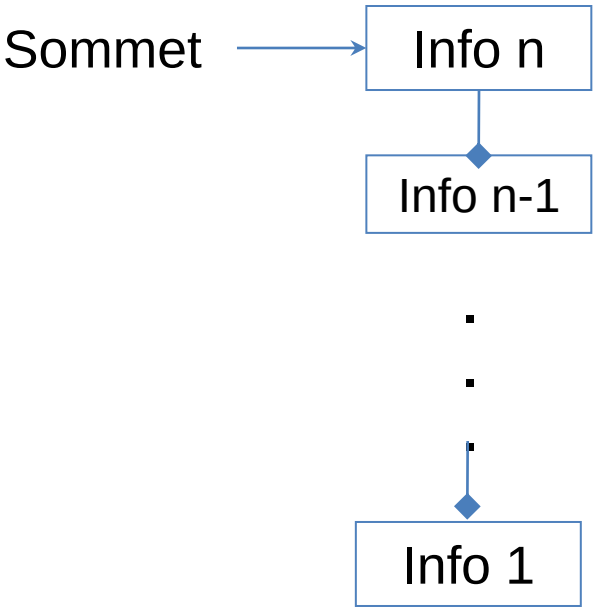
Les Piles et les Files

Les Piles

Dans une pile : l'insertion(**empiler**) et la suppression(**dépiler**) se font à une seule extrémité : le sommet de pile. On parle de structures LIFO : Last In First Out.



Opérations

Init : → Pile {Initialisation de la pile}

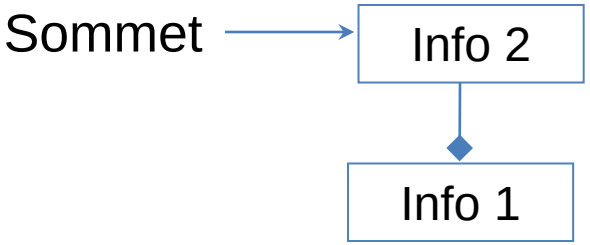
empiler : Pile x Elément → Pile {Ajouter un elt au sommet}

dépiler : Pile → Pile {Retirer un elt du sommet}

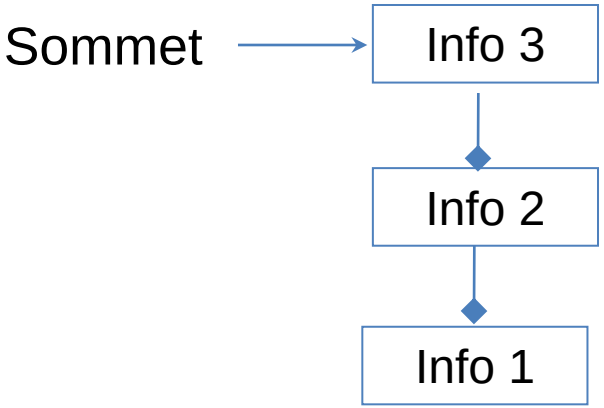
est-vide : Pile → Booléen {vérifier si la pile est vide}

Empilement

Avant l'empilement de info 3 :

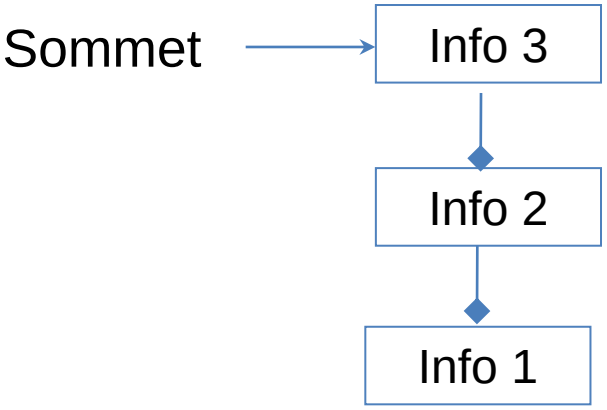


Après l'empilement de info 3 :

