CS & IT ENGINEERING

C Programming

Structure and Union

(In One Shot)

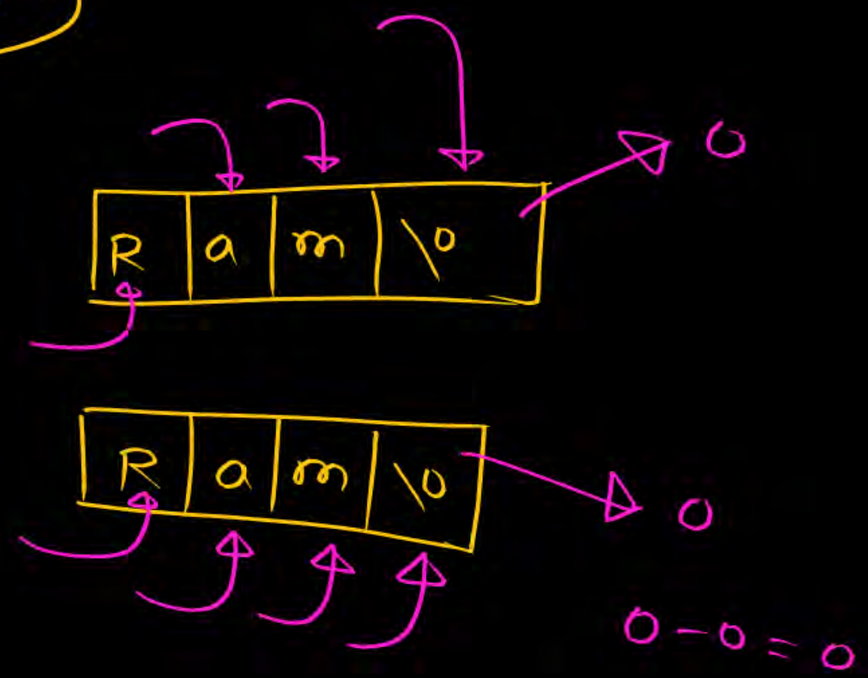
By- Pankaj Sharma Sir

- Structure and Union

strcmp(string1, string2)

strcmp("Ram", "Ram") → 0

```
char *ptr1 = "Ram";
char *ptr2 = "Ram";
```



```
char *ptr1 = "Pankaj";
char *ptr2 = "Pani";
```

p	a	n	k	a	j	\0

p	a	n	i	\0

strcmp(ptr2, ptr1)

'i' - 'k'

⇒ -ve (<0)

strcm(ptr1, ptr2)

'k' - 'i'

= +ve >0

$$\begin{bmatrix} 0 \\ <0 \\ >0 \end{bmatrix}$$

strcmp("Ram", "RaM")

'm' - 'M'

⇒ +ve

Sorting ⇒ <, >, <=, >=

$$\left\{ \begin{array}{l} 10<20 \\ 30>20 \\ 10<=20 \end{array} \right\}$$

Structure

student

name → char arr[20] → derived

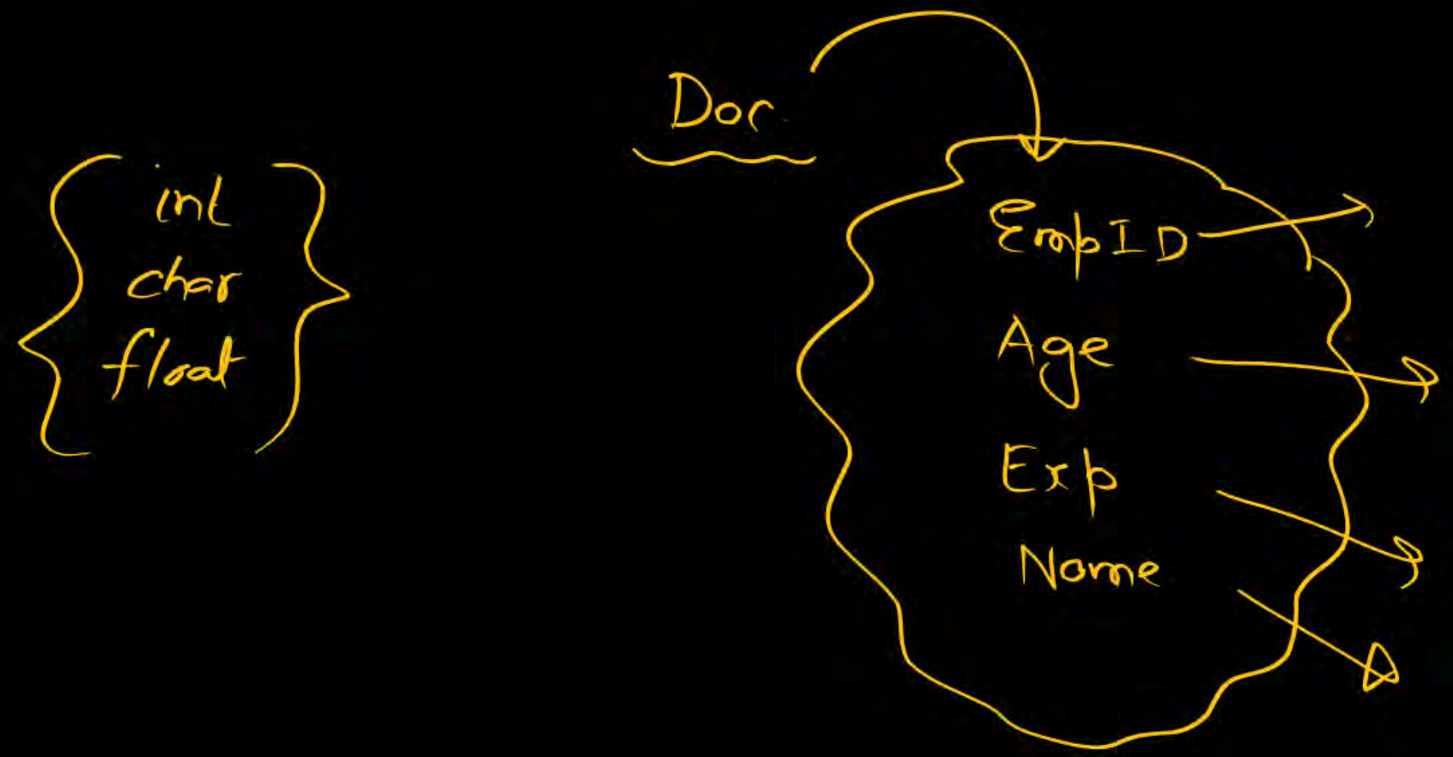
Roll → int Roll → primitive

Address → char Add[40];

Patient

Doc

* Collection of heterogenous type/diff type of data elements.
* User defined data type



Tag of structure

```
struct student {
    char name[20];
    int Roll;
};
```

Struct is the keyword used to create user defined data type

```
struct student {
        char name[20];
        int Roll;
};
```

info:

just like int, char, float
from now onwards a
new data type exist
⇒ struct student

```
void main() {

}
```

```
struct student {
        char name[20];
        int Roll;
};
```

members of structure

No memory is allocated

template/blueprint

```
void main() {
    ___
    ___
    ___
    ___

}
```

```
struct student {
        char name[20],
        int Roll;
    };

struct student {
        char name[20] = "Pankaj";
        int Roll = 10;
    };
```

Ud fe laat padegi
Kaun marega

a) Tiger
b) Vidyut Jamwal
c) Akshay
d) None

```
void main(){
    int ;
    =
}
```

$\Longrightarrow$

```
void main(){
    int a;
    =
}
```

```
struct student {
        char name[20];
        int Roll;
        };
void main(){
    struct student ;
        ≡
        }
```

⟹

```
struct student {
        char name[20];
        int Roll;
        }
void main(){
    struct student s;
        ≡
        }
```

