tp_l8dn003_20_06

April 4, 2020

TP n°6: analyse expressions arithmétiques en notation polonaise inverse

Implémentation d'une pile

Proposer une implémentation au moyen d'une liste des quatre primitives indispensables pour manipuler une pile: cree_pile(), vide(), empiler(), depiler(). La définition fonctionnelle des primitives peut être déduite des exemples suivants:

```
print(depiler(empiler(empiler(empiler(cree_pile(),3),5),7)))
7
print(vide(empiler(cree_pile(),5)))
False
```

Evaluation d'une formule en notation polonaise inverse

Ecrire un programme qui prend en entrée une expression arithmétique en notation polonaise inverse (sous la forme d'une chaîne de caractère), et qui donne en sortie la valeur de l'expression. L'algorithme utilisé repose sur l'idée que chaque opérande rencontré est empilé, et que chaque fois qu'on rencontre un opérateur, on dépile deux arguments (les deux derniers, donc), on calcule le résultat, et on l'empile. Si l'expression est bien formée, la valeur de l'expression se trouve au sommet de la pile à la fin du calcul.

Exemples :

```
3 12 3 - + = 12
7 7 * = 49
23 12 - 8 + 2 * = 38
```

Il n'est pas demandé de vérifier que l'expression est bien formée (mais on peut essayer d'ajouter une vérification pour éviter les erreurs). Noter que si l'on veut ajouter des symboles unaires (le signe "moins" dans l'expression "-3") on doit utiliser des symboles différents de leur correspondant binaire.

Traduction d'une formule arithmétique

Ecrire un programme qui prend en entrée une expression arithmétique en notation polonaise inverse, et donne en sortie une chaîne de caractère correspondant à l'expression équivalente en notation infixe. La pile définie plus haut devrait pouvoir être utilisée.

Exemples:

```
3 12 3 - + = (3 + (12 - 3))
7 7 * = (7 * 7)
23 12 - 8 + 2 * = (((23 - 12) + 8) * 2)
```