Name: Gabriel Novoa

C++ Final Project (Employee Absence Management System)

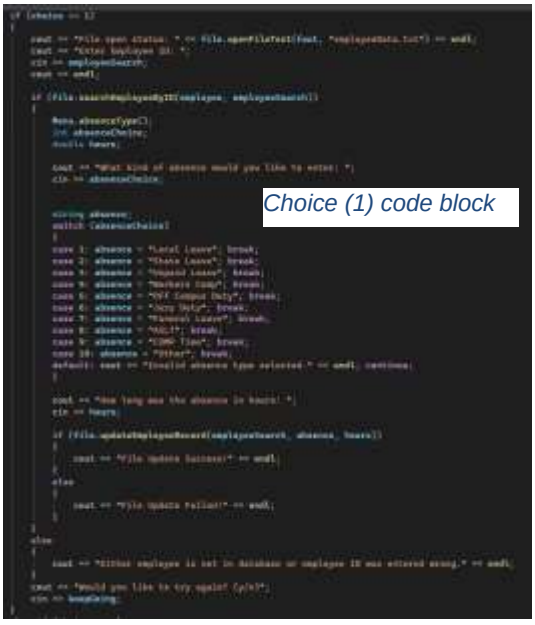
Why did I choose this project?

For my C++ final project, I created an employee absence management system. This project idea was influenced by my mom who works as a high school secretary for the Houston Independent School District. Last semester I was very ecstatic about the idea of finally learning how to code. I remember my mom asking me to create an absence tracker when I got comfortable with coding. The first coding language I learned was Python but I felt like I needed more knowledge to create the project using Python. I told myself that the following semester I was going to pay more attention to my coding professor so that I would be able to create this project for my mom.

What does it compute? What are the inputs and outputs from your program? / Describe How the program works.

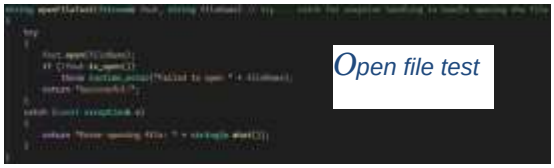
Main Menu Screen

My program consists of a text file that contains employee information (Employee ID, Employee name (Last, First), and the employee's PC# (Position category)) and the program itself which is run on Visual Studio. When you first run the program you will be prompted with a Main Menu screen (shown to the right) which serves to handle the main operations that the code will perform. Once the main menu is shown the program will also ask the user which option they would like to choose. The Program decides what to do from the user's input in the form of a number. Going down the option list, choice (1) is the spotlight of the code which is the option that stores the absence type and the number of hours the said employee was absent. Once number (1) is entered into the console buffer, the first of many (if- if else – else) statements. Then the program will check if the file was able to be opened and if not, it will tell the user. The open file test function works by using a try…catch exception handler. After the open file test is run the user is prompted to enter the id of the employee they'd like to search for. If the employee exists in the file, then the second if statement is run, if not then the user is told that the employee doesn't exist. This work is done by the search employee by id function. If the employee does exist, then the second menu is shown which is the absence type menu. This menu is like



Choice (1) code block

the main menu. From an organizational standpoint and for easy access I've put both menus in a class of their own called the main menu class. After the menu type is selected and the number of hours absent are selected the update employee record function is run. This function creates a vector to store all the file contents temporarily. Then a while loop is used to locate the employee whose information must be updated. A string stream is used to concatenate the already existing employee information with the desired absence type and hours absent. The set precision is used to make sure the hours appended only have two decimal points. After this is don't the modified employee data is pushed back into the vector and the data is truncated to the file. This information appended to the end of the employee who was absent. After this is done the program tells the user that the file was updated and then the user is asked if they would like to keep the program running and choose another option from the main menu screen.

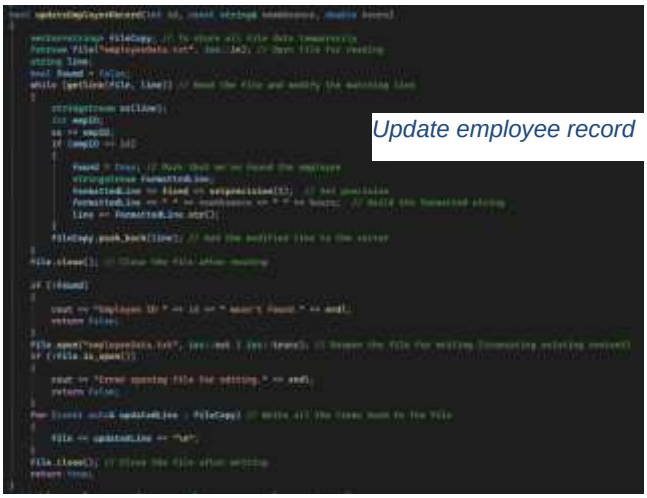


Open file test

Search employee by ID function

```
void absenceType()// Menu to display the absences types
{
    cout << "*****************************************" << endl;
    cout << "1 - Local Leave" << endl;
    cout << "2 - State Leave" << endl;
    cout << "3 - Unpaid Leave" << endl;
    cout << "4 - Workers Comp." << endl;
    cout << "5 - Off Campus Duty" << endl;
    cout << "6 - Jury Duty" << endl;
    cout << "7 - Funeral Leave" << endl;
    cout << "8 - ASLT" << endl;
    cout << "9 - COMP Time" << endl;
    cout << "10 - Other" << endl;
    cout << "*****************************************" << endl;
}
```

Absence type menu screen



Update employee record

```
else if (choice == 2)
{
    file.openFileTest(fout, "employeeData.txt");
    addEmployeeEntry(employee);
    file.writeToFile(employee);
    cout << "Would you like to keep running the program? (y/n) \t";
    cin >> keepGoing;
}
```

