

Homework 4

file input and output

July 11, 2019

Abstract

The objective is to give you practice with text files. Files containing binary data other than text are only slightly more complicated.

The context of this problem is carrying out a hunt for treasure in a “house” that is represented by a graph. The program will accumulate treasure, and keep track of how much is found in what place. The program will accept user input to customize the search process. Beyond this, the program must obtain the data describing the house from a file. Another file stores the treasure data. Moreover, the program must write a (log) file describing the route taken through the house, and the successive subtotals of treasure gathered.

Be sure to construct test cases for each function you build. Put thought into the test cases. The goal is to know that a function works, when it passes all of its test cases. One test case per function is usually not enough.

- Construct a sequence diagram for your project. You can draw it by hand and include a phone/photo, or draw it with a software tool.
- Use the test-driven development style for developing your code. Document this by taking a new screen shot of your test code every time you make a new test (See Figure 1.), and of your production code every time you extend your production code. It could be a large number of screen shots. Be sure to run your code every time you add a few lines of test or production code. (See Figures 2 and 3.) Do not allow the number of errors to get large.
- An explanation of a representation of a graph was given in the previous assignment. New in this assignment is reading from and writing to files.
- As in the previous assignment, the searcher looks for treasure, and accumulates it. The goal is to maximize the treasure found. Functions supporting this include finding out how much treasure has been accumulated, and adding the amount of a newly found treasure. New in this assignment is that the particulars of the house layout and the treasure values in the rooms are provided in files. The files are provided in the starter code. The file formats are shown in Figures 4 and Figure 5, and for the output file, in Figure 6.

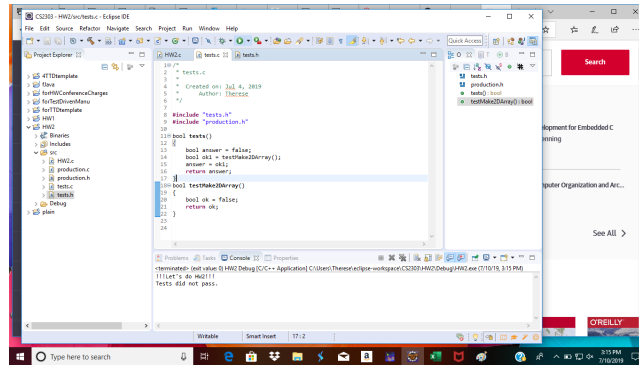


Figure 1: In the process of creating a new test.

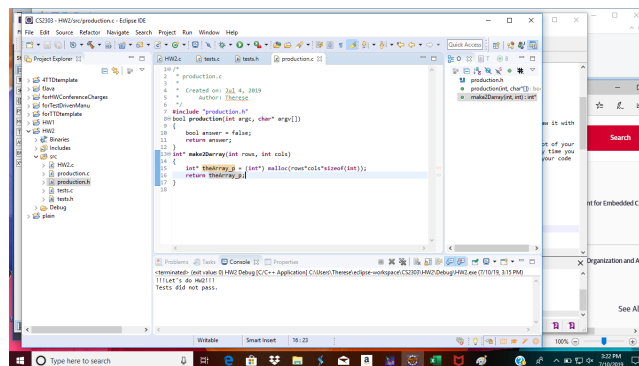


Figure 2: Starting some production code.

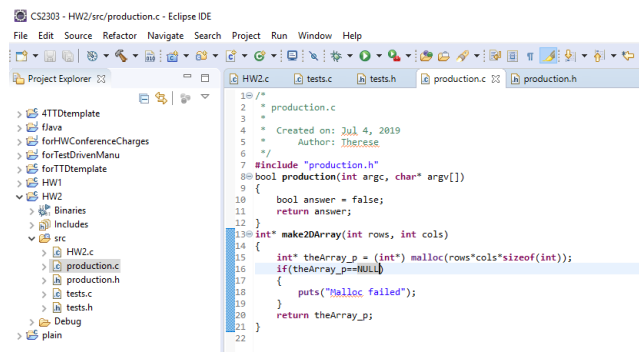


Figure 3: Adding more production code.

```

5 | <-This is the number of rooms
1 1 0 0 0 0
1 1 1 1 0 0
0 1 1 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0
0 0 0 1 0 1
    <-adjacency
    matrix

```


Figure 4: This is the format of the layout file. There is a line containing the number of rooms. That line is followed by that many more lines, giving the adjacency matrix.

```

5
12.2
0
0.3
6.
2.4|

```

Figure 5: This is the format of the treasure allocation file. The first line gives the number of rooms as an integer. The remaining lines contain treasure amounts, as one double per line.



1	12.2
2	16.7
4	18.5
2	18.5
1	18.5
3	25.2
5	36.9

Figure 6: The output file contains multiple lines, each of the form of one integer followed by one double. The integer designates the room the searcher is in, and the double designates the accumulated treasure.

- Whether a room has been searched yet, or not, is a useful idea. Make a data structure to record this information. Don't forget to initialize this so that no room has been searched. If you include the outside world as a room, it has already been searched.
- Print the rooms that have been searched, and the treasure subtotals, as they are accumulated.
- The program must enable the user to specify two limits: an amount of treasure and a number of rooms. If the subtotal of treasure is greater than or equal to the limit, the searcher must exit the house without picking up any more treasure. If the number of rooms searched equals the limit, the searcher must exit the house without picking up any more treasure. One or both of these two values can be entered on the command line. If fewer than two entries appear on the command line, the program must query the user for these values.

Things to do:

1. Either:
 - (a) Make a C project from the Hello,World project.
 - (b) Populate that project with tests.c, tests.h, production.c and production .h.

or use the starter code.
2. Create the sequence diagram and include the electronic file (diagram, screenshot or photo). Make sure your name appears within the sequence diagram.
3. Place function prototypes for all of your functions from the sequence diagram into one or more .h files.
4. As you work on the assignment, collect a sequence of screen shots showing how your test code, and your production code are growing.
5. Be sure to build and run often; do not allow errors to build up.
6. Show the sequence of moves from room to room by listing them, and also show the subtotal of treasure through the house.
7. Receive the values entered on the command line. Ask for values not provided in the command line. In either case, check for reasonable values. Negative treasure or rooms implies no search should be conducted. Use these limit values in execution.

Grading

Criteria	Possible Points
Project that looks like starter code	20
Sequence diagram that reflects the problem statement	20
Documentation of code development (those screenshots) that clearly follows test-driven style	20
File of rooms searched, visited and treasure collected subtotals, that correspond	20
Correct treatment of input files and command line arguments	20
Total	100