

# COURS 10

---

Programmation impérative

## Implémentation de Types Abstraits de données

- Listes
  - Implémentation par liste chaînée
  - Cas particuliers : Piles - Files
- Arbres

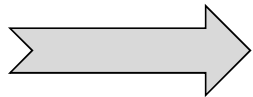
# SOMMAIRE

---

- Informations pratiques
- Introduction
- Éléments de base
  - Programmer en Langage C – Compilation
  - Structure d'un programme / Règles d'écritures
  - Types de base
  - Constantes/Variables
  - Opérateurs
  - Instructions de contrôle
  - Pointeurs
  - Tableaux
- Fonctions
- Chaînes de caractères
- Pointeurs- Tableaux-Fonctions
- Types Construits
- Entrées – Sorties sur Fichiers
- Compilation séparée
- Implémentation de Types Abstraits de Données

# Implémentation de Types Abstraits de Données

- Structures de données algorithmiques
  - Structure de données = brique avec laquelle est construite l'édifice algorithmique
  - Définition de structures de base supportant des fonctions précises
- Listes
  - Cas général
  - Listes particulières
    - Files
    - Piles
- Graphes
  - Arbres



Comment les implémenter en langage C ?

# Implémentation de Listes

- Définition
  - une liste linéaire  $l$  est une suite finie éventuellement vide d 'éléments repérés selon leur rang dans la liste
- Remarques
  - ordre sur les places des éléments et non sur les éléments
  - il existe une fonction de succession **succ** telle que toute place soit accessible en appliquant **succ** de manière répétée à partir de la première place de la liste
- Opérations de base effectuées sur les listes
  - accéder au  $k^{\text{ième}}$  élément
  - insérer un nouvel élément après la  $k^{\text{ième}}$  place
  - supprimer le  $k^{\text{ième}}$  élément

# Implémentation de Listes

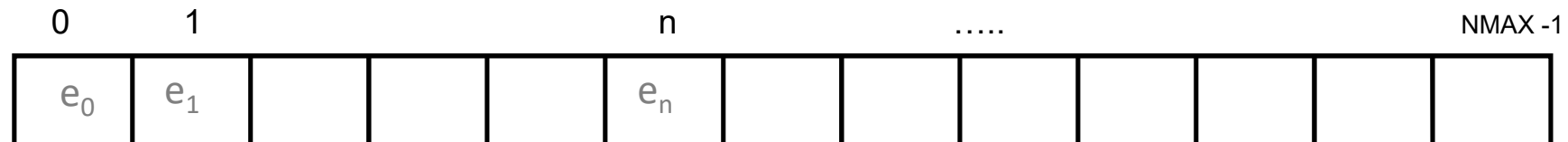
- Représentation contiguë en mémoire

- Liste représentée par un tableau dont la  $i$  ème case est la  $i$  ème place de la liste

- taille du tableau doit être très supérieure à la longueur de la liste pour pouvoir insérer des éléments

⇒ surdimensionnement du tableau

⇒ pour prendre en compte les éléments de la liste et pas toutes les cases du tableau, il faut connaître la longueur de la liste



- l'opération succ est représentée par la succession des cases du tableau en mémoire

# Implémentation de Listes

- Opération d'insertion

- en fin de liste : ex: insérer (l, 6, 'H')

0	1	2	3	4	5							
A	B	C	E	F	G	H						

- en début ou milieu de liste : ex: insérer (l, 3, 'D')

0	1	2	3	4	5	6	7					
A	B	C	E	F	G	H						

0	1	2	3	4	5	6	7					
A	B	C		E	F	G	H					

0	1	2	3	4	5	6	7					
A	B	C	D	E	F	G	H					

# Implémentation de Listes

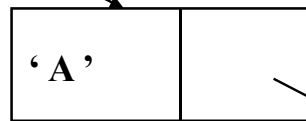
---

- Avantages Représentation contiguë
  - accès direct
  - parcours séquentiel de la liste facile
  - insertion et suppression sur le dernier élément est simple
- Inconvénients Représentation contiguë
  - il faut majorer la taille des listes
  - insertion et suppression ailleurs qu'en fin de liste sont coûteuses

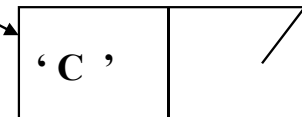
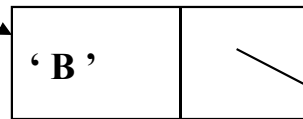
# Implémentation de Listes

