CS 201 Homework 6

Solomon Himelbloom

November 11, 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 seventh homework assignment was created with readability and relatability at the forefront of program design. Code comments organized my thoughts and strategic plan for future additions to the codebase and files only included the minimum #include preprocessor directives needed to ensure optimal performance.

The hw6 folder contains the main program (main), two additional programs (hangman and math-catch), and various practice problems (calculator, shopping, and namespace). As always, each .cpp file starts with the author's name, the computer science class code, the 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 maps, namespaces, and the C++ Standard Template Library (STL). I added

additional comments and notes (for design and style) about my functions to reference on future assignments and included external links in my commit description to cite other sources that I used on this homework.

Some changes that I hope to add for the next homework assignment include revisiting how to add external installations to VSCode and C++, such as previous problems with the FLTK and Catch2 libraries. In conclusion, I hope to build on what I have learned during this assignment to better prepare myself for the CS201 final project.

3 Answers to Questions

1. Containers, as discussed in previous assignments, are collections of data (example: vector and map). Algorithms, on the other hand, sort between items within a range with specific examples such as 'sort' and 'find' in sequential order. Finally, iterators allow for algorithms to communicate and act upon data within containers. In other words, iterators are objects that reference items in a container.

2. Namespace pollution, similar to real-world pollution, is when an item is misplaced or there is a conflict with a similar naming scheme. We often are not allowed to declare multiple things with the same name (or identifier) in computer programming. We can prevent namespace pollution or name conflicts by placing identifiers in the general namespace.

   For example, if we write UA::Fairbanks, we may be referring to the identifier Fairbanks, inside the namespace UA and might have another problem calling two different items Fairbanks. In other words, the C++ Standard Library places all identifiers from corresponding headers into the standard (std) namespace to avoid any additional naming conflicts.

3. This iterator (vector member function end) does not point to the last item in a vector, rather it relates to the past-the-end element in the vector container. In other words, the

past-the-end element returned with the member function points to the element following the last element in the vector.

4. One example of a Standard Template Library (STL) algorithm in C++ is a mismatch. Mismatch, which does a variation of Sequential Search, enables you to look in two ranges at the same time. Rather than searching for a particular value, mismatch looks for corresponding items and ensures that they are not equal. As you might imagine, the mismatch algorithm has numerous unique applications for our programs but importantly returns an iterator for what is found, or "end" if not found.

5. Since maps allow for fast access to the corresponding value using a key, we can access our desired output one of two ways – passing by reference or noting the specific item in a series. Passing by reference allows with a range-based for loop allows for quick iteration and adjustment of previously mapped values, whereas we can access singular elements through an array-like syntax using square brackets.

6. We can remove a key-value pair from an std::map with the member function erase. It is important to note that in this structured data example, the member function erase only takes one argument – the key you want to remove.

7. Stability in sorting elements assists the withe preservation of the relative order within elements. A stable sorting algorithm helps to maintain the original order of the starting information. One example is sorting a deck of cards in ascending order numerically without reference to its respective suit. The unstable sorting algorithm does not preserve the original order of the cards, but the stable sort will accomplish this preservation. Keep in mind that there may also be performance differences between a stable and unstable sorting algorithm.

8. A lambda function is simply a function without a name. Beginning with a pair of brackets, we can also define a lambda

function within another function. Since lambda functions are unnamed, we can call them by setting a variable equal to it, and in-turn using the variable as a function. Pretty nifty, right?

9. Lambda functions have numerous benefits, some of which include more concise code, eliminate repetition, and improve overall readability. It is vital to recognize that similar to automation, lambda functions do not work well with one-time applications in your codebase. In the context of the Standard Template Library algorithms, we do not have to call the lambda function ourselves, rather we can pass it into an STL algorithm (such as sort). Think about ultimately using a lambda when you will be reusing this functionally more than just once or twice across your program.

10. If we use the same pseudorandom number generator (PRN) seed each time the program is run, we will get the same pseudorandom sequence each time the program is executed. Sometimes this behavior is desirable when testing and debugging our applications. For example, within a game, the same value (seed) might be used to generate a world so that the same conditions are present for every player. Notice that there are times that you would want to repeat a sequence whereas other times you may not. This should be decided on a case by case basis with relation to the specific needs of your application.

