L2 Informatique

TP Algorithmique – Piles

Évaluation d'expressions postfixées

Le but de ce TP est d'implémenter une méthode permettant l'évaluation d'une expression arithmétique donnée sous forme *postfixée*. Dans cette notation les opérateurs sont donnés **après** les opérandes. Par exemple, l'expression sous forme postfixée `2 8 +` représente l'expression classique `2 + 8` (dite sous forme *infixée*), dont l'évaluation vaut `10`.

L'intérêt de cette notation est qu'elle ne nécessite pas de parenthésages, puisqu'une seule interprétation de l'expression est possible. En outre, son évaluation peut être effectuée simplement durant sa lecture au moyen d'une pile :

- en empilant les constantes numériques et évaluations partielles au fil de la lecture ;
- en dépilant les opérandes nécessaires lors de la lecture d'un opérateur.

Par exemple, pour évaluer l'expression `2 8 + 3 4 + * 1 -`, on effectue la suite d'opérations suivantes :

- Lire 2, empiler 2 ;
- Lire 8, empiler 8 ;
- Lire +, dépiler (8), dépiler (2), évaluer 2+8 (10), empiler 10 ;
- Lire 3, empiler 3 ;
- Lire 4, empiler 4 ;
- Lire +, dépiler (4), dépiler (3), évaluer 3+4 (7), empiler 7 ;
- Lire *, dépiler (7), dépiler (10), évaluer 10*7 (70), empiler 70 ;
- Lire 1, empiler 1 ;
- Lire -, dépiler (1), dépiler (70), évaluer 70-1 (69), empiler 69 ;
- Fin de la lecture, dépiler (69) et retourner le résultat[1].

Écrire un programme permettant d'évaluer une expression arithmétique postfixée saisie par l'utilisateur. On pourra ajouter progressivement des fonctionnalités.

1. Dans un premier temps, considérer uniquement des expressions composées de nombres réels et de signes `+` (addition), `-` (soustraction), `*` (multiplication), `/` (division réelle). Ces quatre opérateurs sont binaires, c'est-à-dire nécessitant exactement 2 opérandes. On n'utilise pas la version unaire de l'opérateur `-`.
   La saisie de l'expression pourra être effectuée terme par terme. Chaque terme (même les nombres) sera considéré lors de la lecture comme une chaîne de caractères. Tous les termes (nombres et opérateurs) seront stockés dans un tableau de chaînes de caractères symbolisant l'expression postfixée. L'évaluation de l'expression s'effectuera en utilisant une pile d'éléments de type `float`.
   → la fonction `std::stof(s)` convertit une chaine de caractères `s` en un nombre (type `float`).

2. Ajouter les opérateurs `^` (puissance réelle) et `RAC` (racine carrée), ainsi que la constante `PI`.
   → Attention, la racine carée est un opérateur unaire et n'attend qu'un opérande.
   → Pour l'évaluation, utiliser les opérateurs `pow` (puissance réelle) et `sqrt` (racine carrée) de la bibliothèque `cmath`.
   → La constante `PI` pourra être définie comme valant `3.14159`.
   → Exemple 1 : L'expression `2 5 + 4 5 3 + 2 ^ * RAC *` vaut 112.
   → Exemple 2 : L'expression `6.2 0.75 - 3 ^ 2 PI * RAC /` vaut `64.5802`.

3. Modifier la saisie de manière à ce que l'expression soit saisie en ligne, c'est-à-dire stockée dans une seule chaîne de caractères. Écrire un sous-programme permettant de transformer une chaîne de caractères représentant une expresssion arithmétique postfixée (supposée correcte) en un tableau de chaînes de caractères comme utilisé pour l'évaluation.

4. Question supplémentaire : vérifier au préalable que l'expression saisie est correcte et afficher un message d'erreur le cas échéant.

[1] L'expression préfixée `2 8 + 3 4 + * 1 -` correspond à l'expression infixée `((2 + 8) * (3 + 4)) - 1` et vaut `69`.