LAB 15

A structure contains a number of data types grouped together. These data types may or may not be of the same type. The following example illustrates the use of this data type.

```
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 ) ;
```

And here is the output...

```
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
```

This program demonstrates two fundamental aspects of structures:

(a) declaration of a structure
(b) accessing of structure elements

Let us now look at these concepts one by one.

Declaring a Structure

In our example program, the following statement declares the structure type:

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

This statement defines a new data type called **struct book**. Each variable of this data type will consist of a character variable called **name**, a float variable called **price** and an integer variable called **pages**. The general form of a structure declaration statement is given below:

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

} ;
```

Once the new structure data type has been defined one or more variables can be declared to be of that type. For example the variables **b1**, **b2**, **b3** can be declared to be of the type **struct book**, as,

```
struct book b1, b2, b3 ;
```

This statement sets aside space in memory. It makes available space to hold all the elements in the structure—in this case, 7 bytes—one for **name**, four for **price** and two for **pages**. These bytes are always in adjacent memory locations.

If we so desire, we can combine the declaration of the structure type and the structure variables in one statement.

For example,

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

is same as...

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

or even...

```
struct

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

```
/* Memory map of structure elements */
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
```

Actually the structure elements are stored in memory as shown in the Figure 10.1.

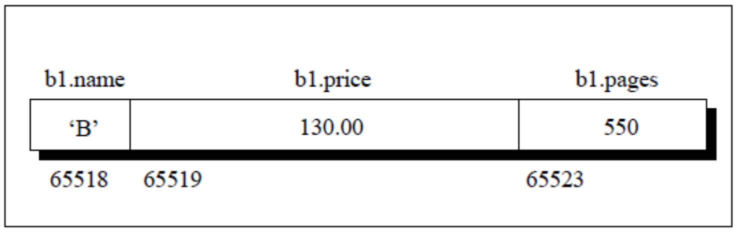

Figure 10.1

File Operations

There are different operations that can be carried out on a file. These are:

a) Creation of a new file
b) Opening an existing file
c) Reading from a file
d) Writing to a file
e) Moving to a specific location in a file (seeking)
f) Closing a file

Opening a File

Reading from File

Let us now write a program to read a file and display its contents on the screen. We will first list the program and show what it does, and then dissect it line by line.

```
main( )
{
FILE *fp ;
char ch ;
fp = fopen ( "PR1.C", "r" ) ;
while ( 1 )
{
ch = fgetc ( fp ) ;
if ( ch == EOF )
break ;
printf ( "%c", ch ) ;
}
fclose ( fp ) ;
}
```

Writing on File

```
main( )
{
FILE  *ft ;
char ch ;
ft = fopen ( "pr2.c", "w" ) ;
if ( ft == NULL )
{
puts ( "Cannot open target file" ) ;
fclose ( fs ) ;
exit( ) ;
}
while ( 1 )
{
ch = fgetc ( fs ) ;
if ( ch == EOF )
break ;
else
fputc ( ch, ft ) ;
}
fclose ( fs ) ;
```

File Opening Modes

In our first program on disk I/O we have opened the file in read ("r") mode. However, "r" is but one of the several modes in which we can open a file. Following is a list of all possible modes in which a file can be opened. The tasks performed by **fopen( )** when a file is opened in Searches file. If the file is opened successfully **fopen( )** loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened **fopen( )** returns NULL.

Operations possible – reading from the file.

each of these modes are also mentioned.	
"r"	Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
"w"	Operations possible – writing to the file.
"a"	Searches file. If the file is opened successfully **fopen( )** loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file. Operations possible - adding new contents at the end of file.
"r+"	Searches file. If is opened successfully **fopen( )** loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file. Operations possible - reading existing contents, writing new contents, modifying existing contents of the file.
"w+"	Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file. Operations possible - writing new contents, reading them

	back and modifying existing contents of the file.
"a+"	Searches file. If the file is opened successfully **fopen( )** loads it into memory and sets up a pointer which points to the first character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file. Operations possible - reading existing contents, appending new contents to end of file. Cannot modify existing contents.

Exercise:

1- There is a structure called **employee** that holds information like employee code, name, date of joining. Write a program to create an array of the structure and enter some data into it. Then ask the user to enter current date. Display the names of those employees whose tenure is 3 or more than 3 years according to the given current date.

2- Write a program to read a file and display contents with its line numbers.
3- Write a program to write data on a text file.
4- Write a program to add the contents of one file at the end of another.