# NSI - Terminale

Structures de données linéaires : travaux dirigés

qkzk

2020/10/10

**Compétence :** *Choisir un type de données en fonction d'un problème à résoudre*

## Exercice 1

Quelle structure de données choisir pour chacune de ces tâches ?

1. Représenter un répertoire téléphonique.
2. Stocker l'historique des actions effectuées dans un logiciel et disposer d'une commande Annuler (ou Undo).
3. Envoyer des fichiers au serveur d'impression.

**Compétence :** *Savoir raisonner à l'aide du type abstrait Liste*

## Exercice 2

On donne la séquence d'instructions suivante :

```
l1 = creer_liste_vide()
l2 = creer_liste_vide()

# paramètres (liste, valeur insérée, position). Modifie la liste.
inserer(l1, 1, 1)
inserer(l1, 2, 2)
inserer(l1, 3, 3)
inserer(l1, 4, 4)

# paramètres (liste, position). Renvoie la valeur à cette position.
a = lire(l1, 1)

inserer(l2, lire(l1, 1), 1)
inserer(l2, lire(l1, 2), 1)
inserer(l2, lire(l1, 3), 1)
inserer(l2, lire(l1, 4), 1)
```

1. Illustrer le résultat de chaque étape de cette séquence.
2. Quelle est l'opération effectuée ?

**Compétence :** *Savoir raisonner à l'aide du type abstrait File.*

## Exercice 3

On donne la séquence d'instruction suivante :

```
f = creer_file_vide()

# paramètre (file, valeur enfilée)
enfiler(f, 4)
enfiler(f, 1)
```

```
enfiler(f, 3)

# paramètre (file), renvoie la valeur défilée
n = defiler(f)
enfiler(f, 8)
n = defiler(f)
```

Illustrer le résultat de chaque étape de cette séquence.

**Compétence :** *Savoir raisonner à l'aide des types abstraits File et Pile.*

## Exercice 4

On suppose que l'on a déjà une file `f1` qui contient les éléments suivants saisis dans l'ordre alphabétique :

`f1 = ('A', 'B', 'C', 'D', 'E')`

1. Quel est l'élément issu d'un défilage de `f1` ?
2. Proposer une séquence d'instruction (à l'aide de deux piles `p1` et `p2`) permettant la saisie d'affilée (sans sortie intermédiaire) des 5 éléments `'A', 'B', 'C', 'D'` et `'E'` et de sortir ces éléments comme s'ils sortaient d'une file.
3. Que faudrait-il faire pour avoir exactement le même fonctionnement qu'avec une file, c'est-à-dire avec sortie éventuelle d'élément ?

## Exercice 5

La notation polonaise inverse (NPI) permet d'écrire des opérations arithmétiques, sans utiliser de parenthèses. Ici nous nous limiterons à des nombres entiers naturels et aux opérations `+, -, *` et `/` sur eux. Dans cette notation, les opérateurs sont écrits après les opérandes (nombres entiers naturels).

Par exemple l'expression classique :

```
13 * (3 + 2)
```

Donne en NPI :

```
3 2 + 13 *
```

On écrit et on exécute les opérations dans le sens des priorités vues en cours de mathématiques. Dans cette notation on réalise :

- L'addition entre 3 et 2 (`3 2 +`)
- La multiplication entre le précédent résultat et 13 (`13 *`)
- On a ainsi le résultat.

1. Donner la file correspondant à la saisie NPI de l'exemple. Faire de même avec la pile.
2. Quelle est la structure adaptée à la résolution de l'expression ?

   **Note :** On remarquera qu'on doit toujours avec 2 opérandes pour un opérateur. Il faut stocker le résultat intermédiaire dans la structure pour effectuer la suite des calculs.
3. En utilisant les opérations du type abstrait pile, proposer une fonction permettant d'afficher le résultat d'une expression en NPI.

   **Note :** On pourra considérer qu'on a déjà la fonction `inverser_pile(p)` qui renvoie une pile qui est déjà en ordre inverse de celle donnée en argument. On supposera également que la syntaxe en NPI est correcte.

**Compétence :** *savoir implémenter une structure à l'aide d'un type donnée en Python*

## Exercice 6.

Étudier les méthodes `append` et `pop` du type `list` de Python à l'aide de la documentation Python.

1. Proposer une implémentation des opérations classiques de la pile à l'aide des méthodes `pop` et `append` du type liste de Python

2. Proposer de la même manière une implémentation des opérations classiques de la file en Python.

## Exercice 7 : Bilan

Dans un logiciel de calcul formel ou, plus généralement dans un éditeur de texte (par exemple utilisé pour écrire des programmes), il y a une gestion dynamique du parenthésage. Par exemple, les deux expression suivantes sont erronées :

$$A = (4 + \sqrt{3}))^2$$

*deux parenthèses fermantes pour une ouvrante*

$$B = ((4 + n)^2$$

*deux parenthèses ouvrantes pour une fermante*

L'objectif de cet exercice est de proposer une solution informatique pour programmer une fonction qui reçoit comme argument une chaîne de caractères constituée uniquement de parenthèses ouvrantes et fermantes (pas d'autres caractères), cette fonction analyse le parenthésage et renvoie à l'utilisateur un message adapté.

On se propose de plus d'indiquer, pour chaque parenthèse ouvrante, la position de la parenthèse fermante correspondante. Pour le mot `'(())()'`, on donnera les couples d'indices `(0, 3)`, `(1, 2)` et `(4, 5)`

L'idée consiste à parcourir le mot de la gauche vers la droite et à utiliser une pile pour indiquer les indices de toutes les parenthèses ouvertes, et non encore fermées, vues jusqu'à présent.

1. En utilisant les opérations du type abstrait pile, proposer une fonction booléenne permettant de retourner Vrai si la chaîne donnée en argument est bien parenthésée et Faux sinon.
2. Implémenter cette fonction en Python. On pourra utiliser l'implémentation de la pile vue dans l'exercice 4.