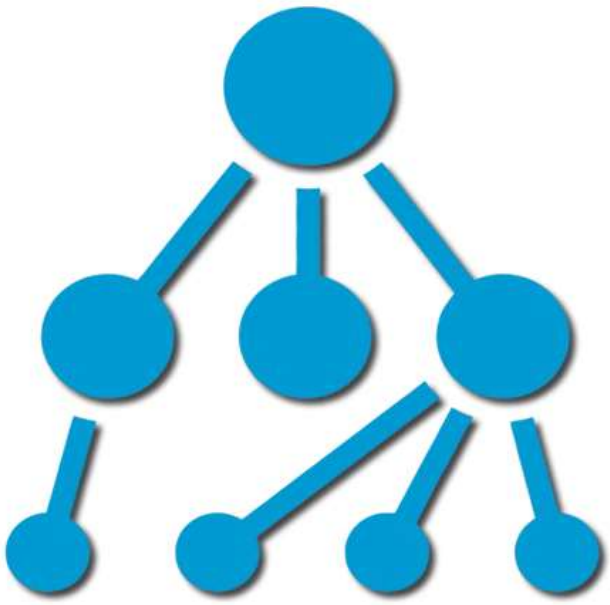
STRUCTURES DE DONNÉES

NSI 01



PILES, FILES

[Frédéric Peurière – Marion Szpieg]

1. Introduction sur les structures de données

Définition : un **type abstrait** ou une **structure de données abstraite** est une spécification mathématique d'un ensemble de données et de l'ensemble des opérateurs associés. On dit que ce type de données est abstrait car il correspond à un cahier des charges qu'une structure de données qu'il est ensuite possible d'implémenter.

On peut donc détailler une structure de données précise sans écrire aucune ligne de code : on appelle cela son **interface**. Une fois que l'interface est bien définie, on peut alors la programmer en Python (ou un autre langage de programmation) : on appelle cela **l'implémentation** de la structure de données.

Voici quelque avantages à la séparation entre l'interface et l'implémentation :
- une interface donnée a plusieurs implémentations différentes
- cette séparation rend plus facile la répartition des tâches dans un travail d'équipe sur une structure de donnée

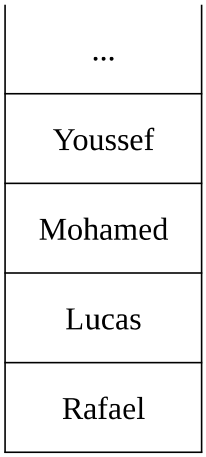
2. Les piles

2.1. Qu'est-ce que c'est ?

La structure de données linéaire «pile» est une suite ordonnées d'éléments qu'on construit avec le principe du « dernier arrivé, premier servi ». En informatique, ce principe de construction est communément appelé LIFO (car « Last In, First Out »). Une pile peut-être vide.

« Dans la vraie vie », on retrouve cette structure dans une pile d'assiettes : les dernières assiettes empilées dans votre placard seront les premières que vous utiliserez lors du prochain repas.

Exemples :



Dans cette pile, on a d'abord empilé 'Rafael', puis 'Lucas', puis 'Mohamed' et enfin 'Youssef'.

Le sommet de la pile est 'Youssef' et la queue de la pile est constituée, dans l'ordre, des éléments suivants : 'Mohamed' puis 'Lucas' et enfin 'Rafael'.

En informatique, ce type de structure de données est souvent utilisée. Par exemple :

- dans un navigateur internet, l'historique des pages consultées est une pile
- dans n'importe quel logiciel de traitement de texte, les modifications apportées au document sont enregistrées sous forme d'une pile

2.2. Interface d'une pile

Voici l'interface minimale d'une pile (à connaître !!) :
- `creer_p()` : retourne une pile vide
- `estVide_p(P)` : retourne vrai si la pile P est vide et faux sinon
- `empiler(P,e)` : ajoute l'élément e au sommet de la pile P (puis retourne la nouvelle pile)
- `depiler(P)` : enlève l'élément au sommet de la pile P si elle n'est pas vide (puis retourne la nouvelle pile)

- `sommet(P)` : retourne le 1er élément de la pile P, si elle n'est pas vide
- `base(P)` : retourne la queue de la pile P, si elle n'est pas vide

Certaines implémentations modifieront directement la structure de la pile et d'autres non. Dans le 2e cas, il faut retourner la nouvelle pile à chaque modification.

