

***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
- C Code
  - main.c
  - student\_database.h (\*partial)
  - studentDatabaseInit()
  - AddStudentManually()
  - AddStudentFromFile()
  - FindStudentByRollNumber()
  - FindStudentByFirstName()
  - FindStudentsByCourseID()
  - FindStudentTotalNumber()
  - DeleteStudentbyRollNumber()
  - UpdateStudentbyRollNumber()
  - ShowStudentsInformation()
  - Helper Functions

# ● 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

```
9
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include "platform_types.h"
13 #include "student_database.h"
14 #include "fifo.h"
15
16 int main()
17 {
18     char_t option[MAX_MENU_OPTIONS_CHARACTERS];
19     studentDatabaseInit();
20
21     while(TRUE)
22     {
23         /* print the main menu */
24         MY_PRINTF("-----\n");
25         MY_PRINTF("Choose The Task that you want to perform: \n");
26         MY_PRINTF("-----\n");
27         MY_PRINTF("1. Add the Student Details Manually \n");
28         MY_PRINTF("2. Add the Student Details From Text File \n");
29         MY_PRINTF("3. Find the Student Details by Roll Number \n");
30         MY_PRINTF("4. Find the Student Details by First Name \n");
31         MY_PRINTF("5. Find the Student Details by Course Id \n");
32         MY_PRINTF("6. Find the Total number of Students \n");
33         MY_PRINTF("7. Delete the Students Details by Roll Number \n");
34         MY_PRINTF("8. Update the Students Details by Roll Number \n");
35         MY_PRINTF("9. Show all information \n");
36         MY_PRINTF("10. To Exit \n");
37         MY_PRINTF("-----\n");
38
39         MY_PRINTF("\nEnter your choice to perform the task: ");
40         MY_FGETS(option, MAX_MENU_OPTIONS_CHARACTERS, stdin);
41
42         //Options switching
43         switch(strtol(option, NULL, 10))
44         {
45             case 1:
46                 AddStudentManually();
47                 break;
48             case 2:
49                 AddStudentFromFile();
50                 break;
51             case 3:
52                 FindStudentByRollNumber();
53                 break;
54             case 4:
55                 FindStudentByFirstName();
56                 break;
57             case 5:
58                 FindStudentsByCourseID();
59                 break;
60             case 6:
61                 FindStudentTotalNumber();
62                 break;
63             case 7:
64                 DeleteStudentbyRollNumber();
65                 break;
66             case 8:
67                 UpdateStudentbyRollNumber();
68                 break;
69             case 9:
70                 /* Student_Database_head->next->next = Student_Database_head; */
71                 ShowStudentsInformation();
72                 break;
73             case 10:
74                 MY_PRINTF("Exiting...\n");
75                 exit(EXIT_SUCCESS);
76                 break;
77             default:
78                 MY_PRINTF("Wrong Option!!!\n");
79                 break;
80         }
81     }
82
83     return EXIT_SUCCESS;
84 }
85
```

## ○ *student\_database.h (\*partial)*

```
9  #ifndef STUDENT_DATABASE_H_
10 #define STUDENT_DATABASE_H_
11
12 #include "platform_types.h"
13
14 #define MAX_STUDENT_NAME_LENGTH      (50)
15 #define MAX_STUDENTS_NUMBER          (55)
16 #define MAX_MENU_OPTIONS_CHARACTERS (5)
17 #define MAX_SUBJECTS_OPTIONS         (5)
18 #define MAX_INPUT_NUMBER              (11)
19
20 /* my printf macro */
21 #define MY_PRINTF(...)                {printf( _VA_ARGS_ ); \
22                                         fflush(stdout); \
23                                         fflush(stdin);}
24
25 /* my fgets macro */
26 #define MY_FGETS(...)                {fgets( _VA_ARGS_ ); \
27                                         fflush(stdin);}
28
29 /* Student database typedef */
30 typedef struct
31 {
32     char_t firstName[MAX_STUDENT_NAME_LENGTH];
33     char_t lastName[MAX_STUDENT_NAME_LENGTH];
34     uint32 studentRollNumber;
35     float GPA;
36     uint32 coursesID[MAX_SUBJECTS_OPTIONS];
37 } studentInfo;
38
39 /* Student database status enumeration */
40 typedef enum
41 {
42     Student_Database_error,
43     Student_Database_no_error,
44     Student_Database_empty,
45     Student_Database_done
46 } studentDatabase_Status;
47
```

