- Fonction d'Ackermann - Trace de Ack(2, 1)

$$Ack(m,n) = \begin{cases} n+1, & \text{si } m = 0 \\ Ack(m-1,1), & \text{si } m \neq 0 \text{ et } n = 0 \\ Ack(m-1, Ack(m, n-1)), & \text{sinon} \end{cases}$$

Ack(2,1) = Ack(1,Ack(2,0))

Ack(1,1)

=

5

Ack(0,Ack(1,0))

Ack(0,1)=2

Ack(0,2) = 3

Ack(1,3) = Ack(0, Ack(1,2))

Ack(0,Ack(1,1))

Ack(0,Ack(1,0))

Ack(0,1) = 2

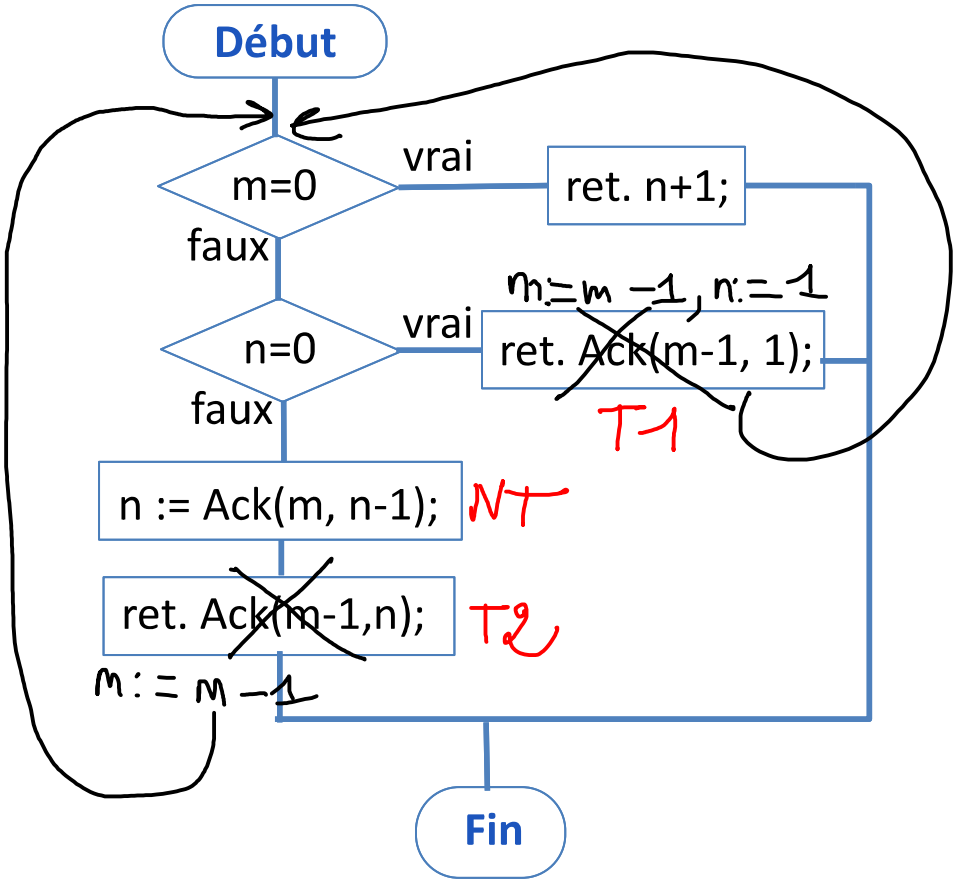
Ack(0,2)=3

Ack(0,3)=4

Ack(0,4) = 5

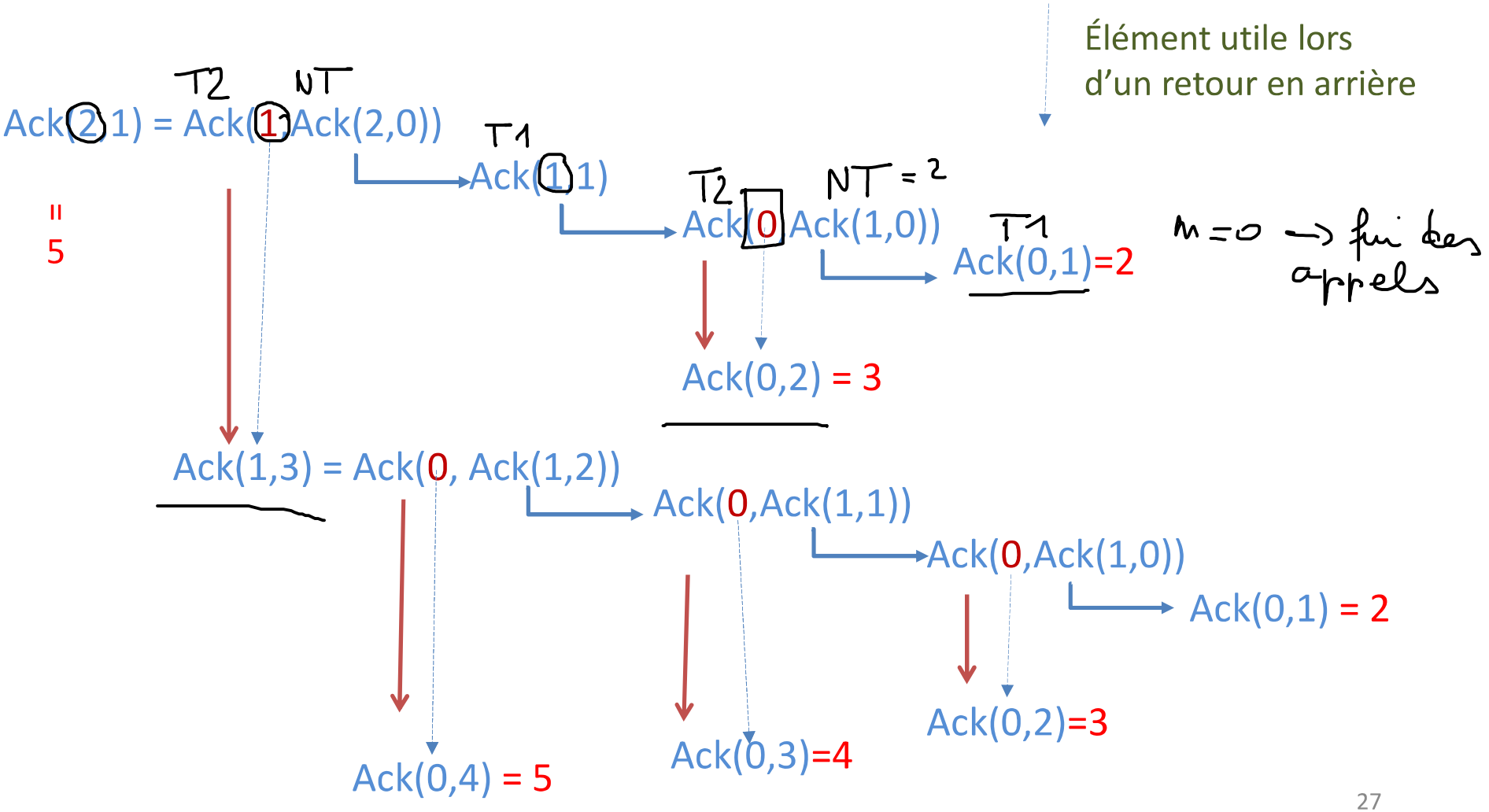
- Fonction d'Ackermann - récursive

```
Fonction Ack(m,n):
    Si m=0 alors
        retourner n+1;
    Sinon
        Si n=0 alors
            retourner Ack(m-1,1);
        Sinon
            n := Ack(m,n-1);
            retourner Ack(m-1, n);
        fsi;
    fsi;
Fin;
```