4 Program 1

4.1 Sample Output/Screenshot

Listing 1: Sample Program Output

```
Randomly-chosen mean: 1
Normal distribution around 1:
```

```
-7 n-6 n-5 n-4 n-3 *n-2 ***n-1 ******n 0 ********n 1
   **********n 2 ********n 3 *****n 4 ***n 5 *n 6 n 7
   n 8 n

0
0
0

Printed Distribution (Testing Key + Value Pairs):
Key: Alaska | Value: 907
Key: Fairbanks | Value: 99775
Key: United States | Value: 50
```

4.2 Git Commit Messages

Date	Message
2020-11-11	fix: spacing issues, comment typos, syntax
2020-11-11	add: PrintDistribution key value pairs
2020-11-11	add: RandomBetween and PrintDistribution
2020-11-11	add: pseudo-random number generation
2020-11-11	add: hw6 (initial files)

4.3 Source Code

4.4 random-map.cpp

```cpp
// random-map.cpp
// Solomon Himelbloom
// 11 November 2020
// Random map example for CS 201.

#include <iostream>
#include <iomanip>
#include <string>
#include <map>
#include <random>
#include <cmath>
using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::map;

```

```cpp
// Returns a uniform random number between first and last, inclusively.
int RandomBetweenU(int first, int last) {
    return 0;
}

// Returns a normally distributed random number between first and last, inclusively.
int RandomBetweenN(int first, int last) {
    return 0;
}

// Returns numbers using the rand() function from the C standard library <stdlib.h>.
int RandomBetween(int first, int last) {
    return 0;
}

// Prints a list of the random numbers clearly showing they are normally or uniformly distributed.
void PrintDistribution() {
    std::cout << "Printed Distribution (Testing Key + Value Pairs): " << endl;
    map<string, int> test_map;
    test_map["Fairbanks"] = 99775;
    test_map["Alaska"] = 907;
    test_map["United States"] = 50;

    for (const auto & printer_dist : test_map) {
        auto test_map_key = printer_dist.first; // Key
        auto test_map_value = printer_dist.second; // Associated Value
        cout << "Key: " << test_map_key << " | Value: " << test_map_value;
        cout << endl;
    }
}

int main() {
    // Seed with a real random value, if available.
    std::random_device r;

    // Choose a random mean between 1 and 6.
    std::default_random_engine e1(r());
    std::uniform_int_distribution<int> uniform_dist(1, 6);
    int mean = uniform_dist(e1);
    std::cout << "Randomly-chosen mean: " << mean << '\n';

    // Generate a normal distribution around that mean.
    std::seed_seq seed2{r(), r(), r(), r(), r(), r(), r(), r()};
    std::mt19937 e2(seed2);
    std::normal_distribution<> normal_dist(mean, 2);

    std::map<int, int> hist;
    for (int n = 0; n < 10000; ++n) {
        ++hist[std::round(normal_dist(e2))];
    }

    std::cout << "Normal distribution around " << mean << ":\n";
    for (auto p : hist) {
        std::cout << std::fixed << std::setprecision(1) << std::setw(2)
                  << p.first << ' ' << std::string(p.second/200, '*') << 'n';
    }

    int first, second, third;
    first = RandomBetweenU(1, 2);
    second = RandomBetweenU(3, 4);
    third = RandomBetween(5, 6);
    std::cout << "\n" << endl;
    std::cout << first << std::endl;
```

```
81     std::cout << second << std::endl;
82     std::cout << third << std::endl;
83     std::cout << "" << endl;
84     PrintDistribution();
85 }
```

5 Program 2

5.1 Sample Output/Screenshot

Listing 2: Sample Program Output

```
Starting hangman game... [complete]
Generating secret word... [complete]

Guess a letter!

Remaining: a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v
   ,w,x,y,z

 _________
|          |
|          0
|         /|\
|         / \
|
|


Game over!
```

5.2 Git Commit Messages

Date	Message
2020-11-16	refactor: vector wordlist to std::map
2020-11-16	add: check for user input in hangman.cpp
2020-11-16	refactor: split game into four function stages
2020-11-16	refactor: HANGMANWIKIPEDIACPP list
2020-11-11	add: wordbank and hangmanASCIIart
2020-11-11	add: hw6 (initial files)

