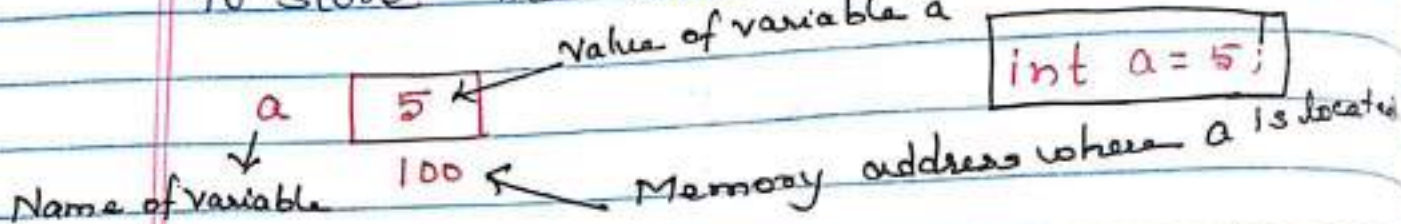
UNIT-8

POINTERS

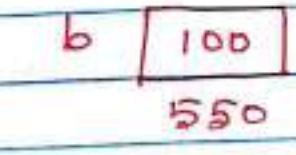
[Idea of pointers, Defining pointers, Use of pointers in self-referential structures, notion of linked list]

POINTER: Pointer is the variable used to store the address of another variable.



`int *b;` [Create a pointer variable] `b = &a;`

→ b is a pointer variable which stores the address of a



b is a pointer variable having address of a

`printf("%d", *b)` o/p 5

`printf("%d", a)` o/p 5

* A pointer assigned NULL is null pointer.
  `int *ptr = NULL;` [NULL = 0 is defined in many standard libraries]
* Pointers are initialized to zero
* Value of null pointer is 0 which means the pointer is not pointing to anything
* & Symbol is used to get address of variable
* * is used to get value of variable, the pointer points to

Example:

```
#include<stdio.h>
int main()
{
    int var = 20;
    int *ip;
    ip = &var;
    printf("Address of variable var: %x",
                                &var);
    printf("Address stored in ip variable
                        : %x", ip);
    printf("Value of *ip variable: %d",
                                *ip);
    return 0;
}
```

Output:

```
Address of variable var : bffd8b3c
Address stored in ip variable: bffd8b3c
Value of *ip variable : 20
```

Pointer arithmetic.

* 2 pointers can be subtracted to to find the number of elements between the two.
* Arithmetic operators ++, --, + & - and comparison operators <, >, ==, >=, <= are used with pointers and are meaningful when pointers are used with arrays.

Example:

```
int var[3] = {10, 15, 20};
int *ptr;
ptr = &var[0];
```

ptr points to first element
ptr++ can be used to find next elements

Self referential structures

The structure which has one or more pointer that points to the same type of structure as member is called self referential structure.

Example:

```
struct node
{
    int data1;
    char data2;
    struct node *link;
};
int main()
{
    struct node ob;
    return 0;
}
```

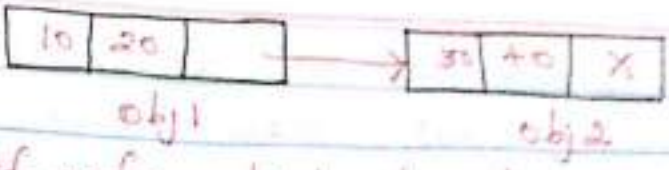
- link is a pointer to structure of type node.
- node is a self referential structure
- link is the referencing pointer
- pointer should be initialized properly, because in default it has garbage value.

Types of self referential structures:

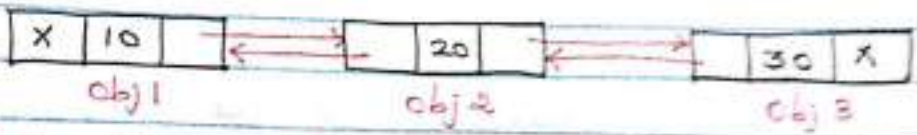
i) Self referential structure with Single link
ii) Self referential structure with multiple links

i) Self referential structure with single link

- has only one self pointer as member



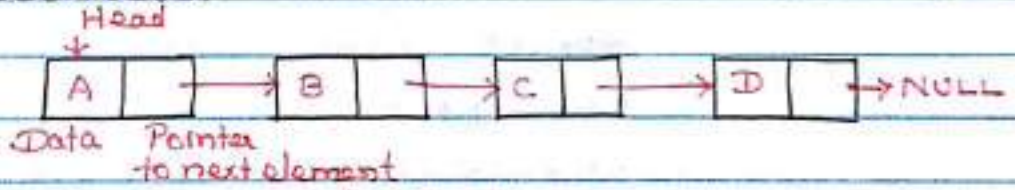
ii) Self-referential structure with multiple links
- has more than one self-pointers as their members



Advantage of self-referential structures:
- Used for creation of complex data structures like
  * linked list
  * stack
  * queue
  * trees
  * graphs

Linked List:

Linked list is linear data-structure, where elements are not stored at contiguous location (opposite to array). The elements are linked using pointers.



Advantage of linked list over array:
* dynamic size
* insertion/deletion is easy

Representation:

Each node in the list has 2 parts:
i) data

ii) pointer to the next node

The first node is called head. If the linked list is empty, value of head is NULL. The last node has pointer to NULL

Drawbacks of linked list compared to array

* Not possible for random accessing.
* Extra memory space for pointer
* Not cache friendly because of non contiguous location of elements

Example : [Self-referential structure with single link]

```
#include <stdio.h>
struct node
{   int data1;
    char/int data2;
    struct node *link;
}; int main()
  {  struct node ob1;
     ob1.link=NULL;
     ob1.data1=10; ob1.data2=20;
     struct node ob2;
     ob2.link = NULL;
     ob2.data1 = 30; ob2.data2 = 40;
     ob1.link = &ob2; //link ob1 to ob2
    //accessing datamembers of ob2 using ob1
    printf("%d", ob1.link->data1);
    printf("\n %d", ob1.link->data2);
    return 0;
}
```

O/P

30
40