Piles, files et applications

1 Implémentation d'un paquetage de gestion d'une pile

Une pile est une structure de données permettant de stocker des objets et que l'on manipule à l'aide de deux opérations :

- empiler : ajoute un objet en haut de la pile
- depiler : retire l'objet situé en haut de la pile

La mise en oeuvre d'une pile peut se faire soit à l'aide d'un tableau, soit à l'aide d'une liste chaînée. C'est de cette dernière façon que nous allons procéder.

Vous devez écrire un paquetage `paq_pile` respectant la spécification suivante:

```
with ada.unchecked_deallocation;

package paq_pile is

   type PILE is private;

   -- ajoute un element en haut de la pile
   procedure empiler (p : in out PILE; elt : in NATURAL);

   -- supprime l'element situe en haut de la pile
   procedure depiler (p : in out PILE; elt : out NATURAL);

   -- retourne le nombre d'éléments de la pile
   function hauteur (p : PILE) return NATURAL;

   -- vrai si la pile ne contient pas d'elements
   function est_vide (p : PILE) return BOOLEAN;

   -- affiche la pile a l'ecran
   procedure afficher (p : PILE);

private
   type BLOC;
   type PILE is access BLOC;
   type BLOC is record
     elt : NATURAL;
     en_dessous : PILE;
   end record;
end paq_pile;
```

2 Calcul de la suite de Fibonacci

Il est possible de simuler le calcul de la suite de Fibonacci à l'aide d'une pile. L'idée est d'utiliser la pile pour empiler les problèmes à traiter. Ainsi calculer $fibo(n) = fibo(n-1) + fibo(n-2)$ consiste à :

1. empiler n
2. dépiler n : si $n = 0$ ou 1, ajouter n au résultat, sinon empiler $n-1$ et $n-2$
3. continuer 2 jusqu'à ce que la pile soit vide

Implantez ce calcul en affichant la pile à chaque fois.

Question 1 Que représente le nombre d'appels à dépiler ?

Question 2 Que représente la hauteur maximale de la pile ?

3 Expressions bien parenthésées

On s'intéresse à la vérification syntaxique d'expressions constituées de parenthèses. On veut donc vérifier que chaque parenthèse ouvrante possède une parenthèse fermante qui lui correspond et qu'il n'existe pas trop de parenthèses fermantes.

Modifier le paquetage `pile` pour que les éléments soient des `CHARACTER`. Implantez une fonction de test qui lit une expression écrite avec les parenthèses `() {} []` et qui indique si elle est bien parenthésée et sinon qui indique où est l'erreur.

4 Les files

Une autre structure de données, très similaire à la pile est la file. Dans la pile le premier élément sorti est celui qui est au dessus de la pile, c'est à dire le dernier entré (First In Last Out); dans une file c'est le contraire : le premier élément sorti est le premier entré (First In First Out). Implantez un paquetage `paq_file` de gestion d'une file.

Question 3 On se donne trois piles, P_1, P_2 et P_3. P_1 contient une suite d'entiers. Ecrivez une procedure qui met dans P_2 les entiers pairs de P_1 et laisse dans P_1 les entiers impairs. Attention, les entiers doivent être dans le même ordre qu'au début. Par exemple, si l'affichage de P_1 donnait la suite $1, 2, 3, 4, 5, 6$, l'affichage de P_1 doit donner $1, 3, 5$ et celui de P_2 doit donner $2, 4, 6$.

Question 4 On refait le même exercice avec deux piles P_1 et P_2 et une file F_3 à la place de la pile P_3.