Storing and Sorting the Records of 20 Cricketers

Aman Raj
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Dhanbad, Jharkhand
iec2021050@iiita.ac.in

Vamsi K
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Hyderabad, Telangana
iec2021052@iiita.ac.in

Nimish Rajurkar
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Nanded, Maharashtra
iec2021051@iiita.ac.in

Sai Praneeth
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Hyderabad, Telangana
iec2021053@iiita.ac.in

***Abstract*— Given a record containing the Name, Age, No. of tests played and the average runs in each test. Create an array of structure to hold records of 20 such cricketer and then write a program to read these records and arrange them in ascending order by average runs.**

I. Introduction

We are given record containing the Name, Age, No. of tests played and the average runs in each test.
We will be using structure to store the information of the cricketers and qsort() to sort the cricketers by their average runs. For loops are also used to store the data in structure.

II. Structures in C

A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type. A structure can be easily explained by the following Fig.1

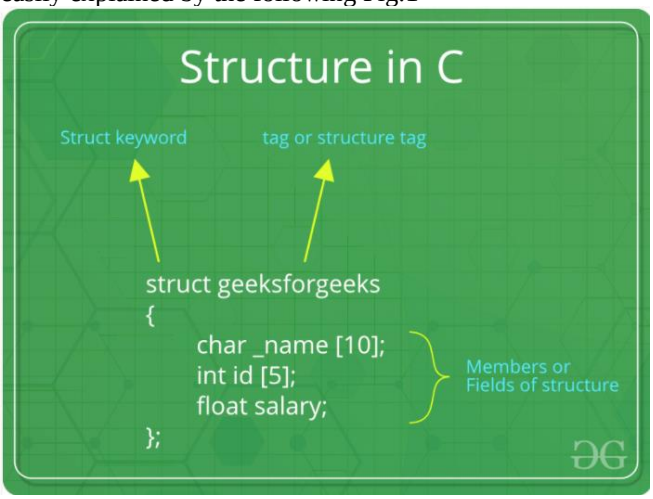

Fig.1:- Pictorial representation of a structure.

A structure variable can either be declared with structure declaration or as a separate declaration like basic types and Structure members are accessed using dot (.) operator as shown in Fig.2

```c
#include<stdio.h>

struct Point
{
    int x, y;
};

int main()
{
    struct Point p1 = {0, 1};

    // Accessing members of point p1
    p1.x = 20;
    printf ("x = %d, y = %d", p1.x, p1.y);

    return 0;
}
```

Fig.2:- Accessing a member in a structure.

III. For Loop

A for loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line. The basic form of declaring a for loop is by the syntax:

```
for (initialization; condition; increment;)
{
    // body of the loop
    // statements we want to execute
}
```

IV. qsort() function

Standard C library provides qsort() that can be used for sorting an array. As the name suggests, the function uses QuickSort algorithm to sort the given array. A prototype of qsort() is shown in the following Fig.3

```c
void qsort (void* base, size_t num, size_t size,
            int (*comparator)(const void*,const void*));
```

The key point about qsort() is comparator function *comparator*. The comparator function takes two arguments and contains logic to decide their relative order in sorted output. The idea is to provide flexibility so that qsort() can be used for any type (including user defined types) and can be used to obtain any desired order (increasing, decreasing or any other).
The comparator function takes two pointers as arguments (both type-casted to const void*) and defines the order of the elements by returning (in a stable and transitive manner as shown in fig.4

```
int comparator(const void* p1, const void* p2);
Return value meaning
<0 The element pointed by p1 goes before the element pointed by p2
0  The element pointed by p1 is equivalent to the element pointed by p2
>0 The element pointed by p1 goes after the element pointed by p2
```

V. PROCESS

The following procedure requires a few header files.

```
#include <stdio.h>
#include<string.h>
#include<stdlib.h>
```

First we a structure to store the required information of each cricketer. So we declare a structure of name 'Cricketer' and now
we initialize the following members in the structure.

