Unit-3

Structure, Union & Enum

* Structure *

•> Introduction to Structure in C

As we know that Array is a collection of the elements of same type, but many time we have to store the elements of the different data types.

Suppose Student record is to be stored, then for storing the record we have to group together all the information such as Roll. No., Name, Perient which may be of different data types.

Structure is a collection of different variables under single name. It is used for storing the complicated data. A Structure is a convinient way of grouping several pieces of related information together.

•> Defination of Structure

Structure is a composition of different variables of different data types, grouped under same name. Each member declare in structure is called data member. Name given to structure is called a tag or structure name. Structure member may be of different data type including user define datatype.

•> Declaring Structure

In C we can group some of the user defined or primitive data types together and form another compact way of storing complicated information is called as structure.

Syntax -

```
struct <Structure_Name>
{
   Data member-1;
   Data member-2;
   - - -
   - - -
};
```

- Struct Keyword is used to declare structure.
- Member of structures (Data members) are enclosed within opening & closing braces.
- Declaration of structure reserve no space.
- Memory is created, very 1st time when the variable is created.

•> Different ways of Declaring Structure Variable

1.) Immediately after Structure Creation. Example -

```
struct date
{
    int date;
    char month[20];
    int year;
} today ;    // It is a structure variable.
```

2) Declare variables using struct keyword. Example -

```
Struct date
{
    int date;
    char month[20];
    int year;
};
Struct date today ;    // It can also used inside main.
```

Here "date" is structure name, "today" is variable name

3) Declaring multiple structure variable . Example -

```
Struct date
{ int date;
  char month[20];
  int year;
} day1, day2, day3 ;
```

We can declare multiple variables separated by comma directly after closing curly braces.

•> C Structure Initialization -

When we declare a structue, memory is not allocated for un-initialized variable. We can initialize structure variable in different ways -

1) ~~way~~ Declare and Initialize -

```
struct Student
{
    char name[20];
    int roll;
    float marks;
} std1 = {"Rahul", 67, 78.3};
```

2) Declaring & Initializing multiple Variable -

```
Struct Student
{
   char name[20];
   int roll;
   float marks;
}
std1 = {"Rahul", 67, 78.3};
std2 = {"Ajay", 62, 71};
```

3) Initializing Single member -

```
Struct Student
{
   ~~char name[20];~~
   int m1, m2, m3;
} std1 = {67};
```

There are 3 members of structure, only one is initialized. Then remaining two members are initialized with zero. If there are variables of other data type then their initial value will be -

Integer = 0 , float = 0.00 , Char = NULL.

4) Initialize Inside main() -

```
struct Student
{ int m1, m2, m3;
};
void main()
{
    struct student s1 = {89, 54, 65};
        ---
        ---
}
```

⇒ Important Notes for Declaring Structure Variable -

1) Closing braces of structure type declaration must be followed by semicolon.
2) Don't forgat to use 'struct' keyword.
3) Memory will not be allocated just by creating instance or by declaring structure. We need to initialize structure variable to allocate memory to the structure.
4) Structures are generally written in Global Declaration Section, But they can be written inside main. Example -

```
void main()
{
   struct student
   {
      char name[10];
      int roll#, marks;
   } stud1;
      ---
      ---
}
```

5) It is not necessary to initialize all members of structure.
6) For larger program, structures may be embedded in separate header file.

⇒ Accessing Structure members -

1) Array elements are accessed using the Subscript variable, simply structure members are accessed using dot [.] operator. it is known as "structure member operator".
2) Use this operator in between "structure name" & "member name".
3)

```
Example -
#include <stdio.h>
#include <conio.h>
struct vehicle
{
   int wheels;
   char vname[20], color[10];
} v1 = {4, "Nano", "Red"};
void main()
{
   printf("vehicle No. of wheels: %d", v1.wheels);
   printf("vehicle name : %s", v1.vname);
   printf("vehicle color : %s", v1.color);
}
```

O/p:-
vehicle No. of wheels : 4
vehicle name : Nano
vehicle Color : Red.

Note:- Dot operator has heighest priority than unary, arithmetic relational, logical operator.

•> Uses / Applications of Structure -

1) Adjusting cursor position.
2) Drawing any graphics shape on screen.
3) Finding out the list of equipment attached to computer.
4) Changing the size of cursor.
5) Formatting a Floppy.
6) Hiding a file from the directory.
7) Displaying the directory of a disk.
8) Checking the memory size.
9) Store huge data, structure acts as a database.
10) Send data to the printer.
11) Intract with Keyboard / mouse to store the data.
12) Used to clear output screen contents.

* C structure using typedef *

The C programming language provides a keyword called typedef, which you can use to give a type, a new type. typedef is a keyword that is used to give a new symbolic name for the existing name in C program. This is same like defining alias for the commands.

