Les Piles et Files

Introduction

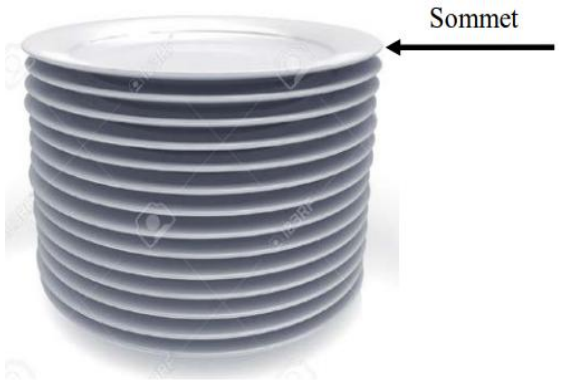
- Ce sont des structures de données linéaires
- Ce sont des structures de données ordonnées, mais qui ne permettent l'accès qu'à une seule donnée.
- Elles sont très souvent utiles, elles servent à mémoriser des évènements en attente de traitement.

La Pile

Les Piles

Définition d'une Pile (Stack)

- La Pile est une structure de données linéaire qui suit le principe LIFO (Last In First Out) → le dernier élément ajouté est le premier à sortir.



Les Piles

Exemple d'application d'une Pile (Stack):

- Gestion du retour arrière (Undo)
- Navigation dans un navigateur web
- Pile d'appel des fonctions (Call Stack)
- Système d'annulations dans les jeux
- ….

Les Piles

Les primitives d'une pile :

- **Empiler** : Ajouter un élément (push)
- **Dépiler** : Enlever un élément (pop)
- **Consulter** : Voir l'élément accessible au sommet (peek)
- **EstVide** : Tester si la pile est vide
- **EstPleine** : Tester si la pile est pleine (que pour les tableaux)

Les Piles

Implémentation d'un Pile :

- Il n'y a pas de structures spécifiques prévues dans tous langages de programmation, il faut donc les créer de toute pièce.

- Pour les piles, on utilisera :

  - Un tableau unidimensionnel en cas de piles de hauteur maximale prévisible

  - Une liste en cas de longueur très variable.

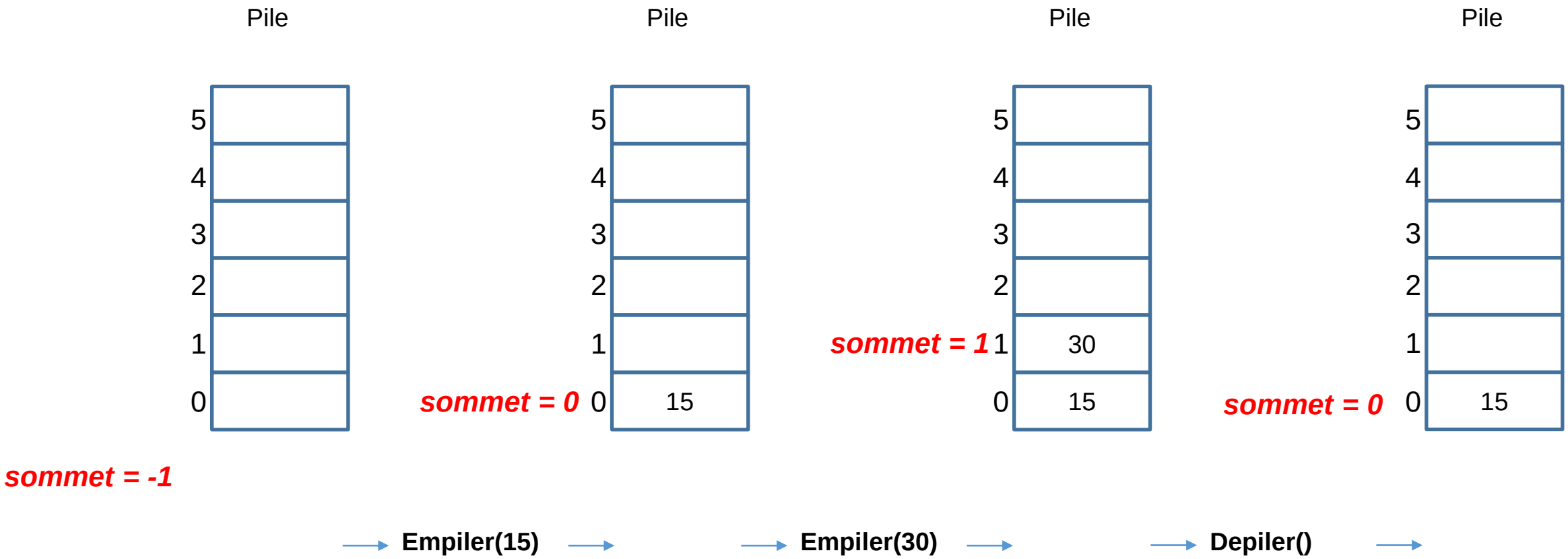
Les Piles

Implémentation d'un Pile : Par un tableau

- Une pile sera implémentée par un tableau avec une taille fixe, et une variable entière "sommet" qui contient l'indice du dernier élément de la pile.
- Pour une pile vide : **sommet = -1**
- Pour faciliter la gestion le tableau et l'entier seront mit dans un type enregistrement

