Learn In Depth

Be Professional In Embedded System

Report on:

First Term Project 2: Student management system using queue

Prepared by:

Abanoub Salah

Submitted To:

Engr. Keroles Shenouda

Submitted Date:

9/19/2022

Contents

• System Description

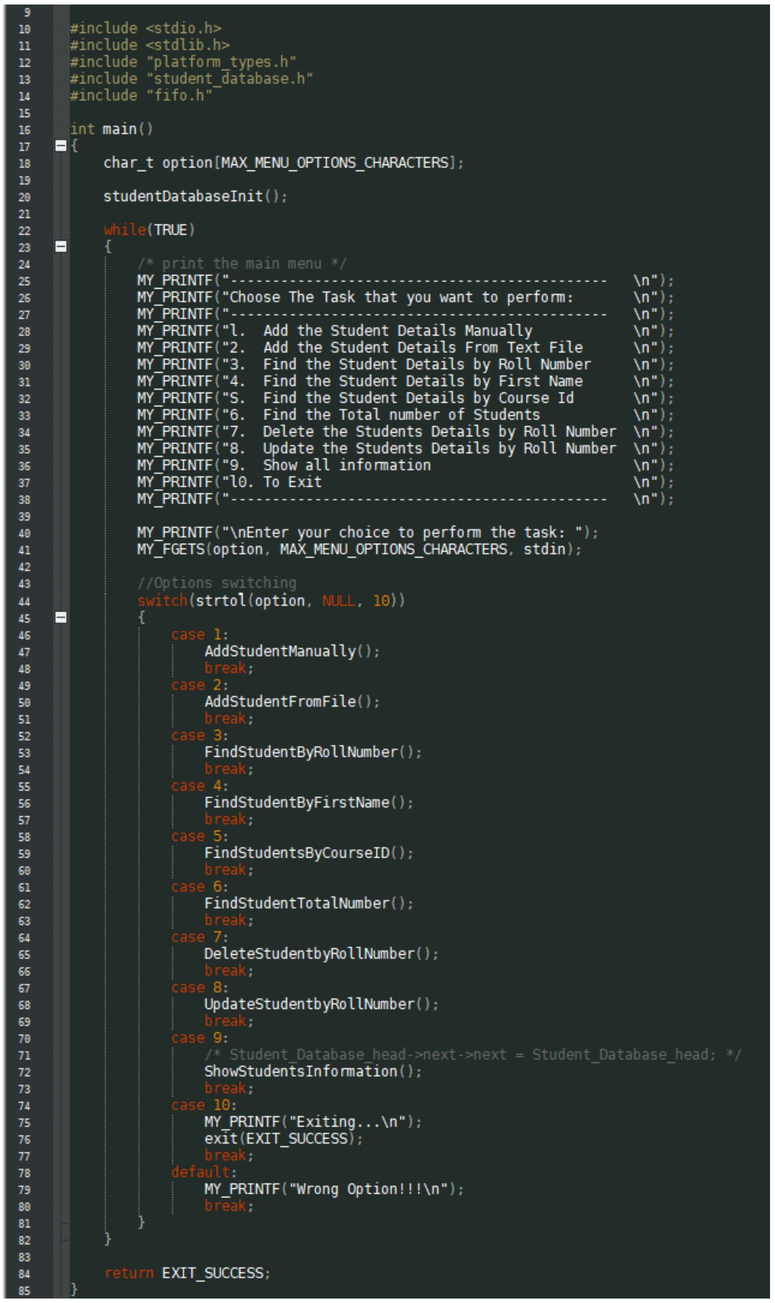
Write a program to build a simple Software for Student Information Management System which can perform the following operations:

1. Store the First name of the student.
2. Store the Last name of the student.
3. Store the unique Roll number for every student.
4. Store the GPA of every student.
5. Store the courses registered by the student.

Assumptions:

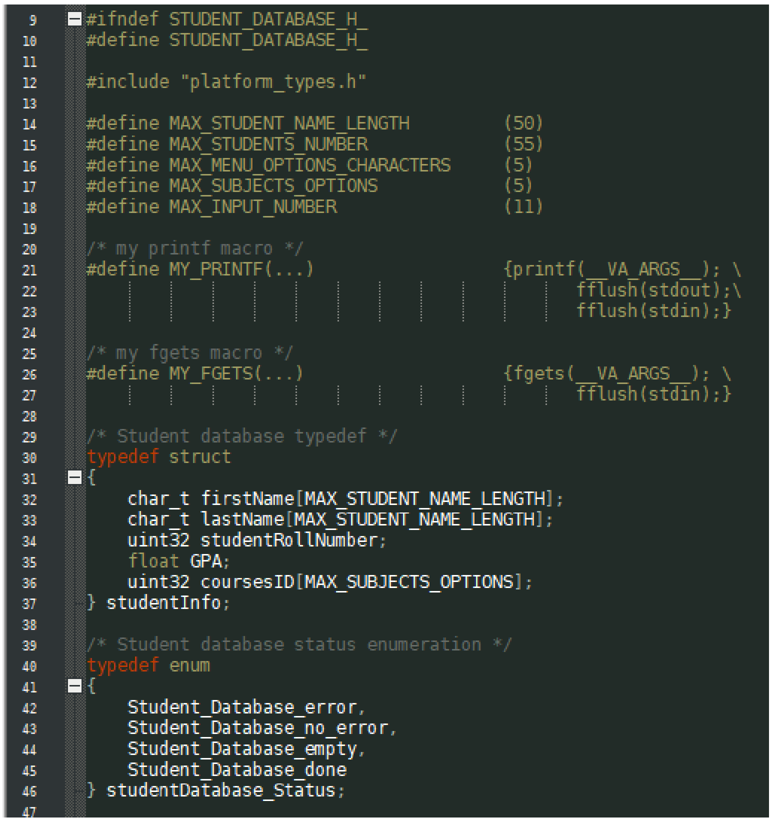
1. Input is always valid and within specified length.
2. File named “studentsFile.txt” exist within the same folder as the executable that contains a valid student data.

