Structures

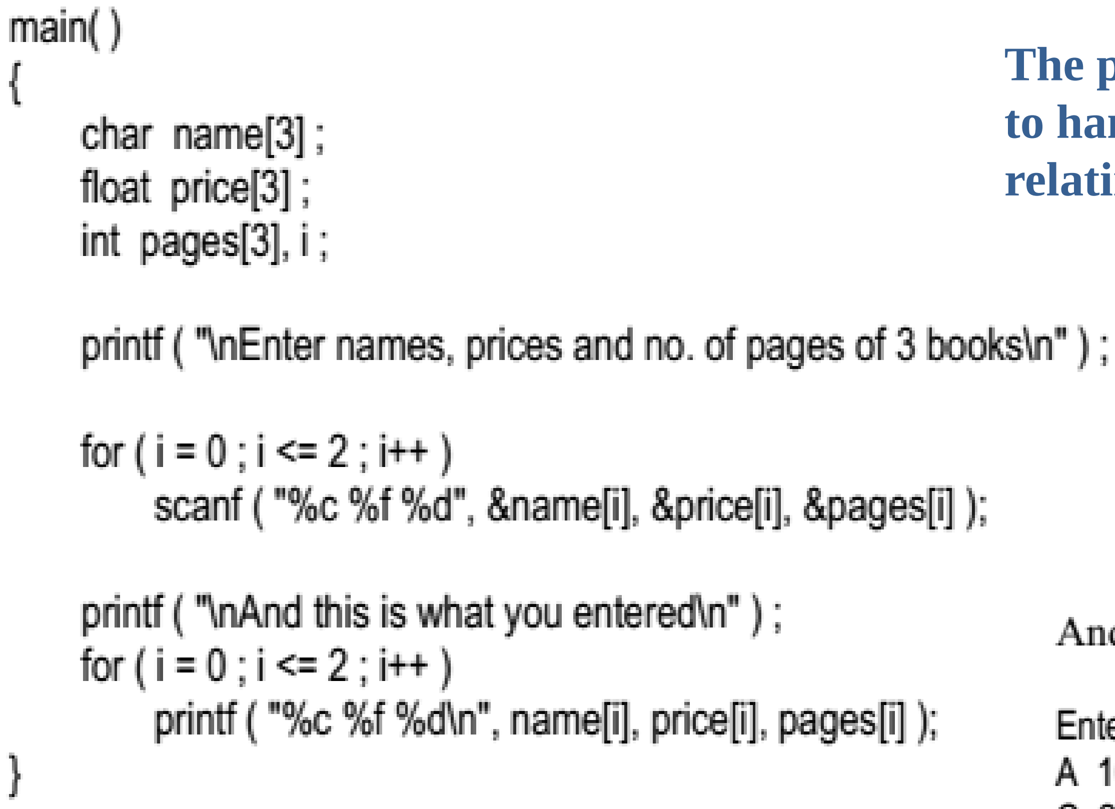
Motilal Nehru National Institute of Technology Allahabad
Department of Computer Science & Engineering,
Computer Programming (CS12101)
B.Tech 1st Semester (A, B, C,D and E sections, 2019-2020)

Prepared By: Dr. J Sathish Kumar,
Assistant Professor, CSED, MNNIT Allahabad, Prayagraj
Email: sathish613@mnnit.ac.in

Introduction **Structures**

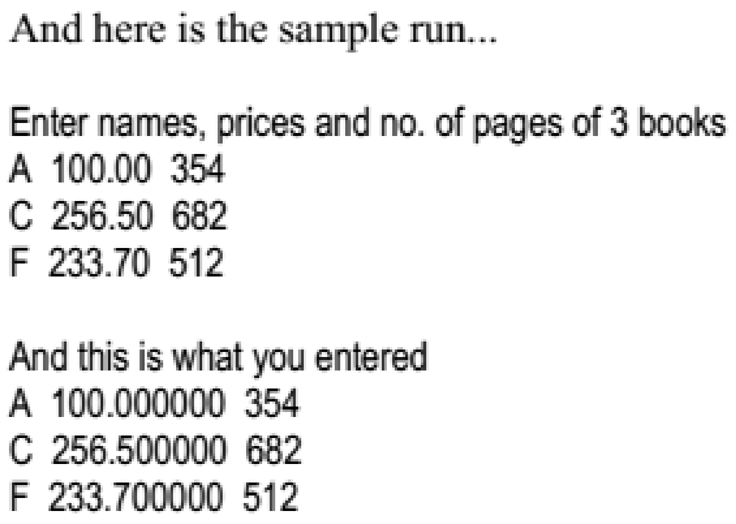
- C language. It wouldn’t have been so popular had it been able to handle only all **int**s, or all **float**s or all **char**s at a time.
- Real World data deal with entities that are collections of things, each thing having its own attributes.
  - For example, just as the entity we call a ‘book’ is a collection of things such as title, author, call number, publisher, number of pages, date of publication, etc.
- Structures is a collection of dissimilar data types.

