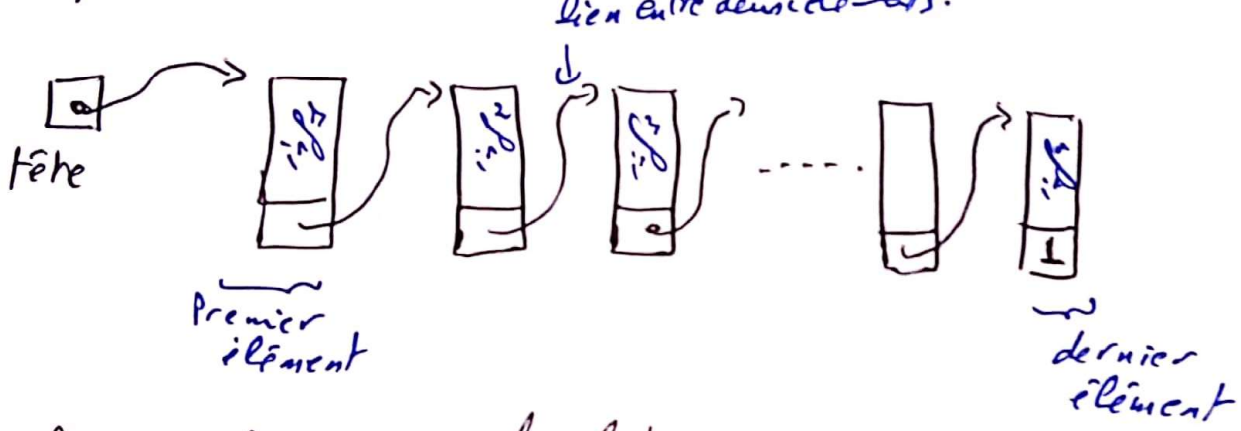
Cours des listes linéaires chaînées

Déf Une liste linéaire chaînée est un ensemble de cellules (éléments) ~~De chaque~~ ~~cet~~ relié entre elle de façon à ce que on puisse passé d'un éléments à un autre, La liste est caractérisé par ~~ça~~ tête qui permet de localisé le premier éléments



Chaque élément de la liste est Co-Posé de deux Parties : - Une Partie infor-ation qui contient la donnée à enregistré

- Une Partie adresse (Pointeur) Pour localisé l'élément suivant si il existe sinon on doit signaler la fin de la liste (nil)

Déclaration

la déclaration suit la for-e suivante

```
type élém = enregistrement(<infos> : <types>;
                           suiv : Pointeur de élém;
                          )
Var
    tête : Pointeur de élém;
```

exemple

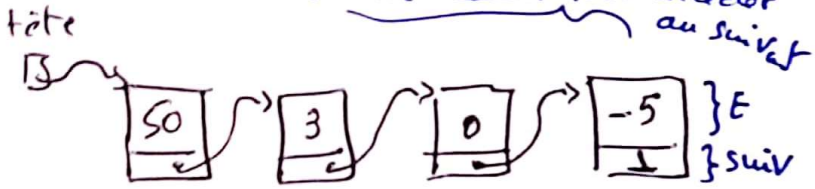
```
type
    élém = enregistrement
            ( E : entier;
              suiv : Pointeur de élém;
            )

Var
    tête : Pointeur de élém
```

structure d'un élément de la liste

création du point d'entrée de la liste.

chaque élément comporte une valeur entière et une adresse pour accéder au suivant



Ajout des éléments à la liste

après la déclaration de la structure des élément et la création de la tête de liste.

la liste est initialement vide (tête = nil).

l'ajout des élément peut se faire de deux manière :

a) les élément seron ajouté à la fin (FIFO)

b) les élément seron ajouté au début (LIFO)

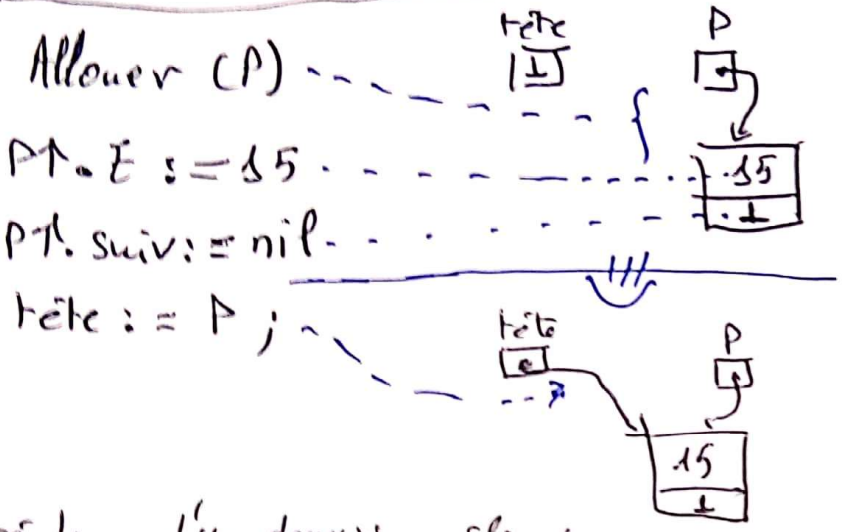
Remarque la présentation des Algorithme vont se basé sur la même déclaration que l'exemple

```
type élémen = enregistrement ( E : entier;
                               suiv : Pointeur de élém;
                             )
Var tête, P : Pointeur de élém;
```

