ASSIGNMENT 2

BBM203

ATA BUDAK
2200356033
30.11.2021

Introduction

The goal of the project was to implement a console application which will act as a menu for assigning new employees to the system and performing numerous operations on them. This goal was to be achieved through usage of doubly dynamic linked list and circular array implementation of linked list.

Method:

I have several methods to address this problem. These methods can be found in the classes DoublyLinkedList and CircularArrayLinkedList. To give some examples for these methods, there are insert and delete functions to modify the lists, sort functions to keep the list ordered in a certain manner, and many more functions to traverse and perform operations on the lists which will be explained in more detail later in this report. Plus, operators such as '<<','>=','<=','==','<','>' are overloaded as a necessity for this assignment. Also, Inheritance is used to gather Temporary Employee and Permanent Employee classes in a parent class called Employee for convenience and for organization. In addition, if a need for a function which will affect both child classes arises, Employee class can include that function.

Development

My plan for this project was to first create my classes which are Employee with the child classes Temporary Employee and Permanent Employee, Date class whose birthdate and appointmentdate objects are then hold as a member in the Employee child classes, CircularArrayLinkedList class which will have the functions that I will perform on my array implementation of circular linked list, and DoublyLinkedList class which holds functions that works on the doubly linked list.

After I have created my classes, I have done analysis on my functions and their efficiency. I mostly analyzed sortList() and traverseList() functions which are O(n^2) and O(n). I found these time results acceptable for the solution of this problem.

My next approach to the solution was to create the menu of the application which consists of 12 different options the user can choose. Each option uses different functions of the circularLinkedList (for Temporary Employee) and DoublyLinkedList(for Permanent Employee) classes. These options and how my functions handle them will be explained now.

1. and 2. Operations are operations which the user can insert employee's to the system. Through the use of console interaction, the type of employee is obtained and according to the type, Temporary and Permanent Employee classes' insert functions are called. (insertEmp() for temporary and insertAtFront() for permanent employee). Right after the insert functions, both lists are sorted with sortList() functions so that they remain ordered. This can also be done in a different manner. The lists could have been sortened inside the insert() functions by checking the order at insertion time and adding the new employee to the correct place. This method wasn't used in my assignment but it was a possible alternative. The reason I eliminated this method that if this project is used in the future and modified, to change the behaviors of the list a whole new

insert function has to be rewritten, on the other hand, the way it is now, it is sufficient to just change the aspect of the sort method in the sort function, which is much more efficient than rewriting a new insert function.

3. operation allows the user to change the title and the salary of the employee with the given id. This operation is performed with the basic setter methods used inside the updateEmp(id,title,salary) function for temporary employee assisted with the empExists(id) function to check there is an employee with the given id. Similar process takes place with permanent employee with the use of editEmp(id,title,salary) and searchEmp(id) function.

4. operation is the deletion of an employee with the given id. deleteEmp(id) function is used for deleting temporary employees. For deleting permanent employees, removeNode(Node n) and returnEmp(id) functions are used. returnEmp(id) function returns the list node which contains the employee with the given id and removeNode(Node n) takes it as a parameter and deletes this node. Thus, the permanent employee is deleted.

5. operation is listing the information of the employee with the given id. printEmp(id) function is used to print temporary employee's and searchAndPrint(id) function is used to traverse the list for the employee with the given id and print it.

