CS 201 Homework 04

Bryan Beus

October 16, 2019

Source Code Link: `https://github.com/siddhartha-crypto/cs201/tree/master/hw4`

1 Design

1.1 Tokenizer

With Tokenizer, the instructions provide enough details to follow. I am adding in the additional special characters.

1.2 Database

For this project, am interested in creating a database that performs CRUD operations on a vector of vectors. I am using the Tao Te Ching library from `archive.org` as a simple data set to populate the database.

The focus on the CRUD operations will be to perform database tasks only on the lines within any one of the 81 chapters. No new chapters will be created, nor will any be deleted.

1.3 Bulls and Cows

The goal here is to keep the program simple. I am spending a lot of time on the other two programs, and so I would like to keep this program within a reasonable scope.

2 Post Mortem

2.1 Tokenizer

This was amazingly time consuming. I probably spent over 20 hours on this project alone.

One of the challenges I encountered was accessing vectors within vectors. I found that if I attempted to use the commands with which I was familiar, such as `.at()` or `.begin()`, I experienced frequent issues. Instead, I used the brackets more often, and this reduced error counts.

2.2 Database

Once again, this project was surprisingly time consuming. I also spent at least 10 hours on this project, if not more.

One of the challenges that I experienced was in finding the right combination of types to use while accessing or updating vectors. For example, the nuances between a `"` double quote and `'` single quote could sometimes result in twenty minutes of searching for an error before I could deduce the nature of the issue.

2.3 Bulls and Cows

The program logic on this challenge was not difficult. However, the logic involved with creating the proper response was actually a little bit challenging.

There are still some small instances where errors occur in the response given to the user. Specifically, when there is more than one of an integer in the initial pattern, in edge cases a cow can be reported to the user, even if the cow should have been cancelled. I could probably fix this error, if I had infinite time. But I am probably well over 30 hours of work into this week's homework project, and with the midterm exam tomorrow and work today, I am out of time.

3 Answers to Questions

- A container is a collection of objects. Common examples include string and vector
- `std::string` is a container because it is a collection of char-type objects
- `std::vector` is a container because it is a container of whatever type of object it is designed to collect
- A vector can also be called an array, list, or sequence
- To find the size of a vector, use `size_t` type object and the `.size()` method
- In my database program, I use the following method: `vector <vector <string> > taoTeChing (81);`. In the course materials, we have the `resize(INT, VAL)` method, that allows us to set a preset number of variables with a preset value in the container.
- Indices range from 0 to size - 1
- One can call a vector a template, depending on how we interpret the word, "template." A template can be anything that is a preset framework for application to other use cases. Classes and libraries can be templates, and vector is a library, and therefore it is also a template.

4 Sample Output

Listing 1: "Tokenizer"

```
Please type text. When you are done, type any
   variation of "End":
Program "helloworld"
Begin program
        i = 3 + 5
```

```
        Begin "program"
        /.,m
        $asdf
        asdf%
        "asdf"
End
[identifier]          "Program"
[string]              "\"helloworld\""
[whitespace]          ""
[identifier]          "Begin"
[identifier]          "program"
[whitespace]          ""
[identifier]          "i"
[other]               "="
[integer]             "3"
[other]               "+"
[integer]             "5"
[whitespace]          ""
[identifier]          "Begin"
[string]              "\"program\""
[whitespace]          ""
[unknown]             "/.,m"
[whitespace]          ""
[unknown]             "$asdf"
[whitespace]          ""
[other]               "asdf%"
[whitespace]          ""
[string]              "\"asdf\""
[whitespace]          ""
[identifier]          "End"
```

Listing 2: Database

```
====================================
Welcome to the Tao Te Ching library
====================================
```

```
This program allows the user to practice CRUD database
    techniques, with the Tao Te Ching as a reference

Please select an option from the following menu:
1) Select a chapter to read from the Tao Te Ching
2) Add a line to a specific chapter
3) Update a line from a specific chapter
4) Delete a line from a specific chapter
0) Exit program

>
```

Listing 3: Cows and Bulls

```
Enter four integers to guess the correct pattern

2150
0 bulls and 2cows


Enter four integers to guess the correct pattern

2199
1 bulls and 1cows


Enter four integers to guess the correct pattern

2188
0 bulls and 2cows
```

