F.Y.B.C.A. (SCIENCE) SEM-II
ADVANCED C PROGRAMMING

Unit 4

Structures

Mrs. Madhuri Abhijit Darekar
Assistant Professor
MIT Arts, Commerce and Science College, Alandi (D)

STRUCTURE

- In C language arrays require that all its elements should be of the same data type. But many times it is necessary to group information of elements of different data types. An example is a student information. It includes Roll Number, Name of Student, Percentage.
- C language support data structures which can store group of **elements of different data type such as character, integer, float, string, array, pointer, structure** etc. called as '**structure**'.
- A Structure is a collection or group of related data items, possibly of different types. **It is heterogeneous which can contain data of different data types.** In contrast, array is homogeneous since it can contain only data of the same type.
- A structure is a composition of variables grouped together under a single name. Each variable within the structure is

- **Syntax of a structure:**

```
struct <struct_name>
{
        <data_type>
        <data_member1>;
        <data_type>
        <data_member2>;
        ...
};
```

- **Declaration of a structure:**

```
struct StudentInfo
{
        int Rno;
        char Sname[20];
        float Percentage;
};
```

- Here **struct** is a keyword,
- StudentInfo is a **name of structure** which is group of elements of different data types like int, string and float.
- Rno, Sname and Percentage are **members of structure** StudentInfo enclosed in { }, structure declaration ends with semi-colon (;)
- **Declaration of structure does not reserve any space in memory.**

- **Creation of a structure variable:**
- **There are two different ways to create structure variable:**

1) struct StudentInfo
{ int Rno;
char Sname[20];
float Percentage;
}s1, s2;

2) struct StudentInfo
{ int Rno;
char Sname[20];
float Percentage;
};
struct StudentInfo s1,s2;

Here s1 and s2 are **structure variables** of structure StudentInfo.

- **Creation of a structure variable reserves memory space.**

- **Calculation of size of a structure:**
  - Size of structure is the addition of individual sizes of each data member of that structure.
  - We can calculate size of structure by using sizeof operator.
  - sizeof is an operator which is used to calculate size of any data type or variable.
  - So here size of structure variable s1 can be calculated as:

sizeof(s1)=sizeof(Rno)+sizeof(Sname)+sizeof(Percentage)

sizeof(s1)= 2 + 20 + 4

sizeof(s1)=26 bytes

(if int occupies 2 bytes)

Initialization of a structure variable:

- **Assigning default values to a structure variable** at the time of its declaration is called as initialization of structure variable. **Values get assigned to the structure data members from first to last** so it should match the data type of the data members and it should follow order. **Structure data members get initialized using curly brackets '{ }'.**
- If we initialize less number of members remaining members will get initialized to 0 and **if we initialize more number of members then compiler will give an error.**

There are two different ways to initialize structure variable

1) struct StudentInfo
{ int Rno;
char Sname[20];
float Percentage;
}s1={1,"xyz",80.25};

2) struct StudentInfo s1={1,"xyz",80.25};

Accessing structure members:

There are two different ways to access structure members

1) using dot (.) operator 2) using pointer to structure

Accessing structure members using dot operator

In this dot(.) operator is used between structure variable and structure member name.

Example:

```
struct StudentInfo s1;
s1.Rno;
```

Here ‘s1’ is variable and ‘Rno’ is member of structure StudentInfo.

Accessing structure members using pointer

In this -> operator is used between structure pointer variable and member name.

Example:

```
struct StudentInfo *ptr, s1;
ptr=&s1;
ptr->Rno;
```

Here ‘*ptr’ is a pointer to structure StudentInfo which stores address of structure variable ‘s1’. So now through ‘ptr’ we can access ‘Rno’.

- **Copying structure variable:**
- There are two different ways to copy one structure variable into another.

1) Copying each structure member individually

```
struct StudentInfo s1={1,"xyz",80.25};
struct StudentInfo s2;
 s2.Rno=s1.Rno;
 strcpy(s2.Sname,s1.Sname);
 s2.Percentage=s1.Percentage;
```

2) Copying entire structure variable using assignment operator

```
struct StudentInfo s1={1,"xyz",80.25};
struct StudentInfo s2;
s2=s1;
```

Here all members of s1 will get copied into members of s2.