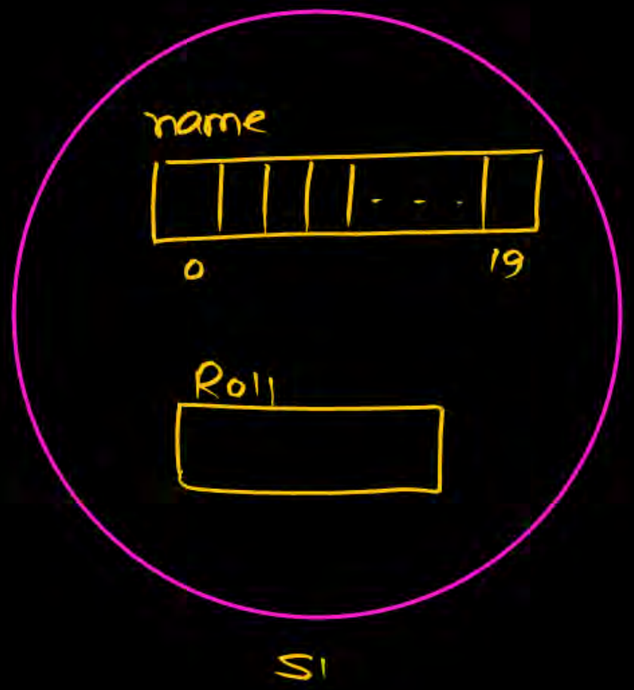
```
struct student {
        char name[20];
        int Roll;
        };
```

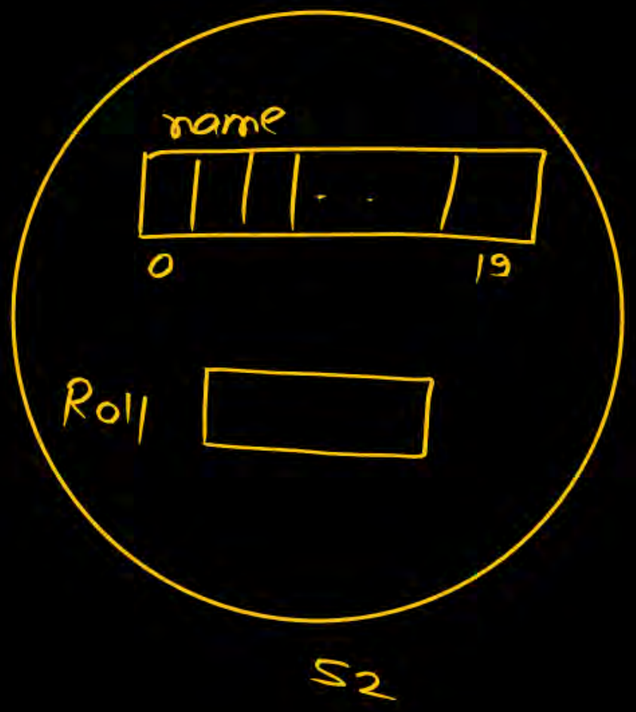
global

```
void main(){
        struct student s;
}
```

```
void f(){
        struct student s1;
}

void g(){
        struct student s2;
}
```

```
struct student {
        char name[20];
        int Roll;
        };
void main() {
struct student s1, s2;
printf("%s", name);
        s1.Roll = 10;
```

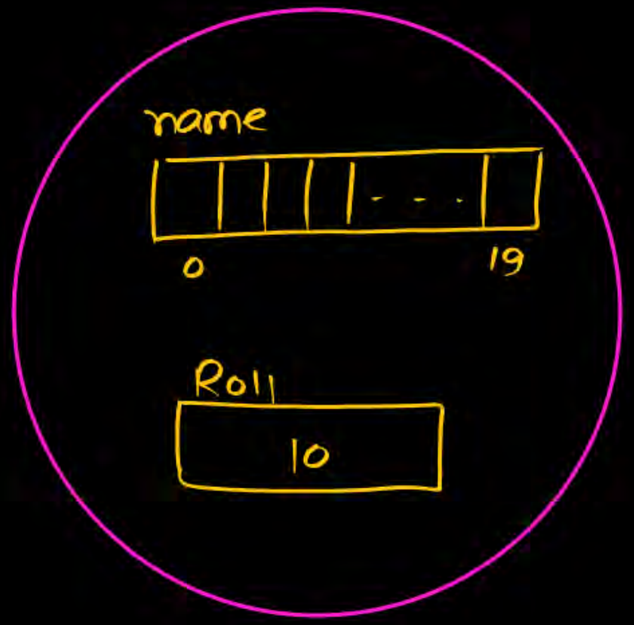




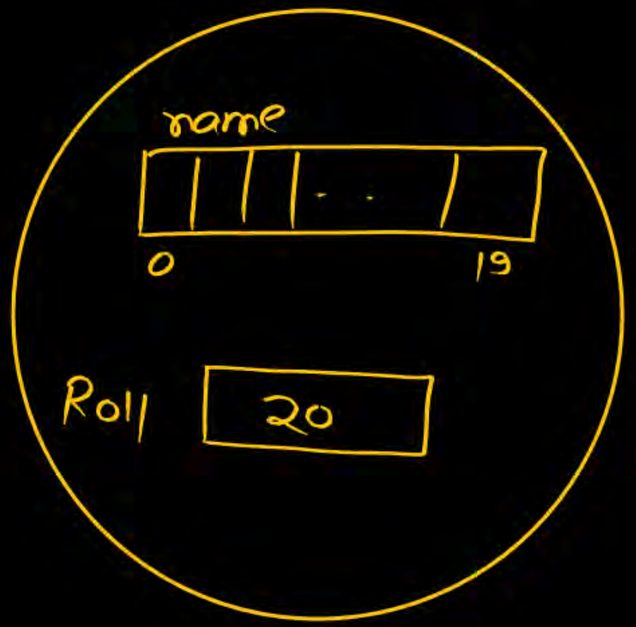
```
struct student {
        char name[20];
        int Roll;
        };
void main() {
struct student s1, s2;
printf("%s", name);
```

```
s1.Roll = 10;
s2.Roll = 20;
```

. ⇒ membership operator



s1



s2

① No Tag

```
struct {
    char name[20];
    int Roll;
} s1,s2 ;
void main(){
```

we can't create a variable

```
}
```

②

```
struct student {
    char name[20];
    int Roll;
};
void f(){
    struct student s1;
    =
}
void main(){
    struct student x;
    =
}
```

③

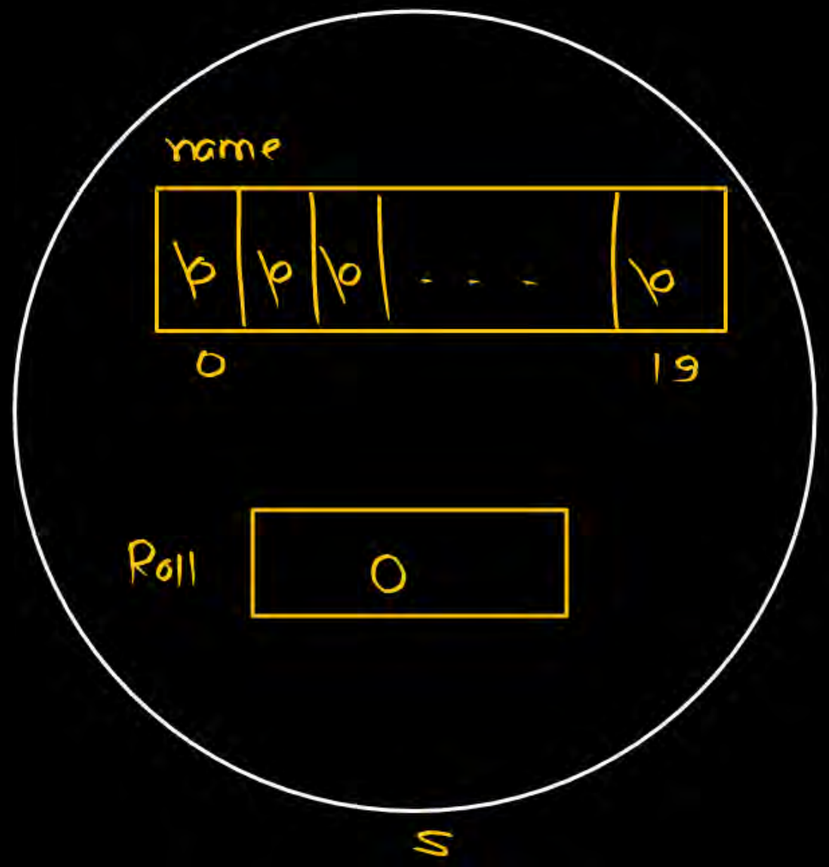
```
typedef struct student{
    char name[20];
    int Roll;
} x ;

void main(){
    x s1;
```

or

```
struct student s1;
    =
}
```

```
struct student{
    char name[20];
    int Roll;
    };
void main(){
    struct student s;
    pf("%d",s.Roll);
    pf("%s",s.name);
}
```