Exemple : voici une suite de commandes :

```
1  p=créer_p()
2  p=empiler(p,'Francisco')
3  p=empiler(p,'Lucas')
4  p=empiler(p,'Youssef')
5  p=depiler(p)
6  p=empiler(p,'Mohamed')
7  p=empiler(p,Rafael')
8  p=depiler(p)
9  p=empiler(p,'Frédéric')
10 p=depiler(p)
```

Faire une représentation de la pile après la 5e commande, puis après la 10e commande :

À partir des fonctions primitives, l'interface d'une pile est améliorable par d'autres fonctions, par exemple :

- `longueur_p(P)` : retourne le nombre d'éléments de la pile P
- `dernier_p(P)` : retourne le dernier élément de la pile P (si elle n'est pas vide)
- `inserer_p(P,e,i)` : ajoute l'élément e à la i^{ème} place si c'est possible (puis retourne la nouvelle pile)
- `ieme_p(P,i)` : retourne l'élément à la i^{ème} place dans la pile P s'il existe
- `remplacer_p(P,e,i)` : remplace le i^{ème} élément de la pile P s'il existe par e (puis retourne la nouvelle pile)
- `supprimer_p(P,i)` : supprime le i^{ème} élément de la pile P s'il existe (puis retourne la nouvelle pile)
- …

Encore une fois, certaines implémentations modifieront directement la structure de la pile et d'autres non. Dans le 2e cas, il faut retourner la nouvelle pile à chaque modification.

3. Les files

3.1. Qu'est-ce que c'est ?

La structure de données linéaire «file» est une suite ordonnées d'éléments qu'on construit avec le principe du «premier arrivé, premier servi ». En informatique, ce principe de construction est communément appelé FIFO (car «First In, First Out »). Une file peut-être vide.

Cette structure de données illustre le principe de la file d'attente à un guichet.

Exemple :

↓
Francisco
Rafael
Mohamed
↓

Dans cette file, on a d'abord enfilé 'Mohamed', puis 'Rafael', puis 'Francisco'.

Le début de file est 'Mohamed' et la queue de la file est constituée, dans l'ordre de 'Rafael' puis de 'Francisco'.

En informatique, ce type de structure de données est souvent utilisée. Par exemple :

- le serveur d'impression d'une imprimante
- la politique d'ordonnancement des processus par le noyau

3.2. Interface d'une file

Voici l'interface minimale d'une file (à connaître !!) :

- `creer_f()` : retourne une file vide
- `estVide_f(F)` : retourne vrai si la file F est vide et faux sinon
- `enfiler(F,e)` : ajoute l'élément e à la fin de la file F (puis retourne la nouvelle file)
- `defiler(F)` : enlève l'élément au début de la file F si elle n'est pas vide (puis retourne la nouvelle file)

- `début(F)` : retourne l'élément du début de la file F,si elle n'est pas vide
- `suite(F)` : retourne la file F sans l'élément du début si elle n'est pas vide, sans la modifier

Exemple : voici une suite de commandes :

```
1  f=créer_file()
2  f=enfiler(f,'Frédéric')
3  f=enfiler(f,'Mohamed')
4  f=enfiler(f,'Rafael')
5  f=defiler(f)
6  f=enfiler(f,'Francisco')
7  f=enfiler(f,Lucas')
8  f=defiler(f)
9  f=enfiler(f,'Youssef')
10 f=defiler(f)
```

Faire une représentation de la file après la 5e commande, puis après la 10e commande :

Comme avec les piles, à partir des fonctions primitives, l'interface peut-être améliorée par d'autres fonctions.