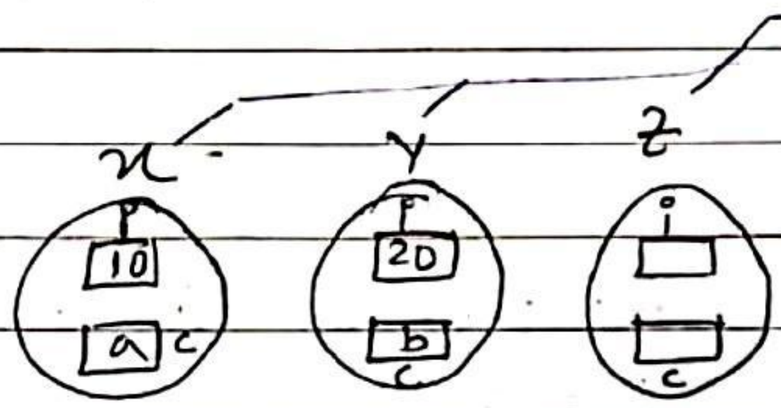
Structures

- Introduction of Structure :

```
Struct
{ int i;
  char c;
} x, y, z;
```

→ declaration



x.i = 10 | y.i = 20

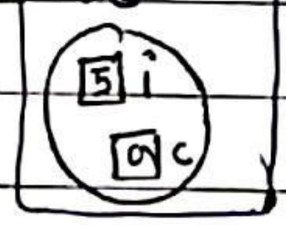
x.c = 'a' | y.c = 'b'

→ member operater

Struct ex → Tag
{ int i;
  char c;
};

struct ex x, y, z;

struct ex x = {5, 'a'};

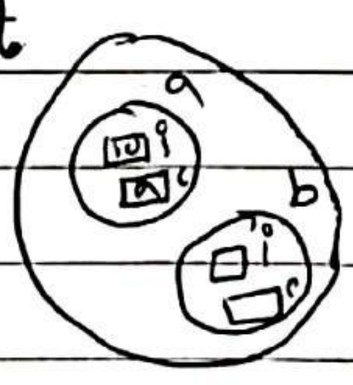



```
struct ex1
{
   struct ex a;
   struct ex b;
};
```

→ declaration.

struct ex1 t;

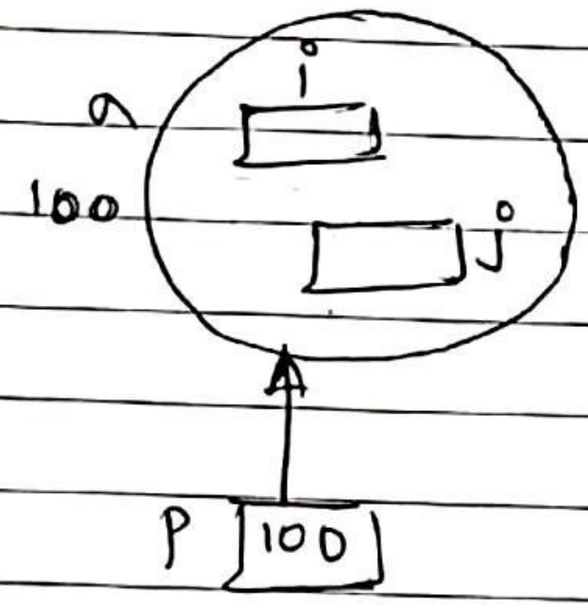
t.a.i = 10
a, a.c = 'a'
→ define.



- Example on Structures, arrays and pointers =

```
struct node
{ int i;
  int j;
};
```
→ declaration (no memory allocated)

```
Struct node a, *p;
p = &a;
```
→ Tag (not necessary)



→ access of member of Structure using Pointer.

(*p).i

a.i → access by name of structure.

p → i° same as (*p).i°

→ structures can be pass to a function as well as returning by a function.

struct node fun (struct node n1, struct node n2);

Example:

```
struct node
{ int i;
  int *c;
};
struct node a[2], *p;
int b[2] = {30,40};
p = &a[0];
a[0].i = 10; a[1].i = 20;
a[0].c = b;
```