default value
↳ 0



int R = "Pankaj"; ✗

Invalid

```
struct student {
    char name[20];
    int Roll;
};
```

array

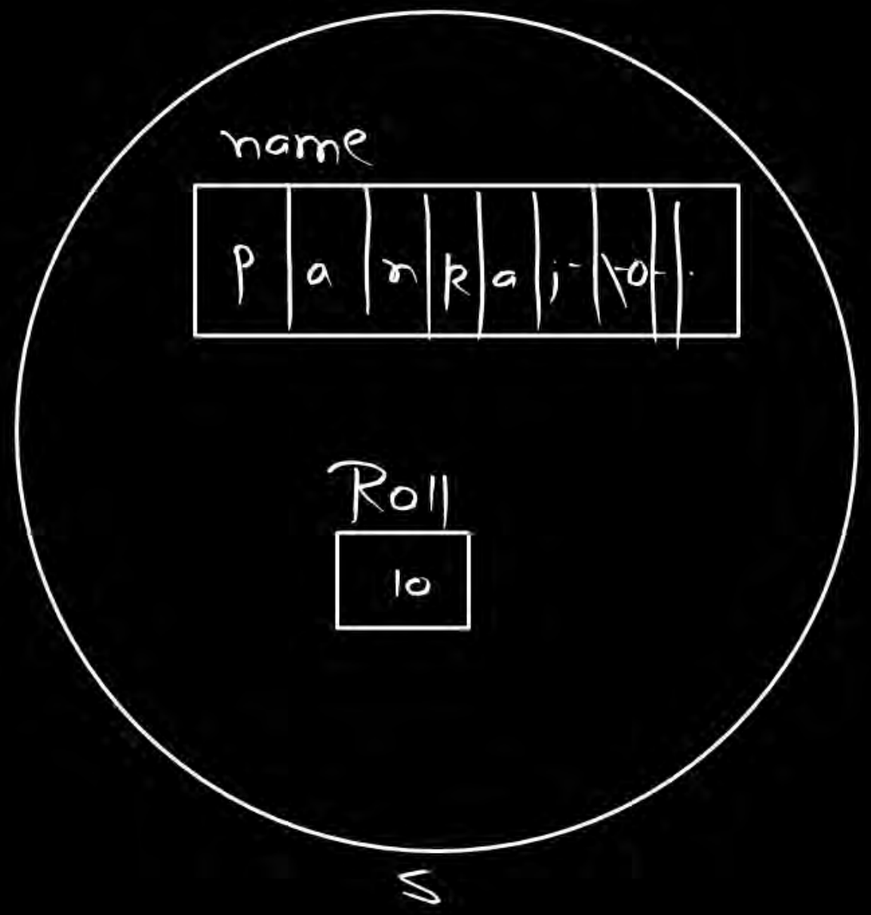
```
void main() {
struct student s = { 10, "Pankaj" };
```

```
{s.name, s.Roll} = {10, "Pankaj"};
```

```
struct student {
    char name[20];
    int Roll;
};
void main() {
struct student s = {"Pankaj", 10};
```

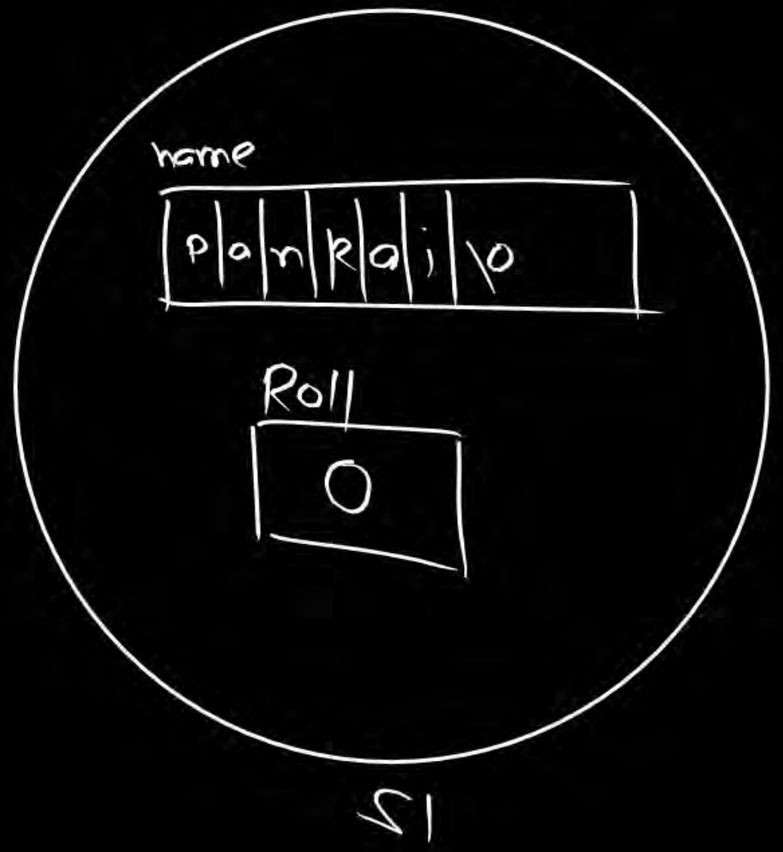
array

```
struct student s1 = {"Pankaj"};
```



```
struct student {
    char name[20];
    int Roll;
};
void main() {
struct student s = {"Pankaj", 10};

struct student s1 = {"Pankaj"};
```

array

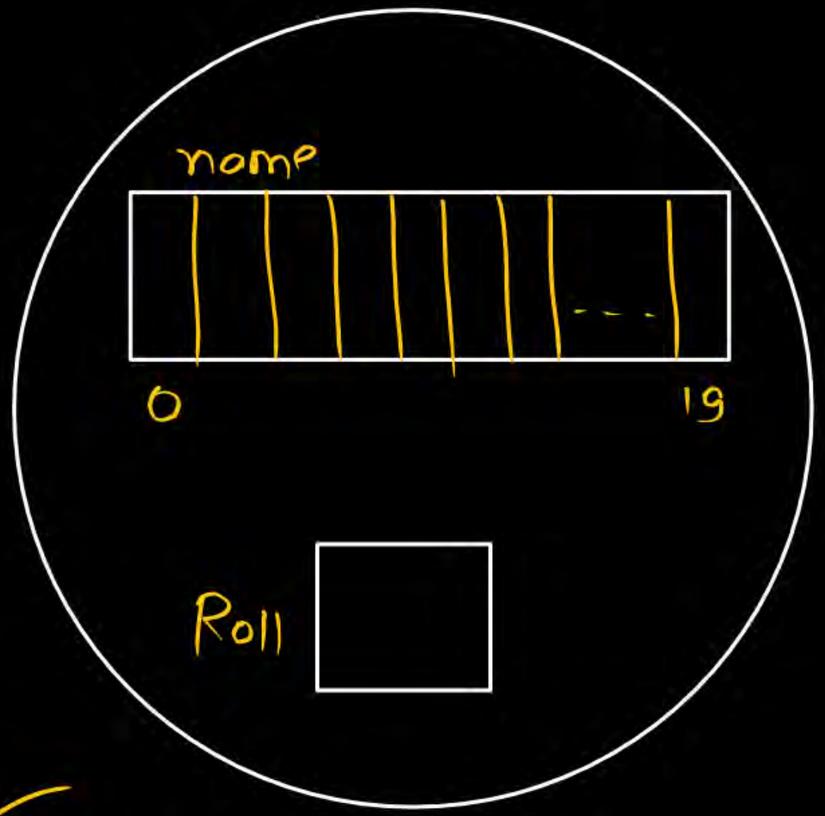


```
struct student {
        char name[20];
        int Roll;
        }

void main() {
    struct student s;
    s.name = "Pankaj" ;
```

