Date :   Page No :

Assignment Number

Problem Statement

Program in C to store name, roll number and marks of a multiple students, and to perform the following operations on the database :

1. Finding details of a particular student
2. Calculating the average marks in each subject among all students
3. Finding the details of the students who scored highest marks in each subject

Theory

The particulars of a student can be seen as a collection of heterogenous elements, like an integer containing his/her roll number, a string containing his/her name etc. This type of heterogenous collection can be managed using different arrays for each type of item in the collection, however handling of those multiple arrays itself would be a dauting task. C provides the `struct` construct to manage a collection of heterogenous items, which will be used throughout the program to identify or address a 'Student' entity. Each variable of this type therefore will contain all the details about each student, and hence an array of this type of variable can be seen as a collection of students. Rest of the program is merely the sum of separate small mathematical functions, like finding the average or the maximum of N items, here N being the number of students registered.

Algorithm

Input : The details of number of students to be registered, say N.
Output : Queries performed successfully or suitable unsuccessful message.
Data Structure Used : An array of structures, say students[1..N], where each structure contains the following members :

a) The roll number of a student, say roll
b) The name of the student, say name
c) The respective marks of the student in both subjects, say mar1 and mar2

Steps :

Algorithm_Entry(x)

```
Step 1 : Print "\nEnter details of student ", i, " : "
Step 2 : Print "\nRoll : "
Step 3 : Input roll
Step 4 : Print "\nName : "
Step 5 : Input name
Step 6 : Print "\nMarks in subject 1 : "
Step 7 : Input m1
Step 8 : Print "\nMarks in subject 2 : "
Step 9 : Input m2
Step 10 :       Set students[x] = Get_Student(name, roll, m1, m2)
    // Get_Student is a function which returns a new instance of the
    // 'Student' structure initialized with given arguments
```

Algorithm_Display(roll)

```
Step 1 : Set i = 1
Step 2 : Repeat through steps 2.a to 2.b while(i <= n)
    a) If(students[i].roll == roll)
        I.   Print "\nDetails of student"
        II.  Print "\n================="
        III. Print "\nRoll : ", students[i].roll
        IV.  Print "\nName : ", students[i].name
        V.   Print "\nMarks in =>"
        VI. Print "\nSubject 1 : ", students[i].mar1
        VII. Print "\nSubject 2 : ", students[i].mar2
        VIII. Return
        [End of if structure]
    b) Set i = i + 1
    [End of while loop]
Step 3 : Print "\nRecord does not exist for roll ", roll
```

Algorithm_Average()

```
Step 1 : Set i = 1, t1 = 0.0, t2 = 0.0, avg1 = 0.0, avg2 = 0.0
Step 2 : While(i <= n)
    a) Set t1 = t1 + students[i].mar1
    b) Set t2 = t2 + students[i].mar2
    c) Set i = i + 1
    [End of while loop]
Step 3 : Set avg1 = t1/n, avg2 = t2/n
Step 4 : Print "\nAverage marks in subject 1 : ", avg1
Step 5 : Print "\nAverage marks in subject 2 : ", avg2
```

Algorithm_Highest()

```
Step 1 : Set max1 = 0, max2 = 0, c1 = 0, c2 = 0, i=1
Step 2 : Repeat through steps 2.a to 2.b while(i <= n)
    a) If(students[i].mar1 > max1)
        I. Set max1 = students[i].mar1, c1 = i
        [End of if structure]
    b) If(students[i].mar2 > max2)
        I. Set max2 = students[i].mar2, c2 = i
        [End of if structure]
    c) Set i = i + 1
Step 3 : EndWhile
Step 4 : If(c1 == c2)
    a) Print "\nName : ", students[c1].name
    b) Print "\nRoll : ", students[c1].roll
    c) Print "\nHas scored highest marks in both subjects!"
Step 5 : Else
```

```
    a) Print "\nDetails of students who scored the highest marks"
    b) Print "\nName : ", students[c1].name
    c) Print "\nRoll : ", students[c1].roll
    d) Print "\nScored highest marks in subject 1"
    e) Print "\nName : ", students[c2].name
    f) Print "\nRoll : ", students[c2].roll
    g) Print "\nScored highest marks in subject 2"
    [End of if structure]
```

Algorithm_Main()

```
Step 1 : Print "\nEnter number of students : "
Step 2 : Input n
Step 3 : Repeat through steps 3.a to 3.b while(i <= n)
    a) Call Entry(i)
    b) Set i = i + 1
Step 4 : EndWhile
Step 5 : Set ch = 1
Step 6 : Repeat through steps 6.a to 6.g while(ch != 4)
    a) Print "\n1. Details of student\n2. Average marks\n3. Highest
       marks\n4. Exit"
    b) Print "\nEnter your choice : "
    c) Input ch
    d) If(ch == 1)
        I.   Print "\nEnter the roll no : "
        II.  Input roll
        III. Call Display(roll)
    e) Else If(ch == 2)
        I.   Call Average()
    f) Else If(ch == 3)
        I.   Call Highest()
    g) Else
        I.   Print "\nWrong choice!"
    [End of if structure]
    [End of while loop]
```

