الأكاديمية العربية للعلوم والتكنولوجيا والنقل البحري

Arab Academy for Science, Technology & Maritime Transport

University registration system project

Data structures and algorithms

Report to:
Prof. Osama Ismail
Eng. Nagy Khairt Ali

Report by:
Belal Mohamed Sameh
Email: belalsameh188@gmail.com

Abstract

Abstract As the technology develops the way everything done is even developed every day. The registration systems have begun since the 16th century and is still being developed until this day. At this report the way of developing an online registration system is being discussed. The report typically explains the data used and how it has been divided, the model used to develop the program, the functions that are used to make the program user friendly, validation techniques to make sure that the data is entered in the correct format, the equations while developing the program, the functions that can be performed using the program, the simulation of the program running with all the test cases, and the future work to be done in order to develop the program to work more efficiently.

Contents

1 Introduction

1.1 Historical background

Registration systems, the third part of a demographic data system, are generally designed to count vital events: births, deaths, marriages, and entries and exits at international boundaries [1]. In our daily lives we can have a registration system for everything. The vital registration system was started in England in the 16th century for the first time. It is the important sources of demographic data. It is also known as the civil registration system [2]. At the earlier centuries, registration systems were paper based to save all the required data until it was developed to the current registration systems using a large database of the information of the required people.

1.2 Statement of problem

The problem to be solved in this project is to design a university registration system which requires a database of the student's data this data is the used to

perform different functions on it. The tasks that a registration system can do from the student's data are uncountable. Mainly in this project I will try to perform tasks such as: allow the student to access the data and configure it only by entering the correct username and password, display all the data of a specific student, display the wanted data, configure the GPA of the student if any of the data of the his/her course data is changed, configure the student contact details that is already saved in the database, add any new payment that has been done, configure any course details, and sort the students according to their GPAs.

1.3 Report layout

First, Section 2 will contain the literature review which is the publications and previous work that has been done on this project previously. Secondly, Section 3 will contain theoretical work done which will present the data used, the way at which this data has been used, the implementation process and how it works, and the equations used in some cases to implement a specific task. Section 4 and 5 will contain a brief conclusion about the whole project and the future work to be implemented.

2 Literature review

Designing an online registration system has been developed since years which made a lot of researchers to write different research papers about the online registration system. One of the fond papers was written by Rattan Singh, Ravinder Singh, Harpreet Kaur and O. P. Gupta. This paper was published 20 June 2016. The paper mainly explains the development and designing of an online registration system and how to make it more effective [3].

3 Theoretical works

3.1 Data representation

A student registration system would contain a lot of data that belongs to a student. This data cannot be stored as one unit as this may cause a lot of corruptions and misuse. In this project the data will be divided into three different linked lists that are related to each other and connected with pointers. The first linked list is the student data linked list that is made of a struct that contains all the data that is related to the student personal information, contact details, university information, payment details. The second linked list is the finished courses linked list that is made of a struct that contains all the data that is related to the finished courses that has been studied by the student. The third linked list is the current courses linked list that is made of a struct that contains all the data that is related to the current courses that is being studied by the student. Each list will be discussed in more details throughout the report.

3.2 Modelling

There are a lot of models that can be used to design the program in the most efficient way. The way of modelling used in this project as there are a lot of data is to divide the program into separate functions. A single function is used to scan every set of data that was mentioned in the data representation section. After scanning all the required data all this data is then inserted in its node in the linked list one after the other. As we cannot use a database program the program at the beginning asks the user to insert the data of two different students to use them as the database of the program. The program then starts by asking the user if they are registered or not. If the student is registered the student is asked to enter their registration number and password to access the system, else if they are not registered the student is asked to fill in his/her information to register.

3.3 Implementation

The implementation process is divided into many steps as follow.