- **C Code**
  - *Main*



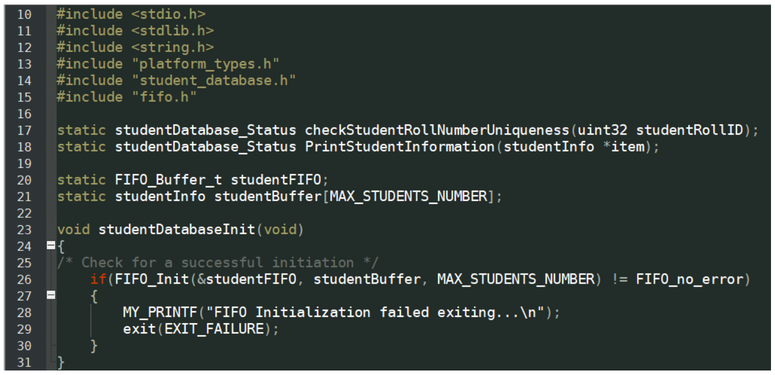
```c
#include <stdio.h>
#include <stdlib.h>
#include "platform_types.h"
#include "student_database.h"
#include "fifo.h"

int main()
{
    char_t option[MAX_MENU_OPTIONS_CHARACTERS];

    studentDatabaseInit();

    while(TRUE)
    {
        /* print the main menu */
        MY_PRINTF("---------------------------------------------   \n");
        MY_PRINTF("Choose The Task that you want to perform:       \n");
        MY_PRINTF("---------------------------------------------   \n");
        MY_PRINTF("1.  Add the Student Details Manually            \n");
        MY_PRINTF("2.  Add the Student Details From Text File      \n");
        MY_PRINTF("3.  Find the Student Details by Roll Number     \n");
        MY_PRINTF("4.  Find the Student Details by First Name      \n");
        MY_PRINTF("5.  Find the Student Details by Course Id       \n");
        MY_PRINTF("6.  Find the Total number of Students           \n");
        MY_PRINTF("7.  Delete the Students Details by Roll Number  \n");
        MY_PRINTF("8.  Update the Students Details by Roll Number  \n");
        MY_PRINTF("9.  Show all information                        \n");
        MY_PRINTF("10. To Exit                                     \n");
        MY_PRINTF("---------------------------------------------   \n");

        MY_PRINTF("\nEnter your choice to perform the task: ");
        MY_FGETS(option, MAX_MENU_OPTIONS_CHARACTERS, stdin);

        //Options switching
        switch(strtol(option, NULL, 10))
        {
            case 1:
                AddStudentManually();
                break;
            case 2:
                AddStudentFromFile();
                break;
            case 3:
                FindStudentByRollNumber();
                break;
            case 4:
                FindStudentByFirstName();
                break;
            case 5:
                FindStudentsByCourseID();
                break;
            case 6:
                FindStudentTotalNumber();
                break;
            case 7:
                DeleteStudentbyRollNumber();
                break;
            case 8:
                UpdateStudentbyRollNumber();
                break;
            case 9:
                /* Student_Database_head->next->next = Student_Database_head; */
                ShowStudentsInformation();
                break;
            case 10:
                MY_PRINTF("Exiting...\n");
                exit(EXIT_SUCCESS);
                break;
            default:
                MY_PRINTF("Wrong Option!!!\n");
                break;
        }
    }

    return EXIT_SUCCESS;
}
```