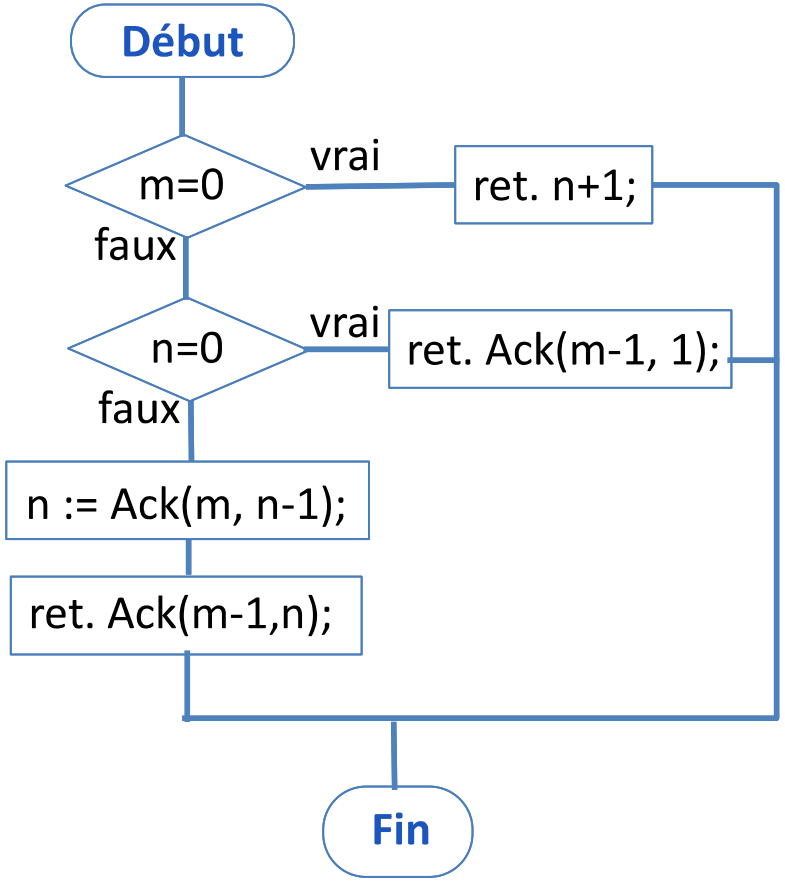


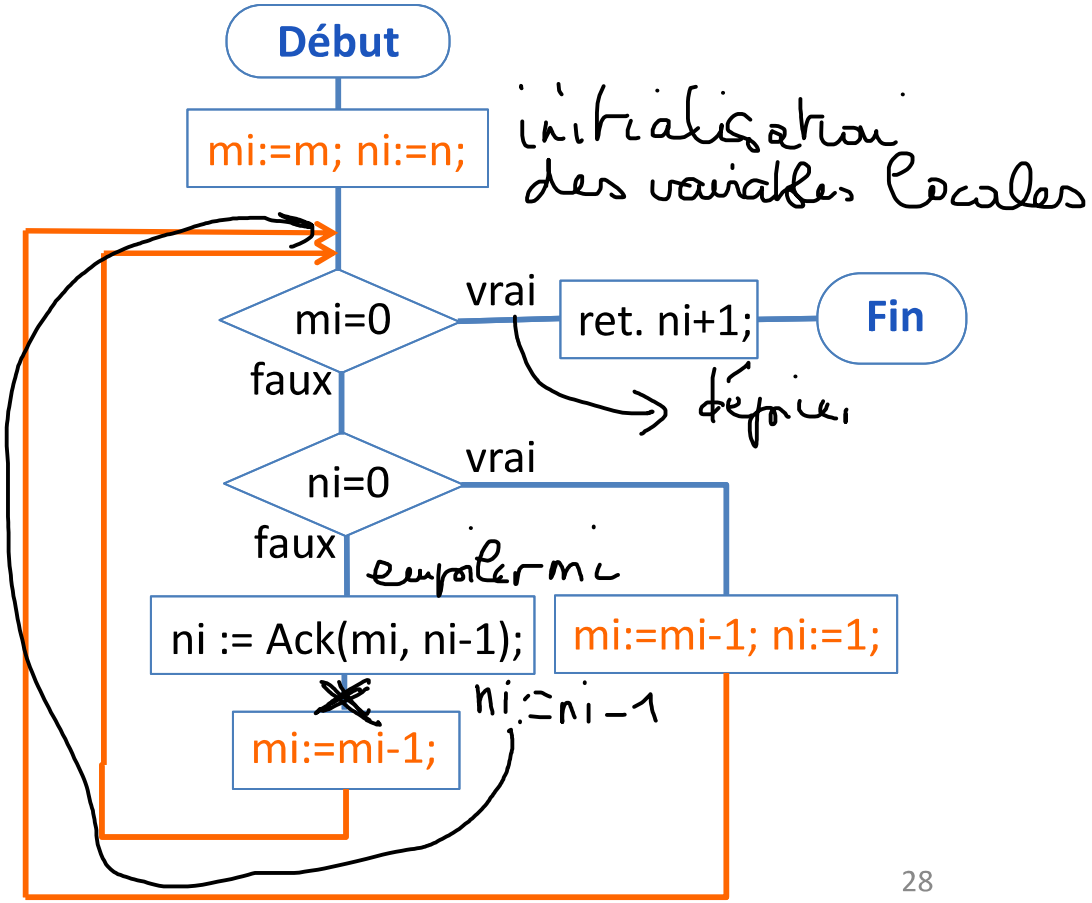
- Fonction d'Ackermann - Trace de Ack(2, 1)

