CS 202 Homework 001

Kelby Hubbard

January 26, 2020

- Repository Link: `https://github.com/krhubbard2/CS202`
- Git Commits: `https://github.com/krhubbard2/CS202/commits`
- This homework took approximately 06 hours to complete.

1 Design

The design I took on this program was a little odd. First I got a clock working without implementing it in a class. I did this for a couple reasons, the primary being I never tried to implement a clock before, so I wanted to ensure I could get it working first off, then simplified it and implemented in a class, for easier uses later on. After the StopWatch class was finished, the algorithms were pretty straight forward as they are things I've done before. I just implemented timing each algorithm and printing it out to the user.

2 Post Mortem

This assignment was pretty fun as I was able to see how quickly my computer could process mass quantities of items (although small). Everything went pretty smoothly. I stumbled a bit trying to understand how to set _start and _end as I needed to set them as a specific type, and couldn't figure out what type to set it as.

But once I figured out to just set it as exactly what I use it for, it was simple.

3 Answers to Questions

1. Processing items I believe is the slowest part of the process as you are building everything from scratch. I also think it is the most acceptable for the processing portion to be the longest as it doesn't need to be done often. It took my computer roughly 10 seconds to process 1B ints, so I could see how 1 trillion and 1 quadrillion could take quite a bit of time, as it will only get slower. Anything less should be very quick though.
2. I think searching items should be quick. A key example of this is binary search. There are many ways to optimizing searching, even if searching up to 1 quadrillion items, as searching is very important in today's society. You do not want to have to way 3 minutes for a search result to come up.
3. Based off the assignment we did, I was able to sort ints up to 1B objects in under a few seconds, which I believe is very acceptable. Although I think sorting up to 1 quadrillion objects could take a bit of time, it shouldn't take a massive amount as sorting may need to be done often (any time a new object is added to the data).
4. A struct is plain old data where as a class you can implement private, public, and protected. A class simply put a user defined data type.
5. Private are variables that can only be accessed inside the class, public can be accessed anywhere. Protected can only be accessed through friend functions within the class.
6. A method is the same as a member function. A member function is a function within a class. A member variable is part of a class.

7. A const member function is very similar to a member function except for a key fact that it cannot be modified and the object cannot be modified.

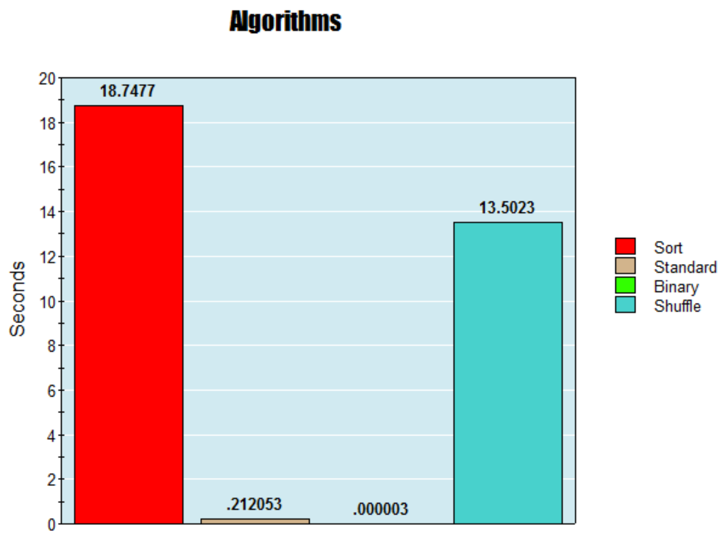
8. A byte is 8 bits. A char is also 8 bits. An unsined char is also 8 bits. A short is 2 bytes. An int is 4 bytes. A long int is also 4 bytes. A long long is 8 bytes.