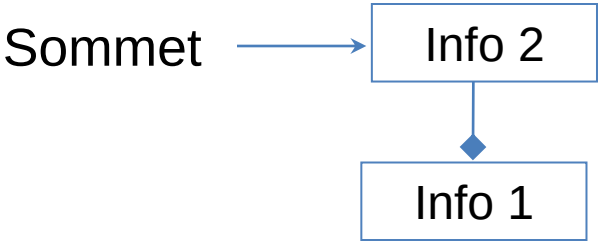


Dépilement

Avant dépilement de info 3 :



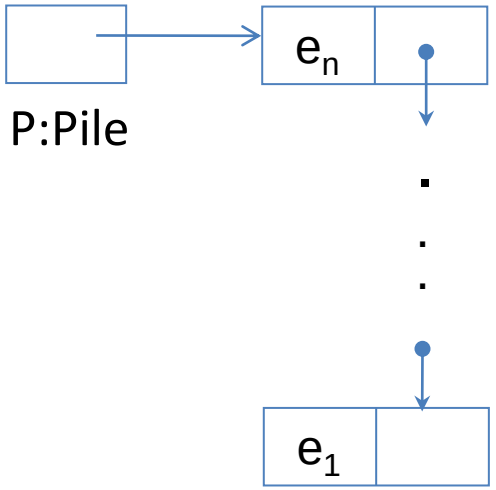
Après dépilement de info 3 :



Représentation des piles

Représentation chaînée : Les différents éléments sont chaînés entre eux. Le sommet est représenté par le 1er élément de la liste (dernier inséré).

La pile vide est donnée par le pointeur NIL.

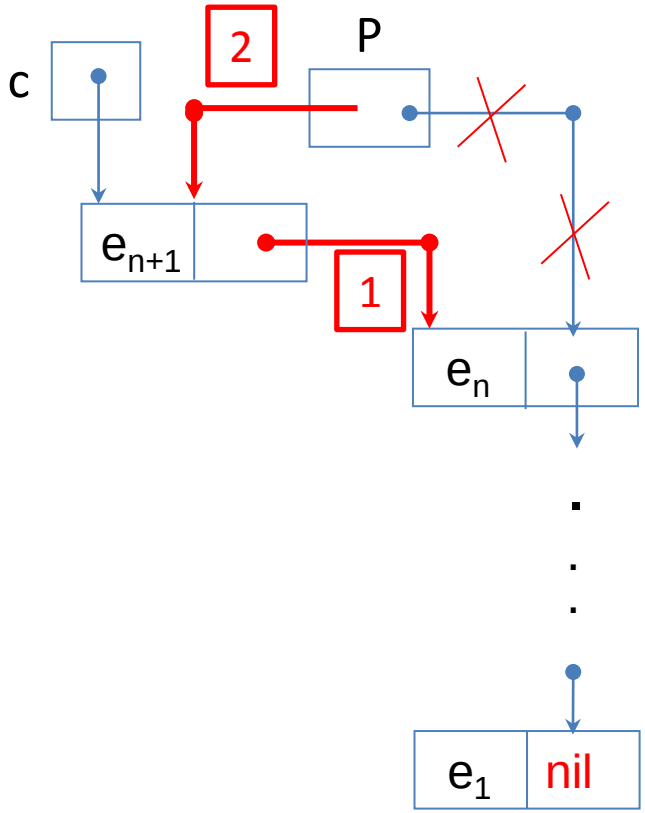



```
Type Pile=^cel;
     cel=record
           val : type_val;
           suiv : pile;
     end;

Var PileD : Pile;
```

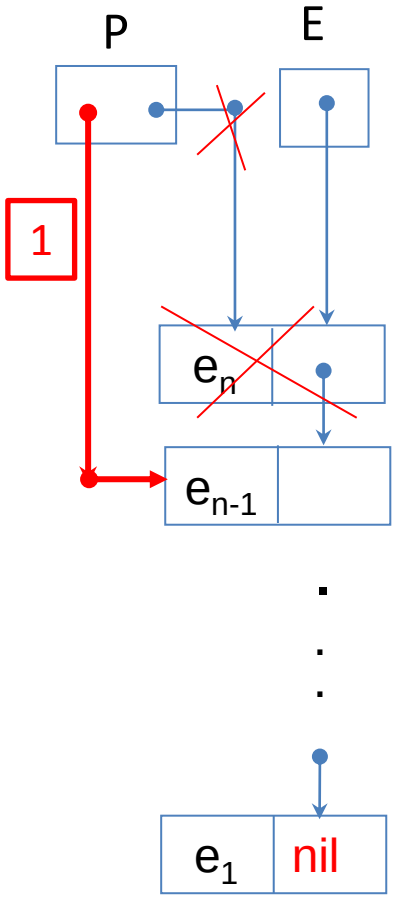
Procédure d'empilement:

```
Procedure empiler (Var P : Pile);
Var c: Pile;
Begin        New(c);
             readln(c^.val);
             c^.suiv:=P;      1
             P:=c;            2
End;
```



