**Programming Assignment 1**

Structures, arrays of structures, functions, header files, program logging

Program description:

Read student and class data and compute the GPA. Use arrays of structures to hold data to process.

1. You are to read a data file (students.txt, provided) containing student data for students taking classes at a local community college.

You are to read each line of student data for a student and calculate the student’s GPA.

The student file consists of several rows of data that contain:

First Name

Last Name

ID

Number of class – grade pairs to read

Class – grade pairs.

For example (not actual data):

Pam Smith 123 2 CSC201 A PSYC100 B

Student first name is Pam, student last name is Smith, ID 123, 2 is number of class/grade pairs to read, CSC201 A, PSYC100 B are the classes and grades. The read for the next student would then begin.

1. Create a **structure** to contain the information for a student:

First Name

Last Name

Student ID

GPA

Declare the structure in a header file.

1. You are also to read a data file (classHours.txt, provided) containing class and hours information, which consists of a class identifier and the number of credit hours.

For example:

CSC201 4

PSYC200 3

CSC201 consists of 4 credit hours, PSYC200 consists of 3 credit hours

1. Create a **structure** to contain the information for the class and credit hours data:

Class Designation

Credit Hours

Declare the structure in the header file

1. Declare the structures for the student and class data in the header file. Declare an array variable for each of the student and class structures **LOCALLY** in main. Declare an array size for the student data as 20; for the class data as 15.
2. Read and store the class data into the class structure array. Read until end of file. Use a counter to count and track how many rows read. Pass the counter in to any processing functions that loop through structure array (do not use a global variable).
3. Read and store the student data into the student structure array. Read until end of file. Use a counter to count and track how many rows. Pass the counter in to any processing functions that loop through the structure array (do not use a global variable).
4. Calculate the GPA for each student person **while reading the data** and then store the GPA in the in the structure. You ARE **NOT** to **store** the class/grade pairs in the **student array** for each student. The total GPA calculation will involve reading the class data into the class array structure first and looking up the credit hours based on the class ID and looking up grade point values while reading the student data.
5. The grade point values per grade are below. Look up grade values using a function.

|  |  |
| --- | --- |
| A | 4.0 |
| B | 3.0 |
| C | 2.0 |
| D | 1.0 |
| F | 0.0 |

1. Read and store the class and credit hour data into the structure array using a function.
2. Read and store the student data in into the structure student using a function.
3. The data file reads may be in the same function.
4. Process the all student data using a function. This can also be inside the read function.
5. Calculate the GPA using a function. GPA is calculated as follows:

The formula:  
Grade Points earned / total credit Hours = GPA

**The Calculation**

1. Total the credit hours attempted, i.e., the number of hours per class taken, and total the grade points earned for each class taken.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Credit Hours* |  | *Grade* |  | *Grade Points* |
| 4 hours | x | A (4.0) | = | 16.0 |
| 3 hours | x | B (3.0) | = | 9.0 |
| 5 hours | x | C (2.0) | = | 10.0 |
| 12 hours |  |  |  | 35.0 |

1. Grade points are calculated by multiplying the class credit hours by the grade points assigned to the grade earned.
2. Once all student class data is read and totaled, divide the total grade points by the credit hours attempted.

35.0/12 = 2.916 GPA. Display 2 decimal points 2.92

You must read and total the grade points earned by reading the grade and looking up the grade point value.

You must look up the credit hour values for each class by searching for the class in the classHours structure array.

The grade point lookup and the GPA calculation must be in separate functions.

1. Using a function, output the student data to the console (cout) and to an output data file including the following information:

First Name, Last Name, ID, GPA

Ensure each line of data is neat and readable. All GPA values must have two decimal places.

1. You will be required to log the progress of the program in a log file (name of your choice). For program logging, write to the log file at the beginning of each function and the end of each function. This also helps in the debugging process. For example:

Entering function OpenFiles;

Exiting function OpenFiles;

Entering function readData.

Exiting function readData.

Also output the number of rows of class data read, along with an informational message

Also output the number of rows of student data read, along with an informational message

You may supplement the content of the log file with any other messages you wish

1. Use a function to open all input and output files. Request all file names from the user EXCEPT the log file, which you may choose and hard code in the program. You may open all files in the same function. You will open four (4) files, students.txt, classHours.txt, an output file and a log file. Check the file state after each open. If the file does not open, display a message and exit the program.
2. Main should contain function calls only; no processing.
3. Close all files prior to program end. The close statements may be in main. **Three (3) points extra credit** will be awarded if the close statements are included in a function and the function is called by main.
4. All structure DECLARATIONS, global constants and function prototypes MUST be in a header file. Include the header file in the program. DO NOT CODE THE FUNCTIONS IN THE HEADER FILE. Functions MUST be coded in the .cpp file AFTER main.
5. Each function **must** have required documentation (pre and post conditions).

**Test and evaluate calculations for accuracy. Points will be taken for inaccurate calculations, improper formatting, directions not followed.**

Turn in file code \*.cpp; README file; header file\*.h and both input files, log file and output file. You DO NOT need to submit the executable (\*.exe). You MAY zip all the files and submit if you choose. You MUST include all the files indicated or points will be deducted.

README must contain instructions for location of input/output files

**Points WILL BE taken if MINIMUM requirements and submissions not included.**

**1 point extra credit will be awarded if assignment is zipped for submission**

Each solution is to be uniquely your own; minimal student collaboration allowed.

See Canvas and Syllabus for due date