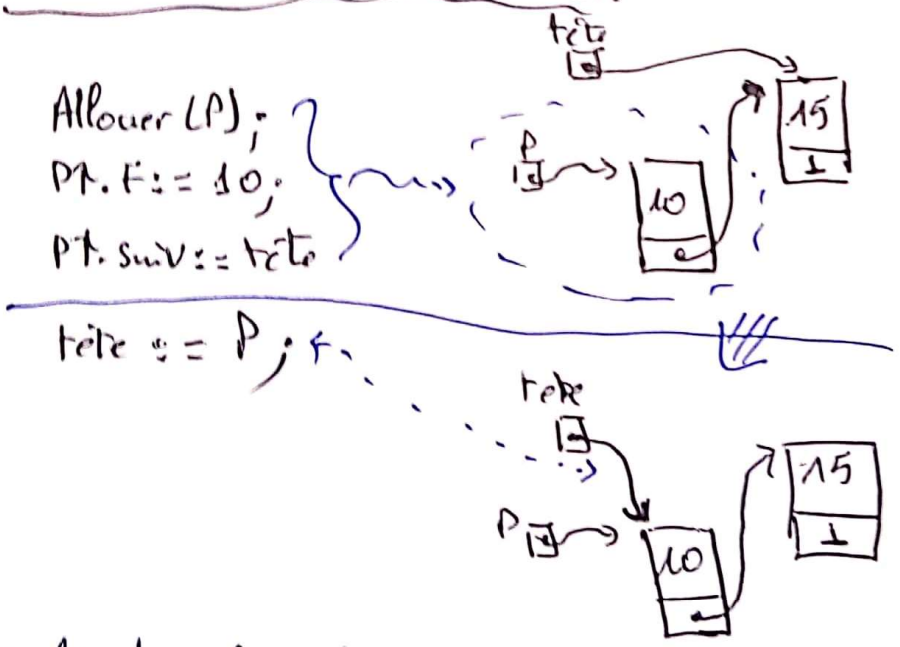
a) Ajouter des éléments au début (LIFO)

1 - création d'une liste vide tête := nil;

2 - création du premier élément



3 - création d'un deuxième élément



Généralisation pour la création de n cellules

```
lire (n);
tête := nil;
pour i := 1 à n
    faire
        Allouer (P);
        lire (P↑.E);
        P↑.suiv := tête;
        tête := P;
    fait;
```

Affichage des élément de la liste

```
P := tête ;
tant que P <> nil } on avance jusqu'à la fin de la liste
faire
    écrire (P↑.E);
    P := P↑.suiv ; } permet de pointé à l'élément suivant
fait;
```

Supression de la liste (supprimer tout les élément)

```
tant que tête <> nil
faire
    P := tête;
    tête := tête↑.suiv;
    libérer (P);
fait;
```

} Peut être formuler en une procédure dont l'entête sera Procédure Libération (Els tête : Pointeur de élém)

b) Ajout des élément à la fin de la liste

1) liste vide : tête := nil;
2) Premier élément : Comme précédemment
3) Création d'un autre élément

```
Allouer (P);
P↑.E := 40;
P↑.suiv := nil
```

} Créer et remplir un nouveau élément

```
Q := tête;
tant que Q↑.suiv <> nil
faire
    Q := Q↑.suiv;
fait;
```

} Avancé jusqu'à la fin de la liste - à la sortie Q pointe vers le dernier élément

```
Q↑.suiv := P;
```

} effectuer le chainage vers le nouveau élément.

On peut voir la boucle de recherche de la fin comme une fonction

```
fonction Rech_Queue (E/ L : pointeur de élément) : pointeur de élém ;
Début
    tant que (L↑.suiv <> nil)
        faire
            L := L↑.suiv ;
        fait ;
    Rech_Queue := L ;
fin ;
```

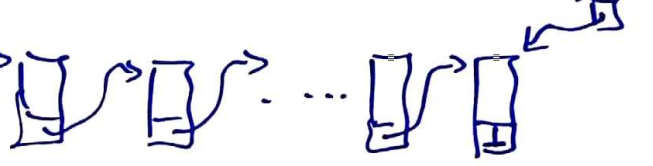
Généralisation de l'Ajout à la fin (FIFO)

```
lire (n) ;
Allouer (P) ;
lire (P↑.E) ;
P↑.suiv := nil ;
tête := P ;
Pour i := 2 à n
    faire
        Allouer (P) ;
        lire (P↑.E) ;
        P↑.suiv := nil ;
        Q := Rech_Queue (tête) ;
        Q↑.suiv := P ;
    fait ;
```

s'il s'élèvent

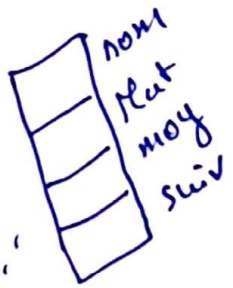
Ajout des éléments de 2 à n

Remarque : il est préférable d'utiliser un autre pointeur comme queue de la liste, utilisé pour localiser le dernier élément



et puis l'insertion à la fin se fait de la manière suivante

```
Allouer (P);
lire (P↑. E);
P↑. suiv := nil;
~~Queu↑. suiv := P;~~
Queu↑. Suiv := P;
Queu := P;
```



Autre forme de déclaration

```
type
    Ptr_list = Pointeur de élém;
    élém = enregistrement (
                nom : chaine (50);
                Mat : entier;
                moy : Réel;
                suiv : Ptr_list;
           )
```

(nouveau type déclaré (au lieu d'écrire à chaque fois Pointeur de élém))

```
Var
    tête, Queu, P, Q : Ptr_list;
```