CS 201 Homework 4
Containers

Sarah Carter

October 16, 2019

Source Code Link: `https://github.com/SarahFDAK/CS201.git`
This homework took approximately XX hours to complete.

1 Design

In the tokenizer program, I set it up to ask for the user line input in main, and also had the getline in a loop in main. The condition for the loop was that the user had not input ”end”, ”End”, or ”END” yet, so as long as that was true, it took their input and passed it to StringsToTokensWS, which separated each element of the input and put them into the tokens vector. After the user enters the termination word, it prints the number of elements in the string, and then, using if statements to test for type in AnalyzeTokens, prints each token and its type.

2 Post Mortem

This started out as a pretty confusing and challenging program for me. I struggled with the syntax of getting StringToTokensWS to actually work correctly. I really had trouble with the condition for the while statement to successfully loop through the user input and get all the elements correctly pushed to the vector. I

think I have a lot more to learn to clarify some of the technicalities of vectors and their methods in my mind.

3 Answers to Questions

In this section, you will write the answers to the questions in the homework assignment.

- A container is a data structure that holds a collection of items, usually of the same type.
- A string is a container for chars. These can be letters, numbers, or really any other symbol that has an ASCII number.
- A vector is a container that can hold however many elements, all of the same type.
- Three synonyms for vector are array, list, and sequence.
- The method ”size” gives us the size of a vector; it returns an int.
- The method ”resize” sets the size of a vector. You can also set the default value. (e.g. vector.resize(5, ”hi”);)
- Indices range from 0 to size()-1.
- The keyword ”auto” is useful when a type has a long and complicated name. As long as it’s initialized properly, auto will work.
- Vector is just a class template, meaning that it is essentially an empty container waiting for the programmer to declare what type of data it will hold.

4 Sample Output

Listing 1: Sample Program Output

```
Please enter a string - it can contain spaces and
   numbers. Type "end" or "End" or "END" to finish
   your input:
Program helloworld
Begin
        Print "Hello"
        I = 3 + 5
End
15
[identifier]    "Program"
[identifier]    "helloworld"
[whitespace]    ""
[identifier]    "Begin"
[whitespace]    ""
[identifier]    "Print"
[string]        "\"Hello"\"
[whitespace]    ""
[identifier]    "I"
[unknown]       "="
[integer]       "3"
[unknown]       "+"
[integer]       "5"
[whitespace]    ""
[identifier]    "End"
Program ended with exit code: 0
```

5 My Program

```
#include <iostream>
#include <vector>
#include "tokenizer.hpp"

int main(int argc, const char * argv[]) {
    std::string str;
    std::vector<std::string> tokens;

    std::cout << "Please enter a string - it can contain spaces and numbers. Type \"
```

```
    //As long as the user hasn't entered any of the termination strings, the input i
    while (Finish(str)) {
        std::getline(std::cin, str);
        StringToTokensWS(str, tokens);
    }

    //Prints the number of elements in the vector once the user has completed entry.
    std::cout << tokens.size() << std::endl;

    //Lists all the elements and their types.
    AnalyzeTokens(tokens);
    return 0;
}
```

5.1 Tokenizer Header

```
//
//  tokenizer.hpp
//  hw04
//
//  Created by Sarah Carter on 10/13/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#ifndef tokenizer_hpp
#define tokenizer_hpp

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


//Declare function prototypes
bool ReadLine(std::string & str);
unsigned StringToTokensWS(std::string & str, std::vector<std::string> & tokens);
void AnalyzeTokens(const std::vector<std::string> & tokens);
bool Finish(const std::string & str);

#endif /* tokenizer_hpp */
```

5.2 Tokenizer Source

```
//
//  tokenizer.cpp
//  hw04
//
//  Created by Sarah Carter on 10/13/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//
```

```cpp

#include "tokenizer.hpp"
#include <vector>
#include <iostream>
#include <string>
#include <sstream>

//This function checks to see if the user entered a blank line. If so, it returns fa
bool ReadLine(std::string & str){

    if(str == ""){
        return false;
    }

    return true;
}

//This function takes the strings from user input that are separated by whitespace a
unsigned StringToTokensWS(std::string & str, std::vector<std::string> & tokens){

    std::istringstream token(str);
    std::string text;

    while(token >> text) {
        tokens.push_back(text);
    }
    if(Finish(str)){
        tokens.push_back("\n");
    }

    return int(tokens.size());
}

//This function analyzes the tokens that were placed in the vector by StringToTokens
void AnalyzeTokens(const std::vector<std::string> & tokens){
    for(auto token: tokens){
        if(token[0] <= '9' && token[0] >='0'){
            std::cout << "[integer]" << '\t' << "\"" << token << "\"" << std::endl;
        }
        else if(token[0] == '"' && token[token.size()-1] == '"'){
            std::cout << "[string]" << '\t' << "\"\\" << token << "\\""\"" << std::e
        }
        else if(token[0] == '_' || (token[0] >='A' && token[0] <= 'z')){
            std::cout << "[identifier]" << '\t' << "\"" << token << "\"" << std::end
        }
        else if(token == "\n"){
            std::cout << "[whitespace]" << '\t' << "\"" "\"" << std::endl;
        }
        else {
            std::cout << "[unknown]" << '\t' << "\"" << token << "\"" <<std::endl;
        }
    }
}

