TP n° 6 : les piles et la vérification syntaxique des expressions

Vérification syntaxique des expressions

Présentation

Pour vérifier si une expression est bien parenthésée (autant de parenthèses ouvrantes et fermantes), on peut utiliser le principe des piles.

L'objectif du TP est d'écrire avec une pile une fonction qui contrôle si une expression mathématique, donnée sous forme d'une chaîne de caractères, est bien parenthésée, c'est- à-dire s'il y a autant de parenthèses ouvrantes que de fermantes, et qu'elles sont bien placées.

Exemple : l'expression $(\dots(\dots)\dots)$ est bien parenthésée, tandis que l'expression $(\dots(\dots(\dots)\dots)$ ne l'est pas.

Dans la suite, le terme *parenthèse* désigne, sans plus de précision, une parenthèse, un crochet ou une accolade ; le terme *expression* désigne une chaîne de caractères.

Principe

- L'analyse de l'expression se fait caractère après caractère.
- On ajoute à la pile chaque parenthèse ouvrante trouvée.
- À chaque parenthèse fermante, on vérifie que la pile n'est pas vide, puis on dépile la parenthèse ouvrante au sommet de la pile.
- À la fin de l'expression, toutes les parenthèses ouvertes doivent être fermées.

1. Exercice

Donner l'état de la pile aux différentes étapes de l'exécution lors de la vérification syntaxique des expressions suivantes :

1. $a(b(c(d)e)f$
2. $a(b(c)d)e)f$
3. $a(bc(d)e)f$

2. Exercice : débranché

Écrire un algorithme d'une fonction nommée *verification_syntaxique* qui prend en argument une expression et qui retourne *True* si l'expression est syntaxiquement correcte ou *False* sinon.

Il faut n'utiliser que le concept du TAD pile, c'est-à-dire les méthodes qui lui sont associées :

- *creer_pile()* crée une pile vide ;
- *est_vide(p)* retourne *True* ou *False* si la pile p est vide ou non ;
- *empiler(p, x)* empile l'élément x à la pile p ;
- *depiler(p)* renvoie le dernier élément ajouté à la pile p et le supprime de la pile ;
- *sommet(p)* renvoie le dernier élément ajouté à la pile p sans le supprimer.

3. Exercice : implémentation en Python

- Sur votre disque travail, créer un dossier *TP06_utilisation_des_piles*, et y placer une copie du programme *TP06_piles_fonctions* disponible dans les ressources.
- Créer un nouveau programme python, et le sauvegarder dans ce même dossier sous le nom *TP06_verification_syntaxique*.
- Dans ce programme, commencer par importer le programme *TP06_piles_fonctions*.
- Utiliser les fonctions du TAD pile pour implémenter l'algorithme de l'exercice précédent.
- Ajouter une assertion pour arrêter le programme si l'expression est vide.
- Tester la fonction créée.
- Créer une seconde fonction *verification_syntaxique2* pour gérer la vérification syntaxique d'une expression comportant des parenthèses et des crochets.
- Tester la fonction créée.

Refaire le même exercice, mais en important cette fois le programme *TP06_piles_classe*.