CSE 1320 - Fall 1999
Postlab 9

Objective

This is the first in a series of 3 postlabs that are designed to be the capstone of this course. This series will utilize the structured program design concepts that have been taught this semester. Starting with analysis and preliminary design, through the module by module development and culminating in the integration of these modules into the solution of the given problem.

Further, the final programming concepts utilizing data structures, binary files and link lists will be practiced.

Assignment

1. Analysis (35%)

2. Structure Chart (65%)

3. Submit the above at the beginning of your Lab 10 period.

Problem Description

A registration processing program to handle students who register for CSE 1320. A list of the students who pre-registered for the class is in stuList.dat in the public data directory.

The registration program will need a module that can read this text file, store each student's data in a struct, and write that struct out to a binary file called stuList.bin. This includes creating the student account, which is of the same form as your omega accounts: fmlxxxx, where f is first initial, m is the first middle initial, l is the last initial, and xxxx is the last 4 digits of the student id number. You should have a separate module that when passed a student id number and a name returns this account. Further, it includes reading in the first 9 lab grades and calculating the average. You should have a separate modules for returning the floating point average of an array of integers.

To determine if the read text file module should be executed, the stuList.bin file should be opened to see if it exists. If stuList.bin does exist, then the module should NOT be executed; otherwise, it should be executed to create stuList.bin.

If the stuList.bin file does exist or after it is created, then the file should be read into an array of structs.

The data to be included in the struct is as follows:

Student ID number (5 digit int, first column of stuList.dat)
Student Name (Up to 35 character string in the form:

Last, First, Middle, Other
e.g. Ramirez, Jorge, Carlos, Guillermo)
Student Acct (7 character string)
Student Grades (12 ints)
Student Average (a float)

Then the program should repeated present the user with the following options:

1) Process Add/Drop File
2) Process Individual Add/Drop
3) Sort by Student Name (Alphabetic order)
4) Sort by Student Average (Descending order)
5) Update Grades
6) Print list
9) Exit

To Process Add/Drop File, the array should be sorted into student ID number order automatically, if it is not already in that order (which it would be immediately after having been read in and up until the user chooses to resort it via option 2 or 3). Then the user should be prompted for which add/drop file to process. The add/drop files are text files (located in the public data directory) and look much like the original stuList.dat file, except the first column contains an A or D. If it is a D, then only the student ID number is listed, if it is an A, then the student's name and first 9 lab grades are also given. For each add, a struct should be created and inserted in student id number order. For each drop, the struct should be deleted from the array. Once all add/drops are processed, the stuList.bin file should be re-written with the new data.

To Process Individual Add/Drop, it doesn't matter what order the array is in. You will prompt the user for add, drop or quit. If it is drop, then prompt the user for the student ID number, search the array, display the student information, and ask the user if it is the right one. If so, delete it, if not, go back to prompting for add, drop or quit. If it is add, then you need to prompt for the appropriate information, including the correct number of grades that have been already entered for the other students. Then, figure out where to insert in the array based on the order the array is sorted on.

The sort modules should simply reorder the array into the appropriate order (note that you will have 3 separate sort modules, but if you use bubble sort, only one swap module is needed.

To Update Grades, the array should again be in student ID number order. The user should then be prompted for the file to process. The grades should be added to the students' records, and the grade average updated based on the number of grades the student should have which is given by the number in the first line of the file. The format of the grade file is:

The first line contains what grade number this file has in it.
Each successive line has:
The student id numbers (in order) followed by the grade they

made on that assignment.
If they have make-up work or grade corrections, the next number on the line is the assignment number, then the grade for that assignment, and this is repeated as necessary.

When the grade updating is concluded, the stuList.bin file should be re-written.

Print list should print a nicely formated list in the order that the array is currently sorted. The columns should include the student id number, the name, the account and the grade average.