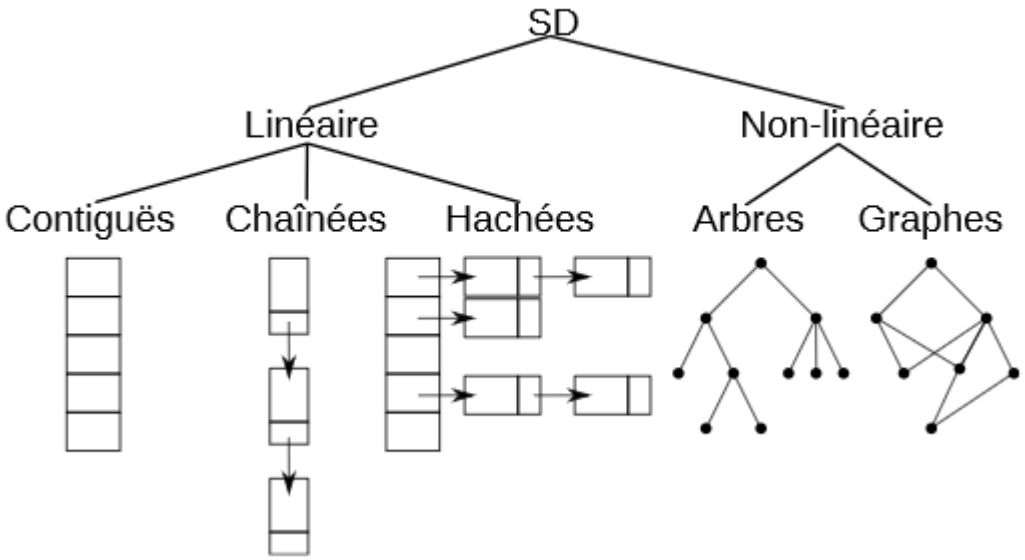
Développement informatique

Cours 6 : structures de données

Classification des structures de données



Pile et File

Une pile est une séquence dans laquelle on ne peut ajouter et supprimer un élément qu'a une seule extrémité : le«sommet» de la pile

Une file est une séquence dans laquelle on ne peut ajouter un élément qu'a une seule extrémité et ne supprimer un élément qu'a l'autre extrémité : la«tête»de la file.

Pile et File

Les Piles (Stack , LIFO)

Structure de données fondée sur le principe « dernier arrivé, premier sorti » (ou LIFO pour Last In, First Out)

Opérations fondamentales :