++p → i

n = (++p) → i

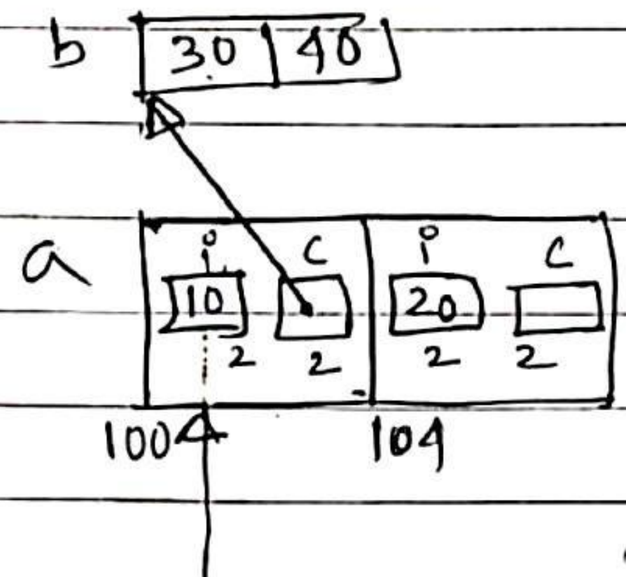
n = (p++) → i

n = *p → c

n = *p → c++

n = (*p → c)++

n = *p++ → c.



$x = (++(p \to i))$ / $x = 11$

$x = (++p) \to i$ / $x = 20$

$x = (p++) \to i$ / means $p \to i$ means $p = p+1$ / $x = 10$

$x = (*(p \to c))$ / $x = 30$

$x = (*(p \to c))++$ / $x = 30$ (For post increment $x = 31$).

$x = (*p \to c)++$ / $x = 30$.

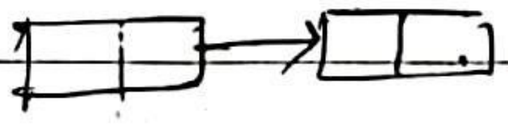
$x = (*(p++) \to c))$ / $x = 30$

- **Self referential structures:**

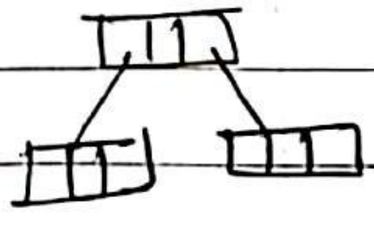
```
struct ex
{ int i;
  struct ex *link;
};
struct ex abc;
```

example of self referential structure:

(1) Link list.

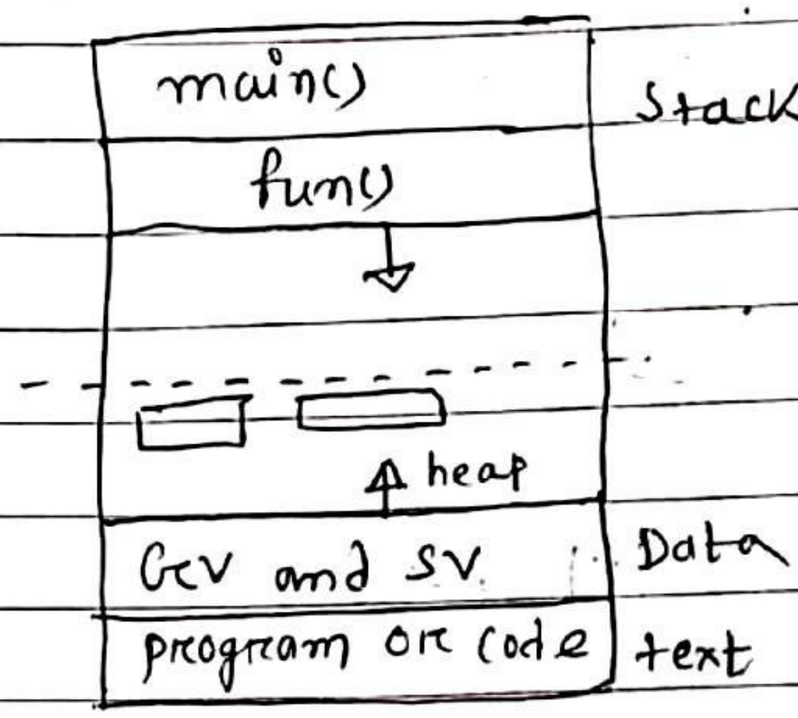


(2) tree



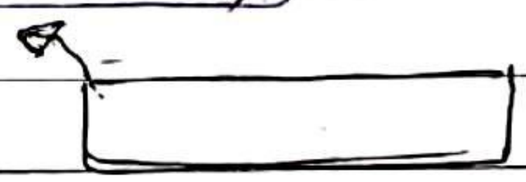
- **Malloc** =

→ static and global variables : memory allocated before run the program.



10

`void * malloc (int);`



malloc function call make a space in Dinamicly of process in the heap and then return the starting address of this location.

`int *p = (int*) malloc (2);`

`void * malloc (sizeof(int));` — use this one

`int*p= (int*) malloc (size of (2))`

↳ type casting

syntax (which use to make struct and get pointer)

```
struct node
{ int i; 2
  struct node *l; 2
};
```

`struct node* p = (struct node*) malloc(sizeof(struct node))`

4

malloc(sizeof(struct node));