Student Database System Report

Embedded Systems Online Diploma

Eng. Mahmoud Essam Mahfouz

Progress Page

6/12/2022

Problem Statement

A simple software for student information management system which can perform the following operations:

1. Store first name of the student.
2. Store last name of the student.
3. Store unique roll number for every student.
4. Store GPA for every student.
5. Store courses registered by the student.

Approach

The idea is to form an individual functions for every operation. All the functions are unified to form software.

1. Add student details manually.
2. Add student details from file.
3. Find the student by the given roll number.
4. Find the student by the given first name.
5. Find the student registered in a course.
6. Count number of students.
7. Delete a student by the given roll number.
8. Update a student by the given roll number.
9. Print all student's data.
10. Exit the program.

Main.c

```c
#include "SDB_Queue.h"
DataBase_t base;
element_type arr[50];
int main()
{
    DataBase_init(&base, arr);
    while(1)
    {
        char temp_text[40];
        DPRINTF("\n==================================");
        DPRINTF("\n\tChoose one of the following options");
        DPRINTF("\n1: Add student manually");
        DPRINTF("\n2: Add students from file");
        DPRINTF("\n3: Find student by ID");
        DPRINTF("\n4: Find student by the first name");
        DPRINTF("\n5: Find Students enrolled in one course");
        DPRINTF("\n6: Update Student");
        DPRINTF("\n7: Delete Student");
        DPRINTF("\n8: View All Students");
        DPRINTF("\n9: Print Students Count");
        DPRINTF("\n10: Exit");
        DPRINTF("\nEnter Your option: ");
        gets(temp_text);
        switch(atoi(temp_text))
        {
        case 1: Add_Student_Manually(&base);
        break;
        case 2: Add_Students_From_File(&base);
        break;
        case 3: Find_Student_By_ID(&base);
        break;
        case 4: Find_Student_By_FName(&base);
        break;
        case 5: Find_Students_in_Course(&base);
        break;
        case 6: Update_Student(&base);
        break;
        case 7: Delete_Student(&base, arr);
        break;
        case 8: view_All(&base);
        break;
        case 9: print_student_count(&base);
        break;
        case 10: exit(0);
        break;
        default:
            DPRINTF("\nWrong Option");
            break;
        }

    }
    return 1;
}
```

SDB_Queue.h

```c
#ifndef SDB_QUEUE_H_
#define SDB_QUEUE_H_
#define element_type SData_t
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

#define DPRINTF(...)    {fflush(stdout);\
        fflush(stdin);\
        printf(__VA_ARGS__);\
        fflush(stdout);\
        fflush(stdin);}

// Data types Definition
typedef struct
{
    int ID;
    float GPA;
    char FName[40];
    char LName[40];
    int course_id[5];
}SData_t;

typedef struct
{
    unsigned int length;
    unsigned int count;
    element_type* head;
    element_type* base;
    element_type* tail;
}DataBase_t;

typedef enum
{
    DataBase_no_error,
    DataBase_full,
    DataBase_empty,
    DataBase_null
}DataBase_status;

// APIs

DataBase_status DataBase_init(DataBase_t* Database,element_type* arr);
DataBase_status Add_Student_Manually(DataBase_t* Database);
DataBase_status Add_Students_From_File(DataBase_t* Database);
DataBase_status Find_Student_By_ID(DataBase_t* Database);
DataBase_status Find_Student_By_FName(DataBase_t* Database);
DataBase_status Find_Students_in_Course(DataBase_t* Database);
DataBase_status view_All(DataBase_t* Database);
DataBase_status print_student_count(DataBase_t* Database);
DataBase_status Update_Student(DataBase_t* Database);
DataBase_status Delete_Student(DataBase_t* Database,element_type* arr);
#endif /* SDB_QUEUE_H_ */
```

SDB_Queue.c

This file includes the functions used in the main program:

DataBase_init

```
DataBase_status DataBase_init(DataBase_t* Database,element_type* arr)
{
    DataBase_t *pDatabase = Database;
    pDatabase->base=arr;
    pDatabase->tail=arr;
    pDatabase->head=arr;
    pDatabase->count=0;
    pDatabase->length=50;
    DPRINTF("\tInitialization Done\n");
    return DataBase_no_error;
}
```