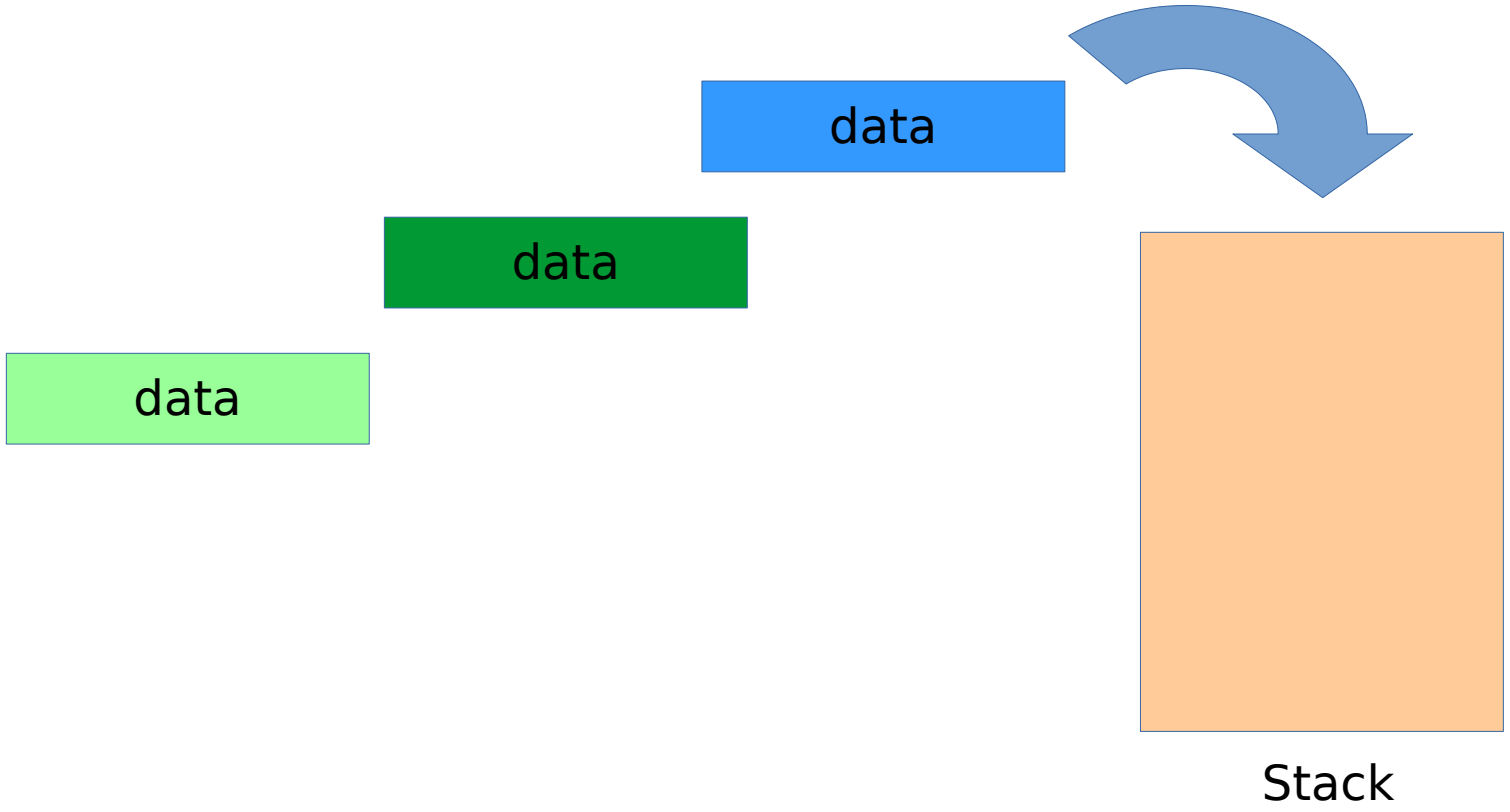
- **→ push : empilement de données**
- **→ pop : récupération des données**

Opération secondaire :

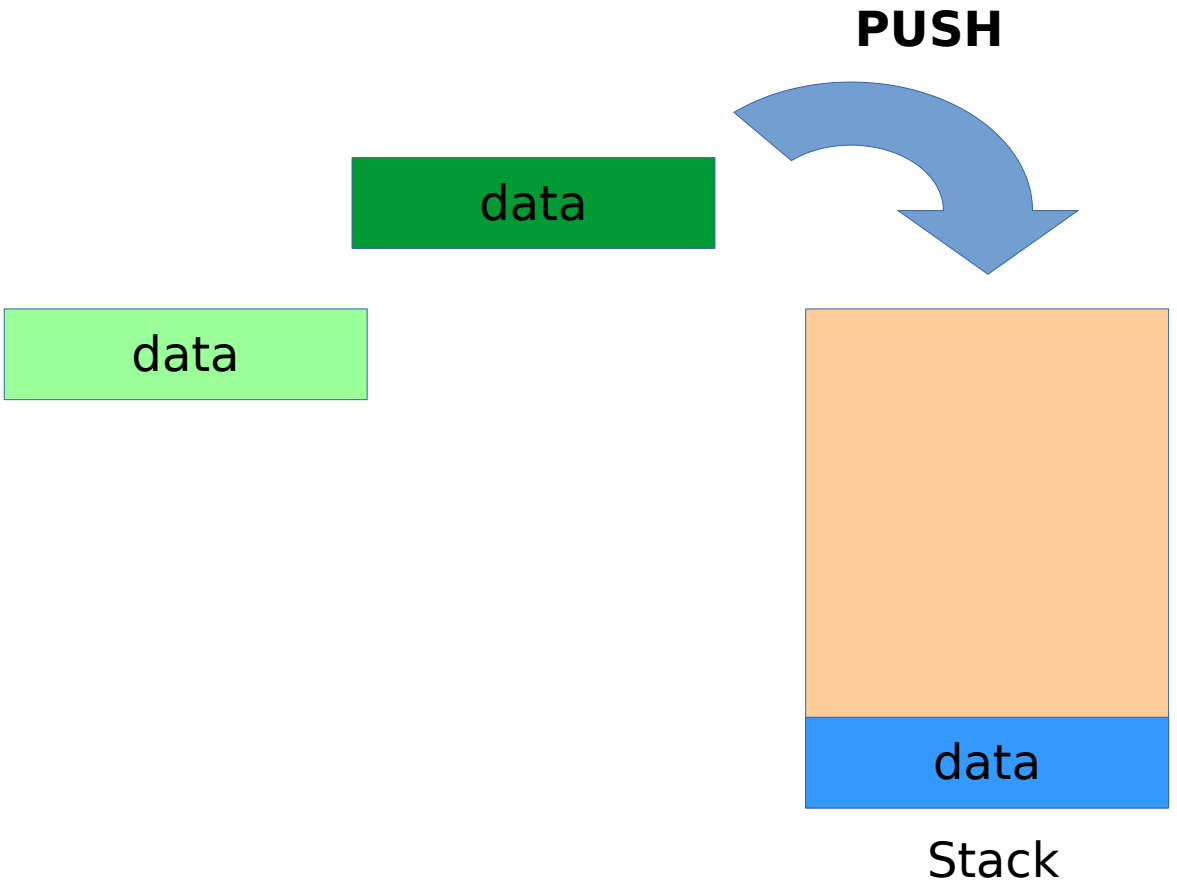
- **→ is_empty : état de la pile**
- **→ size : nombre de données sur la pile**
- **→ peek : récupération de données sans des-empiler**
- **→ dup : duplique le sommet de la pile**
- **→ swap : échange les deux éléments au sommet**