It allows us to ~~intro~~ introduce synonyms for data types which could have been declared some other way. It is used to give new name to Structure. New name is used for creating instances, passing values to function, declaration etc.

Example :-

```
struct student
{
    int marks[2];
    char name[10];
    float avg;
};
```

•> Variables for the above structure can be declared in two ways -

1> struct student record; //for normal variables

2> typedef struct student status;

When we use "typedef" keyword before struct <tag_name> like above, after that we can simply use type defination "status" in the C program to declare structure variable.

Now, structure variable declaration will be "status record". This is equal to "struct student record". Type defination for "struct student" is status i.e. status = "struct student".

•> Alternative way to Structure Declaration using typedef in C -

```
typedef struct student
{
    int mark[2];
    char name[10];
    float avg;
} status;
```

⇒ Variable Declaration for above example -

```
status r1;    // r1 is structure variable.
status r2;    // r2 is structure variable.
```

•> Example -

```
#include<stdio.h>
#include<conio.h>
typedef struct student
{
    int id;
    char name[20];
    float per;
} status;
void main()
{
    status r1;
    r1.id = 1;
    r1.name = "Raju";
    r1.per = 86.5;
    printf(" Id is : %d ", r1.id);
    printf(" Name : %s", r1.name);
    printf(" Percentage : %f", r1.per);
    getch();
}
```

Output -

```
ID is : 1
Name : Raju
Percentage : 86.500000
```

* Array of Structures in C *

There can be array of structure in C programming to store many information of different data types. The array of structures is also known as collection of structure.

C structure is collection of different datatypes (variables) which are grouped together. Whereas array of structure is nothing but collection of structures. This is also called as structure array in C.

An array of structure is stored inside the memory is in the same way as a multidimensional array.

Example -

```
#include<stdio.h>
# include <conio.h>
# include <string.h>
 struct student
 {  int rn;
    char name[10];
 };
 void main()
 {  int i;
    struct student st[5];
    clrscr();
    printf(" enter record of 5 students : ");
    for(i=0 ; i<5 ; i++)
    {  printf(" Enter Roll No. : " );
       scanf ("%d", &st[i].rn);
       printf(" Enter Name : ");
       scanf ("%s", &st[i].name);
    }
    printf("\n Student Information List ");
    for(i=0; i<5 ; i++)
    {  printf("\n Roll No. : %d \t Name : %s", st[i].rn,
                                        st[i].name);
    }
    getch();
}
```

* Array Within Structure *

C permits the use of array as structure member. We can use single or multidimensional array of type int or float.

Example -

```
#include<stdio.h>
#include<conio.h>
struct mark
{
   char name[10];
   float m[5];
   float t;
};
void main()
{
   struct mark A1;
   int i;
   clrscr();
   printf("enter name : ");
   scanf("%s", &A1.name);
   printf("enter marks : ");
   for(i=0; i<5; i++)
   {
      scanf("%f", &A1.m[i]);
   }
   A1.t = A1.m[0] + A1.m[1] + A1.m[2] + A1.m[3] + A1.m[4];
   printf("\n Name = %s", A1.name);
   printf(" Marks = ");
   for(i=0; i<5; i++)
      printf("%f \t", A1.m[i]);
   printf("Total = %.2f", A1.t);
   getch();
}
```

* Nested Structure *

Nested structure in C is nothing but structure within structure. One structure can be declare inside other structure as we declare structure member inside a structure.

In Nested structure, There is two ways to make nesting of structure -

1.) Declare Two structure, then create the 1st structure variables inside the 2nd structure.

Example -

```
struct student_college
{
    int clg_id;
    char clg_name[50];
};
struct student_detail
{
    int id;
    char name[20];
    struct student_college c1;
} s1;
void main()
{
    struct student_detail s1 = { 1, "Raju", 71145,
                                 "Sai" };
    clrscr();
    printf(" Id = %d \n", s1.id);
    printf(" Student Name = %s\n ", s1.name);
    printf(" College ID = %d\n", s1.c1.clg_id);
    printf(" College Name = %s\n", s1.c1.clg_name);
    getch();
}
```

2.> Declare & Define a structure inside another structure. Example -

```
#include <stdio.h>
#include <conio.h>
void main()
{
   struct salary
    {
       char name[10];
       float BP;
        struct Allow
        {
          float DA, HRA;
        } A1;
    } S1;
   printf(" enter name & Basic Pay :- ");
   scanf(" %s %f", &S1.name, &S1.BP);
   S1.A1.DA  =  (S1.BP * 50) /100;
   S1.A1.HRA =  (S1.BP * 15) 100;
   printf(" Name = %s \n Basic Pay = %.2f \n DA = %.2f \n
            HRA = %.2f \n", S1.name, S1.BP, S1.A1.DA,
            S1.A1.HRA);
   getch();
}
```


* Passing Structure to Function *

Entire structure passed to functions as arguments. The structure variable is treated as any ordinary variable.