Add_Studnets_Manually

```
DataBase_status Add_Student_Manually(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->head;
    SData_t *id_check = pDatabase->base;
    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }
    // check if the database is full
    if(pDatabase->count == pDatabase->length)
    {
        DPRINTF("DataBase Full\n");
        return DataBase_full;
    }

    //Variables used
    int tmp_id,i;
    DPRINTF("Enter Student ID: ");
    scanf("%d",&tmp_id);
    // check if the id is in the database if yes --> return null
    for(i = 0; i<pDatabase->count; i++)
    {
        if(tmp_id == id_check->ID)
        {
            DPRINTF("The ID is already in the database choose another one\n");
            return DataBase_null;
        }
        else
            id_check++;
    }
    SData->ID = tmp_id;
    DPRINTF("Enter student GPA: ");
    scanf("%f",&SData->GPA);
    DPRINTF("Enter Student First Name: ");
    gets(SData->FName);
    DPRINTF("Enter Student Last Name: ");
    gets(SData->LName);
    for(i = 0 ; i< 5; i++)
    {
        DPRINTF("Enter course id %d : ",i+1);
        scanf("%d",&SData->course_id[i]);
    }
    pDatabase->count++;
    // Make the DataBase Circular
    if(pDatabase->head == (pDatabase->base + (pDatabase->length * sizeof(element_type))))
        pDatabase->head = pDatabase->base;
    else
        pDatabase->head++;

    DPRINTF("\n\tStudent Added Successfully\n");
    return DataBase_no_error;
}
```

Add_Students_From_File

```
DataBase_status Add_Students_From_File(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->head;
    SData_t *id_check = pDatabase->base;

    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }

    // check if the database is full
    if(pDatabase->count == pDatabase->length)
    {
        DPRINTF("DataBase Full\n");
        return DataBase_full;
    }

    //variables
    int i , tmp_id = 0, j;
    FILE* pInfo = fopen("info.txt", "r");
    if(pInfo == NULL)
    {
        DPRINTF("\n Failed To open the file\n");
        return DataBase_null;
    }
    rewind(pInfo);
    while(!feof(pInfo))
    {
        fscanf(pInfo, "%d" , &tmp_id);

        //check if the id in the database if yes --> return null
        for(i = 0; i<pDatabase->count; i++)
        {
            if(tmp_id == id_check->ID)
            {
                DPRINTF("The ID is already in the database choose another one\n");
                return DataBase_null;
            }
            else
                id_check++;
        }
```

```
        // add the id in the database
        SData->ID = tmp_id;

        // scan the remaining data
        fscanf(pInfo, "%f" ,&SData->GPA);
        fscanf(pInfo, "%s" ,SData->FName);
        fscanf(pInfo, "%s" ,SData->LName);
        for(j = 0 ; j < 5; j++)
        {
            fscanf(pInfo, "%d" ,&SData->course_id[j]);
        }

        // Increment the SData pointer to point to the next slot after scanning the data
        SData++;
        // Increment the count var by 1
        pDatabase->count++;

        // Make the DataBase Circular
        if(pDatabase->head == (pDatabase->base + (pDatabase->length * sizeof(element_type))))
            pDatabase->head = pDatabase->base;
        else
            pDatabase->head++;

        DPRINTF("\n\tStudent Added Successfully\n");


    }
    DPRINTF("-----------------File scanned Successfully-----------------\n");
    fclose(pInfo);
    return DataBase_no_error;
}
```