- *student_database.h (*partial)*



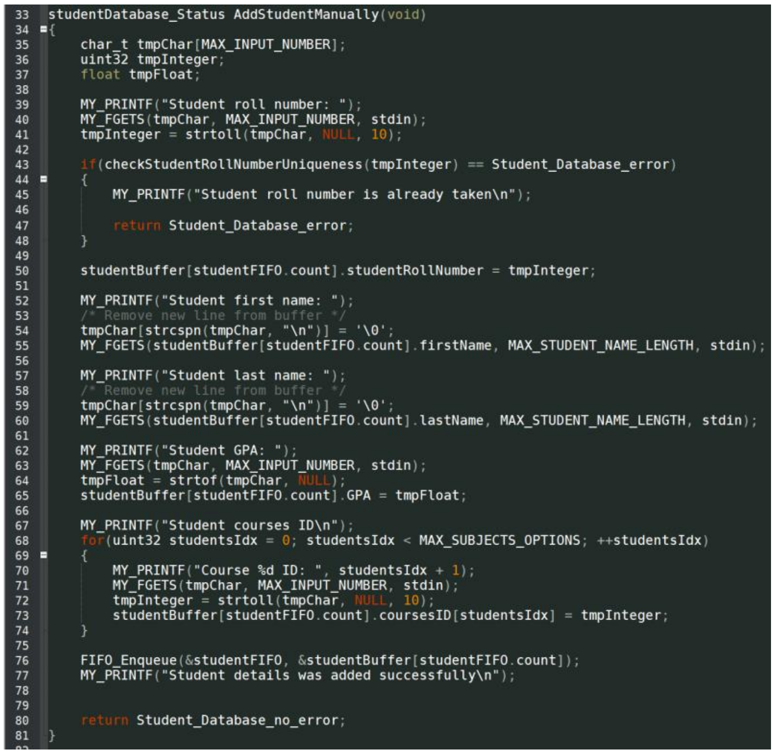
```c
#ifndef STUDENT_DATABASE_H_
#define STUDENT_DATABASE_H_

#include "platform_types.h"

#define MAX_STUDENT_NAME_LENGTH         (50)
#define MAX_STUDENTS_NUMBER             (55)
#define MAX_MENU_OPTIONS_CHARACTERS     (5)
#define MAX_SUBJECTS_OPTIONS            (5)
#define MAX_INPUT_NUMBER                (11)

/* my printf macro */
#define MY_PRINTF(...)                  {printf(__VA_ARGS__); \
                                            fflush(stdout);\
                                            fflush(stdin);}

/* my fgets macro */
#define MY_FGETS(...)                   {fgets(__VA_ARGS__); \
                                            fflush(stdin);}

/* Student database typedef */
typedef struct
{
    char_t firstName[MAX_STUDENT_NAME_LENGTH];
    char_t lastName[MAX_STUDENT_NAME_LENGTH];
    uint32 studentRollNumber;
    float GPA;
    uint32 coursesID[MAX_SUBJECTS_OPTIONS];
} studentInfo;

/* Student database status enumeration */
typedef enum
{
    Student_Database_error,
    Student_Database_no_error,
    Student_Database_empty,
    Student_Database_done
} studentDatabase_Status;
```

- *studentDatabaseInit()*



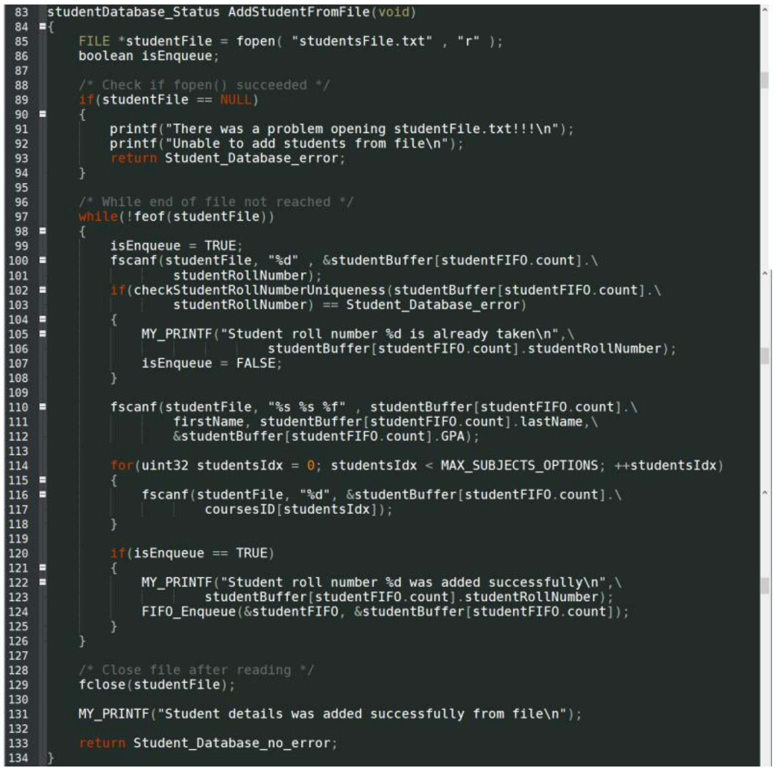
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "platform_types.h"
#include "student_database.h"
#include "fifo.h"

static studentDatabase_Status checkStudentRollNumberUniqueness(uint32 studentRollID);
static studentDatabase_Status PrintStudentInformation(studentInfo *item);

static FIFO_Buffer_t studentFIFO;
static studentInfo studentBuffer[MAX_STUDENTS_NUMBER];