- Représentation chaînée

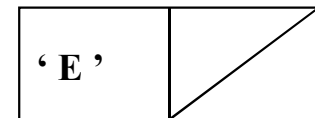
Tête de liste



cellule



Queue de liste

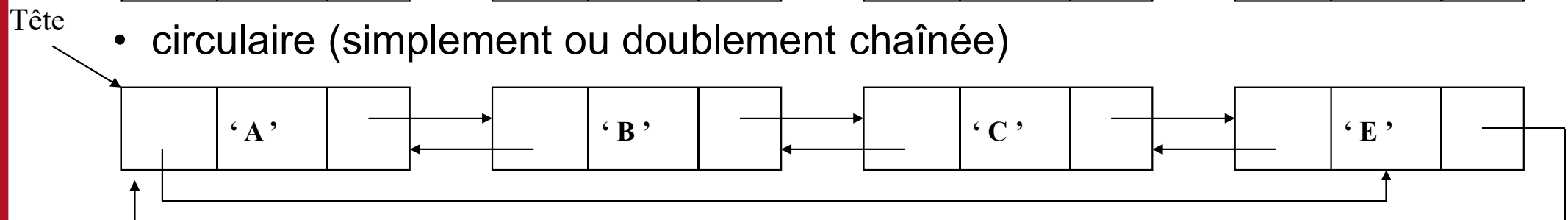
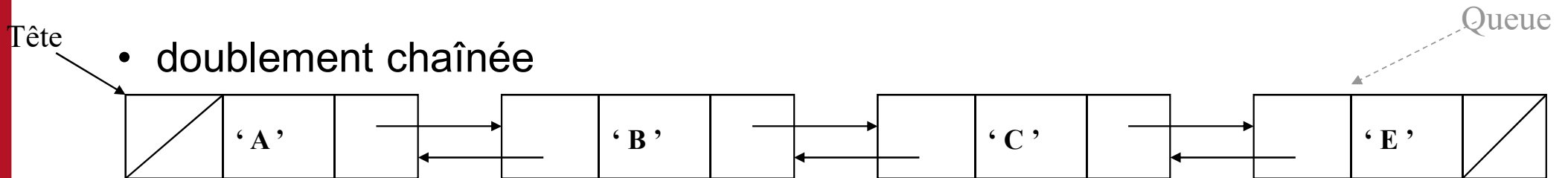
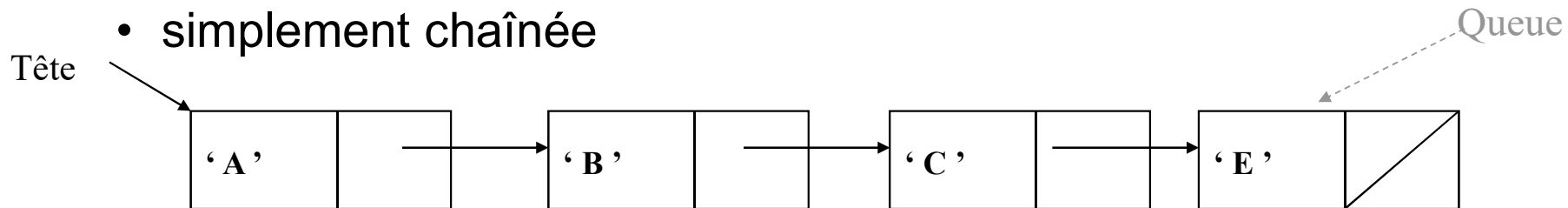


- Tête de liste = adresse de la 1<sup>ère</sup> cellule de la liste
- une cellule contient au minimum 2 champs:
  - un champ **info** qui est l'élément stocké dans la liste
  - un champ **pointeur** qui contient l'adresse de la prochaine cellule
  - fin de liste est matérialisée par champ **pointeur = NULL**



