STRUCTURES

Structure is a user-defined data type that can store related information (of different data types) together. The major difference between a structure and an array is that an array can store only information of same data type. A structure is therefore a collection of variables under a single name. The variables within a structure are of different data types.

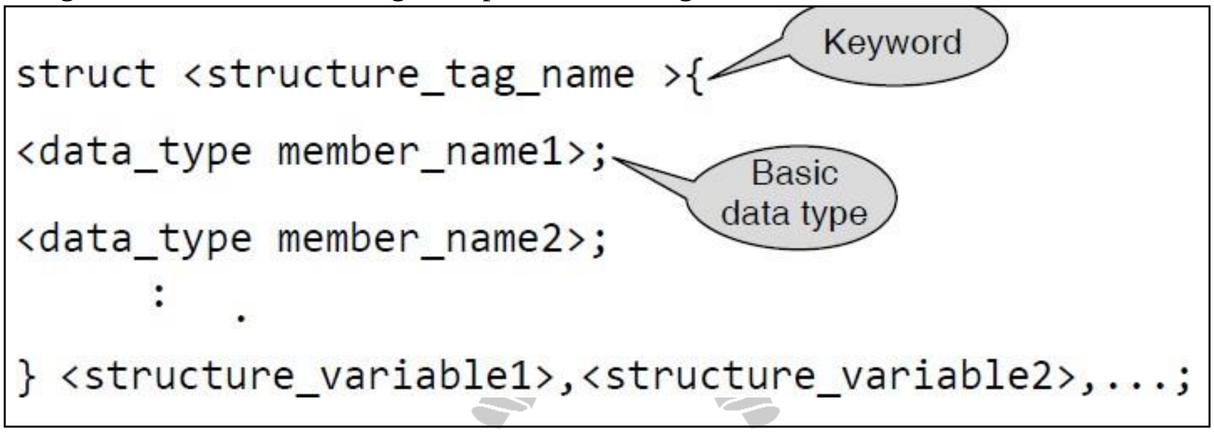
- A structure is a collection of variables under a single name. These variables can be of different types, Therefore, a structure is a convenient way of grouping together several pieces of related information.
- Complex hierarchies can be created by nesting structures.

- Declaration/Initializing/Accessing
- Nesting of Structures
- Arrays of Structures
- Structures and Pointers
- Structures and Functions

8.2.1 DECLARING STRUCTURES AND STRUCTURE VARIABLES

A structure is declared by using the **keyword struct** followed by an **optional structure tag** followed by the **body of the structure**. The ***variables* or *members*** of the structure are declared within the body.

The general format of declaring a simple structure is given as follows.



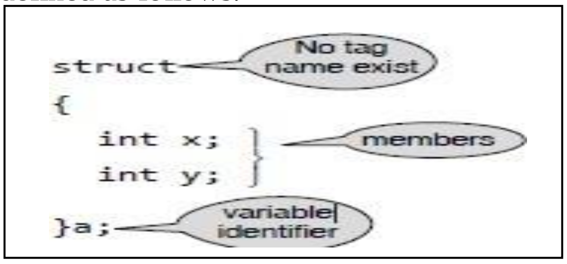
```
struct <structure_tag_name >{
<data_type member_name1>;
<data_type member_name2>;
      :   .
} <structure_variable1>,<structure_variable2>,...;
```

There are three different and/or define a structure. These are

- Variable structure
- Tagged structure
- Type-defined structure

1. A variable structure may be defined as follows.

```
struct
{
member_list
}variable_identifier;
```



```
struct
{
   int x;
   int y;
}a;
```

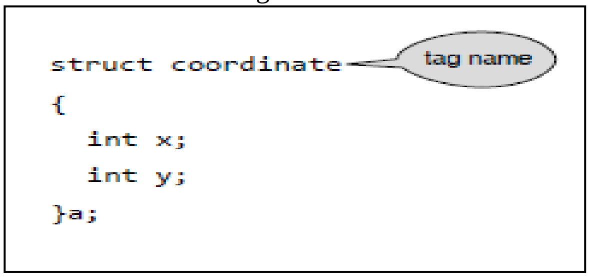
```
struct
{
int r_no;
char name[20];
char course[20];
float fees;
} student1;
```