Pile et File

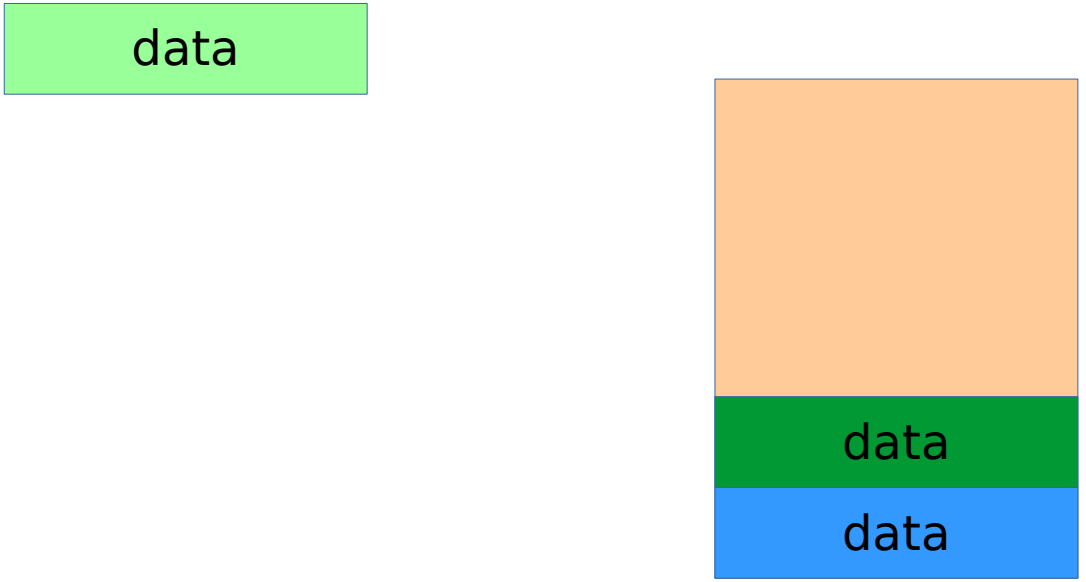
Principe :



