CS & IT ENGINEERING

STRUCTURE AND UNION

Lecture No.16

By- Pankaj Sharma SIR

Structure

* Collection of heterogenous type of data element
* User defined data type

{ int
float
char }

docter :

name : string
Age : float
Empid :

How to create structure data type

* struct is the keyword

Tag of structure

keyword

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

members

int, char, float

struct student

Blueprint / template

```
int ;
char ;
float ;
```

```
int = 10;
```

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

No memory is allocation

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

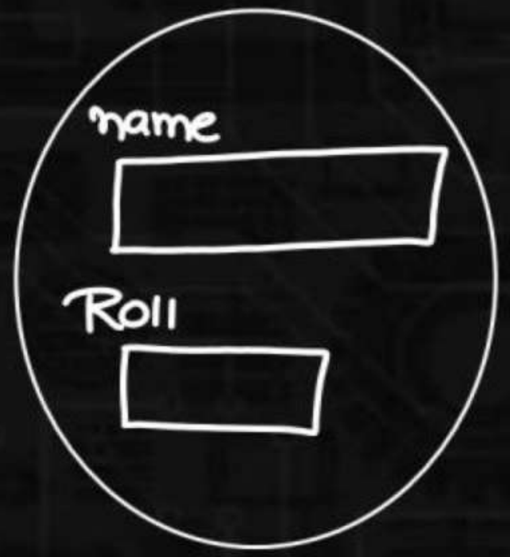
Error

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

Global

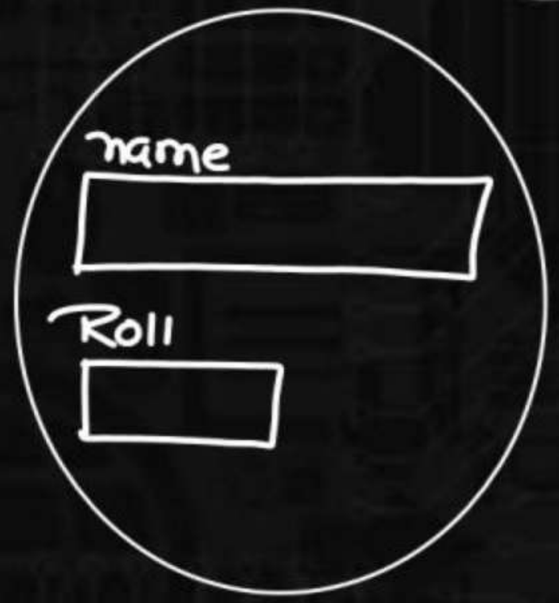
```
void main(){
    struct student s1, s2;
```

membership operator



s1

{Group of 2 elements}



s2

{Group of 2 elements}

```
printf(s1);   ✗
printf("%s", name)   ✗
```

```
printf("%s", s1.name);
```

3 ways

①

```
struct {
    char name[20];
    int Roll;
    } S1,S2;
```

* No Tag

```
void main(){
we can't
create variable
```

②

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

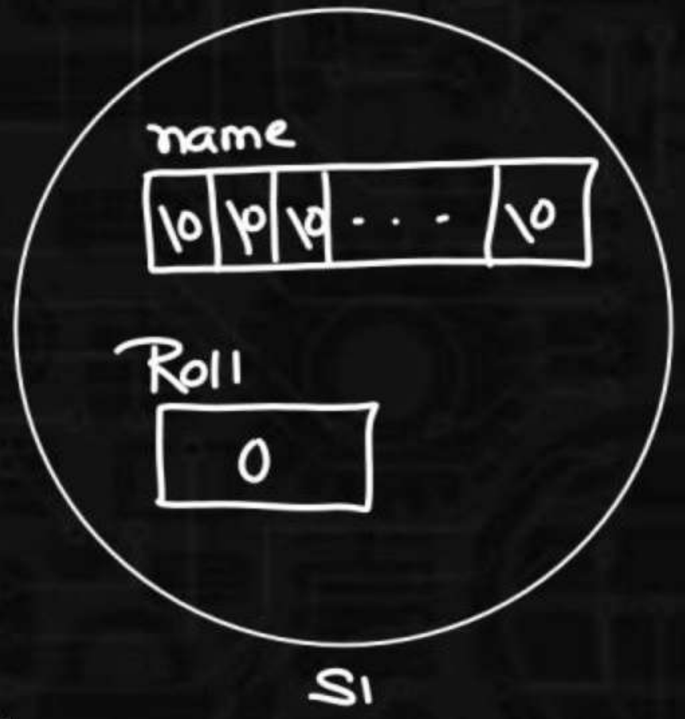
✓

```
void main(){
 struct student S1;
```

