Structures

- It is a heterogeneous (different type of data) collection of elements.
- Means collection of members of different type of data.
- All the members are stored in contiguous memory location.

How to define a structure (Structure Definition)

```
Struct <structname>
{
member 1;
member 2;
.....
member n;
};
```

Eg.

```
Struct student
{
int rollno;
Char name[10];
float marks;
};
```

How to declare a structure

```
Struct student
{
int rollno;
Char name[10];
float marks;
};
Struct student S1, S2, S3;
```

OR

Definition + declaration

```
Struct student
{
int rollno;
Char name[10];
float marks;
} S1, S2, S3;
```

⇓

This means structure of student contains RollNo, Name & Marks. and There are 3 structure variables (S1, S2, S3) Means 3 records / 3 students.

eg

	Roll No	Name	Marks
S1			
S2			
S3			

How to access the values of a structure or how to accessing the members.

Using . (dot) operator.

structure variable . structure member

eg

1st record

```
S1.rollno = 1 ;
S1.name = "Amit" ;   // ⇒ strcpy(S1.name, "Amit");
S1.marks = 80.5 ;
```

2nd record

```
S2.rollno = 2 ;
strcpy( S2.name, "Aman");
S2.marks = 91.5 ;
```

3rd record

```
S3.rollno = 3 ;
strcpy( S3.name, "Ram");
S3.marks = 75.5 ;
```

How to input members of structure during run time

(Using scanf)

```
scanf("%d %s %f", &S1.rollno, &S1.name, &S1.marks);
scanf("%d %s %f", &S2.rollno, &S2.name, &S2.marks);
scanf("%d %s %f", &S3.rollno, &S3.name, &S3.marks);
```

Copy one structure into another structure.

⟹ Using assignment operator (=)

Whole record can be copied into another record.

```
S1 = S2;
```

Now S2 record will be copied into S1 record and S1 record will be overwritten.

But in array, we can only copy one array into another array element by element (one by one) like this: using loop

```
for (i=0; i<10; i++)
{
    b[i] = a[i];
}
```

If a and b are 2 arrays

```
b = a;
```

Imp!!

(4)

Difference between Array and structure

Array	Structure
1. It is homogeneous collection of elements.	1. It is heterogeneous collection of elements
2 Elements are stored in contiguous memory location.	2 All members are stored in contiguous memory location.
3. Arrays are declared and defined at same place int a[10];	3 Structure definition and structure declaration can be seperate.
4 No keyword is used to declare an array	4 struct keyword is used
5 Array can be copied to another array one by one.	5 Structure can be copied into another structure using = operator S1 = S2;
6. Array elements can be accessed using [ ]	6. Structure elements can be accessed using . operator

Eg

```
#include <stdio.h>
#include <conio.h>
void main()
{
  Struct student
  {
    int r;
    char n[10];
    float m;
  } s1, s2, s3;
  s1.r=1;
  strcpy(s1.name, "Amit");
  s1.m=80.5;
  scanf("%d%s%f", &s2.r, &s2.n, &s2.m);
  s3=s2;
  printf("%d%s%f\n", s1.r, s1.n, s1.m);
  printf("%d%s%f\n", s2.r, s2.n, s2.m);
  printf("%d%s%f\n", s3.r, s3.n, s3.m);
  getch();
}
```

I/P

2 Aman 90.5

O/P

1. Amit 80.5] already given
2. Aman 90.5] Input from user
2. Aman 90.5] Copied from s2.

Array of structure

If we want to store multiple records say 10 or 100... we will use array of structure.

```
Struct student
  {
   int r;
   char n[10];
   float m;
  } S[10];
```

There are 10 structures or 10 records.

S[0], S[1], S[2], S[3], ... S[9]

Program to enter record. (Enter 10 records).