1.Average runs
2.Name of the cricketer
3.Age
4.Number of matches played.
Following is the implementation of this.

```
struct Cricketer {
   float avg;
   char name[50];
   int age;
   int matches;
};
```

Now declare an array that can store the contents of the structure which is essentially an 'Array of structures'. This can be done as shown below.

```
struct Cricketer arr[n];
```

The next step would be to take input from the user and store it in the designated member
of the strcuture.

```
int n = 20;
struct Cricketer arr[n];
for(int i = 0; i < n; i++){
   printf("Enter the data of the cricketer.");
   printf("\nAverage: ");
   scanf("%f", &arr[i].avg);
   printf("\nName: ");
   scanf("%s", &arr[i].name);
   printf("\nAge: ");
   scanf("%d", &arr[i].age);
   printf("\nNumber of Test Matches played: ");
   scanf("%d", &arr[i].matches);
   printf("\n\n\n")
}
```

Now comes the important stuff. The requirement is to sort the cricketers in assceding order by their Average runs.
We can accomplish that using a standard library function called qsort(). qsort() uses a comparator function which returns 1 if the pointer 1 comes after pointer 2. In the other case it returns -1.
The implementation of the qsort() and comparator function is shown below. Comparator function should be defined
before calling the qsort().

```
int comparator (const void * a, const void * b){
   if(*(float*)a > *(float*)b){
      return 1;
   }
   else{
      return -1;
   }
}

qsort(arr, n, sizeof(struct Cricketer), comparator);
```

Now that our array of structures has been sorted in ascending order of average runs, the final step would be to print it as shown below.

```
printf("Cricketers arranged in ascending order by their Average: \n\n");
    for(int i = 0; i < n; i++){
       printf("%s\n", arr[i].name);
}
```

VI. CONCLUSION

For a sample input data of 3 cricketers as shown in fig.5

```
Enter the data of the cricketer:
Average: 34.5

Name: Raju

Age: 34

Number of Test Matches played: 56

Enter the data of the cricketer:
Average: 56.54

Name: Virat

Age: 26

Number of Test Matches played: 78

Enter the data of the cricketer:
Average: 48.56

Name: Rahul

Age: 31

Number of Test Matches played: 45
```

Fig.5 a sample input of 3 cricketers.

The output will be displayed in the terminal as shown in Fig.6

```
Cricketers arranged in ascending order by their Average:

Raju     34.500000
Rahul    48.560001
Virat    56.540001
```

Fig.2: The output of a sample matrix

VII. REFERENCES

1. https://www.geeksforgeeks.org/structures-c/
2. https://www.geeksforgeeks.org/comparator-function-of-qsort-in-c/

Final Code

```c
#include <stdio.h>
#include<string.h>
#include<stdlib.h>

int compare (const void * a, const void * b){
  if(*(float*)a > *(float*)b){
      return 1;
  }
  else{
      return -1;
  }
}

struct Cricketer {
    float avg;
    char name[50];
    int age;
    int matches;
};


int main(){
    int n = 3;
    struct Cricketer arr[n];
    for(int i = 0; i < n; i++){
        printf("Enter the data of the cricketer: ");
        printf("\nAverage: ");
        scanf("%f", &arr[i].avg);
        printf("\nName: ");
        scanf("%s", &arr[i].name);
        printf("\nAge: ");
        scanf("%d", &arr[i].age);
        printf("\nNumber of Test Matches played: ");
        scanf("%d", &arr[i].matches);
        printf("\n");
    }
    // Arranging in asscending order by avg runs.

    qsort(arr, n, sizeof(struct Cricketer), compare);
    printf("Cricketers arranged in ascending order by their Average: \n\n");
    for(int i = 0; i < n; i++){
        printf("%s\t %f\n", arr[i].name, arr[i].avg);
    }
}
```