Université Mohamed Khider-Biskra
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie
Département d'informatique

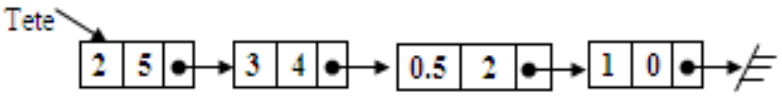
$2^{ème}$ année LMD — Algorithmique et Structures de données
07 Jan 2019 — 12:00-13:30, Amphis 1,2,3

Examen

Exercice 1 Listes linéaires chaînées (10 pts : 2 + 2 + 2 + 4)

On souhaite ajouter d'autres procédures et fonctions au deuxième TP sur les polynômes. On vous rappelle que les polynômes sont représentés par des listes linéaires chaînées, où chaque terme d'un polynôme est rangé dans un maillon de la liste contenant le degré du terme et le coefficient correspondant et que les termes sont stockés dans l'ordre décroissant des degrés.

Exemple : Le polynôme $2x^5 + 3x^4 + \frac{1}{2}x^2 + 1$ est représenté par la liste suivante :



On met aussi à votre disposition les procédures et fonctions suivantes développées dans le TP et que vous pouvez utiliser sans les écrire :

- **Fonction Somme2Poly(P_1,P_2 : pointeur(TMaillon)) : Pointeur(TMaillon) ;**
  Permettant de retourner la tête du polynôme représentant la somme des deux polynômes représentés par les deux listes de têtes P_1 et P_2.
- **Fonction Produit2Poly((P_1,P_2 : pointeur(TMaillon)) : Pointeur(TMaillon) ;**
  Permettant de retourner la tête du polynôme représentant le résultat du produit des deux polynômes représentés par les deux listes de têtes P_1 et P_2.
- **Procédure LibérerPoly(var P : pointeur(TMaillon)) ;**
  Permettant de libérer la liste de tête P représentant un polynôme.

On vous demande d'écrire les procédures et fonctions suivantes :

1. **Fonction ValeurPoly(P :pointeur(TMaillon), x :réel) :réel ;**
   Permettant de calculer la valeur de P pour une valeur x donnée, c-à-d $P(x)$.

2. **Fonction DivTermParTerm(T_1, T_2 :pointeur(TMaillon)) : pointeur(TMaillon) ;**
   Permettant de retourner la tête du polynôme représentant le resultat de division du terme T_1 par le terme T_2.
   Exemple :

$$T_1 = 6x^5 \;\Big|\; \begin{array}{c} T_2 = 2x^2 \\ \hline \text{Résultat}=3x^3 \end{array}$$

3. **Fonction Diff2Poly(P_1, P_2 :pointeur(TMaillon) : pointeur(TMaillon)) ;**
   Permettant de retourner la tete du polynôme représentant le resultat de soustraction du polynôme P_2 du polynôme P_1 c-à-d $P_1 - P_2$.

4. **Procédure Diviser2Poly(P_1, P_2 :pointeur(TMaillon), var Q, R : :pointeur(TMaillon)) ;**
   Permettant de caculer le résultat de division du polynôme P_1 par le polynôme P_2 et mettre le résultat de division dans Q et le reste dans R.

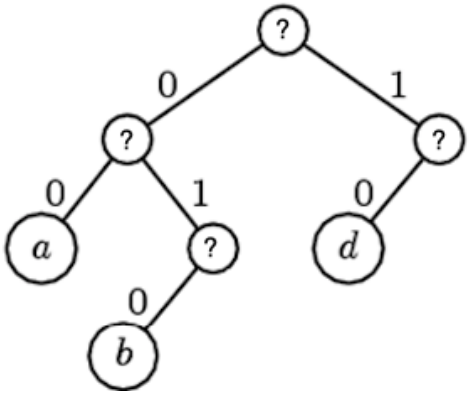
Exemple :

$P_1 = 6x^5 + 5x^3 + x + 2$		$P_2 = 2x^2 + 1$
-	$6x^5 + 3x^3$	$Q = 3x^3 + x$
=	$2x^3 + x + 2$	
-	$2x^3 + x$	
$R = 2$		

Exercice 2 Arbres binaires(10 pts : 4 + 2 + 4)

On souhaite dans cet exercice compléter aussi le dernier TP sur les arbres binaires appliqués au codage de Huffman par d'autres procédures et fonctions. On vous rappelle que l'arbre de Huffman est un arbre binaire représentant les codes binaires des caractères où les feuilles représentent les caractères, les branches droites représentent le code 1 et les branches gauches représentent le code 0.

Exemple :



On vous demande d'écrire les procédures et fonctions suivantes :

1. **Fonction CodeCar(**R **:Poniteur(TNoeud),** C **:caractères) :Chaine de caractère ;**
   Permettant de retourner le code binaire du caractère C dans l'arbre de huffman de racine R. Si le caractère C n'existe pas dans l'arbre, la fonction doit retourner '?'.

   Exemple :

   CodeCar(R,'b')=010

2. **Fonction CoderMessage(**R **:Pointeur(TNoeud),** $Message$ **:Chaine de caractère) : Chaine de caractères ;**
   Permettant de retourner le code binaire de la chaine de caractères $Message$ à partir de l'arbre de Huffman de racine R.

   Exemple :

   CoderMessage(R,'adkba')=0010 ?01000

3. **Procédure SupprimerCode(**R **:Pointeur(TNoeud),** $Code$ **:Chaine de caractères) ;**
   Permettant de supprimer le code binaire $Code$ de l'arbre de racine R. On vous conseille d'utiliser une pile.

