STRUCTURES

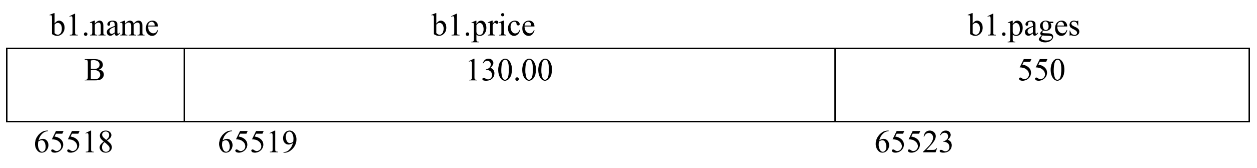
Definition, Declaration, accessing structures, initialization, operations on structures, structures containing arrays, structures containing pointers, nested structures, self-referential structures, arrays of structures, structures and functions, structures and pointers.

Memory allocation for structure:

- ✓ Whatever be the elements of a structure, they are always stored in contiguous memory locations.
- ✓ The following program would illustrate this:

```
 /* 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 below



Note: Differences between Arrays and Structures

Array	Structure
Array is a derived data type.	Structure is a programmer or user - defined data type.
It allocates memory only for the elements of the subscripts. Array allocates static memory and uses index / subscript for accessing elements of the array.	It does not allocate memory till the elements of the structure are accessed. Structures allocate dynamic memory and uses (.) operator for accessing the member of a structure.

It allocates memory of same files that is if we declare integer type array then it allocates 2 byte memory for each cell.	It allocates the memory for the highest data type.
It does not contain structure within itself.	It contain array within itself.
It can contain only homogeneous data types.	It can contain only non-homogeneous data types.
Array is a pointer to the first element of it We can access the array by using index number.	Structure is not a pointer
The elements of the array are contiguous in memory	The elements of structure are accessed by using dot (.) operator with structure reference name.
Array element access takes less time v/s structures.	The elements of a structure may not be contiguous.

Sample Program:

Define a structure for student which include roll number ,name,age and marks . Write a program to read and display the information of 3 students.

```
#include<stdio.h>
#include<string.h>
struct student
 {
 int rno;
 char name[10];
 int marks,age;
 };
void main()
{
//assigning values to structure variable s1 using initialization

struct student s1={2,"Gandhi",89,18};
struct student s2,s3;

//assigning values to s2 using assignment operator.

s2.rno=5;
s2.marks=90;
s2.age=18;
strcpy(s2.name,"Ram ");

//reading values for s3 using standard input function.
printf("\n enter rno,name ,marks,age of student: ");
scanf("%d%s%d%d",&s3.rno,s3.name,&s3.marks,&s3.age);
printf("\n\n");
printf("Details of student 1:\n");
printf("\n roll number: %d",s1.rno);
printf("\n name :%s",s1.name);
printf("\n marks: %d",s1.marks);
printf("\n age: %d",s1.age);
printf("\n\n");
printf("Details of student 2:\n");
printf("\n roll number: %d",s2.rno);
printf("\n name :%s",s2.name);
printf("\n marks: %d",s2.marks);
printf("\n age: %d",s2.age);
printf("\n\n");
printf("Details of student 3:\n");
printf("\n roll number: %d",s3.rno);
printf("\n name :%s",s3.name);
printf("\n marks: %d",s3.marks);
printf("\n age: %d",s3.age);
}
```

Output :

```
enter rno,name,marks,age of student:1

Raju
92
18

Details of student 1:
roll number: 2
name :Gandhi
marks :89
age: 18

Details of student 2:
roll number: 5
name :Ram
marks :90
age: 18

Details of student 3:
roll number: 1
name :Raju
marks :92
age: 18
```

Operations on Structure Variable:

- ✓ The only operation that is allowed on structure variables is assignment operation. Two variables of same structure can be copied similar to ordinary variables. If P1 and P2 are variables of struct P, then P1 values can be assigned to P2 as

  P2=P1;

- ✓ where values of P1 will be assigned to P2 member by member.

Sample Program: Illustrating operation on structure variables