③ Aliasing

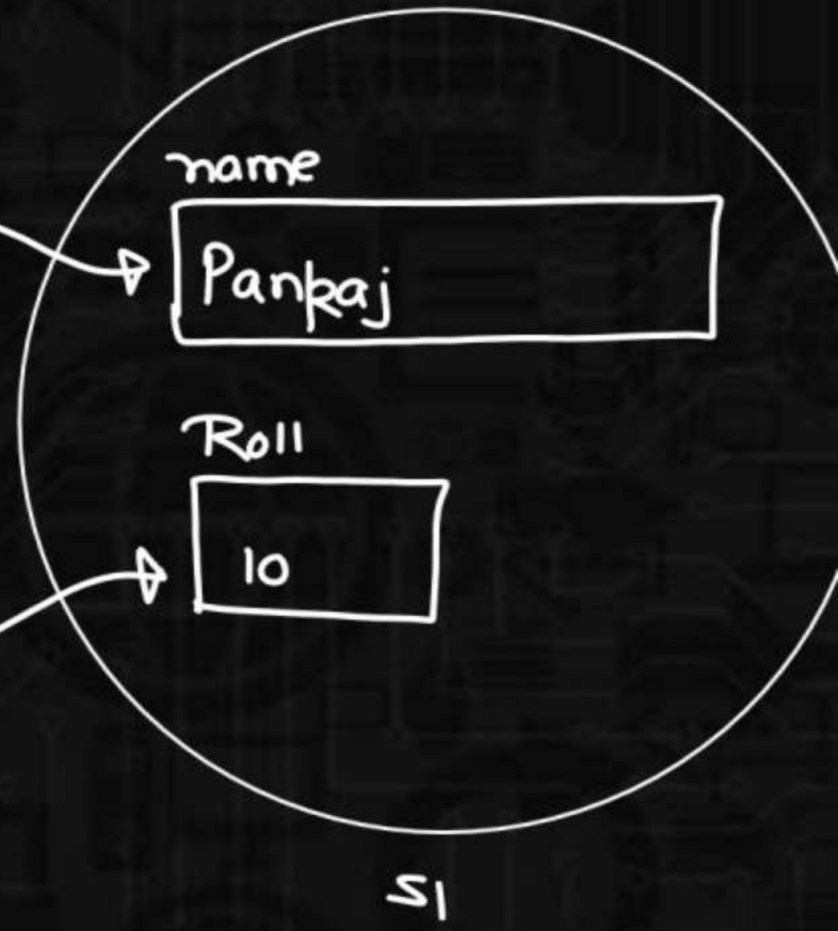
```
typedef struct student{
    char name[20];
    int Roll;
    } X;

void main(){
    X S1,S2;
    =
    }
```

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

void main(){
    struct student s1;
printf("%d", s1.Roll);
printf("%s", s1.name);
```



default values for members ⇒ 0

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

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



```
int a,b;  ⟹  int a[2];
```

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

≡

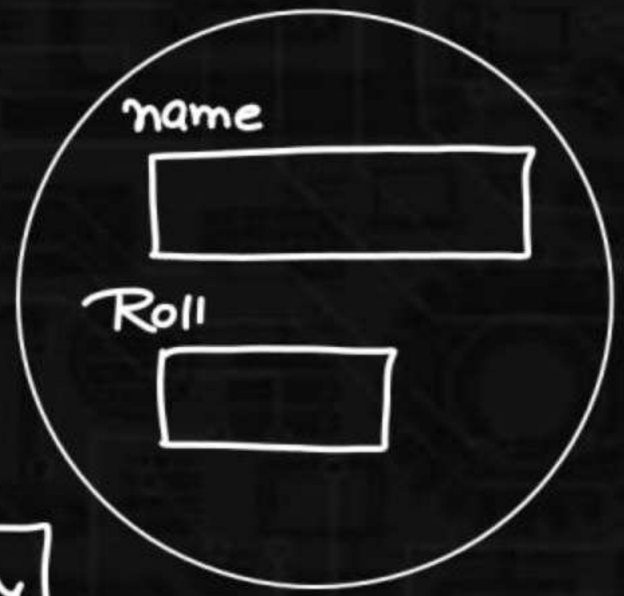
Array of structure