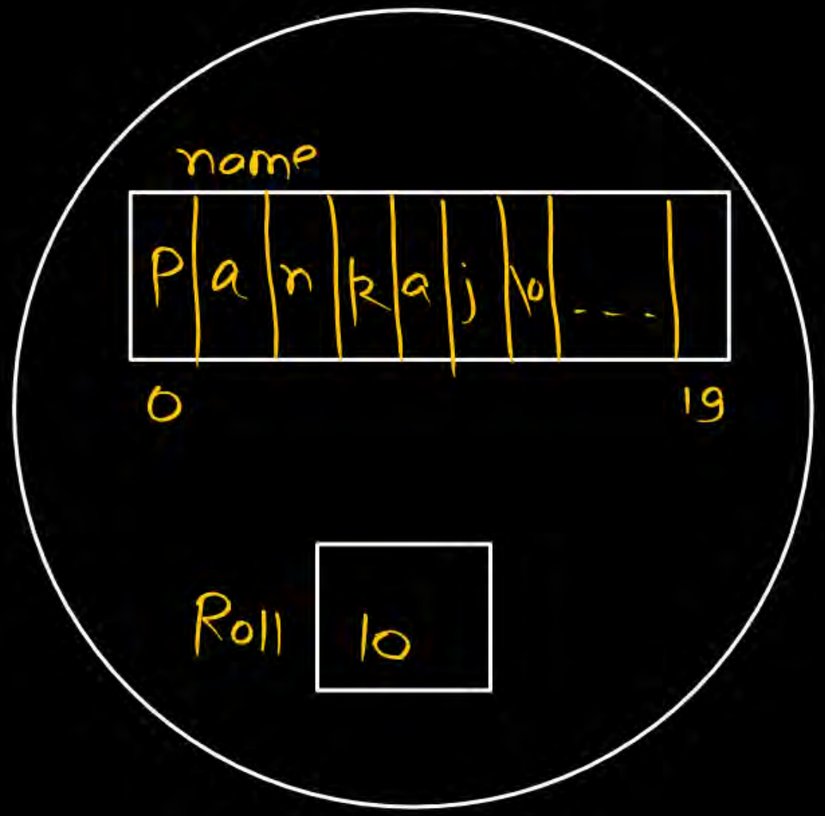
array name can't be Lvalue

valid ✗

Invalid ✓



```
struct student {
        char name[20];
            int Roll;
                }

void main() {
    struct student s;
    strcpy(s.name, "Pankaj");
    s.Roll = 10;
```



int a,b; ⟹ int a[2]

struct student s1,s2; ⟹ struct student s[2]; → s[0], s[1]

Array of structure

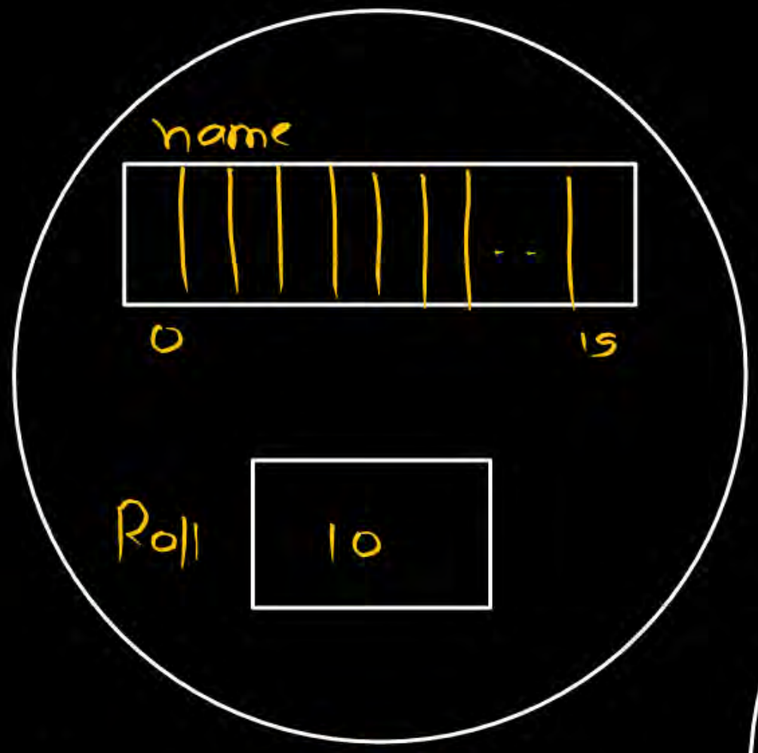
```
struct student {
        char name[20];
        int Roll;
        };
void main() {
    struct student s[2];
      s[0].Roll = 10;
      s[1].Roll = 2;
```

```
s[0].name = "Pankaj";
s[1].name = "Neeraj";
```

