UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

TPs n°4
Structures et fonctions

Exercice n°1

Une pile est une structure de données de type LIFO (Last In, First Out – le dernier élément entrée est le premier sorti). Autrement dit, il s'agit d'une liste dans laquelle les ajouts et les suppressions n'ont lieu que sur une même extrémité appelée "sommet de pile", à l'image d'une pile d'assiettes, piles de livres, etc.
Une pile est manipulable au travers des 5 fonctions ci-dessous (EL désigne un élément de type quelconque, PILE une pile d'éléments EL) :

- `creerPile` : elle permet de créer une pile vide — ∅ → PILE
- `pileVide` : elle permet de tester si une pile est vide ou pas — PILE → bool
- `sommetPile` : elle retourne l'élément situé en sommet de pile — PILE → EL
- `empiler` : elle permet d'ajouter un élément en sommet de pile — PILE x EL → PILE
- `dépiler` : elle permet de supprimer l'élément situé en sommet de pile — PILE → PILE x EL

On choisit de représenter une pile par une structure composée d'un tableau de `MAX` cases contenant un caractère et du nombre de cases effectivement utilisées parmi les `MAX`.
Les ajouts et suppressions seront effectués en sommet de pile. Bien réfléchir (en termes notamment de performance) à l'endroit où doit se situer le sommet de pile dans une pile de type `pile`.

a) Définir une structure `pile` permettant de représenter une pile statique de caractères.

b) Écrire les fonctions associées à une pile (`creerPileVide`, `pileVide`, `sommetPile`, `empiler`, `dépiler` dont les entêtes sont ci-dessus).

c) Écrire une fonction `afficherPile` permettant d'afficher le contenu d'une pile.

d) Écrire une fonction `inverserPile` permettant d'inverser le contenu d'une pile.

e) Écrire un programme principal pour tester vos fonctions.

Exercice n°2

En mathématiques, la notation infixée des expressions arithmétiques est celle qui est la plus utilisée : (a+b), (a-b), plus généralement (terme opérateur terme) où un terme peut lui-même être une expression infixe ou un identificateur de variable. Dans le cadre de cet exercice, nous travaillerons sur des expressions complètement parenthésées, autrement dit nous travaillerons sur des expressions où chaque terme situé de part et d'autre d'un opérateur est entouré d'une parenthèse, sauf s'il s'agit d'un terme élémentaire c'est-à-dire d'un identificateur de variable.
Exemples : (a+b), ((a+b)*c), (a+(b/c)).

La notation postfixée est une notation dans laquelle les opérateurs suivent immédiatement les opérandes sur lesquels ils agissent.
Exemples : (a+b) s'écrit ab+ en notation postfixée, ((a+b)*c) s'écrit ab+c*, (a+(b/c)) s'écrit abc/+.

L'objectif de cet exercice est de traduire une expression infixée en notation postfixée. Pour cela, on suppose que :

- les opérateurs binaires utilisés sont la multiplication, la division entière, l'addition et la soustraction respectivement représentés par les caractères '*', '/', '+' et '-',
- chaque opérande est représenté par un unique caractère alphabétique ('a', 'b', 'c', …, 'z')

On utilisera pour cela deux piles :

- une première pile "p1" dans laquelle on empilera les opérandes au fur et à mesure de leur apparition dans l'expression et dans laquelle on construira l'expression postfixée,
- une deuxième pile "p2" dans laquelle on empilera les opérateurs.

Exemple : On souhaite traduire l'expression ((a+b)*c) en notation postfixée. Comme l'illustre la dernière pile p1, le résultat obtenu est ab+c*

Etat des piles après traitement	p1 / p2	p1 / p2	p1 / p2	p1 / p2	p1 / p2	p1 / p2	p1 / p2	p1 / p2	p1 / p2
			a	a +	b a +	+ b a	+ b a *	c + b a *	* c + b a
Caractère en cours de traitement	(	(	a	+	b	)	*	c	)

a) Écrire une fonction `postfixe` permettant de transformer une expression infixée donnée en paramètre sous forme de chaîne de caractères en une expression postfixée contenue dans une pile de type `pile` (exercice 1).

b) Compléter le programme principal pour tester votre fonction.

Exercice n°3 : pour les plus rapides

On suppose l'expression postfixée "ab*c+ ". En attribuant respectivement des valeurs (a = 2, b = 3 et c = 4) aux opérandes, on obtient l'expression postfixée 23*4+ qui sera évaluée comme indiqué ci-dessous :

Etat de la pile après traitement	2	3 2	6	4 6	10
Caractère en cours de traitement	a	b	*	c	+

a) On propose de créer un tableau de `MAXLET` (26) cases (une case pour chaque lettre de l'alphabet, la case d'indice 0 contiendra la valeur de la lettre 'a', la case d'indice 1 contiendra la valeur de la lettre 'b', …, la case d'indice 25 contiendra la valeur de la lettre 'z').
Écrire une fonction `initTab` permettant d'initialiser ce tableau, en mettant -1 dans chacune de ses cases.

b) Écrire une fonction `valeur` qui demande à l'utilisateur la valeur associée à une lettre donnée et inscrit cette valeur dans la bonne case du tableau (la case d'indice 0 pour la lettre 'a', la case d'indice 1 pour la lettre 'b', …, la case d'indice 25 pour la lettre 'z'. Cette fonction retournera la valeur saisie par l'utilisateur.

c) Écrire une fonction `calcul`, qui, à partir de deux opérandes et un opérateur (supposés former une expression postfixe), calcule la valeur de l'expression postfixe considérée.

d) Écrire une fonction `évaluation` évaluant une expression postfixée contenue dans une pile de caractères représentée par une structure de type `pile` (cf. exercice 1).

c) Compléter le programme principal pour tester vos fonctions.