# Implémentation de Listes

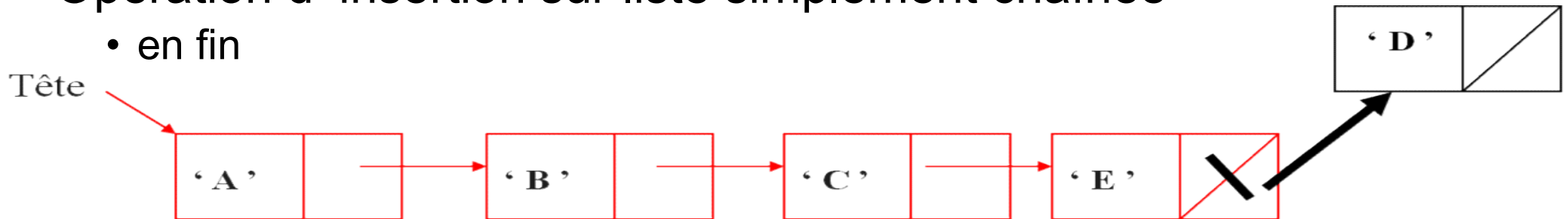
- Il existe plusieurs types de listes chaînées
  - simplement chaînée



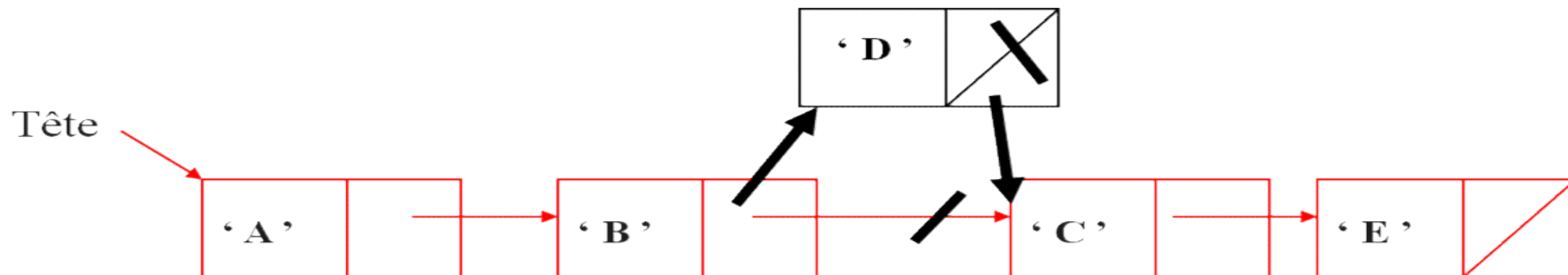
# Implémentation de Listes

- Opération d'insertion sur liste simplement chaînée

- en fin

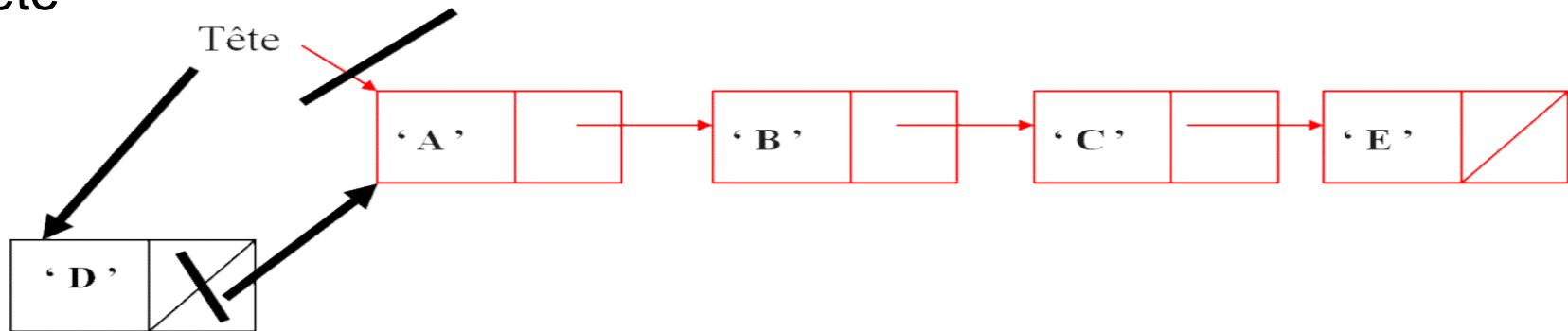


- en milieu

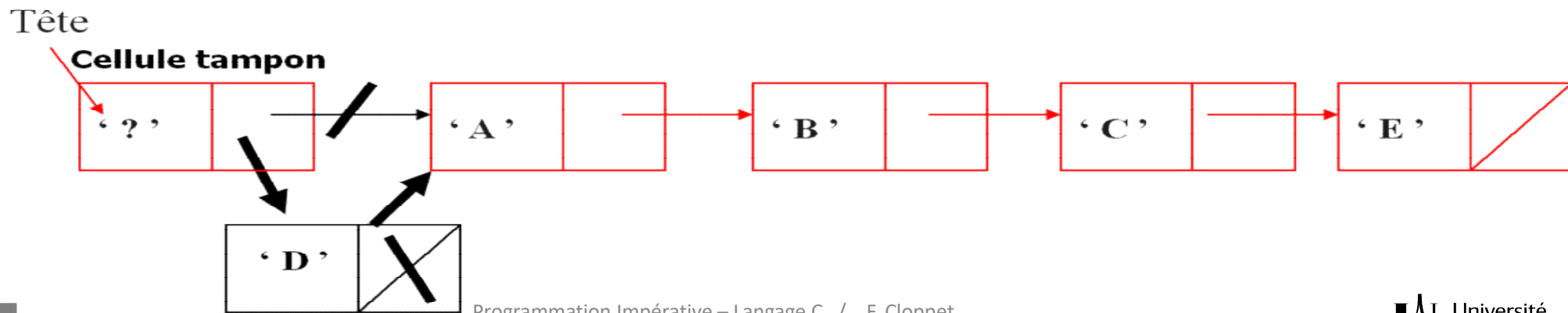


# Implémentation de Listes

- En tête



- Choix d'implémentation pour supprimer le cas insertion en tête de liste



# Implémentation de Listes

- Avantages Représentation chaînée
  - permet de faire évoluer la liste en fonction des besoins de l'application  
⇔ pas de surdimensionnement
  - Cases mémoires ne sont pas nécessairement contiguës
    - On optimise l'espace mémoire réservé
  - insertion ou suppression sont peu coûteuses quelle que soit la place où elles ont lieu
