CS 201 Homework 4

Solomon Himelbloom

October 12, 2020

- Repository Link: `https://github.com/techsolomon/cs201`
- Git Commits: `https://github.com/techsolomon/cs201/commits`
- This homework took approximately 8 hours to complete.

1 Design

This fifth homework assignment was designed with future reference and refactoring in mind. I also followed the directions of including the file name in the commit message in when code was added or deleted with functions broken down into much smaller parts than before. This made it simple to add references to vectors and other important functions.

The hw4 folder contains a main program (tokenizer), two additional programs (database and bull-and-cows), and a practice folder (for extra credit, such as fifo). Header files were created to allow for linking multiple files and removing repetition for defining each function. Once again, each .cpp file starts with the author's name, the computer science class code, date file was created or last edited, and finally a short description with an included Makefile.

2 Post Mortem

During this homework assignment, I learned more about vectors and the fundamentals of extrapolating information from data. For style and design, I enjoyed using detailed comments and descriptive functions to better organize and reference programs on the next assignments.

Some changes that I hope to add for the next homework assignment include reviewing textbook, midterm, and lab submissions before diving into the larger programs. In conclusion, I also hope to ask for help when clarifications are needed, in addition to referencing online C++ guides, before submitting my final draft of the project.

3 Answers to Questions

1. A container holds objects and stores collections or series of other objects in a format easily read by the compiler.

2. The std::string type is considered a container as it can hold text, such as alphanumeric characters.

3. The std::vector type is considered a container as, similar to arrays, std::vector can hold data, such as dictionaries of names or keys. It is also worth mentioning that the vector storage is handled automatically with expansion and contraction dealt with accordingly.

4. Three different synonyms on similar data structures to std::vector include "array," "list," or "sequence."

5. The method of .size() gives us the size of a container. This method returns with a type of integer.

6. We can use the resize to set the size of a container. Yes, you can set a default value for the respective container in C++.

7. Computer programmers tend to start counters, arrays, and other integer initialization values at zero – makes sense that indices have a range starting at zero instead of one.

8. It is appropriate to use the keyword auto when you are dealing with finding keys within vectors. Make sure that you are on the desired version of C++ (11 or 14) to avoid any unnecessary warnings or error messages.

9. Yes, std::vector can be considered a template. I believe that this means that we do not have to define its respective constraints each time it is called. This is because information about how vectors are defined is hard coded in the C++ Standard Library (STL). We can conclude that a vector is a reusable template as it can be called with a stand-alone include statement for memory optimization purposes.

4 Program 1

4.1 Sample Output/Screenshot

Listing 1: Sample Program Output

```
Please type in some text. When you are done, type
      End    ,     end     or     END    : Program
   helloworld
Begin
Print "Hello"
I = 3 + 5
End

TOKENIZER OUTPUT:

End

[integer]
[identifer]
[string]
```

```
[whitespace]
[special_character]
[unknown]
```

4.2 Git Commit Messages

Date	Message
2020-10-12	add: TokenOutput in tokenizer.cpp
2020-10-12	add: do/while to tokenizertest.cpp
2020-10-12	add: searchToken if/else and tokenSearchCompletion
2020-10-09	Implemented AnalyzeTokens() in tokenizer.cpp
2020-10-09	Implemented StringToTokenWS() in tokenizer.cpp
2020-10-09	Implemented ReadLine() in tokenizer.cpp
2020-10-09	add: tokenizer.cpp/hpp
tokenizertest.cpp template	
2020-10-04	add: hw4 (initial files)

4.3 Source Code

4.4 tokenizertest.cpp

```cpp
// tokenizertest.cpp
// Solomon Himelbloom
// 7 October 2020
// Tokenizer example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <vector>

#include "tokenizer.hpp"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

int main() {
    std::string str;
    std::string exit_code;
    std::string user_input = "";
    std::vector<std::string> tokens;

    cout << "Please type in some text. When you are done, type \End", \end" or \END": ";

    do {
        cin >> exit_code;

        if (ReadLine(str)) {
            StringToTokenWS(str, tokens);
        }

    } while (exit_code != "End" && exit_code != "end" && exit_code != "END");

    cout << "\n" << "TOKENIZER OUTPUT:"
    << "\n" << "\n" << exit_code << "\n" << endl;

    TokenOutput();
    AnalyzeTokens(tokens);

    return 0;
}
```

