Structure In C Programming:

array has the capability to store more than one element but they all must be of same type. A structure is a user define data type that can be used to group elements of different into a single type. You can also use Typedef concept in structure concepts. structure is a way to group variables. structure can be a collections of dissimilar elements. defining structure is creating a custom data type. Array used only if you want same types and same nature of data. you can declare structure definition in two types “GLOBAL DEFINITION”, “LOCAL DEFINITION”.

```c
#include<stdio.h>
#include<conio.h>

struct emp
{
    int empid;
    char name[50];
    float salary;
};
struct birth
{
    int date, month, year;
};
```

```c
#include<stdio.h>
#include<conio.h>

struct emp //global definition
{
    int empid;
    char name[50];
    float salary;
};
int main()
{
    struct birth    //local definition
    {
        int date, month, year;
    };
}
```

```c
#include<stdio.h>
#include<conio.h>

struct emp //global definition
{
    int empid;
    char name[50];
    float salary;
} d1, d2, d3;
```

If you declare structure variable inside a function then those variables become local variables, if you declare structure variable outside of the function then those variables become global variables, but for global declaration the structure must be defined as a global definition.

```c
#include<stdio.h>
#include<conio.h>

struct birth
{
    int date, month, year;
}d1, d2, d3;
```

d1, d2, d3 = structure variable
date, month, year = structure member variable
d1 (date, month, year) = 12 bytes
d2 (date, month, year) = 12 bytes
d3 (date, month, year) = 12 bytes

Initialize Structure Variable During Declaration & After Declaration:

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct employee
{
    int empid;
    char empname[100];
    float salary;
};
int main()
{
    struct employee emp1 = {100, "Atish Kumar Sahu", 50000.0}, emp2;

    emp2.empid = 101;
    strcpy(emp2.empname,"Lipun Kumar Sahu");
    emp2.salary = 60000.0;

    printf("Emp1 Empid : %d\n",emp1.empid);
    printf("Emp1 Empname : %s\n",emp1.empname);
    printf("Emp1 Salary : %f\n",emp1.salary);

    printf("--------------------------------\n");

    printf("Emp2 Empid : %d\n",emp2.empid);
    printf("Emp2 Empname : %s\n",emp2.empname);
    printf("Emp2 Salary : %f\n",emp2.salary);

    getch();
}
```

```
E:\C CODE\STRUCTURE.exe

Emp1 Empid : 100
Emp1 Empname : Atish Kumar Sahu
Emp1 Salary : 50000.000000
--------------------------------
Emp2 Empid : 101
Emp2 Empname : Lipun Kumar Sahu
Emp2 Salary : 60000.000000

--------------------------------
Process exited after 8.802 seconds with return value 0
Press any key to continue . . .
```

In structure member variable don't have separate identity.

Take Input From User In Structure:

STRUCTURE.c

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct employee
{
    int empid;
    char empname[100];
    float salary;
};
int main()
{
    struct employee num1;

    printf("Employee Detail : \n");
    printf("Employee Id : ");
    scanf("%d",&num1.empid);

    printf("Employee Name : ");
    scanf("%s",num1.empname);

    printf("Employee Salary : ");
    scanf("%f",&num1.salary);

    printf("----------------------\n");

    printf("EmpId : %d\n",num1.empid);
    printf("EmpName : %s\n",num1.empname);
    printf("EmpSalary : %f\n",num1.salary);

    getch();
}
```

E:\C CODE\STRUCTURE.exe

```
Employee Detail :
Employee Id : 108
Employee Name : ATISH
Employee Salary : 2050.63
------------------------
EmpId : 108
EmpName : ATISH
EmpSalary : 2050.629883

--------------------------------
Process exited after 23.44 seconds with return value 0
Press any key to continue . . .
```

Return A Structure Variable Data From A Function:

```c
#include<stdio.h>
#include<conio.h>

struct date
{
int d,m,y;
}d1;
struct date inputdate()
{
    struct date DD;
    printf("Enter date : ");
    scanf("%d/%d/%d",&DD.d,&DD.m,&DD.y);
    return DD;
}
int main()
{
d1 = inputdate();
printf("%d-%d-%d",d1.d,d1.m,d1.y);
}
```

E:\C CODE\STRUCT.exe

```
Enter date : 07/01/2023
7-1-2023
--------------------------------
Process exited after 6.952 seconds with return value 0
Press any key to continue . . .
```