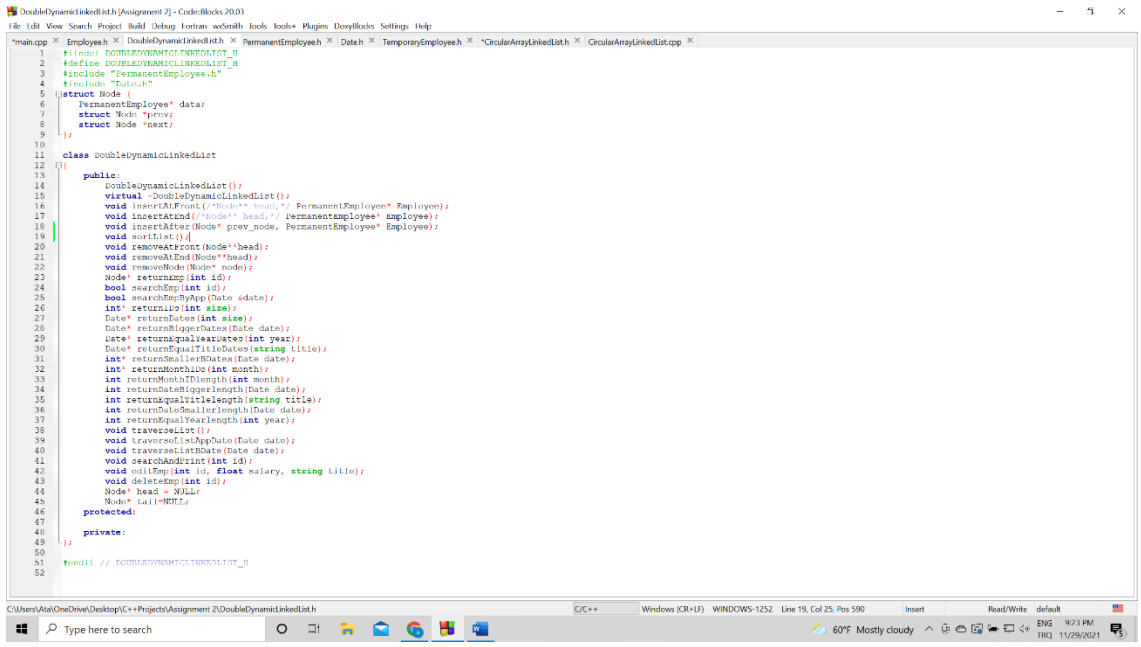
6. operation is listing all the employees in ascending order of their id numbers. To be able to sort all employee's, both lists were traversed and their id's were stored in a temporary array. Then, this temporary array is sorted with the use of std::sort() function which comes with the #include <bits/stdc++.h> library. After the array is sorted, for each id, both temporary and employee list is traversed, and if there is a match with the id (there must be) it is printed. This was achieved with using printEmp(id) and searchAndPrint(id) functions mentioned above.

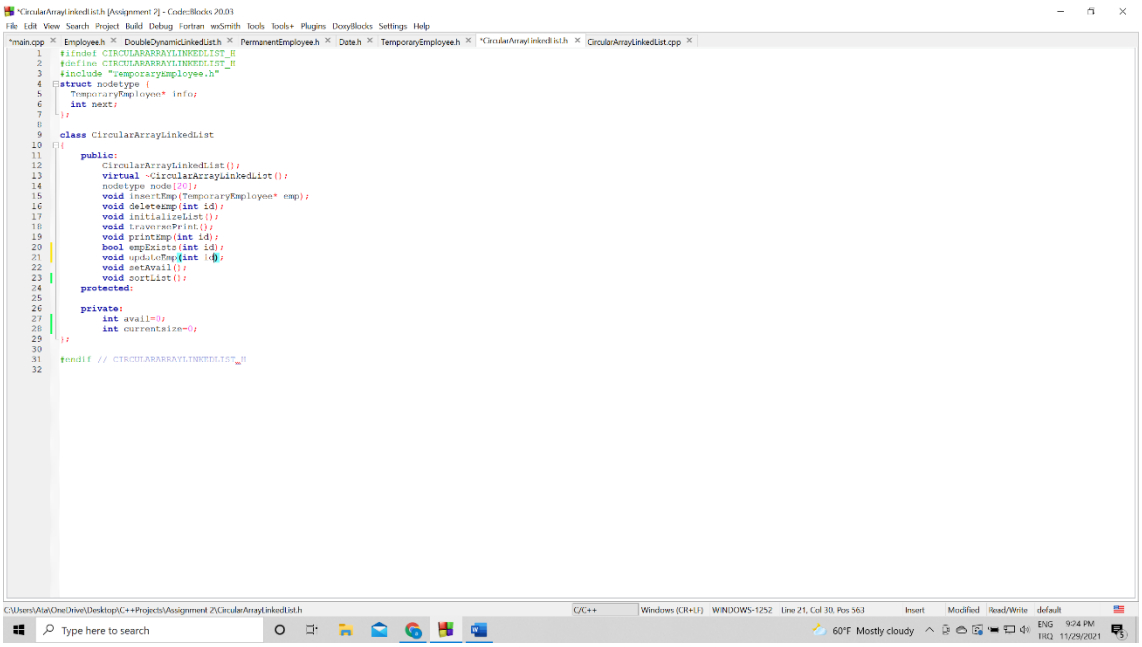
Similar approach was used for operations 7,8 and 9. Instead of storing id's of the employees in an array, their dates were stored, sorted with a custom sortByDate() function which is found in the main class is used, and again both lists were traversed by using traverseListAppDate(date), traverseList(date) to check if there is a matching date (there must be) and print it. There are helper functions such as return…length() and return…Dates() that helps to store Date objects of permanent employees in the temporary array.

