

Chapter 10 Programming Assignment Documentation

Ashton Hellwig

May 10, 2020

Contents

1	Problem Analysis	2
1.1	Data	2
1.2	Desired Output	2
2	Algorithm	3
3	User Documentation	4
3.1	Build	4
3.1.1	With GNU Make	4
3.1.2	Bundled Release	4
	Works Consulted	6
A	Images	7
B	Unit Tests	8
B.1	Unit Test File	9

List of Algorithms

2.1	Chapter 10 Program Algorithm	3
-----	--	---

Listings

1	ashellwig_m5c10_programming_assignment output (stdout)	2
2	Chapter 10 Program Build Commands	4
3	00_CatchMain.cxx	9
4	01_TestBookType.cxx	9

1 Problem Analysis

The problem states:

This assignment relates to content from Chapter 10 of the eText.

Instructions

1. Review the general programming assignment instructions.
2. Write a program that:
 - A. Placeholder.

1.1 Data

- Placeholder.

1.2 Desired Output

```
1 echo "TODO!"
```

Listing 1: ashellwig_m5c10_programming_assignment output (stdout)

2 Algorithm

Below is the algorithm for the program.

Algorithm 2.1 Chapter 10 Program Algorithm

```
1: procedure MAIN
2:   return 0
3: end procedure
```

3 User Documentation

Please see Appendix A for images showing the compilation and running of the program.

3.1 Build

The following are instructions with two use cases:

- With GNU Make
- Bundled Release

3.1.1 With GNU Make

1. Navigate to the unzipped folder containing the project, **with a terminal emulator or command prompt**, this will (most likely) mean running:

```
1 cd ~/Downloads/ashellwig_m5c10_programming_assignment
```

2. Compile the program and documentation¹ using GNU automake after switching to the source directory:

```
1 # precondition: Must be located in project root
2 export INPUT_FILE="${PWD}/data/bookData.txt"
3 export OUTPUT_FILE="${PWD}/out/bin/output_data.txt"
4
5 make debug
6
7 ./out/bin/ashellwig_m5c10_programming_assignment.bin \
8   -f "${INPUT_FILE}" \
9   -o "${OUTPUT_FILE}"
10
11
12 cat "${OUTPUT_FILE}" # Verify contents are present
13
14 make clean-all      # Removes object files, binaries, and docs
```

Listing 2: Chapter 10 Program Build Commands

3.1.2 Bundled Release

1. Navigate to the unzipped folder containing the binary, **with a terminal emulator or command prompt**, this will (most likely) mean running:

```
1 cd ~/Downloads/ashellwig_m5c10_programming_assignment
```

2. To run the program simply issue this within the command prompt

```
1 export INPUT_FILE="${PWD}/data/Ch9Ex2/Ch9Ex2Data.txt"
2 export OUTPUT_FILE="${PWD}/data/Ch9Ex2OUT.txt"
3
4 ./out/bin/ashellwig_m5c10_programming_assignment.bin \
5   -f "${INPUT_FILE}" \
6   -d "${OUTPUT_FILE}"
7 cat "${OUTPUT_FILE}" # Verify contents are present
```

¹**Note:** This requires the whole `texlive` suite as well as `latexmk` to be installed.

Of course if preferred, you may also navigate to the build folder in file explorer and double click the executable (`./ashellwig_m5c10_programming_assignment`).

Works Consulted

Malik, D. S. (2015). *C programming: Program design including data structures* (7th ed.). Cengage Learning.

A Images

```
/usr/bin/g++ -c -std=gnu++2a -Wall -Wextra -ggdb -DDEBUG=1 -c src/chapter8.cxx -o out/obj/chapter8.o -Iinclude
/usr/bin/g++ -c -std=gnu++2a -Wall -Wextra -ggdb -DDEBUG=1 -c src/general_functions.cxx -o out/obj/general_functions.o -Iinclude
/usr/bin/g++ -c -std=gnu++2a -Wall -Wextra -ggdb -DDEBUG=1 -c src/main.cxx -o out/obj/main.o -Iinclude
/usr/bin/g++ \
  -std=gnu++2a -ggdb -DDEBUG=1 \
  -o out/bin/ashellwig_m4c8_programming_assignment.bin \
  out/obj/chapter8.o out/obj/general_functions.o out/obj/main.o \
  -Iinclude
```

Figure 1: Compiling Chapter 10's Program

```
Enter candidate's last name and the votes received by the candidate.
Smith 12345
Jones 4567
Adams 555
Washington 888888
Jefferson 456789
Candidate      Votes Received      % of Total Votes
Smith          12345                    0.91
Jones          4567                    0.34
Adams          555                    0.04
Washington      888888                    65.21
Jefferson      456789                    33.51
Total          1363144
The winner of the election is Washington.
Press enter to continue...
```

Figure 2: Running Chapter 10's Program

B Unit Tests

Tests were written with the [Catch2 library](#). The output is shown below.