NB : Pour les deux exercices, une procédure ou fonction écrite peut être utilisée dans les questions suivantes.

★★★ Bonne chance ★★★

Corrigé type

Exercice 1 LLCs

1. Fonction ValeurPoly

```
Fonction ValeurPoly( P : Pointeur(TMaillon), x : réel) : réel;
var S, Puiss : réel;
    i : entier ;
Début
    S ← 0 ;
    Tant que (P ≠ Nil) faire
        Puiss ← 1 ;
        Pour i de 1 à Degré(P) faire
            Puiss ← Puiss × x ;
        Fin Pour;
        S ← S + Coef(P) × Puiss ;
        P ← Suivant(P) ;
    Fin TQ;
    ValeurPoly ← S ;
Fin;
```

2. Fonction DivTermParTerm

```
Fonction DivTermParTerm( T1, T2 : Pointeur(TMaillon)) : Pointeur(TMaillon);
var Q : Pointeur(TMaillon) ;
Début
    Si (T1=Nil ou T2=Nil ou Degré(T1) < Degré(T2)) Alors
        Q ←Nil ;
    Sinon
        Allouer(Q) ;
        Aff_Adr(Q,Nil) ;
        Aff_Degré(Q,Degré(T1)-Degré(T2) ;
        Aff_Coef(Q,Coef(T1)/Coef(T2)) ;
    Fin Si;
    DivTermParTerm ← Q ;
Fin;
```

3. Fonction Diff2Poly

```
Fonction Diff2Poly( P1, P2 : Pointeur(TMaillon)) : Pointeur(TMaillon);
var Somme, Prod, Negatif : Pointeur(TMaillon) ;
Début
    Allouer(Negatif) ;
    Aff_Adr(Negatif,Nil) ;
    Aff_Degré(Negatif,0) ;
    Aff_Coef(Negatif,-1) ;
    Prod ← Produit2Poly(P2,Negatif) ;
    Somme ← Somme2Poly(P1,Prod) ;
    Libérer(Negatif) ;
    LibérerPoly(Prod) ;
    Diff2Poly ← Somme ;
Fin;
```

4. Procédure Diviser2Poly

```
Procédure Diviser2Poly( P1, P2 : Pointeur(TMaillon), var Q, R : Pointeur(TMaillon));
var Q1, Temp, Prod : Pointeur(TMaillon) ;
Début
    Q ← Nil ;
    Q1 ← Nil ;
    Prod ← Nil ;
    R ← P1 ;
    Tant que (Degré(R) ≥ Degré(P2)) faire
        Q1 ← DivTermParTerm(R, P2) ;
        Prod ← Produit2Poly(P2, Q1) ;
        Temp ← Q ;
        Q ← Somme2Poly(Q, Q1) ;
        LibérerPoly(Temp) ;
        Temp ← R ;
        R ←Diff2Poly(R, Prod) ;
        LibérerPoly(Temp) ;
        LibérerPoly(Prod) ;
        LibérerPoly(Q1) ;
    Fin TQ;
Fin;
```

Exercice 2 Arbres(10 pts : 4 + 2 + 4)

1. **Fonction CodeCar**

```
Fonction CodeCar( R : Poniteur(TNoeud), C : Caractères) : Chaîne de caractères;
var  S : Chaîne de caractères ;
Début
    Si (R=Nil) Alors
        CodeCar ←'?' ;
    Sinon
        Si (Valeur(R)=C) Alors
            CodeCar ← '' ;
        Sinon
            S ← CodeCar(FG(R),C) ;
            Si (S≠'?') Alors
                CodeCar ← '0'+ S ;
            Sinon
                S ← CodeCar(FD(R),C) ;
                Si (S ≠'?') Alors
                    CodeCar ← '1'+ S ;
                Sinon
                    CodeCar ← '?' ;
                Fin Si;
            Fin Si;
        Fin Si;
    Fin Si;
Fin;
```

2. **Fonction CoderMessage**

```
Fonction CoderMessage( R : Poniteur(TNoeud), Message : Chaîne de Caractères) :
Chaîne de caractères;
var  i : entier ;
     S : Chaîne de caractères ;
Début
    S ←'' ;
    Pour i de 1 à Longueur(Message) faire
        S ← S + CodeCar(Message[i]) ;
    Fin Pour;
    CoderMessage ← S ;
Fin;
```

3. **Procédure SupprimerCode**

```
Procédure SupprimerCode( R : Poniteur(TNoeud), Code : Chaîne de Caractères);
var  i : entier ;
     S : Chaîne de caractères ;
Début
    i ← 1 ;
    Tant que (i ≤Longueur(Code) et R ≠ Nil)) faire
        Empiler(R) ;
        Si (Code[i]='1') Alors
            R ← FD(R) ;
        Sinon
            R ← FG(R) ;
        Fin Si;
        i ← i + 1 ;
    Fin TQ;
    Si (R=Nil ou Valeur(R)='?') Alors
        Ecrire('Ce code n'existe pas') ;
    Sinon
        Libérer(R) ;
        Continue ← Vrai ;
        i ← Longuer(Code) ;
        Tant que (PileVide=Faux et Continue=vrai ) faire
            Déplier(R) ;
            Si (Code[i]='1' et FG(R) = Nil ou Code[i]='0' et FD(R) = Nil) Alors
                Libérer(R) ;
            Sinon
                Si (Code[i]='1) Alors
                    Aff_FD(R,Nil) ;
                Sinon
                    Aff_FG(R,Nil) ;
                Fin Si;
                Continue ← Faux ;
            Fin Si;
            i ← i − 1 ;
        Fin TQ;
    Fin Si;
Fin;
```