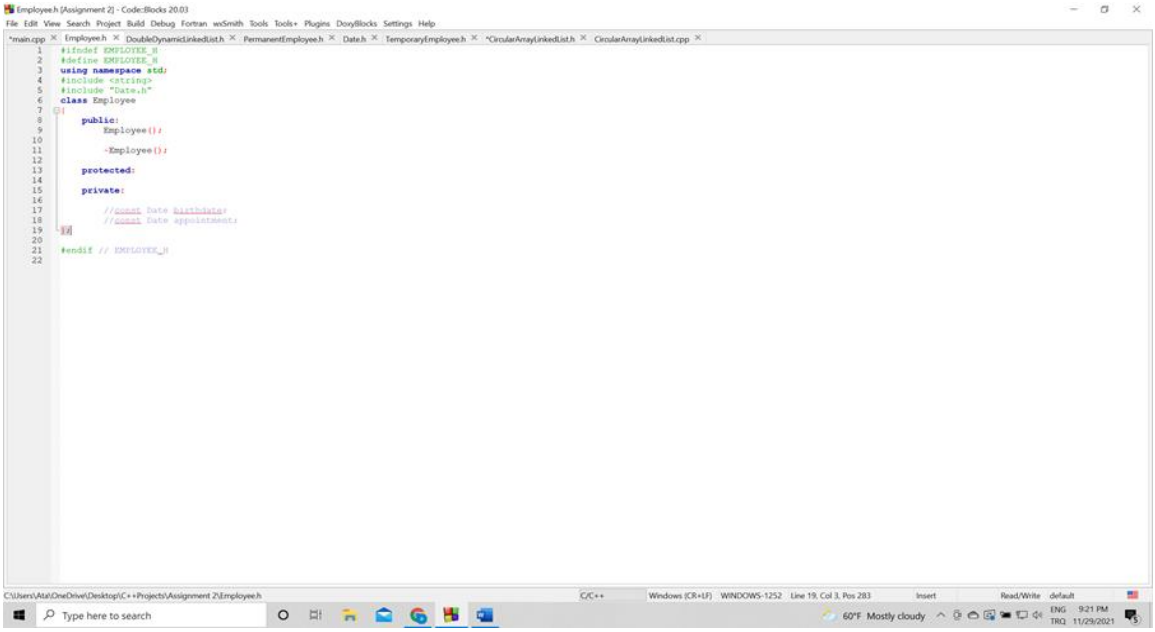
Operation 10 is listing employees born before a certain date. Date is taken from the user with console interaction and both lists are traversed to find employees born before the date. Date class overloads < operator so that comparison between 2 date classes can be done. Since the assignment requires these employees that are born before the date to be ordered by ascending id, these employee's id's are stored in a temporary array. Array is then sorted and both lists are traversed again with printEmp(id) and searchAndPrint(id) methods to print the employees.

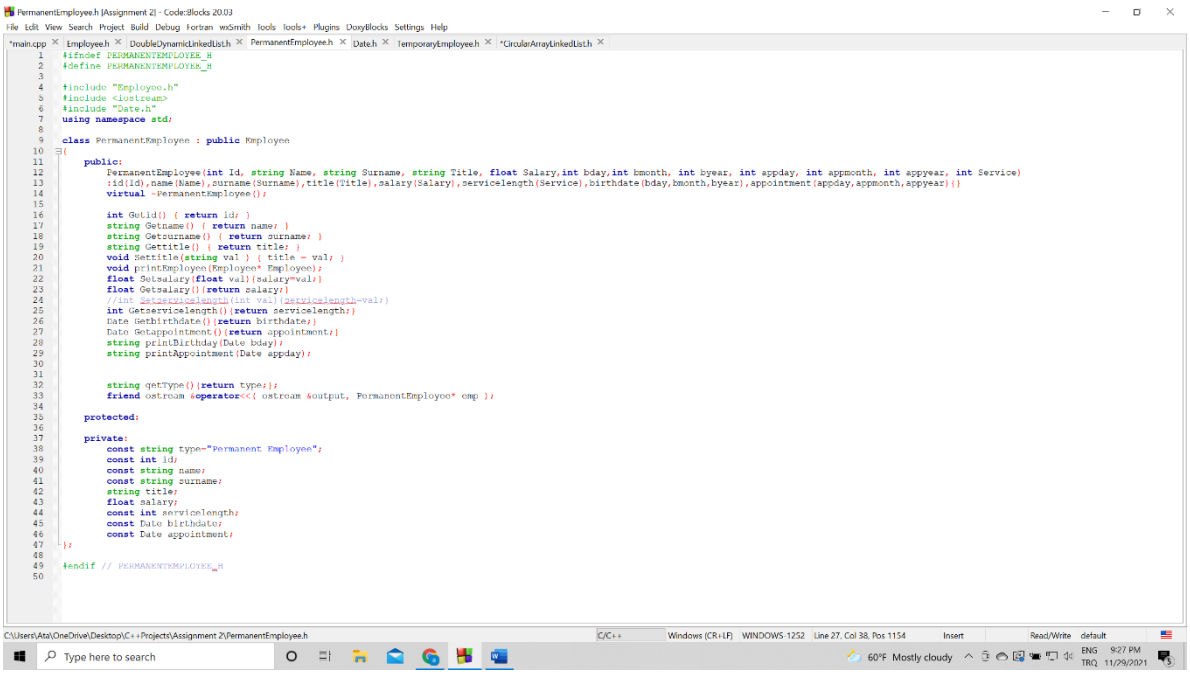
Operation 11 is the same with 10. operation except not a whole date but a month is taken from the user and the usage of functions are the same with 10. operation.