5 My Programs

5.1 Tokenizer

```
/**
 * tokenizerMain.cpp
```

```
 * CS 201
 * Bryan Beus
 * October 12, 2019
 * The main file for the tokenizer project
 */

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <sstream>

#include "tokenizer.hpp"

using std::cout;
using std::cin;
using std::endl;
using std::vector;
using std::string;
using std::noskipws;
using std::getline;
using std::istringstream;
using std::find;

int main(int argc, char **argv) {

    // Declare input vector<string> that will continually capture
    ↪  user input
    string input;

    // Declare vector<string> tokens that will collect all tokens
    ↪  from every input
    vector<string> tokens;

    // Declare bool for tracking user input of "end"
    bool isFinished;

    // Request user input
    cout << "Please type text. When you are done, type any
    ↪  variation of \"End\": " << endl;

    while (true) {

        // Capture user input
        ReadLine(input);

        // Iterate through input and place ordered responses into
        ↪  tokens container
        StringToTokensWS(input, tokens, isFinished);

        // Test whether there is a new token that is any
        ↪  combination of the string "end"
        // If so, break the endless while loop
```

```
        if (isFinished) {
            break;
        }
    }

    // Analyze tokens
    AnalyzeTokens(tokens);

    return 0;
}
```

5.2 Database

```
/**
 * tokenizerMain.cpp
 * CS 201
 * Bryan Beus
 * October 12, 2019
 * The main file for the tokenizer project
 */

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <sstream>

#include "tokenizer.hpp"

using std::cout;
using std::cin;
using std::endl;
using std::vector;
using std::string;
using std::noskipws;
using std::getline;
using std::istringstream;
using std::find;

int main(int argc, char **argv) {

    // Declare input vector<string> that will continually capture
    ↪  user input
    string input;

    // Declare vector<string> tokens that will collect all tokens
    ↪  from every input
    vector<string> tokens;

    // Declare bool for tracking user input of "end"
```

```
36      bool isFinished;
37
38      // Request user input
39      cout << "Please type text. When you are done, type any
        ↪  variation of \"End\": " << endl;
40
41      while (true) {
42
43          // Capture user input
44          ReadLine(input);
45
46          // Iterate through input and place ordered responses into
            ↪  tokens container
47          StringToTokensWS(input, tokens, isFinished);
48
49          // Test whether there is a new token that is any
            ↪  combination of the string "end"
50          // If so, break the endless while loop
51          if (isFinished) {
52              break;
53          }
54      }
55
56      // Analyze tokens
57      AnalyzeTokens(tokens);
58
59      return 0;
60  }
```

5.3 Bulls and Cows

```
1   /**
2    * bulls-and-cowsMain.cpp
3    * CS 201
4    * Bryan Beus
5    * October 15, 2019
6    * The main file for bulls-and-cows
7    */
8
9
10  #include <iostream>
11  #include <string>
12  #include <vector>
13  #include <algorithm>
14  #include <sstream>
15  #include <map>
16  #include <fstream>
17  #include <stdlib.h>
18
19  #include "bulls-and-cows.hpp"
```

```
20
21 using std::cout;
22 using std::cerr;
23 using std::cin;
24 using std::endl;
25 using std::vector;
26 using std::string;
27 using std::noskipws;
28 using std::getline;
29 using std::find;
30 using std::istringstream;
31 using std::stringstream;
32 using std::ifstream;
33 using std::rand;
34
35 int main(int argc, char **argv) {
36
37     // Clear the console
38     clearConsole();
39
40     // Initiate a string to hold the correct pattern
41     string pattern = "";
42
43     // Call the setPattern function to set the pattern
44     setPattern(pattern);
45
46     // Initiate endless while loop to repeat until user makes a
       ↪  correct guess
47     while (true) {
48
49         // Print the main user prompt
50         cout << endl;
51         cout << endl;
52         cout << "Enter four integers to guess the correct
           ↪  pattern" << endl;
53         cout << endl;
54
55         // Initiate the bulls and cows variables for this round
56         int bulls = 0;
57         int cows = 0;
58
59         // Collect user input
60         string userInput;
61         getline(cin, userInput);
62
63         // Ensure that user input is valid
64         bool isValid = testUserInput(userInput);
65
66         // If the input is not valid, restart loop
67         if (!cin || cin.fail() || !isValid) {
68             cin.clear();
69             cin.ignore(1000, '\n');
70             cout << "Please enter four integers: ";
```

```
            continue;
        }

        // Calculate the number of bulls and cows in the user input
        calculateRes(pattern, userInput, bulls, cows);

        // Report the number of bulls and cows
        cout << bulls << " bulls and " << cows << "cows" << endl;

        // If the user guessed correctly, end the program
        if (bulls == 4) {
            cout << "Congrats, you win!" << endl;
            waitForContinue();
            break;
        }
    }

  return 0;
}
```