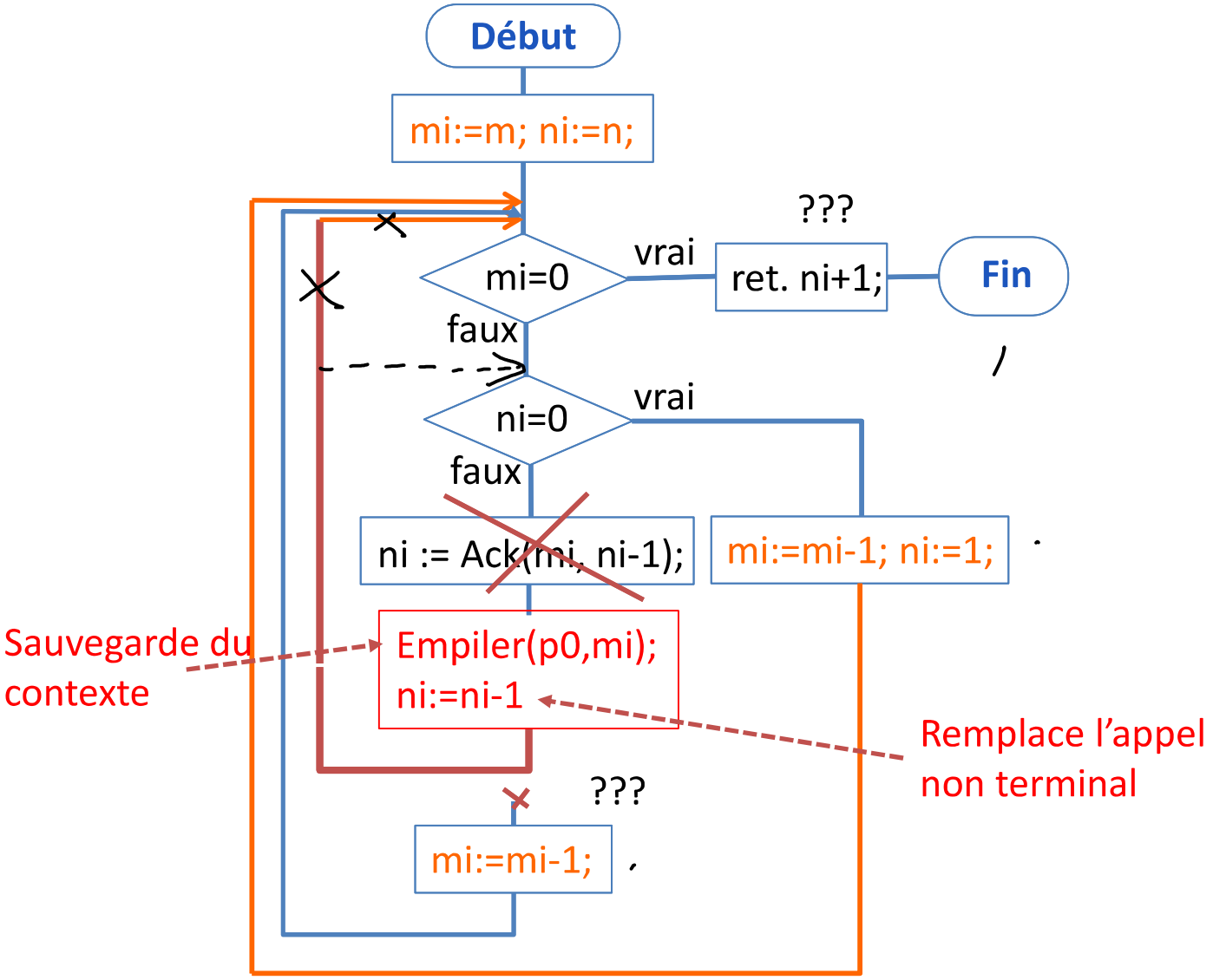
$$Ack(m,n) = \begin{cases} n+1, & \text{si } m = 0 \\ Ack(m-1,1), & \text{si } m > 0 \text{ et } n = 0 \\ Ack(m-1, Ack(m,n-1)), & \text{sinon} \end{cases}$$



