CS 201 Homework 08

Bryan Beus

December 9, 2019

Source Code Link: `https://github.com/siddhartha-crypto/cs201/tree/master/hw8`

1 Design

1.1 Thermostat

I do not want to create any kind of ASCII GUI for either of the AI based projects. HTML5 is much more efficient for this type of work, and without any kind of ASCII animation library, it's just not worth the time. I intend to print out everything in a simple text-based response in the console, and to clear the screen for each askOwner() function.

1.2 Vacuum

As before, I do not want to create a complex animation for this project. I'll follow the simple instructions, choose up to 8 rooms, and let the vacuum roam on a set course to keep things simple.

1.3 TF

I am interested in AI and data-science regarding language. This project is intriguing. However, with the remaining time for this homework assignment (the other two took a long time before I began this project), I will refrain from attempting the full (optional) version of this project unless I finish early.

2 Post Mortem

2.1 Thermostat

The largest challenge of this project was to create the file and class structure. I spent several hours working on the structure of the functions, classes, and header files. Once those were in place, debugging my initial code took less than 5 minutes.

2.2 Vacuum

This project was very similar to the previous project. I found that it was not difficult. To make things more interesting, I added random variables and used pointers to pass the random generator around. Solving this problem took several hours, and I feel I have a better understanding of both topics now.

2.3 TF

Setting up the initial template for the project took most of my time, but debugging for this project was more challenging. I struggled with the `itr->second` code for a long time until I received help from Dr. Metzgar.

3 Answers to Questions

- A class is a template for objects. The class gives ranges of values and functions for a potential object. The object is an instance of a class.
- `x.foo()`

- Items that are `private` can only be accessed by other members within the class. Items that are public can be accessed by objects that do not belong to this instance of this class.
- A const member function does not have permission to modify the current object
- The `const` instance goes after the parentheses of the member function
- A constructor is a set of instructions that create an instance of the class; this allows a programmer to initiate the class with default settings, or settings based on parameters passed to it, etc.
- The same name as the class
- A constructor that takes no arguments
- Overloading is when we give a function multiple ways of behaving, depending on the types and quantity of parameters passed to it
- We can have multiple constructors that initiate the instance of the class differently
- To define a member function outside of the class, we start with the name of the class followed by two colons, `Name::member_function()`
- One way to create a class-wide solution is to use static members
- A static data member is a definition that is persistent throughout all instances of the class
- Similarly, a static function member is also independent of all instances of the class
- Inside the class: `static int data`, outside the class: `int Name::data`

4 Sample Output

4.1 Thermostat

```
The current temperature is: 54
The heater is: On

Please indicate the desired temperature (celsius).
 -- To stop the program, enter a value less than absolute zero
  ↪  (-274 or below))
155
```

4.2 Vacuum

```
Current status of each room:
Room 1: clean
Room 2: clean
Room 3: clean
Room 4: clean
Room 5: clean
Room 6: clean
Room 7: clean
Room 8: clean

Vacuum status:
Room: 2
Action: Move
Press enter to continue...
```

4.3 TF

```
 1:  grail                                 occurances:  290
 2:  king                                  occurances:  155
 3:  ritual                                occurances:  120
 4:  form                                  occurances:  117
 5:  evidence                              occurances:  116
 6:  life                                  occurances:  110
 7:  story                                 occurances:  100
 8:  character                             occurances:  93
 9:  fact                                  occurances:  88
10:  certain                               occurances:  87
11:  thus                                  occurances:  86
12:  origin                                occurances:  85
13:  nature                                occurances:  84
```

```
14:  find                                  occurances:  81
15:  land                                  occurances:  79
16:  tradition                             occurances:  79
17:  perceval                              occurances:  78
18:  found                                 occurances:  78
19:  between                               occurances:  77
20:  gawain                                occurances:  77
```