Figure 3: Test Output

```
-----
Scenario: The obscure function is successful
  Given: The desired output's input string
  When: We attempt to obscure a string
  Then: The information should be obscured
-----
test/TestCase.cxx:12
.....

test/TestCase.cxx:13: PASSED:
  REQUIRE( obscureData(inputData) == targetString )
with expansion:
  "Jane Smith xxx-xx-xxxx S12345 xxxxxxxx"
  ==
  "Jane Smith xxx-xx-xxxx S12345 xxxxxxxx"

0.000 s:      Then: The information should be obscured
0.000 s:      When: We attempt to obscure a string
0.000 s:      Given: The desired output's input string
0.000 s: Scenario: The obscure function is successful
-----
Scenario: The obscure function is successful
  Given: The desired output's input string
  When: We attempt to obscure a string
  Then: The information should be obscured
-----
test/TestCase.cxx:21
.....

test/TestCase.cxx:22: PASSED:
  REQUIRE( obscureData(inputData) == targetString )
with expansion:
  "Ashton Hellwig xxx-xx-xxxx S02075840 xxxxxxxx"
  ==
  "Ashton Hellwig xxx-xx-xxxx S02075840 xxxxxxxx"

0.000 s:      Then: The information should be obscured
0.000 s:      When: We attempt to obscure a string
0.000 s:      Given: The desired output's input string
0.000 s: Scenario: The obscure function is successful
=====
All tests passed (2 assertions in 1 test case)
```


B.1 Unit Test File

Below I have included the code used to run the unit test for reference.

```

1 // 00_CatchMain.cxx
2
3 // In a Catch project with multiple files, dedicate one file to
4 // compile the
5 // source code of Catch itself and reuse the resulting object file
6 // for linking.
7
8 // Let Catch provide main():
9 #define CATCH_CONFIG_MAIN
10
11 #include "../include/catch2/catch.hh"
12
13 // That's it
14
15 // Compile implementation of Catch for use with files that do contain
16 // tests:
17 // - g++ -std=c++11 -Wall -I$(CATCH_SINGLE_INCLUDE) -c 000-CatchMain.
18 //   cpp
19 // - cl -EHsc -I%CATCH_SINGLE_INCLUDE% -c 000-CatchMain.cpp

```

Listing 3: 00_CatchMain.cxx

```

1 // main() provided in 00_CatchMain.cxx
2
3 #include "../include/book.hh" // TBD
4 #include "../include/catch2/catch.hh"
5 #include "../include/options.hh" // Argument parsing
6 #include <fstream> // std::fstream
7 #include <iostream> // std::cout
8 #include <string> // std::string
9
10 class DBConnection {
11 public:
12     static DBConnection createConnection(std::string const & /*dbName*/
13     ) {
14         return DBConnection();
15     }
16
17     bool executeSQL(std::string const & /*query*/, int const /*id*/,
18     std::string const & arg) {
19         if (arg.length() == 0) {
20             throw std::logic_error("empty SQL query argument");
21         }
22         return true; // ok
23     }
24 };
25
26 class UniqueTestsFixture {
27 protected:
28     UniqueTestsFixture() : conn(DBConnection::createConnection("myDB"))
29     {}
30
31     int getID() { return ++uniqueID; }

```

```

31 protected:
32     DBConnection conn;
33
34 private:
35     static int uniqueID;
36 };
37
38 int UniqueTestsFixture::uniqueID = 0;
39
40 TEST_CASE_METHOD( UniqueTestsFixture , "Create Employee/No Name", "[
    create]") {
41     REQUIRE_THROWS(conn.executeSQL(
42         "INSERT INTO employee (id, name) VALUES (?, ?)", getID(), ""));
43 }
44
45 TEST_CASE_METHOD( UniqueTestsFixture , "Create Employee/Normal", "[
    create]") {
46     REQUIRE(conn.executeSQL("INSERT INTO employee (id, name) VALUES (?,
47         ?)",
48         getID(), "Joe Bloggs"));
49 }
50 // Compile & run:
51 // - g++ -std=c++2a -Wall -Iinclude -isystem include/catch2 -isystem
52 // include/cxxopts -o out/bin/test.bin test/01_TestStudentClass.cxx
53 // 000-CatchMain.o && 110-Fix-ClassFixture
54 // --success
55 // - c1 -EHsc -I%CATCH.SINGLE.INCLUDE% -Iinclude 110-Fix-ClassFixture
56 // 000-CatchMain.obj && 110-Fix-ClassFixture --success

```

Listing 4: 01_TestBookType.cxx