CS & IT ENGINEERING

Data Structures

Linked List
Chapter- 3
Lec- 01

By- Pankaj Sharma sir

Linked List-I

Array

```
#define SIZE 10
#include<stdio.h>
void main(){
```

$n-1$

0	1	2	3	4	5	6	7	8	9
10	20	30	40	100					

$n \rightarrow$ no. of elements

}

$n = 4$
SIZE = 10

1.) Insert an element

$x = 100$

End

$A[n] = x$
$n = n+1$

Array

```
#define SIZE 10
#include<stdio.h>
void main(){
```

0	1	2	3	4	5	6	7	8	9
10	20	30	40						

n-1

n → no. of elements

}

n = 4
SIZE = 10

1.) Insert an element

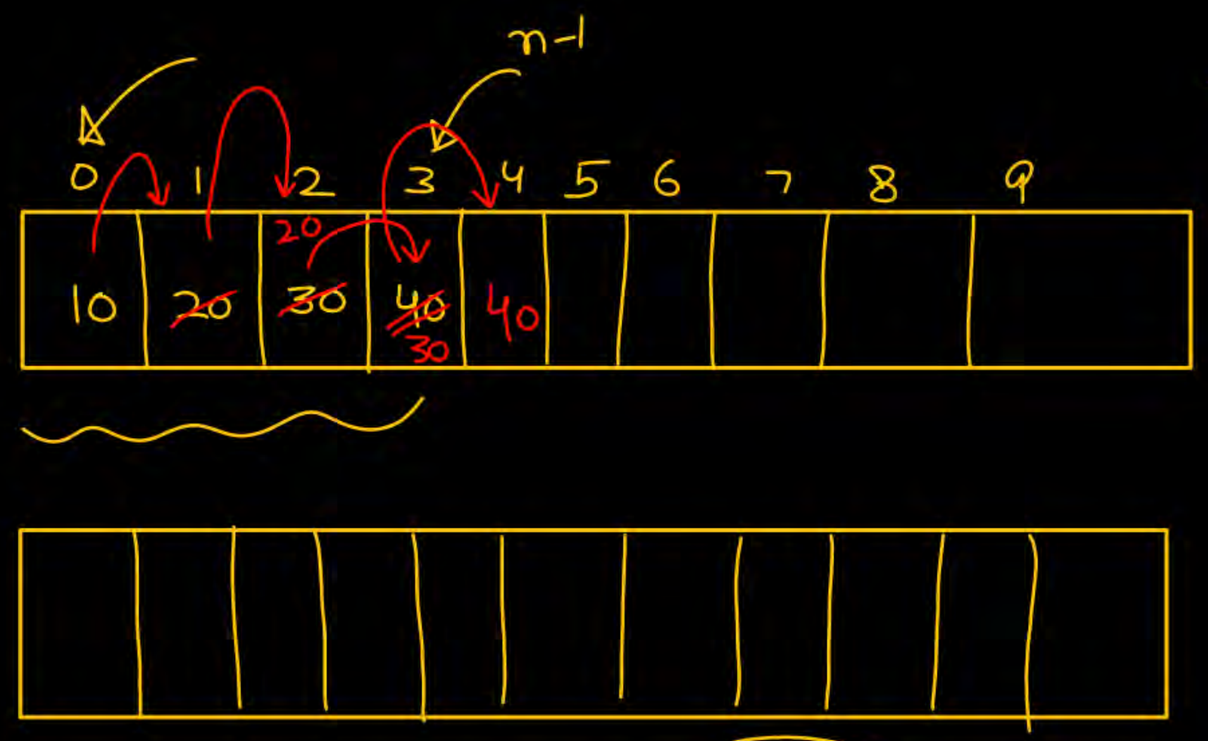
x = 100

index = 0

100	10	20	30	40

Array

insert 100 at index 0




```
for(i=4; i>=1; i--)
    A[i] = A[i-1];
```

A[4] = A[3]
A[3] = A[2]
A[2] = A[1]
A[1] = A[0]

Insert operation

→ (n) statement

⇒ O(n)

delete operation

$n = 5$

SIZE = 10

0	1	2	3	4	5	6	7	8	9
10	30	~~30~~ 40	~~40~~	50					

delete element at $\boxed{\text{index } 1}$

$$\begin{cases} A[1] = A[2] \\ A[2] = A[3] \\ A[3] = A[4] \end{cases}$$

```
for(i=1; i<=3; i++)
    A[i] = A[i+1]
    n = n-1;
```

