BBM203:
Software Laboratory 1
2021 Fall
ASSIGNMENT-2
REPORT

Name: Mert Ali

Surname: Yalçın

Number: 21946682

Delivery date: 28.11.2021

Due date: 30.11.2021

Language: C++

1. Introduction

In this assignment, we are expected to write an “Employee Recording System” program. We will design it as a console menu application program and we will use our own data structures. This project is aiming to teach us designing well-organized class structures and implementing data structures in our own way.

2. Method

Since we are assigning employees to the system. Firstly, I created a class called “Employee”. It has many fields and functions in it. There are two types of employees: Temporary and Permanent. They have common fields and functions in lots of ways so I derived them from “Employee” class. Now, we need to store them in two different data structures. Circular Array Linked List for temporary employees and Double Dynamic Linked List for permanent employees. Since temporary employees have limited number (maximum: 20), I choose the array implementation of a linked list. Number of maximum permanent employees is unknown, so I choose using a double linked list with its dynamic implementation. Later on, I have implemented some other functions for required operations and used them in my main source block.

3. Development

3.a. Plan:

Firstly, I had to research some topic regarding to my project.

How can I write a console menu program? How to overload operators?(*1) Etc. After I did my research, I started to create classes and create fields and functions in it. I started from the smallest component of my design and later on used it in much bigger components. This kind of modularity in my design provides me enormous help when I am debugging. Details of my design will be explained in a later section.

3.b. Analysis:

Since the program will always come back to the main menu, it is ok for me to use an infinite while loop. User will be exiting from the program by closing the window. Program has 12 functionalities and each of them will be accessed by requesting the corresponding function number from the user. After getting the number from the user, rest of the work will be done by executing the functionality that the user requested. All of this functions can be done in separate branches of a multiple if statement. With the help of some helper function, an easy to read main source block can be created.

3.c. Design:

Firstly, I have created a “Date” class. It has its own day, month, year attributes and a special way of comparison. This class is a field type of “birth” and “appointment” fields which

is in the “Employee” class. This class is the main element of our project. It has all the necessary fields, getters, setters and overloaded operators. The next two classes “TemporaryEmployee” and “PermanentEmployee” are derived from this class. These two classes have their own functionalities but they act like “nodes” to our self-implemented data structures. These data structures are “CircularArrayLinkedList” and “DoubleDynamicLinkedList”. They store the necessary data and perform requested operations for our desired methods in our program.

Lastly, the program takes necessary input from the user and executes the requested functionality. In below, you can see the abstract data types that I needed to use for my solution:

- CircularArrayLinkedList.h
- Date.h
- DoubleDynamicLinkedList.h
- Employee.h
- PermanetEmployee.h
- TemporaryEmployee.h

3.d. Implementation

Firstly, program creates two lists (CircularArrayLinkedList and DoubleDynamicLinkedList) then enters an infinite loop. Beginning of this loop is my main menu, it shows all operations and their corresponding numbers. Then it takes a number that the user inputs and executes that operation:

1. Appointment of an employee,

User provides all of the necessary inputs for the addition of an employee and then employee is inserted to its corresponding list.

2. Appointment of a transferred employee

This operation does the exact same operation as the first one but it has a little difference. Since the employee is transferred, this operation asks the length of service in other institutions.

3. Updating the title and salary coefficient of an employee

Operation takes an employee number updates its two fields, title and salary coefficient.

4. Deletion of an employee

Program finds employee with the given number deletes it from the list that it belonged.

5. Listing the information of an employee

It takes the desired employee number and list its information in the given order.

6. Listing employees ordered by employee number

This operation calls my helper method printAllByNum. This method takes two parameters (the two lists) and merges them. Then it sorts the merged list and prints out the information of employees. From now on, all of the operations will call a helper method that I created. The method will take the necessary parameters and then executes the same working principle by slight differences according to the

desired output and then prints out the information of all the elements in the merged list.

3.e. Programmer Catalog

Analysis, design, testing, reporting = each took half a day.

Implementation = roughly took one and a half week.

Other programmers can use my program on their preferred data type. For example, my program can be used in a school for storing the information of students and other workers in school. It will only require desired object type for the proper situation. In the example I have given, you should alter the employee class and use it according to a student's information. In below, you can see the description of all data members and classes in my program:

- Date.h

Private:

//each of the fields in this section has a getter and a setter

Int day, month, year; // fields of a date.

Public:

Date();

Date(int d, int m, int y);

Date(string emp_date);

//Date class has 3 different constructors. Each has its proper //place and time

Friend bool operator <, >, ==, <<;

//Each of the operators above have been overloaded for their
//proper uses. Comparison operators compares years. If years
//are equal it compares months and if they are equal, it
//compares days.

// << operator has been overloaded to provide us the desired
// output format: “dd-mm-yyyy”

- Employee.h

Private:

//each of the fields in this section has a getter and a setter.

int number;
int type;
string name;
string surname;
string title;
string salaryCo; // salary coefficient
Date birth;
Date appointment; // date of appointment to the institution.
int service; // length of service in other institutions.

Public:

// Has two constructors, one is default and other is to be used
//in operation 1 and 2.

Employee(int number, int type, string name, string surname,
string title, string salaryCo, string birth, string appointment,
int service);

Employee();

Friend bool operator <<, <, >;

// << operator is overloaded to print out every field of the
//employee in the given order.