Function Call By Passing Structure:

```c
#include<stdio.h>

struct date
{
int d,m,y;
}d1;
struct date inputdate()
{
    struct date DD;
    printf("Enter date : ");
    scanf("%d/%d/%d",&DD.d,&DD.m,&DD.y);
    return DD;
}
void show(struct date AA)
{
    printf("%d-%d-%d",AA.d,AA.m,AA.y);
}
int main()
{
d1 = inputdate();
printf("%d-%d-%d\n",d1.d,d1.m,d1.y);
show(d1);
}
```

```
E:\C CODE\STRUCT.exe

Enter date : 20/04/2025
20-4-2025
20-4-2025
--------------------------------
Process exited after 4.814 seconds with return value 0
Press any key to continue . . .
```

Structure & Array:

STRUCT.cpp

```c
#include<stdio.h>

struct date
{
    int d,m,y;
};
struct date inputdate()
{
    struct date DD;
    printf("Enter date : ");
    scanf("%d/%d/%d",&DD.d,&DD.m,&DD.y);
    return DD;
}
void show(struct date AA)
{
    printf("%d-%d-%d\n",AA.d,AA.m,AA.y);
}
int main()
{
    struct date dob[3];
    for(int i = 0; i < 3; i++)
    {
        dob[i] = inputdate();
    }
    for(int i = 0; i < 3; i++)
    {
        show(dob[i]);
    }
}
```

```
E:\C CODE\STRUCT.exe

Enter date : 15/02/2023
Enter date : 28/07/2021
Enter date : 12/12/2022
15-2-2023
28-7-2021
12-12-2022

--------------------------------
Process exited after 18.29 seconds with return value 0
Press any key to continue . . .
```

Structure & Pointer:

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>

struct subject
{
    char subname[50];
    int subid;
    char name[50];
};
int main()
{
    struct subject sub1;
    struct subject *ptr;
    ptr = &sub1;

    strcpy(sub1.subname,"Computer Science");
    sub1.subid = 101;
    strcpy(sub1.name,"Atish Kumar Sahu");

    printf("Subject Name : %s\n",(*ptr).subname);
    printf("Subject Id : %d\n",(*ptr).subid);
    printf("Student Name : %s\n",(*ptr).name);

    getch();
}
```

```
E:\C CODE\STRUCTURE.exe

Subject Name : Computer Science
Subject Id : 101
Student Name : Atish Kumar Sahu

--------------------------------
Process exited after 2.565 seconds with return value 0
Press any key to continue . . .
```

Union In C Programming:

Introduction:

Union is a special data type available in C programming that enables you to store different data types in the same memory location. You can define a union with many members, but only one member can contain a value at given time. Union provides as efficient way of using the same memory location for multiple purpose. Union can be defined as a user-defined data type which is a collection of different variables of different data types in the same memory location. The union can also be defined as many members, but only one member can contain a value at a particular point in time. Union is a user-defined data type, but unlike structures, they share the same memory location.

Difference Between Union & Structure:

1. The keyword struct is used to define a structure. The keyword union is used to define a union.

2. Each member within a structure is assigned unique storage area of location. Memory allocated is shared by individual members of union.

3. Individual member can be accessed at a time. Only one member can be accessed at a time.

4. Several members of a structure can initialize at once. Only the first member of a union can be initialized.

5. Altering the value of a member will not affect other members of the structure. Altering the value of any of the member will after other members values.

6. Structure allows accessing and retrieving any data member at a time. Union allows accessing and retrieving any one data member at a time.

7. Structure is used to store different data type values. Union is used to storing one at a time from different data type values.

Advantages:

1. Union takes less memory space as compared to the structure. Only the largest size data member can be directly accessed while using a union.

3. It is used when you want to use less memory for different data members.

4. It allocates memory size to all its data members to the size of its largest data member.

Disadvantages:

1. It allows access to only one data member at a time. Union allocates one single common memory space to all its data members, which are shared between all of them.

2. Not all the union data members are initialized, and they are used by interchanging values at a time.

UNIONC.c

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
union UN
{
    int num1;
    char ch1[100];
    double fl1;
}demo;
int main()
{
    printf("Size Of Union Is : %d\n",sizeof(union UN));

    demo.num1 = 1000;
    strcpy(demo.ch1,"Atish Kumar Sahu");
    demo.fl1 = 5.1234;

    printf("UN num1 : %d\n",demo.num1);
    printf("UN ch1 : %s\n",demo.ch1);
    printf("UN fl1 : %lf\n",demo.fl1);
}
```

E:\C CODE\UNIONC.exe

```
Size Of Union Is : 104
UN num1 : -1848553924
UN ch1 : <N╪æ\~¶@mar Sahu
UN fl1 : 5.123400

--------------------------------
Process exited after 0.03831 seconds with return value 0
Press any key to continue . . .
```