$\Rightarrow$

n element

last element $\Rightarrow (n-1)$

Array → Insert, delete : Time consuming operation

vectors

Size ⇒ pre-define → static

advantages of array : → (1) random access → constant time

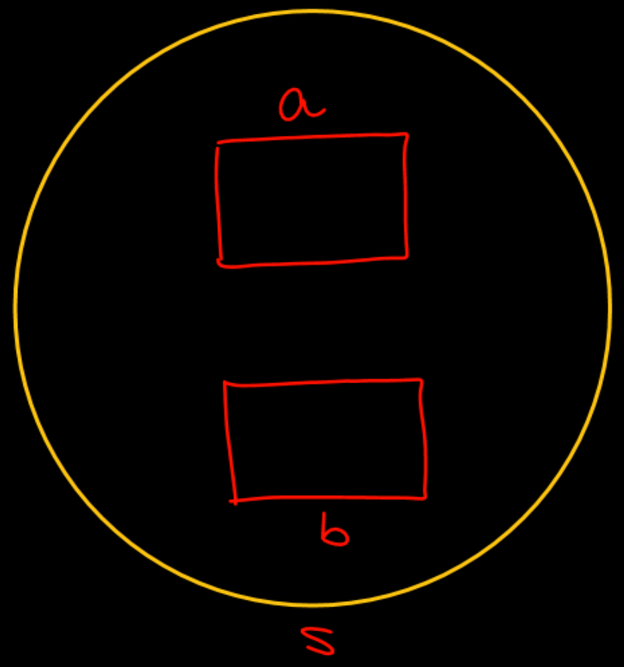
(2) Cache friendliness

performance. ↑↑

Structure

```
struct my_struct {
        int a ;
        float b ;
};
```

diff type



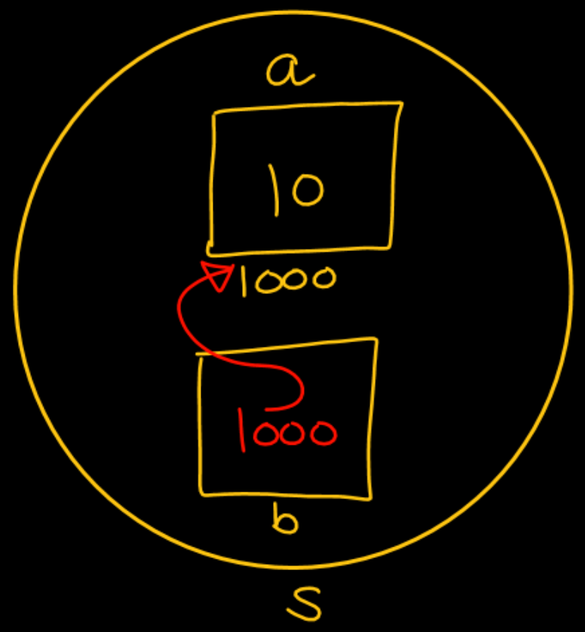

```
void main(){
    struct my_struct s;
        int i
```

```
struct my{
        int a ;
        int * b ;
    };

void main( ){
    struct my s ;
        s.a = 10;
        s.b = &s.a ;
```

s.b

s.&b ✗



Pointer to int

valid

address of a integer variable

mem. location 1000

```
struct Pankaj {
        int a ;
        float *ptr ;
}
```

```
struct Pankaj {
        int a ;
        char *ptr ;
}
```

```
struct Pankaj {
        int a ;
struct Pankaj *ptr ;
};
```