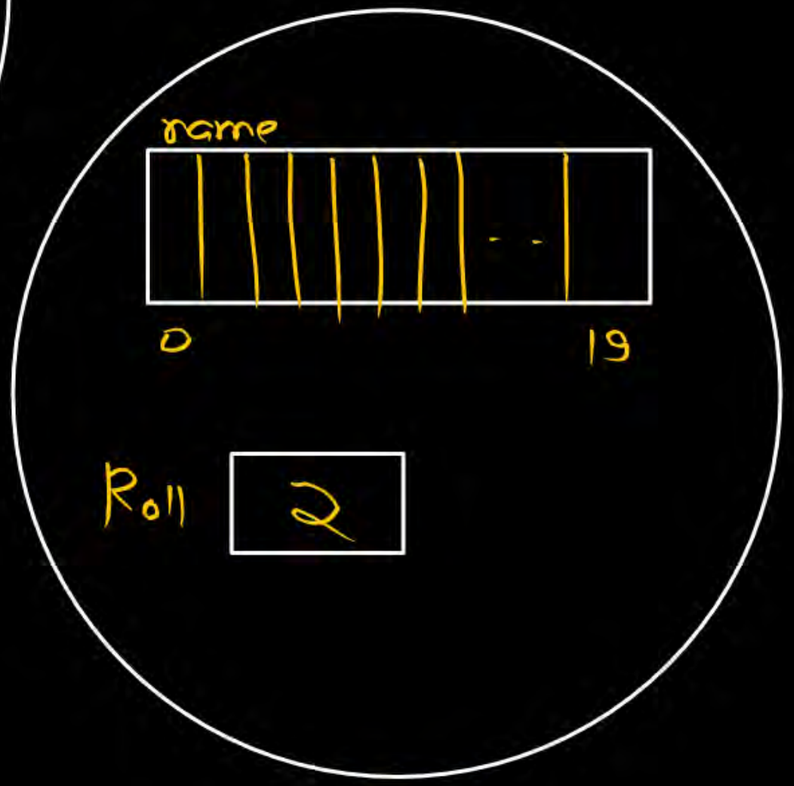
Ud ke load Padegi → a, b, c, d

⇓ solution

```
strcpy(s[0].name, "Pankaj");
strcpy(s[1].name, "Neeraj");
```

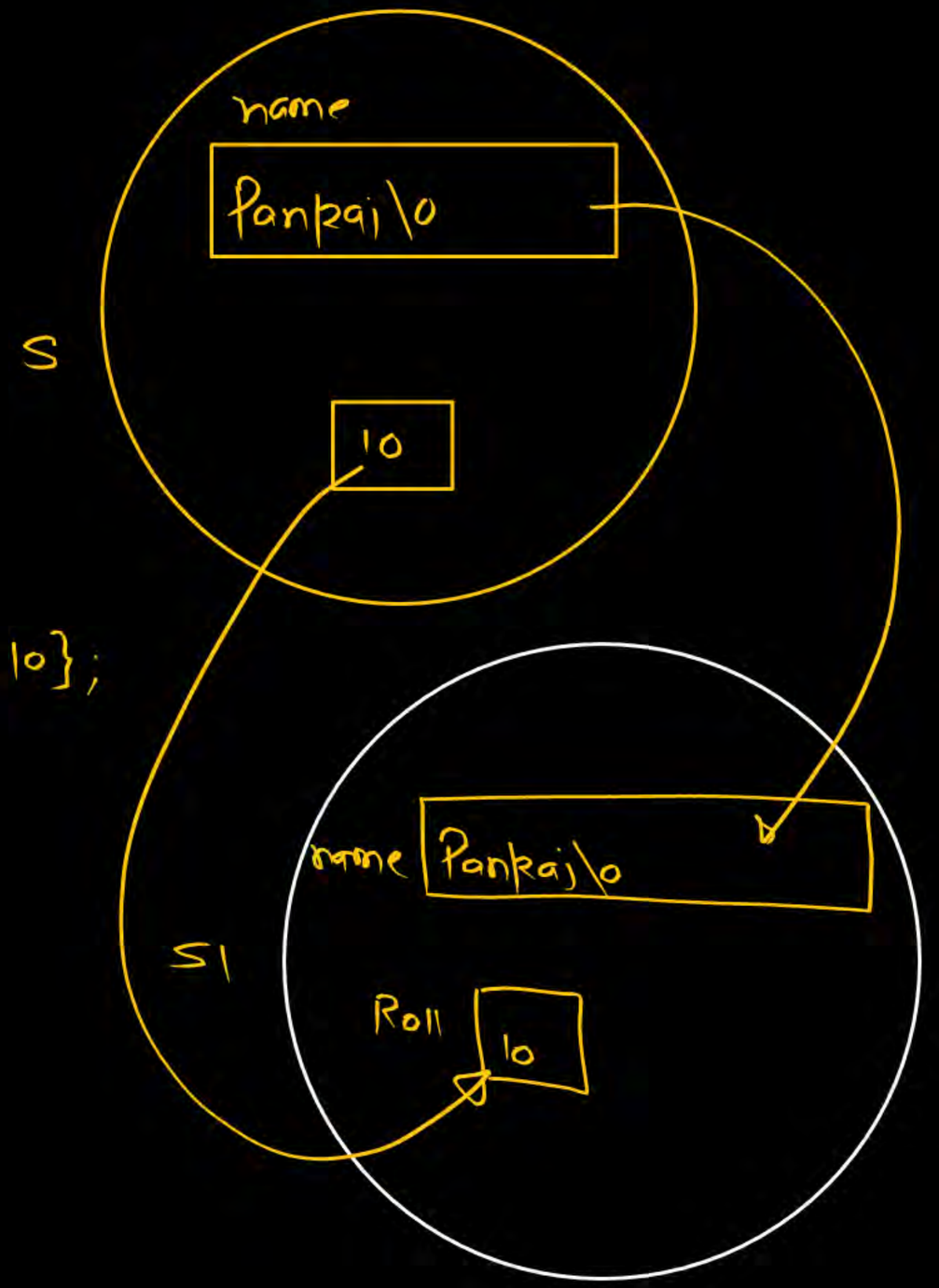


s[0]



s[1]

```
struct student{
        char name[20];
        int Roll;
        };
void main(){
        struct student s = {"Pankaj", 10};
✓       struct student s1 = s;
```



Student → name
Student → Roll
Student → DOB → 02|03|1982

day month year

```
struct date_of_birth{
    int day;
    int month;
    int year;
};
```

```
struct date_of_birth{
        int day;
        int month;
        int year;
            };
```

→ template

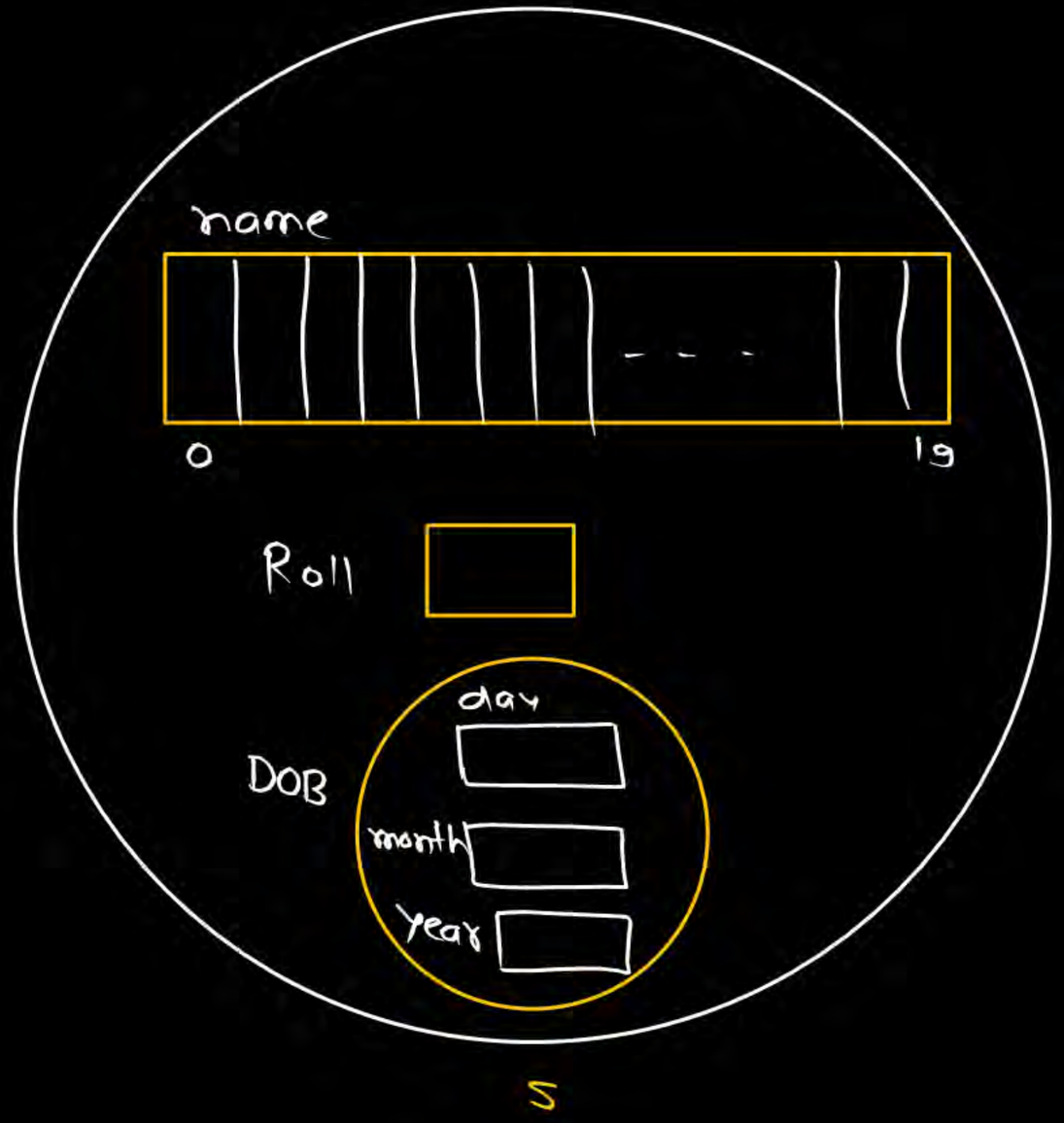
```
struct student{
        char name[20];
        int Roll;
  struct date_of_birth   DOB;
        }
```

→ derived

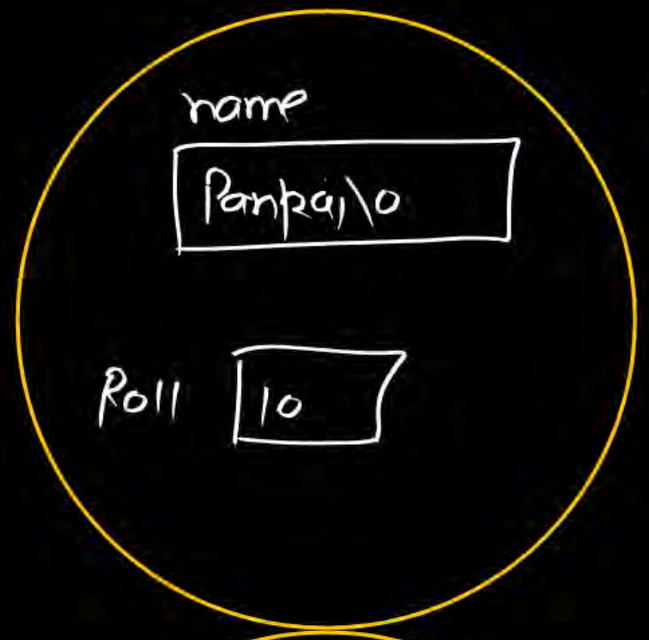
→ primitive

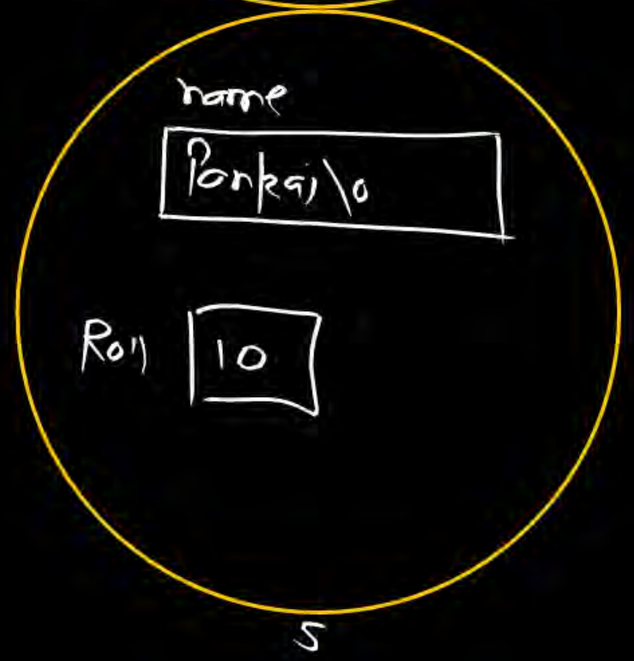
→ user defined

```
int month;
int year;
};

struct student{
    char name[20];   → 1st member
    int Roll;        → 2nd member
    struct date_of_birth DOB;   → 3rd member
}

void main(){
    struct student s;
    strcpy(s.name, "Pankaj");
    s.Roll = 10;
    s.DOB
        s.DOB.day = 2;
        s.DOB.month = 3;
        s.DOB.year = 1982;
```



