Prabodh
PAGE No. :
DATE :

CHAPTER 9
Structure

Arrays and Strings => similar data (int, float, char)
Structure can hold => dissimilar data

Syntax for creating structure
A C structure can be created as follows:

```
struct employee {   // This declares a new
    int code;       // user defined data-types
    float salary;
    char name[10];
};
```
↳ Semicolon is important.

We can use this user defined data-type as follows:

```
struct employee e1;
```
=> creating a structure variable.

```
strcpy(e1.name, "Abhinav");
e1.code = 100;
e1.salary = 71.22;
```

So a structure in C is a collection of variables of different types, under a single name.

Quick Quiz: Write a program to store the details of 3 employee from user defined data. Use the structure declared above.

Why use structure ?

We can creat the data types in the employee structure separately but when the number of properties in a structure increases, it becomes difficult for use to create data variable without structures.
In a not shell:

(a) Structure keep the data organized.
(b) Structure make data mangement easy for the programmer.

Array of Structure

Just like an array of integers, an array of floats and an array of character, we can create an array of structures.

```
Struct employee facebook[100];
                    // an array of structure.
```

We can access the data using:

```
facebook[0].code = 100;
facebook[1].code = 101;
        .
        .
        .
        So on...
```

Initializing structures

Structures can also be initialized as follows:

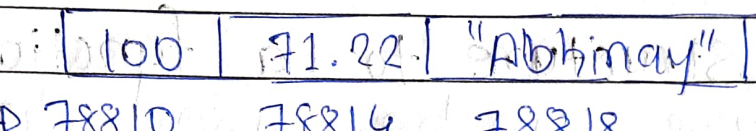
```
struct employee abhinay = {100, 71.22, "Abhinay"};
```

DATE :

```
struct employee twinkle = {0};
                    // all elements set to 0
```

Structures in memory

Structures are stored in contiguous memory locations for the structure e1 of type struct employee, memory layout looks like this:



In an array of structure, these employee instances are stored adjacent to each other.

Pointer to Structures

A pointer to structure can be created as follows:

```
struct employee *ptr;
ptr = &e1;
```

Now we can print structure elements using:

```
printf ("%d", *(ptr).code);
```

Arrow operators

Instead of writing *(ptr).code, we can use arrow operator to access structure properties as follows.

*(ptr).code or ptr->code

Here -> is known as the arrow operator.

Passing structure to a function

A structure can be passed to a function just like any othe data type.

```
void show(struct employee e); // function prototy
```

Quick Quiz: Complete this show function to display the content of employee.

Typedef Keyword

We can use the typedef keyword to create an alias name for data types in C. typedef is more commonly used with structures.

```
struct complex{
    float real;
    float img;
};
```

=> struct complex C1, C2; for defining complex numbers

```
typedef struct complex{
    float real;
    float img;
} complexNo;
```

=> ComplexNo C1, C2; for defining complex numbers

Chapter 9
Practice Set

Que 1 Creat a two dimensional vector using structures in C.

Que.2 Write a function sumVector which returns the sum of two vectors passed to it. The vectors must be two-dimensional.

Que 3. twenty integers are to be stored in memory what will you prefer - Array or Structure?

Que.4. Write a program to illustrate the use of arrow operator → in C.

Que 5. Write a program with a structure representing a complex numbers.

Que 6. Creat an array of 5 complex numbers creat in problem 5 and display then with the help of a display function. The value must be taken as an input from the user.

Que 7. Write problem 5 structure using typedef keyword.

Que 8. Creat a structure representing a bank account of a coustomer. What fields did you use and why?

Que 9. Write a structure capable of storing date write a function to compare these dates.

Que 10. Solve problem 9 for time using typedef keyword.