- Fonction d'Ackermann – suppression des appels terminaux

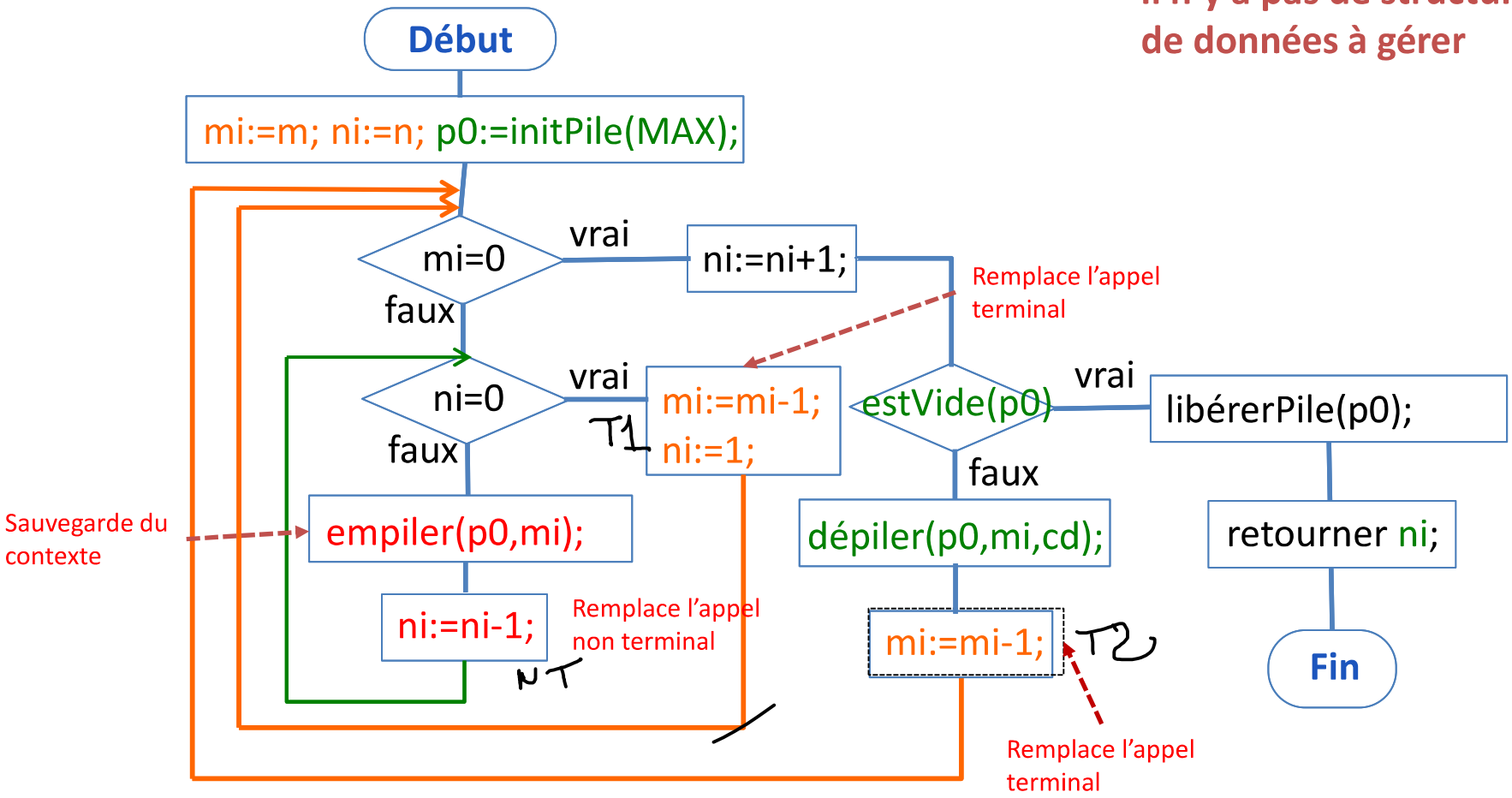





Début
mi:=m; ni:=n;
???
mi=0
vrai
ret. ni+1;
Fin
faux
ni=0
vrai
faux
ni := Ack(mi, ni-1);
mi:=mi-1; ni:=1;
Sauvegarde du contexte
Empiler(p0,mi);
ni:=ni-1
Remplace l'appel non terminal
???
mi:=mi-1;

- Fonction d'Ackermann – suppression de l'appel non terminal

Notation : sans m et cm, il n'y a pas de structure de données à gérer



```
fonction ackinter0(m,n)
mi:=m;ni:=n;
p0:=initPile(MAX);
fin:=faux;

tant que non fin faire

si mi=0 alors ni:=ni+1;
                si non estVide(p0) alors
                    dépiler(p0,mi,cd);
                    mi:=mi-1;
                sinon fin:=vrai;
                fsi;

sinon     si ni=0 alors mi:=mi-1; ni:=1;
          sinon
                empiler(p0,mi, cd);
                ni:=ni-1;
          fsi;
fsi;

fait;
libérerPile(p0);
retourner ni;
```

- Fonction d'Ackermann - itérative

```
Fonction AckIter(m, n):
    mi := m; ni := n;
    p0 := initPile(MAX);
    fin := Faux;
    Tant que NON fin faire
        Si mi≠0 alors
            Tant que ni≠0 faire
                empiler(p0, mi);
                ni := ni-1;
            fait
            mi := mi-1;
            ni := 1;
        Sinon          [ mi=0 ]
            ni := ni+1;
            Si NON estVide(p0) alors
                dépiler(p0, mi, cd);
                mi := mi-1;
            Sinon
                fin := Vrai;
            fsi;
        fsi;
    fait;
    libérerPile(p0)
    retourner ni;
Fin;
```

sauvegarde du contexte

remplace l'appel non terminal

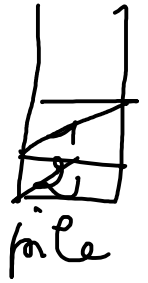
remplace l'appel T1

restauration du contexte

remplace l'appel T2

Trace ack(2,1)

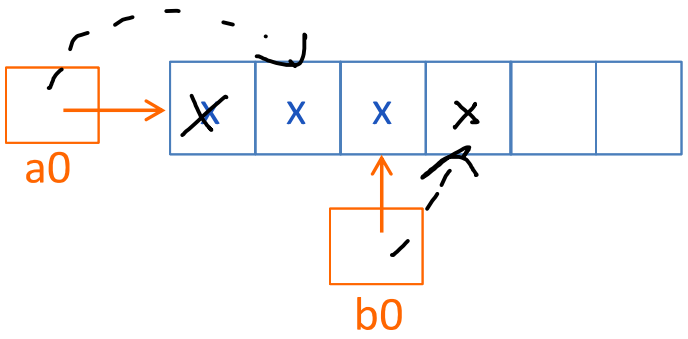
mi	ni
2	1
1	0
	1
0	0
	1
1	2
0	3
2	
1	



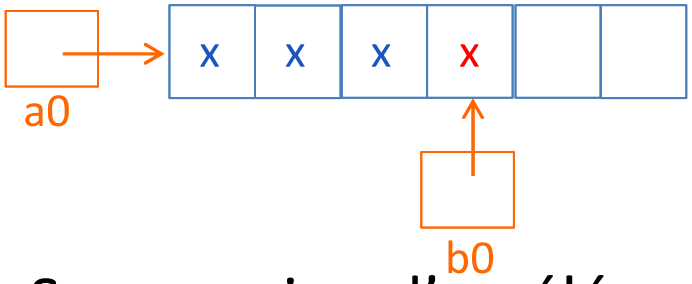
Les files

II.5.5 Les files (FIFO)

