5.3 POINTERS TO STRUCTURES

You can define pointers to structures in the same way as you define pointers to any other variable.

a) Declaration of Pointer to a Structure

You can declare a pointer to a structure (or structure pointer) as follows −

```
struct Books *struct_pointer;
```

b) Initialization of Pointer to a Structure

You can store the address of a structure variable in the above pointer variable **struct_pointer**. To find the address of a structure variable, place the '&' operator before the structure's name as follows −

```
struct_pointer = & book1;
```

Let's store the address of a struct variable in a struct pointer variable.

```
struct book{
  char title[10];
  char author[20];
  double price;
  int pages;
};
struct book book1 = {"Learn C", "Dennis Ritchie", 675.50, 325},
struct book *strptr;
```

c) Accessing Members Using Pointer to a Structure

To access the members of a structure using a pointer to that structure, you must use the → **operator** as follows −

```
struct_pointer->title;
```

C defines the → symbol to be used with struct pointer as the **indirection operator** (also called **struct dereference operator**). It helps to access the elements of the struct variable to which the pointer reference to.

Example

In this example, **strptr** is a pointer to **struct book book1** variable.

Hence, **strrptr→title** returns the title, just like **book1.title** does.

Open Com

```c
#include <stdio.h>
#include <string.h>

struct book{
   char title[10];
   char author[20];
   double price;
   int pages;
};

int main (){
   struct book book1 = {"Learn C", "Dennis Ritchie", 675.50, 325};
   struct book *strptr;
   strptr = &book1;

   printf("Title: %s \n", strptr -> title);
   printf("Author: %s \n", strptr -> author);
   printf("Price: %lf \n", strptr -> price);
   printf("Pages: %d \n", strptr -> pages);
   return 0;
}
```

Output

Title: Learn C

Author: Dennis Ritchie

Price: 675.500000

Pages: 325

Example 2: DECLARING A POINTER TO A STRUCT ARRAY

We can also declare a pointer to a struct array. C uses the indirection operator (→) to access the internal elements of struct variables.

Example

The following example shows how you can declare a pointer to a struct array −

```c
#include <stdio.h>

struct book {
   char title[15];
   double price;
   int pages;
};

int main(){

   struct book b[3] = {
      {"Learn C", 650.50, 325},
      {"C Pointers", 175, 225},
      {"C Pearls", 250, 250}
   };

   struct book *ptr = b;

   for(int i = 0; i < 3; i++){
printf("Title: %s \tPrice: %7.2lf \tPages: %d\n", ptr -> title, ptr -> price, ptr -> pages);
    ptr++;
  }
 return 0;
}
```

Output

Title: Learn C Price: 650.50 Pages: 325

Title: C Pointers Price: 175.00 Pages: 225

Title: C Pearls Price: 250.00 Pages: 250