Université Bordeaux 1 LST Mention Informatique Semestre 3 2011-2012

Algorithmique 1 : Devoir Surveillé 1

Types de données abstraits. Listes.

Durée : 45mn
Sans documents

Exercice 1.1 *Récursivité*
Soit les fonctions `f` et `g`.

```
fonction f(val a: entier, val b: entier): entier;
debut
  si b == 0 alors
    retourner a;
  sinon
    retourner f(a+1, b-1);
  fsi
fin

fonction g(val a: entier, val b: entier): entier;
debut
  si b == 0 alors
    retourner a;
  sinon
    retourner 1 + g(a, b-1);
  fsi
fin
```

1. Donner les résultats des appels :
   `f(4,2)`, `f(5,1)`, `f(6,0)`, `g(4,2)`, `g(5,1)` et `g(6,0)`.
2. Quelle fonction utilise une récursivité terminale ?

Exercice 1.2 *Listes simplement chaînées*
Dans cet exercice on considère le type abstrait `listeSC` d'objets défini avec l'ensemble des primitives données en *annexe A*.
Écrire une fonction qui supprime de la liste `L` la deuxième occurrence de l'objet `X`.

```
fonction supprimer_deuxieme_occurrence(ref L:listeSC d'objet , val X: objet):vide
```

Soit une liste `L` de caractères, `L=<'a', 'b', 'c', 'b', 'b', 'd'>`.
L'appel `supprimer_deuxieme_occurrence(L, 'a')` laisse `L` inchangée.
Après l'appel `supprimer_deuxieme_occurrence(L, 'b')` , `L` est modifiée comme il suit, `L=<'a', 'b', 'c', 'b', 'd'>`.

Exercice 1.3 *Listes doublement chaînées*
Dans cet exercice on considère le type abstrait `listeDC` d'objets défini avec l'ensemble des primitives données en *annexe B*.
On définit le type `monome` comme il suit :

```
monome= structure
          degre: entier;
          coefficient: reel;
        finstructure
```

Le type `monome` sera utilisé pour représenter un terme d'un polynôme.
Un polynôme sera défini comme une liste `listeDC` de monomes.

```
polynome= listeDC de monome;
```

Par exemple le polynôme $p(x) = 2x^2 + 2x^1 - 3x^4 + 1 - 2x^2$ sera representé par
une liste `L`, `L=<` m_0, m_1, m_2, m_3, m_4`>`,
$m_0 = (2, 2.), m_1 = (1, 2.), m_2 = (4, -3.), m_3 = (0, 1.), m_4 = (2, -2.)$.

1. Écrire une fonction `creerPolynome` qui construit un polynome, `P`, à partir d'un tableau de monomes, `Coef[1..n] tableau de monome`.

   ```
   fonction creerPolynome(ref P: polynome, val Coef[1..n]: tableau de monome):vide;
   ```

2. Écrire une fonction `reduirePolynome` qui reduit les monomes semblables d'un polynôme.
   Pour $p(x) = 2x^2 + 2x^1 - 3x^4 + 1 - 2x^2$ après reduction on aura $p(x) = 2x^1 - 3x^4 + 1$.
   `reduirePolynome(ref P: polynome):vide`

Annexe A **Liste simplement chaînée listeSC**

```
listeSC= liste d'objets;
fonction valeur(val L:listeSC) : objet;
fonction debutListe(ref L:listeSC) :vide;
fonction suivant(ref L:listeSC) : vide;
fonction listeVide(val L:listeSC) : booleen;
fonction getCleListe(val L: listeSC) : curseur;
fonction creerListe(ref L:listeSC) : vide;
fonction insererApres(ref L:listeSC, val x:objet;) : vide;
fonction insererEnTete(ref L:listeSC, val x:objet) : vide;
fonction supprimerApres(ref L:listeSC) : vide;
fonction supprimerEnTete(ref L:listeSC) : vide;
fonction setCleListe(ref L: listeSC, val c:curseur) : vide;
fonction detruireListe(ref L:listeSC) : vide;
```

Annexe B **Liste doublement chaînée listeDC**

```
listeDC= liste d'objets;
fonction valeur(val L:listeDC) : objet;
fonction debutListe(ref L:listeDC) : vide;
fonction finListe(ref L:listeDC): vide;
fonction suivant(ref L:listeDC) : vide;
fonction precedent(ref L:listeDC) : vide;
fonction listeVide(val L:listeDC) : booleen;
fonction getCleListe(val L:listeDC) : curseur;
fonction creerListe(ref L:listeDC) : vide;
fonction insererApres(ref L:listeDC, val L:objet;) : vide;
fonction insererEnTete(ref L:listeDC, val X:objet) : vide;
fonction supprimerApres(ref L:listeDC) : vide;
fonction supprimerEnTete(ref L:listeDC) : vide;
fonction detruireListe(ref L:listeDC) : vide;
fonction setCleListe(ref L:listeDC, curseur c) : vide;
```