```
void main(){
    struct student s[2];
```

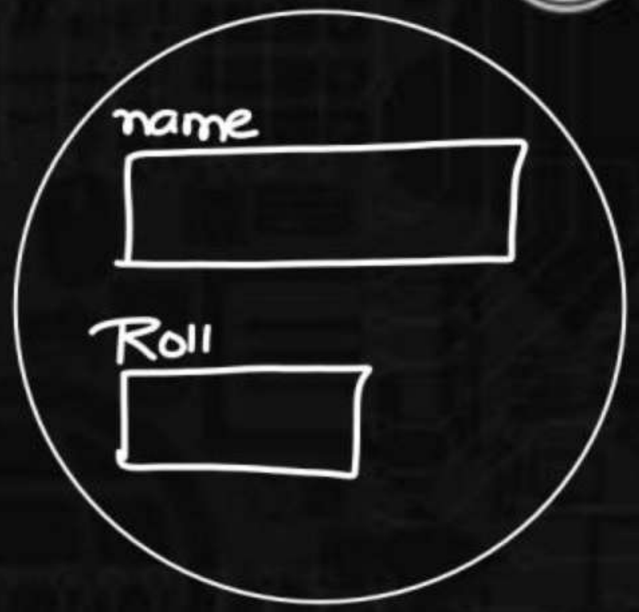
```
struct student{
    char name[20];
    int Roll;
        };

void main(){

stouct studen s[2];
s[0].Roll = 10;
s[1].Roll = 11;
s[0].name = "Pankaj";
```

strcpy

invalid



s[0]



s[1]

```
char name[20];
name = "Pankaj";
```

declare

Lvalue

```
char name[20];
name = "Pankaj";
```

Lvalue ✗

```
a = 10; ✗
constant = 10; ✗
```

```
char name[20];
strcpy(name, "Pankaj");
```

✓

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

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

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

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

void main(){
  struct student s1;
```