- Inconvénients Représentation chaînée
  - pas d'accès direct ⇔ parcours peut être relativement coûteux
- Remarque:
  - faire attention de ne jamais perdre le point d'entrée dans la liste
    - Préférer le choix d'une implémentation avec une cellule tampon

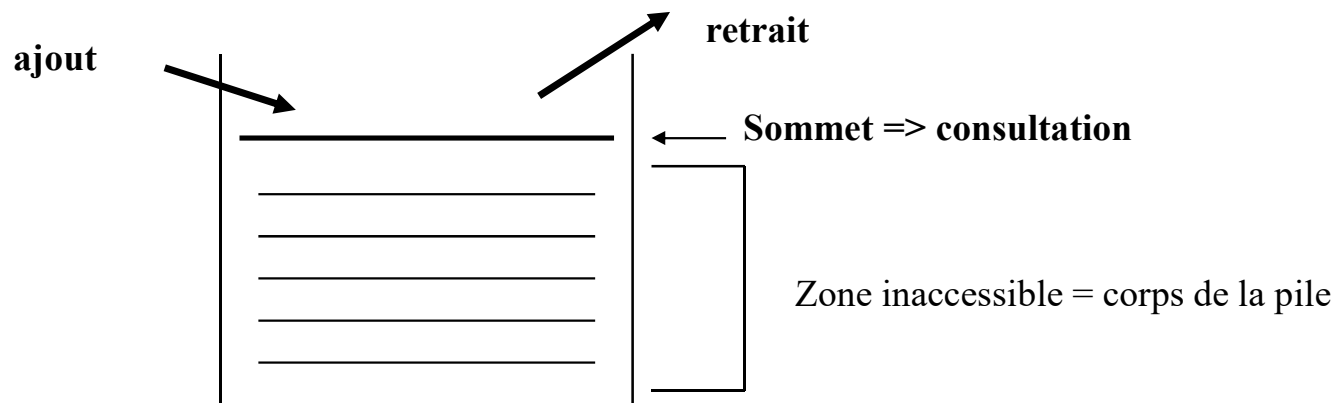
# Implémentation de Listes

---

- Complexité : Représentation chaînée
  - Espace mémoire occupé  $O(n)$  si liste a  $n$  élt
  - Opérations successeur, prédécesseur (si liste doublement chaînée), insertion, suppression sont effectuées en temps constant
    - 👉 1 liste peut être créée et parcourue en  $O(n)$  si  $n$  est la longueur de la liste
  - Si pointeur sur 1er et dernier élt, les opérations de concaténation et partition sont effectuées en temps constant

# Implémentation de Listes

- Cas particuliers Listes
  - Pile = structure de type LIFO (Last In First Out)