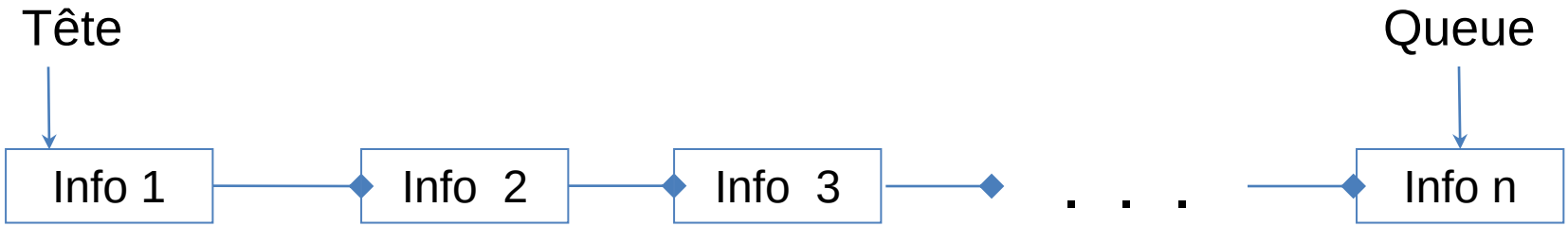
Procédure de dépilement:

```
Procedure dépiler (Var P : Pile);
Var E: Pile;
Begin E:=P;
        If E=nil then
           writeln('Pile vide')
        Else
        Begin
                P:=P^.suiv;   1
                dispose(E);
        End;
End;
```



Les files

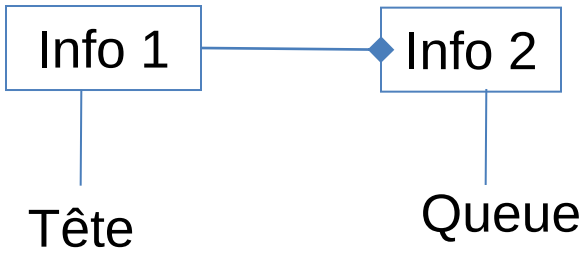
- insertions (ajouts) à une extrémité (queue).
- suppressions(retraits) à l'autre extrémité (tête).
- Analogie avec les files d'attente : FIFO ou First In First Out.



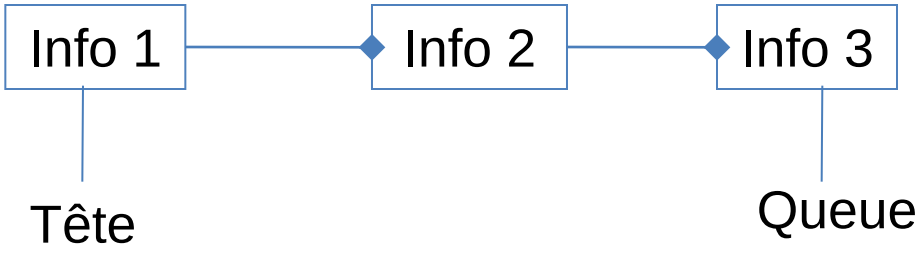
Init : → File
ajouter : File x Elément → File {enfiler}
retirer : File → File {défiler}
premier : File → Elément
est-vide : File → Booléen

Ajout ou Enfilement

Avant l'enfilement de info 3 :