s1.name ✓

s1.Roll ✓

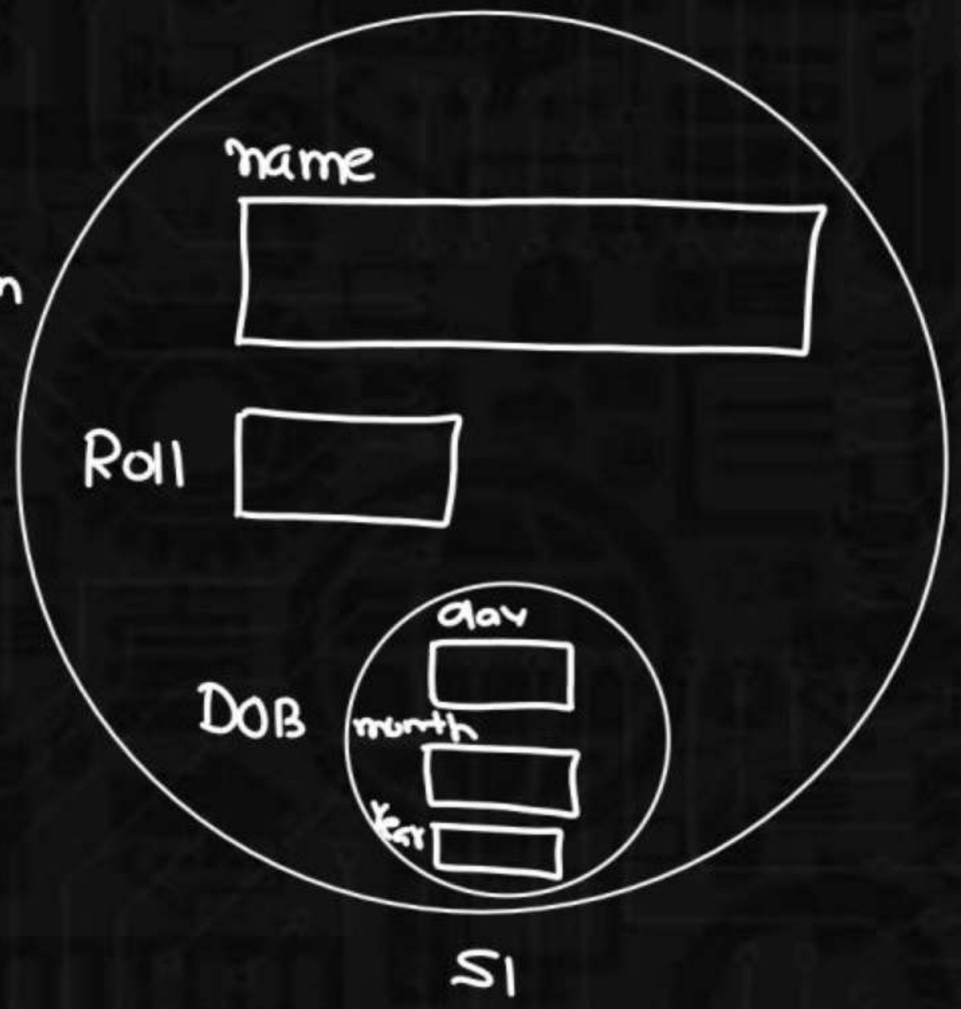
s1.DOB → collection

s1.DOB.day ✓

s1.DOB.month ✓

s1.DOB.Year ✓

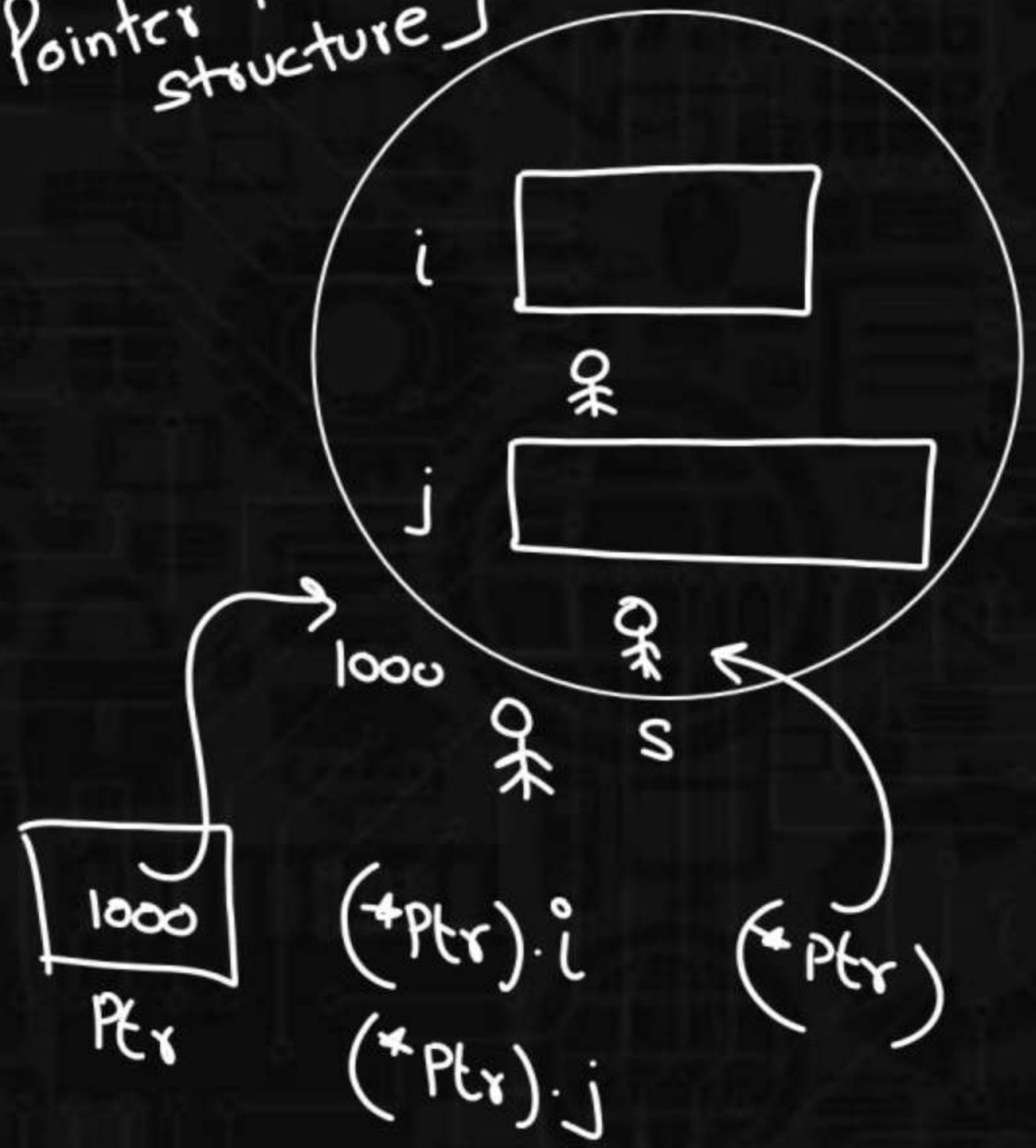
Nesting of structure



Structure Vs Arrays

```
struct my_structure {
        int i;
        float j;
        };

void main() {
    struct my_structure s;