5.3 Source Code

5.4 hangman.cpp

```
// hangman.cpp
// Solomon Himelbloom
// 11 November 2020
// Hangman example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <vector>
#include <random>
#include <map>
using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;
using std::map;
using std::mt19937;
using std::random_device;
using std::uniform_int_distribution;

void startHangmanGame() {
    std::cout << "\nStarting hangman game... " << endl;
}

void generateSecretWord() {
    std::cout << "Generating secret word... " << endl;

  std::map<string, int> word_list = {
    { "developed", 1 },
    { "computer",  2 },
    { "scientist", 3 },
    { "since", 4 },
    { "language", 5 },
    { "wanted", 6 },
    { "efficient", 7 },
    { "language", 8 },
    { "similar", 9 },
    { "provided", 10 },
```

```
  };

  /**
     * First two paragraphs of C++ on Wikipedia.org
     * (https://en.wikipedia.org/wiki/C%2B%2B)
     */
    std::vector<std::string> HANGMAN_WIKIPEDIA_CPP = {
    "programming", "language", "that", "developed", "extension", "language",
    "classes", "imperative", "object", "oriented", "generic", "programming",
    "features", "while", "also", "providing", "facilities", "level", "memory",
    "manipulation", "almost", "always", "implemented", "compiled", "language",
    "many", "vendors", "provide",  "compilers", "including", "free", "software",
    "foundation", "microsoft", "intel", "available", "platforms", "designed", "bias",
    "toward", "system", "programming", "embedded", "resource", "constrained",
    "software", "large", "systems", "performance", "efficiency", "flexibility",
    "design", "highlights", "found", "useful", "other", "contexts", "strengths",
    "being", "software", "infrastructure", "resource", "constrained", "applications",
    "including", "desktop", "applications", "servers", "commerce", "search", "servers",
    "performance", "critical", "applications", "telephone", "switches", "space",
    "probes", "standardized", "international", "organization",  "standardization",
    "latest", "standard", "version", "ratified", "published", "informally", "known",
    "programming", "language", "initially", "standardized", "which", "then", "amended",
    "standards", "current",  "standard", "supersedes", "these", "new", "features",
    "enlarged", "standard", "library", "before", "initial", "standardization",
    "developed", "computer", "scientist", "bell", "labs", "since", "extension",
    "language", "wanted", "efficient", "flexible", "language", "similar", "provided",
    "high", "level", "features", "program", "organization", "next", "planned",
    "standard", "keeping", "current", "trend", "version", "every", "three", "years" };
}

void hangmanASCIIart() {
  int totalGuessCount = 0;
    std::cout << " _________      " << std::endl;
    std::cout << "|         |     " << std::endl;
    std::cout << "|         0     " << std::endl;
    std::cout << "|        /|\\   " << std::endl;
    std::cout << "|        / \\   " << std::endl;
    std::cout << "|               " << std::endl;
    std::cout << "|               " << std::endl;
  std::cout << "\n" << endl;
    std::cout << "Game over { thanks for playing!" << endl;
}

int main() {
  startHangmanGame();
  generateSecretWord();
  string guess = "";
  string letters = "abcdefghijklmnopqrstuvwxyz";

  srand((unsigned) time(0));
  int random_number = 0;
  random_number = (rand() % 100) + 1;
  string random_word = "computer";
  int word_length = 8;

  for (int i = 0; i < word_length; i++) {
    std::cout << "Guess a letter: ";
    cin >> guess;

    if ((guess >= "a" && guess <= "z") || (guess >= "A" && guess <= "Z")) {
      // hangmanASCIIart();
```

```
    }

    else {
      std::cout << "Error: please try again with a letter.";
      break;
    }
  }

    // std::cout << "\nRemaining: " << letters << endl;
}
```

6 Program 3

6.1 Sample Output/Screenshot

Listing 3: Sample Program Output

```
Hello, world.
```

6.2 Git Commit Messages

Date	Message
2020-11-16	add: struct record, unitPrice, units
2020-11-11	add: hw6 (initial files)

6.3 Source Code

6.4 shopping.cpp

```
// shopping.cpp
// Solomon Himelbloom
// 11 November 2020
// Shopping example for CS 201.

#include <iostream>
#include <stdio.h>
#include <string>
#include <map>
using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::map;

```

```cpp
int main() {
    std::cout << "Hello, shopping." << endl;

    struct Record {
        double unitPrice;
        int units;
    };
}
```

7 Program 4

7.1 Sample Output/Screenshot

Listing 4: Sample Program Output

```
===============================================================================

All tests passed (21 assertions in 5 test cases)
```

7.2 Git Commit Messages

Date	Message
2020-11-16	add: accumulate and innerproduct tests catch.cpp
2020-11-16	add: atan2(x) STL function catch.cpp
2020-11-16	add: sin(x) unit test to catch.cpp
2020-11-16	add: template catch.cpp and catchamalgamated
2020-11-11	add: hw6 (initial files)