Pile et File

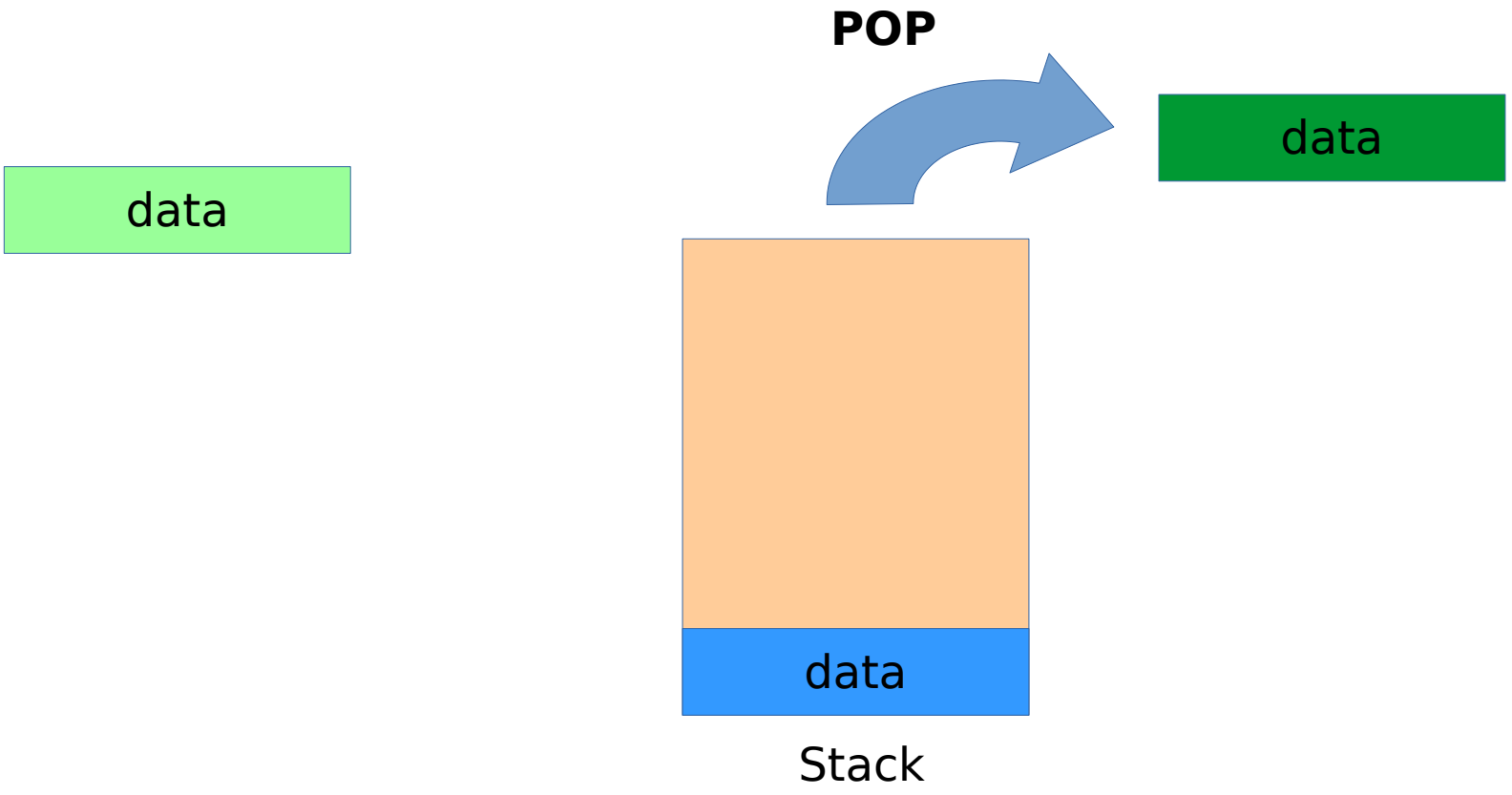


Pile et File



Stack

Pile et File



Pile et File

Cas d'utilisation

→ automate à pile avec la notation post-fixé ou pré-fixé

→ Simulation d'algorithmes récursifs

- **recherche en profondeur (labyrinthe, arbre)**
- **palindrome**
- **tri rapide (heap sort → voir arbres)**
- **tours de Hanoï**

→ fonction undo

→ données en attende de traitement

Pile et File

Implémentation :

-Avec un tableau

-Avec une liste chaînée

Pile et File

Les files (Queue, FIFO)

Structure de données basée sur le principe du « Premier entré, premier sorti » (ou FIFO pour First In, First Out)

