NIELIT GORAKHPUR

Course Name: O Level (2nd Sem B1, B2 and B3 Batch)
Topic: Structure in C
Subject: C Language
Date: 01-June-2020

What is Structure?

Structure is user defined data type which is used to store heterogeneous data under unique name. Keyword 'struct' is used to declare structure. The variables which are declared inside the structure are called as 'members of structure'.

Defining a structure

Syntax:

```
struct structure_nm
{
        <data-type> element 1;
        <data-type> element 2;
        - - - - - - - - - - - -
        - - - - - - - - -
        <data-type> element n;
}struct_var;
```

Members of structure with different data types

In the above syntax, start with the ***struct*** keyword, and give a structure name, then inside the curly braces, we have to mention all the member variables, which are nothing but normal C language variables of different data types like ***int, float, array*** etc.

After the closing curly brace, we can specify one or more structure variables, again this is optional.

Note: The closing curly brace in the structure type declaration must be followed by a semicolon (***;***).

Example of Structure

```
struct student
{
        int roll;
        char name [25];
        float per;
        char gender;
}s1;
```

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 data type, like in this case, name is an array of ***char*** type and roll is of ***int*** type and percentage is of ***float*** type etc. **Student** is the name of the structure and is called as the **structure tag**.

Declaring Structure Variables

- It is possible to declare variables of a structure, either along with structure definition or after the structure is defined.
- Structure variable declaration is similar to the declaration of any normal variable of any other datatype.
- Structure variables can be declared in following two ways:

- **Declaring Structure variables separately**

```
struct student
{
          int roll;
          char name [25];
          float per;
          char gender;
};
struct student s1,s2;
```

Declaring variables of struct student

- **Declaring Structure variables with structure definition**

```
struct student
{
          int roll;
          char name [25];
          float per;
          char gender;
}s1,s2;
```

Here s1 and s2 are variables of structure student.

<u>Accessing Structure Members</u>

- Structure members can be accessed using member operator '**.**' .
- It is also called as *'**dot operator**'* or *'**period operator**'*.
- To access members of a structure using pointers, use the *'->'* ***Arrow operator***.

<u>Structure Initialization</u>

Like a variable of any other datatype, structure variable can also be initialized at compile time.

```
struct student
{
          int roll;
          char name [25];
          float per;
          char gender;
}s1;
```

struct student s1 = { 101, "amit kumar", 50.65,'M' };

Initialization of each member of structure separately using (**. dot**) operator

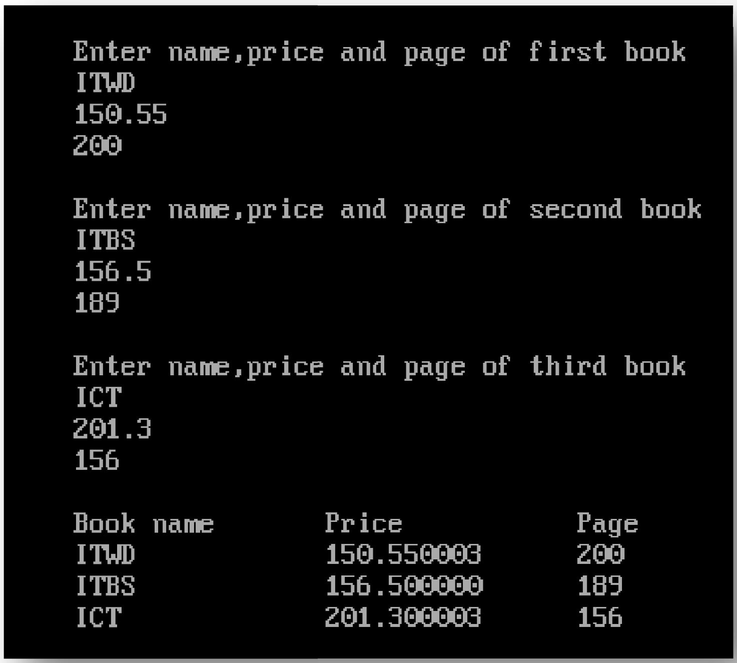
s1.roll = 101;
s1.name="amit kumar";
s1.per=50.65;
s1.gender='M';