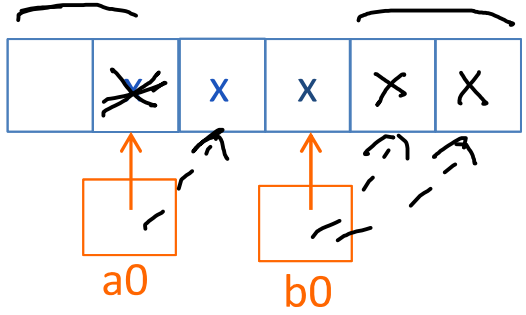
- Adjonction (entrée) en fin de liste; suppression (sortie) en début de liste (politique premier arrivée, premier servi ou FIFO)
- Représentation chaînée possible (plus tard)
- Représentation contiguë avec accès au premier et au dernier



- Ajout d'un élément : m(b0) := cm(b0) + 1; [ou +K]
  m(cm(b0)) := v;



- Suppression d'un élément : m(a0) := cm(a0) + 1; [ou +k]

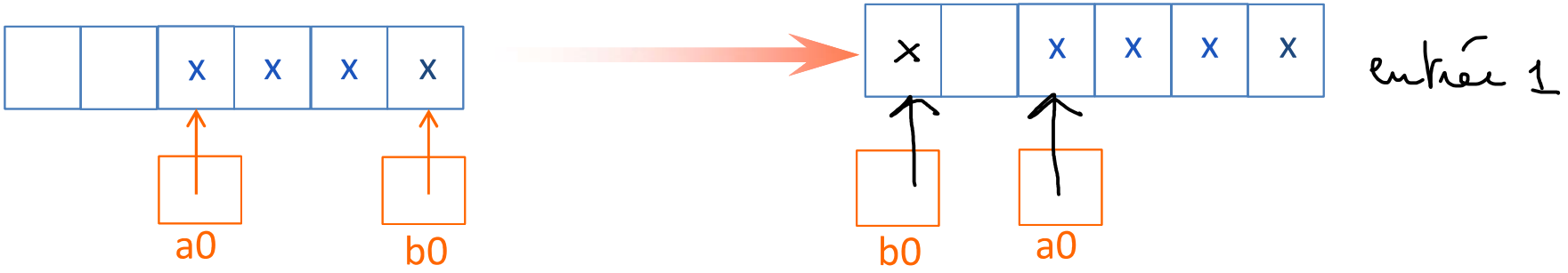


- Places libres en début et en fin de file

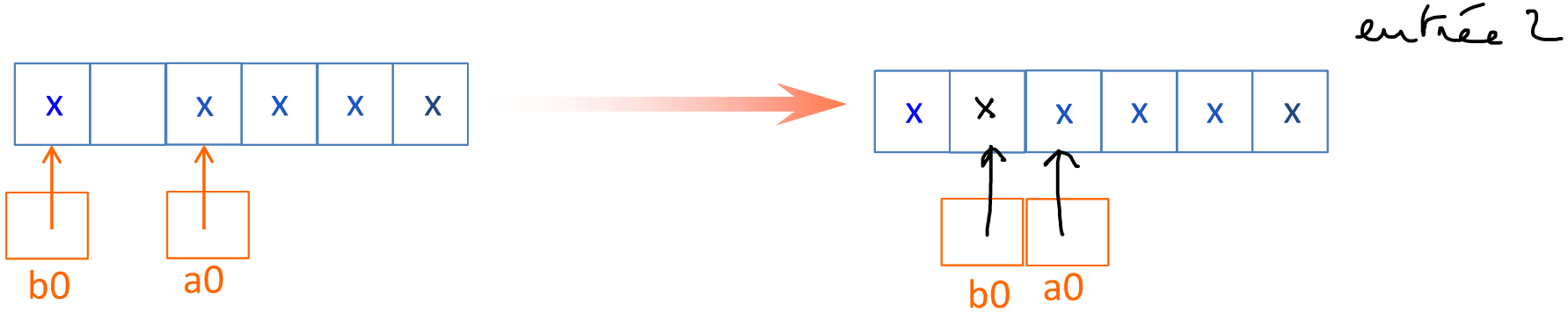
Détection de la file pleine

- Ajout d'un élément : m(b0) := (cm(b0) + 1) mod n;