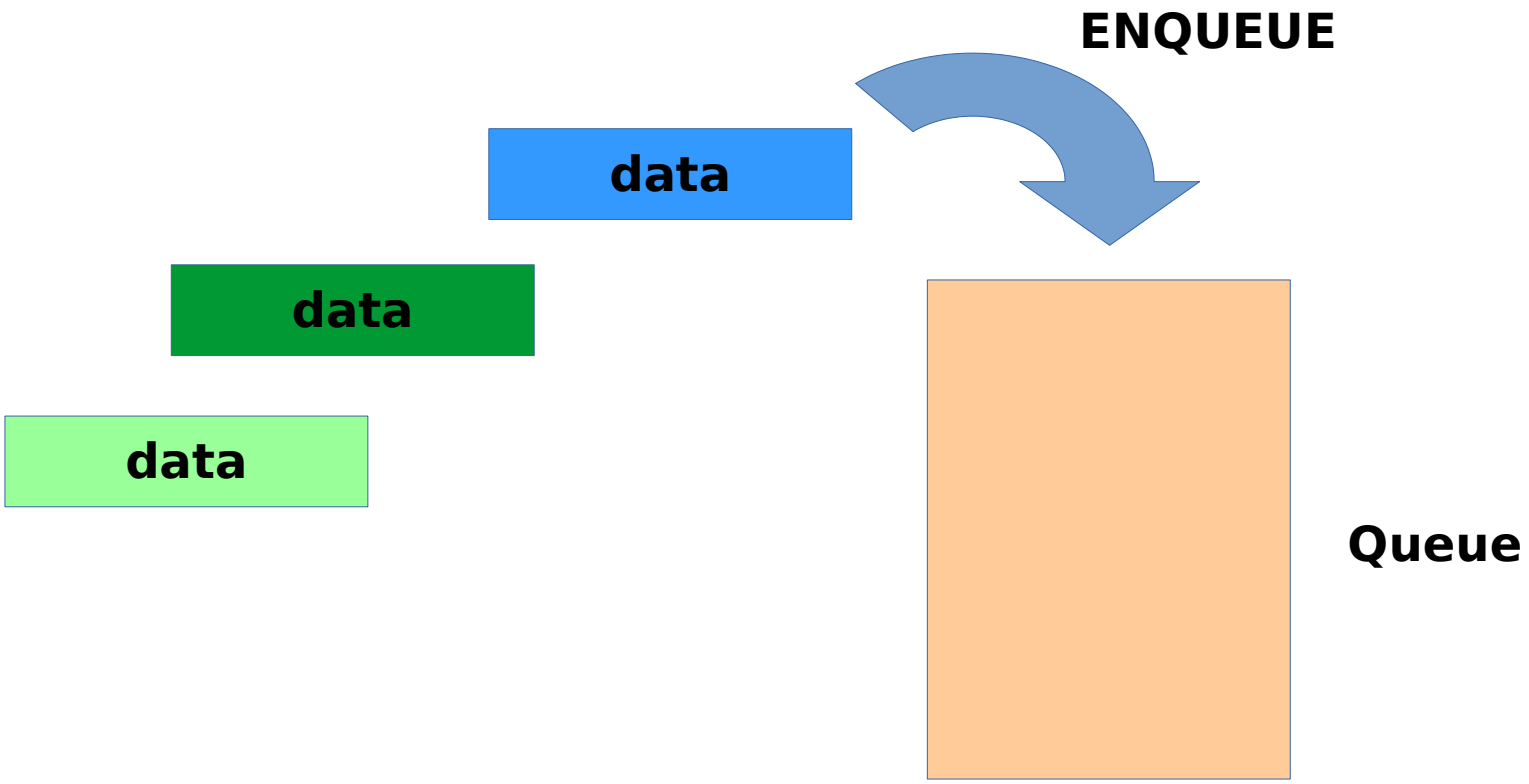
Opérations fondamentales :

