Contrôle continu 2008-2009
Pointeurs - Récursivité - Listes

Questions de cours (4 points)

- Soit une liste de taille n. Quelle est l'occupation en mémoire d'une liste doublement chainée ?

*Si p est la taille d'un curseur et t la taille de l'information à stocker (2*p+t)*n, complexité O(n).*

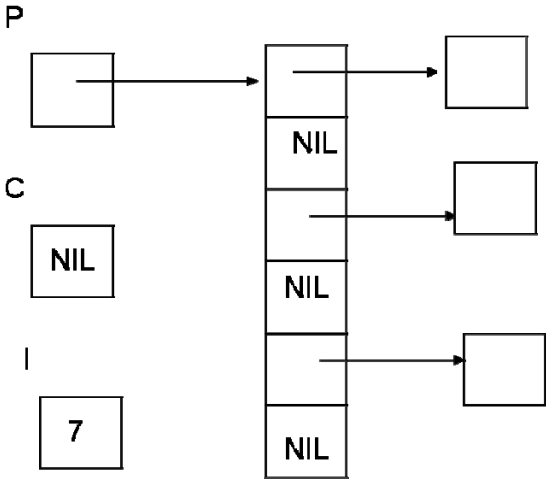
- Quelle est la différence entre le type « pointeur » et le type « curseur » ?

Un pointeur contient une adresse mémoire qui permettra d'accéder à un objet.
Un curseur est un objet qui permet d'accéder à un objet. Un curseur peut être de type pointeur mais peut aussi être de type entier (index dans un tableau)

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

```
var p : ^tableau[1..6] de ^entier ;
var c :^entier ;
var i :entier ;
new(p) ;
pour i=1 à 6 par pas de 2 faire
   new(p^[i]) ;
   p^^[i]=i ;
finpour
```

L'instruction p^^[i]=i n'a pas de sens car c'est p^ qui est un tableau et pas p^^ !



Dans ce contexte, quel est le problème de la séquence suivante ?

```
var l: entier ;
  l=p^[2]^
```

Cette instruction accède au 2ième élément du tableau pointé par p mais celui-ci est à NIL, donc il n'y a pas d''entier associé à cette cellule.

Exercice 1 (4 points)

Soit la fonction

```
fonction compte(ref T:tableau[1..NMAX] d'entier,val clé :entier) : entier ;
 var p,tmp :entier ;
   début
    tmp=0 ;
```

```
pour p allant de 1 à NMAX faire
        si T[p]<=clé alors
                tmp=tmp+1 ;
        finsi ;
 finpour ;
 retourner(tmp) ;
fin ;
```

Ecrire cette fonction sous une forme récursive.

Il suffit de manipuler le tableau comme une liste et de remplacer la boucle pour par l'appel récursif. Autrement dit un tableau est soit réduit à un élément soit c'est un élément suivi de ceux qui restent. Pour gérer la boucle dans la fonction récursive il faut ajouter des paramètres à la fonction : l'indice du tableau et le compteur.

```
Fonction compteRec(ref T :tableau[1..NMAX] d'entier,val clé :entier,
                   Val tmp :entier ;val I :entier) :entier ;
 Début
  Si I>NMAX alors
    Retourner(tmp)
 Sinon
   Si T[I]>clé alors
     Retourner(compteRec(T,clé,tmp,I+1)) ;
   Sinon
     Retourner(compteRec(T,clé,tmp+1,I+1))
  Finsi
finsi
Fin
```

L'appel se fait par compteRec(T,clé,0,1).

A noter que cette fonction est moins efficace que la fonction itérative (mémoire en O(1)) puisque l'environnement de la fonction est créé à chaque appel (mémoire en O(NMAX). Cependant, elle est récursive terminale, donc un bon compilateur la remet en forme itérative.

Exercice 2 (12 points)

On dit qu'une liste L1 est suffixe d'une liste L2 si la liste L2 finit par la liste L1. Par exemple, la liste L1=(2,4) est suffixe de la liste L2=(6,8,9, 2,4). On considèrera que la liste L1 est plus petite que la liste L2.

- Ecrivez une fonction qui teste si L1 est un suffixe de L2 pour une liste simplement chainée.

```
Fonction suffSC(ref L1,L2 :listeSC d'objet) :booleen ;
    Var signal :booleen ;
   Var L3,L4 :listeSC d'objet ;
   Début
      L3=mirroir(L1) ;
      L4=mirroir(L2) ;
     tantque !estFinListe(L3)et valeur(L3)=valeur(L4)  faire
              suivant(L3) ;
              suivant(L4) ;
     fintantque ;
     signal=estFinListe(L3) ;
    détruireLIste(L3) ;
```

```
    détruireLIste(L4) ;
   retourner(signal)
  fin
fonction mirroir(ref L :listeSC d'objet) :listeSC d'objet ;
  var LC :listeSC d'objet ;
  début
   créerListe(LC) ;
  débutListe(L) ;
  tantque !estFinListe(L) faire
     ajouterEnTete(LC,valeur(L)) ;
    suivant(L) ;
  fintantque
  debutListe(LC) ;
  retourner(LC)
fin
```

- Ecrivez une fonction qui teste si L1 est un suffixe de L2 pour une liste doublement chainée.

```
 Fonction suffDC(ref L1,L2 :listeDC d'objet) :booleen ;
  Début
     finListe(L1) ;
     finListe(L2) ;
     tantque valeur(L1) !=NULL et valeur(L1)=valeur(L2)  faire
              précédent(L1) ;
              précédent(L2) ;
     fintantque ;
   retourner(estDebutLIste(L1)))
  fin
```

Primitives et Fonctions

Liste Simplement Chainées

fonction valeur(val L:liste d'objet):objet;
fonction debutListe(ref L:liste d'objet);
fonction suivant(ref L:liste d'objet);
fonction listeVide(val L:liste d'objet): booleen;
fonction créerListe(ref L:liste d'objet):vide;
fonction insérerAprès(ref L:liste d'objet; val x:objet;):vide;
fonction insérerEnTete(ref L:liste d'objet,val x:objet):vide;
fonction supprimerAprès(ref L:liste d'objet):vide;
fonction supprimerEnTete(ref L:liste d'objet):vide;
fonction setCléListe(ref L: liste d'objet;val c:curseur):vide;
fonction detruireListe(ref L:lisetd'objet):vide;

Liste Doublement Chainées (ajout de)

fonction finListe(ref L:liste d'objet):vide;
fonction précédent(ref L::liste d'objet): vide;

Fonctions sur les listes

fonction estFinListe(val L:liste d'objet):booléen;
fonction chercher(ref L:liste d'objet; ref x:objet): booleen;
fonction supprimer(ref L:liste d'objet; ref x:objet): vide;