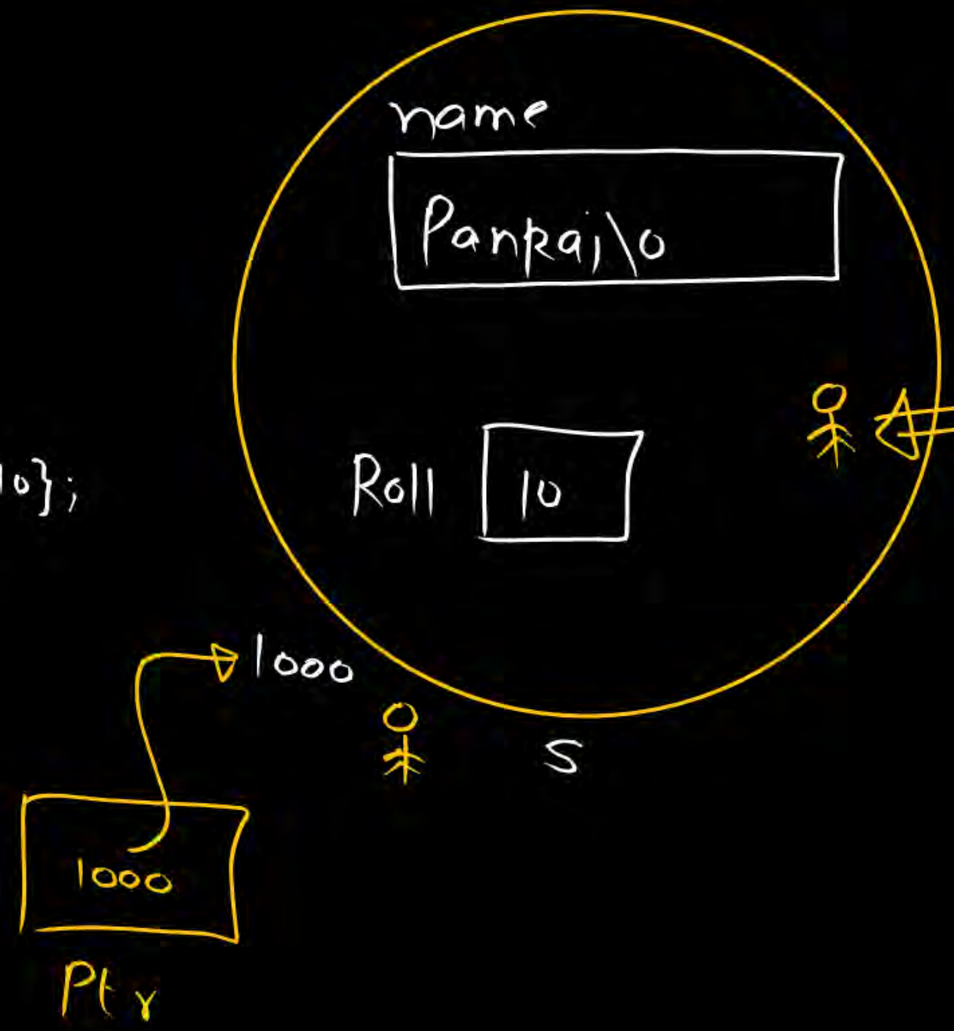
```
void display( struct student)
struct student{
        char name[20];
        int Roll;
            };
void main() {
struct student s;
strcpy(s.name, "Pankaj");
s.Roll = 10;
display(s);
}
```





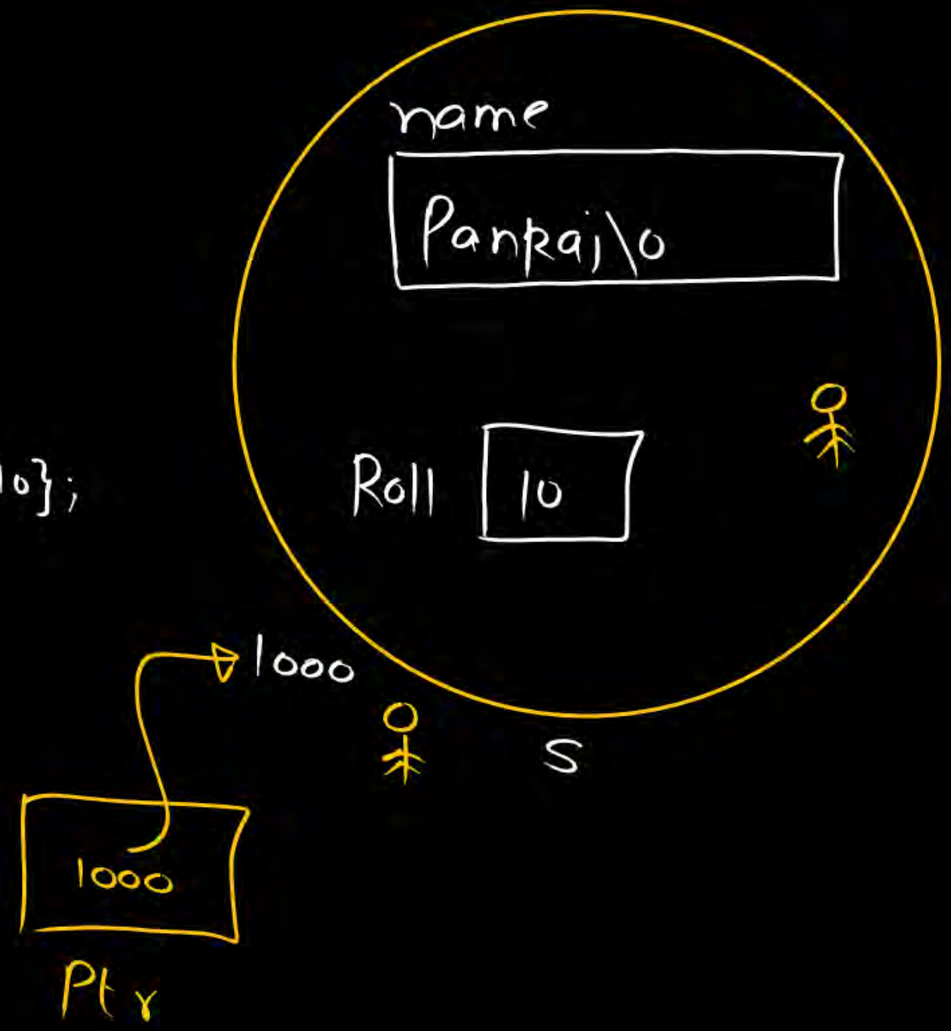

```
void display( struct student t)
{
pf("%s", t.name);
pf("%d", t.Roll);
}
```

```
struct student{
    char name[20];
    int Roll; };
void main()
    {
struct student s = {"Pankaj", 10};
                  1000
    display(&s);
    }
```




```
void display(struct student *Ptr)
{
(*Ptr)
}
```

```
struct student{
    char name[20];
    int Roll; };
void main()
    {
struct student s = {"Pankaj",10};
                1000
    display(&s);
    }
```




```
void display(struct student *Ptr)
    {
pf("%s",(*Ptr).name);
pf("%d",(*Ptr).Roll);
    }
```

$(*Ptr).member1$ ⇒ Ptr is a pointer to structure
& we can access member1 like this

