TD n° 2 : Les TDA PILE et FILE

(D'après E. DELOZANNE, P. JACOBONI)

Objectifs du TD1 :

- Assimiler les notations du cours et les différentes réalisations des listes et des piles étudiées en cours
- Exercice 1 : Préparer le TP2 et utiliser des TDA sans en modifier le code
- Exercice 2 : Exemple de question d'examen ; s'entraîner à être un virtuose de la manipulation de pointeurs en C

Exercice 1 : Évaluation d'une expression arithmétique infixée

Le but de cet exercice est d'écrire un programme qui évalue une expression arithmétique. La démarche classique consiste à traduire dans un premier temps la notation infixée (usuelle) de l'expression en une notation polonaise postfixée, puis dans un deuxième temps, à évaluer l'expression postfixée. Les éléments d'une expression, dans une notation ou dans une autre, sont représentés par une liste d'objets de type ELEMENT.

On utilisera les TDA LISTE, PILE et ELEMENT étudiés en cours.

Exemples :

expression infixée : (, 12, + , 2 , * , 3, - , 10 , * , 4,), /, 2

expression postfixée : 12 , 2, 3, *, + , 10, 4 , *, -, 2, /

résultat de l'évaluation : -11

Algorithmes informels :

a) Traduction infixée vers postfixéz

entrée : LI, liste en notation infixée ;
sortie : LP, liste en notation postfixée ;
1) Créer une nouvelle liste LP
2) Créer une pile
3) Parcourir LI du début à la fin
soit x l'élément courant
Au cas où
• x est une parenthèse ouvrante, l'empiler
• x est un nombre, l'insérer à la fin de LP
• x est un opérateur
tant que le sommet de pile a une priorité ≥ à la priorité de x
le dépiler et l'insérer à la fin de LP
empiler x
• x est une parenthèse fermante, dépiler (jusqu'à ouvrante compris) et insérer à la fin de LP (sauf la parenthèse ouvrante)
passer à l'élément suivant dans LI
4) Dépiler (jusqu'à pile vide) et insérer tous les opérateurs dépilés à la fin de LP

b) Évaluation de l'expression postfixée

entrée : L une liste, expression en postfixée ;
sortie : un élément,représentant la valeur de l'expression
1) Créer une pile
2) Parcourir la liste L du début à la fin
soit x l'élément courant
Si x est un nombre,
alors l'empiler
sinon (c'est un opérateur)
dépiler l'opérande droit
dépiler l'opérande gauche
appliquer l'opérateur à ces deux opérandes
empiler le résultat
passer à l'élément suivant dans L
3) Retourner le sommet de pile

Première question : Simuler les algorithmes Inf2Post et EvalPost sur l'expression :

7 * 3 - (5 + 3 - 4)

(a + b + c * d) * e

Deuxième question : On fait les hypothèses suivantes

1/ On dispose des primitives sur les listes et sur les piles

2/ On dispose de cinq prédicats

- int Nombre(ELEMENT x) retourne vrai (1) si x est un nombre 0 sinon
- int Operateur(ELEMENT x) retourne vrai (1) si x est un opérateur 0 sinon
- int Ouvrante(ELEMENT x) retourne vrai (1) si x est une parenthèse ouvrante, 0 sinon
- int Fermante(ELEMENT x) retourne vrai (1) si x est une parenthèse fermante, 0 sinon
- int Prioritaire(ELEMENT x1, ELEMENT x2) retourne vrai (1) si x1 a une priorité supérieure ou égale à celle de X2, 0 sinon

3/ L'élément "(" a une priorité de moins l'infini

4/ On dispose d'une fonction

- ELEMENT Applique(ELEMENT op, ELEMENT xg, ELEMENT xd)
  qui retourne l'élément correspondant à l'application de l'opérateur op aux deux opérandes gauche (xg) et droit (xd).
- int ElementToEntier (ELEMENT x) convertit l'élément en entier
- char ElementToCar (ELEMENT x) convertit l'élément en caractère
- ELEMENT EntierToElement (int x) convertit l'entier en élément

Écrire les algorithmes Inf2Post et EvalPost en utilisant les primitives des TDA LISTE et PILE

Exercice 2 : Files circulaires (5 points de l'examen de janvier 1998)

Écrire les primitives nécessaires à une mise en œuvre des files circulaires dans une réalisation par pointeur sur une cellule qui pointe sur celle qui contient le dernier élément de la liste, cette dernière cellule pointe sur celle qui contient le premier élément.

Exercice 3 : Files à double sens

Une file à double sens est une liste où l'insertion et la suppression peuvent être opérées aux deux extrémités. Donner les implantations par tableau et par pointeur de ce TDA.

```c
/*************************************************************
*       Fichier :   PILEPRIM.H
*       Format :    Source C
*       Version  :  21/9/96
*       Programmeurs :  Delozanne/Jacoboni, Futtersack
*       Contenu :       Déclaration des primitives du TDA PILE
*
*************************************************************/

/* Indépendant de la réalisation*/
#ifndef PILEPRIM_H
#define PILEPRIM_H

#include "eltprim.h"    /*   le TDA PILE utilise le TDA ELEMENT */

#include "pilesdd.h"    /* inclure le fichier où est déclarée la représentation inter

PILE PileCreer (int profondeur) ;
   /* crée et retourne une pile en lui allouant de la mémoire dynamique
   La pile ainsi crée est pour l'instant vide
   Le paramètre entier désigne la profondeur maximale de la pile (pleine)*/
void PileDetruire(PILE);
   /* libére la mémoire allouée dynamiquement pour la pile paramètre*/
bool PileVide(PILE);
   /* teste si la pile est vide */
ELEMENT PileSommet(PILE);
   /* retourne l'element au sommet de la pile (sans le dépiler) */
ELEMENT PileDepiler(PILE);
   /* retourne l'élément au sommet de la pile ET l'enlève de la pile) */
bool PileEmpiler(ELEMENT, PILE);
   /* ajoute l'élément au sommet de la pile */
void PileAfficher (PILE ) ; /* pour test et mise au point */
/* void PileRaz(PILE) ;
void PileNettoyer(PILE) ;  ça peut être utile à faire en exercice */
#endif
```



```c
/*************************************************************
*       Fichier :   FILEPRIM.H
*       Format :    Source C
*       Version  :  21/9/96
*       Programmeurs :  Delozanne/Jacoboni
*       Contenu :       Déclaration des primitives du TDA FILE
*                       (indépendant de la représentation)
*************************************************************/

#ifndef FILEPRIM_H                  /* pour inclusion conditionnelle */
#define FILEPRIM_H
#include "eltprim.h"            /* le TDA FILE utilise le TDA ELEMENT */
#include "filesdd.h"            /*   inclure le type concret Pile */

FFILE FileCreer (int profondeur) ;
   /* crée et retourne une file vide en lui allouant dynamqiuement de la mémoire */
void FileDetruire(FFILE);
   /* libère la mémoire allouée pour la file */
int FileVide(FFILE);
   /* teste si la file est vide */
ELEMENT FileDebut(FFILE);
   /* retourne l'élément au début de la file sans le retirer de la file */
ELEMENT FileSortir(FFILE);
   /* retourne l'élément au début de la file en le retirant de la file */
int FileEntrer(ELEMENT, FFILE);
   /* place l'élément à la fin de la file */
void FileAfficher (FFILE ) ; /* pour test et mise au point */

#endif

```