STRUCTURES

Definition, Declaration, accessing structures, initialization, operations on structures, structures containing arrays, structures containing pointers, nested structures, self-referential structures, arrays of structures, structures and functions, structures and pointers.

Why Use Structures

- ✓ The ordinary variables can hold one piece of information and how arrays can hold a number of pieces of information of the same data type. These two data types can handle a great variety of situations. But quite often we deal with entities that are collection of dissimilar data types.

Definition:

- ✓ A structure is a collection of heterogeneous data items that are stored in consecutive memory locations.
- ✓ The structure comes under the category of user defined data types.

For example, suppose you want to store data about a book. You might want to store its name (a string), its price (a float) and number of pages in it (an int). If data about say 3 such book is to be stored, then we can follow two approaches:

- ✓ Construct individual arrays, one for storing names, another for storing prices and still another for storing number of pages.
- ✓ **Use a structure variable.**

Declaration Syntax:

```
struct structure_name
 {
      datatype1 var1,var2,…;
      datatype2 var1,var2,…;
       ……..
      datatypen var1,var2,…;
  };
```

- ✓ The struct keyword is used to declare structures and identifies the beginning of a structure definition.
- ✓ It's followed by a structure name or tag that is the name given to the structure.
- ✓ Following the tag are the structure members, enclosed in braces.
- ✓ A structure can contain any of C's data types, including arrays and other structures.
- ✓ Each variable within a structure is called a member of the structure.
- ✓ The above structure declaration is also known as template.
- ✓ The template is terminated with semicolon (;).

Examples:

- ✓ Define a structure by name student which includes student name,roll number and marks.

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

This statement defines a new data type called **struct student**.

- ✓ Define a structure by name book which includes name of the book, price and number of pages.

```
struct book
{
      char name[10];
      float price ;
      int pages ;
} ;
```

This statement defines a new data type called **struct book**.

Declaring Structure Variable

- ✓ After defining the structure, variables for that structure are to be declared to access the members of structures.
- ✓ *Syntax:* **struct** structure_name **var1,var2,var3,……var_n;**
- ✓ ***Example*:** struct book b1, b2, b3 ;

The variables **b1**, **b2**, **b3** are variables of the type **struct book.**

Note:

[1]The memory is allocated for structures only after the declaration of structure variable.

[2] The memory allocated is equal to the sum of memory required by all the members in structure definition.

Example :

```
struct book
{
      char name[10];
      float price;
      int pages;
} ;
struct book b1, b2, b3 ;
```

This statement assigns space in memory for b1,b2,b3 each. It allocates space to hold all the elements in the structure—in this case, 16 bytes—ten for name, four for price and two for pages. These bytes are always in adjacent memory locations.

✓ The variables for structure can be declared in three different ways:

Method 1:

```
struct structure_name
 {
       datatype1 var1,var2,…;
       datatype2 var1,var2,…;
            ……..        ……..
       datatypen var1,var2,…;
};
```

struct structure_name **var1,var2,var3,……var_n;**

Method 2:

```
struct structure_name
 {
          datatype1 var1,var2,…;
          datatype2 var1,var2,…;
          ……..      .       ……..
         datatypen var1,var2,…;
  }var1,var2,var3,…..varn;
```

Method3:

```
struct
{
datatype1 var1,var2,…;
datatype2 var1,var2,…;
datatypen var1,var2,…;
}var1,var2,var3,…..varn;
```

✓ The method 3 cannot be used for future declaration of variables as structure tag is not present.

Note: The use of structure tag is optional in C language.

✓ For example, Variables for structure book are declared in 3 different ways:

Example1:

```
struct book
  {
     char name ;
     float price ;
     int pages ;
  } ;
  struct book b1, b2, b3 ;
```

Example2:

```
struct book
{
   char name ;
   float price;
   int pages ;
} b1, b2, b3 ;
```

Example3:

```
struct
{
  char name ;
  float price ;
  int pages ;
  } b1, b2, b3 ;
```

Accessing Members of Structure

- ✓ The members of structure are accessed by dot (.) operator.

  Syntax: Structure_varname.member name;

- ✓ So to refer to pages of the structure defined in our structure book example we have to use, b1.pages
- ✓ Similarly, to refer to price we would use, b1.price.

Structure Initialization

- ✓ Like primary variables and arrays, structure variables can also be initialized where they are declared.

  Syntax: struct structure_name varname={list of values);

- ✓ The values should be assigned in the same order of declaration of the members in the definition.

  Example:

```
struct book
{
char name[10] ;
float price ;
int pages ;
} ;
struct book b1 = { "Basic", 130.00, 550 } ;
struct book b2 = { "Physics", 150.80, 800 } ;
```

Note:

1. Structure Members can be initialized using the assignment operator (=) using structure variable.
2. Structure Members can be initialized using scanf() – function.