    struct my_structure *Ptr;
        Ptr = &s;
    (*Ptr)
```

{ Ptr : Pointer to structure }



```
int i;
int *ptr;
```

```
struct student s;
struct student *ptr;
```

```
float j;
float *ptr;
```

Ptr: Pointer to structure

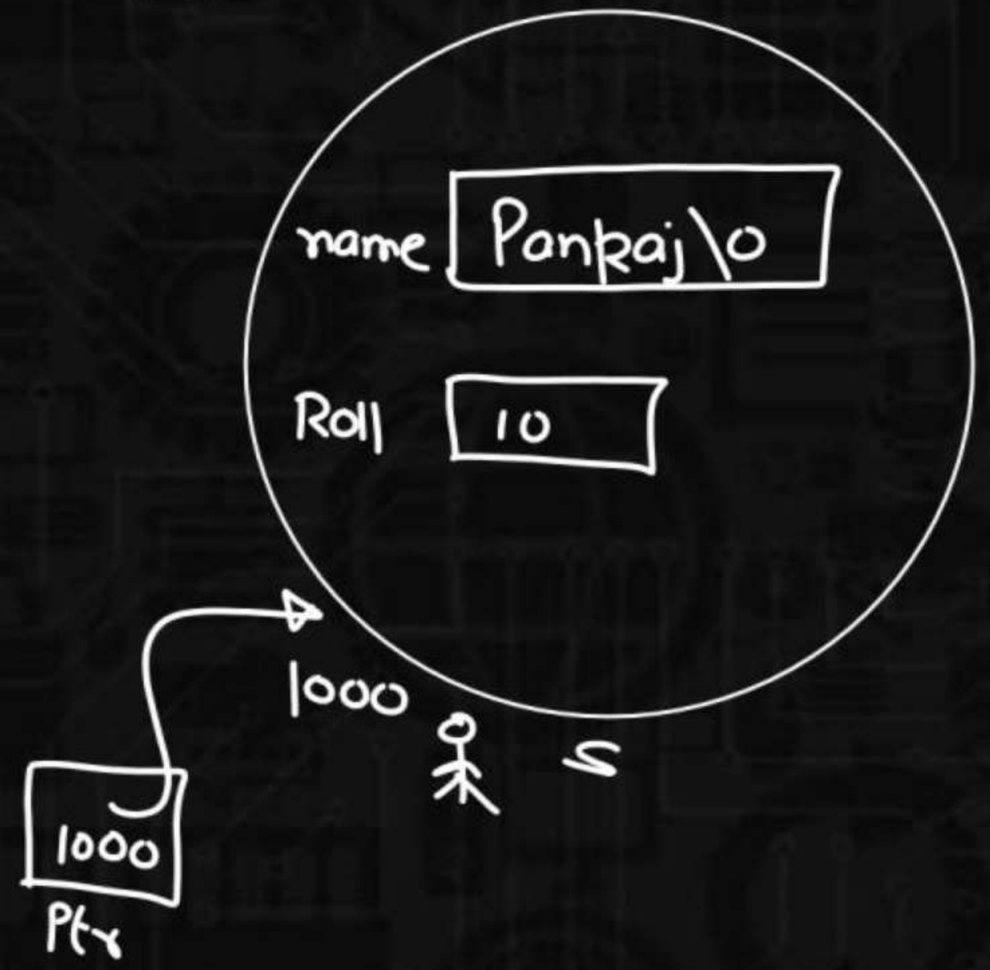
$$\left\{ \begin{array}{l} (*Ptr).member1 \\ (*Ptr).member2 \end{array} \right\}$$

