Programmation Impérative - TP 2
Une calculatrice à pile

Rohan Fossé - Léo Mendiboure - Guillaume Mercier
{rohan.fosse,leo.mendiboure}@labri.fr, mercier@enseirb-matmeca.fr

2019-2020

1 Présentation

L'objectif de ce TP est d'implémenter une calculatrice qui utilise une structure de donnée de type *pile* afin d'évaluer une expression arithmétique qui sera passée en argument au programme. Les expressions arithmétiques seront passées en notation *postfixe*, c'est à dire que les opérandes sont placées avant l'opérateur, par exemple : 2 3 + renverra 5 tandis que 3 5 + 2 $*$ renverra 16. Le fonctionnement d'une calculatrice est simple :

- on lit les arguments du programme séquentiellement : ce sont soit des nombres, soit des opérateurs arithmétiques
- s'il s'agit d'un nombre (une future opérande), on l'empile (sur le haut de la pile, donc)
- s'il s'agit d'un opérateur, on dépile les deux élements sur le haut de la pile, on applique l'opérateur à ces deux valeurs et le résultat est empilé à son tour
- une fois qu'il n'y a plus d'arguments à traiter, on dépile l'élément restant dans la pile : c'est le résultat de l'évaluation de la suite d'opérations

Nous allons formuler les hypothèses suivantes :

- les arguments du programme lui seront passés via la *ligne de commande*.
- la suite d'arguments passée au programme est *correcte*, c'est à dire qu'il n'est pas nécessaire de gérer le cas où l'utilisateur fait une erreur de saisie quand il rentre une suite d'opérations
- les arguments passés au programme sont de deux types uniquement : des nombres ou bien des opérateurs (c.f liste ci-dessus). Cas particulier : l'opérateur de multiplication $*$ est interprèté d'une façon non convenable par le shell, il faudra donc mettre des doubles guillemets (donc `"*"` à la place de $*$).
- les nombres sont des entiers positifs ou nuls (pour le moment)
- les opérateurs sont tous binaires (pour le moment)

2 Première version : tableau

Dans cette première version, la pile utilisée sera stockée à l'aide d'un tableau, dont la taille (*en nombre d'éléments*) est connue statiquement, par exemple une constante :

```
#define STACK_SIZE_MAX 1024
```

Ce tableau pourra être déclaré en tant que variable globale au programme ou bien en tant que variable locale à la fonction `main` de votre programme.

Question
Implémentez la calculatrice : il vous faudra pour cela écrire deux fonctions : une première permettant d'empiler un élément `push` et une seconde fonction permettant de dépiler un élément `pop`. En utilisant ces deux fonctions , la fonction `main` devra permettre d'afficher le résultat du calcul passé en argument, par exemple :

```
$ ./calculatrice 2 3 "*"
resultat : 6
```

Un peu d'aide :

- Pour la gestion des différents cas : nombre ou opérateur binaire (multiplication, soustraction, addition et division) la fonction `main` pourra par exemple utiliser la construction `switch/case` ; on pourra également réfléchir à la pertinence de l'utilisation de la fonction `isdigit()`
- La fonction `push` doit permettre d'ajouter un élément au tableau, tandis que la fonction `pop` doit quant à elle permettre d'accéder au dernier élément ajouté et de le *retirer* de la pile.

3 Deuxième version : pointeurs

Dans cette deuxième version, la pile sera stockée à l'aide d'un tampon alloué en mémoire (avec `malloc` par exemple).

Question
Donnez une borne supérieure sur la taille du tampon pour l'allocation.

Question
Implémentez la calculatrice en n'utilisant que la notation pointeur (et plus la notation tableau `[]`). Pour ce faire, il vous era nécessaire de modifier les fonctions `push` et `pop` préalablement définies ainsi que le fonctionnement de la fonction `main`.

4 Troisième version : liste chaînée

Dans cette version, la pile sera stockée sous forme de liste chaînée.

Question
Définissez le type de donnée qui va servir à représenter un maillon de la chaîne.

Question
L'accès à la liste chaînée va se faire à l'aide d'un pointeur sur l'élément de tête : où ce pointeur peut-il être déclaré dans le programme ?

Question
En fonction de la façon dont le pointeur de tête est déclaré (mais pas que), réfléchissez aux prototypes des fonctions `push` et `pop`. Rappel : dans le cas d'une liste chaînée, la fonction `push` va rajouter un élément en début de liste tandis que la fonction `pop` va retirer la tête courante de la liste.

Un peu d'aide :

- Pour ce qui est de la définition du type de donnée il vous faudra réfléchir à ce que doit contenir la structure correspondant à un élément de la liste chaînée.
- En fonction de la déclaration du pointeur de tête, il pourra être nécessaire que les fonctions `push` et `pop` prennent ou non en argument ce pointeur.

5 Améliorations

Question
Que faut-il modifier aux programmes pour pouvoir gérer des entiers quelconques ?

Question
Que faut-il modifier aux programmes pour pouvoir gérer des nombres flottants ?

Question
Quelle est la taille *maximale* de pile dont le programme a réellement besoin pour fonctionner ?

Question
Implémentez la gestion d'une calculatrice postfixe à pile avec des opérateurs n-aires, par exemple :

```
$ ./calc 4 5 6 +
resultat = 15
```

Aide : Utilisez un compteur pour déterminer le nombre d'opérandes ajoutés au tableau depuis le dernier empilement d'un opérateur.