**Example – C Structure Declaration & Initialization**

In the following C example program, a student structure is declared, a variable of type student structure is created, values are read from console and assigned to the members of the structure and finally those member values are printed to the console.

**C Program**

|  |
| --- |
| #include<stdio.h>  main()  {      struct student  {          int rollno;          char name[20];          float fee;          char dob[30];      }stud;       printf("\n Enter roll number:");      scanf("%d",&stud.rollno);       printf("\n Enter name:");      scanf("%s",stud.name);       printf("\n Enter fee:");      scanf("%f",&stud.fee);       printf("\n Enter DOB:");      scanf("%s",stud.dob);        printf("\n \*\*\*\*\*\*\*DETAILS\*\*\*\*\*\*\*\*\*\*");      printf("\n Rollno=%d",stud.rollno);      printf("\n Name=%s",stud.name);      printf("\n Fee=%f",stud.fee);      printf("\n DOB=%s",stud.dob);  } |
|  |

When the above program is compiles and run, following would be the output.

**Output**

|  |
| --- |
| Enter roll number:01  Enter name:peter  Enter fee:10000  Enter DOB:31-08-1990    \*\*\*\*\*\*\*DETAILS\*\*\*\*\*\*\*\*\*\*  Rollno=1  Name=peter  Fee=10000.000000  DOB=31-08-1990 |

## C Source Code: Read Records of n Students & Sort Marks in Ascending Order

#**include**<stdio.h>

**struct** student

{

**char** name[30];

**int** roll;

**float** marks;

};

main()

{

**struct** student s[20], temp;

**int** i,j,n;

printf("Enter n:\n");

scanf("%d",&n);

**for**(i=0;i< n;i++)

{

printf("Enter name, roll and marks of student:\n");

scanf("%s%d%f",s[i].name, &s[i].roll, &s[i].marks);

}

**for**(i=0;i< n-1;i++)

{

**for**(j=i+1;j< n;j++)

{

**if**(s[i].marks>s[j].marks)

{

temp = s[i];

s[i] = s[j];

s[j] = temp;

}

}

}

printf("Sorted records are:\n");

**for**(i=0;i< n;i++)

{

printf("Name: %s\n", s[i].name);

printf("Roll: %d\n", s[i].roll);

printf("Marks: %0.2f\n\n", s[i].marks);

}

}

**Pointer to a structure**

If the structure is quite large, it is more efficient to create a pointer to the structure and pass it for function calls as arguments, instead of the structure itself.

In the following example, we will use pointer to a structure.

**C Program**

|  |
| --- |
| #include<stdio.h>    struct student  {      int rollno;      char name[20];      float fee;      char dob[30];  }stud;    main()  {      struct student \*ptr;      ptr=&stud;        printf("\n Enter roll number:");      scanf("%d",&ptr->rollno);        printf("\n Enter name:");      scanf("%s",ptr->name);        printf("\n Enter fee:");      scanf("%f",&ptr->fee);        printf("\n Enter DOB:");      scanf("%s",ptr->dob);        printf("\n\n \*\*\*\*\*\*\*DETAILS\*\*\*\*\*\*\*\*\*\*");        printf("\n Rollno=%d",ptr->rollno);      printf("\n Name=%s",ptr->name);      printf("\n Fee=%f",ptr->fee);      printf("\n DOB=%s",ptr->dob);    } |

Compile and run the above C program. Following would be the output.

**Output**

|  |
| --- |
| Enter roll number:02  Enter name:john  Enter fee:20000  Enter DOB:12-03-1990    \*\*\*\*\*\*\*DETAILS\*\*\*\*\*\*\*\*\*\*  Rollno=2  Name=john  Fee=20000.000000  DOB=12-03-1990 |

**Example – C Program demonstrating passing of a structure pointer to a function as argument**

The given program passes a pointer for a structure to a function .

* In this,the address of the structure variable is passed as an actual argument to a function.
* That argument must be a structure type pointer variable .

**Note** : Any changes made to a the members in the called function are directly reflected in the calling function. Passing structures directly to a function could use up a lot of stack space if they are large. However simply using pointers would use up the memory, which can be freed easily.

**C Program**

|  |
| --- |
| #include<stdio.h>  #include<malloc.h>    typedef struct student{      int rollno;      char name[20];      float fee;      char dob[30];  };    void display(struct student \*);    int main() {      struct student \*ptr;        ptr=(struct student \*)malloc(sizeof(struct student));        printf("\n Enter roll number:");      scanf("%d",&ptr->rollno);        printf("\n Enter name:");      scanf("%s",ptr->name);        printf("\n Enter fee:");      scanf("%f",&ptr->fee);        printf("\n Enter DOB:");      scanf("%s",ptr->dob);        display(ptr);        return 0;  }    void display(struct student \*ptr) {      printf("\n \*\*\*\*\*\*\*DETAILS\*\*\*\*\*\*\*\*\*\*");      printf("\n Rollno=%d",ptr->rollno);      printf("\n Name=%s",ptr->name);      printf("\n Fee=%f",ptr->fee);      printf("\n DOB=%s",ptr->dob);  } |

**Output**

|  |
| --- |
| Enter roll number:05  Enter name:ram  Enter fee:45000  Enter DOB:12-10-1980    \*\*\*\*\*\*\*DETAILS\*\*\*\*\*\*\*\*\*\*  Rollno=5  Name=ram  Fee=45000.000000  DOB=12-10-1980 |

**C Nested Structure**

A structure can be placed within another structure i.e.,a structure may contain another structure as its member.A structure that contains another structure as its member is called a nested structure.

Following is an example C program to illustrate C Nested Structures.