## ○ *studentDatabaseInit()*

```
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <string.h>
13 #include "platform_types.h"
14 #include "student_database.h"
15 #include "fifo.h"
16
17 static studentDatabase_Status checkStudentRollNumberUniqueness(uint32 studentRollID);
18 static studentDatabase_Status PrintStudentInformation(studentInfo *item);
19
20 static FIFO_Buffer_t studentFIFO;
21 static studentInfo studentBuffer[MAX_STUDENTS_NUMBER];
22
23 void studentDatabaseInit(void)
24 {
25     /* Check for a successful initiation */
26     if(FIFO_Init(&studentFIFO, studentBuffer, MAX_STUDENTS_NUMBER) != FIFO_no_error)
27     {
28         MY_PRINTF("FIFO Initialization failed exiting...\n");
29         exit(EXIT_FAILURE);
30     }
31 }
```

## ○ AddStudentManually()

```
33 studentDatabase_Status AddStudentManually(void)
34 {
35     char t tmpChar[MAX_INPUT_NUMBER];
36     uint32 tmpInteger;
37     float tmpFloat;
38
39     MY_PRINTF("Student roll number: ");
40     MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
41     tmpInteger = strtoll(tmpChar, NULL, 10);
42
43     if(checkStudentRollNumberUniqueness(tmpInteger) == Student_Database_error)
44     {
45         MY_PRINTF("Student roll number is already taken\n");
46         return Student_Database_error;
47     }
48
49     studentBuffer[studentFIFO.count].studentRollNumber = tmpInteger;
50
51     MY_PRINTF("Student first name: ");
52     /* Remove new line from buffer */
53     tmpChar[strcspn(tmpChar, "\n")] = '\0';
54     MY_FGETS(studentBuffer[studentFIFO.count].firstName, MAX_STUDENT_NAME_LENGTH, stdin);
55
56     MY_PRINTF("Student last name: ");
57     /* Remove new line from buffer */
58     tmpChar[strcspn(tmpChar, "\n")] = '\0';
59     MY_FGETS(studentBuffer[studentFIFO.count].lastName, MAX_STUDENT_NAME_LENGTH, stdin);
60
61     MY_PRINTF("Student GPA: ");
62     MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
63     tmpFloat = strtod(tmpChar, NULL);
64     studentBuffer[studentFIFO.count].GPA = tmpFloat;
65
66     MY_PRINTF("Student courses ID\n");
67     for(uint32 studentsIdx = 0; studentsIdx < MAX_SUBJECTS_OPTIONS; ++studentsIdx)
68     {
69         MY_PRINTF("Course %d ID: ", studentsIdx + 1);
70         MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
71         tmpInteger = strtoll(tmpChar, NULL, 10);
72         studentBuffer[studentFIFO.count].coursesID[studentsIdx] = tmpInteger;
73     }
74
75     FIFO_Enqueue(&studentFIFO, &studentBuffer[studentFIFO.count]);
76     MY_PRINTF("Student details was added successfully\n");
77
78
79     return Student_Database_no_error;
80 }
81
82
```