void studentDatabaseInit(void)
{
/* Check for a successful initiation */
    if(FIFO_Init(&studentFIFO, studentBuffer, MAX_STUDENTS_NUMBER) != FIFO_no_error)
    {
        MY_PRINTF("FIFO Initialization failed exiting...\n");
        exit(EXIT_FAILURE);
    }
}
```

- *AddStudentManually()*



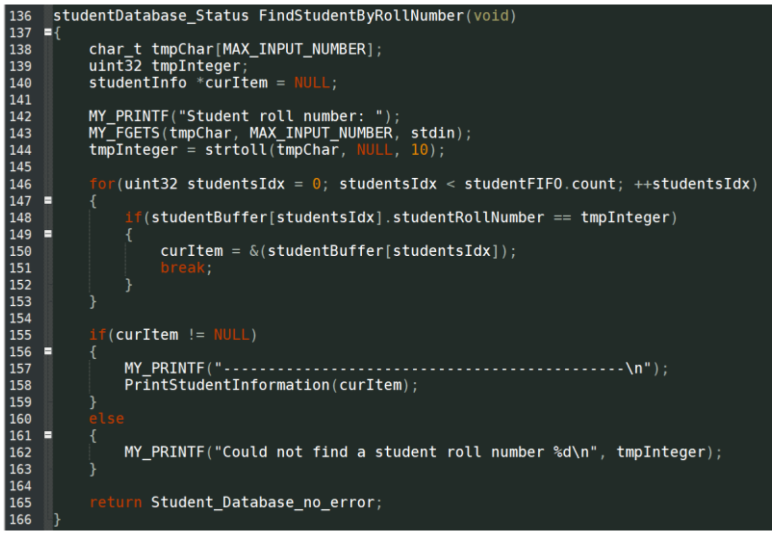
```c
studentDatabase_Status AddStudentManually(void)
{
    char_t tmpChar[MAX_INPUT_NUMBER];
    uint32 tmpInteger;
    float tmpFloat;

    MY_PRINTF("Student roll number: ");
    MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
    tmpInteger = strtoll(tmpChar, NULL, 10);

    if(checkStudentRollNumberUniqueness(tmpInteger) == Student_Database_error)
    {
        MY_PRINTF("Student roll number is already taken\n");

        return Student_Database_error;
    }

    studentBuffer[studentFIFO.count].studentRollNumber = tmpInteger;

    MY_PRINTF("Student first name: ");
    /* Remove new line from buffer */
    tmpChar[strcspn(tmpChar, "\n")] = '\0';
    MY_FGETS(studentBuffer[studentFIFO.count].firstName, MAX_STUDENT_NAME_LENGTH, stdin);

    MY_PRINTF("Student last name: ");
    /* Remove new line from buffer */
    tmpChar[strcspn(tmpChar, "\n")] = '\0';
    MY_FGETS(studentBuffer[studentFIFO.count].lastName, MAX_STUDENT_NAME_LENGTH, stdin);

    MY_PRINTF("Student GPA: ");
    MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
    tmpFloat = strtof(tmpChar, NULL);
    studentBuffer[studentFIFO.count].GPA = tmpFloat;

    MY_PRINTF("Student courses ID\n");
    for(uint32 studentsIdx = 0; studentsIdx < MAX_SUBJECTS_OPTIONS; ++studentsIdx)
    {
        MY_PRINTF("Course %d ID: ", studentsIdx + 1);
        MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
        tmpInteger = strtoll(tmpChar, NULL, 10);
        studentBuffer[studentFIFO.count].coursesID[studentsIdx] = tmpInteger;
    }

    FIFO_Enqueue(&studentFIFO, &studentBuffer[studentFIFO.count]);
    MY_PRINTF("Student details was added successfully\n");


    return Student_Database_no_error;
}
```

- *AddStudentFromFile()*



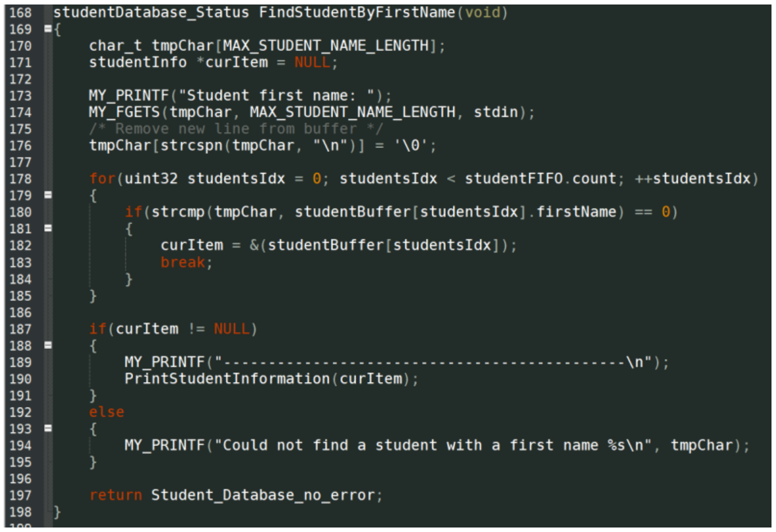
```c
studentDatabase_Status AddStudentFromFile(void)
{
    FILE *studentFile = fopen( "studentsFile.txt" , "r" );
    boolean isEnqueue;

    /* Check if fopen() succeeded */
    if(studentFile == NULL)
    {
        printf("There was a problem opening studentFile.txt!!!\n");
        printf("Unable to add students from file\n");
        return Student_Database_error;
    }

    /* While end of file not reached */
    while(!feof(studentFile))
    {
        isEnqueue = TRUE;
        fscanf(studentFile, "%d" , &studentBuffer[studentFIFO.count].\
                studentRollNumber);
        if(checkStudentRollNumberUniqueness(studentBuffer[studentFIFO.count].\
                studentRollNumber) == Student_Database_error)
        {
            MY_PRINTF("Student roll number %d is already taken\n",\
                            studentBuffer[studentFIFO.count].studentRollNumber);
            isEnqueue = FALSE;
        }

        fscanf(studentFile, "%s %s %f" , studentBuffer[studentFIFO.count].\
                firstName, studentBuffer[studentFIFO.count].lastName,\
                &studentBuffer[studentFIFO.count].GPA);

        for(uint32 studentsIdx = 0; studentsIdx < MAX_SUBJECTS_OPTIONS; ++studentsIdx)
        {
            fscanf(studentFile, "%d", &studentBuffer[studentFIFO.count].\
                    coursesID[studentsIdx]);
        }

        if(isEnqueue == TRUE)
        {
            MY_PRINTF("Student roll number %d was added successfully\n",\
                    studentBuffer[studentFIFO.count].studentRollNumber);
            FIFO_Enqueue(&studentFIFO, &studentBuffer[studentFIFO.count]);
        }
    }