Les Piles

Implémentation d'un Pile : Par un tableau



Les Piles

Implémentation d'un Pile : Par un tableau

- Modéliser une pile avec une structure :

```
#define MAX 500

typedef struct {

    int data[MAX];
    int sommet;

} Pile;
```

Les Piles

Implémentation d'un Pile : Par un tableau

- Les primitives :

```c
int estVide(Pile* p) {
    return p->sommet == -1;
}

int estPleine(Pile* p) {
    return p->sommet == MAX - 1;
}
```

```c
int consulter(Pile* p) {
    if (estVide(p)) {
        exit(1); // Échec de l'accès au sommet
    }
    return p->data[p->sommet];
}
```

Les Piles

Implémentation d'un Pile : Par un tableau

- Les primitives :

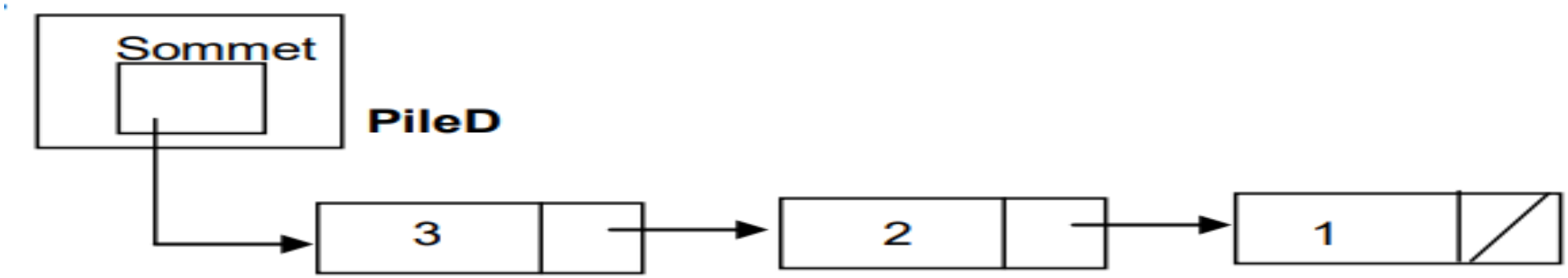
```
void empiler(Pile* p, int val) {
    if (estPleine(p)) {
        printf("Pile pleine");
        exit(1);
    }
    p->sommet++;
    p->data[p->sommet] = val;
}
```

```
int depiler(Pile* p) {
    if (estVide(p)) {
        printf("Pile vide");
        exit(1);
    }
    int val = p->data[p->sommet];
    p->sommet--;
    return val;
}
```

Les Piles

Implémentation d'un Pile : Avec une liste chaînée

- Une pile dynamique est une liste à laquelle on attache un pointeur sommet. C'est un enregistrement à une seule case : pointeur qui pointe la dernière valeur traitée dans la liste (sommet).



Les Piles

Implémentation d'un Pile : Avec une liste chaînée

Les primitives :

- Empiler : Ajouter un élément au début
- Dépiler : Supprimer le premier élément
- Consulter : Retourner la valeur du premier élément
- EstVide : Tester si la liste est vide

HESTIM
ENGINEERING & BUSINESS SCHOOL

La File

Les Files

Définition d'une File

- La File est une structure de données linéaire qui suit le principe FIFO (First In First Out) → le premier élément ajouté est le premier à sortir.

Les Files

Exemple d'application d'une File :

- File d'attente dans un service (banque, hôpital, guichet)
- Gestion des impressions (spooler d'impression)
- Gestion des requêtes dans un serveur web
- Simulation de circulation / gestion d'un pont, tunnel, péage
- ….

Les Files

Les primitives d'une file :

- **Enfiler** : Ajouter un élément
- **Défiler** : Enlever un élément
- **Consulter** : Voir l'élément prochain à sortir
- **EstVide** : Tester si la file est vide
- **EstPleine** : Tester si la file est pleine (que pour les tableaux)

Les Files

Implémentation d'une File :

- Il n'y a pas de structures spécifiques prévues dans tous langages de programmation, il faut donc les créer de toute pièce.

- Pour les files, on utilisera :

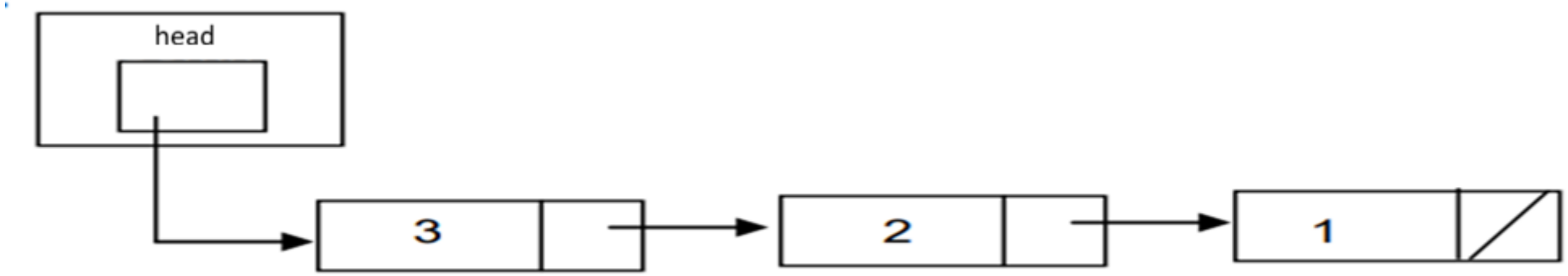
  - Une liste en cas de longueur très variable.