- Insertions et suppressions se font à une seule extrémité de la liste  
= sommet de la pile


# Implémentation de Listes

- Cas particuliers Listes

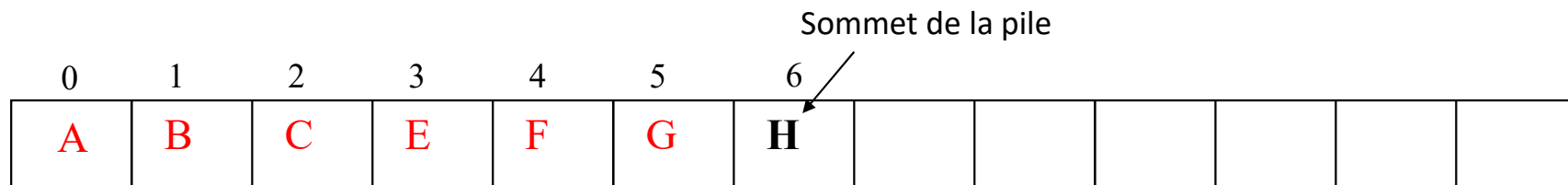
- Pile = structure de type LIFO (Last In First Out)

- Implémentation sous forme de tableau

-  Insertion et suppression sont simples (pas de décalage)

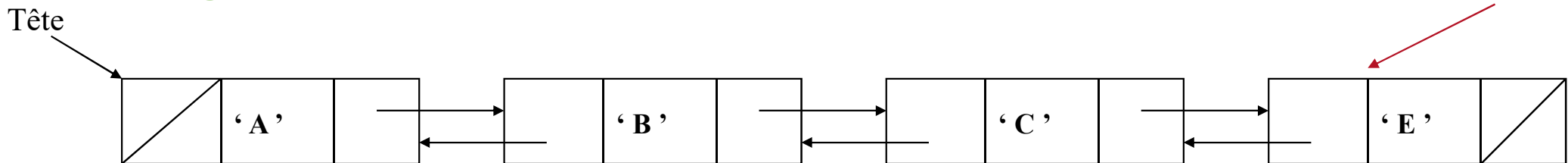
-  Accès direct au sommet de la pile

-  Surdimensionnement nécessaire du tableau



# Implémentation de Listes

- Cas particuliers Listes
  - Pile = structure de type LIFO (Last In First Out)
    - Implémentation sous forme de liste chaînée
      - 😊 Insertion et suppression sont simples
      - 😊 Accès direct au sommet de la pile
      - 😊 Pas de surdimensionnement nécessaire

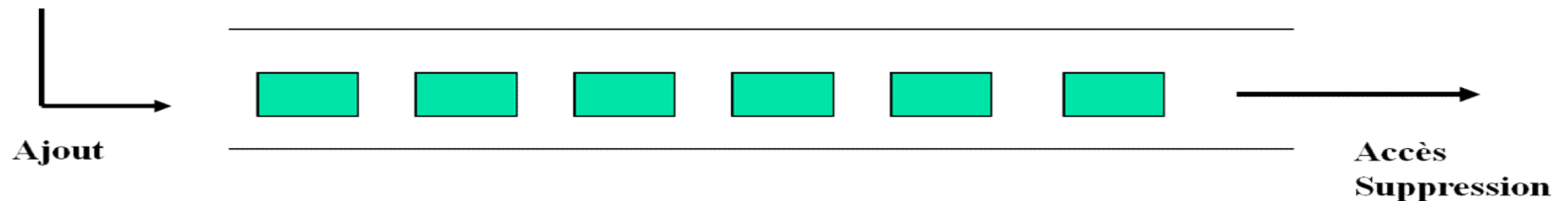




# Implémentation de Listes

- Cas particuliers Listes

- File = structure de type FIFO (First In First Out)
  - Les insertions se font à une extrémité
  - Les accès et les suppressions se font à l'autre extrémité de la liste



