A student management system is a software application that helps educational institutions to manage student data and handle various student-related tasks. It can be used to store and organize information such as student personal and contact details, enrollment information, grades, and attendance records.

In this post I will explain with examples how can you implement A simple student management system project using the C programming language. C is a general-purpose, procedural programming language that is widely used for developing systems software and applications. It is a popular choice for student management systems because it is efficient, simple to learn, and has a rich set of libraries and functions that can be used to implement various features of the system.

Some Common Features Of A Student Management System

1. **Student registration:** This feature allows students to register for courses and enter their personal and contact information into the system.
2. **Course management:** This feature allows the educational institution to manage courses and class schedules, including adding or deleting courses and assigning teachers to teach them.

Student management systems are designed to streamline and automate various processes related to managing students, such as student registration, course management, attendance tracking, and grades and transcripts. They can be used by schools, colleges, and universities to manage student data and improve the efficiency of various student-related tasks.

Steps Of Creating A Simple Student Management System?

It is easy to create a simple student management system using C programming. C is a general-purpose, procedural programming language that is widely used for developing systems software and applications, and it has a rich set of libraries and functions that can be used to implement various features of a student management system.

Plan And Design Your System

Before you start coding, it is important to plan and design your student management system. Determine the features that you want to include, such as student registration, course management, attendance tracking, and grades and transcripts. You should also decide on the overall architecture and structure of your system, including the data structures that you will use to store and organize student information.

Set Up Your Development Environment

To create a student management system in C, you will need a C compiler and a text editor. There are several C compilers available, such as GCC (GNU

Compiler Collection) and Microsoft Visual C++, and you can use any text editor that you are comfortable with, such as Notepad++ or Sublime Text.

Write And Test Your Code

Once you have set up your development environment, you can start writing and testing your code. You can start by implementing the basic functionality of your student management system, such as student registration, course management, and attendance tracking. As you develop your system, be sure to test your code thoroughly to ensure that it is working correctly and to identify and fix any bugs or issues.

Debug And Optimize Your Code

As you develop your student management system, you may encounter bugs or issues that need to be addressed. To fix these issues, you will need to debug your code and identify the root cause of the problem. You may also want to optimize your code to improve its performance and efficiency.

Deploy Your System

Once you have completed the development and testing of your student management system, you will need to deploy it in a production environment. This may involve installing your system on a server or hosting it on a cloud platform, and configuring any necessary security or access controls.

Overall, creating a simple student management system using C programming requires planning and design, setting up a development environment, writing and testing code, debugging and optimizing code, and deploying the system. With careful planning and attention to detail, you should be able to create a functional and reliable student management system using C programming.

A Simple Student Management System.

You may already know that in c programming language there is no UI(user interface) and it is not enough high-level language. So you can’t actually develop a real-life student management system using the C programming language. So now we will create a simple version of the Student Management System using the C-Programming Language.

So to keep our project simple, we will store the following data of a student.

- **Student Information**
    1. Student ID
    2. Name
    3. Email
    4. Phone
    5. Number of Courses
- **Course Information**
    1. Course Code
    2. Course Name

C-Programming Technology We Will Use

To develop this simple student management system project we will use the common functionality of the c-programming language like:

1. For loop
2. While loop
3. If-else
4. Structure
5. Function
6. Array

Some Built-In Functions We Will Use

1. **system("cls")**: The *system("cls")* function is used to clear the window.
2. **strlen():** The *strlen()* function is used to calculate the string length.
3. **strcpy():** The *strcpy()* function is used to copy one string to another.
4. **strcmp():** The strcmp() function is used to compare two strings, is they are the same or not?

Feature Of Student Management System Project

You already know we will create a simple student management system so that we will focus only on the main logic(CURD) and entity(student). We will store basic information about a student and then edit, update, delete and search for the student's knowledge.