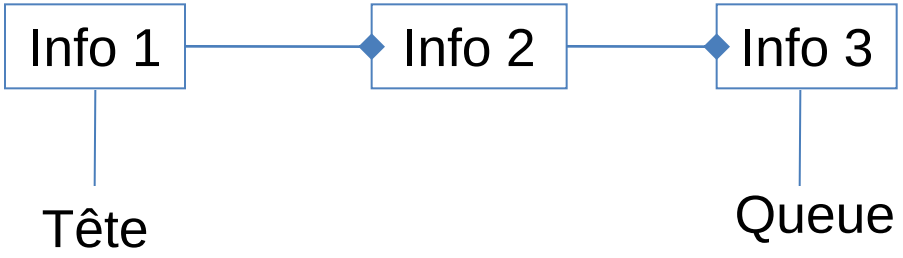


Après l'enfilement de info 3 :

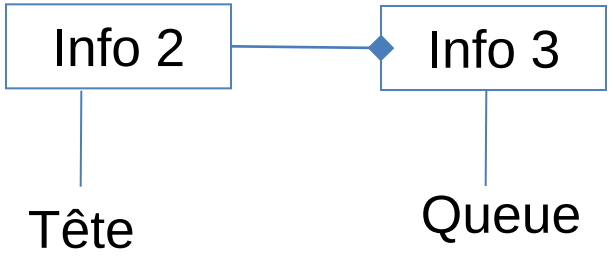


Retrait ou Défilement

Avant défilement de info 1 :



Après défilement de info 1 :



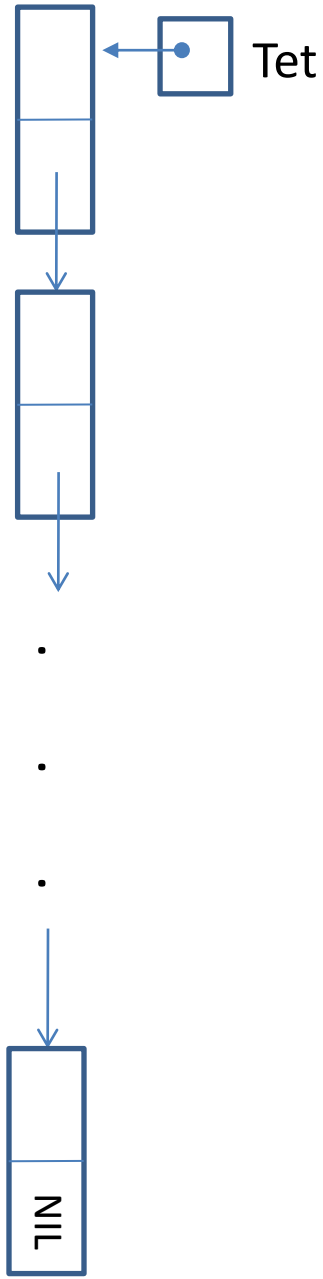
Représentation des Files

Représentation chaînée : Les différents éléments sont chaînés entre eux.

La File vide est donnée par le pointeur NIL.

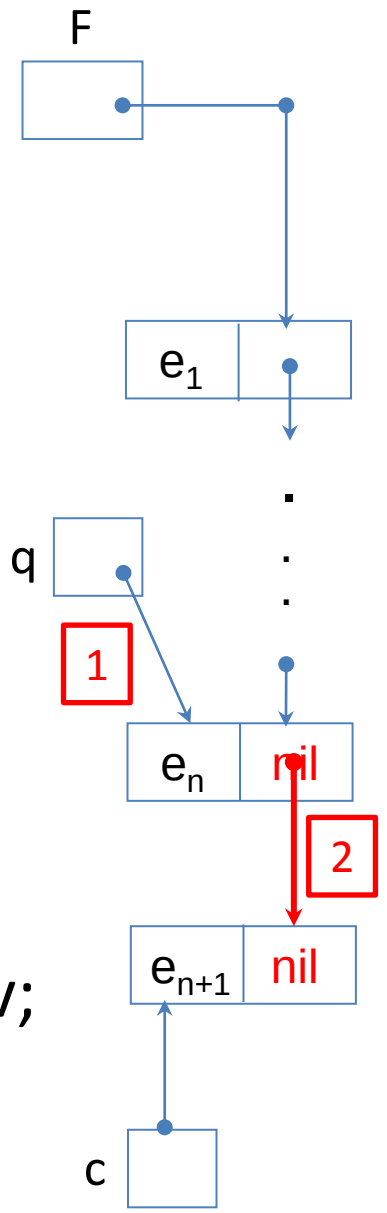
```
Type File=^cel;
     cel=record
          val : type_val;
          suiv : file;
     end;

Var Tet : File;
```