Find_Student_By_ID

```
DataBase_status Find_Student_By_ID(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->base;
    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }

    // check if the database is empty
    if(pDatabase->count == 0)
    {
        DPRINTF("DataBase Empty\n");
        return DataBase_empty;
    }
    //Variables used
    int tmp_id,i,j;
    DPRINTF("Enter Student ID: ");
    scanf("%d",&tmp_id);
    for(i=0 ;i < pDatabase->count; i++)
    {
        //check if the id in the database if yes --> print the student data
        if(tmp_id == SData->ID)
        {
            DPRINTF("Student Data are below\n");
            DPRINTF("Student GPA : %.2f\n",SData->GPA);
            DPRINTF("Student First Name is %s\n",SData->FName);
            DPRINTF("Student Last Name is %s\n",SData->LName);
            for(j = 0; j < 5; j++)
            {
                DPRINTF("Course id %d is %d\n",j+1,SData->course_id[j]);
            }
            return DataBase_no_error;
        }
        // if no --> make the pointer points to the next slot in the array
        else
            SData++;
    }
    DPRINTF("The id isn't in the DataBase\n");
    return DataBase_null;
}
```

Find_Student_By_FName

```
DataBase_status Find_Student_By_FName(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->base;
    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }

    // check if the database is empty
    if(pDatabase->count == 0)
    {
        DPRINTF("DataBase Empty\n");
        return DataBase_empty;
    }
    // Variables
    int i, j;
    char fname[40];

    DPRINTF("Enter the Student's First name: ");
    gets(fname);

    for(i = 0 ; i< pDatabase->count; i++)
    {
        // check if the scanned name is in the database if yes --> print student data
        if(strcmp(fname,SData->FName) == 0)
        {
            DPRINTF("Student Data are below\n");
            DPRINTF("Student ID : %d\n",SData->ID);
            DPRINTF("Student GPA : %.2f\n",SData->GPA);
            DPRINTF("Student First Name is %s\n",SData->FName);
            DPRINTF("Student Last Name is %s\n",SData->LName);
            for(j = 0; j < 5; j++)
            {
                DPRINTF("Course id %d is %d\n",j+1,SData->course_id[j]);
            }
            return DataBase_no_error;
        }
        // if no --> make the pointer points to the next slot in the array
        else
            SData++;
    }
    DPRINTF("There is no such first name in the database\n");
    return DataBase_null;
}
```

Find_Students_In_Course

```
Close
DataBase_status Find_Students_in_Course(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->base;
    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }

    // check if the database is empty
    if(pDatabase->count == 0)
    {
        DPRINTF("DataBase Empty\n");
        return DataBase_empty;
    }

    // Variables
    int i,j , c_id, counter = 0;
    DPRINTF("Enter course id: ");
    scanf("%d",&c_id);

    for(i = 0; i < pDatabase->count; i++)
    {
        for(j = 0; j < 5; j++)
        {
            //check if the course id is in the database if yes --> print all the students data enrolled in it
            if(c_id == SData->course_id[j])
            {
                DPRINTF("Students enrolled in this course are\n");
                DPRINTF("Student ID : %d\n",SData->ID);
                DPRINTF("Student GPA : %.2f\n",SData->GPA);
                DPRINTF("Student First Name is %s\n",SData->FName);
                DPRINTF("Student Last Name is %s\n",SData->LName);
                DPRINTF("----------------------------------------\n");
                counter++;
            }

        }
        // if no --> increment the pointer to point to the next slot in the array
        SData++;
```

```
    }
    // if the counter > 0 means that there is at least one student enrolled
    if(counter > 0)
        return DataBase_no_error;
    // else there is no student enrolled
    else
    {
        DPRINTF("No student is enrolled in this course\n");
        return DataBase_null;
    }
}
```

Print_Students_Count

```
DataBase_status print_student_count(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    DPRINTF("The Student Count is %d\n",pDatabase->count);
    return DataBase_no_error;
}
```

Delete_Student

```c
DataBase_status Delete_Student(DataBase_t* Database,element_type* arr)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->base;

    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }

    // check if the database is empty
    if(pDatabase->count == 0)
    {
        DPRINTF("DataBase Empty\n");
        return DataBase_empty;
    }

    // variables
    int i, tmp_id, j;

    DPRINTF("Enter Student ID to delete: ");
    scanf("%d",&tmp_id);

    for(i = 0; i < pDatabase->count;i++)
    {
        // check if the scanned id is in the database if yes --> delete the student
        if(tmp_id == SData->ID)
        {
            //code to delete the student data
            for(j = i ;j < pDatabase->count; j++)
            {
                arr[j] = arr[j+1];
            }
            pDatabase->count--;
            pDatabase->head--;
            DPRINTF("Student Data Deleted Successfully\n");
            return DataBase_no_error;
        }
        // if no --> increment the pointer to point to the next slot in the array
        else
            SData++;
    }
    DPRINTF("This id isn't in the database\n");
    return DataBase_null;
}
```