    /* Close file after reading */
    fclose(studentFile);

    MY_PRINTF("Student details was added successfully from file\n");

    return Student_Database_no_error;
}
```

- *FindStudentByRollNumber()*



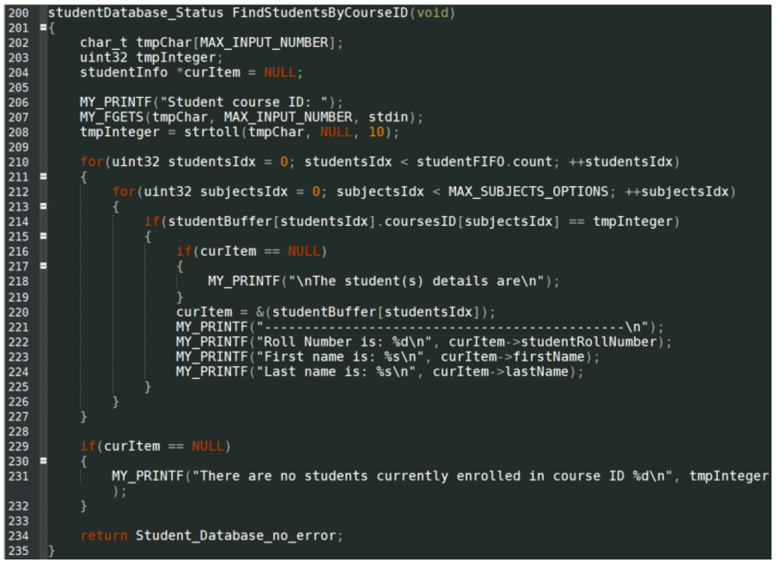
```c
studentDatabase_Status FindStudentByRollNumber(void)
{
    char_t tmpChar[MAX_INPUT_NUMBER];
    uint32 tmpInteger;
    studentInfo *curItem = NULL;

    MY_PRINTF("Student roll number: ");
    MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
    tmpInteger = strtoll(tmpChar, NULL, 10);

    for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        if(studentBuffer[studentsIdx].studentRollNumber == tmpInteger)
        {
            curItem = &(studentBuffer[studentsIdx]);
            break;
        }
    }

    if(curItem != NULL)
    {
        MY_PRINTF("----------------------------------------------\n");
        PrintStudentInformation(curItem);
    }
    else
    {
        MY_PRINTF("Could not find a student roll number %d\n", tmpInteger);
    }

    return Student_Database_no_error;
}
```

- *FindStudentByFirstName()*



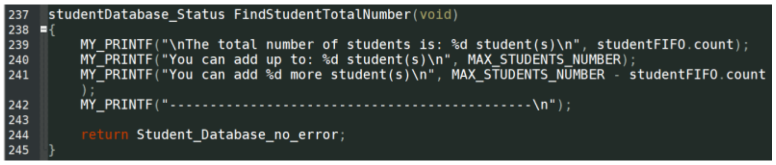
```c
studentDatabase_Status FindStudentByFirstName(void)
{
    char_t tmpChar[MAX_STUDENT_NAME_LENGTH];
    studentInfo *curItem = NULL;

    MY_PRINTF("Student first name: ");
    MY_FGETS(tmpChar, MAX_STUDENT_NAME_LENGTH, stdin);
    /* Remove new line from buffer */
    tmpChar[strcspn(tmpChar, "\n")] = '\0';

    for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        if(strcmp(tmpChar, studentBuffer[studentsIdx].firstName) == 0)
        {
            curItem = &(studentBuffer[studentsIdx]);
            break;
        }
    }

    if(curItem != NULL)
    {
        MY_PRINTF("---------------------------------------------\n");
        PrintStudentInformation(curItem);
    }
    else
    {
        MY_PRINTF("Could not find a student with a first name %s\n", tmpChar);
    }