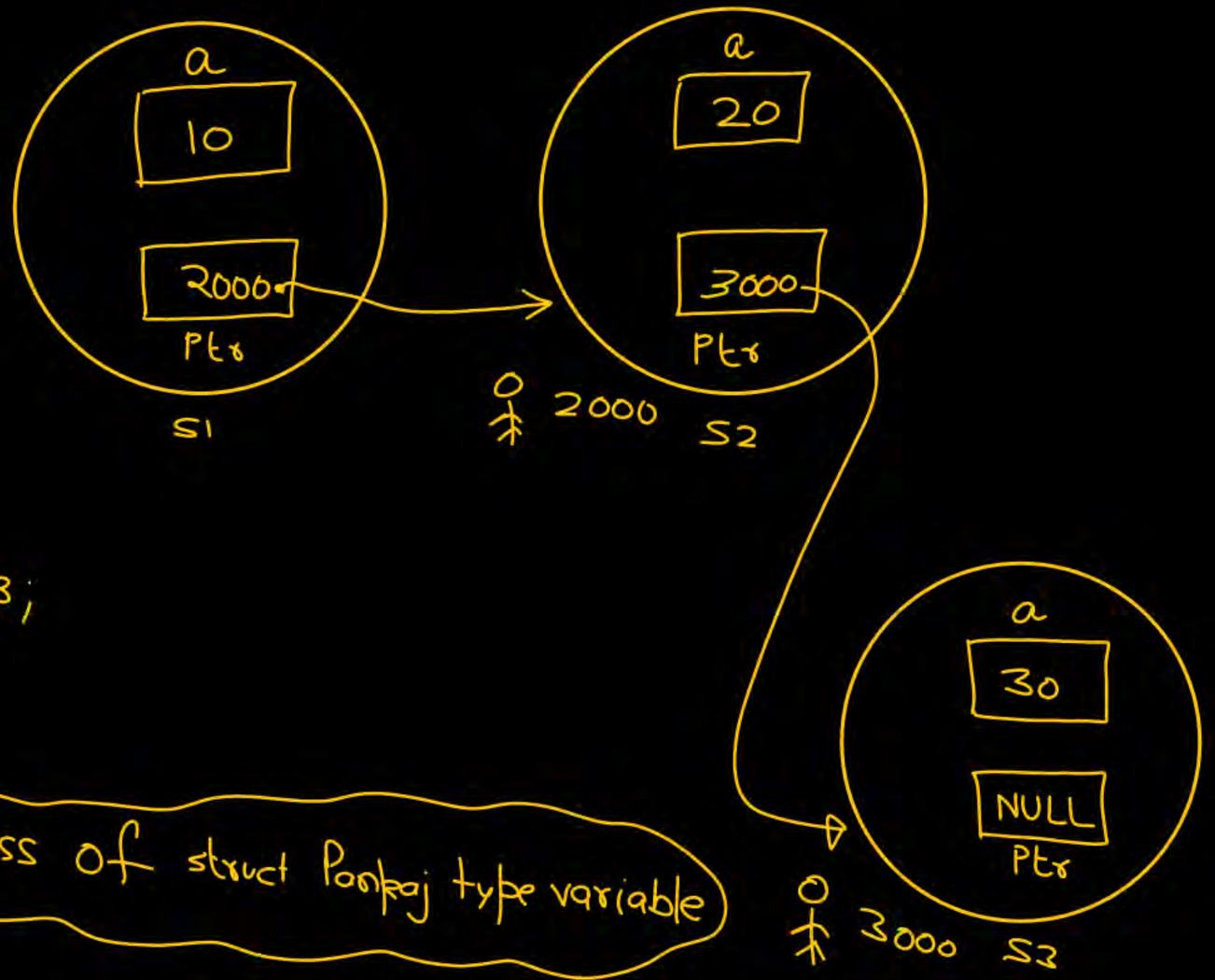
Valid

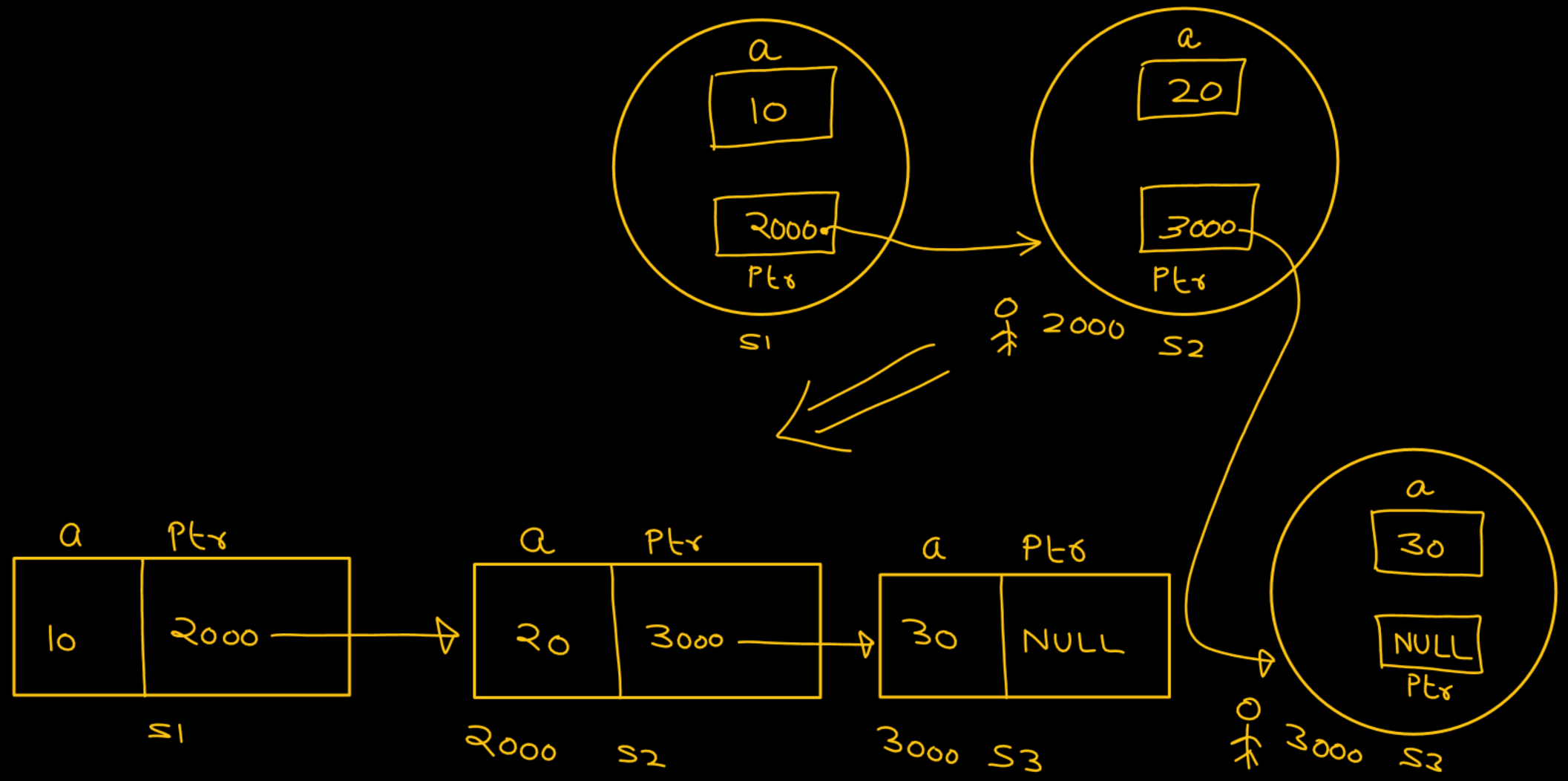
```
struct Panpaj {
        int a;
        struct Panpaj *Ptr;
};

void main(){
    struct Panpaj s1,s2,s3;
    s1.a = 10;
    s2.a = 20;
    s3.a = 30;
    s1.Ptr = Address of struct Panpaj type variable

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

Self referential structure




a
10
2000
Ptr
S1
a
20
3000
Ptr
2000
S2
a
30
NULL
Ptr
3000
S3
