DEUG MIAS 2 - année 2000-2001
Institut Galilée
Informatique

TD 4
Récursivité et manipulation d'expressions

On considère une fonction f() de 2 arguments. On appelle expression une séquence de '0', 'f','(', ',' et ')'. On dira qu'une expression est bien formée si elle s'écrit '0', ou bien f(expression, expression). Dans la suite, on dira « expression » pour « expression bien formée ».

1. Ecrire quelques expressions.

```
Solution : 0 ; f(f(0,0),0) ;...
```

2. Ecrire une fonction récursive qui vérifie si une expression est bien formée. L'expression sera stockée dans une chaîne de caractères.

```
Solution :

fonction Bien_formee_rec (e : chaine ; d,f :entier) :
boolean ;
debut
  si d>f alors retourne FAUX
  sinon
  si d=f alors retourne e[d]='0'
  sinon
    si non(e[d]='f' et e[d+1]='(' et e[f]=')') alors
      retourne FAUX
    fin si
    c fl 0 ;
    p fl d+1 ;
    repeter
      si e[p]='(' alors c fl c+1
      sinon si e[p] = ')' alors c fl c-1
            fin si
      fin si
      p fl p+1
    jusqua (c=0 et e[p]=',') ou p>f
    retourne p<=f et e[p]=','
             et Bien_formee_rec(e,d+1,p-1)
             et Bien_formee_rec(e,p+1,f-1) ;
  fin si
  fin si
fin

fonction Bien_formee(e :chaine) :boolean ;
```

```
debut
  retourne Bien_formee_rec(e,1,length(e)) ;
fin
```

3. On veut stocker une expression dans une structure de données plus maniable qu'une chaîne de caractères. Pour cela, on va utiliser un tableau dont chaque case sera :

- Soit 0
- Soit f, suivi des indices des 2 cases où commencera la description des 2 arguments.

On convient que la description commence à la case 1.

a) Ecrire le type Texpression

Solution :

```
type Tcase = enregistrement
               zero : boolean ;
               fg,fd : entier ;
             fin enregistrement ;
type Texpression = enregistrement
                     long : entier
                     tab : tableau(1..N) de Tcase ;
                   fin enregistrement ;
```

b) Exhiber le tableau qui stocke f(f(0,f(0,0)),0)

c) Ecrire une procédure récursive qui affiche une expression

Solution :

```
procedure Affiche_rec(e : Texpression ; i : entier) ;
debut
  si e.tab[i].zero alors write('0')
  sinon
    write ('f(')
    Affiche_rec(e.tab[i].fg)
    write(',')
    Affiche_rec(e.tab[i].fd)
    write(')')
  fin si
fin

procedure Affiche(e : Texpression) ;
debut
  Affiche_rec(e,1)
fin
```

d) On veut écrire une fonction qui renvoie l'expression qui correspond à une chaîne. Pour çà :

(i) Ecrire une fonction zero : Texpression qui renvoie l'expression '0'.

Solution :

```
fonction zero :Texpression ;
debut
  e.longfl1 ;
  e.tab[1].zero fl true ;
  retourne e ;
fin ;
```

(ii) Ecrire une fonction f(e1,e2 : Texpression) : Texpression qui renvoie f(e1,e2).

Solution :

```
fonction f(e1,e2 : Texpression) : Texpression ;
debut
  e.long fl e1.long+e2.long+1 ;
  e.tab[1].zero fl FAUX
  e.tab[1].fg fl 2 ;
  e.tab[2].fd fl e1.long+2

  pour i de 1 à e1.long faire
    e.tab[i+1] fl e1.tab[i] ;
    si non (e.tab[i+1].zero) alors
      e.tab[i+1].fg fl e.tab[i+1].fg+1
      e.tab[i+1].fd fl e.tab[i+1].fd+1
    fin si
  fin pour

  pour i de 1 à e2.long faire
    e.tab[i+e1.long+1] fl e1.tab[i] ;
    si non (e.tab[i+e1.long+1].zero) alors
      e.tab[i+e1.long+1].fg fl e.tab[i+e1.long+1].fg+1+e1.long
      e.tab[i+e1.long+1].fd fl e.tab[i+e1.long+1].fd+1+e1.long
    fin si
  fin pour
  retourne e
fin
```

(iii) En déduire une fonction récursive qui renvoie l'expression qui correspond à une chaîne. On supposera l'expression bien formée.

Solution :

```
fonction Expr_rec (e : chaine ; d,f :entier) : boolean ;
debut
  si d=f alors retourne zero
  sinon
    c fl 0 ;
    p fl d+1 ;
    repeter
      si e[p]='(' alors c fl c+1
      sinon si e[p] = ')' alors c fl c-1
```

```
            fin si
      fin si
      p ← p+1
    jusqua (c=0 et e[p]=',') ou p>f
    retourne f(Expr_rec(e,d+1,p-1),Expr_rec(e,p+1,f-1));
  fin si
fin

fonction Expr(e :chaine) :Texpression ;
debut
  retourne Expr_rec(e,1,longueur(e)) ;
fin
```

d) En convenant que '0' = 0 et que f(n,m) = (n+1)*(m+1), écrire une fonction récursive qui évalue une expression.

Solution :

```
fonction Evalue_rec (e : Texpression ; i : entier) : entier ;
debut
  si e.tab[i].zero alors retourne 0
  sinon retourne (evalue_rec(e,e.tab[i].fg)+1)*
                 (evalue_rec(e,e.tab[i].fd)+1) ;
fin

fonction Evalue (e : Texpression) : entier ;
debut
  retourne Evalue_rec(e,1) ;
fin
```

e) Ecrire une fonction récursive qui met une expression sous forme post-fixe. Là encore, on supposera l'expression bien formée.

Exemple. la forme post-fixe de f(f(0,f(0,0)),0) est : 0 0 0 f f 0 f

Solution :

```
fonction Post_fixe_rec (e : Texpression ; i : entier) :
chaine ;
debut
  si e.tab[i].zero alors retourne '0'
  sinon retourne Post_fixe_rec(e,e.tab[i].fg)+
                 Post_fixe_rec(e,e.tab[i].fd)+'f'
  fin si
fin

et bien sur :

fonction Post_fixe(e : Texpression) : chaine
debut
  retourne post_fixe_rec(e,1) ;
fin
```

Et en TP : Ecrire les fonctions + si on a le temps, implémenter conversion expression ⇄ chaine préfixe, puis infixe.