5 My Programs

5.1 Thermostat

```cpp
/*
 * main.cpp
 * CS 201
 * December 6, 2019
 * Bryan Beus
 * Main file for themostat main project in hw8
 */

#include <iostream>
#include <iomanip>
#include <vector>
#include <string>

#include "Environment.hpp"
#include "Agent.hpp"
#include "Simulator.hpp"
#include "Miscellaneous.hpp"

using std::vector;
using std::string;
using std::cout;
using std::cin;
using std::endl;
using std::getline;
using std::istringstream;
using std::setw;
using std::left;

int main() {

    // Inform user of the nature of the software
    clearConsole();
    cout << "Welcome to the Temperature Simulator" << endl;
    cout << "\nThe simulator will now create an environment" <<
    ↪   endl;
    waitForContinue();

    // Create initial environment, iteration, and quit vars
    Environment env;
    Agent agt;
    Simulator sim;
    bool calibrated = false;
    int iter = 0;
    bool isFinished = false;

    // Clear screen and print introduction to console
    clearConsole();
    cout << "Environment created" << endl;
    waitForContinue();

    // Initiate while loop
    while (!isFinished) {

        // Clear screen
        clearConsole();

        // Affect the environment
        env.iteration();

        // The agent performs its duties
        agt.perceive(env);
        agt.think(calibrated);
        agt.act(env);

        // If iteration is divisible by 10, Agent requests user
        ↪  input
        // Test whether user wants to quit
        if (iter % 10 == 0) {
            sim.askOwner(isFinished, agt, env);
        }

        if (isFinished)
            break;

        // Increase iteration count
        iter++;
    }

    return 0;
}
```

5.2 Vacuum

```
/*
 * main.cpp
 * CS 201
 * December 7, 2019
 * Bryan Beus
 * Main file for vacuum project in hw8
 */

#include <iostream>
#include <iomanip>
#include <random>
#include <cmath>
#include <stdlib.h>

#include "Environment.hpp"
#include "Agent.hpp"
#include "Simulator.hpp"
#include "Miscellaneous.hpp"

using std::cout;
using std::cin;
using std::endl;
using std::random_device;
using std::seed_seq;
using std::mt19937;

int main() {

  // Create pseudo-random device
  random_device r;
  seed_seq seedObj{r(), r(), r(), r(), r(), r(), r(), r()};
  mt19937 e1(seedObj);

    // Inform user of the nature of the software
    clearConsole();
    cout << "Welcome to the Vacuum Simulator" << endl;
    cout << "\nThe simulator will now create an environment" <<
     ↪  endl;
    waitForContinue();

    // Create initial environment, iteration, and quit vars
    clearConsole();
    Environment env(e1);
    Agent agt;
    Simulator sim;
    cout << "Environment created" << endl;
    waitForContinue();

    // Initiate while loop
    while (true) {

        // Clear screen
        clearConsole();

        // Affect the environment
        env.iteration();

        // The agent performs its duties
        agt.perceive(env);
        agt.think();
        agt.act(env);

        printState(env, agt);

        sim.askOwner();
    }

    return 0;
}
```

5.3 TF

```
/*
 * main.cpp
 * CS 201
 * December 8, 2019
 * Bryan Beus
 * The main file for the tf project of hw8
 */

#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <iomanip>
#include <utility>
#include <stdio.h>
#include <ctype.h>
#include <algorithm>
#include <iterator>

using std::vector;
using std::pair;
using std::string;
using std::cout;
using std::endl;
using std::ofstream;
using std::setw;
using std::make_pair;
using std::ifstream;
using std::sort;
using std::left;
using std::right;
using std::find_if;

