otherway;

```
typedef struct employee
{
    char name[30];
    int age;
    float bs;
} EMP;
```

→ create one name

```
EMP e1,e2;
```

→ use that in main.

Array of structures

If the no:of books increase, create an array of structures.

```
#include <stdio.h>
struct book
{
    char name;
    float price;
    int pages;
};
struct book b[10];
void main()
{
    int i;
    int dh;
    for(i=0; i<=9; i++)
    {
        Printf("Enter name, price & pages");
        scanf("%c%f%d", &b[i].name,
              &b[i].price, &b[i].pages);
```

```
        while ((dh = getchar()) != '\n');
    }
    for (i=0; i<=9; i++)
        printf("%c %f %d\n", b[i].name,
                b[i].price, b[i].pages);
    return 0;
}
```

↳ provides memory for 10 structures of type book.

thus by 1 array we can take care of many books, each having many data items.

The enter that we hit remains in the keyboard buffer. If we leave it, the next call to scanf() would take this enter and move ahead.

to prevent this we have to flush out the keyboard buffer.

[fflush(stdin)] can also be used.

Nested Structures

one structure can be nested within another structure. using this facility complex data type can be created.

1st way:-

ex)
```
struct outer
{
    int out1;
    int out2;
```

```
struct inner
{
    int in1;
    int in2;
}invar;
};
void main()
{
    struct outer outvar;
    outvar.out1 = 2;
    outvar.out2 = 5;
    outvar.invar.in1 = 44;
    outvar.invar.in2 = outvar.out2 + 34;
    Printf(" out1 = %d   out2 = %d
           in1 = %d   in2 = %d",
           outvar.out1, outvar.out2,
           outvar.invar.in1, outvar.invar.in2
}
```

O/P

out 1 = 2 out 2 = 5

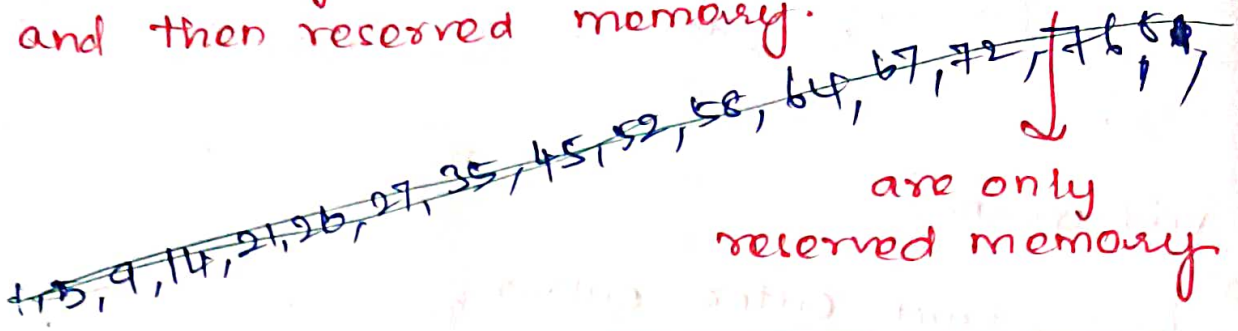
in 1 = 44 in 2 = 39

2nd way:-

```
#include <stdio.h>
struct address
{
    char phone[15];
    char city[15];
    int pin;
};
```

members don't have any memory space reserved.

They are associated with Str. variables and then reserved memory.




```
Struct emp
{
    char name[25];
    struct address a;
};
Void main()
{
    Struct emp e = {"jeru", "2531046", "nagpur",
                                        10};
    Printf ("name = %s, phone = %s \n",
                    e.name, e.a.phone);
    Printf ("city = %s  pin = %d \n",
                    e.a.city, e.a.pin);
}
```

o/p

name = jeru phone = 2531046

city = nagpur pin = 10

bunch-8

Nested structures can be surprisingly self-descriptive.

maruti.engine.bolt.large.qty.

↓

we are referring to the quantity of large sized bolts that fit on an engine of a maruti car.

Rules for initializing structures:-

1. we cannot initialize individual members inside the structure template.
2. The order of values must match the order of members in the structure.
3. partial initialization is permitted. uninitialized members should be only at the end of the list.
4. def values zero for int, float
   '\0' for char, strings.

Arrays within structure

→ use array as structure member.

ex: struct marks
{
int number;
float subject[3];
} student[2];

→ subject[0]
subject[1]
subject[2]

student[1].subject[2] → 3rd sub mark of 2nd student

ex) #include<stdio.h>

```
struct marks
{
    int sub[3];
    int total;
};
struct marks student[3] = { 45,67,81,0,75,53,69
                            57,36,71,0 };
void main()
{
    struct marks total = {0};
    int i,j;
    for(i=0; i<=2; i++)
    {
        for(j=0; j<=2; j++)
        {
            student[i].total += student[i].sub[j];
            total[i].sub[j] += student[i].sub[j];
        }
        total.total += student[i].total;
    }
    Printf("student total\n");
    for(i=0; i<=2; i++)
        printf("student[%d] %d\n",
                i+1, student[i].total);
    Printf("subject total\n");
    for(j=0; j<=2; j++)
```

total.sub[0]
sub[1]
sub[2]

```
Printf ("subject - %d  %d\n",
                j+1, total.sub[j]);
Printf ("\nGrand total = %d\n",
                total.total);
}
```

O/P:-

Student	total
Student[1]	193
Student[2]	197
Student[3]	164

i=0 stud[0].total → 45 + 67 + 81 = 193

i=1 stud[1].total → 75 + 53 + 69 = 197

i=2 stud[2].total → 57 + 36 + 71 = 164

Subject	total
~~Student~~-1 Subject	177
Subject-2	156
Subject-3	221

Grand total = 554.

	j=0	j=1	j=2
i=0 total.sub[0]	45	67	81
	+	+	+
i=1 total.sub[1]	75	53	69
	+	+	+
i=2 total.sub[2]	57	36	71
	177	156	221

(structure pointer)

Pointers to a structure

To access the elements of a structure with pointer, we use indirection operator (→).
(or) structure dereference operator.

ex):-

```
#include <stdio.h>
struct person
{
```

```
    char name[20];
    int age;
    float weight;
};
void main()
{
    struct person person1, *strptr;
    strcpy (person1.name = "meena");
    person1.age = 40;
    person1.weight = 60;
    strptr = &person1;
    Printf("Name: %s\n", strptr → na
    ("Age: %d\n", strptr → age);
    ("weight: %f", strptr → weigh
}
```

Passing structure elements to functions

we may either pass individual structu elements / entire structure variable to a fun

```
#include <stdio.h>
struct book
{
    char name[25];
    char author[25];
    int pages;
};
```