

SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore - 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23ITT101- PROBLEM SOLVING & C PROGRAMMING

I YEAR /I SEMESTER

Unit V – STRUCTURE AND UNION

Topic : Structure









- Structures and Unions
 - Introduction
 - Defining a Structure
 - Declaring Structure Variables
 - Accessing Structure Members





- C supports a constructed data type known as structures, a mechanism for packing data of different types.
- It is a convenient tool for handling a group of logically related data item.



Structures and Unions - Introduction



Example:

Time - Seconds, minutes and hours

Date - Day, month and year

Book - Author, title, book and year

It is used to organize complex data in a more meaningful way.





- Unlike arrays, Structures must be defined first for their format that may be used later to declare structure variables.
- Consider a book database consisting of book name, author, number of pages and price.
- We can define a structure to hold this information as follows.





struct book_bank char title[20]; char author[15]; int pages; float price; };





 The keyword struct declares a structure to hold the details of four data fields, namely title, author, pages and price are called structure elements or members.

book_bank - name of the structure or structure tag.

Here tag name used to declare variables that have the tag's structure.





Arrays allow to define type of variables that can hold several data items of the same kind. Similarly **structure** is another user defined data type available in C that allows to combine data items of different kinds.

Structures are used to represent a record. Suppose you want to keep track of your books in a library. You might want to track the following attributes about each book –

Title Author Subject Book ID Pages Price





Each members may belong to a different type of data. The

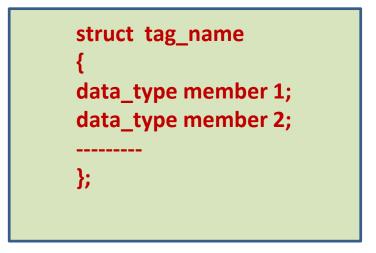
tag name may be used subsequently to declare variables that have the tag's structure

Title→ Char array
Author \rightarrow Char array
Subject→ Char array
Book ID→ Integer
Pages→ Integer
Price→ Float





General format of a structure:





Example of a structure



struct Student
{
 char name[25];
 int age;
 char branch[10];
 // F for female and M for male
 char gender;
 };

- Here struct Student declares a structure to hold the details of a student which consists of 4 data fields, namely name, age, branch and gender. These fields are called structure elements or members.
- Each member can have different datatype, like in this case, name is an array of char type and age is of int type etc. **Student** is the name of the structure and is called as the **structure tag**.







- In defining a structure, note the following syntax:
- The template is **terminated with a semicolon**.
- While the entire definition is considered as a statement, each member is declared independently for its name and type in a separate statement inside the template.
- The tag name book-bank can be used to declare structure variables of its type.





- Array is a collection of related data elements of same type.
- Array is a derived data type
- Array behaves like built-in data type. Here we have to declare an array variable and use it.

Structures

- Structure can have elements of different types.
- Structure is a programmer defined one.
- Here we have to design and declare a data structure before the

variables of the type are declared and used.





Arrays Vs Structures

- Both the arrays and structure are classified as structured data types as they provide a mechanism that enable us to access and manipulate the data in easy manner, but they differ in the following manner.
- After defining a structure format we can declare variables of that type.
- A structure variable declaration is similar to the declaration of variables of any other data type.





It includes the following elements.

- The **keyword** struct
- The structure **tag name**
- List of variable names separated by commas
- A terminating **semicolon**.

EXAMPLE:

struct book_bank book1, book2, book3;

Book1, book2 and book3 are variables of type struct book_bank.





```
struct book_bank
{
    char title[20];
    char author[15];
    int pages;
    float price;
}
struct book_bank, book1, book2, book3;
```





Declaring Structure variables separately

```
struct Student
{
    char name[25];
    int age;
    char branch[10];
    //F for female and M for male char gender;
    };
    struct Student S1, S2; //declaring variables of struct Student
```



Declaring Structure variables with structure definition



struct student
{
 char name[25];
 int age;
 char branch[10];
 //F for female and M for male char gender;
}S1, S2;

Here S1 and S2 are variables of structure Student. However this approach is not much recommended.