## ○ AddStudentFromFile()

```
83 studentDatabase_Status AddStudentFromFile(void)
84 = {
85     FILE *studentFile = fopen( "studentsFile.txt" , "r" );
86     boolean isEnqueue;
87
88     /* Check if fopen() succeeded */
89     if(studentFile == NULL)
90     {
91         printf("There was a problem opening studentFile.txt!!!\n");
92         printf("Unable to add students from file\n");
93         return Student_Database_error;
94     }
95
96     /* While end of file not reached */
97     while(!feof(studentFile))
98     {
99         isEnqueue = TRUE;
100         fscanf(studentFile, "%d" , &studentBuffer[studentFIFO.count].\
101             studentRollNumber);
102         if(checkStudentRollNumberUniqueness(studentBuffer[studentFIFO.count].\
103             studentRollNumber) == Student_Database_error)
104         {
105             MY_PRINTF("Student roll number %d is already taken\n",\
106                 studentBuffer[studentFIFO.count].studentRollNumber);
107             isEnqueue = FALSE;
108         }
109
110         fscanf(studentFile, "%s %s %f" , studentBuffer[studentFIFO.count].\
111             firstName, studentBuffer[studentFIFO.count].lastName,\
112             &studentBuffer[studentFIFO.count].GPA);
113
114         for(uint32 studentsIdx = 0; studentsIdx < MAX_SUBJECTS_OPTIONS; ++studentsIdx)
115         {
116             fscanf(studentFile, "%d" , &studentBuffer[studentFIFO.count].\
117                 coursesID[studentsIdx]);
118         }
119
120         if(isEnqueue == TRUE)
121         {
122             MY_PRINTF("Student roll number %d was added successfully\n",\
123                 studentBuffer[studentFIFO.count].studentRollNumber);
124             FIFO_Enqueue(&studentFIFO, &studentBuffer[studentFIFO.count]);
125         }
126     }
127
128     /* Close file after reading */
129     fclose(studentFile);
130
131     MY_PRINTF("Student details was added successfully from file\n");
132
133     return Student_Database_no_error;
134 }
```

## ○ FindStudentByRollNumber()

```
136 studentDatabase_Status FindStudentByRollNumber(void)
137 = {
138     char t tmpChar[MAX_INPUT_NUMBER];
139     uint32 tmpInteger;
140     studentInfo *curItem = NULL;
141
142     MY_PRINTF("Student roll number: ");
143     MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
144     tmpInteger = strtoll(tmpChar, NULL, 10);
145
146     for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
147     {
148         if(studentBuffer[studentsIdx].studentRollNumber == tmpInteger)
149         {
150             curItem = &(studentBuffer[studentsIdx]);
151             break;
152         }
153     }
154
155     if(curItem != NULL)
156     {
157         MY_PRINTF("-----\n");
158         PrintStudentInformation(curItem);
159     }
160     else
161     {
162         MY_PRINTF("Could not find a student roll number %d\n", tmpInteger);
163     }
164
165     return Student_Database_no_error;
166 }
```

## ○ FindStudentByFirstName()

```
168 studentDatabase_Status FindStudentByFirstName(void)
169 = {
170     char_t tmpChar[MAX_STUDENT_NAME_LENGTH];
171     studentInfo *curItem = NULL;
172
173     MY_PRINTF("Student first name: ");
174     MY_FGETS(tmpChar, MAX_STUDENT_NAME_LENGTH, stdin);
175     /* Remove new line from buffer */
176     tmpChar[strcspn(tmpChar, "\n")] = '\0';
177
178     for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
179     {
180         if(strcmp(tmpChar, studentBuffer[studentsIdx].firstName) == 0)
181         {
182             curItem = &(studentBuffer[studentsIdx]);
183             break;
184         }
185     }
186
187     if(curItem != NULL)
188     {
189         MY_PRINTF("-----\n");
190         PrintStudentInformation(curItem);
191     }
192     else
193     {
194         MY_PRINTF("Could not find a student with a first name %s\n", tmpChar);
195     }
196
197     return Student_Database_no_error;
198 }
```

## ○ FindStudentsByCourseID()

```
200 studentDatabase_Status FindStudentsByCourseID(void)
201 = {
202     char_t tmpChar[MAX_INPUT_NUMBER];
203     uint32 tmpInteger;
204     studentInfo *curItem = NULL;
205
206     MY_PRINTF("Student course ID: ");
207     MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
208     tmpInteger = strtoll(tmpChar, NULL, 10);
209
210     for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
211     {
212         for(uint32 subjectsIdx = 0; subjectsIdx < MAX_SUBJECTS_OPTIONS; ++subjectsIdx)
213         {
214             if(studentBuffer[studentsIdx].coursesID[subjectsIdx] == tmpInteger)
215             {
216                 if(curItem == NULL)
217                 {
218                     MY_PRINTF("\nThe student(s) details are\n");
219                     curItem = &(studentBuffer[studentsIdx]);
220                     MY_PRINTF("-----\n");
221                     MY_PRINTF("Roll Number is: %d\n", curItem->studentRollNumber);
222                     MY_PRINTF("First name is: %s\n", curItem->firstName);
223                     MY_PRINTF("Last name is: %s\n", curItem->lastName);
224                 }
225             }
226         }
227     }
228
229     if(curItem == NULL)
230     {
231         MY_PRINTF("There are no students currently enrolled in course ID %d\n", tmpInteger);
232     }
233
234     return Student_Database_no_error;
235 }
```