Union & Pointer:

UNIONC.c

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
union UN
{
    int num1;
    char ch1[100];
    double fl1;
};
int main()
{
    union UN *ptr;
    union UN demo;
    demo.num1 = 5000;
    strcpy(demo.ch1,"Atish");
    ptr = &demo;

    printf("The Value Of Num : %d\n",ptr->num1);
    printf("The Value Of Ch1 : %s\n",ptr->ch1);
}
```

E:\C CODE\UNIONC.exe

```
The Value Of Num : 1936290881
The Value Of Ch1 : Atish

--------------------------------
Process exited after 0.03146 seconds with return value 0
Press any key to continue . . .
```

Enumeration In C Programming:

Introduction:

Enum is a list of constant integer value. Enum is a keyword which is used to define enumerated type data. Enum is a user define data type that consists of integer values, and it provides meaningful names to these values. The use of enum in C makes the program easy to understand and maintain. The enum is defined by using the enum keyword.

The enum names available in an enum type can have the same value. Let's look at the example. If we do not provide any value to the enum names, then the compiler will automatically assign the default values to the enum names starting from 0. We can also provide the values to the enum name in any order, and the unassigned names will get the default value as the previous one plus one.

The values assigned to the enum names must be integral constant, i.e., it should not be of other types such string, float, etc. All the enum names must be unique in their scope, i.e., if we define two enum having same scope, then these two enums should have different enum names otherwise compiler will throw an error. In enumeration, we can define an enumerated data type without the name also.

ENUM.c

```c
#include<stdio.h>
#include<conio.h>

enum days{
    SUNDAY, MONDAY, TUESDAY,
    WEDNESDAY, THURSDAY,
    FRIDAY, SATURDAY
};
int main()
{
    for(int i = SUNDAY; i <= SATURDAY; i++)
    {
        printf("%d\n",i);
    }
    getch();
}
```

E:\C CODE\ENUM.exe

```
0
1
2
3
4
5
6

--------------------------------
Process exited after 25.01 seconds with return value 0
Press any key to continue . . .
```

ENUM.c

```c
#include<stdio.h>
#include<conio.h>

enum days{
    SUNDAY, MONDAY,
    TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY,
    SATURDAY
};
int main()
{
    enum days DAY;

    printf("Size Of Enum : %d\n",sizeof(DAY));

    getch();
}
```

E:\C CODE\ENUM.exe

```
Size Of Enum : 4

--------------------------------
Process exited after 10.41 seconds with return value 0
Press any key to continue . . .
```

```c
#include<stdio.h>
#include<conio.h>

enum days{
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
};
int main()
{
    enum days D;
    printf("Enter The Number : ");
    scanf("%d",&D);

    switch(D)
    {
        case SUNDAY:
            printf("Today Is Sunday");
            break;

        case MONDAY:
            printf("Today Is Monday");
            break;

        case TUESDAY:
            printf("Today Is Tuesday");
            break;

        case WEDNESDAY:
            printf("Today Is Wednesday");
            break;

        case THURSDAY:
            printf("Today Is Thursday");
            break;

        case FRIDAY:
            printf("Today Is Friday");
            break;

        case SATURDAY:
            printf("Today Is Saturday");
            break;

        default:
            printf("Invalid Number");
            break;
    }
    getch();
}
```

E:\C CODE\ENUM.exe

```
Enter The Number : 0
Today Is Sunday
--------------------------------
```

E:\C CODE\ENUM.exe

```
Enter The Number : 1
Today Is Monday
--------------------------------
```

E:\C CODE\ENUM.exe

```
Enter The Number : 2
Today Is Tuesday
--------------------------------
```

E:\C CODE\ENUM.exe

```
Enter The Number : 3
Today Is Wednesday
--------------------------------
```

E:\C CODE\ENUM.exe

```
Enter The Number : 4
Today Is Thursday
--------------------------------
```

E:\C CODE\ENUM.exe

```
Enter The Number : 5
Today Is Friday
--------------------------------
```

E:\C CODE\ENUM.exe

```
Enter The Number : 6
Today Is Saturday
--------------------------------
```

ENUM.c

```c
#include<stdio.h>
#include<conio.h>

typedef enum {
    male = 1,
    female = 2
}gender;
int main()
{
    gender value = male;

    printf("Value : %d\n",value);

    gender value1 = female;

    printf("Value1 : %d\n",value1);

    getch();
}
```

E:\C CODE\ENUM.exe

```
Value : 1
Value1 : 2

--------------------------------
Process exited after 1.94 seconds with return value 0
Press any key to continue . . .
```