Licence L3 Math Info **2018/2019**
Université d'Artois

Compilation
TP 5 - Analyse Lexicale

Une calculette avec des variables

Description générale :

On veut réaliser une petite calculette permettant d'évaluer des expressions arithmétiques contenant éventuellement des variables. Les expressions seront données en notation polonaise postfixée. Pour pouvoir utiliser des variables, on rajoute à la calculette une instruction permettant d'associer la *dernière valeur* calculée à une variable (un nom) ainsi qu'une instruction permettant de visualiser le contenu d'une variable.

L'interaction avec la calculette se fait par l'entrée standard (le clavier). Chaque ligne de commandes est terminée par un point (.)
L'utilisateur demande l'évaluation d'une expression en tapant une ligne au format suivant : ? *expr* .
L'utilisateur demande à la calculette de mémoriser le résultat de la dernière évaluation en tapant une ligne au format suivant : ! *var* .
L'utilisateur demande l'affichage du contenu d'une variable en tapant une ligne au format : $ *var* .
L'utilisateur quitte l'application en tapant simplement un point.

En cas d'occurrence d'un caractère erroné, la calculette ignorera tous les caractères lus jusqu'à retrouver le caractère de fin de ligne de commande (le point). Les erreurs dues à des expressions mal formées sont plus délicates à gérer. Nous laissons l'examen et le traitement de ce problème à votre sagacité.

L'interface entre la ligne de commande entrée par l'utilisateur (une suite de caractères) et le processus d'évaluation (et de mémorisation) est réalisé en utilisant l'analyseur lexical Lex
L'analyseur découpe la ligne de commande en *unités syntaxiques* (ou *lexèmes*) auxquelles sont associées, dans le processus d'évaluation, un certain nombre d'actions.

1. Principe d'évaluation d'une expression

L'évaluation d'une expression postfixée est réalisée à l'aide d'une pile destinée à recevoir les valeurs entières lues au clavier, les résultats intermédiaires ainsi que le résultat final.

Le principe d'évaluation est le suivant : on empile les entiers et les valeurs des variables rencontrées lors de la lecture de l'expression ; lorsque l'on rencontre un opérateur, ses arguments se trouvent sur la pile, on leur applique donc l'opération correspondante ; on remplace, sur la pile, les arguments par le résultat obtenu.
Par exemple, l'évaluation de `2 m1 + 4 +` se décompose en :

- Empiler `2` puis la valeur de la variable `m1` (disons `3`)

- Calculer 2+3 (les valeurs 2 et 3 sont récupérées sur la pile)

- Dépiler `2` et `3`, puis empiler `5`

- Empiler 4

- Calculer 5+4

- Dépiler 5 et 4, puis empiler 9

2. L'architecture de la calculette

La mémoire de la calculette est un *nombre fini* de variables nommées, pour se faciliter la vie, de façon uniforme : m1, m2, m3, .. Elle est concrètement implantée comme un tableau d'entiers appelé mem. La variable m1 sera en mem[1], la variable m2 en mem[2], etc... Les accès, en lecture et écriture à cette mémoire seront réalisés par l'accès et la modification des éléments du tableau. La première case mémoire (mem[0]) sera réservée à usage interne comme variable tampon.
La pile sera également un tableau d'entiers appelé stack. On définira les primitives d'empilement push, de dépilement pop, d'accès au premier élément de la pile top. On définira également un prédicat is_empty indiquant si la pile est vide ou non.

L'architecture logicielle de la calculette peut être spécifié sous forme d'un automate à cinq états :

WAIT : La calculette est en attente d'une commande ;
DISCMD : Pour affichage du contenu d'une variable ;
STOCMD : Pour sauvegarder le dernier résultat calculé (m[0]) dans une variable ;
EVALCMD : Pour évaluation d'une expression.
SKIP : Oubli des erreurs de saisie.

Une ligne de commande est analysée (par l'analyseur lexical) en sept unités lexicales différentes :

DOT : Le point qui termine une ligne de commande ;
QMARK : Le point d'interrogation qui commence une ligne de commande d'évaluation ;
EXMARK : Le point d'exclamation qui commence une ligne de commande de sauvegarde ;
DOLLAR : Le ``dollar'' ($) qui commence une ligne de commande d'affichage du contenu d'une variable ;
OP : Un symbole d'opérateur arithmétique ;
NUM : Une constante entière en notation décimale ;
VAR : Un nom de variable.

On décrit alors schématiquement le comportement de la calculette selon l'unité lexicale courante et les cinq états possibles :

DOT

> WAIT : Quitter l'application.
> EVALCMD : Stocker le résultat du calcul dans mem[0] et l'afficher.
> Sinon, repasser à l'état WAIT.

QMARK

> WAIT : Passer à l'état EVALCMD.
> SKIP : Ne rien faire.
> Sinon, afficher un message d'erreur puis passer à l'état SKIP.

EXMARK

> WAIT : Passer àl'état STOCMD.
> SKIP : Ne rien faire.

Sinon, afficher un message d'erreur puis passer à l'état SKIP.

DOLLAR

WAIT : Passer à l'état DISCMD.
SKIP : Ne rien faire.
Sinon, afficher un message d'erreur puis passer à l'état SKIP.

OP

EVALCMD : Evaluer l'application de l'opérateur à ses arguments, mettre à jour la pile.
SKIP : Ne rien faire.
Sinon, afficher un message d'erreur puis passer à l'état SKIP.

NUM

EVALCMD : Empiler.
SKIP : Ne rien faire.
Sinon, afficher un message d'erreur puis passer à l'état SKIP.

VAR

DISCMD : Afficher.
EVALCMD : Empiler le valeur de la variable.
SKIP : Ne rien faire.
STOCMD : Sauver dans la variable la dernière valeur calculée (mem[0]).
Sinon, afficher un message d'erreur puis passer à l'état SKIP.

sinon

SKIP : Ne rien faire.
Sinon, afficher un message d'erreur puis passer à l'état SKIP.

3. Travail à fournir

1. Définir un petit module de gestion de pile à l'aide d'un fichier d'interface stack.h et d'un fichier d'implantation stack.c. Le fichier d'interface contiendra :
   - La définition du type des piles : stack
   - définition des primitives
     stack new_stack(); création d'une nouvelle pile
     void push(int, stack); empiler
     int top(stack); valeur du sommet de pile
     void pop(stack); dépiler

2. Définir un fichier d'interface cal.h contenant :
   - la définition du type *lexical_values* des valeurs associées aux unités lexicales OP, NUM ou VAR;
   - la définition du type (énuméré) states représentant les différents états de la calculette ;
   - les variables nécessaires.

3. Définir le fichier d'application cal.lex
4. Définir le fichier Makefile nécessaire à la compilation de l'application.