Moving on to choose number (2) from the main menu which is adding an employee entry. This choice is used in the beginning of the programs "life cycle". I say this because this function is used to input employee information into the employeeData.txt file which will then be manipulated by the rest of the program if needed. Choice number (2) also uses the open file test function to see if the file is able to be opened. As you can see from the snippet the open file test and the write to file function in choice (2) are both in the same (FILE) class. This class contains all the functions that manipulate the file in some sort of way. Both classes will be show at the end of the file for reference. Following the open file test function is the

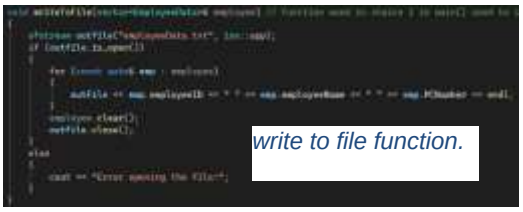
```
void addEmployeeEntry(vector<EmployeeData>& employee)
{
    EmployeeData NewEmployee;
    cout << "Enter Employee Name: ";
    cin.ignore();
    getline(cin, NewEmployee.employeeName);
    cout << "Enter Employee ID: ";
    cin >> NewEmployee.employeeID;
    cout << "Enter Employee PC#: ";
    cin >> NewEmployee.PCNumber;

    NewEmployee.hours.resize(10, 0);
    employee.push_back(NewEmployee);
}
```

Add employee entry function

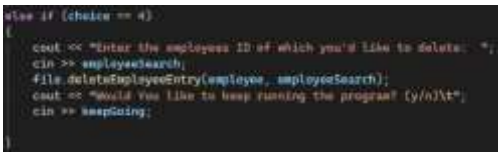
```
struct EmployeeData //STRUCT DATA
{
    string employeeName;
    int employeeID;
    int PCNumber;
    vector<string> absences;
    vector<double> hours;
};
```

add employee entry function. This is the only function that I decided not to in a class because I didn't think it was necessary and it is not a menu screen, and it doesn't modify the file in any way. This function operates by using the employeeData struct. This struct is used to store all the employee data in a vector that is a member in the employeeData struct and this is initialized in the main function. After the employee data is entered by the user then choice(2) if statement is then prompted to use the writeToFile function. The write to file function opens the file for appending, and if the file is opened then the employee's information is added to the file followed by the endl; so that each employees information is separated by the new line member. After the employee's information is written to the file the user is asked if they would like to keep the program running. Of course, if anything other than the character (y) is entered the program will close.



write to file function.

Choice (3) from the main menu function is used to only display the employes information to the using in the console buffer. The program will ask the user to enter the ID of the employee who they'd like to search for. After the ID is entered the searchEmployeeById is run and the employee's information will be shown. If the employees ID can not be found the program will tell the user and the program will ask if you would like to keep going.



Choice (4) is used to delete an employee from the employeeData.txt file. The user is prompted to enter the ID of the employee whos information they like to delete. First the function tests if the file can be opened. Then the file is scanned and every employee who doesn't match the desired ID will be pushed back to the vector. Once the end of the file is reached (\eof) if the Id was not found the program will tell the user. Either way the file is then given the contents of the vector and the file is closed.

Lastly, Choice (0/Zero) is used to quit the program. The program simply tells the user that the program will quit and the break; is used to exit the program.

```
else if (choice == 0)
{
    cout << "Quitting Program";
    break;
}
```

Classes: Main menu class is located on lines 18-47
File operation class is located on lines 49-199

Loops: located on lines 71-74(for loop), 89-101(while loop), 116-130(while Loop), 144-147 (For Loop), 163-176 (While loop), 190-193 (For Loop), 226-324(Do-While loop)

Try…Catch Exception: Lines 54-64

What have you learned from this project?

This program has taught me many things ranging from learning how to properly organize code so that is it readable, to learning how important adding comments to your code is. Before attempting this project, I never previously added comments to my code because I thought that since this is my own code I would be able to understand it but as I was working on the code and would take breaks, I would come back to my code and would confuse myself because couldn't remember what I was trying to do and would end up deleting chunks of my code because I didn't even know what I was trying to do. After going back and inserting comments to describe what certain lines of code did, I found myself easily remembering what my train of thought was. Duning my project I also had to research and stumbled across string streams which I was able to implement into my code. Overall I think that during this project I was able to prove to myself that I could complete a whole project from start to finish.

Website I used for reference about string streams

[stringstream in C++ and its Applications - GeeksforGeeks]

Class Code snippets:

```
class MenuScreens // data class to handle both menu screens
{
public:
    void mainMenuScreen()//Main menu function screen to user
    {
        cout << "*****************************************" << endl;
        cout << "\t Main Menu" << endl;
        cout << "1 - Add Employee Absence" << endl;
        cout << "2 - Create New Employee Entry" << endl;
        cout << "3 - View/Search for Employee Information" << endl;
        cout << "4 - Delete an Employee from database" << endl;
        cout << "0 - Quit Program" << endl;
        cout << "*****************************************" << endl;
    }
    void absenceType()// Menu to display the absences types
    {
        cout << "*****************************************" << endl;
        cout << "1 - Local Leave" << endl;
        cout << "2 - State Leave" << endl;
        cout << "3 - Unpaid Leave" << endl;
        cout << "4 - Workers Comp." << endl;
        cout << "5 - Off Campus Duty" << endl;
        cout << "6 - Jury Duty" << endl;
        cout << "7 - Funeral Leave" << endl;
        cout << "8 - ASLT" << endl;
        cout << "9 - COMP Time" << endl;
        cout << "10 - Other" << endl;
        cout << "*****************************************" << endl;
    }
};
```