Object Oriented Programming – Fall 22

(BS-CS-F22)

Lab-4

Lab Instructor: Maa'm Sanam Ahmad

Instructions:

- **Indent your code properly.**
- **Use meaningful variable and function names.**
- **Use the camelCase notation.**
- **Use meaningful prompt lines/labels for all input/output.**
- **Do NOT use any GLOBAL variable(s). However, global named constants may be used.**
- **This is an individual lab, you are strictly NOT allowed to discuss your solution with fellow colleagues, even not allowed to ask how is he/she is doing, it may result in negative marking. You can ONLY discuss with your TAs or with me. • Anyone caught in an act of plagiarism would be awarded an "F" grade in this Lab.**

Do Validations on inputs where required otherwise 1 mark will be deducted for every wrong validation.

TASK-1:

Task Description:

You are tasked with creating an Employee Management System. Define an Employee class with the following specifications:

Data Members:

- ID (an integer to uniquely identify each employee)
- Name (a **c-string** of size 31 to hold the employee's name)
- Dept. Name (a **c-string** of size 41 to hold the employee's department name)

Note 1: *The sizes of the two c-strings MUST be declared as* ***named constants*** *before the start of the class declaration.*

Note 2: *All member functions of the* `Employee` *class, which are not supposed to change datastore in the calling object, should be declared as* `const` *member functions*

Constructor:

- Implement a **Default constructor** for **`Employee`** class that assigns 0 to the ID, and empty c-strings to the Name and Department name.

- Implement an **Overloaded constructor** for **`Employee`** class that accepts the following 3 values (in the given order) as arguments and assigns them to the appropriate member variables: employee's ID, employee's name, and department name.

Member Functions and Driver:

- Implement the getter and setter functions for each member variable of the **`Employee`** class.

- Implement a member function **`void display ()`** of the **`Employee`** class which should neatly display all attributes of an employee on screen. In this function, **you MUST use the getter functions** to get the values of all attributes of an employee.

- In the main function, dynamically allocate an array of **`Employee`** objects. Your program should ask the user about the size of the array and then allocate the array dynamically. Then, your program should ask the user to enter the values of all attributes for each **`Employee`**. Afterthat, your program should display the details of each object on screen (by calling the **`display`**function for each **`Employee`**).

- Implement a public member function **`void storeInFile(ofstream&)`** of the **`Employee`**class that stores all information of an Employee in the text file which has been opened throughthe file handle which is passed into this function. Now, enhance the main function to store all **`Employee`** objects from the array, which you created above in step **above** into a text file named **`Emps.txt`**. Decide about the format of the fileyourself, but **make sure that all necessary information is stored in the file in a manner sothat it can be read later on** (see the next step).

  Now, deallocate the dynamically allocated array of **`Employee`** objects which you created above.

- Implement a public member function **`void readFromFile(ifstream&)`** of the **`Employee`** class that reads all information of an Employee from the text file (which has been opened through the file handle which is passed into this function) and stores this information in the member variables of the Employee object on which this function has been called.

  Write a driver program (main function) which should open the text file **`Emps.txt`** (which youcreated in step **1.6**) and dynamically allocates an array of **`Employee`** objects after reading thecount of **`Employee`** objects from the text file. After that, your program should read the information of all **`Employee`** objects and store it into the dynamically allocated array. The program should, then, display all information on screen.

  At the end, your program should deallocate all dynamically allocated memory.

Very important information: Carefully deallocate memory in the destructor as well as array of students in main function make pointer nullptr avoiding pointers to become dangling pointers.

In case of a pointer

If(ptr!=nullpr)

{

delete ptr; ptr=nullptr;

}

// in case of dynamically allocated array

If(ptr!=nullptr)

{

delete [] ptr;

ptr=nullptr;

}

Sample Files code

Files

```
#include <iostream>
#include <cstring>
#include <fstream>
#include <string>
using namespace std;
int main() {
	ofstream fout("myfile.txt",  ios::out|ios::ate);
	if (!fout.operator bool() )
		cout << "File handling error " << endl;

	else
	{
		for (int i = 0; i < 10; i++) {
			fout << "This is Nabeel" << endl;
			fout << i << endl;

		}
	}

	fout.close();
		char str[30]; // declare a character array(cstring)
		int number;
		ifstream fin("myfile.txt", ios::in);
```

```
        if (!fin) cout << "File is not found" << endl;
        else{
            while (fin.getline(str, 30)) {

                fin >> number;
                cout << str << endl;
                cout << number << endl;
                fin.ignore();

            }
        }

    return 0;
}
```

```
#include <iostream>
#include <cstring>
#include <fstream>
#include <string>
using namespace std;
int main() {
    fstream file("myfile.txt", ios::in | ios::out|ios::ate);
    if (!file.operator bool() )
        cout << "File handling error " << endl;

    else
    {
        for (int i = 0; i < 10; i++) {
            file << "This is Nabeel" << endl;
            file << i << endl;

        }
    }

    file.seekg(0, ios::beg);

        char str[30]; // declare a character array(cstring)
        int number;

        while (file.getline(str, 30)){

            file >> number;
            cout << str << endl;
            cout << number << endl;
            file.ignore();

    }
    return 0;
}
```

Cstrings

```
#include <iostream>
#include <cstring>
using  namespace std;
int main() {
    const int N{ 100 };
    char name[N];
    cin >> name;
    cout << name << endl;
    cin.ignore();
```

```
        cin.getline(name, N);
        cout << name << endl;
        return 0;
}
```