4 Time It

4.1 Sample Output/Screenshot

Listing 1: Sample Program Output

```
Generating random number to compare to dataset.
Random number generated is 43171733.
Filling dataset of size 100000000.
Starting timer.
Finished job: Sun Jan 26 17:31:37 2020
Elapsed time: 1.08221s
Sorting the dataset
Finished job: Sun Jan 26 17:31:56 2020
Elapsed time: 18.4826s
Using standard search to search the dataset.
Finished job: Sun Jan 26 17:31:56 2020
Elapsed time: 0.131795s
Using binary search to search the dataset.
Finished job: Sun Jan 26 17:31:56 2020
Elapsed time: 3.22e-06s
Shuffling dataset.
Finished job: Sun Jan 26 17:32:10 2020
Elapsed time: 13.7693s
```



4.2 Git Commit Messages

Date	Message
2020-01-25	Created main.cpp, stopwatch.cpp, stopwatch.hpp
2020-01-25	Implamented StopWatch::starttimer
2020-01-26	Finished StopWatch::starttimer StopWatch::stoptimer StopWatch::elapsed
2020-01-26	Implamented std::sort
2020-01-26	Implamented standard search in main.cpp
2020-01-26	implamented binary search in main.cpp
2020-01-26	Implamented shuffle in main.cpp
2020-01-26	Added cout random number shown
2020-01-26	Finished

4.3 main.cpp

```
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It I

```

```
#include "stopwatch.hpp"
#include <vector>
using std::vector;
#include <algorithm>

int main()
{
  /*TO CHANGE DATASET SIZE AND RANDOM NUMBER SIZE PLEASE CHANGE VARIABLE
  "size" BELOW TO DESIGNATED NUMBER. FOR TESTING PURPOSES IT IS SET TO
  100M. IT IS NOT RECOMMENDED TO GO OVER 1B UNLESS YOU HAVE AN EXCESS
  AMOUNT OF RAM (POSSIBLY GREATER THAN 32GB). */

  int size = 100000000;

  StopWatch timer;

  //Generate random number
  cout << "Generating random number to compare to dataset." << endl;
  std::random_device rd;
  std::mt19937 gen(rd());
  std::uniform_int_distribution<> dis(1, size);
  vector<int> random = {dis(gen)};
  cout << "Random number generated is " << dis(gen) << "." << endl;

  //Filling dataset
  cout << "Filling dataset of size " << size << "." << endl
       << "Starting timer." << endl;
  timer.starttimer();
  vector<int> dataset;
  for (int i = 0; i < size; i++)
  {
    dataset.push_back(i);
  }
  timer.stoptimer();
  timer.elapsed();

  //Sorting dataset
  cout << "Sorting the dataset" << endl;
  timer.starttimer();
  std::sort(dataset.begin(), dataset.end());
  timer.stoptimer();
  timer.elapsed();

  //Standard search
  cout << "Using standard search to search the dataset." << endl;
  timer.starttimer();
  std::search(dataset.begin(), dataset.end(), random.begin(),
  random.end());
  timer.stoptimer();
  timer.elapsed();

  //Binary search
  cout << "Using binary search to search the dataset." << endl;
  timer.starttimer();
  std::binary_search(dataset.begin(), dataset.end(), random[0]);
  timer.stoptimer();
  timer.elapsed();

  //Shuffle
  cout << "Shuffling dataset." << endl;
  timer.starttimer();
  std::shuffle(dataset.begin(), dataset.end(), gen);
  timer.stoptimer();
  timer.elapsed();
```

```cpp



  return 0;
}
```

4.4 Stopwatch Header

```cpp
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It I

#ifndef STOPWATCH_HPP_
#define STOPWATCH_HPP_


#include <chrono>
#include <ctime>
#include <iostream>
using std::cout;
using std::endl;
#include <random>

class StopWatch
{
public:

  std::chrono::system_clock::time_point _start;
  std::chrono::system_clock::time_point _end;

void starttimer();
void stoptimer();
void elapsed();
};






#endif
```