- **→ enqueue: ajout de données**
- **→ dequeue : récupération des données**

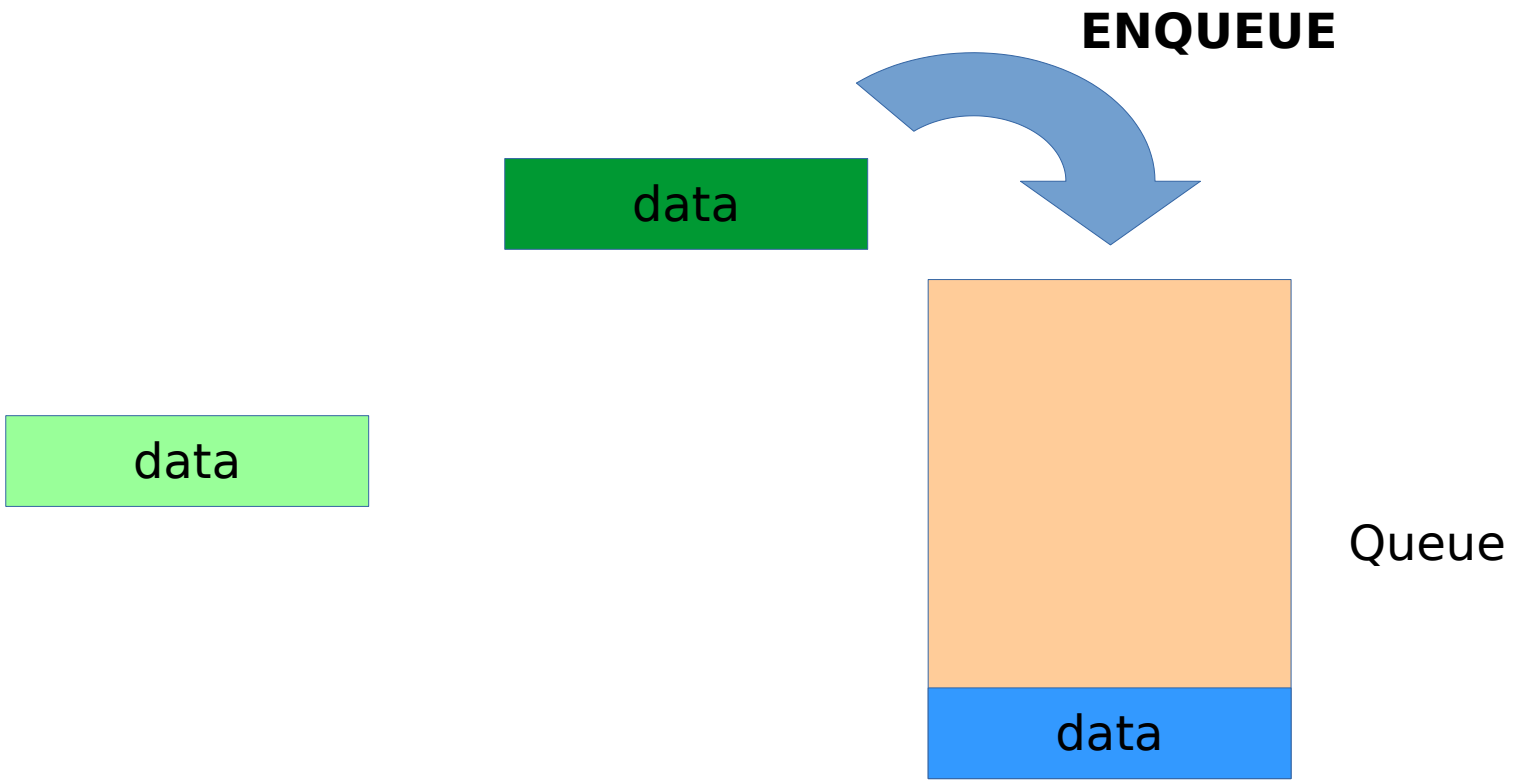
Opération secondaire :

- **→ is_empty : état de la pile**
- **→ size : nombre de données sur la pile**
- **→ peek : récupération de données sans des-empiler**