3.3.1 Structs used and data scanned

The first struct used is the StudentData struct that contains all the shown data in the image this data is scanned by scan student data function that contains a list of functions to scan each part of the struct.

```
struct StudentData
{
    char FirstName[30],MiddleName[30],LastName[30],InstructorName[50],InstructorCode[15],
        Nationality[15],StudentId[15],Password[10],Country[15],Governorate[15],City[15],
        District[15],StreetName[20],HouseNumber[5],FlatNumber[5],PostalCode[10],
        MailingCountry[15],MailingGovernorate[15],MailingCity[15],MailingDistrict[15],
        MailingStreetName[20],MailingHouseNumber[5],MailingFlatNumber[5],MailingPostalCode[10],
        StudentEmail[40],PhoneNumber[20],HomePhoneNumber[20],RegNum[20],Gender[10],IdType[15],
        EnrolmentTerm[9],EnrolmentYear[5],Department[50],TypeOfFunding[15];

    int DayOfBirth,MonthOfBirth,YearOfBirth,Term,MaxCreditHours,Age,NumberOfPayments,FinishedCourses,CurrentCourses;

    double TotalGPA,FinishedCreditHours,CurrentCreditHours,RemainingCreditHours,TotalPayment;

    double AmountPaid[20];

    int DayOfTransaction[20],MonthOfTransaction[20],YearOfTransaction[20];

    long long NumberOfTransaction[20];

    FinishedCoursesPtr  GetFinishedCourses;
    CurrentCoursesPtr GetCurrentCourses;
    StudentDataPtr next;
};
```

```
void ScanStuData(char *FirstName, char *MiddleName, char *LastName, char *InstructorName, char *InstructorCode, char *Nationality,
                 char *StudentId,char *Password,  char *Country, char *Governorate, char *City, char *District, char *StreetName,
                 char *HouseNumber, char *FlatNumber, char *PostalCode,char *MailingCountry, char *MailingGovernorate,char *MailingCity,
                 char *MailingDistrict, char *MailingStreetName, char *MailingHouseNumber,char *MailingFlatNumber,char *MailingPostalCode,
                 char *StudentEmail, char *PhoneNumber,char *HomePhoneNumber, char *RegNum,int *DayOfBirth,int *MonthOfBirth,int *YearOfBirth,
                 char *EnrolmentTerm,char *EnrolmentYear, int *MaxCreditHours,int *Department,int *Gender,int *IdType,int *Age,int *TypeOfFunding,
                 int *NumberOfPayments,double *TotalPayment,double *AmountPaid,long long *NumberOfTransaction,int *DayOfTransaction,
                 int *MonthOfTransaction,int *YearOfTransaction,int *FinishedCourses,int *CurrentCourses)
{
    GetPersonlInfo(FirstName,MiddleName,LastName,Gender,Nationality,IdType,StudentId,Password,DayOfBirth,MonthOfBirth,YearOfBirth);

    *Age=CalcAge(DayOfBirth,MonthOfBirth,YearOfBirth);

    GetContactDetails(Country,Governorate,City,District,StreetName,HouseNumber,FlatNumber,PostalCode,MailingCountry, MailingGovernorate,
                      MailingCity, MailingDistrict, MailingStreetName, MailingHouseNumber, MailingFlatNumber, MailingPostalCode,
                      StudentEmail,PhoneNumber,HomePhoneNumber);

    GetUniDetails(EnrolmentTerm,EnrolmentYear,RegNum,Department,InstructorName,InstructorCode,FinishedCourses,CurrentCourses);

    GetPaymentDetails(TypeOfFunding,NumberOfPayments,TotalPayment,AmountPaid,NumberOfTransaction,DayOfTransaction,MonthOfTransaction,YearOfTransaction);
}
```