    return Student_Database_no_error;
}
```

- *FindStudentsByCourseID()*



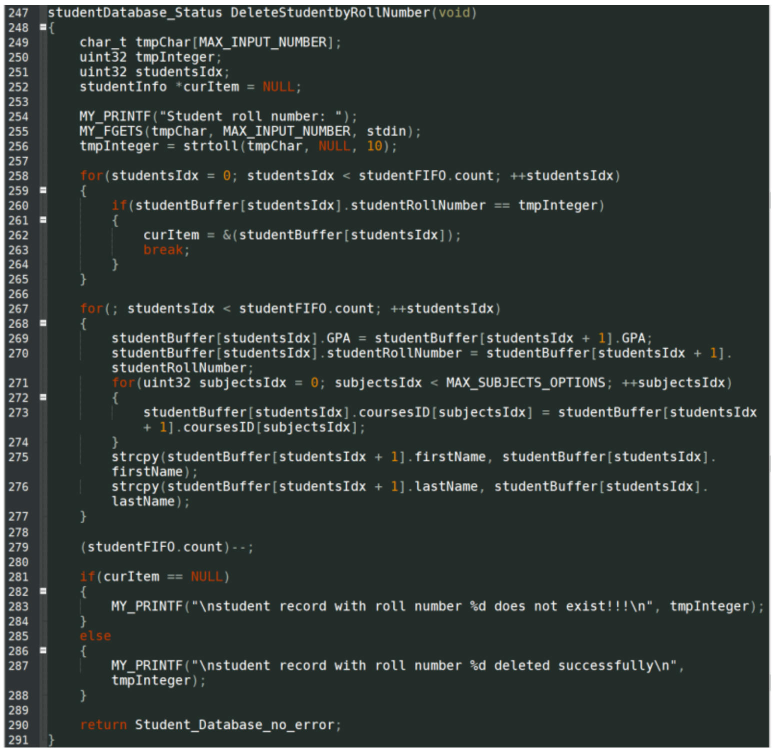
```c
studentDatabase_Status FindStudentsByCourseID(void)
{
    char_t tmpChar[MAX_INPUT_NUMBER];
    uint32 tmpInteger;
    studentInfo *curItem = NULL;

    MY_PRINTF("Student course ID: ");
    MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
    tmpInteger = strtoll(tmpChar, NULL, 10);

    for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        for(uint32 subjectsIdx = 0; subjectsIdx < MAX_SUBJECTS_OPTIONS; ++subjectsIdx)
        {
            if(studentBuffer[studentsIdx].coursesID[subjectsIdx] == tmpInteger)
            {
                if(curItem == NULL)
                {
                    MY_PRINTF("\nThe student(s) details are\n");
                }
                curItem = &(studentBuffer[studentsIdx]);
                MY_PRINTF("---------------------------------------------\n");
                MY_PRINTF("Roll Number is: %d\n", curItem->studentRollNumber);
                MY_PRINTF("First name is: %s\n", curItem->firstName);
                MY_PRINTF("Last name is: %s\n", curItem->lastName);
            }
        }
    }

    if(curItem == NULL)
    {
        MY_PRINTF("There are no students currently enrolled in course ID %d\n", tmpInteger
        );
    }

    return Student_Database_no_error;
}
```

- *FindStudentTotalNumber()*



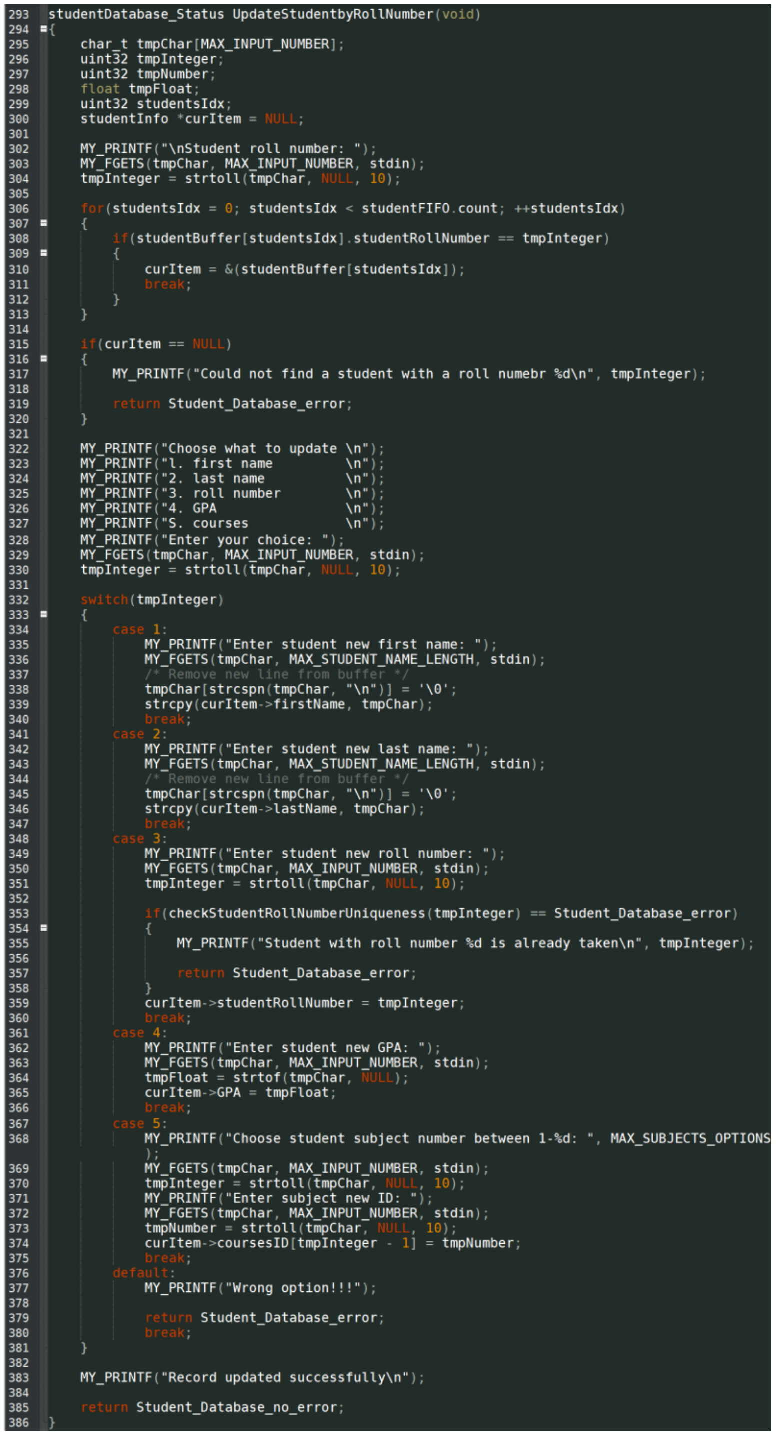
```c
studentDatabase_Status FindStudentTotalNumber(void)
{
    MY_PRINTF("\nThe total number of students is: %d student(s)\n", studentFIFO.count);
    MY_PRINTF("You can add up to: %d student(s)\n", MAX_STUDENTS_NUMBER);
    MY_PRINTF("You can add %d more student(s)\n", MAX_STUDENTS_NUMBER - studentFIFO.count
    );
    MY_PRINTF("---------------------------------------------\n");

