

Contrôle continu 2007-2008
Pointeurs - Récursivité – Listes
Corrigé

Questions de cours (5 points)

Soit une liste de taille n .

- Quelle est la complexité en temps de la recherche du dernier élément si la liste est simplement chaînée ?

La complexité $O(n)$

- Quelle est la complexité en temps de la recherche du dernier élément si la liste est doublement chaînée ?

La complexité $O(1)$

- Dans une liste doublement chaînée, pourquoi n'y a-t-il pas de primitive pour supprimer l'élément avant un élément donné ?

Parce qu'on peut réaliser cette opération en utilisant les primitives qui définissent le type abstrait

fonction supprimerAvant(ref L: listeDC de type_prédéfini; ref P: ^cellule) : vide ;

var PP, PPP: ^cellule ;

début

PP = précédent(L, P) ;

PPP = précédent(L, PP) ;

si PPP = NIL alors

supprimerEnTête(L)

sinon

supprimerAprès(L, PPP)

finsi

fin

- Dessinez la mémoire, après la suite d'opérations suivante :

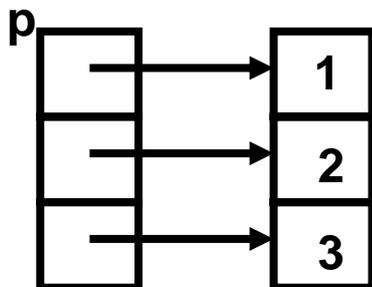
`var p : tableau[1..3] de ^entier ;`

`pour i=1 à 3 faire`

`new(p[i]) ;`

`p[i]^=i ;`

`finpour`



Exercice 1 (3 points)

Soit la fonction

```

fonction essai(ref n ; entier, val p :e.tier) ;entier ;
début
  si n==0 alors
    retourner(p+1)
  sinon
    instruction
  Finsi
Fin
Finfonction

```

Pour chaque fonction définie par l'instruction ci-dessous dire si la fonction est en récursivité terminale ou non. Expliquez pourquoi.

- retourner(essai(n-1,p*n)+4)
Non récursive terminale à cause du « +4 »
- retourner(essai(n-1,p*n+4))
Récursive terminale car renvoie une constante le paramètre p ou une évaluation de la fonction elle-même.
- retourner(1+essai(n-p,p*n))
Non récursive terminale à cause du « 1+ »

Exercice 2 (12 points)

On dit qu'une liste L1 est préfixe d'une liste L2 si la liste L2 commence par la liste L1. Par exemple, la liste L1=(2,4) est préfixe de la liste L2=(2,4,6,8,9).

- Ecrivez une fonction qui teste si L1 est un préfixe de L2.

Algorithme itératif

fonction préfixe(ref L1,L2 :listeSC d'entiers) :booléen ;

var p1,p2 :^cellule ;

var B :booléen ;

début

si listeVide(L1) alors

retourner (VRAI)

sinon

si listeVide(L2) alors

retourner (FAUX)

sinon

B=VRAI ;

P1=premier(L1) ;

P2=premier(L2) ;

tantque contenu(P1)=contenu(P2) et B faire

P1=suivant(L1,P) ;

P2=suivant(L2,P) ;

b=non(P1==NIL) et non(P2==NIL)

fintantque

```

    si P1==NIL alors
        retourner(VRAI)
    sinon
        retourner(FAUX)
    finsi
finsi
finsi
fin

```

Algorithme récursif

```

fonction préfixeRec(ref L1,L2 :listeSC d'entiers ;ref P1,P2 :^cellule) :booléen ;
début
    cas où
        P1= = NIL
            retourner(VRAI)
        P2= = NIL alors
            retourner(FAUX)
        contenu(P1)=contenu(P2)
            retourner(VRAI)
        autrement
            retourner(prefixeRec(L1,L2,suivant(P1,L1),suivant(L2,P2))
    fin casou
fin

```

fonction préfixe (ref L1,L2 :listeSC d'entiers) :booléen ;

```

début
    préfixe(L1,L2,premier(L1),premier(L2))
fin

```

- Ecrire une fonction qui calcule la liste L2 privée de la liste L1 (dans l'exemple (6,8,9))

```

fonction reste(ref L1,L2 :listeSC d'entiers) :listeSC d'entiers ;
var p1:^cellule ;
var L:listesC d'entiers;
début
    L=L2 ;
    si préfixe(L1,L2) alors
        p=premier(L1) ;
        tant que p<>NIL faire
            supprimerEnTete(L) ;
            p=suivant(L1,P) ;
        fintantque
    finsi
    retourner (L)
fin

```

Listes simplement chaînées (listeSC)

```
fonction premier(val L:type_liste):^type_predefini;  
fonction suivant(val L:type_liste; val P:^type_predefini):^type_predefini;  
fonction listeVide(val L:type_liste):booléen;  
fonction créer_liste(ref L:type_liste):vide;  
fonction insérerAprès(val x:type_prédéfini;ref L:type_liste; val P:^type_predefini):vide;  
fonction insérerEnTete(val x:type_prédéfini;ref L:type_liste):vide;  
fonction supprimerAprès(ref L:type_liste;val P:^type_predefini):vide;  
fonction supprimerEnTete(ref L:type_liste):vide;
```

Listes doublement chaînées (listeDC), On ajoute les primitives suivantes

```
fonction dernier(val L:type_liste):^type_predefini;  
fonction précédent(val L:type_liste; val P:^type_predefini):^type_predefini;
```