a
Ptr
10
2000
S1
a
Ptr
20
3000
2000
S2
a
Ptr
30
NULL
3000
S3

Linked List

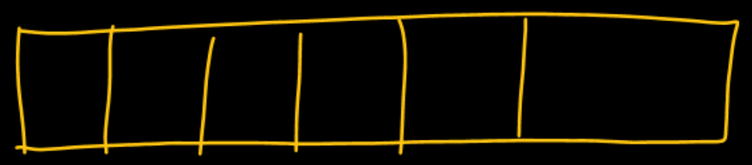
A linear data structure, which is collection of elements called nodes and each node is divided into 2 parts:

(1) data part

(2) contains address of next node in L.L.

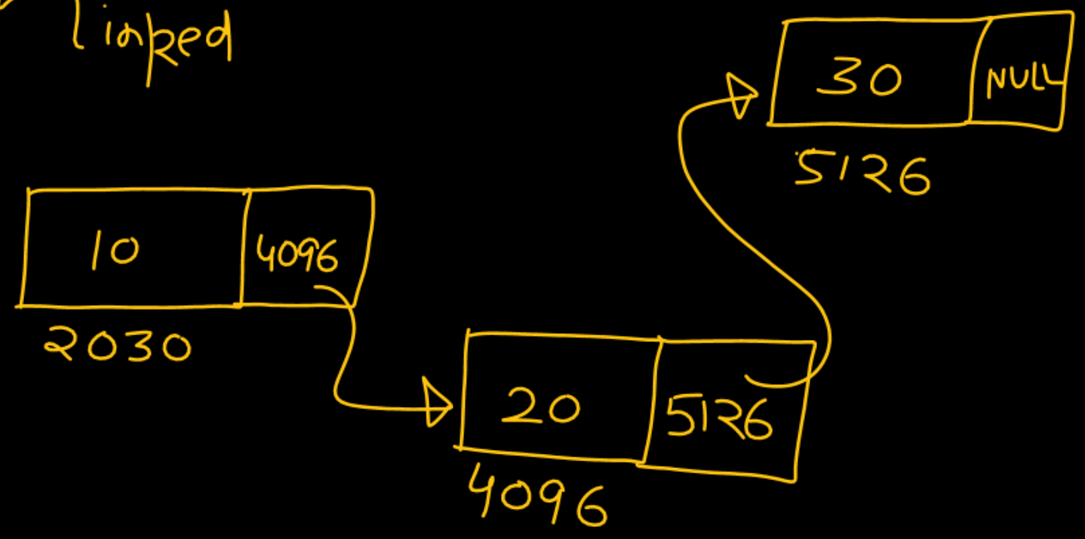
Linear d.s

Arrays



Implicitly

linked



linear order is maintained $\Longrightarrow$ Explicitly by using pointer