# Implémentation de Listes

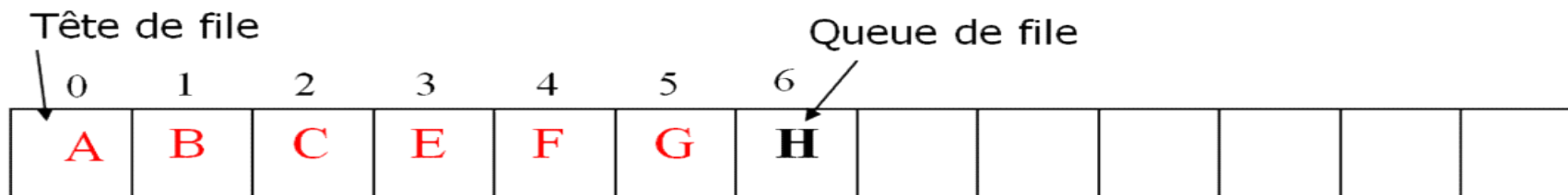
- Cas particuliers Listes

- File = structure de type FIFO (First In First Out)
  - Implémentation sous forme de tableau

😊 Insertion et suppression sont "simples" (pas de décalage) si insertion en fin de liste, et suppression en tête de liste

😊 Accès direct à la tête et à la queue de la file

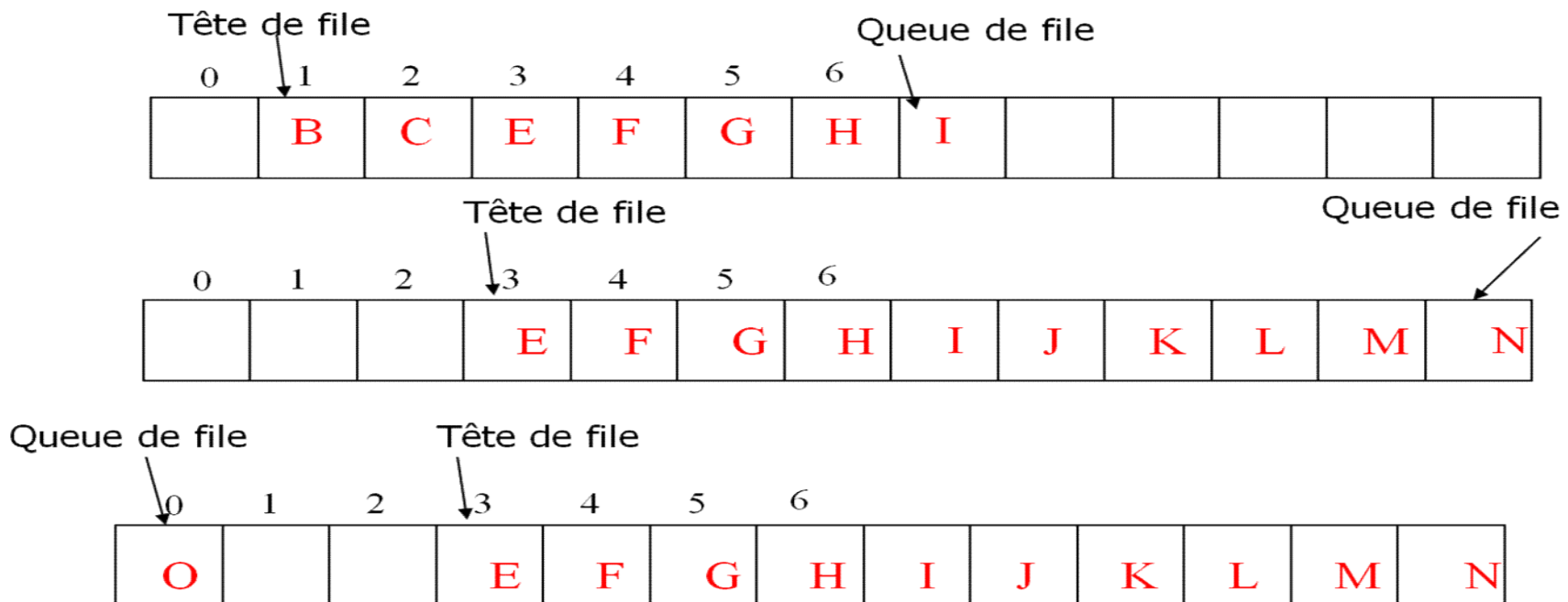
😞 Surdimensionnement nécessaire du tableau



# Implémentation de Listes

## • Cas particuliers Listes

- File = structure de type FIFO (First In First Out)
  - Implémentation sous forme de tableau



# Implémentation de Listes

## • Cas particuliers Listes

- File = structure de type FIFO (First In First Out)
  - Implémentation sous forme de liste chaînée
    - 😊 Insertion et suppression sont simples

😊 Accès direct aux extrémités de la file

😊 Pas de surdimensionnement nécessaire

Tête  
de la file