1. Add new student
2. Show All Student's information
3. Edit Student information
4. Search Student information
5. Delete Student information
6. Delete All Students

```
                MAIN MENU
            ======================
            [1] Add A New student.
            [2] Show All students.
            [3] Search A student.
            [4] Edit A student.
            [5] Delete A student.
            [6] Delete All students.
            [7] Clear The window.
            [8] User Guideline.
            [9] About Us.
            [0] Exit the Program.
```

Add New Student

Adding a new student is one of the most straightforward features of this project. We just need to take input the student information from the user and store it. Note that we will store all of our information in the structure array of c-language.
Also, before storing the student's information we will validate the user inputs. Here are the user input validation rules we will use in this project.

1. A student can have a maximum of 4 courses and a minimum of 1 course.
2. Student ID can be a maximum of 10 characters long and unique for every student.
3. Student Name can be a maximum of 20 characters long
4. Student Email can be a maximum of 30 characters long and unique for every student.
5. Student Phones can be maximum of 20 characters long and unique for every student.
6. Course code can be a maximum of 10 characters long.
7. Course Name can be a maximum of 20 characters long.

```
struct StudentInfo
{
   char ID[10];
   char Name[20];
   char Email[30];
   char Phone[20];
   int NumberOfCourse;
};
struct CourseInfo
{
   char StudentID[10];
   char Code[10];
   char Name[20];
};
```

```
struct StudentInfo Students[100];
struct CourseInfo Courses[500];
// store dummy data in structure array
strcpy(Students[0].ID,"S-1");
strcpy(Students[0].Name,"Student 1");
strcpy(Students[0].Phone,"016111111111");
strcpy(Students[0].Email,"student-1@gmail.com");
Students[0].NumberOfCourse=1;
strcpy(Courses[0].StudentID,"S-1");
strcpy(Courses[0].Code,"CSE-1");
strcpy(Courses[0].Name,"Course - 1");
```

Show All Students

Showing all students is also one of the easiest features of this project. We just need to loop through the structure array and print the student information. In this project, we will show all information in a table.
Note: we will format the table based on the student's information length.

```
void ShowAllStudents()
{

printf("|==========|====================|==============================|==========
==========|=============|\n");
    printf("|    ID    |        Name        |            Email             |
Phone        |  NO.Course  |\n");

printf("|==========|====================|==============================|==========
==========|=============|\n");
    for(i=0; i < TotalStudents; i++)
    {
        printf("|");
        printf("%s",Students[i].ID);
        for(j=0; j < (10-strlen(Students[i].ID)); j++)
        {
            printf(" ");
        }
        printf("|");
        printf("%s",Students[i].Name);
        for(j=0; j < (20-strlen(Students[i].Name)); j++)
        {
            printf(" ");
        }
        printf("|");
        printf("%s",Students[i].Email);
        for(j=0; j < (30-strlen(Students[i].Email)); j++)
        {
            printf(" ");
        }
        printf("|");
        printf("%s",Students[i].Phone);
        for(j=0; j < (20-strlen(Students[i].Phone)); j++)
        {
            printf(" ");
        }
        printf("|");
```

```
        printf("%d",Students[i].NumberOfCourse);
        for(j=0; j < 12; j++)
        {
            printf(" ");
        }
        printf("|\n");
        printf("|----------|--------------------|------------------------------|--
------------------|-------------|\n");
    }
    printf("\n");
}
```