C Program using structure:

```
/* C program to accept Student Information (Rollno, Name, Percentage) and display
   same information using structure. */
#include<stdio.h>
struct StudentInfo
{    int Rno;
   char Sname[20];
   float Percentage;
}s1;
 void main()
{    printf("\n Enter Student Information:");
   printf("\n Student Roll number, Name, Percentage
   ");  scanf("%d",&s1.Rno);
   scanf("%s",s1.Sname);
   scanf("%f",&s1.Percentage);
   printf("\n Roll Number      : %d",s1.Rno);
   printf("\n Student Name     : %s",s1.Sname);
   printf("\n Percentage       : %.2f",s1.Percentage);
}
```

Structure containing an Array

A structure can contain an array as its member. In the StudentInfo structure, the structure contains a string which is an array of characters. If we wish to store the marks of 4 subjects of a student, the declaration will be:

```
struct StudentInfo
{       int Rno;
        char Sname[20];
        int marks[4];
        float avg;
}s;
```

The members will be

s.Rno, s.Sname, s.marks[0]…s.marks[3], s.avg

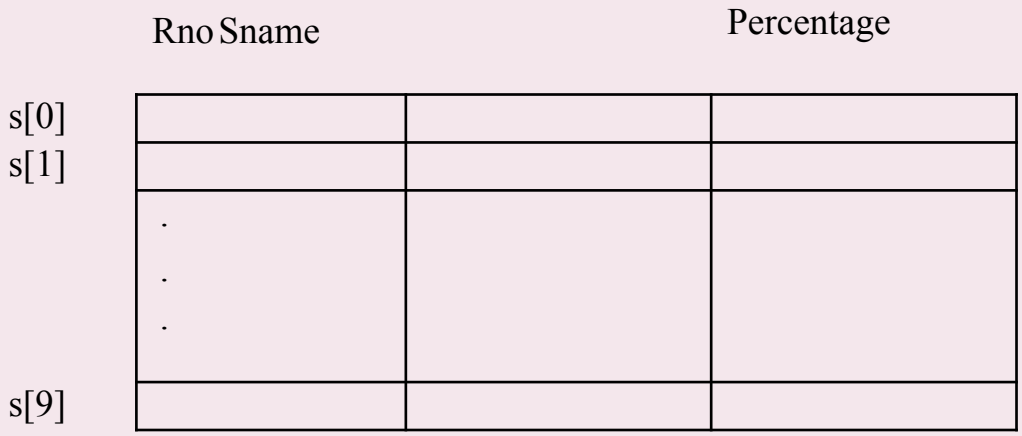
Array of Structure:

- **If we want to store information of many students we need to use array of structure.** To store large number of similar records C allows us to create an array of structure variables.
- Using array of structure **we can easily and efficiently handle large number of records.**
- All array elements of structure occupy consecutive memory locations.
- **Example:**

```
struct StudentInfo
{       int Rno;
        char Sname[20];
        int marks[4];
        float avg;
};
struct StudentInfo s[10]; // Here 's' is an array of structure which can
                                                store 10 students record.
```

- We can also have array within structure, in above example 'marks' is an array within structure StudentInfo. Such members can be accessed by using appropriate subscripts.

Example: struct StudentInfo s[10];



Accessing elements of the array:

Individual elements can be accessed as

s[0].Rno, s[0].Sname, s[0].Percentage

Initializing array of structures:

An **array of structures contains multiple structure variables** and can be initialized as shown:

```
struct StudentInfo s[4]=
{1, ”ABC”, 89,
2, ”DEF”, 64,
3, ”GHI”, 75,
4, ”JKL”, 90,
}
```

s[0] to s[3] are stored sequentially in memory.

