CS1 Spring 2019

Assignment 5: Simple Employee Tracking System

This assignment involves creating a program to track employee information. Keep the following information on an employee:

1. Employee ID (string)
2. Last name (string)
3. First Name (string)
4. Birth date (string as MM/DD/YYYY)
5. Gender (M or F, single character)
6. Start date (string as MM/DD/YYYY)
7. Salary per year (double)

Thus you must create a class that has all of this, and get/set methods for each of these fields. **Note:** The fields that are designated as string should use the *string* class, not a char array.

Your class must have three constructors:

1. No arguments. Just construct an object.
2. Takes only an employee ID
3. Takes all information

When the program starts it must check to see if a file called Employee.txt exists. If it does, read the information into Employee objects which you dynamically allocate and put them into an array of pointers to objects. Data in the file is stored separated by spaces, one employee per line. Assume the company will have no more than 100 employees, but if it does, show an error. You may not use vectors. If the file does not exist, your program will create it in step 5, below.

The program will have a menu that shows the following options:

1. **Enter new employee information**. When the ID is entered, make sure that ID is not in use for another employee. Request the rest of the info and create a new Employee object. While some input validation would be good, the only requirement is that the data not be null and that the salary must be a valid floating-point number greater than zero. (Doing proper input validation is beyond the scope of this exercise and could easily double the size of the program. Dates, in particular, are difficult to validate.)
2. **Display all employee information in alphabetical order by last name.** The list may not have been entered in order, but you must sort it to display it. Show the information in fixed-field columns so that it looks neat. (You can use printf for this if you like.) Show the salary to the nearest dollar.
3. **Look up an employee by ID.** If the ID exists, show all of the information. If not, display a message.
4. **Remove an employee.** Ask for an employee ID, and if the ID exists, delete. If not, display a message that there is no such employee. This should delete the object pointer from your array and remove the Employee object from memory. (How do you handle this in the array?)
5. **Save all data to Employee.txt and exit.** If the file exists, overwrite it. If it does not exist, create it.

Invalid menu options will display a message and return to show the menu. After executing options 1 through 4, return to the menu. Option 5 saves and exits.

The Employee class will not have a method to write all of the employees to the file, since it does not know about more than one employee at a time. However, you will have a method somewhere in your program to write them all out, with each piece of data separated by a blank, with one employee per line.

**To hand in:** Three files, called Employee.h, Employee.cpp. and <netID>Asg5.cpp. Place these into a Zip file called <netID>Asg5.zip and hand that in.

|  |  |
| --- | --- |
| **Grading** | |
| Program initializes properly, reading information if it is there | 10 |
| Each menu item 1 through 5 works correctly. 15 points each | 75 |
| Employee class contains all and only the necessary functions and follows conventions for how classes work | 5 |
| Program comments and good variable names | 10 |

Additional grading guidelines:

1. Method in Employee that does I/O including display: -5
2. Not using pointers to objects. -10
3. Use of Vectors: -10
4. Not closing files that have been opened: -5
5. Not freeing allocated memory: -5