2 entrées



- File pleine

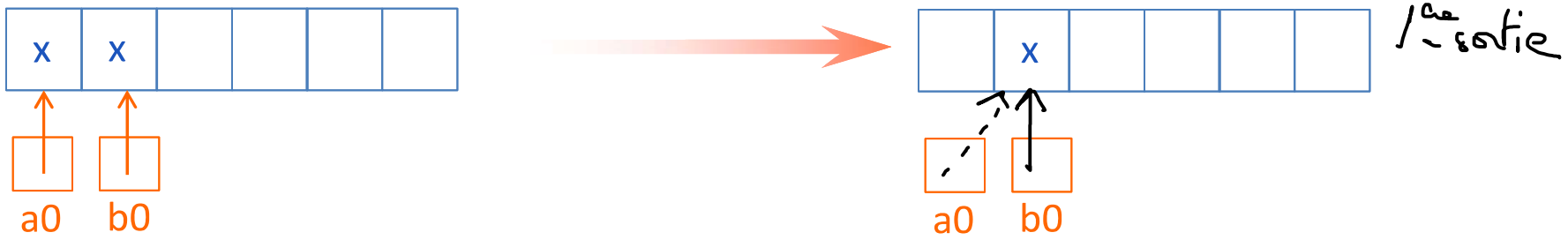


cm(a0) = (cm(b0)+1) mod n

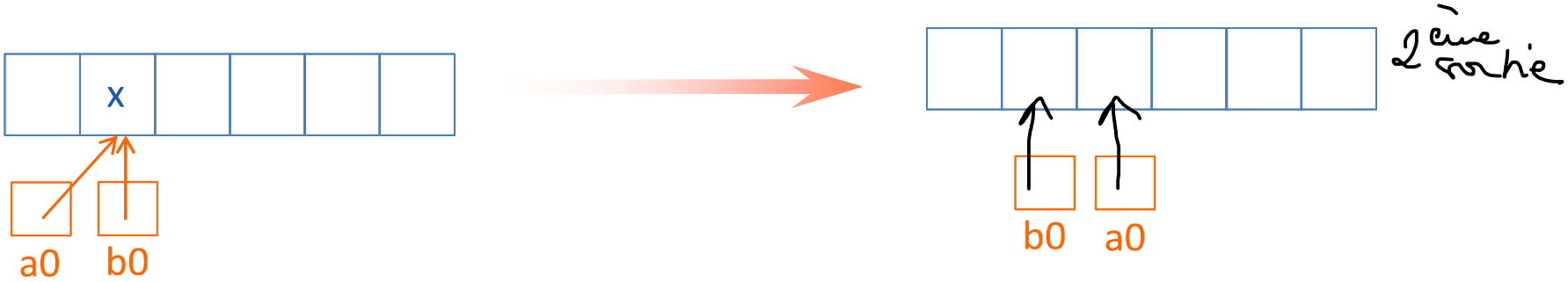
Détection de la file vide

- Suppression d'un élément : m(a0) := (cm(a0) + 1) mod n;