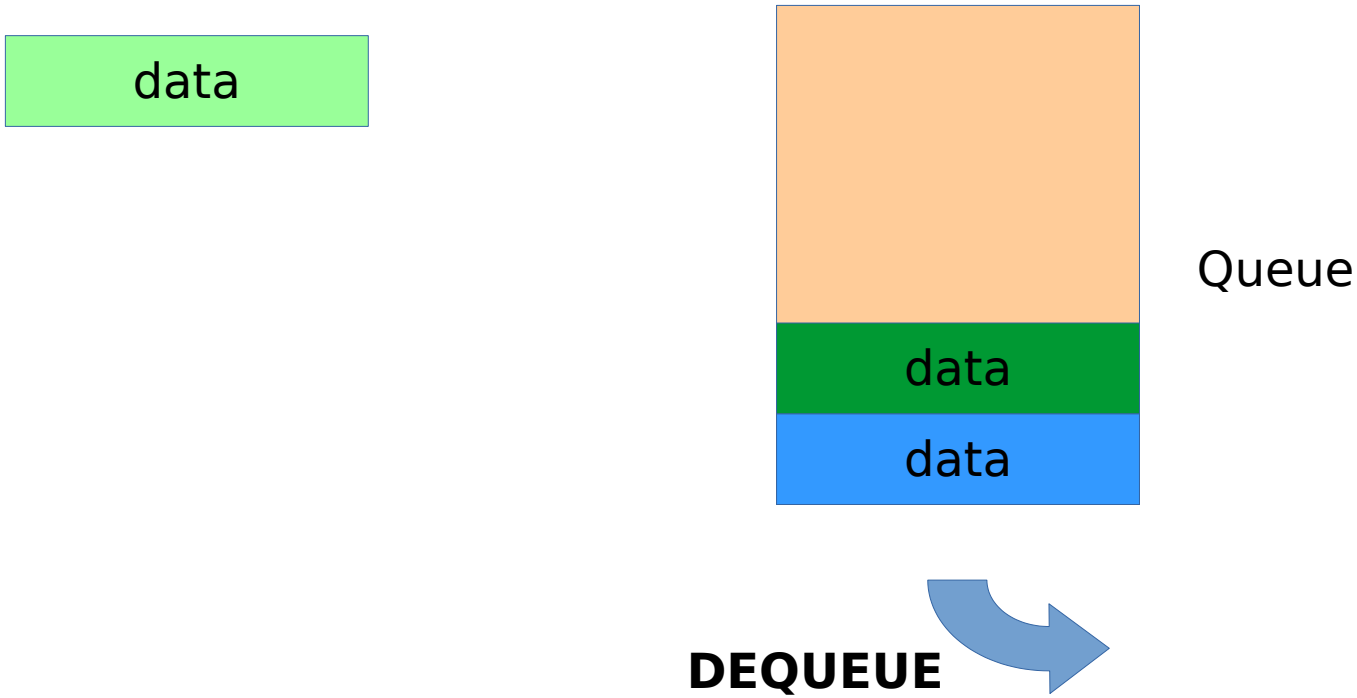
Pile et File



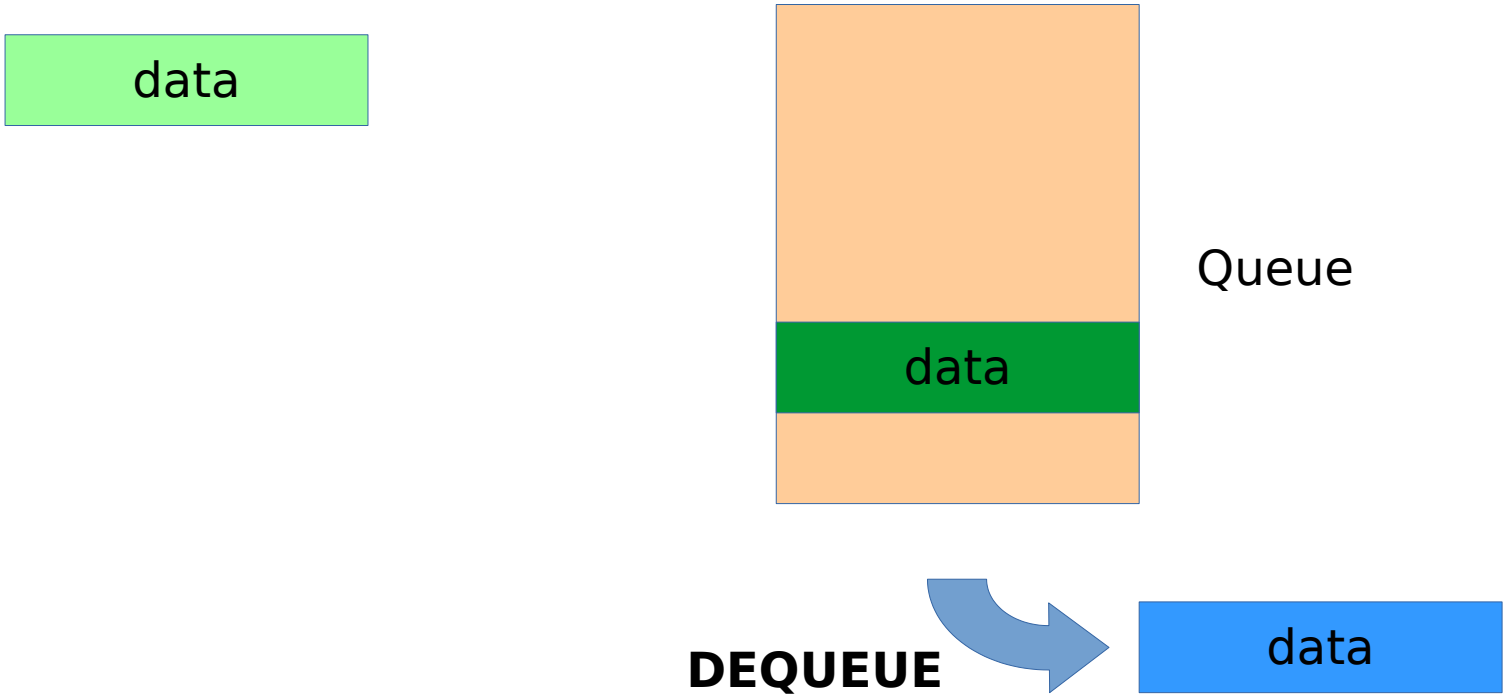
Pile et File



Pile et File



Pile et File



Pile et File

Cas d’utilisation

- → **pile TCP/IP**
- → **Gestion des I/O**
- → **ordonnanceur des OS**
- → **données en attende de traitement**

Pile et File

Implémentation :

-Avec un tableau

-Avec une liste chaînée

Listes

Une liste est une structure de données permettant de regrouper des données de manière à pouvoir y accéder librement (contrairement aux files et aux piles, dont l'accès se fait respectivement en mode FIFO et LIFO).

Une liste chaînée est une suite finie de cellules, chacune formée :

– D’une valeur (par exemple un int pour une liste d’entiers),

– De la référence (c'est-à-dire de l'adresse) vers la cellule suivante.

La liste est à la base de structures de données plus complexes comme la pile, la file, les arbres, etc.

Listes

Une liste est un conteneur d'éléments, où chaque élément contient la donnée, ainsi que d'autres informations permettant la récupération des données au sein de la liste. La nature (les types) de ces informations caractérise un type différent de liste.

On peut distinguer, de manière générale, deux types de liste :