Procédure d'enfilement (ajout):

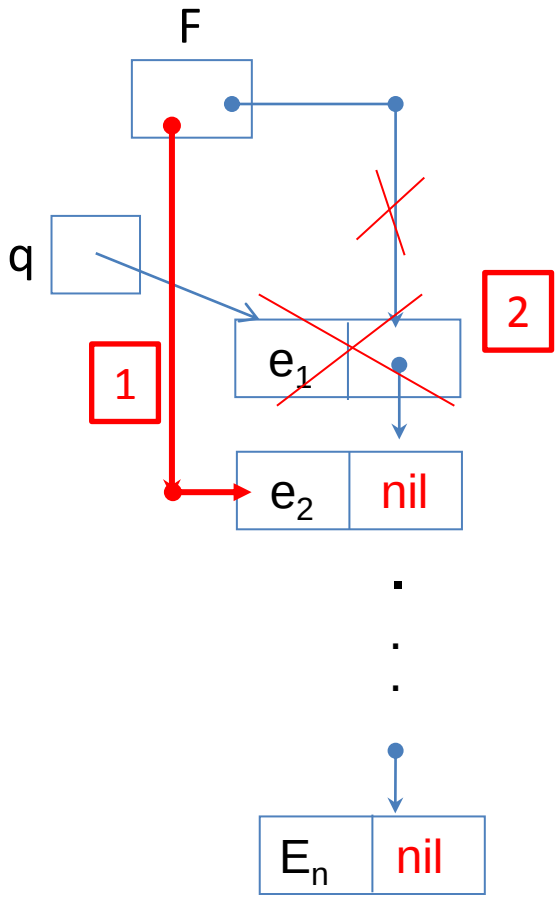
```
Procedure Ajouter (Var F : File);
Var c, q: File;
Begin  New(c);
        readln(c^.val);
        c^.suiv:=nil;
        If F=nil then F:=c
        Else
        Begin  q:=F;
            1  while q^.suiv <> nil do q:=q^.suiv;
                q^.suiv:=c;   2
       end;
End;
```



30/04/2017

Procédure de défilement (Retrait ou suppression):

```
Procedure Retirer (Var F : File);
Var q: File;
Begin   q:=F;
        If q=nil then writeln('File vide')
        Else
        Begin
              F:=F ^.suiv;   1
              dispose(q);    2
        end;
End;
```



Exercice 1 sur Les Piles et les Files

1) Supposons que les éléments A, B, C et D sont insérés dans une pile dans cet ordre. Maintenant on dépile la pile deux fois. Quels éléments sont dépilés et dans quel ordre ?
2) Supposons que les deux éléments que nous avons dépilés ont été insérés dans une file d'attente dans l'ordre dans lequel ils ont été dépilés. Quel est le premier élément à retirer de la file d'attente ?
3) Supposons que nous ayons les éléments A, B, C, D, et que nous les insérons dans une file dans cet ordre. Ensuite, nous retirons le premier élément de la file d'attente, à répétition, jusqu'à ce qu'elle soit vide, en insérant, en même temps, chaque élément enlevé dans une pile. Si nous dépilons, maintenant, tous les éléments de la pile, quel est l'ordre dans lequel les éléments sont dépilés?
4) En tenant compte des résultats ci-dessus, Ecrire un programme qui utilise une pile pour inverser une file.

```
public class QueueReverse {
public static void main( String[] args ) {
LLStack<Integer> stack = new LLStack<Integer>();
LLQueue<Integer> queue = new LLQueue<Integer>();

System.out.println("Original Queue:");

for ( int i = 0; i < 5; ++i ) {
            System.out.println(i);
            queue.insert(i); }

while ( ! queue.isEmpty() )
    stack.push(queue.remove());

while ( ! stack.isEmpty() )
    queue.insert(stack.pop());

System.out.println("\nAfter the reversal:");
while ( ! queue.isEmpty() )
    System.out.println(queue.remove());
}
}
```