Structures

A simple variable can store a single value only & arrays can store multiple values but with similar datatype only. However in many real life applications, we need to store multiple values with dis-similar datatypes. In such cases we use “structures”. For eg if we wish to store the details of an employee like his name, age & salary OR details of a product in a shop/company like prodid, prod\_name, price, wt, qty\_avai, etc. This means we can manage multiple records of say employees, students, products, etc.

Definition – A structure is a collection of values with similar/dis-similar datatypes.

Syntax to declare a structure

struct structure\_name

{

basic\_datatype1 vara, varb, varc, …..;

……

basic\_datatypen var1, var2, var3, …..;

};

For eg to store details of employee

struct emp

{

Fields/Members/Elements of structure

int empid;

char name[10];

float sal;

};

The above is only the definition of a structure with no memory allocated to it. To allocate memory and store data, we need to create structure variables OR array of structures.

struct emp e1, e2, e3;

e1 e2 e3

empid – 10

name –

salary - 25000

empid –

name –

salary -

empid –

name –

salary -

To add data, we use the . (dot) syntax i.e. structure\_varname.fieldname

e1.empid = 10;

e1.salary = 25000;

e1.name = “Sumit”; // error, bcoz name is a string & strings being array, do not support = (assignment) operator

strcpy(e1.name,”Sumit”); // syntax target, source

To store data of say 10 emps, the above method is tedious. Instead use array of structures

struct emp arr\_emps[10];

0 1 2 3 …. 9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10  sumeet  25000 | 6  amit  14000 | 1  veena  19000 | 9  paresh  32000 | … |  |  |

arr\_emps[10] =

empid

name

salary

To display Veena’s record (order to display is independent of the way data is stored in structure)

printf(“Name - %s, Age - %d, Salary Rs. - %.2f”,arr\_emps[2].name,arr\_emps[2].age,arr\_emps[2].salary);

1st preference – Array Name, 2nd preference – index, 3rd preference – member of structure

**CALL BY VALUE** e **CALL BY REFERENCE**

e

empid name salary empid name salary

main()

~~124 sumit 50000.55~~

125 karan 75000.00

124 sumit 50000.55

600

empid name salary ptr\_e

600

~~124 sumit 50000.55~~

125 karan 75000.00

modify()

Here ptr\_e being a pointer, will require only 4 bytes of memory which is much less than the copy of var e in modify()