BBM-203 ASSIGNMENT-2

UMUT GÜNGÖR 21946198

29.11.2021

In this project, I am asked to do the Employee Recording System with using Linked Lists. There are two types of employees. Temporary employee and permanent employee. They both are inherited an employee superclass. And there are also two types of linked lists. Array implementation of circular linked list and Doubly Linked List. Temporary employees store in circular linked list and permanent ones store in dobuly linked list. Also, problem includes that sort,merge and find the employees in particular order. In some cases update them too. The goal of the project is teach linked list implementation and its benefits as well as class structure. One another thing is that how to implement operator overloading and how to use it.

My methods to solve this problem is using a modular approach which includes bunch of classes and several functions in them. For instance, in my project there are six different classes. Each of them stores seperate data. However, they have connection with each others.

My project plan is solve the problem with little piece of classes and functions. I need different classes and relations between these classes, some functions to do the such operations which are find, merge, sort and display.

My first impression of the problem is I have to use classes, inheritance and some functions to manage the data. Because, the linked list implementation in c++ I need structure like class. In class of employees' and linked lists' there are functions, attributes and relations. Using these strategies I try to implement an object oriented program.

I create 6 classes with some functions and attributes. And also, there are seperate functions they are not a member of any class. I split the problem into the small pieces to reduce the complexity However, this problem could have been done using only functions. This solution contains much more complexity. I have Employee class which is superclass of the TemporaryEmployee and PermanenetEmployee classes. A Date class which stores date of the some data such as birth date and appointment of the employee. Finally, there are two seperate linked list classes. CircularArrayLinkedList and DoubleDynamicLinkedList. CircularArrayLinkedList class have TemporaryEmployee objects in an array. DoubleDynamicLinkedList class have PermanentEmployee objects in dynamic memory allocation way with nodes.
All of these classes have interfaces so header files. In hedar files, there exist class declaration, its functions and attributes declaration.

First of all, I create an Employee class. It has attributes such as name, surname, employeeNumber and so on. I initialize these members in constructor with initializer list. PermanentEmployee and TemporaryEmployee classes are subclasses of Employee class. So, they both have same members that they are in Employee class. But, for sure they

have different properties too. There are two type of linked list in my implementation. First one is CircularArrayLinkedList. In this class there is an array which stores the data of the TemporaryEmployee class' objects. It includes other functions. Insert function add employee to the class, Delete function remove employee from the class, Find function if there is an employee with spesific employee number. Second one is DoubleDynamicLinkedList. This class has nodes. Node has three members. data section stores PermanentEmployee class' object. prev and next members point to the previous and next objects respectively. Lastly, Date class has some members and functions. It can convert the date comes from user as string to the avaliable form. Date class overloads compare operators.

I spent most of my time to detect the errors and fix them. Analysis of the problem is not hard but implementation is.

```
class DoubleDynamicLinkedList
{
public:
    Node* head = NULL;
    Node* getNewNode(const PermanentEmployee& permanentEmployee);
    void Insert(const PermanentEmployee& permanentEmployee);
    bool Find(int employeeNumber);
    PermanentEmployee& Get(int employeeNumber);
    void Update(PermanentEmployee& permanenetEmployee, string title, double salaryCoefficient);
    void Delete(int employeeNumber);
    void Display(int employeeNumber);


    friend ostream& operator<<(ostream& output, const DoubleDynamicLinkedList& list);          //
implemented at cpp

};
```

In this part of my source code the declaration of DoubleDynamicLinkedList has been made. This is in header file so there is not any function body. This class overloads the shift(<<) operator.

```
#ifndef Employee_hpp
#define Employee_hpp

#include <iostream>
#include "Date.hpp"
using namespace std;


class Employee
{
protected:
    const int m_employeeNumber;
    int m_employeeType;         // 1 for permanent and 0 for temporary
    const string m_name;
    const string m_surname;
    string m_title;
    double m_salaryCoefficient;     // These are initialized by constructor
    const Date m_birthDate;

    Date m_dateOfAppointment;               // These can be accessed by Set functions if neccessary
    int m_lengthServiceOtherInstitution;


                                // When calling constructor first instantiate date class
public:
    Employee();
    Employee(int employeeNumber, string name, string surname, string title, double salaryCoefficient, Date birthDate) :
    m_employeeNumber {employeeNumber}, m_name{name}, m_surname{surname}, m_title{title}, m_salaryCoefficient{salaryCoefficient}, m_birthDate{birthDate}
    {}

    int GetEmployeeNumber() const {return m_employeeNumber;}
    int GetEmployeeType() const {return m_employeeType;}
    string GetName() const {return m_name;}
    string GetSurname() const {return m_surname;}
    string GetTitle() const {return m_title;}
    void SetTitle(string title) {m_title = title;}
    double GetSalaryCoefficient() const {return m_salaryCoefficient;}
    void SetSalaryCoefficient(double salaryCoefficient) {m_salaryCoefficient = salaryCoefficient;}
    Date GetBirthDate() const {return m_birthDate;}     //reference dondurmek gerekebilir

    Date GetDateOfAppointment() const {return m_dateOfAppointment;}     //reference dondurmek gerekebilir
    void SetDateOfAppointment(Date dateOfAppointment) {m_dateOfAppointment = dateOfAppointment;}
    int GetLengthServiceOtherInstitution() const {return m_lengthServiceOtherInstitution;}
    void SetLengthServiceOtherInstitution(int lengthServiceOtherInstitution) {m_lengthServiceOtherInstitution = lengthServiceOtherInstitution;}

    Employee& operator=(Employee const& other);

    friend ostream& operator<<(ostream& output, const Employee& emp);

};

ostream& operator<<(ostream& output, const Employee& emp);

#endif
```

This if the full content of my Employee header file. It has protected members which can accessed by child classes. There are public methods and constructors. Setters and Getters. Also this class overloads the shift operator too.

```
#ifndef Date_hpp
#define Date_hpp

#include <iostream>

using namespace std;

class Date
{
private:
    int m_day;
    int m_month;
    int m_year;
    int m_appointmentDay;
    int m_appointmentMonth;
    int m_appointmentYear;

public:                     // split - to day month year
    Date();
    int GetDay() const {return m_day;}
    int GetMonth() const{return m_month;}
    int GetYear() const {return m_year;}
    int GetAppointmentDay() const {return m_appointmentDay;}
    int GetAppointmentMonth() const {return m_appointmentMonth;}
    int GetAppointmentYear() const {return m_appointmentYear;}

    void PrepareDate(string unpreparedDate);
    void PrepareAppointmentDate(string appointmentDate);

    bool operator<(Date const &obj);
    bool operator>(Date const &obj);
    bool operator<=(Date const &obj);
    bool operator>=(Date const &obj);
    bool operator==(Date const &obj);

};



#endif
```

This is the Date class' header file. It has private members. There are a constructor getter and setter methods and other functions. This class overloads all of the comparison operators in order to compare the two given dates.

User can use my code for manage the small employee recording system. There are some restrictions. In many cases this code may throw the error. Because, the information comes from the user and user can type the wrong inputs. For instance, if string type would have been written for employeeNumber there will be an error. User should follow the instructions and should write the date in this form DD-MM-YYYY.

I used divide and conquer methodology. Split my code blocks into the smaller ones. When user type all the information correctly this employee adds to the proper data structure. If user wants to show employees all information of the employees will be shown. In some points my code can crush so, I am not sure which point i will get.

References:

stackoverflow.com
BBM-201 lecture notes
geeksforgeeks.org
learncpp.com
cppreference.com