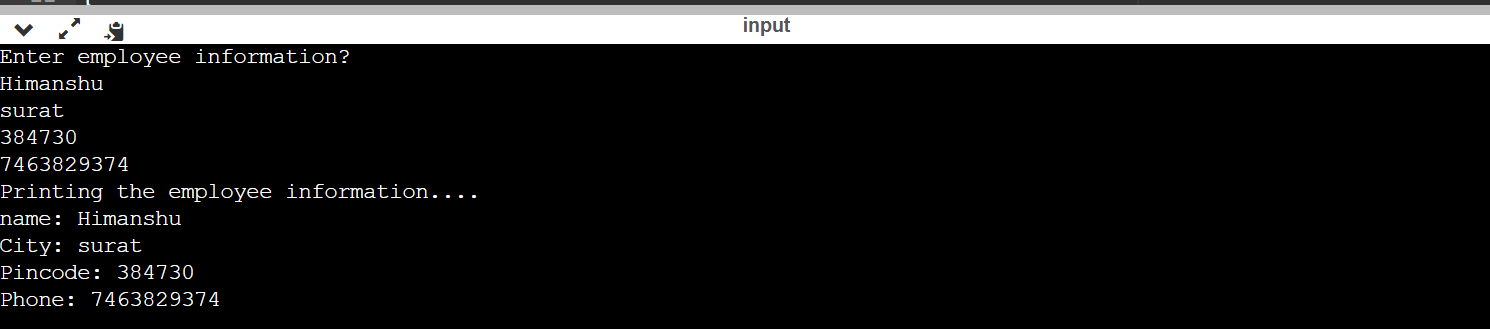
Structures in C

* First of all, as we say Structure is collection of one or more variables of different data types that are grouped together in a single name.
* Each variable in the structure is known as a **member** of the structure.
* Nested structure: C provides us the feature of nesting one structure within another structure by using which, complex data types are created.





1. **Structure types- Separate structure:**

* Here, we create two structures, but the dependent structure should be used inside the main structure as a member.

**struct** Date

{

**int** dd;

**int** mm;

**int** yyyy;

};

**struct** Employee

{

**int** id;

**char** name[20];

**struct** Date doj;

}emp1;

1. Embedded Structure:

**struct** Employee

{

**int** id;

**char** name[20];

**struct** Date

    {

**int** dd;

**int** mm;

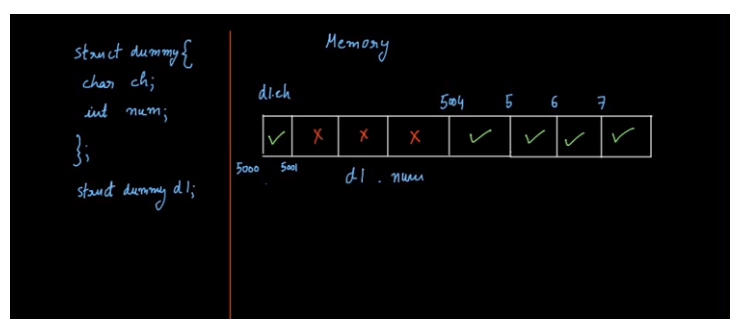
**int** yyyy;

    }doj;

}emp1;

**Padding in C:**

* Many processors expect memory for variables to be aligned based on the size of the variable. Padding is done by compiler to align the data and to increase the performance in cost of memory.
* Here ‘char’ is only 1 byte but after 3-byte padding, the number starts at 4-byte boundary.  For ‘int’ and ‘double’, it takes up 4 and 8 bytes respectively.



#include <stdio.h>

**struct** student

{

**char** a;

**char** b;

**int** c;

};

**int** main()

{

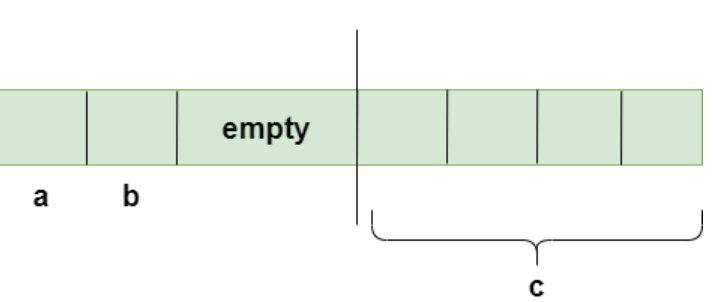
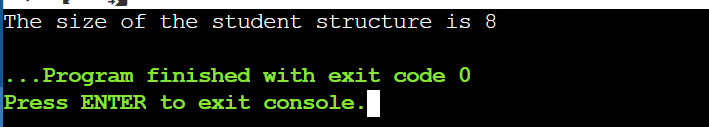
**struct** student stud1;  // variable declaration of the student type..

   // Displaying the size of the structure student.

   printf("The size of the student structure is %d", **sizeof**(stud1));

**return** 0;

}



* Here, a Short of 2 byte can be allocated in memory like 0x100 or 0x101. And an ‘int’ of 4 bytes, must start at a 4-byte boundary like 0x104 or 0x108. And again, Short of 2 byte  
  can be allocated in memory like 0x100 or 0x101.