Ptr : pointer to structure

pointer to structure

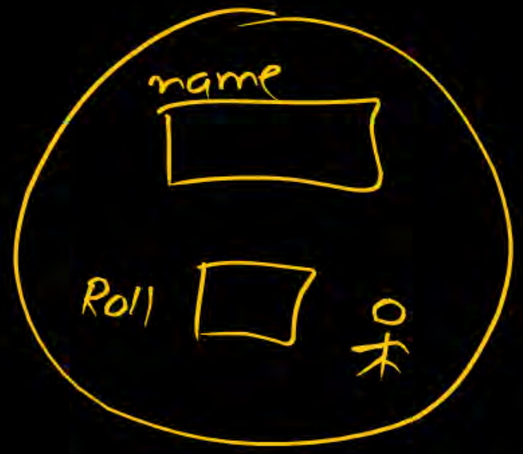
Ptr → member1 ✓

Ptr → member2 ✓

$(*Ptr).member1$ or $Ptr \rightarrow member1$

$(*Ptr).member2$ or $Ptr \rightarrow member2$

```
struct student s;
```



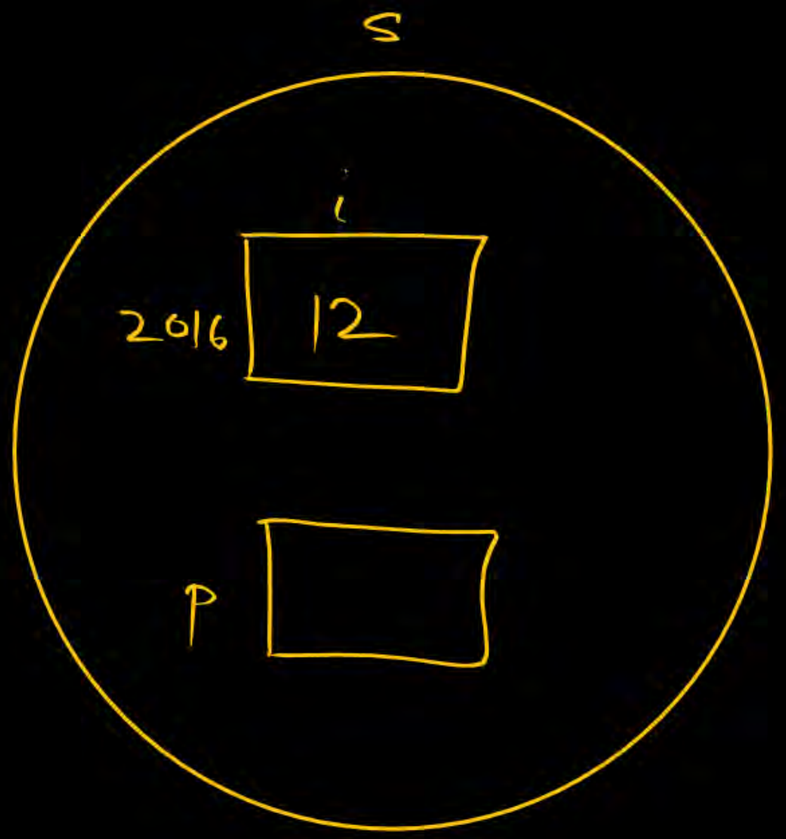
s → group of values

s.name

s.Roll

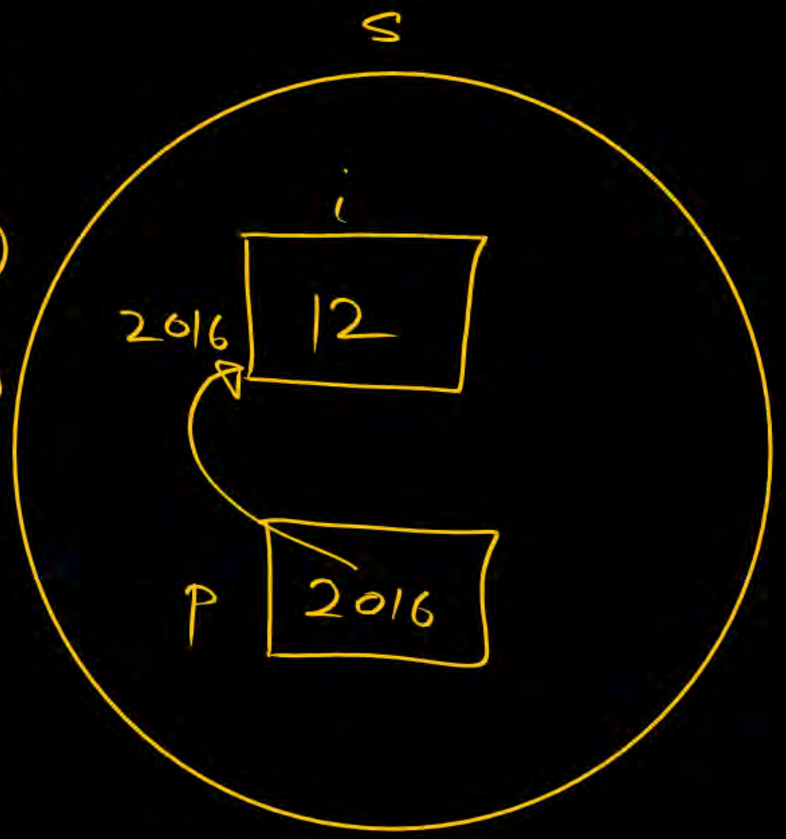
```
struct my_struct {
        int i;
        int *p;
        };

void main() {
    struct my_struct s;
        s.i = 12;
        s.p = 
```

किसी integer var. का address



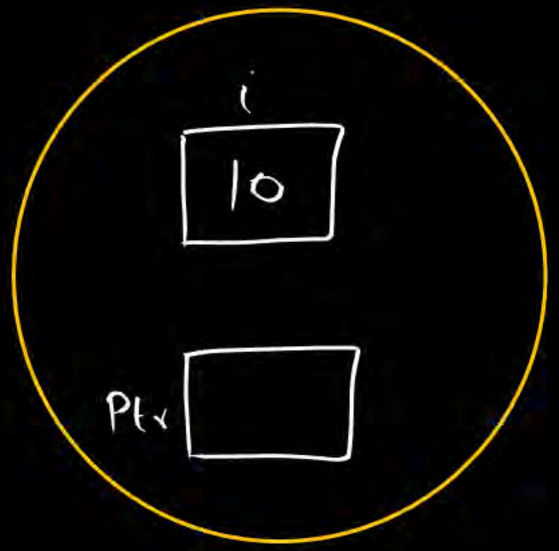
```
struct my_struct {
    int i;
    int *p;
};

void main() {
    struct my_struct s;
    s.i = 12;
    s.p = &s.i;
```

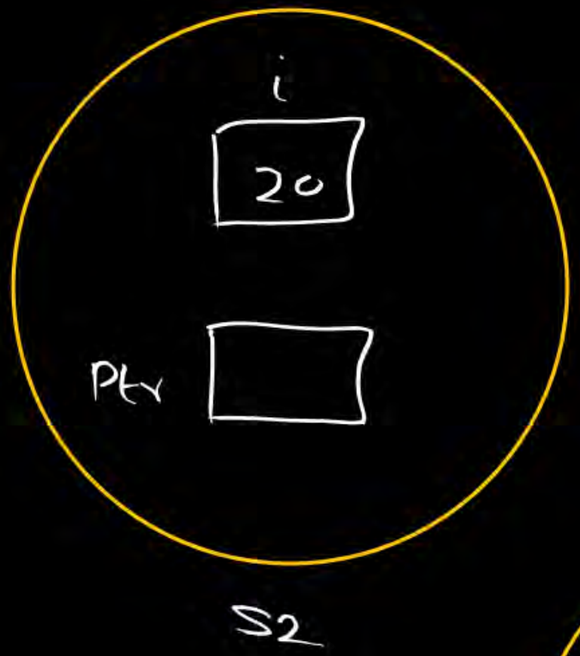
member be a pointer

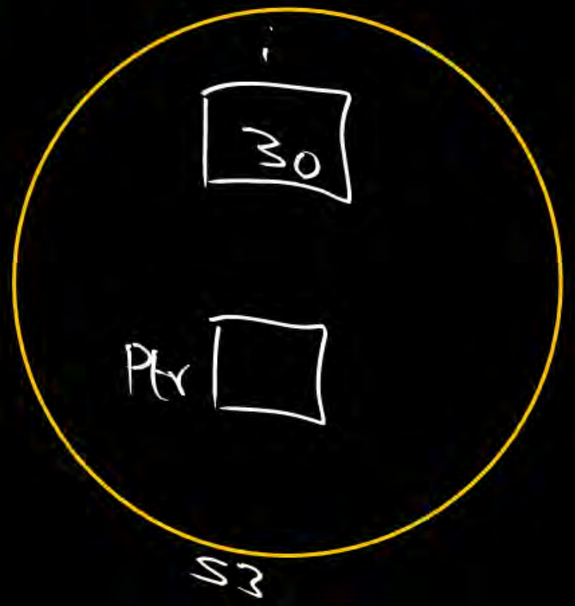


Self Refrential structure

```
struct Pankaj {
    int i;
    struct Pankaj *Ptr;
};

void main() {
struct Pankaj S1, S2, S3;
    S1.i = 10;
    S2.i = 20;
    S3.i = 30;
```