$\longrightarrow$: To access members using pointer to structure

Pointer to structure $\longrightarrow$ member1

Pointer to structure

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

void main(){
    struct student s = {"Pankaj", 10};
    struct student *Ptr;
    Ptr = &s;
printf("%s", Ptr -> name);
printf("%d", Ptr -> Roll);
```

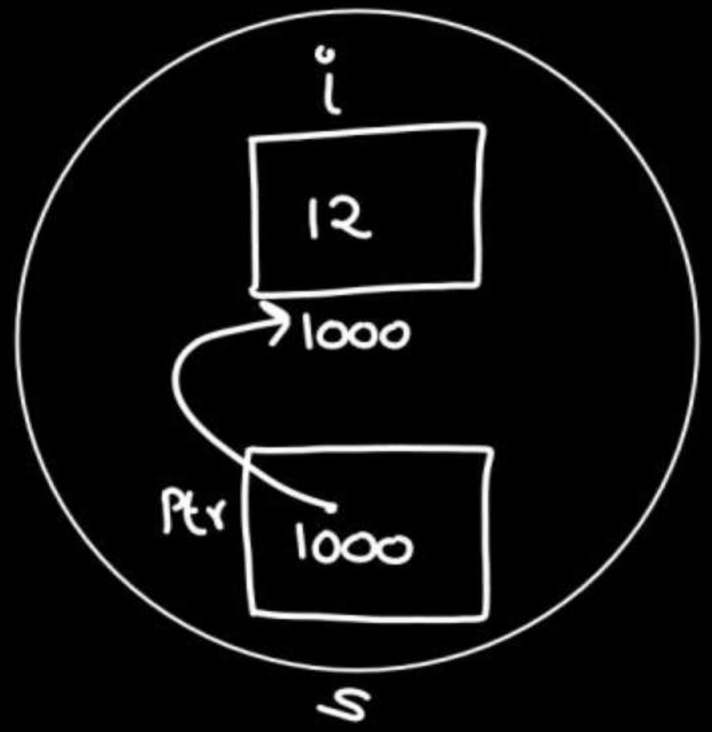