Update_Stuedent

```c
DataBase_status Update_Student(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->base;

    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }

    // check if the database is empty
    if(pDatabase->count == 0)
    {
        DPRINTF("DataBase Empty\n");
        return DataBase_empty;
    }

    // Variables
    int choice, i, j, tmp_id;
    DPRINTF("Enter ID: ");
    scanf("%d",&tmp_id);


    for(i=0 ; i <pDatabase->count;i++)
    {
        if(tmp_id == SData->ID)
        {
            DPRINTF("\n\tChoose one of the following options");
            DPRINTF("\n1: Update Student's ID");
            DPRINTF("\n2: Update Student's GPA");
            DPRINTF("\n3: Update Student's First Name");
            DPRINTF("\n4: Update Student's Last Name");
            DPRINTF("\n5: Update Student's Course ID");
            DPRINTF("\nEnter your Choice: ");
            scanf("%d",&choice);

            switch(choice)
            {
            case 1:
                DPRINTF("Enter New ID: ");
                scanf("%d",&SData->ID);
                DPRINTF("\tStudent Data Update Successfully\n");
                break;
```

```
            switch(choice)
            {
            case 1:
                DPRINTF("Enter New ID: ");
                scanf("%d",&SData->ID);
                DPRINTF("\tStudent Data Update Successfully\n");
                break;
            case 2:
                DPRINTF("Enter New GPA: ");
                scanf("%f",&SData->GPA);
                DPRINTF("\tStudent Data Update Successfully\n");
                break;
            case 3:
                DPRINTF("Enter New First Name: ");
                gets(SData->FName);
                DPRINTF("\tStudent Data Update Successfully\n");
                break;
            case 4:
                DPRINTF("Enter New Last Name: ");
                gets(SData->LName);
                DPRINTF("\tStudent Data Update Successfully\n");
                break;
            case 5:
                for(j = 0; j < 5; j++)
                {
                    DPRINTF("Enter course %d id: ",j+1);
                    scanf("%d",&SData->course_id[j]);
                }
                DPRINTF("\tStudent Data Update Successfully\n");
                break;
            default:
                DPRINTF("\nWrong Option");
                return DataBase_null;
                break;
            }
        }
        else
            SData++;
    }
    DPRINTF("This ID isn't in the database\n");
    return DataBase_null;
}
```

View_All

```c
DataBase_status view_All(DataBase_t* Database)
{
    DataBase_t *pDatabase = Database;
    SData_t *SData = pDatabase->base;

    //check if there is anything wrong with the database
    if(!pDatabase->head || !pDatabase->base || !pDatabase || !pDatabase->tail)
    {
        DPRINTF("DataBase Failed\n");
        return DataBase_null;
    }

    // check if the database is empty
    if(pDatabase->count == 0)
    {
        DPRINTF("DataBase Empty\n");
        return DataBase_empty;
    }

    // Variables

    int i, j;
    for(i = 0 ; i < pDatabase->count; i++)
    {
        DPRINTF("Student number %d data are\n",i+1);
        DPRINTF("Student ID : %d\n",SData->ID);
        DPRINTF("Student GPA : %.2f\n",SData->GPA);
        DPRINTF("Student First Name is %s\n",SData->FName);
        DPRINTF("Student Last Name is %s\n",SData->LName);
        for(j = 0 ; j < 5; j++)
        {
            DPRINTF("course %d id is %d\n",j+1,SData->course_id[j])
        }
        DPRINTF("----------------------------------------\n");
        SData++;
    }
    return DataBase_no_error;
}
```