where. ... a,student1-are called as structure variables

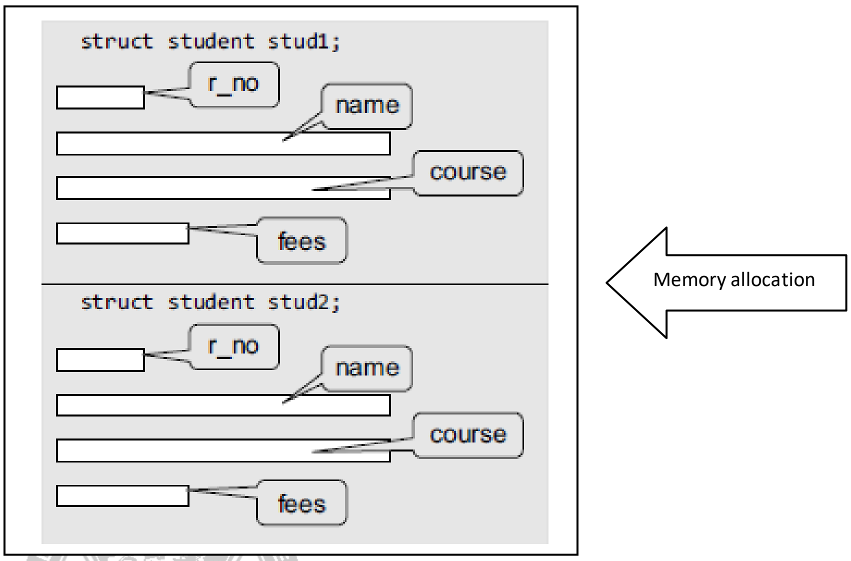
...

2.A tagged structure has been described earlier. It has the following format:

```
struct tag_name
{
member_list
}variable_identifier;
```



```
struct student
{
int r_no;
char name[20];
char course[20];
float fees;
} stud1, stud2;
```



1.Type-defined structure declaration is as follows

- typedef keyword enables the programmer to create a new data type name by using an existing data type.

typedef existing_data_type new_data_type;

```
typedef struct newdatatype
{
member_list;

} newdatatype variable_identifier;
```

```
typedef struct student
{
int r_no;
char name[20];
char course[20];
float fees;
} student stud1;
```

When we precede a struct name with the **typedef** keyword, then the **struct** becomes a new type. **student** becomes a new data type. To declare a variable of structure student, you may write **student stud1;**

Note that we have not written **struct student stud1**.

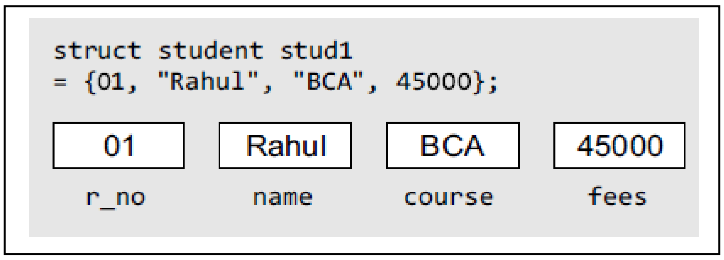
INITIALIZATION OF STRUCTURES

A structure can be initialized in the same way as other data types are initialized. Initializing a structure means assigning some constants to the members of the structure.

```
struct struct_name
{
data_type member_name1;
data_type member_name2;
data_type member_name3;
.......................
}struct_var = {constant1, constant2, constant3,...};
```

Example-1

```
struct student
{
int r_no;
char name[20];
char course[20];
float fees;
}stud1 = {01, "Rahul", "BCA", 45000};
```



```
struct student stud1
= {01, "Rahul", "BCA", 45000};
```

ACCESSING THE MEMBERS OF A STRUCTURE

The members are accessed by relating them to the structure variable with a dot operator. The general form of the statement for accessing a member of a structure is as follows.

```
< structure_variable >.< member_name > ;
```

```
stud1.r_no
stud1.name
stud1.course
```

```
Stud2.r_no
Stud2.name
Stud2.course
```

Each member of a structure can be used just like a normal variable, but its name will be a bit longer. A structure member variable is generally accessed using a '.' (dot) operator.

```
stud1.r_no = 01;
stud1.name = "Rahul";
stud1.course = "BCA";
stud1.fees = 45000;
```

To input values for data members of the structure variable stud1, we may write

```
scanf("%d", &stud1.r_no);
scanf("%s", stud1.name);
```

Similarly, to print the values of structure variable stud1, we may write

```
printf("%s", stud1.course);
printf("%f", stud1.fees);
```

Example-2 both initializing and accessing member data

```
struct {
float p, q,
int r;
} k = {k .p = 3.0, k.q = 7.9, k.r = 5};
```