    return Student_Database_no_error;
}
```

- *DeleteStudentbyRollNumber()*



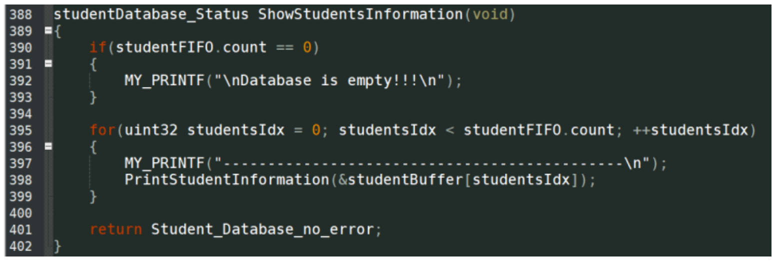
```c
studentDatabase_Status DeleteStudentbyRollNumber(void)
{
    char_t tmpChar[MAX_INPUT_NUMBER];
    uint32 tmpInteger;
    uint32 studentsIdx;
    studentInfo *curItem = NULL;

    MY_PRINTF("Student roll number: ");
    MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
    tmpInteger = strtoll(tmpChar, NULL, 10);

    for(studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        if(studentBuffer[studentsIdx].studentRollNumber == tmpInteger)
        {
            curItem = &(studentBuffer[studentsIdx]);
            break;
        }
    }

    for(; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        studentBuffer[studentsIdx].GPA = studentBuffer[studentsIdx + 1].GPA;
        studentBuffer[studentsIdx].studentRollNumber = studentBuffer[studentsIdx + 1].
        studentRollNumber;
        for(uint32 subjectsIdx = 0; subjectsIdx < MAX_SUBJECTS_OPTIONS; ++subjectsIdx)
        {
            studentBuffer[studentsIdx].coursesID[subjectsIdx] = studentBuffer[studentsIdx
            + 1].coursesID[subjectsIdx];
        }
        strcpy(studentBuffer[studentsIdx + 1].firstName, studentBuffer[studentsIdx].
        firstName);
        strcpy(studentBuffer[studentsIdx + 1].lastName, studentBuffer[studentsIdx].
        lastName);
    }

    (studentFIFO.count)--;

    if(curItem == NULL)
    {
        MY_PRINTF("\nstudent record with roll number %d does not exist!!!\n", tmpInteger);
    }
    else
    {
        MY_PRINTF("\nstudent record with roll number %d deleted successfully\n",
        tmpInteger);
    }

    return Student_Database_no_error;
}
```

- *UpdateStudentbyRollNumber()*



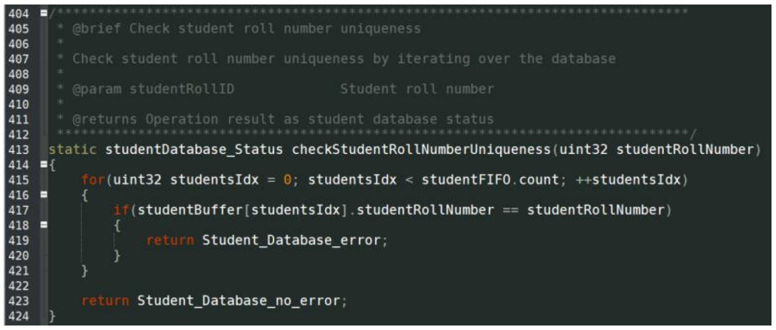
```c
studentDatabase_Status UpdateStudentbyRollNumber(void)
{
    char_t tmpChar[MAX_INPUT_NUMBER];
    uint32 tmpInteger;
    uint32 tmpNumber;
    float tmpFloat;
    uint32 studentsIdx;
    studentInfo *curItem = NULL;