A structure can be passed to any function from main function or from any sub function. Structure defination will be available within the function only. It won't be available to other functions unless it is passed to those function by value or by address.

Else, we have to declare structure variable as global variable. That means, structure variable should be declare outside the main function. So this structure will be visible to all the function in a C program.

Syntax – (i) Declaration

```
ReturnType FunctionName( struct StructureName StructureVari);
```

(ii) Defination.

```
ReturnType FunctionName(struct StructureName StructureVariable)
 {
    ---
    ---
 }
```

(iii) Call

```
functionName (structureVariable);
```

Example –

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
struct student
 {
    int id;
    char name[20];
    float per;
};
void funt (struct student record);
void main( )
 {
    struct student record ;
    record.id = 1;
    strcpy (record.name, "Raju");
    record.per = 86.5;
```

O/p :-

Id = 1
Name = Raju
Percentage = 86.50

```
    fun1 (record);
    getch();
}
void fun1 (struct student record)
{
    printf(" Id = %d \n Name = %s \n Percentage = %.2f ",
                record.id, record.name, record.per);
}
```

Prepared By:-
Chandrashekhar Verma
An IT Instructor

* Function Returning Structure *

In C structure can be returned from function just as variables of any other type. The return type of a function is structure. Example –

```
#include <stdio.h>
#include <conio.h>
struct FRS fun1 (struct FRS v1);
struct FRS
{
    int x,y,z;
};
void main()
{
  struct FRS a,b;
  clrscr();
  printf(" enter two values = ");
  scanf("%d%d", &a.x, &a.y);
        b= fun1 (a);
  printf(" %d ", b.z);
  getch();
}
struct FRS fun1 (struct FRS v1)
{
    v1.z = v1.x + v1.y ;
    return v1 ;
}
```

* UNION *

Unions are very similar to structure. It is a concept borrowed from structure & therefore follow the same syntax as structure.

A union is a datatype which allow to make a group of different variables in the same memory area. There is major distinction between them in terms of storage. In structure each member has its own storage location, whereas all the members of a union use the same location. It can handle only one member at a time. A union also consists of data member but only one data member is active at a time.

> Declaration Unions -

Declaration syntax of union is similar to a structure. A union can be declared using the keyword 'union' as shown below -

```
union <union_name>
{
    data member-1
    data member-2
    - - - - -
    - - - - -
} <union variable>;
```

> Example -

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
union student
{   char name[20], sub[20];
    float per;
};
```

```
void main()
{
    union student r1, r2;
    clrscr();
    strcpy (r1.name, "Raju");
    strcpy (r1.sub, " maths);
    r1.per = 86.50;
    printf("\n Name \t Subject \t Percentage\n");
    printf(" %s \t %s \t %.2f ", r1.name, r1.sub, r1.per);
    strcpy (r2.name, "mani");
    printf(" Name : %s \n", r2.name);
    strcpy (r2.sub, "science");
    printf(" subject : %s\n", r2.sub);
    r2.per = 99.50;
    printf(" percentage : %.2f ", r2.per);
    getch();
}
```


O/P :-

Name	Subject	Percentage
Raju	maths	86.500000

Name : mani
subject : Scienc
percentage : 99.50.

Operations On A Union :-

- An union variable can be assigned to another union variable.
- An union variable can be passed to a function as a parameter.
- The address of the union variable can be extracted by using the address of operator (&).
- A function can accept & return a union or a pointer to a union.

•> Difference between Structure & Union -

SNo.	Structure	Union.
1.	Keyword struct is used to declare structure	Keyword union is used to declare an union.
2.	All data members in a structure are active at time.	Only one data member is active at a time.
3.	It is useful to declare a compound data type to group data members related to a person or item.	It is useful in certain cases where the user will select any one data member in the group of data members.
4.	It is commonly used in most of the applications.	It is not commonly used.
5.	The compound of memory required to store a structure variable is the sum of size of all the members in addition to the padding bytes may be provided by the compiler.	The case of memory required to store a high variable as the same as that required by its largest member.

Prepared By:-
Chandrashekhar Verma
An IT Instructor

•> A program to compair the allocate memory using struct and union.

```
# include <stdio.h>
# include <conio.h>
struct
{   char name[25];
    int code;
    float sal;
} emp;
union
{   char name[25];
    int code;
```

```
    float sal;
} employee;
void main()
{
    clrscr();
    printf("The size of structure : %d", sizeof(emp));
    printf("\nThe size of union : %d", sizeof(employee));
    getch();
}
```

Output :-

The size of structure : 31
The size of union : 25

—X—