7.3 Source Code

7.4 math-catch.cpp

```cpp
// catch.cpp
// Solomon Himelbloom
// 11 November 2020
// Math catch for CS 201.

// This tells Catch to provide a main() - only do this in one cpp file.
#define CATCH_CONFIG_MAIN
#include "catch_amalgamated.hpp"
```

```
#include <cmath>

unsigned int Factorial( unsigned int number ) {
    return number > 1 ? Factorial(number-1)*number : 1;
}

TEST_CASE( "Factorials are computed", "[factorial]" ) {
    REQUIRE( Factorial(0) == 1 );
    REQUIRE( Factorial(1) == 1 );
    REQUIRE( Factorial(2) == 2 );
    REQUIRE( Factorial(3) == 6 );
    REQUIRE( Factorial(10) == 3628800 );
}

TEST_CASE ( "STL Unit Testing: sin(x)", "[sine function]" ) {
    REQUIRE(round(sin(4*M_PI)) == 0);
    REQUIRE(round(sin(8*M_PI/16)) == 1);
    REQUIRE(round(sin(M_PI/6)) == 1/2);
    REQUIRE(round(sin(0)) == 0);
}

TEST_CASE ( "STL Unit Testing: atan2(x)", "[arc-tangant function]" ) {
    REQUIRE(round(atan2(0, 0)) == 0);
    REQUIRE(round(atan2(M_PI, 6)) == 0);
    REQUIRE(round(atan2(M_PI, 2)) == 1);
    REQUIRE(round(atan2(M_PI, 4)) == 1);
}

TEST_CASE ( "STL Unit Testing: accumulate(b,e,i)", "[accumulate function]" ) {
    int fibannaci[] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377};
    REQUIRE(std::accumulate(fibannaci+9, fibannaci+10, 0) == 34);
    REQUIRE(std::accumulate(fibannaci+10, fibannaci+11, 0) == 55);
    REQUIRE(std::accumulate(fibannaci+11, fibannaci+12, 0) == 89);
    REQUIRE(std::accumulate(fibannaci+12, fibannaci+13, 0) == 144);
}

TEST_CASE ( "STL Unit Testing: inner_product(b,e,b2,i)", "[inner-product function]" ) {
    int inner_int = 100, even[] = {20, 30, 40}, odd[] = {1, 2, 3};
    REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
    REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
    REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
    REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
}
```

---

8 Program 5

8.1 Sample Output/Screenshot

Listing 5: Sample Program Output

```
Hello, world.
```

8.2 Git Commit Messages

Date	Message
2020-11-11	add: hw6 (initial files)

8.3 Source Code

8.4 math-catch.cpp

```
// catch.cpp
// Solomon Himelbloom
// 11 November 2020
// Math catch for CS 201.

// This tells Catch to provide a main() - only do this in one cpp file.
#define CATCH_CONFIG_MAIN
#include "catch_amalgamated.hpp"
#include <cmath>

unsigned int Factorial( unsigned int number ) {
    return number > 1 ? Factorial(number-1)*number : 1;
}

TEST_CASE( "Factorials are computed", "[factorial]" ) {
    REQUIRE( Factorial(0) == 1 );
    REQUIRE( Factorial(1) == 1 );
    REQUIRE( Factorial(2) == 2 );
    REQUIRE( Factorial(3) == 6 );
    REQUIRE( Factorial(10) == 3628800 );
}

TEST_CASE ( "STL Unit Testing: sin(x)", "[sine function]" ) {
    REQUIRE(round(sin(4*M_PI)) == 0);
    REQUIRE(round(sin(8*M_PI/16)) == 1);
    REQUIRE(round(sin(M_PI/6)) == 1/2);
    REQUIRE(round(sin(0)) == 0);
}

TEST_CASE ( "STL Unit Testing: atan2(x)", "[arc-tangant function]" ) {
    REQUIRE(round(atan2(0, 0)) == 0);
    REQUIRE(round(atan2(M_PI, 6)) == 0);
    REQUIRE(round(atan2(M_PI, 2)) == 1);
    REQUIRE(round(atan2(M_PI, 4)) == 1);
}

TEST_CASE ( "STL Unit Testing: accumulate(b,e,i)", "[accumulate function]" ) {
    int fibannaci[] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377};
    REQUIRE(std::accumulate(fibannaci+9, fibannaci+10, 0) == 34);
    REQUIRE(std::accumulate(fibannaci+10, fibannaci+11, 0) == 55);
    REQUIRE(std::accumulate(fibannaci+11, fibannaci+12, 0) == 89);
    REQUIRE(std::accumulate(fibannaci+12, fibannaci+13, 0) == 144);
}

TEST_CASE ( "STL Unit Testing: inner_product(b,e,b2,i)", "[inner-product function]" ) {
    int inner_int = 100, even[] = {20, 30, 40}, odd[] = {1, 2, 3};
    REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
```

```
48      REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
49      REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
50      REQUIRE(std::inner_product(even, odd+3, odd, inner_int) == 100);
51  }
```

9 Program 6

9.1 Sample Output/Screenshot

Listing 6: Sample Program Output

```
Hello, world.
```

9.2 Git Commit Messages

Date	Message
2020-11-11	add: hw6 (initial files)

9.3 Source Code

9.4 namespace.cpp

```
1  // shopping.cpp
2  // Solomon Himelbloom
3  // 11 November 2020
4  // Shopping example for CS 201.
5
6  #include <iostream>
7  #include <stdio.h>
8  #include <string>
9  #include <map>
10 using std::cin;
11 using std::cout;
12 using std::endl;
13 using std::string;
14 using std::map;
15
16 int main() {
17      std::cout << "Hello, shopping." << endl;
18
19      struct Record {
20          double unitPrice;
21          int units;
22      };
23 }
```