Edit Student Informations

Editing students' information is the same to add new students, but in our project, we will allow users to skip editing specific information. Suppose we want to edit a student's email only not the name and phone number in this case we need to skip editing the name and phone number.
So we will give an option to the user if the user does not want to edit specific information he can input zero(0).

```
void EditStudent(int StudentFoundIndex)
{
    int IsValidName = 0;
    while(!IsValidName)
    {
        printf(" Enter The New Name(0 for skip): ");
        scanf(" %[^\n]s",&NewName);
        if(strlen(NewName) > 20)
        {
            printf(" Error: Name can not be more than 20 characters.\n\n");
            IsValidName = 0;
        }
        else if(strlen(NewName) <= 0)
        {
            printf(" Error: Name can not be empty.\n\n");
            IsValidName = 0;
        }
        else
        {
            IsValidName = 1;
        }
    }
    int IsValidEmail = 0;
    while(!IsValidEmail)
    {
        printf(" Enter The New Email(0 for skip): ");
        scanf("%s",&NewEmail);
        if(strlen(NewEmail) > 30)
        {
            printf(" Error: Email can not be more than 30 characters.\n\n");
            IsValidEmail = 0;
        }
        else if(strlen(NewEmail) <= 0)
        {
            printf(" Error: Email can not be empty.\n\n");
            IsValidEmail = 0;
```

```
        }
        else if(IsAlreadyExists(NewEmail,'e',StudentID) > 0)
        {
            printf(" Error: This Email Already Exists.\n\n");
            IsValidEmail = 0;
        }
        else
        {
            IsValidEmail = 1;
        }
    }
    int IsValidPhone = 0;
    while(!IsValidPhone)
    {
        printf(" Enter The New Phone(0 for skip): ");
        scanf("%s",&NewPhone);
        if(strlen(NewPhone) > 20)
        {
            printf(" Error: Phone can not be more than 20 characters.\n\n");
            IsValidPhone = 0;
        }
        else if(strlen(NewPhone) <= 0)
        {
            printf(" Error: Phone can not be empty.\n\n");
            IsValidPhone = 0;
        }
        else if(IsAlreadyExists(NewPhone,'p',StudentID) > 0)
        {
            printf(" Error: This Phone Number is Already Exists.\n\n");
            IsValidPhone = 0;
        }
        else
        {
            IsValidPhone = 1;
        }
    }
    int IsValidNumberOfCourse = 0;
    while(!IsValidNumberOfCourse)
    {
        printf(" Number of New courses(0 for skip): ");
        scanf("%d",&NewNumberOfCourses);
        if(NewNumberOfCourses > 4 || NewNumberOfCourses < 0)
        {
            printf(" Error: A Student can have maximum 4 and Minimum 0 number of
courses.\n\n");
            IsValidNumberOfCourse = 0;
        }
        else
        {
            IsValidNumberOfCourse = 1;
        }
    }
    if(strcmp(NewName,"0") != 0)
    {
        strcpy(Students[StudentFoundIndex].Name,NewName);
    }
    if(strcmp(NewEmail,"0") != 0)
    {
        strcpy(Students[StudentFoundIndex].Email,NewEmail);
```

```
    }
    if(strcmp(NewPhone,"0") != 0)
    {
        strcpy(Students[StudentFoundIndex].Phone,NewPhone);
    }
    if(NewNumberOfCourses != 0)
    {
        int OldTotalCourse = Students[StudentFoundIndex].NumberOfCourse;
        Students[StudentFoundIndex].NumberOfCourse = NewNumberOfCourses;
        int FirstCourseIndex;
        int dc;
        for(dc=0; dc < TotalCourse; dc++)
        {
            if(strcmp(StudentID,Courses[dc].StudentID) == 0)
            {
                FirstCourseIndex = dc; // store the index for delete
                break;
            }
        }

        for(dc=1; dc < = OldTotalCourse; dc++)
        {
            DeleteCourseByIndex(FirstCourseIndex);
        }
        char CourseCode[300];
        char CourseName[300];
        for(i=1; i < = NewNumberOfCourses; i++)
        {
            printf(" Enter New Course %d Code: ",i);
            scanf("%s",&CourseCode);
            printf(" Enter New Course %d Name: ",i);
            scanf(" %[^\n]s",&CourseName);
            strcpy(Courses[TotalCourse].StudentID,StudentID);
            strcpy(Courses[TotalCourse].Code,CourseCode);
            strcpy(Courses[TotalCourse].Name,CourseName);
            TotalCourse++;
        }
    }
    printf(" Student Updated Successfully.\n\n"
```

