Employee Attendance System

Documentation

**Introduction**

This is a simple Employee Attendance System that is based on C++ programming language and makes use of its basic concepts, as listed below:

* Data types (int, char, string, bool)
* Variables
* Functions
* Loops (for, while)
* Conditional Statements
* Arrays (1-D, 2-D, linear search)
* Pointers (static variables)
* Object-oriented programming concepts (classes, objects, friend functions)
* Input/Output stream (cin, cout)
* File I/O (fstream)

It supports basic functionalities that would be expected from an attendance program such as:

* Logging In by entering respective employee password
* Logging Out by entering respective employee password

Apart from this, it also features an “Admin Mode” which provides the system administrator with various attendance details and privileges such as:

* View Attendance Log
* Setting date
* View Log details
* Search Login status of a particular employee
* Write the Attendance Log to an Excel Spreadsheet which can be stored on the administrator’s system.

The administrator mode can be accessed with a set password that has been discussed in detail further below.

**Key Components**

This Attendance Program is built upon various components working coherently in order to function. These Components are as listed:

1. Classes: The program consists of two classes which have their own set of class member functions:
   * + - class Employee: This class consists of private and public access modifiers. The private access modifier holds 3 arrays, namely firstName[] , surName[] and password[] , all of which have an array size of dirSize, which is a variable of type ‘static const int’. As the name suggests, the firstName[] array holds the first names of employees,

the surName[] array holds the surnames of the employees, and the password array holds the passwords of the employees. All of these are string typed arrays.

Now ,the public access modifier of this class contains all the checks required for a safe and legitimate login and logout.

Variables: It consists of string data typed variables namely enteredFirstName which will take the first name as the input, enteredSurName which will take the surname as the input and enteredPassword which will take the password as the input. There’s also an int nmeSrchr which will be discussed further.

void readName(): It is this function that will be taking the first name and the surname through input stream and send it over to chkName().

chkName(): The chkName() function makes use of a for loop within which are some conditional statements that have been used to check the presence of the entered name and surname in the firstName and surName array and generate a response accordingly. If the entered first name corresponds with the entered surname at the same index for both arrays, the readPassword() member is called.

readPassword(): Once called, this member first greets the employee with their full name and then prompts them to enter their respective password. This password is then sent over to the chkPassword() member.

chkPassword(): This function checks for the validity of the password at the same index where the first name and surname were found, this index being the static int nmeSrchr we talked about earlier. If the enteredPassword matches the nmeSrchr’th element of the password array, the rechkName() member is called. If the password entered does not match , the user is still given 3 attempts to enter the correct password.

rechkName():This function will run the enteredFirstName and enteredSurName through a 2-D array- atndncArray[][] (discussed further) which will check if the name received from the readName() is already present in the atndncArray. If it is, it would mean the employee has logged in and wishes to log out. If it’s not, the login procedure would continue in the usual way .

* + class adminMode(): The adminMode() is a friend class to the Employee class. The intent behind making it a friend class to Employee is to clear out any restriction or clash it might have from accessing data that could possibly be accessed only through the Employee class. This not only decreases the size of the already packed Employee class, but also makes the code cleaner and maintainable. Functionally, it is this class that carries the various features available to the administrator upon entering the right command. These features being:
    1. showAtndncTable(): This member allows the administrator to access the Attendance Log. It runs a for loop on the atndncArray and prints out its elements in a tabular form to show details such as Event number, employee name, and the action performed by them at that specific event. It also has a date on the top of the table, which is set to “Unset” by default and changed only when it’s set or modified by the administrator.
    2. setDate(): To maintain the attendance record, the admin would certainly need to mention the date of that attendance log. This is precisely what this member does. Upon calling it with the right command, the administrator would be able to set the date in dd/mm/yyyy format . An algorithm used in this function not only converts the entered date into a literal format but also returns the weekday of the entered date. Now, this date , which is a static string data type, will store this date and display it on the top of the attendance table.
    3. seeLogDetails(): On entering the command, this member would allow the administrator to see how many employees have logged in and logged out. It uses 2 static int variables, i.e. logInNum and logOutNum which store the number of logins and log outs each time someone wishes to, well , log in and log out.
    4. searchEmployee(): This member was initially supposed to be in the admin mode but due to data access and global variable issues it had to be stored in the Employee class. This member allows the administrator to see if a certain employee has logged in yet or not. It uses the atndncArray to determine the same.
    5. writeAttendance(): On entering the right command, this member writes the attendance log in csv format which can be viewed in an excel spreadsheet. In short, it writes the Attendance log to an excel spreadsheet for record keeping of the attendance. Note- Make sure the spreadsheet on which the I/O is being performed is closed while entering this command.