Operation 12 allows the user to display the last assigned employee with the given title. Title is obtained from the user through console application. Then both lists are traversed and the appointment dates of the Employee's with matching title's are stored in the array. Then, the array is sorted with sortByDateDescending() custom function. Finally both lists are traversed and the employee with the matching date of the first element of the date array (last appointment date) is printed. Helper functions such as returnEqualTitleDates() and returnEqualTitlelength() functions are used to store dates in the array. Also traverseListAppDate(date) is used to find the last assigned employee. For better understanding, header file implementations are below.






Employee.h [Assignment 2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
*main.cpp Employee.h DoubleDynamicLinkedList.h PermanentEmployee.h Date.h TemporaryEmployee.h *CircularArrayLinkedList.h CircularArrayLinkedList.cpp
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
using namespace std;
#include <string>
#include "Date.h"
class Employee
{
public:
Employee();
~Employee();
protected:
private:
//const Date birthdate;
//const Date appointment;
};
#endif // EMPLOYEE_H
C:\Users\Ata\OneDrive\Desktop\C++Projects\Assignment 2\Employee.h
C/C++ Windows (CR+LF) WINDOWS-1252 Line 19, Col 3, Pos 283 Insert Read/Write default
Type here to search
60°F Mostly cloudy
ENG TRQ 9:21 PM 11/29/2021


PermanentEmployee.h [Assignment 2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
*main.cpp Employee.h DoubleDynamicLinkedList.h PermanentEmployee.h Date.h TemporaryEmployee.h *CircularArrayLinkedList.h
#ifndef PERMANENTEMPLOYEE_H
#define PERMANENTEMPLOYEE_H
#include "Employee.h"
#include <iostream>
#include "Date.h"
using namespace std;
class PermanentEmployee : public Employee
{
public:
PermanentEmployee(int Id, string Name, string Surname, string Title, float Salary,int bday,int bmonth, int byear, int appday, int appmonth, int appyear, int Service)
:id(Id),name(Name),surname(Surname),title(Title),salary(Salary),servicelength(Service),birthdate(bday,bmonth,byear),appointment(appday,appmonth,appyear){}
virtual ~PermanentEmployee();
int GetId() { return id; }
string Getname() { return name; }
string Getsurname() { return surname; }
string Gettitle() { return title; }
void Settitle(string val) { title = val; }
void printEmployee(Employee* Employee);
float Setsalary(float val){salary=val;}
float Getsalary(){return salary;}
//int Setservicelength(int val){servicelength=val;}
int Getservicelength(){return servicelength;}
Date Getbirthdate(){return birthdate;}
Date Getappointment(){return appointment;}
string printBirthday(Date bday);
string printAppointment(Date appday);
string getType(){return type;};
friend ostream &operator<<(ostream &output, PermanentEmployee* emp);
protected:
private:
const string type="Permanent Employee";
const int id;
const string name;
const string surname;
string title;
float salary;
const int servicelength;
const Date birthdate;
const Date appointment;
};
#endif // PERMANENTEMPLOYEE_H
C:\Users\Ata\OneDrive\Desktop\C++Projects\Assignment 2\PermanentEmployee.h
C/C++ Windows (CR+LF) WINDOWS-1252 Line 27, Col 38, Pos 1154 Insert Read/Write default
Type here to search
60°F Mostly cloudy
ENG TRQ 9:27 PM 11/29/2021

