- **STRUCTURES**
-

A structure in C is a collection of heterogeneous type of data referred under a single name. In other word a structure is a derive data type to organize a group of related data item of different data type. A structure can have array or pointer as some of its members, though these can get complicated unless one is careful.

Here is an example of structure definition

```
struct student
{

char name[64];
char course[128];
int age;
int year;
float height;
float weight;
}
Struct Student s;
```

Program

```
#include<stdio.h>
int main()
{
struct student
{
int id;
char name[20];
char address[20];
}s[50];

int n,i;
printf("\n Enter how many records you want to insert");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\n %d.Enter the ID---",i);
scanf("%d",&s[i].id);
printf("\n %d.Enter the Name---",i);
scanf("%s",s[i].name);
printf("\n %d.Enter the Address---",i);
scanf("%s",s[i].address);
}
printf("\n\n          The Student Record\n\n" );
printf("\tID\tName\tAddress\n\n");
```

```
for(i=1;i<=n;i++)
{
printf("\n\t%d\t%s\t%s",s[i].id,s[i].name,s[i].address);
}
return 0;
}
```

Comparison of Array and Structure

Array	Structure
An array is a collection of data items of similar data type.	A structure is a collection of data items of different data types.
It has declaration only	It has declaration and definition
There is no keyword	The struct keyword is used
An array name represents the address of the starting element	A structure name is known as tag. It is a short hand notation of the declaration
An array cannot have bit field	A structure may contain bit fields

- **Array of Structure**

Array of structure is used to create too many objects. To declare array of structures, we must first define a structure and then declare an array variable of that type. Each element of the array of structure variable will contain the structure of its type.

Example:

```
struct student
{
char name[20];
int roll;
char gender;
float height;
float weight;
};
struct student s[120];
```

- **Structure of Structures**

Structure within a structure means nesting of structures, i.e. the individual members of a structure can be other structures as well.

Example:

```
struct date
{
int day;
int month;
int year;
};
struct student
{
```

```
int roll;
char name[20];
char address[20];
struct date dob;
};
struct student s[50];
```

- **Pointer to a Structure**

A pointer to a structure is similar to an ordinary variable. It is created in the same way as a pointer to an ordinary variable is created.

Example:

```
struct student
        {
                int roll;
                char name[20];
        };
        struct student *s1;
```

The individual members can access by the following way

```
(*s1).roll
(*s1).name
Or
s1->roll
s1->name
```

Create a Student database

```
#include<stdio.h>
#include<conio.h>
void main()
{
struct student
{
int id;
char name[20];
char address[20];
}s[50];

int n,i;
printf("\n Enter how many records you want to insert");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\n %d.Enter the ID---",i);
scanf("%d",&s[i].id);
printf("\n %d.Enter the Name---",i);
scanf("%s",s[i].name);
```

```
printf("\n %d.Enter the Address---",i);
scanf("%s",s[i].address);
}
printf("\n\n          The Student Record\n\n" );
printf("\tID\tName\tAddress\n\n");
for(i=1;i<=n;i++)
{
printf("\n\t%d\t%s\t%s",s[i].id,s[i].name,s[i].address);
}
getch();
}
```

Output

```
 Enter how many records you want to insert3
 1.Enter the ID---1
 1.Enter the Name---Avik
 1.Enter the Address---Burdwan
 2.Enter the ID---2
 2.Enter the Name---Arpan
 2.Enter the Address---Kolkata
 3.Enter the ID---3
 3.Enter the Name---Souvik
 3.Enter the Address---Delhi
        The Student Record
       ID      Name     Address
       1       Avik     Burdwan
       2       Arpan    Kolkata
       3       Souvik   Delhi3
```

Example of Pointer to a Structure

```
#include<stdio.h>
#include<conio.h>
void main()
{
int ptr;
struct student
{
int roll;
char name[20];
char address[20];
char trade[10];
}*s1;
ptr=&s1;
clrscr();
printf("\n Enter the Roll Number,Name,Address,Trade---");
scanf("%d%s%s%s",&s1->roll,s1->name,s1->address,s1->trade);
printf("\n%d    %s       %s       %s",s1->roll,s1->name,s1->address,s1->trade);
getch();
```

}

Output

Enter the Roll Number,Name,Address,Trade---1
Antony
Delhi
CSE

1 Antony Delhi CSE

- **Unions**

A Union is a memory location that is shared by two or more different types of variables. A union provides a way of interpreting the same bit pattern in two or more different ways.
Declaration of an **Union** is similar to declaring a structure.

```
union mixed_data
{
int I;
float f;
double d;
…….
…….
};
union mixed_data md;
```

Program

Basic record of a person

```
#include<stdio.h>
#include<conio.h>
void main()
{
struct person
{
char gender;
int age;
float height;
float weight;
};
union bio
{
struct person rec;
};
union bio b;
```

```
b.rec.gender='M';
b.rec.age=18;
b.rec.height=5.9;
b.rec.weight=64.5;
printf("       Person Record\n\n" );
printf("\nSex--%c",b.rec.gender);
printf("\nHeight--%f",b.rec.height);
printf("\nHeight--%d",b.rec.age);

printf("\nWeight--%f",b.rec.weight);
getch();
}
```

Output

```
    Person Record
Sex--M
Height--5.900000
Weight--64.500000
Age=18
```

- **Comparison of Structure and Union**

Structure	Union
Every member has its own memory	All member use the same memory
Keyword **struct** is used	Keyword **union** is used
All members may be initialized	Only its first member may initialized
Different interpretations of the same memory location are not possible	Different interpretations of the same memory location are possible
Consumes more space compared to union	Conservation of memory is possible