Example: Store data about a book



```
main( )
{
    char  name[3] ;
    float  price[3] ;
    int  pages[3], i ;

    printf ( "\nEnter names, prices and no. of pages of 3 books\n" ) ;

    for ( i = 0 ; i <= 2 ; i++ )
        scanf ( "%c %f %d", &name[i], &price[i], &pages[i] );

    printf ( "\nAnd this is what you entered\n" ) ;
    for ( i = 0 ; i <= 2 ; i++ )
        printf ( "%c %f %d\n", name[i], price[i], pages[i] );
}
```

The program becomes more difficult to handle as the number of items relating to the book go on increasing.



And here is the sample run...

```
Enter names, prices and no. of pages of 3 books
A  100.00  354
C  256.50  682
F  233.70  512

And this is what you entered
A  100.000000  354
C  256.500000  682
F  233.700000  512
```

Example: Store data about a book

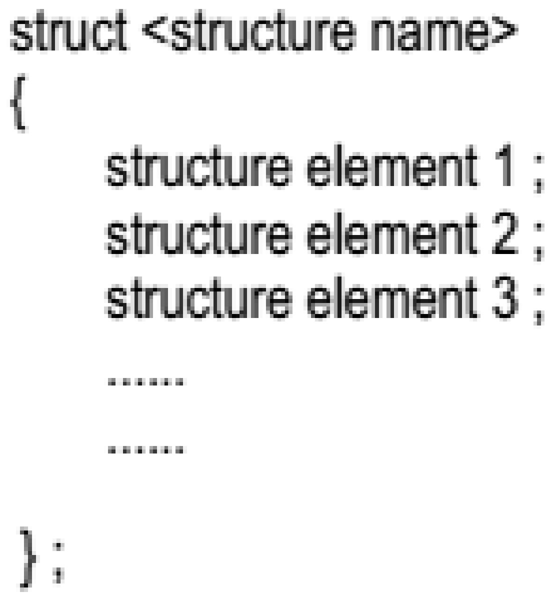
```
main( )
{
    struct book
    {
        char  name ;
        float  price ;
        int  pages ;
    } ;
    struct book  b1, b2, b3 ;

    printf ( "\nEnter names, prices & no. of pages of 3 books\n" ) ;
    scanf ( "%c %f %d", &b1.name, &b1.price, &b1.pages ) ;
    scanf ( "%c %f %d", &b2.name, &b2.price, &b2.pages ) ;
    scanf ( "%c %f %d", &b3.name, &b3.price, &b3.pages ) ;

    printf ( "\nAnd this is what you entered" ) ;
    printf ( "\n%c %f %d", b1.name, b1.price, b1.pages ) ;
    printf ( "\n%c %f %d", b2.name, b2.price, b2.pages ) ;
    printf ( "\n%c %f %d", b3.name, b3.price, b3.pages ) ;
}
```

A structure contains a number of data ypes grouped together. These data types nay or may not be of the same type.

And here is the output...

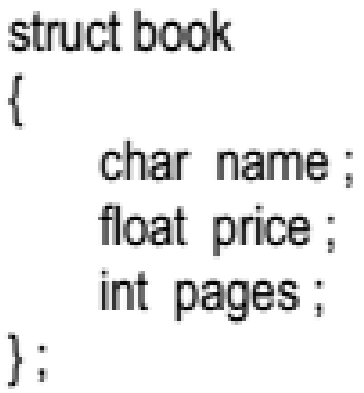
```
Enter names, prices and no. of pages of 3 book
A  100.00  354
C  256.50  682
F  233.70  512

And this is what you entered
A  100.000000  354
C  256.500000  682
F  233.700000  512
```

