AIM: Write a C program to find total, average of n students using structures

Description:

In C, structures are a way to group different types of variables under a single name. A structure can hold variables of different data types, including arrays, pointers, and other structures. This is particularly useful when you need to represent a record, such as a student or an employee, where each record may contain different types of information (name, age, salary, etc.).

Defining a Structure

You can define a structure using the struct keyword.

```
struct Person {
   char name[50];
   int age;
   float salary;
};
```

Array of structures:

An array of structures in programming is a collection where each element of the array is a structure. A structure (often referred to as a struct) is a user-defined data type that groups different data types together. This is useful when you want to represent a record that holds various types of data under one name.

Program:

```
#include <stdio.h>
struct Student
 {
   char name[50];
   int m1,m2,m3;  // Assuming each student has 3 subjects
   float total,avg;
};
```

```
int main()
 {
   int n, i;
   printf("Enter the number of students: ");
   scanf("%d", &n);
   struct Student s[n];
   // Input data for each student
   for(i = 0; i < n; i++)
        {
                printf("enter name,3 subject marks of student %d\n",i+1);
                scanf("%s%d%d%d",s[i].name,&s[i].m1,&s[i].m2,&s[i].m3);
        }
   for(i=0;i<n;i++)
   {
        s[i].total=s[i].m1+s[i].m2+s[i].m3;
        s[i].avg=s[i].total/3.0;
        printf("Total and Average marks of Student
%d=%.2f,%.2f\n",i+1,s[i].total,s[i].avg);
        }
   return 0;
}
```

OUTPUT:

```
Enter the number of students: 2
enter name,3 subject marks of student 1
Srinu 56 78 57
enter name,3 subject marks of student 2
```

```
Vamc 99 98 67
Total and Average marks of Student 1=191.00,63.67
Total and Average marks of Student 2=264.00,88.00
```

AIM: Write a C Program copy one structure variable to another structure of the same type

DESCRIPTION:

A **structure assignment** typically refers to the operation of assigning values between structures or copying the contents of one structure to another. This can be done directly, using the assignment operator (=), if the structures are of the same type. In a structure assignment, you transfer the values of all fields from one structure to another.

PROGRAM:

```c
#include<stdio.h>
struct student
{
	char name[15];
	int rno;
	char gender;
};
int main()
{
	struct student s1={"Srinu",143,'m'},s2;
	s2=s1;
	printf("The Student Details are:\n");
	printf("Name:%s\nRno=%d\nGender=%c",s2.name,s2.rno,s2.gender);
return 0;
}
```

OUTPUT:

```
The Student Details are:

Name:Srinu

Rno=143

Gender=m
```

AIM: Write a C Program read student name marks from the command line and display the student details along with total

DESCRIPTION:

Command Line Arguments in C allow users to pass information to a program when executing it from the command line or terminal. These arguments can be accessed inside the main() function via the parameters argc (argument count) and argv (argument vector).

PROGRAM:

```
#include<stdio.h>

#include<string.h>

struct student

{

        char name[10];

        int s1,s2,s3,s4,s5;

};

int main(int argc,char *argv[])

{

        struct student s;

        int total;

        strcpy(s.name,argv[1]);

        s.s1=atoi(argv[2]);
        s.s2=atoi(argv[3]);
        s.s3=atoi(argv[4]);
        s.s4=atoi(argv[5]);
        s.s5=atoi(argv[6]);
        total=s.s1+s.s2+s.s3+s.s4+s.s5;
        printf("name\ts1\ts2\ts3\ts4\ts5\total\n");
        printf("%s\t%d\t%d\t%d\t%d\t%d\t%d",s.name,s.s1,s.s2,s.s3,s.s4,s.s5,tota
l);
        return 0;

}
```

OUTPUT:

```
C:\Users\Srinu Sivala>arg.exe srinu 56 67 89 90 87
name    s1      s2      s3      s4      s5      total
srinu   56      67      89      90      87      389
```