1. Static Variables- The program contains a bunch of static variables each with their own essentialities:
   * static int loginNum: This static variable holds the count for the number of logins.
   * static int logOutNum: This static variable holds the count for the number of logouts.
   * Static const int dirSize: This variable contains the number of elements that will be stored in the firstName, surName and password array. The reason behind making it const along with int was the fact that since firstName, surName and password are a class member , and they may not be arrays of a variable size. And the reason for declaring it as a global variable is so that other variables and functions might be able to access it. This variable eases the task of adding a new entry to the firstName, surName and password array as the programmer would just have to increase the value of dirSize accordingly without having to worry about modifying more values further in the code.
   * static int addName: This variable is an index variable to the atndncArray. It indexes events such as login and logout to the atndncArray and increments after each iteration through the rechkName() so the next event would be stored to the next index of the atndncArray.
   * static string atndncArray[dirSize][3]: This is a 2-D array of size dirSize and sub-array size 3. It contains the details of every event, i.e the first name of the employee being the first element of the subarray, the sur name being the second and the action (Logged In or Logged Out) the third. This is also the array that is used by the rechkName() member of class Employee to maintain the login and log out flow.
   * Static string date: This variable will store the input date obtained from setDate() member of adminMode class. This string will be printed on top of the attendance log.
2. Functions: Since most of the task is being done by arranging functions into class members, so there are only 2 standalone functions (or free functions) declared and defined in the program:
   * int main(): The int main function is the one which will be in charge of the execution of both of the classes. It first runs a for loop for a continuous login/logout loop that will only end if the program is shut down. This for loop first asks the user if they wish to login or logout. Upon receiving a positive response, i.e Y or y which stands for ‘yes’, a variable that stores the login number, which is loginCount increments by one . After this an array in the form of class object function is made, i.e emp[], which increments after every iteration. This object calls the readName() member, and procedurally other members are called. If the response is ‘n’ or ‘N’, which stands for ‘no’, the loop continues to its next iteration, skipping over all the tasks performed if response was ‘y’ or ‘Y’.

**Entering the Admin Mode**

To enter the admin mode, the response entered will be a password that will only be known by the administrator. This password being **“$adminMode\_enter”**. Upon entering this password the adminModeFunction() will be called.

If the entered response matches neither of the above-mentioned cases, “Invalid Response” will be returned and consequently the loop will move to its next iteration, asking for a response again.

* adminMode(): As mentioned before, this function can be called by entering **“$adminMode\_enter”**. Upon calling, this function will open up the admin command line interface where the admin can write in different commands and receive output. It’s built using a for loop which will keep prompting for a command . Within this for loop, is an input stream where the command will be entered. After this the code goes through the bunch of if statements to match the command. An object to the adminMode class is made(adminEvent) and if the command matches any of the conditional statements, the corresponding member of the adminMode class is called by the adminEvent object. Some of these commands are:
  + 1. getAttendanceLog: This command calls the showAtndncTable() member of the adminMode() class. This will print out the attendance log in a tabular form in the console.
    2. getLogDetails: This command calls the seeLogDetails() member of the adminMode class. This will print out the existing number of employees logged in and logged out.
    3. setDate: This command calls the setDate() member of the adminModeClass. This will enable the admin to set the date.
    4. searchEmployee: This command will call the searchEmployee() member of the Employee class.

The format in which this command should be typically entered is : searchEmployee firstName surName.

* + 1. writeAttendance: This command will call the writeAttendance() member of the adminMode class. This will write the attendance log to an excel spreadsheet.
    2. clearConsole: This command will clear out the console and call main() function.

1. Libraries: One of the most important components of this program is the set of libraries it makes use of, which as follows:
   * 1. iostream: Stands for input-output stream. It is the fundamental library used in any kind of program. In the context of this specific program, it is this header file which supports the usage of input and output services for the console, such as ‘cin’ ,’cout’,etc.
     2. stdlib.h: Stands for ‘standard library’, is also a general-purpose library used oftentimes in C/C++. For this program, it supports the usage of ‘to\_string()’ function used to convert the entered numeric date by the administrator to a string type literal date. It has also been used for the system(“cls”) function which clears the console.
     3. conio.h: Stands for console input-output header. It provides a set of functions for handling keyboard input and screen output , for example, the getch() function used in main() is a part of the conio.h library which awaits the user to press a key on the keyboard to proceed with the next login/logout process.
     4. fstream.h: Stands for file stream. It allows us to operate input and output operations on a file. For this program, this library has been used in the writeAttendance() member of the adminMode class, which appends the attendance log to a csv file which can be viewed on an excel spreadsheet.

**Working**

The flowchart below describes the working of the program by describing the general flow of the program and how various components interact with each other. Since this documentation has already exceeded 2000 words, a diagrammatic representation would be a nice change of pace.

Diagram

Description automatically generated