Université de Lille — FACULTÉ DES SCIENCES ET TECHNOLOGIES — Département Informatique

TDM 4 - Automates et Langages Réguliers

octobre 2020

Analyse lexicale

Ces exercices sont la prolongation de ceux de la fiche précédente.
Appelez `PostfixLexer` votre classe d'analyse lexicale et placez-la dand un fichier `postfix_lexer.py`

Exercice 1 :
Q 1 . Faites en sorte que les opérateurs `add, sub, mul, div` puissent être écrits soient entièrement en minuscules, soit entièrement en majuscules.
Q 2 .
Créez un nouveau type token nommé `OP1` désignant un opérateur à un seul argument.
La seule opération de ce genre sera l'opérateur « opposé » (opposé d'un nombre) et sera désignée par le mot `opp` en majuscules ou minuscules.
Comme pour le token `OP2`, la valeur associée au token `OP1` doit être une fonction. Pour l'opérateur `opp` ce sera la fonction python définie par : `lambda x : -x`
Q 3 .
Lire l'annexe ci-dessous qui explique le principe des expressions arithmétiques postfixées.

Un programme python d'évaluation d'expressions postfixées vous est fourni.
Vérifiez la validité de votre analyseur lexical en testant l'évaluation d'expressions composées uniquement de valeurs entières et d'opérateurs (pas d'identificateur)

Exercice 2 :
Q 1 .

1. créez un nouveau token `POP` qui accepte les chaînes `pop` et `POP`
2. complétez l'évaluateur d'expressions pour traiter la commande POP qui a pour effet d'enlever la valeur située en sommet de pile.

Q 2 . L'évaluateur d'expression postfixée dispose maintenant de variables (de type entier, évidemment). Une variable est dénotée par un identificateur qui suit la même syntaxe que dans le dernier exercice de la fiche précédente.
Dans un nom de variable, majuscules et minuscules ne sont pas équivalentes (`a` et `A` désignent des variables différentes).
Les variables ne sont pas déclarées. Une variable non initialisée possède la valeur 0.
Pour désigner une valeur, on peut maintenant utiliser un nom de variable de la même façon qu'on utilise une valeur immédiate. Par exemple `4 toto +` vaudra 4 si *toto* n'a pas été initialisée (et vaut donc 0).
La commande `->` immédiatement suivie d'un nom de variable (sans espace entre deux) signifie qu'il faut copier la valeur située au sommet de la pile dans la variable (nb : la valeur n'est pas enlevée de la pile). La commande ne doit donc pas être utilisée quand la pile est vide. Par exemple :

- `3 5 + ->a 4 * a *` vaut 256 (a prend la valeur 8)
- `16 ->x 8 * 1 +  x *` vaut 2064 (x prend la valeur 16)

1. définissez un nouveau type de token nommé `SET`. Le motif du token sera celui d'un identificateur précédé d'une « flèche ». Par exemple `->ma_variable`
   La valeur associée au token sera l'identificateur rencontré, sans le préfixe `->` Dans notre exemple, la chaîne : `ma_variable`
2. complétez l'évaluateur d'expressions pour implémenter les opérations
   - `SET` qui associe l'entier situé en sommet de pile à la variable dont le nom est porté par le token SET

   Vous rangerez les valeurs des variables dans un dictionnaire python.

— `IDENT` qui ajoute en sommet de pile l'entier associée à la variable, ou 0 si aucun entier ne lui est associé.
On rappelle qu'un dictionnaire python dispose de la méthode get qui permet de spécifier une valeur par défaut : `get(clé,valeur_par_defaut)`

Expression arithmétique postfixée

Dans une expression arithmétique **en notation postixée**, les opérandes sont écrites avant l'opérateur. Par exemple la valeur asociée à $\boxed{3\ 5\ +}$ est 8 (équivaut à la notation infixée classique : $3 + 5$).
Pour évaluer une expression postfixée, on procède de gauche à droite en remplaçant au fur et à mesure chaque opérateur par le résultat de son évaluation. En procédant de cette manière, chaque opérateur s'applique aux 2 valeurs qui le précèdent. En notation postfixée, il n'y a pas de notion de priorité.
Exemples :

— $\boxed{3\ 5\ +\ 2\ *} \rightarrow \boxed{8\ 2\ *} \rightarrow \boxed{16}$
— $\boxed{3\ 5\ 2\ +\ *} \rightarrow \boxed{3\ 7\ *} \rightarrow \boxed{21}$
— $\boxed{3\ 5\ 2\ *\ +} \rightarrow \boxed{3\ 10\ +} \rightarrow \boxed{13}$

Le nombre d'opérandes requis par un opérateur est fixe. Ce nombre est appelé **arité**. L'arité de $\boxed{+}$ est 2. Idem pour celle de $\boxed{*}$
Une expression postfixée peut être incorrecte : soit par manque d'opérandes, soit par excès. À la fin de l'évaluation il devrait rester une et une seule valeur, et aucun opérateur.

— $\boxed{4\ 3\ 5\ +\ 2\ *} \rightarrow \boxed{4\ 8\ 2\ *} \rightarrow \boxed{4\ 16}$ expression incorrecte (trop d'opérandes).
— $\boxed{5\ 2\ +\ *} \rightarrow \boxed{7\ *}$ expression incorrecte (manque une opérande).

D'une manière algorithmique, l'évaluation utilise une **pile de nombres**. Initialement la pile est vide puis, en lisant l'expression de gauche à droite :

— à la lecture d'une valeur : l'ajouter sur la pile.
— à la lecture d'un opérateur : enlever de la pile les opérandes nécessaires (selon l'arité de l'opérateur), effectuer l'opération et mettre le résultat sur la pile.
NB : si la pile ne contient pas le nombre d'opérandes nécessaires, c'est que l'expression est incorrecte.

À la fin, la pile ne devrait contenir qu'une seule valeur (le résultat de l'expression complète).