Source Code

```
#include <stdio.h>

struct Student{
	int roll;
	char name[20];
	int m1, m2;
};
struct Student stu[20];
int n, i;
void entry(int);
void average();
void highest();
void display(int);

int main(){
	int ch, r;
	printf("\nEnter number of students : ");
	scanf("%d", &n);
	for(i = 0;i < n;i++)
		entry(i); // Calling method entry
	do{
		printf("\n1. Details of student\n2. Average marks\n3. Highest 
marks\n4. Exit");
		printf("\nEnter your choice : ");
		scanf("%d", &ch);
		switch(ch){
			case 1: // Find the details of a particular student
				printf("\nEnter the roll no : ");
				scanf("%d", &r);
				display(r);
				break;
```

```
                case 2: // Displaying average marks
                        average();
                        break;
                case 3: // Displaying highest marks in each subject
                        highest();
                        break;
                case 4:
                        break;
                default:
                        printf("\nWrong choice!");
                        return 1;
        }
    } while(ch != 4);

    return 0;
}

// Definition of 'entry'
void entry(int i){
    printf("\nEnter the details of student %d : ", i+1);
    printf("\nEnter roll : ");
    scanf("%d", &stu[i].roll);
    printf("\nEnter name : ");
    scanf("%s", &stu[i].name);
    printf("\nEnter marks in subject 1 : ");
    scanf("%d", &stu[i].m1);
    printf("\nEnter marks in subject 2 : ");
    scanf("%d", &stu[i].m2);
}

// Definition of 'display'
void display(int r){
    for(i = 0;i <  n;i++){
        if(stu[i].roll == r){
                printf("\nDetails of student");
                printf("\n\tRoll : %d\n\tName : %s", stu[i].roll, stu[i].name);
                printf("\nMarks in ");
                printf("\n\tSubject 1 : %d", stu[i].m1);
                printf("\n\tSubject 2 : %d", stu[i].m2);
                break;
            }
        }
        if(i == n){
            printf("\nRecord does not exist for roll %d!", r);
        }
}

// Definition of method average
void average(){
        int t1 = 0, t2 = 0;
        float avg1, avg2;
        for(i = 0;i<n;){
            t1 += stu[i].m1; // Finding sum of all marks in sub1
            t2 += stu[i++].m2; // Finding sum of all marks in sub2
        }
        avg1 = t1/n; avg2 = t2/n;
        printf("\nAverage marks in sub 1 : %0.2f", avg1);
        printf("\nAverage marks in sub 2 : %0.2f", avg2);
}

//Definition of method highest
void highest(){
        int max1 = 0, max2 = 0, c1, c2;
        for(i = 0;i < n;i++){
            if(stu[i].m1 > max1){
                max1 = stu[i].m1;
                c1 = i;
            }
            if(stu[i].m2 > max2){
                max2 = stu[i].m1;
                c2 = i;
            }
        }
        if(c1 == c2){
            printf("\nDetails of student");
            printf("\n\tRoll : %d\n\tName : %s", stu[c1].roll, stu[c1].name);
            printf("\nScored highest marks in both the subjects!");
        }
        else{
            printf("\nDetails of students who scored the highest marks");
            printf("\n\tRoll : %d\n\tName : %s", stu[c1].roll, stu[c1].name);
            printf("\nScored the highest marks in subject 1!");
            printf("\n\tRoll : %d\n\tName : %s", stu[c2].roll, stu[c2].name);
            printf("\nScored the highest marks in subject 2!");
        }
}
```

Input and Output

```
Enter number of students : 2
Enter the details of student 1 :
Enter roll : 1
Enter name : ABC
Enter marks in subject 1 : 98
Enter marks in subject 2 : 87
Enter the details of student 2 :
Enter roll : 2
Enter name : DEF
Enter marks in subject 1 : 97
Enter marks in subject 2 : 92
1. Details of student
```

```
2. Average marks
3. Highest marks
4. Exit
Enter your choice : 1
Enter the roll no : 1
Details of student
                        Roll : 1
                        Name : ABC
Marks in
                        Subject 1 : 98
                        Subject 2 : 87
1. Details of student
2. Average marks
3. Highest marks
4. Exit
Enter your choice : 2
Average marks in sub 1 : 97.00
Average marks in sub 2 : 89.00
1. Details of student
2. Average marks
3. Highest marks
4. Exit
Enter your choice : 3
Details of student
                        Roll : 1
                        Name : ABC
Scored highest marks in both the subjects!
1. Details of student
2. Average marks
3. Highest marks
4. Exit
Enter your choice : 5
Wrong choice!
```

Discussion

1. Since the database is not dynamically allocated, but is merely statically defined, there is a fundamental limit of how many students we can enlist into it.
2. Names greater than 20 characters will overrun the name buffer, which will cause an easy buffer overflow exploit.
3. Downside of the contiguous array, the allocation may fail and the program may refuse to start if there is insufficient memory to statically allocate the array at runtime.