Write a program using structures to read and display the information about a student

```
#include <stdio.h>
struct student
{
int roll_no;
char name[80];
float fees;
char DOB[80];
};

int main()
{
struct student stud1;

printf("\n Enter the roll number : ");
scanf("%d", &stud1.roll_no);

printf("\n Enter the name : ");
scanf("%s", stud1.name);

printf("\n Enter the fees : ");
scanf("%f", &stud1.fees);

printf("\n Enter the DOB : ");
scanf("%s", stud1.DOB);

printf("\n ********STUDENT'S DETAILS *******");
printf("\n ROLL No. = %d", stud1.roll_no);
printf("\n NAME = %s", stud1.name);
printf("\n FEES = %f", stud1.fees);
printf("\n DOB = %s", stud1.DOB);
getch();
return 0;
}
```

Output

```
Enter the roll number : 01
Enter the name : Rahul
Enter the fees : 45000
Enter the DOB : 25–09–1991
********STUDENT'S DETAILS *******
ROLL No. = 01
NAME = Rahul
FEES = 45000.00
DOB = 25–09–1991
```

C program to read and print employee's record using structure

```
#include <stdio.h>

  struct employee
{
    char    name[30];
    int     empId;
    float salary;
};

int main()
{
    struct employee emp;                // declare structure variable
    printf("\nEnter details :\n");
    printf("\n ************");
    printf("Name ?:");
    scanf("%s",emp.name);
    printf("ID ?:");
    scanf("%d",&emp.empId);
    printf("Salary ?:");
    scanf("%f",&emp.salary);

    /*print employee details*/

    printf("\n Employee detail is:");
    printf("\n ************");
    printf("Name: %s" ,emp.name);
    printf("Id: %d"     ,emp.empId);
    printf("Salary: %f\n",emp.salary);
    return 0;
}
```

Output

```
Enter details :
**********
Name ?:Mike
ID ?:1120
Salary ?:76543

Employee detail is:
***************
Name: Mike
ID: 1120
Salary: 76543.000000
```

<u>Write a program to copy and compare structures for employee details</u>

// A structure can be assigned to another structure of the same type. Here is an example of assigning one structure to another.

```
#include <stdio.h>
struct employee
{
char grade;
int basic;
float allowance;
};

int main()
{
struct employee ramesh={'B', 6500, 812.5};      /* creating & initializing
member of employee */

struct employee vivek;                          /* creating another member of
employee */

vivek = ramesh;                                 /* copy respective members of
ramesh to vivek */

printf("\n vivek's grade is %c, vivek.grade);
printf("\n vivek's basic is Rs %d, vivek. Basic);
printf("\n vivek's allowance is Rs %f",  vivek.allowance);
return 0;
}
```

<u>Output:</u>

```
vivek's grade is B
vivek's basic is Rs 6500
vivek's allowance is Rs
812.500000
```

Using Typedef :- A program that prints the weight of various sizes of fruits.

```
#include <stdio.h>
typedef struct fruits
{
float  big;
float medium;
float small;
}fruits weight;

int main()
{
weight apples={200.75,145.5,100.25};
weight pears={150.50,125,50};
weight mangoes={1000, 567.25, 360.25};

printf("\n\n apples: big size %f kg, medium size %f kg, small size %f kg", apples.big,
apples.medium, apples.small);

printf("\n\n pears: big size %f kg, medium size %f kg,small size %f kg", pears.big,
pears.medium, pears.small);

printf("\n\n mangoes: big size %f kg, medium size %f kg,small size %f kg", mangoes.big,
mangoes.medium, mangoes.small);
return 0;
}
```

Output:

```
apples: big 200.75kg, medium 145.50kg, small 100.25kg
pears: big 150.50kg, medium 125.00kg, small 50.00kg
mangoes: big 1000kg, medium 567.25kg, small 360.25kg
```

Nesting of Structures

A structure can be placed within another structure. In other words, structures can contain other structures as members. A structure within a structure means nesting of structures.

In such cases, the dot operator in conjunction with the structure variables are used to access the members of the innermost as well as the outermost structures.

```
typedef  struct name
{
char first_name[20];
char  mid_name[20];
char last_name[20];
}NAME;

typedef  struct dob
{
int dd;
int mm;
int yy;
}DATE;

typedef struct student
{
int r_no;
NAME name;
char course[20];
DATE DOB;
float fees;
} STU;
```

assign values to the structure fields, we will write

```
student stud1;
stud1.r_no = 01;
stud1.name.first_name = "Janak";
stud1.name.mid_name = "Raj";
stud1.name.last_name = "Thareja";
stud1.course = "BCA";
stud1.DOB.dd = 15;
stud1.DOB.mm = 09;
stud1.DOB.yy = 1990;
stud1.fees = 45000;
```

