Université Paris-Nord
Institut Galilée

Algorithmique, arbres et graphes
Licence 2 - Année 2005-2006

TD 5 & 6 : Structures de données abstraites

Exercice 1 (Listes Chaînées)
Soit la structure ***liste*** *définie en C par :*

```
typedef struct cellule_s{
  element_t element;
  struct cellule_s *suivant;
} cellule_t;
typedef cellule_t * liste_t;
```

1. *Écrire un algorithme récursif (et itératif) qui permet de fusionner deux listes triées dans l'ordre croissant et retourne la liste finale. On pourra utiliser la fonction int cmpListe(liste_t $l1$, liste_t $l2$); qui retourne* 1 *si le premier élément de* $l1$ *est inférieur au premier élément de* $l2$, 0 *s'ils sont égaux et -1 sinon.*
2. *Écrire un algorithme qui permet d'éliminer toutes les répétitions dans une liste chaînée.*

Exercice 2 (Primitives de pile)

1. *Définir une structure pile à l'aide d'un tableau d'éléments (de type element_t) de hauteur maximum* N. *On considérera que* N *est une constante donnée.*
2. *Écrire les fonctions suivantes :*
   - *pile_t creerPile() ; qui crée une pile vide,*
   - *int pileVide(pile_t pile) ; qui retourne* 1 *si la pile est vide et* 0 *sinon,*
   - *element_t depiler(pile_t pile) ; qui retourne le dernier élément après l'avoir retiré de la pile,*
   - *void empiler(pile_t pile, element_t elt) ; qui empile l'élément elt,*
   - *element_t dernierElement(pile_t pile) ; qui retourne le dernier élément empilé,*
   - *void viderPile(pile_t pile) ; qui vide la pile,*
   - *unsigned int hauteurPile(pile_t pile) ; qui retourne la hauteur de la pile.*

Exercice 3 (Déplacement de pile)
On se donne trois piles P_1, P_2 *et* P_3. *La pile* P_1 *contient une suite de nombres entiers positifs. Pour la suite de l'exercice, on utilisera la structure et les opérations définies dans l'exercice précédent avec des éléments de type entier.*

1. *Écrire un algorithme pour déplacer les entiers de* P_1 *dans* P_2 *de façon à avoir dans* P_2 *tous les nombres pairs au dessus des nombres impairs.*
2. *Écrire un algorithme pour copier dans* P_2 *les nombres pairs contenus dans* P_1. *Le contenu de* P_1 *après exécution de l'algorithme doit être identique à celui avant exécution. Les nombres pairs doivent être dans* P_2 *dans l'ordre où ils apparaissent dans* P_1.

Exercice 4 (Test de bon "parenthésage")
Soit une expression mathématique dont les éléments appartiennent à l'alphabet suivant :
$\mathcal{A} = \{0, \ldots, 9, +, -, *, /, (,), [,]\}$.
Écrire un algorithme qui, à l'aide d'une unique pile d'entiers, vérifie la validité des parenthèses et des crochets contenus dans l'expression. On supposera que l'expression est donnée sous forme d'une chaîne de caractères terminée par un zéro. L'algorithme retournera 0 *si l'expression est correcte ou* -1 *si l'expression est incorrecte.*

Exercice 5 (Calculatrice postfixe)
On se propose de réaliser une calculatrice évaluant les expressions en notation postfixe. L'alphabet utilisé est le suivant : $\mathcal{A} = \{0, \ldots, 9, +, -, *, /\}$ *(l'opérateur* $-$ *est ici binaire). Pour un opérateur n-aire* P *et les opérandes* $O_1, \ldots, O_n$*, l'expression, en notation postfixe, associée à* P *sera :* $O_1, \ldots, O_n P$*. Ainsi, la notation postfixe de l'expression* $(2 * 5) + 6 + (4 * 2)$ *sera :* $2\ 5\ *\ 6\ +\ 4\ 2\ *\ +$.
On suppose que l'expression est valide et que les nombres utilisés dans l'expression sont des entiers compris entre 0 *et* 9*. De plus, l'expression est donnée sous forme de chaînes de caractères terminée par un zéro. Par exemple* $(2 * 5) + 6 + (4 * 2)$ *sera donnée par la chaîne "*$25 * 6 + 42 * +$*".*
Écrire un algorithme qui évalue une expression postfixe à l'aide d'une pile d'entiers. (On pourra utiliser la fonction int ctoi(char c){return (int)(c-'0');} pour convertir un caractère en entier).

Exercice 6 (Primitives de file)
On souhaite implémenter une file à l'aide d'un tableau circulaire.

1. *Définir une structure file à l'aide d'un tableau circulaire d'éléments (de type element_t) de taille* N*. Comment doit-on choisir* N *pour que la navigation dans le tableau circulaire se ramène à des opérations très élémentaires en informatique ?*
2. *Écrire les fonctions suivantes :*
   - *file_t creerFile() ; qui crée une file vide,*
   - *int fileVide(file_t file) ; qui retourne* 1 *si la file est vide et* 0 *sinon,*
   - *element_t defiler(file_t file) ; qui retourne le premier élément après l'avoir retiré de la file,*
   - *void enfiler(file_t file, element_t elt) ; qui enfile l'élément elt,*
   - *element_t premierElement(file_t file) ; qui retourne le premier élément enfilé,*
   - *void viderFile(file_t file) ; qui vide la file,*
   - *unsigned int tailleFile(file_t file) ; qui retourne la taille de la file.*

Exercice 7 (Tri avec files)
On se donne une file d'entiers que l'on voudrait trier avec le plus grand élément en fin de file. On n'est autorisé à utiliser que fileVide et les opérations suivantes :

- *defilerEnfiler : Défile le premier élément de la première file et l'ajoute à la deuxième file.*
- *comparer : Retourne 1 si le premier élément de la première file est plus grand ou égal au premier élément de la deuxième file et 0 sinon.*

1. *À partir des primitives de la structure file, proposer un algorithme pour chacune des deux dernières opérations (fileVide ayant été définie dans l'exercice précédent).*
2. *Donner un algorithme de tri qui utilise seulement ces trois opérations et 3 files. La pile* f_1 *contiendra les entiers à trier,* f_2 *contiendra les entiers triés après exécution et la file* f_3 *pourra servir de file auxiliaire. On pourra, aussi, utiliser la fonction void permuteFile(file_p f1, file_p f2) qui permute le contenu des deux files.*
3. *Le tri est-il stable ?*