  - Un tableau unidimensionnel en cas de files de hauteur maximale prévisible

Les Files

Implémentation d'un File : Avec une liste chaînée

- Une file dynamique est une liste à laquelle on attache un pointeur sommet. C'est un enregistrement à une seule case : pointeur qui pointe la dernière valeur traitée dans la liste (tête).



Les Files

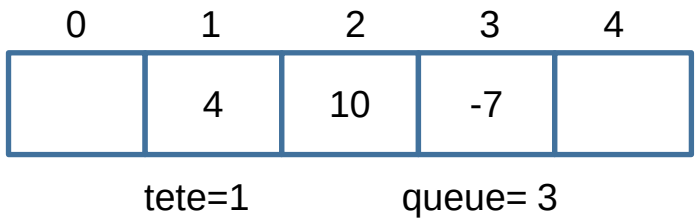
Implémentation d'un File : Avec une liste chaînée

Les primitives :

- Enfiler : Ajouter un élément à la fin
- Défiler : Supprimer le premier élément
- Consulter : Retourner la valeur du premier élément
- EstVide : Tester si la liste est vide

Les Files

Implémentation d'un File : Par un tableau

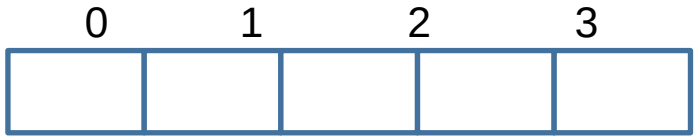


- Une file sera implémentée par un tableau avec une taille fixe, et deux variables entières "**tete**" qui contient l'indice de l'élément à la tête de la file et "**queue**" qui contient l'indice du dernier élément de la file.

Les Files

Implémentation d'un Pile : Par un tableau

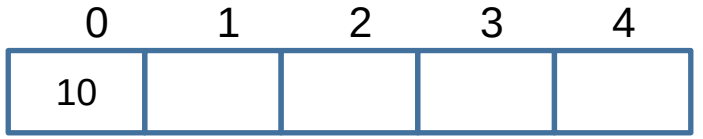
File



queue= -1 tete=0

Enfiler(10)

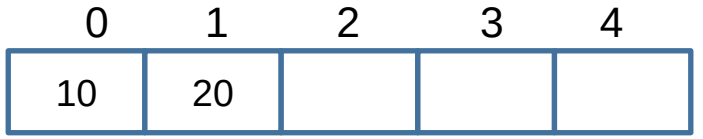
File



tete =0

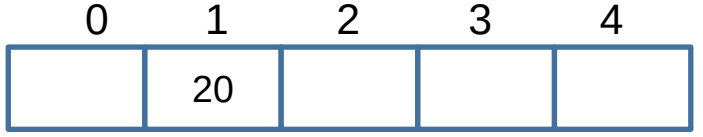
queue =0

Enfiler(20)



tete =0 queue =1

Defiler()



tete =1

queue =1

Defiler()

queue =1 tete =2

Les Files

Implémentation d'un File : Par un tableau

- Modéliser une file avec une structure :

```
#define MAX 500

typedef struct {

    int data[MAX];
    int tete;
    int queue;
    int taille;

} File;
```

Les Files

Implémentation d'un Pile : Par un tableau

- Les primitives :

```
void initialiser(File* f) {
    f->tete = 0;
    f->queue = -1;
    f->taille = 0;
}

int estVide(File* f) {
    return f->taille==0;
}

int estPleine(File* f) {
    return f->taille == MAX;
}
```

```
int consulter(File* f) {
    if (estVide(f)) {
        printf("Erreur: La file est vide.\n");
        exit(1); // File vide
    }
    int valeur = f->data[f->tete];
    return valeur;
}
```

Les Files

Implémentation d'un File : Par un tableau

- Les primitives :

```
void enfiler(File* f, int valeur) {
    if (estPleine(f)) {
        printf("La file est pleine.\n");
        exit(1); // File pleine
    }
    f->queue = ( f->queue + 1 ) % MAX;
    f->data[f->queue] = valeur;
    f->taille++;
}
```

```
int defiler(File* f) {
    if (estVide(f)) {
        printf("La file est vide.\n");
        exit(1); // File vide
    }
    int valeur = f->data[f->tete];
    f->tete = (f->tete + 1) % MAX;
    f->taille--;
    return valeur;
}
```