C Program using array of structure :

```
/* C Program to accept ‘n’ students information like Rollno,
   Name, Marks of 4 subjects . Calculate the total and average
   of marks using structure*/
#include<stdio.h>
struct StudentInfo
{ int Rno;
   char Sname[20];
   int marks[4];
   float avg;
}s[10];
void main()
{ int i, j, n, total;
   printf("\n How many students info you want to enter? ");
   scanf("%d",&n);
   printf("\n Enter student information RNo, Name, Marks of 4
   subjects.");
```

```
for(i=0;i<n;i++)
   {
         total=0;
         scanf("%d%s",&s[i].Rno, s[i].Sname);
         for(j=0;j<4;j++)
         {          scanf("%d", &s[i].marks[j]);
                    total=total + s[i].marks[j];
         }
         s[i].avg=(float)total/4;
   }
   printf("\nThe student details are:");
   for(i=0;i<n;i++)
   {
         printf("\nRollno    %d is   %s and has average
         marks
         .2f",s[i].Rno,s[i].Sname,s[i].avg);
   }
}
```

Pointers and Structures :

We can use pointer with structure in many ways some of them

1) a. **Pointer to a structure:**

The address of structure variable can be stored in pointer variable. Such pointer is called as pointer to a structure. But in this case pointer must be declared as a pointer to structure.

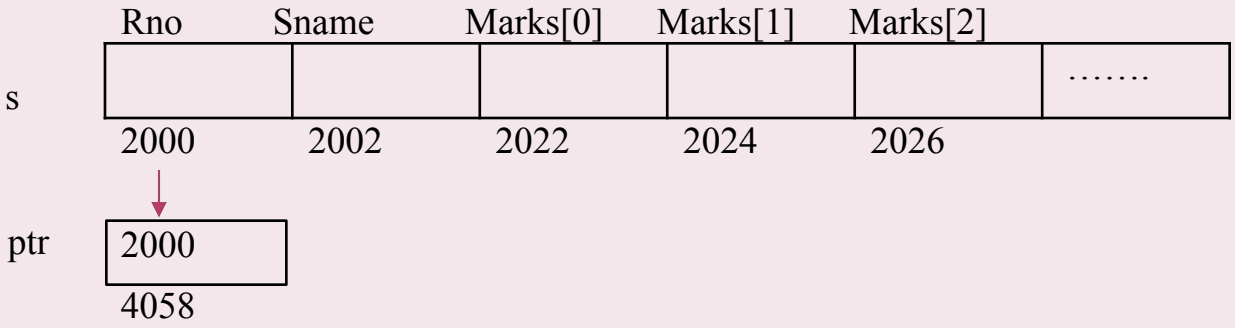
Example:

struct StudentInfo *ptr;

This can be used to store the address of a structure variable.

struct StudentInfo s;

ptr=&s;



Accessing members using pointer

To access the members of a structure using pointer, we use the -> operator. Example:

ptr->Rno, ptr->Sname, ptr->marks[i], ptr->percentage.

```
#include<stdio.h>
struct StudentInfo
{
        char Sname[20];
        int Rno;
        int marks[4];
        float percentage;
};
void main()
{
        struct StudentInfo s, *ptr;
        int i, sum=0;
        ptr = &s;    //initialize pointer
        printf (“Enter the student name :”);
        gets(ptr->Sname);
        printf (“Enter the student roll number :”);
        scanf(“%d”, &ptr-> Rno);
```

```
printf (“Enter the marks of 4 subjects :”);
for (i=0; i<4; i++)
{
        scanf(“%d”, &ptr->marks [i]);
        sum = sum + ptr-> marks[i];
}
ptr->percentage = sum/4.0;
printf(“\n - - - - - - - Student details - - - - - - -\n”);
printf(“\n Name = %s”, ptr -> Sname);
printf(“\n Roll Number     = %d\n”, ptr -> Rno);
printf(“\n Marks =”);
for (i=0; i<4; i++)
printf(“%d\t , ptr -> marks[i]);
printf(“\n Percentage = %f”, ptr->percentage);
}
```

2) **Pointer within a structure:**

We can have pointer as a member of structure. Such a pointer can be used like
any other pointer variable.