R	N	M1	M2	M3	Total	Avg

```
#include <stdio.h>
#include <conio.h>
void main()
  {
   struct student
     {
      int r;
      char n[10];
      float m;
     } S[10];
```

```
for (i=0; i<10; i++)
 {
printf ("Enter rollno");
scanf ("%d", s[i].x);
printf ("Enter name");
gets (s[i].n);
printf ("Enter 3 marks");
scanf ("%f%f%f", &s[i].m1, &s[i].m2, &s[i].m3);
 s[i].total= s[i].m1+ s[i].m2+ s[i].m3;
 s[i].avg= s[i].total /3;
```

Inputting 10 records

```
}
for(i=0; i<10; i++)
 {
printf("%d\t%s\t%f\t%f\n",
                s[i].x, s[i].n, s[i].total, s[i].avg);
 }
getch();
}
```

O/P

1	ABC	Total	Avg
2	XYZ	Total	Avg
}	}	|	|
10	—	—	—

Nesting of structure

(Structure within structure).

Eg

```
Struct student
{
int x;
char n[10];
Struct address a;
} S1, S2, S3;
```

```
Struct employee
{
int empcode;
char empname[10];
struct address a;
} E1, E2, E3;
```

First of all, Address field is using in both structures (Student and Employees). So we will define address structure first which itself contains multiple fields (HNO, Street, City, State, Pincode, Mobile .....). So all these fields don't have to be repeated in both the structures. Define address structure once and include it in both (Student & Employees) structure.

```
Struct address
{
int hno;
char city[10];
   .
   .
   .
};
```

Members of address.

```
S1.a.hno = 101;
S1.a.city = "CHD";
```

How to access nested structure members

```
S1.x = 1
strcpy(S1.n, "Aman");
S1.a.hno = 854;
S1.a.city = "Chd";

S2.x = 2
strcpy(S2.n, "Amit");
S2.a.hno = 763;
S2.a.city = "Delhi";

111y  S3
```

```
E1.empcode = 1001
strcpy(E1.ename, "ABC");
E1.a.hno = 470;
E1.a.city = "Pune";

E2.empcode = 1002
strcpy(E2.ename, "XYZ");
E2.a.hno = 840;
E2.a.city = "Chd";

111y  E3
```

Passing structure to a function.

There are 2 ways to pass a structure to a function.

1. By element or By member one by one. ie Individual element
2. Whole structure can be passed to a function.

Eg

(1) Individual element

```
void display (int, char *, float);
main()
{
  struct student
  {
  int r;
  char n[10];
  float m;
  } S1;
printf("Enter rollno, Name and Marks");
scanf("%d %s %f", &S1.r, &S1.n, &S1.m);
display( S1.r, S1.n, S1.m);
getch();
}
```

⇒ S1 record has passed to display function by individual element (R, N, M).

```
void display( int r, char *n, float m)
{
printf("%d %s %f", r, n, m);
}
```

displaying S1 record by copying record into r, n and m

(2) Whole Structure

eg

```
Struct student
{
int r;
char n[10];
float m;
};
void display(struct student);
#include <stdio.h>
void main()
{ struct student S1;
printf("Enter rollno, name and marks");
scanf("%d %s %f", &S1.r, &S1.n, &S1.m);
display (S1);
getch();
}
void display(struct student S2)
{
printf("%d %s %f", S2.r, S2.n, S2.m);
}
```

Explanation

⇒ Here, structure has been defined outside main() function because whole structure has been passed to a function. So display() function should also know the structure of student..

⇒ If it is defined inside main() function, then this structure will not be visible to display function because of local scope.

⇒ If student structure (s1) will be passed to display() function, then parameter of display() function should also be struct student.

Imp Pointer to structure

Eg
```
void main()
{
    struct student
    {
        int r;
        char n[10];
        float m;
    };
    struct student s1;
    struct student *p;
    p = &s1;
    printf("Enter rollno, name & marks");
    scanf("%d %s %f", &s1.r, &s1.n, &s1.m);
```

Next page

```
printf("%d %s %f", s1.r, s1.n, s1.m);
printf("%d %s %f", p->r, p->n, p->m);
getch();
}
```

Explain - Pointer to structure operator is → (Arrow) or Dereferencing operator.

Pointer variable → structure member

P → R
P → N
P → M

Without Pointer

s1.R
s1.N
s1.M