```
struct Node {
        int data ;
    struct Node *Link ;
        };
```

```
void main( ){
        Insert (100) ;
        ≡
        }
```

```
void Insert(     )
  {
    struct Node s ;
    ≡
  }
```

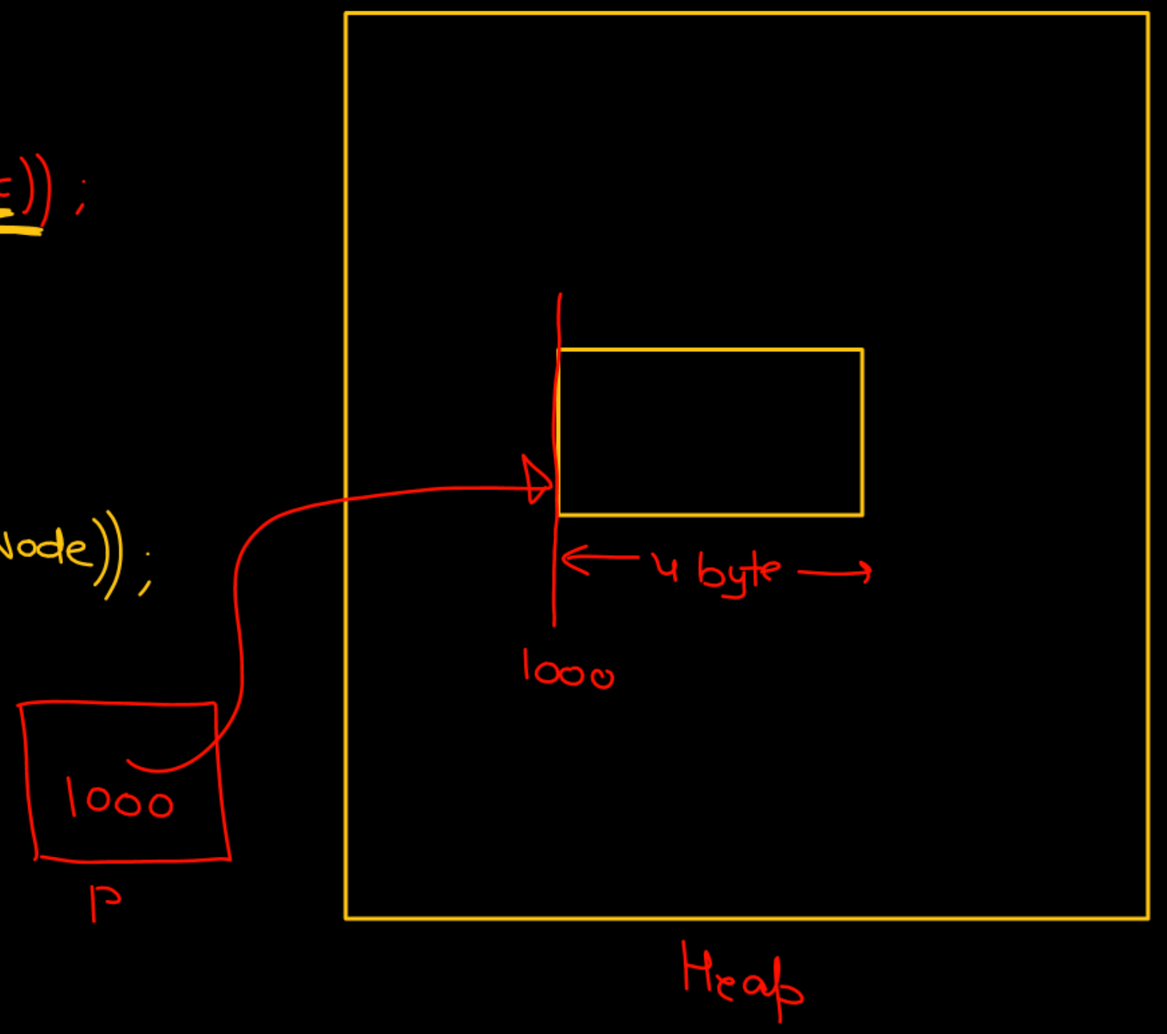
local variable

10	2000

int *p

p = malloc(sizeof(int));

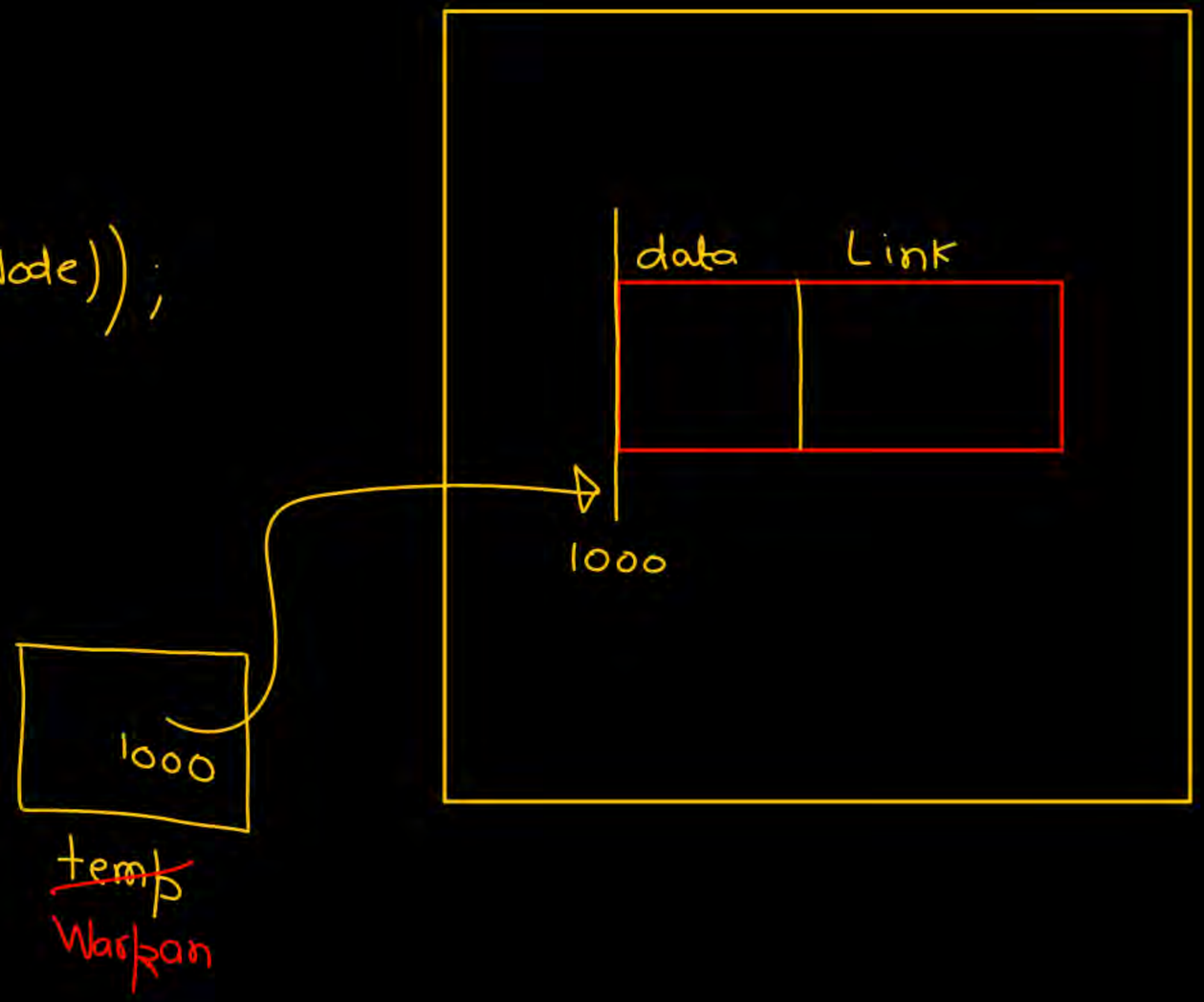
malloc(sizeof(struct Node));

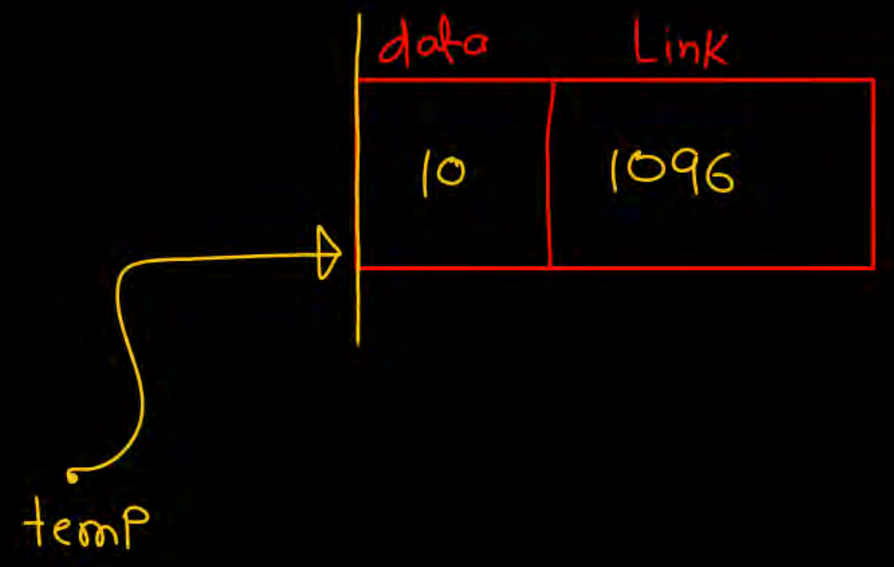


