Université Bordeaux 1

LSTS Informatique $2^{\text{ème}}$ année

INF251 Algo 1
Devoir surveillé n°1
30mn
Aucun document autorisé
06/10/2008

Question de cours : Donner la définition d'une pile.

Exercices :

1. Soit une liste de caractères stockés dans une liste doublement chainée définie par une implémentation dynamique :

```
type curseur = ^cellule

cellule = structure
     valeur : car
     suiv : curseur
     prec : curseur
Finstructure
```

```
ListeDC_car = structure
     premier : curseur
     dernier : curseur
     cle : curseur
Finstructure
```

Soit la fonction `mystere` suivante :

```
1 fonction mystere (ref L1, L2 :ListeDC_car):vide;
2
3 début
4    si (L1.premier et L2.premier) alors
5         Debutliste(L1) ;
6         L1.dernier^.Suiv =L2.premier ;
7         L2.premier^.Prec =L1.dernier ;
8         L1.dernier = L2.dernier;
9         L2.premier = NIL ;
10        L2.dernier = NIL ;
11        L2.clé = NIL ;
12
13   Finsi
14 FinFonction
```

1.1. Soit M1 et M2 deux listes doublement chainées de caractères contenant respectivement les caractères suivants : M1= (C,O, N, T, R, O, L,E) et M2=(C, O, N, T, I, N, U). Quel sera le résultat de `mystere(M1,M2)`.

1.2. Que fait la fonction `mystere` dans le cas général ?

1.3. Quelle est la complexité de cette fonction ?

1.4. Quel est le rôle des instructions 9, 10 et 11 ?

2. Soit une liste de caractères stockés dans une liste simplement chainée définie par une implémentation dynamique :

```
   type curseur = ^cellule

cellule = structure
     valeur : car
     suiv : curseur
Finstructure
```

```
ListeSC_car = structure
     premier : curseur
     cle : curseur
Finstructure
```

2.1. Ecrire la primitive : `supprimerApres(ref L : listeSC_car) :vide`

2.2. Soit L une liste de caractères triée par ordre croissant.
En utilisant les primitives du type abstrait « liste simplement chainée » écrire la fonction :

```
ajouter_un_element (ref L : listeSC_car, val X : car) :vide

/*ajoute un element à la liste en respectant l'ordre*/
```

2.3. Quelle est la complexité de votre fonction ?

Liste accès :

```
fonction valeur(val L:liste d'objet):objet;
/* si la clé==NIL alors le résultat est NULL */

fonction debutListe(ref L:liste d'objet) :vide ;
/* positionne la clé sur le premier objet de la liste */

fonction suivant(ref L:liste d'objet) :vide ;
/* avance la clé d'une position dans la liste */

fonction listeVide(val L:liste d'objet): booleen;
/* est vrai si la liste ne contient pas d'élément */
```

Liste modification :

```
fonction créerListe(ref L:liste d'objet):vide;

fonction insérerAprès(ref L:liste d'objet; val x:objet;):vide;
/* insère un objet après  la clé, la clé ne change pas */

fonction insérerEnTete(ref L:liste d'objet, val x:objet):vide;
/* insère un objet en début de liste la clé est positionnée sur la tête
de liste */

fonction supprimerAprès(ref L:liste d'objet):vide;
/* supprime  l'objet après  la clé, la clé ne change pas */

fonction supprimerEnTete(ref L:liste d'objet):vide;
/* supprime un objet en début de liste , la clé est positionnée sur la
tête de liste */

fonction detruireListe(ref L:liste d'objet):vide;
```

Doublement Chainée :

```
fonction finListe(ref L:liste d'objet):vide;
/* positionne la clé sur le dernier objet de la liste */

fonction précédent(ref L::liste d'objet): vide;
/* recule la clé d'une position dans la liste */
```