Search A Student

Searching for a student will be one of the easiest tasks for you if you know how to search for an element in an array. We will search students on the student and courses array by **student ID**.

```
int SearchStudent(char StudentID[10])
{
    system("cls");
    int StudentFoundIndex = -1;
    int i;
    for(i=0; i < TotalStudents; i++)
    {
        if(strcmp(StudentID,Students[i].ID) == 0)
        {
            StudentFoundIndex = i;
            printf("\n One Student Found for ID: %s\n\n",StudentID);
```

```
            printf(" Student Informations\n");
            printf("-------------------------\n");
            printf(" ID:    %s\n",Students[i].ID);
            printf(" Name:  %s\n",Students[i].Name);
            printf(" Email: %s\n",Students[i].Email);
            printf(" Phone: %s\n",Students[i].Phone);
            printf("\n Total Number of Courses: %d\n",Students[i].NumberOfCourse);
        }
    }
    int CourseCount = 0;
    int j;
    for(j=0; j < TotalCourse; j++)
    {
        if(strcmp(StudentID,Courses[j].StudentID) == 0)
        {
            CourseCount++;
            printf(" Course %d Code: %s\n",CourseCount,Courses[j].Code);
            printf(" Course %d Name: %s\n",CourseCount,Courses[j].Name);
        }
    }
    return StudentFoundIndex;
}
```

Deleting A Student

Deleting a student is the same as removing an item from an array. First, we will search students by **Student ID**, if we found the student we will remove the student from the student and course array.

```
void DeleteStudent(int StudentIndex)
{
    int d;
    int FirstCourseIndexs;
    struct StudentInfo ThisStudents;
    ThisStudents = Students[StudentIndex];
    for(d=0; d < TotalCourse; d++)
    {
        if(strcmp(ThisStudents.ID,Courses[d].StudentID) == 0)
        {
            FirstCourseIndexs = d;
            break;
        }
    }
    for(d=1; d < = ThisStudents.NumberOfCourse; d++)
    {
        DeleteCourseByIndex(FirstCourseIndexs);
    }
    DeleteStudentByIndex(StudentIndex);
    printf(" Student Deleted Successfully.\n\n");
    GoBackOrExit();
}
```

Deleting All Student

Deleting all students is the easiest task we need to remove all items from the student and course array.

```
void DeleteAllStudents()
{
    TotalStudents = 0;
    TotalCourse = 0;
    printf(" All Students Deleted Successfully.\n\n");
    GoBackOrExit();
```

Is C Programming A Good Choice For Implementing A Simple Student Management System Project?

C programming can be a good choice for implementing a simple student management system project. C is a general-purpose, procedural programming language that is widely used for developing systems software and applications. It is known for its efficiency, simplicity, and flexibility, which can make it well-suited for implementing a student management system.

One of the main advantages of using C for a student management system project is its efficiency. C is a compiled language, which means that it is converted into machine code that can be directly executed by the computer's processor. This can make C programs faster and more efficient than programs written in other languages, which can be especially useful for applications that need to handle large amounts of data or perform complex tasks.

Another advantage of C is its simplicity. C has a relatively small and simple syntax, which can make it easier to learn and use than some other programming languages. This can be especially useful for students who are new to programming or who are working on a simple student management system project.

C also has a rich set of libraries and functions that can be used to implement various features of a student management system, such as student registration, course management, attendance tracking, and grades and transcripts. This can make it easier to develop a student management system in C, as you can use these pre-existing libraries and functions to save time and effort.

Overall, C can be a good choice for implementing a simple student management system project due to its efficiency, simplicity, and rich set of libraries and functions. However, it is worth considering other programming languages as well, as different languages may be better suited for different types of projects and different programming environments.

Conclusion

Creating a student management system using a c programming language is easy, all you need to know is how you will store data in an array. In this project, we stored student information using a struct array.