```
struct Node{
        int data ;
    struct Node *Link;
    };
```

```
struct Node *temp;
```
Warkan

```
temp = malloc(sizeof(struct Node));
```
Warkan



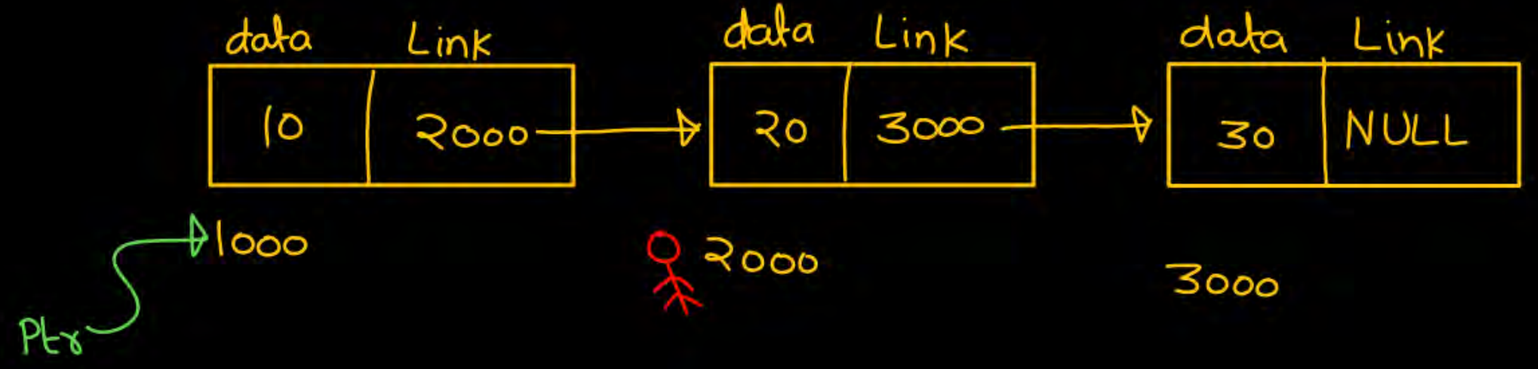


(Pointer to a node (structure))

Pointer to structure के member kaise access ?

temp → data ⇒ 10

temp → Link ⇒ Memory location 1096



Struct Node *Ptr;

Ptr → data : 10

Ptr → Link : Memory loc. 2000 ⇒ Address of 2nd node

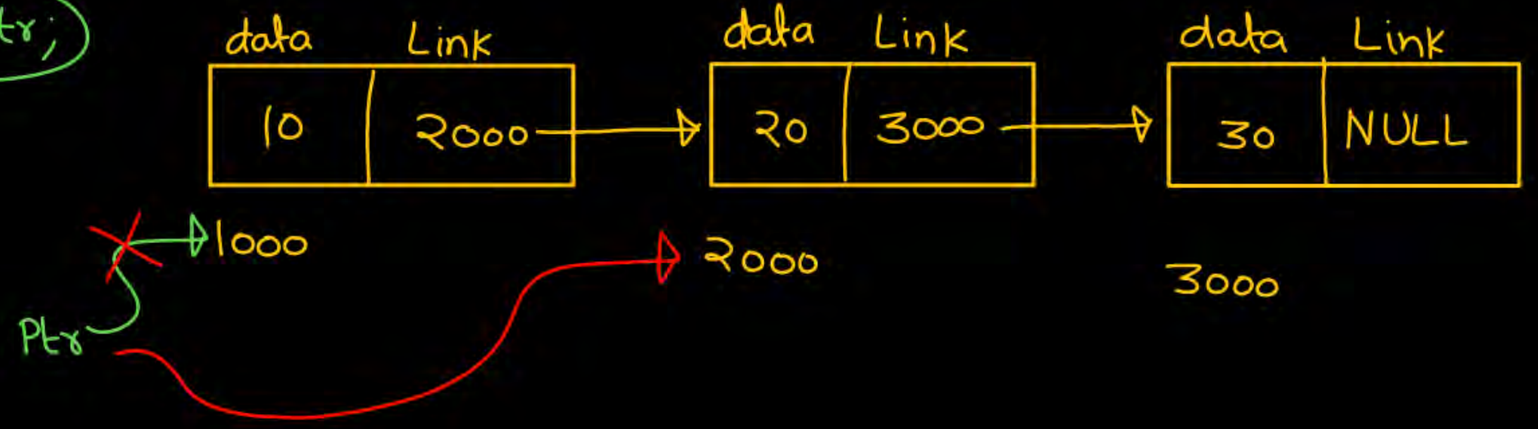
⇒ Pointer to second Node

(i) (Ptr → Link) → data : 20

(Ptr → Link) → Link : Memory loc. 3000 : Add. of 3rd node

⇒ Pointer to 3rd Node