2 sorties



- File vide

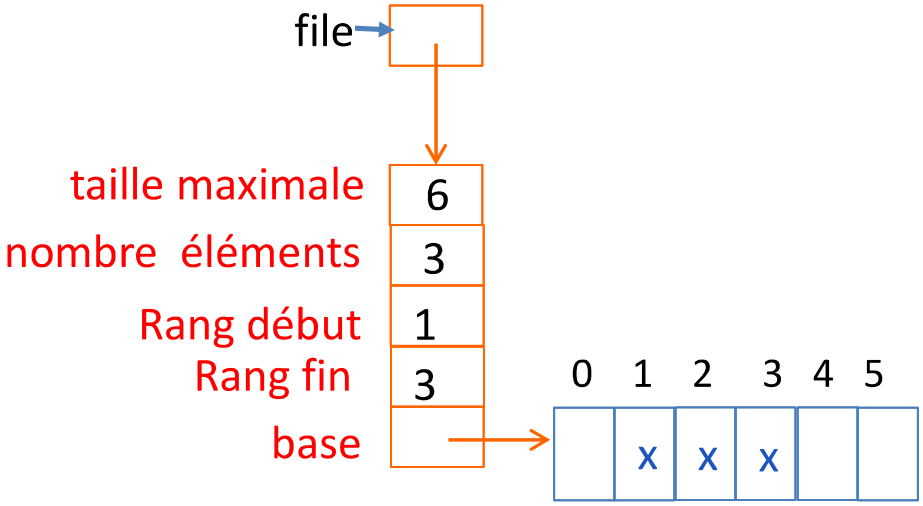


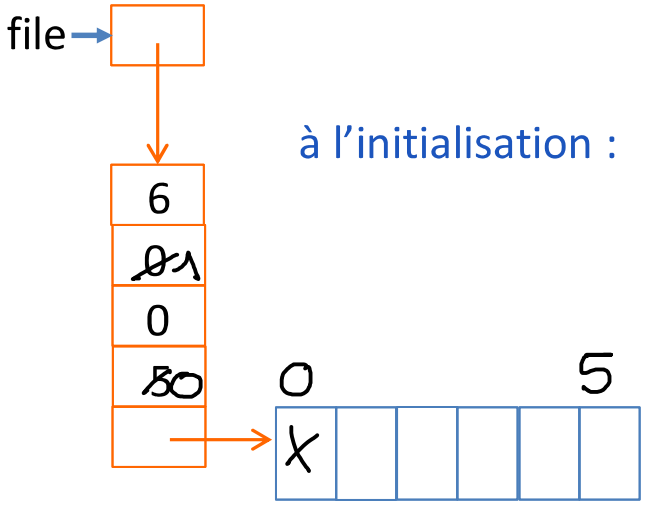
cm(a0) = (cm(b0)+1) mod n

- Remarque : même condition pour une file pleine et une file vide

Solution: gérer un compteur du nombre d'éléments présents dans la file

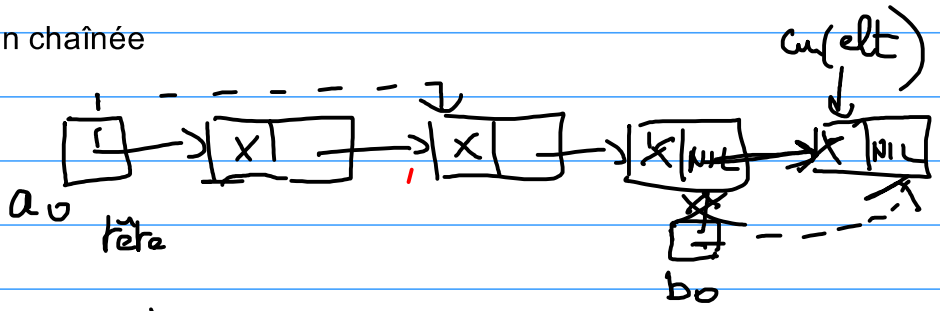
- Structure d'une file





- Opérations de gestion de la file
  - fonction initFile(taille), procédure libérerFile(file)
  - fonction entreeFile(file, v),
  - procédure sortieFile(file,v,code)
  - Fonction estVide(file)
  - Fonction estPleine(file);

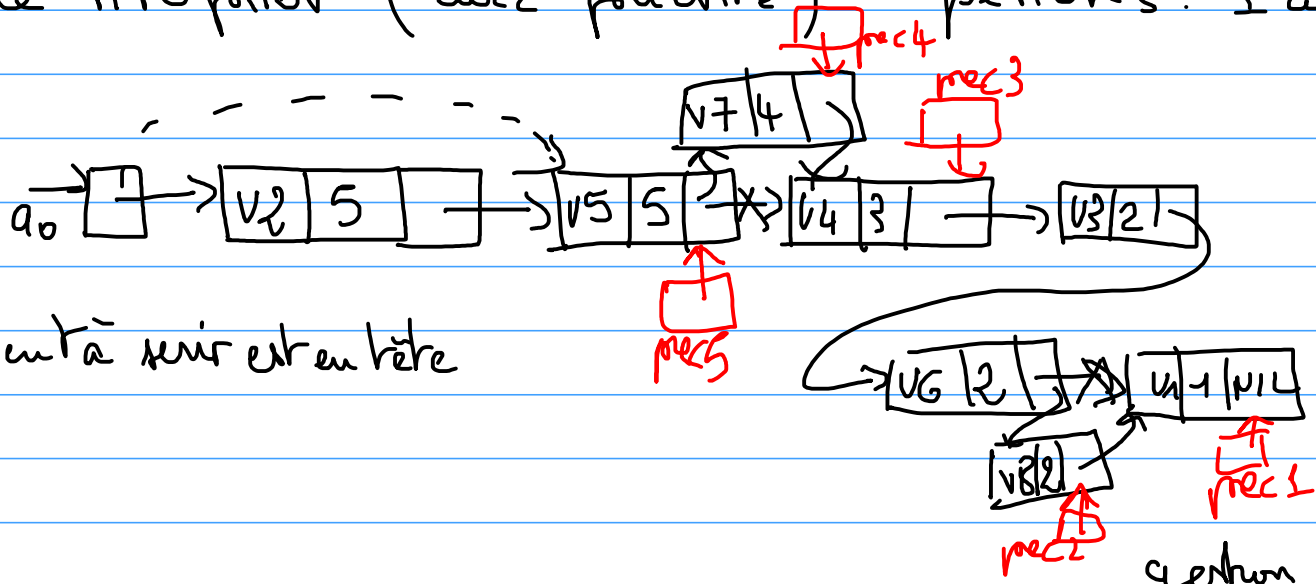
En représentation chaînée



en entrée :
m(elt) := alloc(2) ;
m(cm(elt)) := v ;
ADJ_CELL(bo, elt) ;
m(bo) := cm(elt)+1 ;

une sortie m(v) := cm(cm(a0)) ;
SUP_CEL(a0) ;

file FiFo prioir (avec priorité) priorités : 1 à 5



1e élément à sortir est en tête

V1	1
V2	5
V3	2
V4	3
V5	5
V6	2

gestion d'une entrée : V7 4 V8 2

m(rec) := alloc() ; rec(a0, v)
ADJ_CEL(rec, elt)