## ○ FindStudentTotalNumber()

```
237 studentDatabase_Status FindStudentTotalNumber(void)
238 = {
239     MY_PRINTF("\nThe total number of students is: %d student(s)\n", studentFIFO.count);
240     MY_PRINTF("You can add up to: %d student(s)\n", MAX_STUDENTS_NUMBER);
241     MY_PRINTF("You can add %d more student(s)\n", MAX_STUDENTS_NUMBER - studentFIFO.count);
242     MY_PRINTF("-----\n");
243
244     return Student_Database_no_error;
245 }
```



## ○ DeleteStudentbyRollNumber()

```
247 studentDatabase_Status DeleteStudentbyRollNumber(void)
248 {
249     char_t tmpChar[MAX_INPUT_NUMBER];
250     uint32 tmpInteger;
251     uint32 studentsIdx;
252     studentInfo *curItem = NULL;
253
254     MY_PRINTF("Student roll number: ");
255     MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
256     tmpInteger = strtoll(tmpChar, NULL, 10);
257
258     for(studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
259     {
260         if(studentBuffer[studentsIdx].studentRollNumber == tmpInteger)
261         {
262             curItem = &(studentBuffer[studentsIdx]);
263             break;
264         }
265     }
266
267     for(; studentsIdx < studentFIFO.count; ++studentsIdx)
268     {
269         studentBuffer[studentsIdx].GPA = studentBuffer[studentsIdx + 1].GPA;
270         studentBuffer[studentsIdx].studentRollNumber = studentBuffer[studentsIdx + 1].
            studentRollNumber;
271         for(uint32 subjectsIdx = 0; subjectsIdx < MAX_SUBJECTS_OPTIONS; ++subjectsIdx)
272         {
273             studentBuffer[studentsIdx].coursesID[subjectsIdx] = studentBuffer[studentsIdx
                + 1].coursesID[subjectsIdx];
274         }
275         strcpy(studentBuffer[studentsIdx + 1].firstName, studentBuffer[studentsIdx].
            firstName);
276         strcpy(studentBuffer[studentsIdx + 1].lastName, studentBuffer[studentsIdx].
            lastName);
277     }
278     (studentFIFO.count)--;
279
280     if(curItem == NULL)
281     {
282         MY_PRINTF("\nstudent record with roll number %d does not exist!!!\n", tmpInteger);
283     }
284     else
285     {
286         MY_PRINTF("\nstudent record with roll number %d deleted successfully\n",
            tmpInteger);
287     }
288
289     return Student_Database_no_error;
290 }
291 }
```