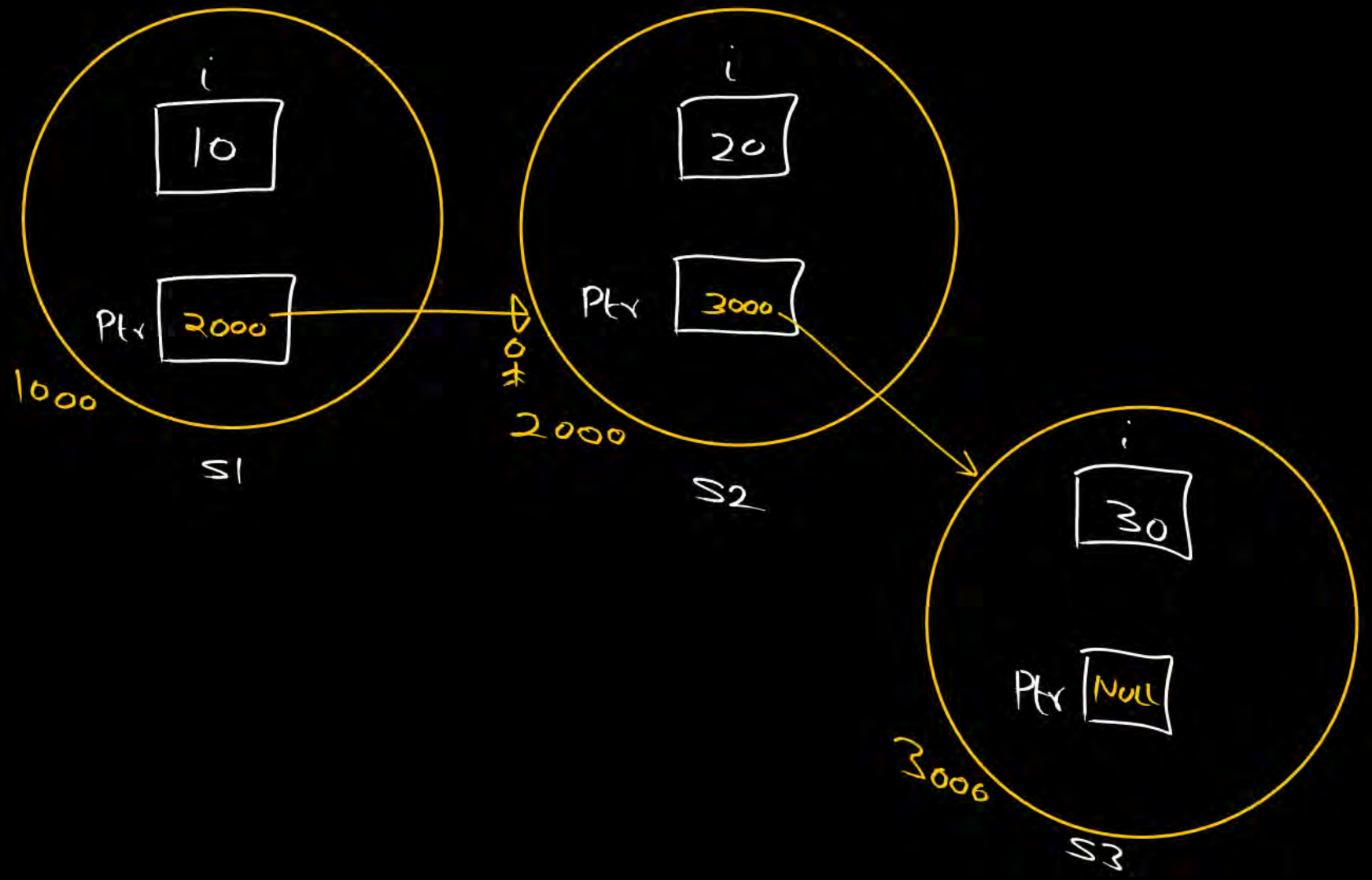






Self Refrential structure

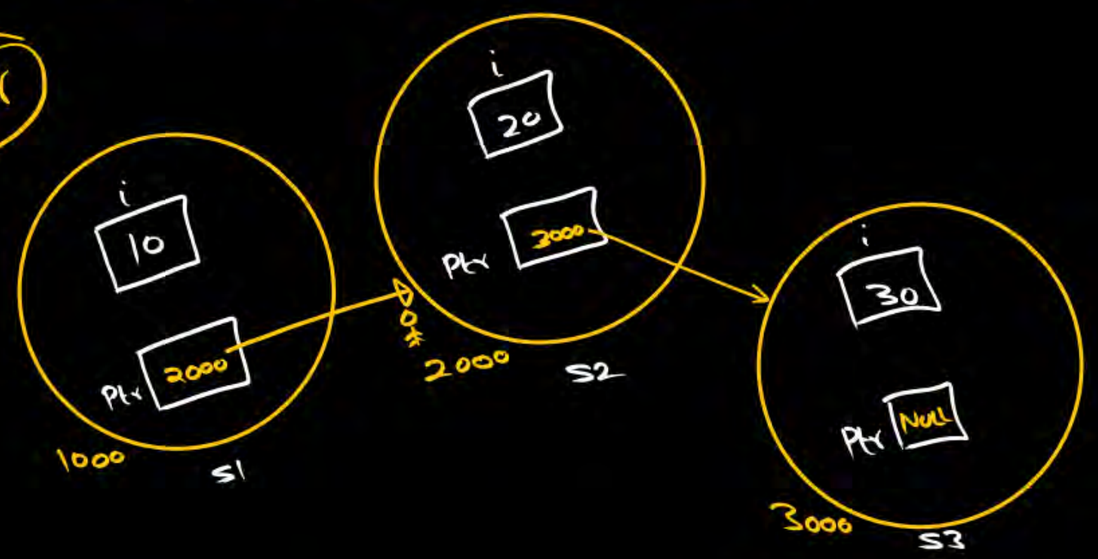
```
struct Pankaj {
        int i ;
struct Pankaj *Ptr ;
        };

    void main() {
struct Pankaj S1,S2,S3;
        S1.i = 10;
        S2.i = 20;
        S3.i = 30;
        S1.Ptr = &S2;
        S2.Ptr = &S3;
        S3.Ptr = NULL;
```



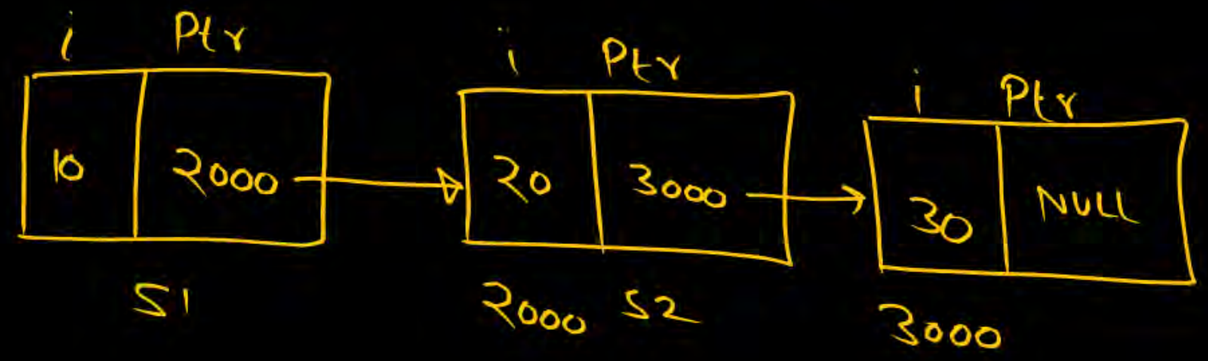
Self Refrential structure

```
struct Pankaj {
    int i;
    struct Pankaj *Ptr;
};
```

member




```
void main() {
struct Pankaj S1, S2, S3;
    S1.i = 10;
    S2.i = 20;
    S3.i = 30;
    S1.Ptr = &S2;
    S2.Ptr = &S3;
    S3.Ptr = NULL;
```



09:00 PM

union
scoping
sizeof
comma


THANK YOU GW
SOLDIERS !