



Unit VII – Part 01 (Structure in C)

Lecture Slide



AS2023



Objectives



By the end of this session, students will be able to:

- Define Structure
- Know how to define template and declare structure variable
- Access structure member
- Differentiate between structure and array
- Perform copy, compare and other operations on structure
- Demonstrate arrays of structure
- Elaborate arrays within structure
- Explain structure within structure
- Write Pointer using Structure
- Allocate dynamic memory for structure
- Use structure in Function.



Introduction



- Structures are constructed data type
- Provides a mechanism for packing data of different data types
- A convenient tool for handling a group of logically related data item
- Structure helps to organize data in a more meaningful way



Defining Structure

- Unlike array, structure must be defined first for their format that may be used later to declare structure variable
- Consider the book database consisting of *title*, *author*, *no of pages* & *price*
- We can define structure to hold this information as follows:

```
struct bookDatabase{  
    char title[20];  
    char author[20];  
    int pages;  
    int price;  
};
```
- **struct** keyword declares a structure to hold the data fields, namely *title*, *author*, *pages* and *price*



Defining Structure

- General format of structure definition is as follows

```
struct    tag_name{  
    data_type    member1;  
    data_type    member2;  
    -----  
    -----  
  
};
```

- Note the following
 - The template is terminated with semi colon
 - While the entire definition is considered as statement, each member is declared as independently for its name and type in a separate statement inside the template
 - The **tag_name** can be used to declare structure variable of its type, later in the program



Array vs Structure



- Both are structure data types as they provide a mechanism that enable us to access and manipulate data in a relatively easy manner.
- But they differ in a number of ways
 - Array is a collection of related data elements of same data type. Structure can have elements of different data types
 - Array behaves like built in data type. All we have to do is to declare an array variable and use it. But in case of structure, first we have to design and declare a data structure before the variable of that type is declared and used



Declaring Structure Variable



- The structure variable(s) can be defined only after defining the template
- Its similar to the declaration of variables of any other data type
- Declaration of structure variable must contain the following
 - The keyword *struct*
 - The structure *tag_name*
 - List of variable names separated by coma
 - A terminating semicolon
- **Example** `struct bookDatabase book1, book2, book3;`
- Each variables will have all the elements as specified by the template



Declaring Structure Variable



- The complete declaration might look like this

```
struct bookDatabase{  
    char title[20];  
    char author[20];  
    int pages;  
    int price;  
};  
struct bookDatabse book1, book2, book3;
```

- Remember members of structure themselves are not variables.
- They doesn't occupy any memory until they are associated with structure variables
- When compiler comes across a declaration statement, it reserves a memory address for structure variable



Declaring Structure Variable



- It is also allowed to combine both the structure definition and variable declaration in one statement.

```
struct bookDatabase{  
    char title[20];  
    char author[20];  
    int pages;  
    int price;  
} book1, book2, book3;
```

- The use of tag name is optional here but not recommended!!!



Example: Declaration of Structure



```
struct Person {  
    char name[50];  
    int cidNo;  
    float salary;  
};  
  
int main() {  
    struct Person person1,  
        person2, p[20];  
    return 0;  
}
```

```
struct Person {  
    char name[50];  
    int cidNo;  
    float salary;  
} person1, person2, p[20];
```



Accessing Structure Member



- As mentioned, structure members themselves are not variables
- They should be linked to the structure variable in order to make them meaningful members
- The link between a member and a variable is established using the *member operator* ‘.’ which is also known as ‘**dot operator**’ or ‘**period operator**’

Example: `book1.price`

- We can assign values to the members as

```
book1.price=250;
```

- We can initialize using `scanf()`

```
scanf("%d", &book1.price);
```

- **DEMO:** WAP program using structure to read the information of one person from the key board and print the same on the screen.



Example



```
// Program to add two distances
//(feet-inch)
#include <stdio.h>
struct Distance {
    int feet;
    float inch;
} dist1, dist2, sum;
int main() {
    printf("1st distance\n");
    printf("Enter feet: ");
    scanf("%d", &dist1.feet);
    printf("Enter inch: ");
    scanf("%f", &dist1.inch);
    printf("2nd distance\n");
    printf("Enter feet: ");
    scanf("%d", &dist2.feet);
    printf("Enter inch: ");
    scanf("%f", &dist2.inch);
```

```
// adding feet
sum.feet = dist1.feet +
           dist2.feet;
// adding inches
sum.inch = dist1.inch +
           dist2.inch;
// changing to feet if inch
//is greater than 12
while (sum.inch >= 12) {
    ++sum.feet;
    sum.inch = sum.inch -
               12;
}
printf("Sum of distances =
       %d\'-%.1f\"",
       sum.feet, sum.inch);
return 0;
```

```
}
```



Rules for Structure Initialization



- We cannot initialize individual members inside the structure template
- The order of values enclosed in braces must match the order of members in the structure definition
- It is permitted to have partial initialization. We can initialize only the first few members and leave the remaining blank. The uninitialized members should be only at the end of the list
- The uninitialized members will be assigned default values as follows
 - Zero for integers & floating point numbers
 - '\0' for characters and strings



Copying and Comparing two Structure variables



- Two variables of the same structure type can be copied the same ways as ordinary variables.
- Lets consider *person1* and *person2* belongs to same structure, then the following are valid:

```
person1=person2;  
person2=person1;
```



Copying and Comparing two Structure variables Cont..



- However, C doesn't permit any logical operations on structure variables such as .

```
person1 == person2;  
person1 != person2;
```

- In cases we need to compare them we can do so by comparing members individually
- **DEMO:** WAP to compare two structure variable



CLASS ACTIVITY



Get your following details using structure and display the information.

1. Your name
2. Your age
3. Your gender
4. October month's stipend
5. Your village Name



Thank you