Write a program to read and display the information of a student using a nested structure.

```
#include <stdio.h>
struct DOB
{
int  day;
int month;
int year;
};

struct student
{
int roll_no;
char name[100];
float fees;
struct DOB date;
};

int main()
{
struct student stud1;

printf("\n Enter the roll number : ");
scanf("%d", &stud1.roll_no);

printf("\n Enter the name : ");
scanf("%s", stud1.name);

printf("\n Enter the fees : ");
scanf("%f", &stud1.fees);

printf("\n Enter the DOB : ");
scanf("%d %d %d", &stud1.date.day, &stud1.date.month, &stud1.date.year);

printf("\n ********STUDENT'S DETAILS *******");
printf("\n ROLL No. = %d", stud1.roll_no);
printf("\n NAME = %s", stud1.name);
printf("\n FEES = %f", stud1.fees);
printf("\n DOB = %d – %d – %d", stud1.date.day, stud1.date.month, stud1.date.year);
getch();
return 0;
}
```

Output

```
Enter the roll number : 01
Enter the name : Rahul
Enter the fees : 45000
Enter the DOB : 25 09 1991
********STUDENT’S DETAILS *******
ROLL No. = 01
NAME = Rahul
FEES = 45000.00
DOB = 25 – 09 – 1991
```

Write A program to demonstrate nesting of structures and accessing structure members.

```
#include <stdio.h>
struct outer                              /* declaration of outer structure */
{
int out1;                                 /* member of outer structure */
float out2;                               /* member of outer structure */

struct inner                              /* declaration of inner structure */
{
int in1;                                  /* member of inner structure */
float in2;                                /* member of inner structure */
}invar;                           /* structure_variable of inner structure*/
};

int main()
{
struct outer outvar;                      /* declaring structure_variable of outer */
outvar.out1= 2;                           /* assigning values to members */
outvar.out2= 10.57;
outvar.invar.in1= 2* outvar.out1;                 /* assigning values to members */
outvar.invar.in2= outvar.out2 + 3.65;

printf(" out1=%d, out2=%f, in1=%d, in2=%f",outvar.out1,  outvar.out2,outvar.invar.in1,
outvar.invar.in2);
return 0;
}
```

Output:

```
out1=2, out2= 10.57, in1=4, in2= 14.22
```

Write a program to read, display, add, and subtract two complex numbers.

```
#inclu
de
<stdio
.h>
#inclu
de
<coni
o.h>
int
main()
{
typedef struct complex
{
In
ra;
int
im
ag
;
}COMPLEX;
COMPLEX c1, c2, sum_c, sub_c;

Int

opti

on;

do
{
printf("\n ******** MAIN MENU
*********");printf("\n 1. Read the
complex numbers"); printf("\n 2.
Display the complex numbers");
printf("\n 3. Add the complex
numbers"); printf("\n 4. Subtract the
complex numbers"); printf("\n 5.
EXIT");
printf("\n Enter

your option : ");

scanf("%d",

&option);

switch(option)
```

Output

```
******** MAIN MENU *********
1. Read the complex numbers
2. Display the complex numbers
3. Add the complex numbers
4. Subtract the complex numbers
5. EXIT

Enter your option : 1
Enter the real and imaginary parts of the first complex
number : 4 5
Enter the real and imaginary parts of the second complex
number : 2 3
Enter your option : 2
The first complex numbers is : 4+5i
The second complex numbers is : 2+3i
Enter your option : 3
The sum of two complex numbers is : 6+8i
Enter your option : 4
The difference between two complex numbers is : 2+2i

Enter your option : 5
```

```
{
case 1:
printf("\n Enter the real and imaginary parts of the first complex
number : ");scanf("%d %d", &c1.real, &c1.imag);
printf("\n Enter the real and imaginary parts of the second complex
number : ");scanf("%d %d", &c2.real, &c2.imag);
break;
case 2:
printf("\n The first complex number is : %d+%di",
c1.real,c1.imag); printf("\n The second complex number is :
%d+%di", c2.real,c2.imag);break;
case 3:
sum_c.real = c1.real
+ c2.real;
sum_c.imag =
c1.imag + c2.imag;
printf("\n The sum of two complex numbers is : %d+%di",sum_c.real,
sum_c.imag);break;
case 4:
sub_c.real = c1.real
– c2.real;
sub_c.imag =
c1.imag – c2.imag;
printf("\n The difference between two
complex numbersis :%d+%di", sub_c.real,
sub_c.imag);
break;
}
}while(option != 5);
return 0;
}
```