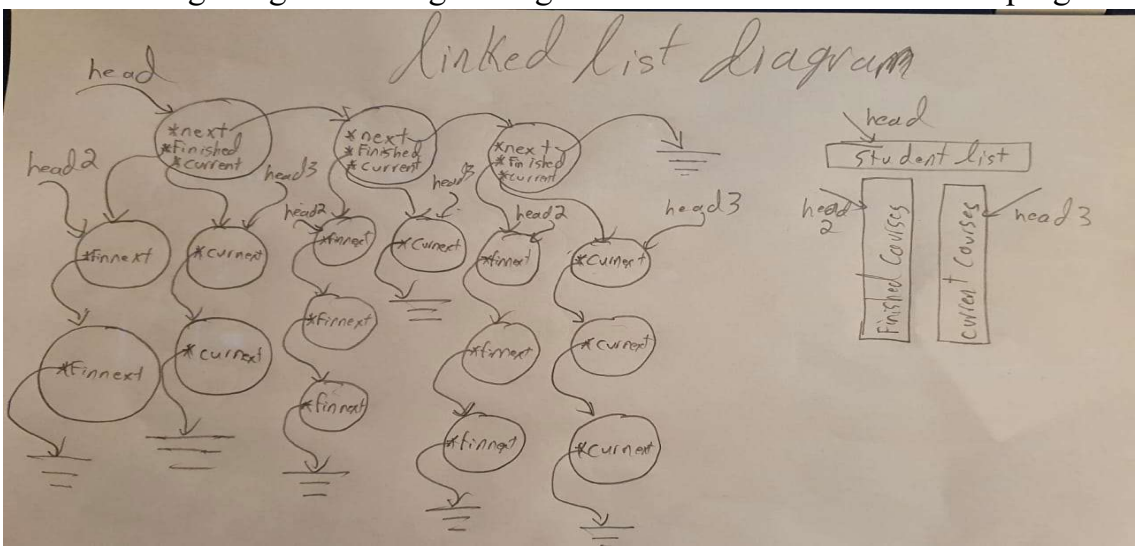
All those variables that are sent to the ScanStuData function is also declared in the main function of the program and passed by reference to the function so that the scanned data is stored in the variable and can be then sent to another function that is called AddStuData to insert the scanned data to the node of the current student. The linked list is implemented to insert each node at the end of the list. Each date input entered by the user is validated such that the day could not be less than 1 or greater than thirty-one, the month should not exceed twelve and should not be less than 1, and the year could not be less than 1980 or more than 2022 as shown in the figure beside.

```
cin>>*(DayOfBirth);
while(*(DayOfBirth)>31 || *(DayOfBirth)<0)
{
    cout<<"Incorrect day entered please enter the correct day of birth : ";
    cin>>*(DayOfBirth);
}
cout<<"Month of birth : ";
cin>>*(MonthOfBirth);
while(*(MonthOfBirth)>12 || *(MonthOfBirth)<0)
{
    cout<<"Incorrect month entered please enter the correct month of birth : "
    cin>>*(MonthOfBirth);
}
cout<<"Year of birth : ";
cin>>*(YearOfBirth);
while(*(YearOfBirth)>2022 || *(YearOfBirth)<1980)
{
    cout<<"Incorrect year entered please enter the correct year of birth : ";
    cin>>*(YearOfBirth);
}
```

```
struct FinishedCourses
{
    char CourseCode[8],CourseName[30],LecturerFirstName[30],LecturerMiddleName[30],LecturerLastName[30],
         GTAFirstName[30],GTAMiddleName[30],GTALastName[30];

    int NumberOfTimes, CourseCreditHours, CourseSemester, CourseYear;

    float Course7thMark,Course12thMark, CourseYearWork, CourseFinalMark, CoursePoints;

    double CourseTotalMark;

    FinishedCoursesPtr FinNextCourse;
};
```

The second struct used is the FinishedCourses struct that contains all the data that is related to the previously studied courses by the student. This data is scanned in the same way by calling the ScanFinishedCoursesDetails function and passing to it the variables and arrays by reference and then adding this data to the current node. The user is first asked to insert the number of finished courses which is set as a maximum of the loop that asks the user to enter the data of this number of courses. This data is inserted to the node by the AddFinishedCoursesDetails function. An extra pointer in the StudentData struct is available that has the type FinishedCoursesPtr this pointer is used so that the node of the StudentData will point on the list of the finished courses by the current student to from a multi list. All the previous steps are exactly repeated with the currently studied courses with the same algorithm. After inserting the current and finished courses of the current student the head pointer of both the FinishedCourses and CurrentCourses list is back to point to NULL to be ready to scan the data of the courses of the upcoming student if needed.

The following image is the logic image of the linked lists used in the program.



3.3.2 Equations

There is a list of equations that has been used to implement different functions in the program.

1- GPA calculation:
The calculation of the GPA is implemented by the formula
TotalGPA=TotalPoints/FinishedCreditHours
The total points are calculated by the summation of the CoursePoints that is determined according to the mark gained by the student in the courses that they have finished.

2- Term calculation:

The student current term is calculated by the formula
Term=(FinishedCreditHours/18) +1.

3- Registration number generation:

The RegNum is generated by the student EnrolmentYear then EnrolmentTerm then 250 which is the college code followed by the department and the student count.

```
cout<<"Your Registration number : ";
strcpy(RegNum,EnrolmentYear);
strcat(RegNum,EnrolmentTerm);
strcat(RegNum,"250");
itoa(*Department,TempDepartment,10);
strcat(RegNum,TempDepartment);
long long RegNumGen=atoi(RegNum);
char RegNumTemp[10],RegNumTemp2[10];
memset(RegNum,'\0',20);
RegNumGen*=10000;
RegNumGen+=StuCon;
StuCon++;
int RegNum1=RegNumGen/100000;
int RegNum2=RegNumGen%1000000;
itoa(RegNum1,RegNumTemp,10);
itoa(RegNum2,RegNumTemp2,10);
strcpy(RegNum,RegNumTemp);
strcat(RegNum,RegNumTemp2);
cout<<RegNum<<endl;
```