4.5 tokenizer.cpp

```cpp
// tokenizer.cpp
// Solomon Himelbloom
// 7 October 2020
// Tokenizer [source file] example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <vector>
#include <sstream>
#include <cctype>
```

```cpp

#include "tokenizer.hpp"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;
using std::istringstream;

// Input a line of text from the user & check contents.
bool ReadLine(std::string& str) {
    std::getline(std::cin, str);
    if (str == "" || std::cin) {
        return false;
    }
    else {
        return true;
    }
}

// Read strings separated by whitespace characters.
unsigned StringToTokenWS(const std::string &input, std::vector<std::string> &tokens) {
    std::istringstream();
    std::vector<std::string> tokenSearchCompletion;
    for (int i = 0; i < input.size(); i++) {
        tokens.push_back(tokenSearchCompletion[i]);
    }
    tokens.push_back("\n");
    return tokens.size();
}

// Determine type of token (integer/identifier/string/whitespace/character/unknown).
void AnalyzeTokens(const std::vector<std::string> &tokens) {
    for (int i = 0; i < tokens.size(); i++) {
        std::string searchToken = tokens[i];
        if (std::isdigit(searchToken.at(0)) != 0) {
            cout << "[integer] " << endl;
        }
        else if (std::isdigit(searchToken.at(0)) != 0 ||
        std::isalpha(searchToken.at(0)) != 0) {
            cout << "[identifer] " << endl;
        }
        else if (searchToken == "\"") {
            cout << "[string] " << endl;
        }
        else if (std::isspace(searchToken.at(0))) {
            cout << "[whitespace] " << endl;
        }
        else if (searchToken == "+" || searchToken == "-" || searchToken == "*" ||
        searchToken == "/" || searchToken == "=" || searchToken == "%") {
            cout << "[special_character] " << endl;
        }
        else {
            cout << "[unknown] " << endl;
        }
    }
}

// Function for testing tokens and printing output.
void TokenOutput() {
    std::string integer_token = "[integer]";
    std::string identifer_token = "[identifer]";
    std::string string_token = "[string]";
    std::string whitespace_token = "[whitespace]";
    std::string special_character_token = "[special_character]";
    std::string unknown_token = "[unknown]";

    cout << integer_token << "\n" << identifer_token << "\n"
    << string_token << "\n" << whitespace_token << "\n"
    << special_character_token << "\n" << unknown_token << endl;
}
```

4.6 tokenizer.hpp

```
// tokenizer.hpp
// Solomon Himelbloom
// 7 October 2020
// Tokenizer [header file] example for CS 201.

#ifndef TOKENIZER_HPP_
#define TOKENIZER_HPP_

#include <string>

void TokenOutput();

bool ReadLine(std::string& str);

unsigned StringToTokenWS(const std::string &input, std::vector<std::string> &tokens);

void AnalyzeTokens(const std::vector<std::string> &tokens);

#endif /* TOKENIZER_HPP_ */
```

5 Program 2

5.1 Sample Output/Screenshot

Listing 2: Sample Program Output

```
KEY: 1
VALUE: 10
KEY/VALUE PAIR STATUS: 110
```

5.2 Git Commit Messages

Date	Message
2020-10-12	add: key/value pairs to the database
2020-10-12	refactor: database to allow new records
2020-10-09	update: database import (map)
2020-10-09	add: ReadRecord to database.cpp
2020-10-09	add: UpdateRecord in database.cpp
rm: MyDatabaseRecord	
2020-10-09	add: bool logic to database.hpp
2020-10-09	add: MyDatabaseRecord struct in database.hpp
2020-10-09	add: crud.cpp and database.cpp template
2020-10-04	add: hw4 (initial files)

5.3 Source Code

5.4 crud.cpp

```cpp
// crud.cpp
// Solomon Himelbloom
// 9 October 2020
// Database [crud source file] example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <map>

#include "database.hpp"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

int main() {
    std::string key = "1";
    std::string value = "10";

    cout << "KEY: " << key << endl;
    cout << "VALUE: " << value << endl;
    cout << "KEY/VALUE PAIR STATUS: " << key << value << endl;
}
```

5.5 database.cpp

```cpp
// database.cpp
// Solomon Himelbloom
// 7 October 2020
// Database example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <map>

#include "database.hpp"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

std::map<std::string, MyDatabaseRecord> theDatabase;


// Creates a new record within the database with a corresponding key value.
// bool CreateRecord(const std::string &key) {
//     auto it = theDatabase.find(key);
//     return true;
// }

// ReadRecord(key, record) copies the information from the database to
// a user supplied record
// @param {string} key
// @param {MyDatabaseRecord} record
// @returns false if the record does not exist
bool ReadRecord(const std::string &key, MyDatabaseRecord &record) {
    auto it = theDatabase.find(key);
    if (it == theDatabase.end()) {
        return false;
    }
    // return = it- >second;
    return true;
}

// UpdateRecord(key, record) sets the databae to the new value
// @param {string} key
// @param {MyDatabaseRecord} record
// @return true if operation successful
bool UpdateRecord(const std::string &key, const MyDatabaseRecord &record) {
    auto it = theDatabase.find(key);
    if (it == theDatabase.end()) {
        return false;
    }
    theDatabase[key] = record;
    return true;
}

// Deletes a record from the database given a key.
// bool DeleteRecord(const std::string &key) {
//     auto it = theDatabase.find(key);
// }

// Inputs a record into the database given a record.
// bool InputRecord(MyDatabaseRecord &record) {
//     auto it = theDatabase.find(key);
//     return true;
// }

// Prints a record from the database given a key.
// bool PrintRecord(const std::string &key) {
//     auto it = theDatabase.find(key);
//     cout << "Key: " << endl;
//     return true;
// }
```