    MY_PRINTF("\nStudent roll number: ");
    MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
    tmpInteger = strtoll(tmpChar, NULL, 10);

    for(studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        if(studentBuffer[studentsIdx].studentRollNumber == tmpInteger)
        {
            curItem = &(studentBuffer[studentsIdx]);
            break;
        }
    }

    if(curItem == NULL)
    {
        MY_PRINTF("Could not find a student with a roll numebr %d\n", tmpInteger);

        return Student_Database_error;
    }

    MY_PRINTF("Choose what to update \n");
    MY_PRINTF("1. first name         \n");
    MY_PRINTF("2. last name          \n");
    MY_PRINTF("3. roll number        \n");
    MY_PRINTF("4. GPA                \n");
    MY_PRINTF("5. courses            \n");
    MY_PRINTF("Enter your choice: ");
    MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
    tmpInteger = strtoll(tmpChar, NULL, 10);

    switch(tmpInteger)
    {
        case 1:
            MY_PRINTF("Enter student new first name: ");
            MY_FGETS(tmpChar, MAX_STUDENT_NAME_LENGTH, stdin);
            /* Remove new line from buffer */
            tmpChar[strcspn(tmpChar, "\n")] = '\0';
            strcpy(curItem->firstName, tmpChar);
            break;
        case 2:
            MY_PRINTF("Enter student new last name: ");
            MY_FGETS(tmpChar, MAX_STUDENT_NAME_LENGTH, stdin);
            /* Remove new line from buffer */
            tmpChar[strcspn(tmpChar, "\n")] = '\0';
            strcpy(curItem->lastName, tmpChar);
            break;
        case 3:
            MY_PRINTF("Enter student new roll number: ");
            MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
            tmpInteger = strtoll(tmpChar, NULL, 10);

            if(checkStudentRollNumberUniqueness(tmpInteger) == Student_Database_error)
            {
                MY_PRINTF("Student with roll number %d is already taken\n", tmpInteger);

                return Student_Database_error;
            }
            curItem->studentRollNumber = tmpInteger;
            break;
        case 4:
            MY_PRINTF("Enter student new GPA: ");
            MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
            tmpFloat = strtof(tmpChar, NULL);
            curItem->GPA = tmpFloat;
            break;
        case 5:
            MY_PRINTF("Choose student subject number between 1-%d: ", MAX_SUBJECTS_OPTIONS
            );
            MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
            tmpInteger = strtoll(tmpChar, NULL, 10);
            MY_PRINTF("Enter subject new ID: ");
            MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
            tmpNumber = strtoll(tmpChar, NULL, 10);
            curItem->coursesID[tmpInteger - 1] = tmpNumber;
            break;
        default:
            MY_PRINTF("Wrong option!!!");

            return Student_Database_error;
            break;
    }

    MY_PRINTF("Record updated successfully\n");

    return Student_Database_no_error;
}
```

○ *ShowStudentsInformation()*

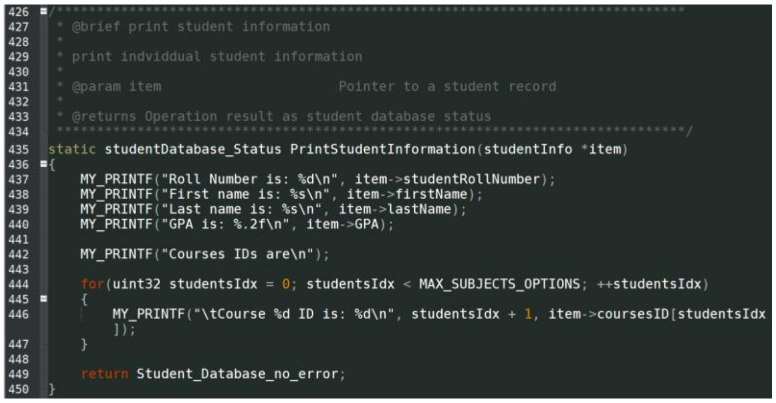


```c
studentDatabase_Status ShowStudentsInformation(void)
{
    if(studentFIFO.count == 0)
    {
        MY_PRINTF("\nDatabase is empty!!!\n");
    }

    for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        MY_PRINTF("--------------------------------------------\n");
        PrintStudentInformation(&studentBuffer[studentsIdx]);
    }

    return Student_Database_no_error;
}
```

○ *Helper Functions*



```c
/******************************************************************************
 * @brief Check student roll number uniqueness
 *
 * Check student roll number uniqueness by iterating over the database
 *
 * @param studentRollID             Student roll number
 *
 * @returns Operation result as student database status
 ******************************************************************************/
static studentDatabase_Status checkStudentRollNumberUniqueness(uint32 studentRollNumber)
{
    for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
    {
        if(studentBuffer[studentsIdx].studentRollNumber == studentRollNumber)
        {
            return Student_Database_error;
        }
    }

    return Student_Database_no_error;
}
```



```c
/******************************************************************************
 * @brief print student information
 *
 * print indviddual student information
 *
 * @param item                      Pointer to a student record
 *
 * @returns Operation result as student database status
 ******************************************************************************/
static studentDatabase_Status PrintStudentInformation(studentInfo *item)
{
    MY_PRINTF("Roll Number is: %d\n", item->studentRollNumber);
    MY_PRINTF("First name is: %s\n", item->firstName);
    MY_PRINTF("Last name is: %s\n", item->lastName);
    MY_PRINTF("GPA is: %.2f\n", item->GPA);

    MY_PRINTF("Courses IDs are\n");

    for(uint32 studentsIdx = 0; studentsIdx < MAX_SUBJECTS_OPTIONS; ++studentsIdx)
    {
        MY_PRINTF("\tCourse %d ID is: %d\n", studentsIdx + 1, item->coursesID[studentsIdx
        ]);
    }

    return Student_Database_no_error;
}
```