Correction Examen d'algorithmique I
EISTI ING1
aucun document autorisé, durée 2heures

17 février 2010

1 Tableaux

Soit T un tableau de n nombres réels qui ne contient pas le 0 :

1. Écrire un algorithme avec complexité linéaire qui trie T de façon que tous les nombres positifs précèdent les nombres négatifs.
2. Prouver que l'algorithme a complexité linéaire.

```
procedure SeparePositifsNegatifs (ES tableau T(N) : reel, E n : entier)
Variables :
i : entier
j : entier

  i <- 1
  j <- n

tant que (i < j) faire
tant que (T(i) > 0 et i < j) faire
i <- i + 1
ftq
tant que (T(j) < 0 et i < j) faire
j <- j - 1
ftq
swap(T(i), T(j))
i <- i + 1
j <- j - 1
  ftq
fin procedure
```

Complexité linéaire car les boucles imbriquées font varier les mêmes indices dans le même sens.

2 Récursivité

Tout le monde le sait : 0 est l'élément neutre pour l'addition. Donc $0 = 0+0$. Nous pouvons l'écrire sous cette forme : $0 = (0+0)$, mais si l'on continue notre logique, nous pouvons continuer à remplacer les 0 par des $(0+0)$, ce qui nous donne : $0 = ((0+0)+(0+0))$, etc...

Nous allons, dans cet exercice, automatiser cette procédure, en supposant que l'on remplace N fois les zéros à droite de l'égalité "$0 = 0$", par $(0+0)$.

Exemple :
$N = 0 : 0 = 0$
$N = 1 : 0 = (0+0)$
$N = 2 : 0 = ((0+0)+(0+0))$

1. Écrire la procédure récursive qui lit un entier N, et affiche la chaîne correspondante.

```
Procédure Toto(N : Entier)
Variables :
Ecrire "0 = "
TotoRec(N)
Fin Procédure

Procédure TotoRec(N : Entier)
Variables :
Si N = 0 Alors
Ecrire "0"
Sinon
Ecrire "("
TotoRec(N-1)
Ecrire "+"
TotoRec(N-1)
Ecrire ")"
Fin Si
Fin Procédure
```

3 Structure et Liste linéaire

Un livre est écrit par un ou plusieurs auteurs. Il est composé d'un titre, d'une introduction, de chapitres et d'une conclusion. Une introduction, un chapitre et une conclusion ont un titre et sont formés de paragraphes. Un paragraphe est un ensemble de phrases. Un titre est une phrase. Une phrase est une chaîne de caractères.

1. On vous demande de définir une ou plusieurs structures utilisant à bon escient des listes linéaires.

```
Type Paragraphe = Struture
  phrases : CelluleLineaire de Chaine
Fin Structure

Type Corps = Structure
titre : Chaine
  Paragraphes : CelluleLineaire de Paragraphe
Fin Structure

Type Livre = Structure
  titre : Chaine
  Tableau auteurs() :  Chaine
  nbAuteurs : Entier
  introduction : Corps
  chapitres : CelluleLineaire de Corps
  conclusion : Corps
Fin Structure
```

2. Écrire une fonction qui permet de saisir au clavier soit un chapitre, soit une introduction, soit une conclusion.

```
fonction saisirCorps() : Corps
Variables locales
  corps : Corps
  carPhrase, carParagraphe : Caractere
  phrase : Chaine
  phraseCur : CelluleLineaire de Chaine
  paraCur : Paragraphe

ecrire("donner le titre")
lire(corps.titre)
nouvelle(corps.paragraphes)
paraCur <- corps.paragraphes
faire
nouvelle(paraCur.phrases)
phraseCur <- paraCur.phrases
  faire
    lire(phraseCur.info)
    ecrire("Autre phrase (o/n) ?")
    lire(carPhrase)
    Si carPhrase = 'o' Alors
      nouvelle(phraseCur.suivant)
      phraseCur <- phraseCur.suivant
    Fin Si
  tantque (carPhrase = 'o')
  ecrire('Autre paragraphe (o/n) ?')
  lire(carParagraphe)
```

```
  Si carParagraphe = 'o' Alors
    Nouvelle(paraCur.suivant)
    paraCur <- paraCur.suivant
  Fin Si
tantque (carParagraphe = 'o')
retourner corps
fin fonction
```

3. Écrire une fonction qui permet de saisir un livre.

```
fonction saisirLivre() : Livre
Variables locales
livre : Livre
nbAuteurs, noAuteur : Entier
carChapitre : Caractere
ChapitreCur : CelluleLineaire de Corps
  ecrire("donner le titre")
  lire(livre.titre)
  ecrire("donner le nombre d'auteurs")
  lire(livre.nbAuteurs)
  creerTabelau(livre.auteurs,livre.nbAuteurs)
  Pour noAuteur <- 1 a livre.nbAuteurs pas 1
    ecrire("Donner un auteur")
    lire(livre.auteurs(noAuteur))
  Fin Pour
  ecrire("Donner l'introduction")
  livre.introduction <- saisirCorps()
  ecrire("Donner les chapitres")
  nouvelle(livre.chapitres)
  chapitreCur <- livre.chapitres
  faire
    chapitreCur.info <- saisirCorps()
    ecrire("Autre chapitre (o/n) ?")
    lire(carChapitre)
    Si carChapitre = 'o' Alors
      Nouvelle(chapitreCur.suivant)
      chapitreCur <- chapitreCur.suivant
    Fin Si
  Tantque carChapitre = 'o'
retourner livre
fin fonction
```

4 Chaînes de caractères

Soit l'algorithme KMP suivant :

```
procedure KMP(E tableau T(N):caractere ,E tableau P(M):caractere ,
```

```
                E n:entier,E m:entier)
  variables
    tableau pi():entier
    i,q:entier
  pi ← calculPrefixe(P)
  q ← 0
  pour i ← 1 a n pas 1
    tant que q>0 et P(q+1) ≠ T(i) faire
      q ← pi(q)
    ftq
    si P(q+1)=T(i) alors
      q ← q+1
    fsi
    si q=m alors
      ecrire("le motif apparaît en position"+(i-m))
      q ← pi(q)
    fsi
  fpour
fprocedure
```

1. En vous inspirant de cet algorithme, écrire une fonction permettant de retourner le nombre d'occurrences d'un motif dans un fichier texte. Votre fonction prendra en paramètres le nom du fichier (chaîne de caractères) et le motif à chercher (tableau de caractères et sa taille).
2. Est-ce que la complexité de votre fonction est identique à celle de la procédure KMP (justifiez votre réponse).

Ce barème + AREL équivaut à un corrigé (aucune difficulté) :

```
Q1: 4 pt
 - 1 pt pour la gestion du fichier (ouverture, lecture, fermeture)
 - 1 pt pour la modification de kmp (comptage au lieu d'affichage)
 - 1 pt pour le passage de "procedure KMP" a "fonction retournant un entier"
 - 1 pt pour le global (respect de la syntaxe, clareté , ...)
Q2: 1 pt
 - la justification doit mettre en avant le danger de comparer
 des acces memoire (KMP) et des acces fichier.
```