4.5 Stopwatch Source

```cpp
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It II

#include "stopwatch.hpp"


void StopWatch::starttimer()
{
  _start = std::chrono::system_clock::now();
}

```

```cpp
void StopWatch::stoptimer()
{
  _end = std::chrono::system_clock::now();
}

void StopWatch::elapsed()
{
  std::chrono::duration<double> elapsed_seconds = _end-_start;
  std::time_t end_time = std::chrono::system_clock::to_time_t(_end);

  std::cout << "Finished job: " << std::ctime(&end_time)
       << "Elapsed time: " << elapsed_seconds.count() << "s\n";
}
```

5 Time It II

5.1 Sample Output / Screenshot

Listing 2: Sample Program Output

```
*******************************************************
************* VECTORS *********************************
*******************************************************
Adding Dracula, Moby Dick, Pride and Prejudice,The
   Scarlet Letter, and War and Peace to a vector.
Added Dracula to Vector.
Added Moby Dick to Vector.
Added Pride and Prejudice to Vector.
Added The Scarlet Letter to Vector.
Added War and Peace to Vector.
Finished job: Sun Jan 26 17:38:13 2020
Elapsed time: 0.0265656s
Sorting vector of books.
Vector is sorted.
Finished job: Sun Jan 26 17:38:13 2020
Elapsed time: 0.0626941s
Searching for a random string in the vector.
String not found.
Finished job: Sun Jan 26 17:38:13 2020
Elapsed time: 0.063771s
*******************************************************
```

```
************* MAPS ***********************************
******************************************************
Adding Dracula, Moby Dick, Pride and Prejudice,The
   Scarlet Letter, and War and Peace to a map.
Added Dracula to Map.
Added Moby Dick to Map.
Added Pride and Prejudice to Map.
Added The Scarlet Letter to Map.
Added War and Peace to Map.
Finished job: Sun Jan 26 17:38:13 2020
Elapsed time: 0.114443s
Searching for a random string in the map.
String not found.
Finished job: Sun Jan 26 17:39:44 2020
Elapsed time: 1.96e-06s
```

5.2 Git Commit Messages

Date	Message
2020-01-26	Created TimeItII, got 5 Project Gutenberg books, and copied stopwatch
2020-01-26	Implamented adding books to vectors
2020-01-26	Implamented std::sort for vector of books
2020-01-26	Implamented filling a map with Gutenberg books
2020-01-26	Organized code
2020-01-26	Imaplemented searching string in a map.
2020-01-26	Finished

