Programming Assignment 5 Report

Josh Zschiesche

April-7-2016

CSCE 221-505

Dr. Teresa Leyk

I) Program Desciption-

The program reads in various files, extracts the UIN and grade from one file and then checks for the matching UIN and grade from the other files and appends them together in memory and them prints all of that out in an output file so that the instructors can use it for determining whether students have submitted certain things for a grade or not.

II) Purpose of the Assignment-

The purpose of the assignment was to learn about the data structure hash table by creating a hash table from the files input.csv and roster.csv and use that hash table to find all of the inputs from input.csv and use it to append the grades onto roster.csv and make that into a new .csv file that can be looked at easily. The whole purpose of using a hash table is so that it is quicker to look up and append these grades because the way the hash table works is by remembering either the position or a way to get the position of any element in the table. This makes run time much quicker, because it doesn't have to search through the table, it already knows each position.

III) Data Structures Description-

The data structure that I learned most about was a hash table, implemented with an array. Hash tables are extremely efficient in terms of storing memory and retrieving information, as the computer only has to have a way of finding the position of each element in the array and then go to that element. Implementation was completed with the use of the mod operator and the use of array. I wrote a function that modded the UIN, which was extracted from roster, with the size of the table (so as to avoid collisions), and returning that as an int to store in the array.

IV) Algorithm Description-

The basic premise of the main algorithm I used is to first read in input.csv, then make a hash table in memory, then it would read in each element of roster.csv, replacing whatever was in the grade position with a negative 1, then it went through the elements of the hash table, found each UIN that matched the UIN in roster.csv and then replaced the grades of the UIN's with the grades that were in the hash table. Finally, it went through all of the elements in the new hash table and if the grade had a negative 1 element as the grade, it would not output the grade to the csv file and if there was an actual grade detected, it would append that to the other parts of the structure.

V) Program Organization and Description of Classes-

The program is completely contained within one file, as there are only two custom classes and one custom function, each having 6 lines or less, making it almost pointless to spread the program among several files.

Classes:

1. record: The class record contains an int to store the grade, an int to store the uin, a string to hold the e_mail and a string to hold the name of the student. This is so that each individual part of input.csv and roster.csv can be read into and be contained in a class to make it easier to place into a hash table and store.
2. uin_grade: this class only is made of two ints, one to store the grade and one to store the uin. This is only used when reading in the input from input.csv. as the only thins that I needed to remember to create the hash table initially is a uin and a grade.

VI) Instructions to Compile and Run my Program-

Simply put all of the files from my tar file in a new directory, and type the standard line g++ -std=c++11 main_PA5.cpp && ./a.out to compile and run. You may have to delete the a.out file in my directory.

VII) I/O Specifications-

The format of the input from the file must be in the form NAME, E-MAIL, UIN, GRADE (example: Jessica Robbins, jr@gmail.com, 123456789, 88) for the input.csv file, and must be in the form NAME, E-MAIL, UIN, (example: Jessica Robbins, jr@gmail.com, 123456789, for the roster.csv file otherwise the code will break, as there are no exceptions that will handle any other type of input.

VIII) Logical Exceptions-

There is no exception handling.

IX) C++ Object Oriented or Generic Programming Features And STL Features & Classes-

I used the STL containers array and vector and the classes string and int. I used the C++11 feature auto so that I could implement a ranged based for so that the compiler could automatically determine the type of any object encountered in the files, hash table, or vectors when reading in files and moving around record objects within the hash table itself.

X) Tests-

To test the program, I simply ran the program several times with the given files until it generated the correct output every time the main program ran. I also checked this output with several other students (their names are on the coverpage) to make sure that my output was correct.

XI) Expected Running Time of Hashing Algorithms-

Because the hashing algorithm only contains one for loop, the algorithm runs with O(n) time complexity, making the algorithm extremely efficient by comparison to other searching algorithms When actually searching, the time complexity is O(1) time because it used only one function to calculate the location of whatever element it is searching for.

XII) Conclusion-

I learned a lot about hash tables and how they work and I also learned more about how to implement various types of STL features and the auto and range based for loop. I really liked the efficiency of hash tables and will likely use them a lot in future classes, as they are easy to implement and very fast.