TemporaryEmployee.h [Assignment 2] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

*main.cpp | Employee.h | DoubleDynamicLinkedList.h | PermanentEmployee.h | Date.h | TemporaryEmployee.h | *CircularArrayLinkedList.h

```
#ifndef TEMPORARYEMPLOYEE_H
#define TEMPORARYEMPLOYEE_H

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

class TemporaryEmployee : public Employee
{
    public:
        TemporaryEmployee(int Id, string Name, string Surname, string Title, float Salary,int bday,int bmonth, int byear, int appday, int appmonth, int appyear,int Service)
        :id(Id),name(Name),surname(Surname),title(Title),salary(Salary),servicelength(Service),birthdate(bday,bmonth,byear),appointment(appday,appmonth,appyear){}
        virtual ~TemporaryEmployee();

        int Getid() { return id; }
        string Getname() { return name; }
        string Getsurname() { return surname; }
        string Gettitle() { return title; }
        void Settitle(string val ) { title = val; }
        void printEmployee(Employee* Employee);
        float Setsalary(float val){salary=val;}
        float Getsalary(){return salary;}
        //int Setservicelength(int val){servicelength=val;}
        int Getservicelength(){return servicelength;}
        Date Getbirthdate(){return birthdate;}
        Date Getappointment(){return appointment;}
        string getType() { return type; }
        string printBirthday(Date bday);
        string printAppointment(Date appday);

        friend ostream &operator<<( ostream &output, TemporaryEmployee* emp );


    protected:

    private:
        const string type="Temporary Employee";
        const int id;
        const string name;
        const string surname;
        string title;
        float salary;
        const int servicelength;
        const Date birthdate;
        const Date appointment;
};

#endif // TEMPORARYEMPLOYEE_H
```

C:\Users\Ata\OneDrive\Desktop\C++Projects\Assignment 2\TemporaryEmployee.h | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 30, Col 46, Pos 1298 | Insert | Read/Write | default

Type here to search | 60°F Mostly cloudy | ENG IRQ 9:27 PM 11/29/2021

Date.h [Assignment 2] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

*main.cpp | Employee.h | DoubleDynamicLinkedList.h | PermanentEmployee.h | Date.h | TemporaryEmployee.h | *CircularArrayLinkedList.h

```
#ifndef DATE_H
#define DATE_H


class Date
{
    public:
        Date(){}
        Date(int a, int b, int c) : day(a),month(b),year(c){}
        virtual ~Date();

         int Getday() { return day; }

         int Getmonth() { return month; }

         int Getyear() { return year; }
        friend bool operator== ( Date a, Date b);
        friend bool operator>= (Date &a, Date &b);
        friend bool operator<= (Date &a, Date &b);
        friend bool operator!= (Date &a, Date &b);
        friend bool operator> (Date a, Date b);
        friend bool operator< (Date a, Date b);

    protected:

    private:
          int day;
          int month;
          int year;
};

#endif // DATE_H
```

C:\Users\Ata\OneDrive\Desktop\C++Projects\Assignment 2\Date.h | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 22, Col 45, Pos 600 | Insert | Read/Write | default

Type here to search | 60°F Mostly cloudy | ENG IRQ 9:30 PM 11/29/2021

A user can use my project if they want to have a system to coordinate their employee's and hold their information. They can also use my project and modify it to fit many other types. For instance they can change the stored objects of employees with kids and make a birthday application where it notifies kids whose birthday has come. The circular linked list can hold the kids with the birthday and the doubly linked list can hold the kids with no birthday. Another implementation might be for school. In 11.th grade most students make a selection between studying abroad or studying in their country. The circular linked list can hold the students going abroad and the doubly linked list can hold students studying in their country. The restrictions for such projects would be the kids with birthday and the students decided to study abroad has to be no more than 20 because of the array implementation style.

Results

Overall I believe this project was very informative and educative to do. Not only it was different to make a console application, it was also very beneficial to get used to linked lists. There were also several restrictions which challenged me to improve myself and come up with new solutions. If I were to grade myself I would have given my project between 95-100. Since my sort function uses O(n^2), it might not be the most efficient, however considering the limitations such as accessing lists only from the head, I think I did a great job.

References

https://web.cs.hacettepe.edu.tr/~bbm201/Lectures/BBM201-Lecture5.pdf

https://www.geeksforgeeks.org/sorting-algorithms/