3.3.4 Logging in into the system

```
cout<<"Please choose the required function"<<endl;
cout<<"1- Display student data"<<endl;
cout<<"2- Display personal information"<<endl;
cout<<"3- Display contact details"<<endl;
cout<<"4- Display university information"<<endl;
cout<<"5- Display payment details"<<endl;
cout<<"6- Display finished courses"<<endl;
cout<<"7- Display current courses"<<endl;
cout<<"8- Change any course details"<<endl;
cout<<"9- Change any contact details"<<endl;
cout<<"10- Add new payment"<<endl;
cout<<"11- List of sorted student by GPA"<<endl;
cout<<"----------------"<<endl;
cout<<"Choice : ";
cin>>Choice;
cout<<"----------------"<<endl;
while(Choice>11 || Choice<0)
{
```

After getting all the required data of the student the student is then allowed to log in into the system. The log in process is requires the input of the student registration number and password that has been already set. Those data are then compared to all the data in all the nodes of the StudentData linked list of both are not identical to one of the nodes an error message is shown that an incorrect data has been entered and the student is asked to re-enter the data. The student is not allowed to log in into the system until both the registration number and password are found successful. This function then returns a pointer with the same type as the struct, that points to the node that is needed. The user is then shown all the list of functions that can be performed using the system. The input choice is validated so that the user could not choose an input that is not available. Each function takes the pointer that points to the node that its password has been entered as its parameter in order to perform the required task just on this node without changing anything. After performing each task, the student is asked whether they need to perform another function for the same student or not. If yes, the same list of function is shown again to the student, else if not the student is logged out of the system and the user is asked whether they need to log in into the system again or not. If yes, the student is asked to enter the registration number and password once more else the program is completely closed.

3.3.5 Functions used

All the display functions are simple as they just print the required information using the password pointer that points to the required node.

3.3.5.1 Change any course detail's function

This function allows the student to configure any of the courses that has been already finished. The student is asked to enter the course code required that is then compared to all the courses in the FinishedCourses linked list if the course is found, the student is allowed to change the information available in node of this course then after configuring the details the GetGpa function is called to configure the GPA of the student according to the change that has been done to the course details, else if the course code is not found in the list the student is shown a message that the course has not been found and is asked if the need to perform another function using the system.

3.3.5.2 Change any contact detail's function

The function uses the same algorithm as the previous function by asking the user on each contact details whether they want to change it or not. If yes, the new contact detail is scanned and then stored in place of the previous contact detail in the StudentData struct, else if no nothing is changed in that contact detail and the student is asked about another contact detail the same question.

3.3.5.3 Add new payment

The payments done by the student is a running process that is repeatedly done that is why all the payment details of each student are stored in parallel arrays so that each index has the information of one transaction. If a new payment has been done by the student, the student is asked to enter the new transaction details. The information of this transaction is simply added as the next element of each array that are related to the payment details in the StudentData struct and the total payment is changed by adding the new AmountPaid to the previous total and the array index is incremented by one so that if a new payment is going to be added it will be added at the next element.

3.3.5.4 Sorting the nodes according to the GPA

The insertion sort algorithm is used to sort the nodes of this linked list. The algorithm is basically built on declaring a new pointer that points to NULL at the beginning. The algorithm is based on pointing to the unsorted the node of the list and then point back using the original pointer in its sorted position. The same process is repeatedly done to all the nodes to sort them that is why it requires a $O(n^2)$ time complexity. After sorting the nodes, the list of student names and their GPAs is shown in ascending order.

4 Conclusion

To sum up, the program mainly asks a user who is the register of the student that acts as the database of the students then the program starts running by showing the student a log in screen that requires him/her to enter their registration number and password that will allow them to access the system. The program then shows a list of functions that the student can perform using the registration system. All the data entered by the student in the main program is validated to make sure that the program processes correctly without any invalid inputs. The student is allowed to perform infinite number of functions until he/she decides to log out from the system. The program is always ready to perform all the determined functions for all the students registered in the system until the program is fully closed.

5 Future work

A lot of future work can be done to improve the registration system. The program needs also to validate all the possible inputs such as “Id number” and “passport number”. I also intend to implement the program using graphical user interface (GUI) to allow the user to make the program more user-friendly. I will also implement the program to be usable by different colleges and universities. Lastly, I would also implement the program to run on a local host by many different users at the same time.

6 List of References

1. https://www.sciencedirect.com/topics/computer-science/registration-system#:~:text=Registration%20systems%2C%20the%20third%20part,and%20exits%20at%20international%20boundaries.
2. https://kullabs.com/class-9/enviroment-population-and-health-9/demography,-population-change-and-its-management/vital-registration-system-vrs
3. https://www.computerscijournal.org/vol9no2/development-of-online-student-course-registration-system/