IN 101 - TD 09
18 novembre 2011

Énoncé

ENSTA ParisTech

Exercice 1 : Pile et calculatrice

Quand on écrit une expression arithmétique, on utilise généralement une notation infixée : les opérateurs $+$, $-$, $\times$ et $/$ sont placés entre les deux expressions auxquelles ils s'appliquent. On écrit par exemple : $(3+4)\times(7-5)$. Cette notation est la plus lisible pour un humain, mais elle n'est pas la plus simple pour un ordinateur. Il est nécessaire d'utiliser des parenthèses et l'évaluation doit donc en tenir compte. Il existe donc d'autres notations qui rendent les calculs plus simples. La notation postfixée, aussi appelée notation polonaise inverse, place les opérateurs après leurs arguments :

$$3\ 4 + 7\ 5 - \times$$

Avec cette notation, il suffit d'empiler les arguments numériques (dans une pile), puis à chaque apparition d'un opérateur $+$, $-$, $\times$ ou $/$, d'extraire les deux éléments en haut de la pile, de leur appliquer l'opérateur et d'empiler le résultat. À la fin de l'expression, le résultat est le dernier élément présent dans la pile.

Le but de cet exercice est de programmer une calculatrice en notation polonaise inverse. Il faudra donc écrire un programme capable de lire une expression passée en argument (dans `argv`) et qui affiche le résultat. Chaque élément de l'expression doit être séparé des autres par des espaces pour qu'il aille dans une case différente de `argv`. La multiplication ne peut pas utiliser le symbole `*` qui serait interprété par le shell, vous pourrez donc utiliser un `x` à la place.

1.a] Commencez par définir la structure de pile d'entiers. Vous pouvez utiliser au choix une liste chaînée ou un tableau pour l'implémentation et la pile.

1.b] Programmez la fonction `push` qui insère un élément dans la pile. Cette fonction doit normalement prendre en argument l'entier à insérer et la pile dans laquelle l'insérer. Pour cet exercice il n'y a besoin que d'une pile, donc vous pouvez, si vous préférez, utiliser une variable globale pour la pile et ne passer qu'un entier en argument.

1.c] Programmez la fonction `pop` qui extrait un élément de la pile. Comme pour `push`, cette fonction prend en argument soit une pile, soit rien si vous utilisez une variable globale. Elle doit retourner la valeur extraite de la pile.

1.d] Programmez l'évaluation d'une expression en notation polonaise inverse passée sur la ligne de commande. Votre `main` devra regarder les arguments un par un et pour chacun tester si c'est un opérateur (et dans ce cas appliquer l'opération correspondante s'il y a assez d'éléments dans la pile) ou si c'est un nombre (et dans ce cas l'ajouter à la pile). À la fin il suffit d'afficher l'élément restant dans la pile.

1.e] *(pour ceux qui vont vite)* On veut maintenant faire une calculatrice interactive, qui ne prend plus ses arguments en ligne de commande, mais avec des `scanf` à la place. `scanf` permet de lire une chaîne de caractères quelconque avec `%s` et vous pouvez donc réutiliser en grande

partie le `main` de la question précédente. Pour simplifier l'utilisation de votre calculatrice il faudra afficher le contenu de la pile avant de demander à l'utilisateur de rentrer une valeur.

Exercice 2 : Conversion en notation polonaise inverse

Il est possible de convertir la notation infixée standard en notation polonaise inverse assez facilement si elle est entièrement parenthésée (chaque opération est entourée de parenthèses). L'ordre des entiers n'a pas à être modifié, et il suffit de déplacer les opérateurs à la fin des sous-expressions : chaque opérateur rencontré peut donc être ajouté à une pile et chaque fois qu'une parenthèse fermante est rencontrée un opérateur est extrait de la pile. À la fin de l'expression, si un opérateur est encore présents dans la pile il faut l'extraire.

2.a] Programmez la conversion d'une expression infixée (passée en argument dans `argv` comme dans l'exercice précédent) en notation postfixée. Les parenthèses ont un sens pour le shell et il est donc plus simple d'utiliser les symboles `[ ]` à la place des parenthèses `( )`.

2.b] *(pour ceux qui vont vite)* Modifiez votre calculatrice interactive pour qu'au lieu de stocker des entiers dans la pile elle stocke des chaînes de caractères. Au lieu de calculer une expression elle doit maintenant convertir l'expression qu'on lui entre en notation polonaise inverse en notation infixée avec des parenthèses. Pour rappel, une chaîne de caractères est un tableau de *`char`* se terminant par un caractère nul. Pour mettre bout à bout deux chaînes de caractères il faut allouer un tableau suffisamment grand (donc mesurer la longueur des chaînes à joindre) et les recopier dedans. Évitez si possible les fuites mémoire.

Exercice 3 : Polynômes creux

On veut manipuler des polynômes creux en ne stockant que les monômes effectivement présents. Par exemple, le polynôme $5x^7 + 3x + 2$ peut être représenté par 3 monômes $(5, 7)$, $(3, 1)$ et $(2, 0)$. Chaque polynôme peut donc être représenté par une liste chaînée de monômes.

3.a] Définissez la structure d'un monôme et d'un polynôme creux (à coefficients entiers) et écrivez une fonction qui affiche un polynôme creux (pas forcément dans l'ordre).

3.b] Écrivez une fonction qui ajoute un monôme à un polynôme creux et essayez la. Cette fonction modifie le polynôme passé en argument est doit donc faire du passage par adresse.

3.c] Écrivez une fonction qui ajoute un polynôme creux à un autre (sans simplification si plusieurs monômes du même degré sont présents). Le premier polynôme (passé par adresse) sera modifié pour contenir la somme, le deuxième polynômes ne doit pas changer.

3.d] Écrivez une fonction qui multiplie deux polynômes creux (sans modifier ces polynômes). Vous aurez besoin d'écrire une fonction qui retourne une copie d'un polynôme et d'une autre qui multiplie un polynôme creux par un monôme (et aussi d'une fonction pour libérer un polynôme).

3.e] *(plus difficile)* Écrivez une fonction qui réduit un polynôme creux en mettant les puissances dans l'ordre (décroissant) et en ajoutant les termes de même degré (et en enlevant les monômes nuls).

3.f] *(plus difficile)* Programmez une fonction qui calcule le pgcd de deux polynômes creux.