Example 1: Write a program to define a structure name book and store information of three books and then display (using dot operator).

```
#include<stdio.h>
#include<conio.h>
void main()
{
    struct book
    {
      char x[10];
      int page;
```

```
       float pri;
      }
   struct book b1,b2,b3;
   clrscr();
   printf("\nEnter name,price and page of first book\n");
   scanf("%s%f%d",&b1.x,&b1.pri,&b1.page);
   printf("\nEnter name,price and page of second book\n");
   scanf("%s%f%d",&b2.x,&b2.pri,&b2.page);
   printf("\nEnter name,price and page of third book\n");
   scanf("%s%f%d",&b3.x,&b3.pri,&b3.page);
   printf("\nBook name\tPrice\t\tPage");
   printf("\n%s\t\t%f\t%d",b1.x,b1.pri,b1.page);
   printf("\n%s\t\t%f\t%d",b2.x,b2.pri,b2.page);
   printf("\n%s\t\t%f\t%d",b3.x,b3.pri,b3.page);
   getch();
   }
```

Output:



```
Enter name,price and page of first book
ITWD
150.55
200

Enter name,price and page of second book
ITBS
156.5
189

Enter name,price and page of third book
ICT
201.3
156

Book name       Price           Page
ITWD            150.550003      200
ITBS            156.500000      189
ICT             201.300003      156
```

<u>**Array of Structure**</u>

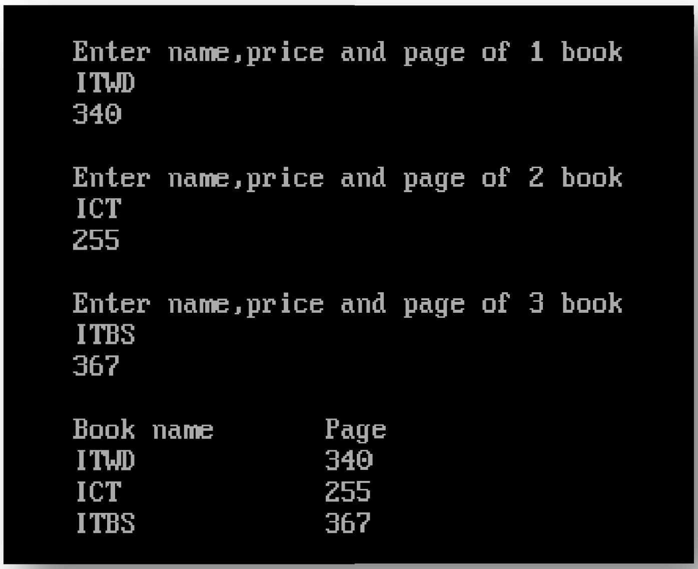
We can also declare an array of **structure** variables. in which each element of the array will represent a **structure** variable.

Example: struct book b[3];

```
#include<stdio.h>
#include<conio.h>
void main()
{
  struct book
   {
    char x[10];
    int page;
   };
struct book b[3];
int i;
clrscr();
for(i=0;i<3;i++)
```

```
 {
  fflush(stdin);
  printf("\nEnter name,price and page of %d book\n",i+1);
  scanf("%s%d",&b[i].x,&b[i].page);
 }
printf("\nBook name\tPage");
for(i=0;i<3;i++)
printf("\n%s\t\t%d",b[i].x,b[i].page);
getch();
}
```

Output:



```
Enter name,price and page of 1 book
ITWD
340

Enter name,price and page of 2 book
ICT
255

Enter name,price and page of 3 book
ITBS
367

Book name       Page
ITWD            340
ICT             255
ITBS            367
```

Nested Structures

Nesting of structures, is also permitted in C language. Nested structures means, that one structure has another structure as member variable.

Syntax:

```
struct structure_nm
{
        <data-type> element 1;
        <data-type> element 2;
        - - - - - - - - - - - - -
        - - - - - - - -
        <data-type> element n;
                struct structure_nm {
                        <data-type> element 1;
                        <data-type> element 2;
                        - - - - - - - - - - - - -
                        - - - - - - - - - -
                        <data-type> element n;
                        }inner_struct_var;
}outer_struct_var;
```