```

```
bool readFile(string& filename, ifstream& ifs);
bool parseBook(string& filename, vector< pair<string, int> >&
    ↪  vec);
bool parseWordList(string& filename, vector<string >& word_list);
bool sortBook(vector< pair<string, int> >& vec);
bool loadStopWords(vector<string>& stop_words);
bool filterRes(vector< pair<string, int> >& vec, vector<string>&
    ↪  stop_words);
bool printRes(vector< pair<string, int> >& vec);

int main() {

    // Set initial filename
    string filename = "from_ritual_to_romance_jessie_weston.txt";

    // Create initial vectors for holding data
    vector< pair<string, int> > vec;
    vector<string> stop_words;

    // Create res for software kill switch, if anything fails
    bool res;

    // Import and parse the chosen book
    res = parseBook(filename, vec);
    if (!res)
        return 0;

    // Sort the book's values
    res = sortBook(vec);
    if (!res)
        return 0;

    // Load the stop words
    res = loadStopWords(stop_words);
    if (!res)
        return 0;

    // Filter the book based on the stop words
    res = filterRes(vec, stop_words);
    if (!res)
        return 0;

    // Print the result
    res = printRes(vec);

  return 0;
}

// Parse the book into a vector that holds each word and its count
bool parseBook(string& filename, vector< pair<string, int> >&
    ↪  vec) {

    ifstream file;
    file.open(filename);

    if (!file.is_open()) {
        cout << "Error parsing book" << endl;
        return false;
    }

    string s1;
  while (file >> s1) {

        // To keep words that end with a period or comma, truncate
        ↪  these word's
        // strings
        if (s1.back() == '.' || s1.back() == ',') {
            s1 = s1.substr(0, s1.length() - 1);
        }

        // Make all letters lowercase
        std::for_each(s1.begin(), s1.end(), [](char& c) {
            c = std::tolower(c);
        });

        // Ensure that we have a regular word, and not a special
        ↪  character
        // value
        if (s1.find_first_not_of("abcdefghijklmnopqrstuvwxyz") !=
        ↪  std::string::npos) {
            continue;
        }

        // Search the vec vector to see if this s1 word has
        ↪  already occured
        auto it = find_if( vec.begin(), vec.end(), [&s1](const
        ↪  pair<string, int>& element) {
           return element.first == s1;
        });

        // If it has not occurred, add it to the vector and set
        ↪  the initial
        // value
        if ( it == vec.end() ) {
            vec.push_back(make_pair(s1, 1));

        // Otherwise, increase the iteration->second value for
        ↪  the discovered
        // vector/pair value
        } else {
            it->second++;
        }
  }
    return true;
}
```

```cpp

// Sort the book with highest occuring values towards the front
bool sortBook(vector< pair<string, int> >& vec) {
    sort(vec.begin(), vec.end(), [](const pair<string, int>& a, const pair<string, int>& b) {
            return (a.second > b.second);
    });
    return true;
}

// Load the list of stop words to avoid
bool loadStopWords(vector<string>& stop_words) {
    string filename = "stop_word_list.txt";
    ifstream file(filename);

    if (!file) {
        return false;
    }

    string s1;
    while (file >> s1) {
        stop_words.push_back(s1);
    }

    return true;
}

// Filter the resulting vec vector by the stop_words vector
bool filterRes(vector< pair<string, int> >& vec, vector<string>& stop_words) {

    for (size_t i = 0; i < stop_words.size(); i++) {
        string currStop = stop_words[i];
        auto it = find_if(vec.begin(), vec.end(), [&currStop](const pair<string, int>& element) {
            return element.first == currStop;
        });

        // If a matching stop_words word is found in the vector, erase it
        if ( it != vec.end() ) {
            it = vec.erase(it);
        }
    }
    return true;
}

// Print the result to the screen
bool printRes(vector< pair<string, int> >& vec) {

    // Test that the vector is longer than 20, to ensure no undefined behavior
    // below
    if (vec.size() < 21) {
        cout << "Vector is not valid" << endl;
        return false;
    }

    for (int i = 0; i < 20; i++) {
        auto it = vec.begin() + i;
        cout << setw(2) << right << i + 1 <<  ":  ";
        cout << setw(35) << left << it->first;
        cout << setw(10) << left << "occurances:  ";
        cout << setw(15) << left << it->second << endl;
    }

    return true;
}
```