Declare Linked list

```
Structure = node
    head : type
    suivant : ^node
fin structure
type list = ^node
```

null in Algo for pointers : nil.

```
fonction search (L : node, x : int) : ^node
Var
    select : ^node
```

① to access to the object in Linked List.

```
Debut
    select ← L
    tant que (select) faire
        si (select^.Val) = x Alors
            Retourner select
        fin Si
        select ← select^.Suivant
    fin tant que
    retourner select
Fin
```

```
procedure printL(L: node)
Debut
    tantque L ≠ nil faire
        ecrire(L^.Val)
    Fin tantque.
Fin
```

Inserting in the beginning of the list:
given(x), and list L:
The proc insertion head inserts an element in the list.

```
procedure insert(Val t :Node, x :entier)
Var
  tmp : ^node

Debut
  Allouer(tmp)
  tmp^.Val ← x.
  tmp^.Suivante ← L
  L ← tmp.
Fin
```

Delete the first element:

```
procedure Delete(Var L: Node).
Var
  tmp : ^node

Debut
  si (L ≠ nil) Alors
      tmp ← L.
      L ← L^.next.
      Liberer(tmp)
```

Fin Si

Fin.

- the procedure Delete element eliminates an element x of linked list L.

This procedure needs a pointer on element x to be deleted.

we can use search proc if the pointer is not given.

Delete

```
procedure Delete (Var L : node, X : entier)
  Si (L ≠ nil) Alors
    Si (L^.Val = x) Alors supprimer (L)
    Sinon
    tmp ← L, coment ← L
    tant que (tmp^.Val ≠ x et tmp^.Suivante ≠ nil) faire
        coment ← tmp
        tmp ← tmp.Suivant.

    Fin tant que.
    Si (tmp^.Val = x) alors
        coment^.Suivant ← tmp^.Suivant
        libérer (tmp)
    fin Si
```

free up the full List:

it's a procedure to free the whole list.

procedure free(var L: node)

- Debut
  tant que non est vide (L) faire
    Supprete (L)
  Fin tant que

Fin

Basic Algos :

- Declaring :

  - Structure = Node
    head : entier
    next : ^Node

  Fin structure.

Number of nodes :

fonction nNodes (list : ^Node) : entier.

Var
  count : entier
  l1 : ^Node.

Debut
  count ← 0
  l1 ← list
  tant que (l1 ≠ nil) faire
    count ← count + 1.
    l1 ← l1.next
  fin tant que
  retourner count.

Fin

filling mode of n elements ?

Demonstration :

tmp.

l ← tmp.

free tmp.