Struct Node *Ptr;



Ptr → Link (2000)

↓

Address of a Node

In array

index

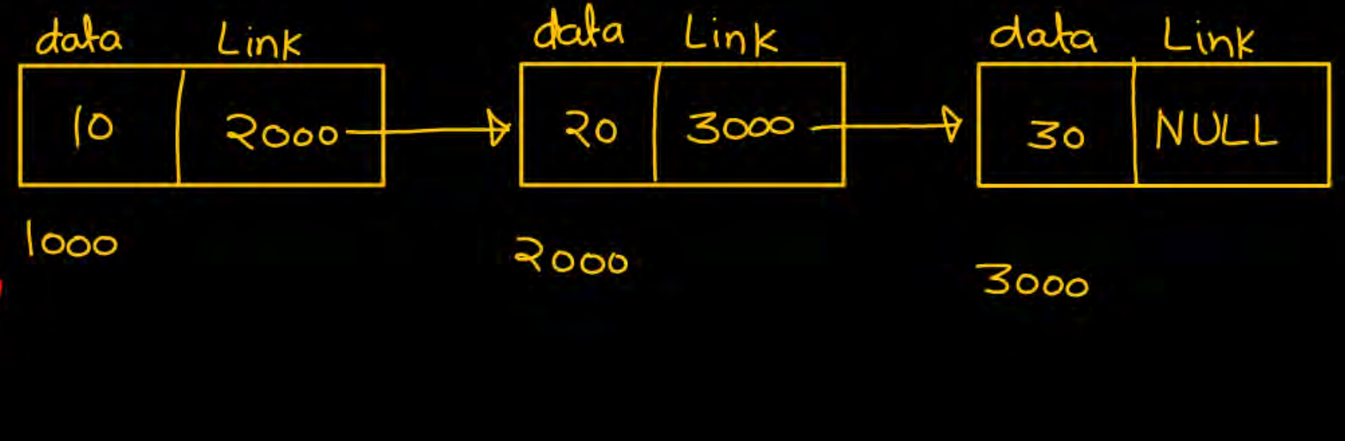
next element ओर index

index++;

$$\boxed{\text{Ptr} = \text{Ptr} \rightarrow \text{Link}}\ ;$$

2000

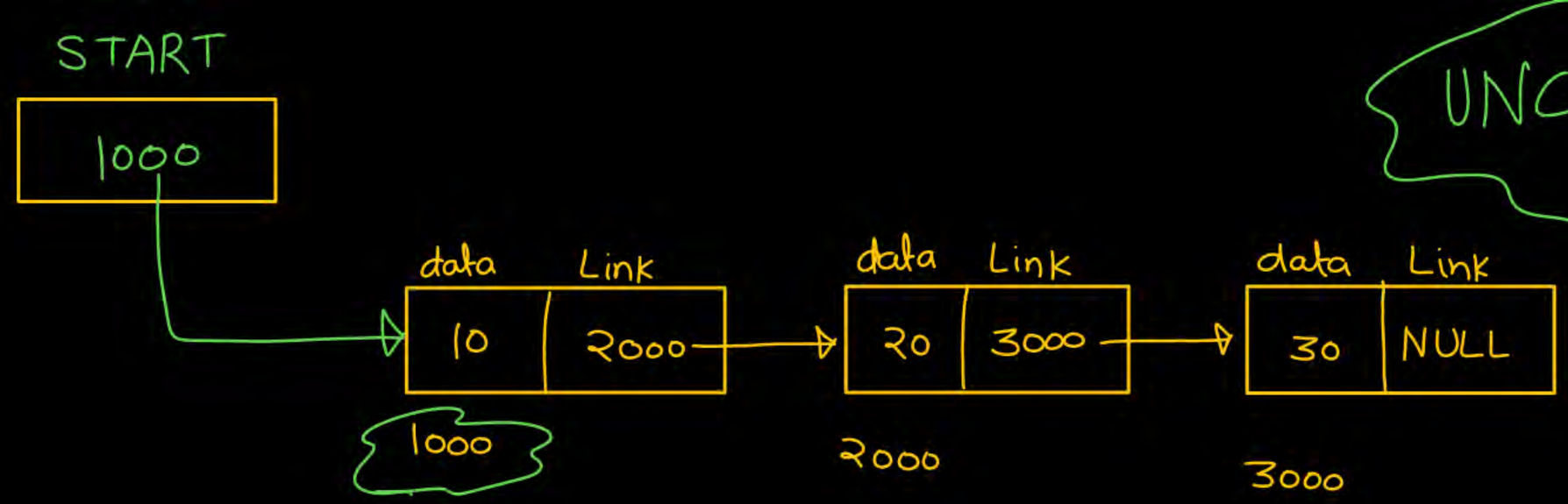
Struct Node *Ptr;



Struct Node * START;

⇓

hold address of first node in linked list



UNCOVER the Syllabus

linked list Empty ⇒ there is no node

START == NULL

PYQs

Live

1000 elements

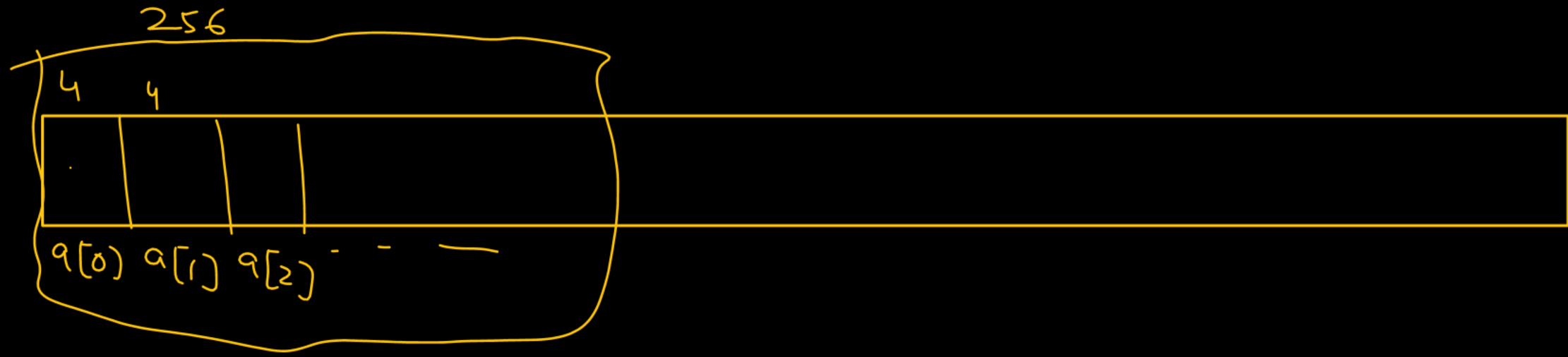
4 bytes

1KB ⇒

```
for(i = 0; i< 1000; i++)
{
printf("%d", a[i]);
}
```

a[0]

4byte




THANK YOU GW
SOLDIERS !