5.3 Source Code

```cpp
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It II
#include <iostream>
using std::cout;
using std::endl;
#include <string>
using std::string;
using std::getline;
```

```cpp
#include <sstream>
using std::istringstream;
#include <fstream>
using std::ifstream;
#include <vector>
using std::vector;
#include "stopwatch.hpp"
#include <algorithm>
#include <map>
using std::map;

void vecadd(vector<string>& vec, string book)
{
  ifstream fin(book);

  //Can it read file?
  if (!fin)
  {
    cout << "Can't open file." << endl;
  }
  else
  {
    bool read = true;
    while(read)
    {
      string line;
      getline(fin, line);
      vec.push_back(line);
      //eof checking
      if (!fin)
        {
          if (fin.eof())
          {
            read = false;
          }
          else
          {
            read = true;
          }
        }
      }
    }
  }

void mapadd(map<string, int>& map1, string book)
{
  ifstream fin(book);
  int i = 1;
  //Can it read file?
  if (!fin)
  {
    cout << "Can't open file." << endl;
  }
  else
  {
    bool read = true;
    while(read)
    {
      string line;
      getline(fin, line);
      map1.insert({ line, i });
      i++;
      //eof checking
      if (!fin)
        {
          if (fin.eof())
          {
            read = false;
          }
          else
          {
            read = true;
          }
        }
      }
    }
}
int main()
{
  StopWatch timer;

  //Making a vector of 5 Project Gutenberg books

  cout << "*******************************************************\n"
       << "************* VECTORS *********************************\n"
       << "*******************************************************\n";

  //Filling vector
  vector<string> books;
  cout << "Adding Dracula, Moby Dick, Pride and Prejudice,"
       << "The Scarlet Letter, and War and Peace to a vector."
       << endl;
  timer.starttimer();
  vecadd(books, "Dracula.txt");
  cout << "Added Dracula to Vector." << endl;
  vecadd(books, "Moby Dick.txt");
  cout << "Added Moby Dick to Vector." << endl;
  vecadd(books, "Pride and Prejudice.txt");
  cout << "Added Pride and Prejudice to Vector." << endl;
  vecadd(books, "The Scarlet Letter.txt");
  cout << "Added The Scarlet Letter to Vector." << endl;
  vecadd(books, "War and Peace.txt");
  cout << "Added War and Peace to Vector." << endl;
  timer.stoptimer();
  timer.elapsed();

  //Sort
  cout << "Sorting vector of books." << endl;
  timer.starttimer();
  std::sort (books.begin(), books.end());
  cout << "Vector is sorted." << endl;
  timer.stoptimer();
  timer.elapsed();

  //std::find a random string in the vector
  cout << "Searching for a random string in the vector." << endl;
  string random = "This string is not in the books.";
  //timer.starttimer();
  if (std::find(books.begin(), books.end(), random) != books.end())
  {
    cout << "String found!" << endl;
  }
  else
  {
    cout << "String not found." << endl;
  }
  timer.stoptimer();
  timer.elapsed();


  //Maps
  cout << "*******************************************************\n"
       << "************* MAPS ************************************\n"
       << "*******************************************************\n";

  //Filling map
  cout << "Adding Dracula, Moby Dick, Pride and Prejudice,"
       << "The Scarlet Letter, and War and Peace to a map."
       << endl;
  map<string, int> m;
  timer.starttimer();
  mapadd(m, "Dracula.txt");
  cout << "Added Dracula to Map." << endl;
  mapadd(m, "Moby Dick.txt");
  cout << "Added Moby Dick to Map." << endl;
  mapadd(m, "Pride and Prejudice.txt");
  cout << "Added Pride and Prejudice to Map." << endl;
  mapadd(m, "The Scarlet Letter.txt");
  cout << "Added The Scarlet Letter to Map." << endl;
  mapadd(m, "War and Peace.txt");
  cout << "Added War and Peace to Map." << endl;
  timer.stoptimer();
  timer.elapsed();

  //Searching map
  cout << "Searching for a random string in the map." << endl;
  timer.starttimer();
  auto key_count = m.count(random);
  if (key_count != 0)
  {
    cout << "String found!" << endl;
  }
  else
  {
    cout << "String not found." << endl;
  }
  timer.stoptimer();
  timer.elapsed();

  return 0;
}
```

5.4 Stopwatch Header

```
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It I

#ifndef STOPWATCH_HPP_
#define STOPWATCH_HPP_

```

```

#include <chrono>
#include <ctime>
#include <iostream>
using std::cout;
using std::endl;
#include <random>

class StopWatch
{
public:

  std::chrono::system_clock::time_point _start;
  std::chrono::system_clock::time_point _end;

void starttimer();
void stoptimer();
void elapsed();
};






#endif
```

5.5 Stopwatch Source

```
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It II

#include "stopwatch.hpp"


void StopWatch::starttimer()
{
  _start = std::chrono::system_clock::now();
}

void StopWatch::stoptimer()
{
  _end = std::chrono::system_clock::now();
}

void StopWatch::elapsed()
{
  std::chrono::duration<double> elapsed_seconds = _end-_start;
  std::time_t end_time = std::chrono::system_clock::to_time_t(_end);

  std::cout << "Finished job: " << std::ctime(&end_time)
       << "Elapsed time: " << elapsed_seconds.count() << "s\n";
}
```