6. STRUCTURES, UNION & FILES

* structures:-

→ structures are 'user defined functions'.

→ We need to use 'struct' keyword.

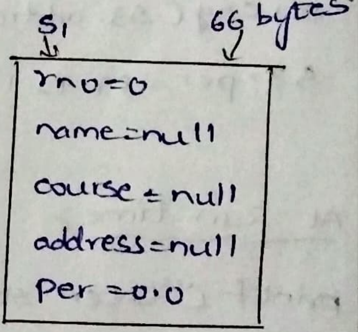
syntax:-

```
struct <structure name>
{
    data type  var1;
    data type  var2;
        |          |
        |          |
        |          |
};
```

→ Initially, it makes a template only after giving variable then memory will be created.

Example:-

```
struct student
{
    int rno;
    char name[20];     ← 66 bytes
    char course[20];
    char address[20];
    float per;                 main()
} s1, s2, s3;      (or)       {
                               struct student s1, s2, s3;
                               }
```



→ We can use structure globally or locally.

* typedef:-

typedef student s;

* **Initialization :-**

① At compile time
② Individually
③ At Run time.

① **At Compile time :-**

struct student s2 = { 102, "PPP", "java", "N42", 97.8 };

```
r.no = 102
name = ppp
course = java
address = N42
Per  = 97.8
```

s2 - variable

66 bytes

② **Individually :-**

```
struct student s3;
s3. r.no = 103;
strcpy (s3.name, "PPP");
strcpy (s3.course, "cpp");
strcpy (s3. address, "zzz");
s3. per = 89.9;
```

③ **At Run time :-**

```
printf ("Enter student r.no, name, course, ______");
scanf ("%d, %s, %s, %s, %f", &s3.rno, s3.name, &s3.course
                                    &s3.address, &s3.per).
```

(or)

printf ("student ~~debails~~ details");

```
printf ("sno=%d", s1.rno);
printf ("s name = %s", s1.name);
```

______

______

______

______

______

* For multiple structures:

Ex: For 5 students

```
int i;
for(i=0; i<5; i++)
{
  printf("%d- student r.no, name, course, address, per", i+1);
  scanf("%d %s %s %s %f", &s[i].rno, &s[i].name,
           &s[i].course, &s[i].address, &s[i].per);
}
for (i=0; i<5; i++)
{
  printf("%d student details \n", i+1);
  printf("stu.r.no=%d", s[i].rno);
  printf("stu name = %s", s[i].name);
         ______
         ______
         ______
         ______
}
```

* Pointers with ~~Addr~~ structures:

```
typedef struct student
{
   int rno;
   char name [20];
   char course [20];
   float per;
};
struct student s1;
struct student *sp;
   sp = &s1;
 s1.rno=101
 sp->rno=101;
 gets(sp->name);
 gets(sp->course);
  sp->per=75.5;
 strcpy (sp->name, "veda");
```

* Functions with structures:

① Passing members of structures.

② Passing Entire structure to function.

③ Passing Address of structures.

① Passing members of structures:

```
typedef
struct student
 {
    int r_no;
    char name[20];
    char course[20];
    float per;
 };

void main()
{
  student s1={101, "veda", "c", 98.5};
  display(s1.rno, s1.per);
  display2(s1);
}

void display(int no, float p):
{
   printf("rno=%d", no);
   printf("per =%f", p);
}
```

② Entire structure:

```
void display2(student s)
{
   printf("rno=%d", s.rno);
   printf("name=%s", s.name);
   printf("course=%s", s.course);
   printf("per=%f", s.per);
}
```

```
Void main()
{
  student s1, s2, r;
   r=Reading (s1);
   display (r);
}
student Reading (student s)
{
  scanf ("%d, %s, %s, %f", &s.rno, &s.name, &s.course, &s.pe
  return s;
}
```

③ Passing address of structure :

```
void main( )
{
   student s1, s2, s3;
   display3 (&s3);

}
void display3(student *sp)
{
  printf("r_no=%d", *(sp->r_no);
              :
              :
}
```

* Nested structures:-

```
struct student
{
   int rno;
   struct DOJ
   {
     int d, m,y;
   }
};
```

(or)

```
struct DOJ
{
  int d, m, y;
};

struct Address
{
  char city[20];
  char street[20];
  int pin;
};

struct student
{
   int rno;
   DOJ doj;
   Address ad;
};
s1.rno=101
s1.doj.d=5;
s1.doj.m=1;
gets(s1.ad.street);
```