```

```cpp
//This function checks if the user has entered the program termination string.
bool Finish(const std::string & str){
    if(str == "end" || str == "End" || str == "END"){
        return false;
    }

    return true;
}
```

5.3 Bulls-And-Cows Source

```cpp
//
//  bulls-and-cows.cpp
//  Sarah Carter
//  10/15/19
//  This program requests four digits from the user and reports back (in the form of
//

#include <iostream>
#include "guesscheck.hpp"

int main(int argc, const char * argv[]) {
    //Set answer and declare guess string
    std::string answer = "5740";
    std::string guess;
    //Loop through guesses until there are 4 correct
    do {
        std::cout << "Guess what 4 digits are the answer. 4 bulls means you've guess

        std::cin >> guess;
        //Checks if guess is 4 characters long and if characters are integers
        if(!CheckGuessLength(guess) || !CheckGuessType(guess)){
            std::cin.clear();
            std::cout << "Please enter 4 integers." << std::endl;
            std::cin >> guess;
        }
        //Reports not totally correct answer status. CountCows function counts total
        std::cout << "Your guess has " << CountBulls(guess, answer) << " bulls and "

    } while(CountBulls(guess, answer) < answer.size());
    //Prints correct guess
    std::cout << guess << " is correct!" << std::endl;
    return 0;
}
```

5.4 GuessCheck Header

```
//
//  guesscheck.hpp
//  bulls-and-cows
//
//  Created by Sarah Carter on 10/15/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#ifndef guesscheck_hpp
#define guesscheck_hpp

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

//Declare all function prototypes
bool CheckGuessLength(const std::string & guess);
bool CheckGuessType(const std::string & guess);
int CountBulls(const std::string & guess, const std::string & answer);
int CountCows(const std::string & guess, const std::string & answer);

#endif /* guesscheck_hpp */
```

5.5 GuessCheck Source

```
//
//  guesscheck.cpp
//  bulls-and-cows
//
//  Created by Sarah Carter on 10/15/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#include "guesscheck.hpp"
#include <iostream>

//Check if guess string is 4 characters long
bool CheckGuessLength(const std::string & guess){
    if(guess.size() != 4){
        return false;
    }
    return true;
}

//Check to make sure guess is all integers
bool CheckGuessType(const std::string & guess){
    for(int i = 0; i < guess.size(); i++){
        if(guess[i] < '0' || guess[i] > '9'){
            return false;
```

```
        }
    }
    return true;
}

//Count the exact correct guesses
int CountBulls(const std::string & guess, const std::string & answer){
    int bcount = 0;

    for(int b = 0; b < answer.size(); b++){

        if(guess[b] == answer[b]){
            bcount++;
        }
    }
    return bcount;
}

//Count the correct number, wrong placement guesses
int CountCows(const std::string & guess, const std::string & answer){
    int ccount = 0;

    for(int c = 0; c < answer.size(); c++){

        for(char g: guess){

            if(g == answer[c]){
                ccount++;
            }
        }
    }
    return ccount;
}
```

5.6 FIFO-LIFO Source

```
//
//  main.cpp
//  fifo-lifo
//
//  Created by Sarah Carter on 10/15/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#include <iostream>
#include <vector>
#include "fofo.hpp"

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

```

int main(int argc, const char * argv[]) {
    vector<string> container;
    string item;

    cout << "Enter items or data into inventory. Enter \"stop\" to exit: " << endl;

    while(std::cin >> item) {
        if(item == "stop"){
            break;
        }
        FifoPush(container, item);
    }

    PrintContainer(container);
    FifoPop(container, item);

    PrintContainer(container);

    return 0;
}
```

5.7 Fofo Header

```
//
//  fofo.hpp
//  fifo-lifo
//
//  Created by Sarah Carter on 10/15/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#ifndef fofo_hpp
#define fofo_hpp

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

void FifoPush(std::vector<std::string> & container, const std::string & item);
void FifoPop(std::vector<std::string> & container, const std::string & item);

void LifoPush(std::vector<std::string> & container, const std::string & item);
void LifoPop(std::vector<std::string> & container, const std::string & item);

bool IsContainerEmpty(const std::vector<std::string> & container);
void PrintContainer(const std::vector<std::string> & container);

bool TestFifo();
bool TestLifo();

#endif /* fofo_hpp */
```

5.8 Fofo Source

```cpp
//
//  fofo.cpp
//  fifo-lifo
//
//  Created by Sarah Carter on 10/15/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#include "fofo.hpp"
#include <iostream>
#include <vector>

using std::vector;
using std::string;
using std::cout;


void FifoPush(vector<string> & container, const string & item){
    container.insert(container.begin(), item);
}


void FifoPop(vector<string> & container, const string & item){
    container.pop_back();
}


void LifoPush(vector<string> & container, const string & item){
    container.push_back(item);
}


bool IsContainerEmpty(const vector<string> & container){
    if(container.size()==0){
        return true;
    }
    return false;
}


void PrintContainer(const vector<string> & container){
    cout << "The entered items are: " << std::endl;

    for(string item: container){
        cout<< item << std::endl;
    }
}
```