//comparison operators overloaded to compare two
//employees by their numbers.

- TemporaryEmployee.h

//This class inherits from Employee.

//It has same constructors as Employee.

//Default constructor is used for testing

Public:

İnt next;

//this fields will provide this class to act like a node in
//CircularArrayLinkedList

- PermanentEmployee.h

//This class inherits from Employee.

//It has same constructors as Employee.

//Default constructor is used for testing

Public:

PermanentEmployee* prev;

PermanentEmployee* next;

//this fields will provide this class to act like a node in
//DoubleDynamicLinkedList

- CircularArrayLinkedList.h

Public:

```
TemporaryEmployee* listArray;
int head; // tracks the head of the list.
int avail; // shows the next available slot in the list
CircularArrayLinkedList(); // used for testing
CircularArrayLinkedList(int size); // can increase max size
~CircularArrayLinkedList(); // destructor

//Adds the item to the end of the circular list
void append(TemporaryEmployee* e);

void printList();

//Ckecks if array contains employee with given number
bool contains(int num);

//Insert employee to the circular array according to its
//number
void insert(TemporaryEmployee* e);

//Updates salary coefficient and title of an employee
void update(int emp_num, string newSalCo, string newTitle);

//Delete the employee
void deletion(int emp_num);

//Information of the employee
void info(int emp_num);
```

- DoubleDynamicLinkedList.h

```
public:
    PermanentEmployee* head; //tracks the head of the list
    DoubleDynamicLinkedList(); // used for testing

   //inserts the element at the head of the list
    void insertFront(PermanentEmployee* e);

   //inserts the element at the back of the list
    void insertBack(PermanentEmployee* e);
    void printList();

   //checks if the list has wanted element
    bool contains(int num);

    //Updates salary coefficient and title of an employee
    void update(int emp_num, string newSalCo, string
newTitle);

    //Deletion of the employee
    void deletion(int emp_num);
```

```
//Information of the employee
void info(int emp_num);


//returns the length
int length();
```

- Main.cpp

```
//main.cpp has some helper methods to provide a cleaner
//source code

//used for method: 6
void printAllByNum(CircularArrayLinkedList* cList,
DoubleDynamicLinkedList* dList)

//compares the date of appointment for employees

bool comparator(const Employee& e1,const Employee& e2)

//used for method: 7
void printAllByDate(CircularArrayLinkedList* cList,
DoubleDynamicLinkedList* dList)

//used for method: 8
void printAllAfterDate(CircularArrayLinkedList* cList,
DoubleDynamicLinkedList* dList, string emp_date)

//used for method:9
void printAllInYear(CircularArrayLinkedList* cList,
DoubleDynamicLinkedList* dList, string emp_year)

//used for method:10
void printAllBeforeDate(CircularArrayLinkedList* cList,
DoubleDynamicLinkedList* dList, string emp_date)
```

//used for method:11
void printAllInMonth(CircularArrayLinkedList* cList, DoubleDynamicLinkedList* dList, string emp_month)

//method:12
void printTitle(CircularArrayLinkedList* cList, DoubleDynamicLinkedList* dList, string emp_title)

3.f. User Catalog

Since my program has no exception handling mechanism. It is mandatory for the user to provide correct inputs for each asked input. To give an example, “Enter number:” statement asks user to provide an integer for number. If the user enters “abcd” the program will crash and close. Other important thing is that user must enter the input “Date” in the format of “dd-mm-yyyy” (mind the “-“). If the program takes these properties in a different order, the operations executed later on might not be correct for the user. Other than these restrictions, user can use my program by just altering the Employee class and other classes depending on it. As I told before, this program can be used in school to keep track of the student’s informations. Another example of the usage of this program can be Movie Database System. By changing the Employee Class to Movie and deriving sub-classes for Movie class can be made easily. Rest is just the alteration of the fields and the program can be use for storing movies and their information. Many other uses are available.

4. Results, discussion and conclusion

In this assignment, we have designed and implemented many things but to me, the most useful thing was operation overloading. I had no idea such a thing exists before and it came in very in my solution. This will be a technique that will use many times in my future projects. I like using as many functions as I can in my projects because it provides modularity to the overall design of the project. Thus, it provides a much easier debugging process when I am finished with my project. The one thing that bothered me was using multiple “cin” operators back to back. Because it counts “enter” as newlines, it caused some unwanted newlines when I got input from the user. I used some “endlines” just before of these unwanted newlines. The final look of the input taking procedure is not what I completely wanted but is as good as I can make it. Lastly, if I have to evaluate my overall assignment according to the evaluation table given in the pdf, here is my result;

- Efficiency of the program: 5/5

Most of the operations has O(NlogN) complexity. The worst complexity that I have is O(N).

- Compliance with OOP principles: 25/25

I think I have used principles of OOP correctly in this assignment such as inheritance, encapsulation, polymorphism etc. I have utilized most of them and It was helpful in my designing process.

- Correctly working parts: 45/45

I have tested my program on DEV server of Hacettepe for every operation. It worked correctly.

- Report: 25/25

I have paid attention to everything required in the given report format. I think I have clarified most of the asked questions and answered them as intended.

5. References

- BBM201 Lecture Notes
- BBM203 Lecture Notes
- (*1) https://www.geeksforgeeks.org/operator-overloading-c/