```
struct Pankaj{
    int i;
    int *Ptr;
};
```

valid

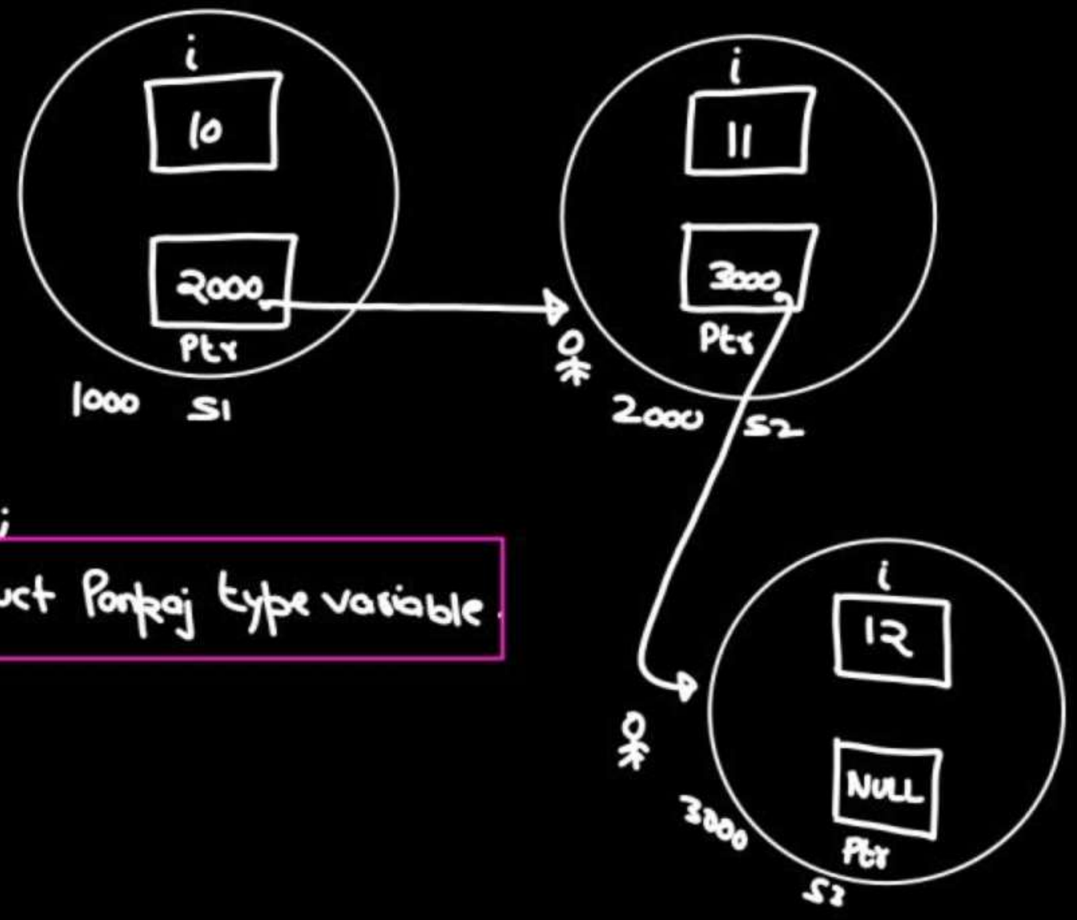
```
void main(){
    struct Pankaj s;
    s.i = 12;
    s.Ptr = Address of any integer variable
    s.Ptr = &s.i;
```

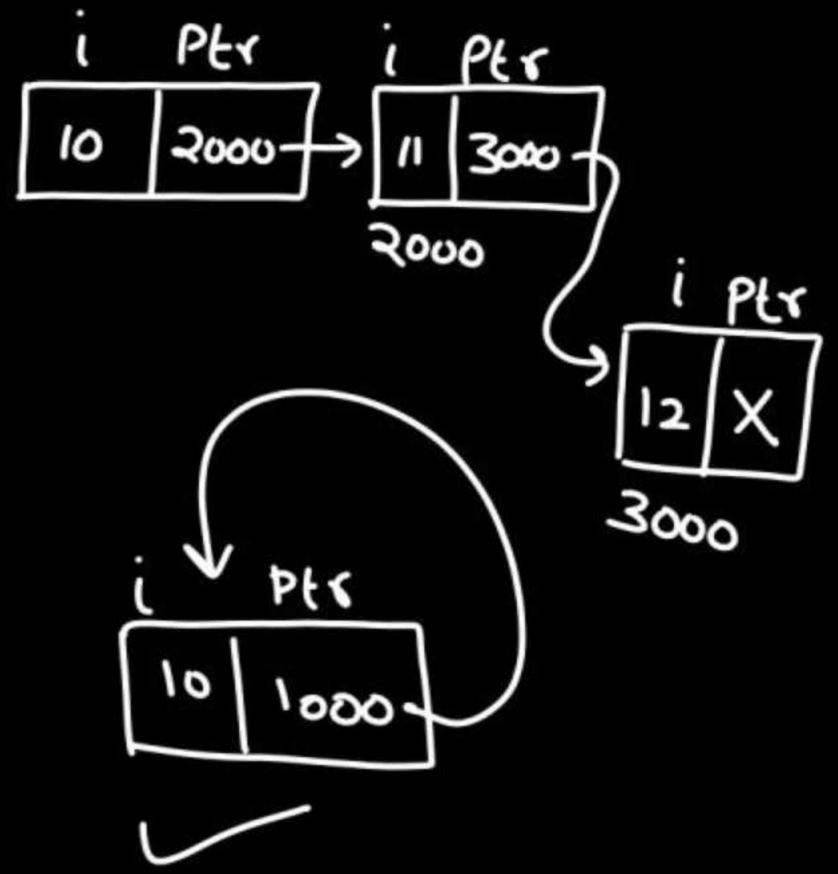
valid



Self-referential structure

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

void main(){
struct Pankaj s1,s2,s3;
s1.i=10; s2.i=11; s3.i=12;
s1.Ptr = Add. of struct Pankaj type variable.

s1.Ptr = &s2;
s2.Ptr = &s3;
s3.Ptr = NULL;
```





Union : All members share same memory space.

```
struct Pankaj{
    char x;   1
    int a;    4
    float b;  8
    };
void main(){
    struct Pankaj P;
```

$1+4+8=13$

```
union Pankaj{
    char x;
    int b;
    float c;
    };
void main(){
    union Pankaj P;
```

$\max(1,4,8) = 8$


THANK YOU GW
SOLDIERS !