Declaring a Structure



```
struct <structure name>
{
        structure element 1 ;
        structure element 2 ;
        structure element 3 ;
        ......
        ......
 } ;
```

Example



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

Inside main or Outside main

Accessing of structure elements

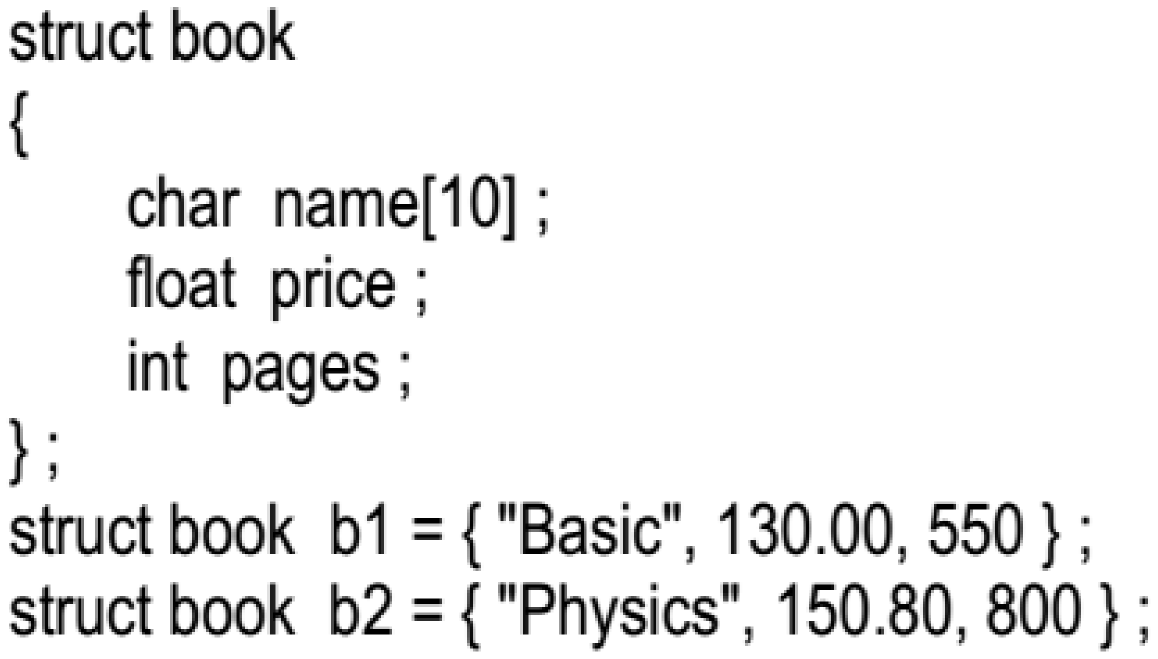
```
struct book
{
      char  name ;
      float  price ;
      int  pages ;
} ;
struct book  b1, b2, b3 ;
```

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

Structure use a dot (.) operator.

b1.pages   or   b1.price

Static Initialization



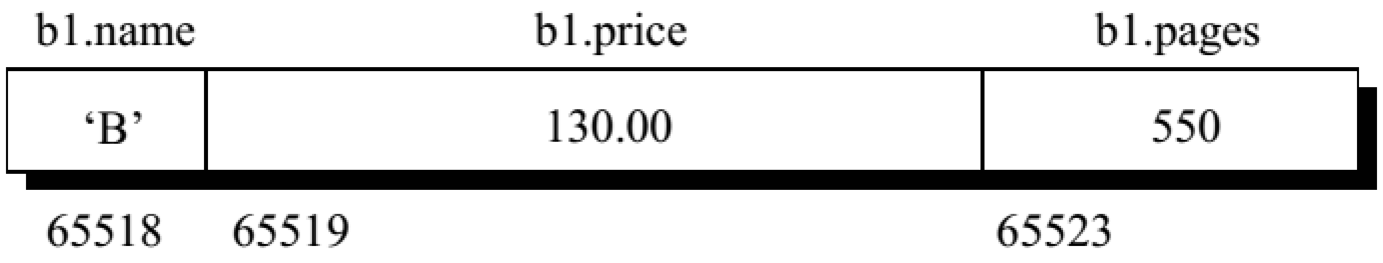
```
struct book
{
        char  name[10] ;
        float  price ;
        int  pages ;
} ;
struct book  b1 = { "Basic", 130.00, 550 } ;
struct book  b2 = { "Physics", 150.80, 800 } ;
```

