5.4 NESTED STRUCTURES

C language allows us to insert one structure into another as a member. This process is called nesting and such structures are called nested structures. There are two ways in which we can nest one structure into another:

a). Embedded Structure Nesting

In this method, the structure being nested is also declared inside the parent structure.

Example

```
struct parent {
   int member1;
   struct member_str member2 {
      int member_str1;
      char member_str2;
      ...
   }
   ...
}
```

b). Separate Structure Nesting

In this method, two structures are declared separately and then the member structure is nested inside the parent structure.

Example

```
struct member_str {
   int member_str1;
   char member_str2;
   ...
}
struct parent {
   int member1;
   struct member_str member2;
   ...
}
```

One thing to note here is that the declaration of the structure should always be present before its definition as a structure member. For example, the **declaration below is invalid** as the struct mem is not defined when it is declared inside the parent structure.

```
struct parent {
   struct mem a;
};

struct mem {
   int var;
};
```

c) Accessing Nested Members

We can access nested Members by using the same (.) dot operator two times as shown:

str_parent.str_child.member;

Example of Structure Nesting

// C Program to illustrate structure nesting along with forward declaration

```
#include <stdio.h>
// child structure declaration
struct child {
    int x;
    char c;
};
// parent structure declaration
struct parent {
    int a;
    struct child b;
}
// driver code
int main()
{
    struct parent var1 = { 25, 195, 'A' };
    // accessing and printing nested members
    printf("var1.a = %d\n", var1.a);
    printf("var1.b.x = %d\n", var1.b.x);
    printf("var1.b.c = %c", var1.b.c);
    return 0;
}
```

Output

```
var1.a = 25

var1.b.x = 195

var1.b.c = A
```