Example:

```
struct StudentInfo
{       int Rno;
        char *Sname;
        int no_subjects;
        int *marks;
        float percentage;
};s;
```

Here, s has two pointers as members. They can be used as shown below:

```
s.Sname="ABC";
s.marks=(int *)malloc(s.no_subjects * sizeof(int));
for(i=0;i<s.no_subjects;i++)
{       scanf("%d",&s.marks[i]);
        sum=sum+s.marks[i];
}
s.percentage=(float) sum/s.no_subjects;
```

3) Pointer to an array of structures:

A pointer can point to an array of structures by assigning it the base address of the array. The array elements can be accessed sequentially by incrementing the pointer. Incrementing the pointer makes the pointer makes the pointer to automatically point to the next structure variable.

```
struct student
{           char name[10];
            int rollno;
}
stud[4]={
"ABC",1,
            "DEF",2,
            "GHI",3,
            "JKL",4,
           };
main()      struct student *ptr=stud;
{           for(i=0;i<4;i++)
            {           printf("At address %u: %s %d", ptr, ptr->name,
                        ptr->rollno);  ptr++;       }
}
```

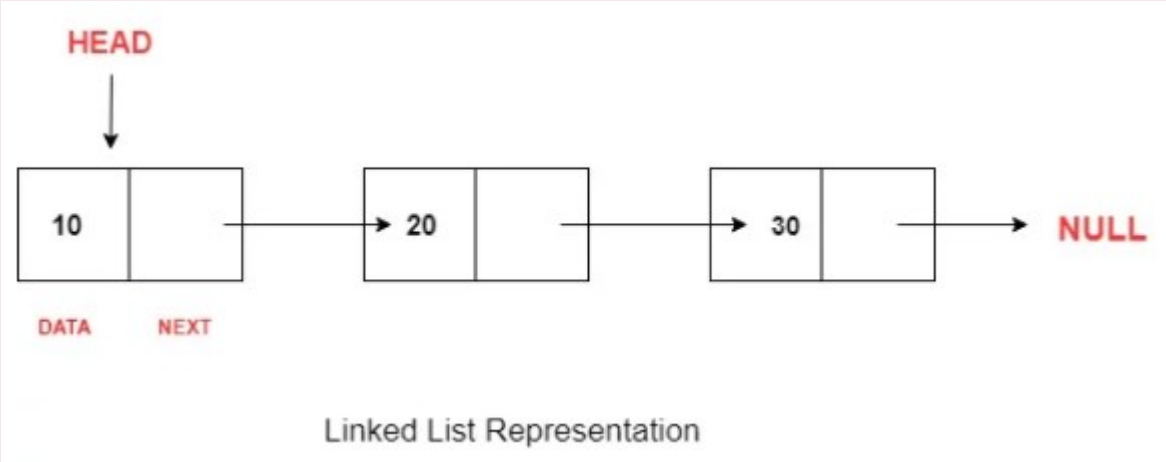
4) Structure having pointer to itself (self-referential structure)

In this type of structure, the structure contains a pointer to itself. It is called a self-referential structure.

Example:

```
struct node
{
        int data;
        struct node *ptr;
}
```

In this example, the structure node contains pointer to itself. We can see its use in a linked list. A linked list is a collection of nodes each linked to the next using pointer.



Linked List Representation

- **Structure and function :**
- We can pass structure variable to a function in following different ways as per the need:

1) **Passing structure members to function**

```
struct employee {
        int emp_id;
        char emp_name[10];
        char desg[10];
};e;
display(e.emp_id); // call to display( ) function
```

2) **Passing structure variable to function by value**

```
display(e);
```

3) **Passing structure variable to function by address**

```
display(&e);
```

- If we want to pass one of the member of a structure to a function we can pass only required member instead of passing entire structure variable.
- Instead of passing members of structure individually we can pass entire structure variable to a function if needed which will pass all the members of a structure at a time.
- If we want to modify the contents of structure members through function we need to pass structure variable to a function by address and not by value.

