# Structure IN C

**Dr. Sumit Srivastava**
**Dept. of CSE, BIT Mesra Ranchi**
**Email:- sumit@bitmesra.ac.in**





## Structure

- Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a member of the structure.
- Unlike an array, a structure can contain many different data types (int, float, char, etc.).





## Structure Declaration

- To declare structure in C before using it in program.
- In structure declaration, specify its member variables along with their datatype.
- Use the struct keyword to declare the structure.





## Structure Declaration

- **Syntax**

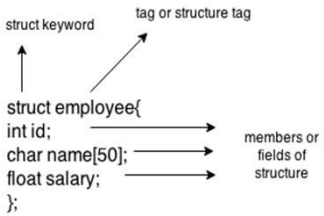
```
struct structure_name {
    data_type member_name1;
    data_type member_name1;
    ....
    ....
};
```

- The above syntax is also called a structure template or structure prototype and no memory is allocated to the structure in the declaration.

# Structure Declaration (Example)

- Let's see the example to define a structure for an entity employee in c.

```
struct employee
{   int id;
    char name[20];
    float salary;
};
```

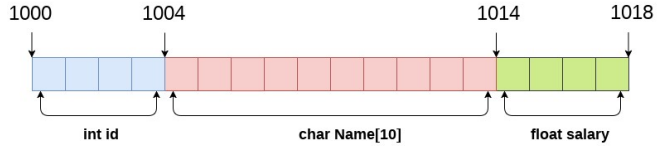







# Structure Declaration (Example)

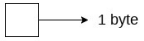
The following image shows the memory allocation of the structure employee that is defined in the above example.




```
struct Employee
{
    int id;
    char Name[10];
    float salary;
} emp;
```

sizeof (emp) = 4 + 10 + 4 = 18 bytes

where;
sizeof (int) = 4 byte
sizeof (char) = 1 byte
sizeof (float) = 4 byte








# Structure Declaration

- We can declare a variable for the structure so that we can access the member of the structure easily.
- There are two ways to declare structure variable:
  1. By struct keyword within main() function
  2. By declaring a variable at the time of defining the structure.





# Structure Declaration

- **1st way:**

Let's see the example to declare the structure variable by struct keyword. It should be declared within the main function.

```
struct employee
{   int id;
    char name[50];
    float salary;
};
```

Now write given code inside the main() function.

```
struct employee e1, e2;
```

*(The variables e1 and e2 can be used to access the values stored in the structure)*

# Structure Declaration

- **2nd way:**

Let's see another way to declare variable at the time of defining the structure.

```
struct employee
{   int id;
    char name[50];
    float salary;
}e1,e2;
```

Which approach is good

If number of variables are not fixed, use the 1st approach. It provides you the flexibility to declare the structure variable many times.

If no. of variables are fixed, use 2nd approach. It saves your code to declare a variable in main() function.





# Accessing members of the structure

- There are two ways to access structure members:
  1. By . (member or dot operator)
  2. By -> (structure pointer operator)





# C Structure example

```
#include<stdio.h>
#include <string.h>
struct employee
{  int id;
   char name[50];
}e1;  //declaring e1 variable for structure
int main( )
{
  //store first employee information
  e1.id=101;
  strcpy(e1.name, "Amit Srivs");//copying string into char array
  //printing first employee information
  printf( "employee 1 id : %d\n", e1.id);
  printf( "employee 1 name : %s\n", e1.name);
return 0;
}
```

**Output:**

employee 1 id : 101 employee 1 name : Amit Srivs





# C Structure example

```
#include<stdio.h>
#include <string.h>
struct employee
{  int id;
   char name[50];
   float salary;
}e1,e2;  //declaring e1 and e2 variables for structure

int main( )
{
  //store first employee information
  e1.id=101;
   strcpy(e1.name, "Sonoo Jaiswal");//copying string
into char array
  e1.salary=56000;

  //store second employee information
  e2.id=102;
  strcpy(e2.name, "James Bond");
   e2.salary=126000;

//printing first employee information
 printf( "employee 1 id : %d\n", e1.id);
 printf( "employee 1 name : %s\n", e1.name);

 printf( "employee 1 salary : %f\n", e1.salary);


 //printing second employee information
 printf( "employee 2 id : %d\n", e2.id);
 printf( "employee 2 name : %s\n", e2.name);

 printf( "employee 2 salary : %f\n", e2.salary);

return 0;   }
```

**Output:**

employee 1 id : 101
employee 1 name : Sonoo Jaiswal
employee 1 salary : 56000.000000
employee 2 id : 102
employee 2 name : James Bond
employee 2 salary : 126000.000000

# Copy Structures

- You can also assign one structure to another.
- In the following example, the values of s1 are copied to s2:

```
#include <stdio.h>

struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];
};

int main() {
  // Create a structure variable and assign values to it
  struct myStructure s1 = {13, 'B', "Some text"};
  // Create another structure variable
  struct myStructure s2;

  // Copy s1 values to s2
  s2 = s1;

  // Print values
  printf("%d %c %s", s2.myNum, s2.myLetter, s2.myString);

  return 0;
}
```

Output:
13 B Some text





# Modify Structures

- If you want to change/modify a value, you can use the dot syntax (.).
- And to modify a string value, the strcpy() function is useful again:

```
#include <stdio.h>
#include <string.h>

// Create a structure
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];
};

int main() {
  // Create a structure variable and assign values to it
  struct myStructure s1 = {13, 'B', "Some text"};
  // Modify values
  s1.myNum = 30;
  s1.myLetter = 'C';
  strcpy(s1.myString, "Something else");

  // Print values
  printf("%d %c %s", s1.myNum, s1.myLetter, s1.myString);

  return 0;
}
```

Output:
30 C Something else