminal in this same folder (right click on the folder name and pick *Open in Terminal*) and type in the commands shown in the model, pressing `Enter` after each line.

a) What is output to the screen?

b) What new files now appear in the folder?

c) Which file contains the output from `cout`?

d) Which file contains the output from `cerr`?

9. Recall that the command `./program.o < input.txt` *redirects* the contents of the `input.txt` file to the program as if it were entered from the keyboard. The command above redirects the program output to various files.

a) What part of the second terminal command redirects the output set to `cout` to a file?

b) What par tof the command redirects the output sent to `cerr` to a file?

10. Pick a previous homework assignment that produces output. Open a terminal in that folder, compile the program, and save the output to a file. Recall that we have not used `cerr` previously, so you only need to direct the output from `cout`. What command did you use?

Model 3   A Configuration Function

```
void getSetup(int &rows, int &cols, char &charOne, char &charTwo) {
  ifstream configIn;
  configIn.open("gameBoard.cfg");
  if (configIn.is_open()) {
    configIn >> rows;
    configIn >> cols;
    configIn >> charOne;
    configIn >> charTwo;
  } else {
    cout << "Enter Number of Rows: ";
    cin >> rows;
    cout << "Enter Number of Columns: ";
    cin >> cols;
    cout << "Enter Player One Symbol: ";
    cin >> charOne;
    cout << "Enter Player Two Symbol: ";
    cin >> charTwo;
  }
}

```

"gameBoard.cfg"

```
6
5

#
```

Refer to Model 3 above as your team develops consensus answers to the questions below.

Questions (20 min) Start time:

11. Without running the program, determine what the function `getSetup` does.

12. Suppose that this function were called with the contents of `gameBoard.cfg` as shown. To what value would each of the following variables be set?

a) `rows` =

b) `cols` =

c) `charOne` =

d) `charTwo` =

13. What is missing from the function with regard to the `configIn` stream?

14. This function is part of the program in `activity18c.cpp`. Run it and record what it does.

15. Now delete the file `gameBoard.cfg` and run the program again. What does it do differently?

16. A *configuration file* is a file that saves the setup for a program so that it does not have to be entered every time the program runs. Recreate the file `gameBoard.cfg` and adjust its contents so that an 8 by 10 board is printed using characters `$` and `@`. What should appear on each line of `gameBoard.cfg`?

a) Line 1:

b) Line 2:

c) Line 3:

d) Line 4:

17. Instead of changing the contents of the configuration file by hand, it makes sense to have the program save them. Answer the questions below to help you think through adding this feature to the function.

a) What type of variable will we need to declare in the function body?

b) Where in the function should we write to the file?

c) Should we call `.open()` with or without `ofstream::app`?

18. Modify the function so that it saves the configuration file. Indicate what you add.