→ Listes contiguës : la liste est représentée par un tableau. Ce n'est pas toujours une bonne solution en pratique, mais nous allons le faire ici, dans l'objectif d'illustrer la dissociation entre un type abstrait et son implémentation.

→ les listes chaînées : les différents éléments de la liste sont créés au fur et à mesure des besoins (implémentation dynamique)

Liste contiguë

Illustration avec une liste contiguë :

type liste : « Article »

{

T: tableau[1..lmax] d'élements

longueur: 0..lmax

}

Listes

Exemple de code C :

```
typedef struct {
    int tab[taille];
    int longueur;
} Liste ;
```

Listes

Les opérations les plus importantes :

- Le parcours
- La suppression
- L'insertion

Les avantages

Accès directe.

Suppression et insertion en fin de liste.

Implémentation facile.

Inconvénients

Il faut savoir modifier la longueur.

les suppressions et les insertions impliquent des décalages.

Liste Chainée

Elle est composée d'une suite de maillon (bloc, nœud) et chaque maillon est constitué de deux champs:

Un champ élément.

Un champ adresse qui pointe vers l'élément suivant.

Avantages

Parcours séquentiel.

Suppression d'un maillon.

Insertion d'un maillon.

Inconvénients

Accès indirecte à un élément, il faut parcourir i-1 maillons pour atteindre le ième maillon.

Perte de place mémoire pour les liens.