## ○ UpdateStudentbyRollNumber()

```
293 studentDatabase_Status UpdateStudentbyRollNumber(void)
294 = {
295     char_t tmpChar[MAX_INPUT_NUMBER];
296     uint32 tmpInteger;
297     uint32 tmpNumber;
298     float tmpFloat;
299     uint32 studentsIdx;
300     studentInfo *curItem = NULL;
301
302     MY_PRINTF("\nStudent roll number: ");
303     MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
304     tmpInteger = strtoll(tmpChar, NULL, 10);
305
306     for(studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
307     {
308         if(studentBuffer[studentsIdx].studentRollNumber == tmpInteger)
309         {
310             curItem = &(studentBuffer[studentsIdx]);
311             break;
312         }
313     }
314
315     if(curItem == NULL)
316     {
317         MY_PRINTF("Could not find a student with a roll numebr %d\n", tmpInteger);
318         return Student_Database_error;
319     }
320
321     MY_PRINTF("Choose what to update \n");
322     MY_PRINTF("1. first name      \n");
323     MY_PRINTF("2. last name       \n");
324     MY_PRINTF("3. roll number      \n");
325     MY_PRINTF("4. GPA              \n");
326     MY_PRINTF("5. courses          \n");
327
328     MY_PRINTF("Enter your choice: ");
329     MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
330     tmpInteger = strtoll(tmpChar, NULL, 10);
331
332     switch(tmpInteger)
333     {
334     case 1:
335         MY_PRINTF("Enter student new first name: ");
336         MY_FGETS(tmpChar, MAX_STUDENT_NAME_LENGTH, stdin);
337         /* Remove new line from buffer */
338         tmpChar[strcspn(tmpChar, "\n")] = '\0';
339         strcpy(curItem->firstName, tmpChar);
340         break;
341     case 2:
342         MY_PRINTF("Enter student new last name: ");
343         MY_FGETS(tmpChar, MAX_STUDENT_NAME_LENGTH, stdin);
344         /* Remove new line from buffer */
345         tmpChar[strcspn(tmpChar, "\n")] = '\0';
346         strcpy(curItem->lastName, tmpChar);
347         break;
348     case 3:
349         MY_PRINTF("Enter student new roll number: ");
350         MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
351         tmpInteger = strtoll(tmpChar, NULL, 10);
352
353         if(checkStudentRollNumberUniqueness(tmpInteger) == Student_Database_error)
354         {
355             MY_PRINTF("Student with roll number %d is already taken\n", tmpInteger);
356             return Student_Database_error;
357         }
358         curItem->studentRollNumber = tmpInteger;
359         break;
360     case 4:
361         MY_PRINTF("Enter student new GPA: ");
362         MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
363         tmpFloat = strtod(tmpChar, NULL);
364         curItem->GPA = tmpFloat;
365         break;
366     case 5:
367         MY_PRINTF("Choose student subject number between 1-%d: ", MAX_SUBJECTS_OPTIONS);
368         MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
369         tmpInteger = strtoll(tmpChar, NULL, 10);
370         MY_PRINTF("Enter subject new ID: ");
371         MY_FGETS(tmpChar, MAX_INPUT_NUMBER, stdin);
372         tmpNumber = strtoll(tmpChar, NULL, 10);
373         curItem->coursesID[tmpInteger - 1] = tmpNumber;
374         break;
375     default:
376         MY_PRINTF("Wrong option!!!");
377         return Student_Database_error;
378         break;
379     }
380
381     MY_PRINTF("Record updated successfully\n");
382
383     return Student_Database_no_error;
384 }
385
386 }
```

## ○ *ShowStudentsInformation()*

```
388 studentDatabase_Status ShowStudentsInformation(void)
389 {
390     if(studentFIFO.count == 0)
391     {
392         MY_PRINTF("\nDatabase is empty!!\n");
393     }
394
395     for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
396     {
397         MY_PRINTF("-----\n");
398         PrintStudentInformation(&studentBuffer[studentsIdx]);
399     }
400
401     return Student_Database_no_error;
402 }
```

## ○ *Helper Functions*

```
404 * @brief Check student roll number uniqueness
405 *
406 * Check student roll number uniqueness by iterating over the database
407 *
408 * @param studentRollID      Student roll number
409 *
410 * @returns Operation result as student database status
411 *
412 * =====/
413 static studentDatabase_Status checkStudentRollNumberUniqueness(uint32 studentRollNumber)
414 {
415     for(uint32 studentsIdx = 0; studentsIdx < studentFIFO.count; ++studentsIdx)
416     {
417         if(studentBuffer[studentsIdx].studentRollNumber == studentRollNumber)
418         {
419             return Student_Database_error;
420         }
421     }
422
423     return Student_Database_no_error;
424 }
```

```
426 * @brief print student information
427 *
428 * print individual student information
429 *
430 * @param item                Pointer to a student record
431 *
432 * @returns Operation result as student database status
433 *
434 * =====/
435 static studentDatabase_Status PrintStudentInformation(studentInfo *item)
436 {
437     MY_PRINTF("Roll Number is: %d\n", item->studentRollNumber);
438     MY_PRINTF("First name is: %s\n", item->firstName);
439     MY_PRINTF("Last name is: %s\n", item->lastName);
440     MY_PRINTF("GPA is: %.2f\n", item->GPA);
441
442     MY_PRINTF("Courses IDs are\n");
443
444     for(uint32 studentsIdx = 0; studentsIdx < MAX_SUBJECTS_OPTIONS; ++studentsIdx)
445     {
446         MY_PRINTF("\tCourse %d ID is: %d\n", studentsIdx + 1, item->coursesID[studentsIdx]);
447     }
448
449     return Student_Database_no_error;
450 }
```