Structure Elements Memory Storage

```
main( )
{
    struct book
    {
        char  name ;
        float  price ;
        int  pages ;
    } ;
    struct book  b1 = { 'B', 130.00, 550 } ;

    printf ( "\nAddress of name = %u", &b1.name ) ;
    printf ( "\nAddress of price = %u", &b1.price ) ;
    printf ( "\nAddress of pages = %u", &b1.pages ) ;
}
```

Here is the output of the program...

```
Address of name = 65518
Address of price = 65519
Address of pages = 65523
```



Array of Structures

```
/* Usage of an array of structures */
main( )
{
    struct book
    {
        char  name ;
        float  price ;
        int  pages ;
    } ;

    struct book  b[100] ;
    int  i ;

    for ( i = 0 ; i <= 99 ; i++ )
    {
        printf ( "\nEnter name, price and pages " ) ;
        scanf ( "%c %f %d", &b[i].name, &b[i].price, &b[i].pages ) ;
    }

    for ( i = 0 ; i <= 99 ; i++ )
        printf ( "\n%c %f %d", b[i].name, b[i].price, b[i].pages ) ;
}
```

Additional Features of Structures

- The values of a structure variable can be assigned to another structure variable of the same type using the assignment operator.
- It is not necessary to copy the structure elements piece-meal.

```
main( )
{
    struct employee
    {
        char  name[10] ;
        int  age ;
        float  salary ;
    } ;
    struct employee  e1 = { "Sanjay", 30, 5500.50 } ;
    struct employee  e2, e3 ;

    /* piece-meal copying */
    strcpy ( e2.name, e1.name ) ;
    e2.age = e1.age ;
    e2.salary = e1.salary ;

    /* copying all elements at one go */
    e3 = e2 ;

    printf ( "\n%s %d %f", e1.name, e1.age, e1.salary ) ;
    printf ( "\n%s %d %f", e2.name, e2.age, e2.salary ) ;
    printf ( "\n%s %d %f", e3.name, e3.age, e3.salary ) ;
}
```

The output of the program would be...

```
Sanjay 30 5500.500000
Sanjay 30 5500.500000
Sanjay 30 5500.500000
```

Additional Features of Structures

- One structure can be nested within another structure.

```
main( )
{
    struct address
   {
        char  phone[15] ;
        char  city[25] ;
        int  pin ;
   } ;

   struct emp
   {
        char  name[25] ;
        struct address  a ;
   } ;
        struct emp  e = { "jeru", "531046", "nagpur", 10 };

        printf ( "\nname = %s phone = %s", e.name, e.a.phone ) ;
        printf ( "\ncity = %s pin = %d", e.a.city, e.a.pin ) ;
}
```

And here is the output...

```
name = jeru phone = 531046
city = nagpur pin = 10
```

Accessing variables
e.a.pin or e.a.city

Additional Features of Structures

- Like an ordinary variable, a structure variable can also be passed to a function.

```
struct book
{
    char  name[25] ;
    char  author[25] ;
    int  callno ;
} ;

main( )
{
    struct book  b1 = { "Let us C", "YPK", 101 } ;
    display ( b1 ) ;
}
```

```
display ( struct book  b )
{
    printf ( "\n%s %s %d", b.name, b.author, b.callno ) ;
}
```

And here is the output...

```
Let us C YPK 101
```

Additional Features of Structures

- Usage of **Structures a**nd Pointers

```
main( )
{
    struct book
    {
        char  name[25] ;
        char  author[25] ;
        int  callno ;
    } ;
    struct book  b1 = { "Let us C", "YPK", 101 } ;
    struct book  *ptr ;

    ptr = &b1 ;
    printf ( "\n%s %s %d", b1.name, b1.author, b1.callno ) ;
    printf ( "\n%s %s %d", ptr->name, ptr->author, ptr->callno ) ;
}
```

Additional Features of Structures

- Usage of **Structures a**nd Pointers using functions

```
struct book
{
    char  name[25] ;
    char  author[25] ;
    int  callno ;
} ;

main( )
{
    struct book  b1 = { "Let us C", "YPK", 101 } ;
    display ( &b1 ) ;

 }
```

```
display ( struct book  *b )
{
    printf ( "\n%s %s %d", b->name, b->author, b->callno ) ;
}
```

And here is the output...

```
Let us C YPK 101
```