Passing structure variable to a function by value:

```
#include<stdio.h>
struct student
{       char name[20];
        int rollno;
        float percentage;
};
void main()
{       struct student s={“ABC”,1,78.5};
        void display(struct student s);
        display(s);
}
void display(struct student s)
{
        printf(“\n----- Student details-------\n”);
        printf(“\n Name: %s”, s.name);
        printf(“\n Roll Number: %d”, s.rollno);
        printf(“\n Percentage: %f”, s.percentage);
}
```

```
/* C program to create structure employee having fields emp_id, emp_name,
   designation. Pass this entire structure to function and display the structure
   elements using pointers. */

#include<stdio.h>
struct employee
{    int emp_id;
   char emp_name[20];
   char desg[20];
}e;
void display(struct employee *);
void main()
{
   printf("\nEnter employee details :");
   printf("\nEnter employee id, name and designation : ");
   scanf("%d", &e.emp_id);
   scanf("%s", e.emp_name);
   scanf("%s", e.desg);
   display(&e);
}
```

```
void display(struct employee *ptr)
{
        printf("\n Employee id : ");
        printf("%d", ptr->emp_id);
        printf("\n Employee name : ");
        printf("%s", ptr->emp_name);
        printf("\n Employee designation : ");
        printf("%s", ptr->desg);
}
```

- In above program when we pass address of structure variable ‘e’ to display ( ) function it will get copied into a structure pointer ‘ptr’.
- And in display( ) function now we can display structure members through ‘ptr’ pointer (pointer->member).

Nested Structure:

- If a structure contains another structure within it, it is called as nested structure.
- For nested structure one structure need to be declared inside another structure, now this inner structure becomes a member of outer structure.
- **Example:**

```
struct Employee
{       int emp_id;
        char emp_name[20];
        float Salary;

        struct Address
        {       char HouseNo[20];
                char City[20];
                int PinCode;

        } Addr;
}Emp;
```

- Here we have declared two structures named ‘Address’ and ‘Employee’, where structure ‘Address’ is nested within ‘Employee’ structure.
- ‘Employee’ is a outside structure and ‘Address’ is inner structure.

Accessing members of nested structures:

- To access members of **outer structure** we need to use dot operator once.

```
Emp.emp_id
Emp.emp_name
Emp.Salary
```

- Now to access member of **inner structure** we need to use dot operator two times.

```
Emp.Addr.HouseNo;
Emp.Addr.City;
Emp.Addr.PinCode
;
```

- The same nesting can be done in another way also:

```
struct Address
{       char HouseNo[20];
        char City[20];
        int PinCode;

};
struct Employee
{       int emp_id;
        char emp_name[20];
        struct          Address
        Addr;  float Salary;

}Emp;
```

- **Typedef with structure:**
- The typedef is used in combination with struct to declare a synonym (or an alias) for a structure, means it is a keyword which is used to give new name to the structure.
- This new name we can use as a data type like other built-in data type while declaring structure variable.
- If we rename structure using typedef then there is no need to use keyword struct while declaring structure variable.
- **Example:**

```
typedef struct employee
{ int emp_id;
      char emp_name[20];
      char desg[20];
}EMPLOYEE;
EMPLOYEE e1,e2;          //declaration of structure variables
                         without using keyword struct.
```

- Here the structure name ‘employee’ is renamed to ‘EMPLOYEE’.

Passing Array of Structure as a parameter to function :

```
/* C Program to accept customer details such as custid , name , city , phone ,
   email . Read the details of n number of customers write a function that will
   display the details of particular customer. Pass custid as a parameter to this
   function . */

#include<stdio.h>
#include<string.h>
struct customer
{
        int custid;
        char name[20];
        char city[10];
}c[10];

void display(int cid, struct customer c[10], int n);
void main()
{
        int i,n,cid;
        printf("\n Enter how many details of customers u want to enter : ");
        scanf("%d",&n);
```

```
printf("\n Enter customer details : ");

for(i=0;i<n;i++)
{
        printf("\n Enter customer ID : ");
        scanf("%d",&c[i].custid);
        printf("\n Enter name : ");
        scanf("%s",c[i].name);
        printf("\n Enter city : ");
        scanf("%s",c[i].city);
}

   printf("\n Enter customer ID whose record u want to display);
   scanf("%d",&cid);

   display(cid,c,n);
}
```

```
void display(int cid, struct customer c[10], int n)
{
    int i, f=0;

    for(i=0; i<n; i++)
    {
            if(c[i].custid==cid)
            {
                    printf("\n Customer ID : %d ",c[i].custid);
                    printf("\n Name : ");
                    puts(c[i].name);
                    printf("\n City : ");
                    puts(c[i].city);
                    f=1;
            }
    }

    if(f==0)
            printf("\n Record is not present");
}
```