- **Members** of a structure themselves are **not variables**.
- They do not occupy any memory until they are associated with the structure variables.
- When the **compiler comes across declaration** statement, it **reserves memory space** for the structure variables.
- It is also allowed to combine both the structure definition and variables declaration in one statement.





- The use of tag name is optional.
- Without a tag name we cannot use it for future declarations.
- Structure definitions appear at a beginning of the program file before any variable or functions are defined.
- In such cases the definition is global and can be used by other functions as well.





Type - Defined Structure:

Use keyword typedef to define a structure,

typedef struct
{
type number 1;
type number 2;
} type_name;





type_name represents structure definition associated with it and it is used to declare structure variables. type_name variable1, variable2, ..;

- 1. Type_name is a the type definition name, not a variable
- 2. We cannot define a variable with typedef declaration.





- We can access and assign values to the members of a structure in a number of ways.
- Members are not variables
- Structure members have no meaning individually without the structure.
- Here the link between member and a variable is established using a member operator '.' which is also known as 'dot operator' or period operator.

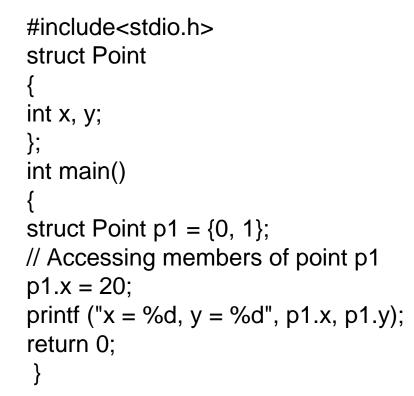




Example:

book1.price; book1.title; book1.author; book1.pages;











```
#include<stdio.h>
#include<string.h>
struct Student
char name[25];
int age;
char branch[10];
//F for female and M for male
char gender;
};
int main()
struct Student s1;
/*s1 is a variable of Student type and
age is a member of Student */
s1.age = 18;
/* using string function to add name */
strcpy(s1.name, "Arun");
/* displaying the stored values */
printf("Name of Student 1: %s\n", s1.name);
printf("Age of Student 1: %d\n", s1.age);
return 0;
```





#include <stdio.h> #include <string.h> /* print Book1 info */ struct Books { printf("Book 1 title : %s\n", Book1.title); char title[50]: printf("Book 1 author : %s\n", Book1.author); char author[50]: printf("Book 1 subject : %s\n", Book1.subject); char subject [100]; printf("Book 1 book id : %d\n", Book1.book id); int book id: /* print Book2 info */ int main() { printf("Book 2 title : %s\n", Book2.title); printf("Book 2 author : %s\n", Book2.author); struct Books Book1: printf("Book 2 subject : %s\n", Book2.subject); /* Declare Book1 of type Book */ printf("Book 2 book id : %d\n", Book2.book id); struct Books Book2: /* Declare Book2 of type Book */ return 0; /* book 1 specification */ strcpy(Book1.title, "C Programming"); strcpy(Book1.author, "Nuha Ali"); strcpy(Book1.subject, "C Programming Tutorial"); Output:: Book1.book id = 6495407; Book 1 title : C Programming Book 1 author : Nuña Ali /* book 2 specification */ Book 1 subject : C Programming Tutorial strcpy(Book2.title, "Telecom Billing"): Book 1 book_id : 6495407 strcpy(Book2.author, "Zara Ali"); strcpy(Book2.subject, "Telecom Billing Tutorial"); Book 2 title : Telecom Billing Book2.book_id = 6495700; Book 2 author : Zara Ali Book 2 subject : Telecom Billing Tutorial Book 2 book id : 6495700







- C supports a constructed data type known as structures, a mechanism for packing data of different types.
- Structures must be defined first for their format that may be used later to declare structure variables.
- A structure variable declaration is similar to the declaration of variables of any other data type.