```
#include<stdio.h>
#include<string.h>
 struct student
 {
   int rno;
   char name[10];
   int marks,age;
 };
 void main()
 {
   //assigning values to structure variable s1 using initalization struct student
   s1={2,"Gandhi",89,18};
   struct student s2; s2=s1;
   printf("\nDetails of student 1:\n");
   printf("\n roll number: %d",s1.rno);
   printf("\n name :%s",s1.name);
   printf("\n marks: %d",s1.marks);
   printf("\n age: %d",s1.age);
   printf("\n\n");
   printf("Details of student 2:\n");
   printf("\n roll number: %d",s2.rno);
   printf("\n name :%s",s2.name);
   printf("\n marks: %d",s2.marks);
   printf("\n age: %d",s2.age);
 }
```

output:

```
   Details of student 1:
   roll number: 2
   name :Gandhi
   marks :89
   age: 18

   Details of student 2:
   roll number: 2
   name :Gandhi
   marks :89
   age: 18
```

Note: *C does not permit any logical operations on structure variables.*

Operation On Individual Members of Structures

- ✓ All operations are valid on individual members of structures.

Sample Program: Illustrating operations on structure members

```
#include<stdio.h>
#include<string.h>
 struct student
 {
    int rno;
    char name[10];
    int marks,age;
  };
void main()
{
    int m;
    //assigning values to structure variable s1 using initalization
    struct student s1={2,"Gandhi",89,18};
    struct student s2;
    s2=s1;
    printf("\nDetails of student 1:\n");
    printf("\n roll number: %d",s1.rno);
    printf("\n name :%s",s1.name);
    printf("\n marks: %d",s1.marks);
    printf("\n age: %d",s1.age);
    printf("\n\n");
    printf("Details of student 2:\n");
    printf("\n roll number: %d",s2.rno);
    printf("\n name :%s",s2.name);
    printf("\n marks: %d",s2.marks);
    printf("\n age: %d",s2.age);

    //comparsion of two student details.

    m=((s1.rno==s2.rno)&&(s1.marks==s2.marks))?1:0;

    if(m==1)
    printf("\n both the details are same");
    else
    printf("\n both the details are not same");
}
```

Output:

```
Details of student 1:
roll number: 2
name :Gandhi
marks :89
age: 18

Details of student 2:
roll number: 2
name :Gandhi
marks :89
age: 18

both the details are same
```

Array within Structure:

- ✓ As we know, structure is collection of different data type. Like normal data type, It can also store an array as well.
- ✓ In arrays within structure the member of structure is array.

Syntax for array within structure

```
                 struct struct-name
                 {
                       datatype var1;         // normal variable
                       datatype array [size]; // array variable

                       - - - - - - - - - -
                       - - - - - - - - - -
                       datatype varN;
                };
                struct struct-name obj;
```

Example for array within structure

```
#include<stdio.h>
struct Student
{
  int Roll;
  char Name[10];
  int Marks[3];  //array of marks
  int Total;
  float Avg;
};
void main()
  {
    int i;
    struct Student S;
    printf("\n\nEnter Student Roll : ");
    scanf("%d",&S.Roll);
    printf("\n\nEnter Student Name : ");
    scanf("%s",&S.Name);
    S.Total = 0;
    for(i=0;i<3;i++)
    {
      printf("\n\nEnter Marks %d : ",i+1);
      scanf("%d",&S.Marks[i]);
      S.Total = S.Total + S.Marks[i];
    }
    S.Avg = S.Total / 3;
    printf("\nRoll : %d",S.Roll);               printf("\nName
    : %s",S.Name);                printf("\nTotal :
    %d",S.Total);               printf("\nAverage :
    %f",S.Avg);
  }
```

Output :

```
Enter  Student Roll : 4
Enter Student Name : GANDHI
Enter marks 1 : 80
Enter marks 2 : 90
Enter marks 3 : 75
Roll : 4
Name : GANDHI
Total : 245
Average : 81.000000
```

✓ The memory allocated for structure variable s is 24 bytes(2-roll,10-name,6-marks,2-total,4avg).