5.6 database.hpp

```
// database.hpp
// Solomon Himelbloom
// 9 October 2020
// Database [header file] example for CS 201.

#ifndef DATABASE_HPP_
#define DATABASE_HPP_

#include <string>
#include <map>
```

```cpp
struct MyDatabaseRecord {
    // Replace this with information related to your database.
    // std::string objectName{"obj"};
    // std::string materialName{"mtl"};
    // std::string diffuseColor{"diff"};
    // std::string specularColor{"spec"};
    // bool twoSided{false};

    std::string key;
    std::string value;
};

bool CreateRecord(const std::string &key);
bool ReadRecord(const std::string &key, MyDatabaseRecord &record);
bool UpdateRecord(const std::string &key, const MyDatabaseRecord &record);
bool DeleteRecord(const std::string &key);
bool InputRecord(MyDatabaseRecord &record);
bool PrintRecord(const std::string &key);

#endif /* DATABASE_HPP_ */
```

6 Program 3

6.1 Sample Output/Screenshot

Listing 3: Sample Program Output

```
Enter a four digit number (no repeats & negative to
   quit): 1234
numberVector size: 4
guessVector size: 0
0 0 0 0
1 bull and 1 cows
```

6.2 Git Commit Messages

Date	Message
2020-10-12	refactor: database to allow new records
2020-10-12	add: if/else grammar logic in bulls-and-cows.cpp
2020-10-12	refactor: createNumberVector in bulls-and-cows.cpp
2020-10-09	add: if/else to bulls-and-cows.cpp
2020-10-09	add: randomNumber and if/else to bulls-and-cows.cpp
2020-10-09	add: guessingGame function to bulls-and-cows.cpp
2020-10-04	add: hw4 (initial files)

6.3 Source Code

```cpp
// bulls-and-cows.cpp
// Solomon Himelbloom
// 7 October 2020
// Bulls and cows game example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <vector>

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

void randomNumber() {
    int random_number = 1357;
    cout << "Random integer: " << random_number << endl;

    std::vector<int> createNumberVector { 1, 3, 5, 7 };

    for (int& v : createNumberVector) {
        std::cout << v << "";
    }

    cout << endl;
}

void guessingGame() {
    int bulls = 1;
    int cows = 1;

    vector<int> numberVector(4);
    vector<char> guessVector(0);
    cout << "numberVector size: " << numberVector.size() << endl;
    cout << "guessVector size: " << guessVector.size() << endl;

    for (auto i = 0; i < numberVector.size(); ++i) {
        cout << numberVector[i] << " ";
    }

    cout << endl;

    if (bulls != 1 && cows != 1) {
        cout << bulls << " bulls and " << cows << " cows." << endl;
    }

    else if (bulls != 1 && cows == 1) {
        cout << bulls << " bulls and " << cows << " cow." << endl;
    }

    else if (bulls == 1 && cows == 1) {
        cout << bulls << " bull and " << cows << " cows" << endl;
    }

    else {
        cout << bulls << " bull and " << cows << " cow" << endl;
    }

}

int main(int argc, char *argv[]) {
    int user_input = 0;
    int random_number = 1357;

    cout << "Enter a four digit number (no repeats & negative to quit): ";
    cin >> user_input;

    if (user_input == random_number) {
        randomNumber();
    }

    else if (user_input > 9999) {
        cout << "Input must be 4 digits in length (less than 10,000)." << endl;
    }

    else if (0 < user_input && user_input < 1000) {
        cout << "Input must be 4 digits in length (greater than 999)." << endl;
    }

    else if (user_input < 0) {
        randomNumber();
    }

    else {
        guessingGame();
    }

    return 0;
}
```

7 Program 4

7.1 Sample Output/Screenshot

Listing 4: Sample Program Output

```
Hello, FIFO/LIFO.
```

7.2 Git Commit Messages

Date	Message
2020-10-09	update: fifo-lifo.cpp comments
2020-10-09	add: Fifo, Lifo, Push, Pop function templates
2020-10-09	add: fifo-lifo.cpp template file
2020-10-04	add: hw4 (initial files)

7.3 Source Code

```cpp
// fifo-lifo.cpp
// Solomon Himelbloom
// 9 October 2020
// FIFO/LIFO example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <vector>

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

// First-in First-Out
void FifoPush(vector<string> &container, const string &item);
void FifoPop(vector<string> &container, string &item);

// Last-In First-Out
void LifoPush(vector<string> &container, const string &item);
void LifoPop(vector <string> &container, string &item);

// Shared functionality
bool IsContainerEmpty(const vector<string> &container);
void PrintContainer(const vector<string> &container);

// Implement these two tests to verify your functions work
// with at least the sequence:
//       push "A", push "B", push "C", push "D"
//       push    , pop     , pop     , pop
bool TestFifo();
bool TestLifo();

int main() {
    cout << "Hello, FIFO/LIFO." << endl;
}
```