Département STPI 16 juin 2014

I3 - Algorithmique

Durée : *3h*
Documents autorisés : **AUCUN** (calculatrice comprise) [1]

Remarques :
- Veuillez lire attentivement les questions avant de répondre.
- Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
- Rendez une copie propre.
- N'utilisez pas de crayon à papier.

1 Question de TD (6 points)

Pour rappel, le type `ListeChaineeDEntiers` est défini de la façon suivante :

Type ListeChaineeDEntiers **=** ^ NoeudDEntier
Type NoeudDEntier **= Structure**
entier : **Entier**
listeSuivante : ListeChaineeDEntiers
finstructure

Et nous l'utilisons à l'aide des fonctions et procédures suivantes :
- **fonction** listeVide **() :** ListeChaineeDEntiers
- **fonction** estVide **(**uneListe : ListeChaineeDEntiers**) : Booleen**
- **procédure** ajouter **(E/S** uneListe : ListeChaineeDEntiers,**E** element : Entier**)**
- **fonction** obtenirEntier **(**uneListe : ListeChaineeDEntiers**) :** Entier
  |précondition(s) non(estVide(uneListe))
- **fonction** obtenirListeSuivante **(**uneListe : ListeChaineeDEntiers**) :** ListeChaineeDEntiers
  |précondition(s) non(estVide(uneListe))
- **procédure** fixerListeSuivante **(E/S** uneListe : ListeChaineeDEntiers,**E** nelleSuite : ListeChaineeDEntiers**)**
  |précondition(s) non(estVide(uneListe))
- **procédure** supprimerTete **(E/S** l :ListeChaineeDEntiers**)**
  |précondition(s) non estVide(l)
- **procédure** supprimer **(E/S** uneListe : ListeChaineeDEntiers**)**

1. Proposez une fonction itérative, `longueur`, qui permet de connaître le nombre d'éléments d'une liste chaînée. Quelles sont les complexités (pire et meilleur des cas) de votre algorithme ? Justifiez.
2. Proposez la même fonction mais de manière récursive. Votre fonction est-elle récursive terminale ou non-terminale ? Justifiez.
3. Proposez une procédure récursive qui permet d'insérer dans une liste chaînée l un entier e à une position p (avec $1 \leq p \leq longueur(l) + 1$).

Solution proposée :
Voir TD

1. Sauf les dictionnaires pour les étudiants non francophones

2 Distance de Hamming (6 points)

2.1 Distance de Hamming entre deux mots

La distance de Hamming entre deux mots (chaîne de caractères) de même longueur est égale au nombre de lettres, à la même position, qui diffèrent. Par exemple la distance de Hamming entre "rose" et "ruse" est de 1, alors que la distance de Hamming entre "110110" et "000101" est de 4.

Proposez le corps de la fonction itérative suivante qui permet de calculer la distance de Hamming de deux mots, non vides, que l'on sait de même longueur :

— **fonction** distanceHammingDeuxMots (mot1,mot2 : **Chaine de caracteres) : Naturel**

⌊**précondition(s)** $mot1 \neq$ "" et $longueur(mot1) = longueur(mot2)$

Solution proposée :

fonction distanceHammingDeuxMots (mot1,mot2 : **Chaine de caracteres) : Naturel**

Déclaration i,resultat : **Naturel**

debut
resultat ← 0
pour i ←1 **à** longueur(mot1) **faire**
si iemeCaractere(mot1,i)≠iemeCaractere(mot2,i) **alors**
resultat ← resultat+1
finsi
finpour
retourner resultat
fin

2.2 Distance de Hamming d'un langage

Un langage est un ensemble de mots. La distance de Hamming d'un langage est égale au minimum des distances de Hamming entre deux mots de ce langage différents deux à deux.

Proposez le corps de la fonction itérative suivante qui permet de calculer la distance de Hamming d'un langage d'au moins 2 mots qui possède uniquement des mots de même longueur et tous différents deux à deux :

— **fonction** distanceHammingLangage (leLangage : **Tableau[**1..MAX**] de Chaine de caracteres**, nbMots : **NaturelNonNul) : Naturel**

⌊**précondition(s)** $nbMots > 1$ et $nbMots \leq MAX$ et
$\forall i, i > 0, i \leq nbMots, longueur(leLangage[i]) = longueur(leLangage[1])$ et
$\forall i, \forall j, i > 0, i \leq nbMots, j > 0, j \leq nbMots, i \neq j, leLangage[i] \neq leLangage[j]$

Solution proposée :

fonction distanceHammingLangage (leLangage : **Tableau[**1..MAX**] de** Chaîne, nbMots : **NaturelNonNul) : Naturel**

⌊**précondition(s)** $nbMots > 1$ et $nbMots \leq MAX$ et
$\forall i, i > 0, i \leq nbMots, longueur(leLangage[i]) = longueur(leLangage[1])$ et
$\forall i, \forall j, i > 0, i \leq nbMots, j > 0, j \leq nbMots, i \neq j, leLangage[i] \neq leLangage[j]$

Déclaration i,j,min,temp : **Naturel**

debut
min ← distanceHammingDeuxMots(leLangage[1],leLangage[2])
pour i ←1 **à** nbMots-1 **faire**
pour j ←i+1 **à** nbMots **faire**
temp ← distanceHammingDeuxMots(leLangage[i],leLangage[j])
si temp<min **alors**
min ← temp

```
            finsi
        finpour
    finpour
    retourner min
fin
```

3 Fractal (8 points)

3.1 Point2D (1 point)

Soit le type `Point2D` permettant de représenter des points ou vecteurs mathématiques dans un espace de $\mathbb{R}^2$.

Proposez les signatures des procédures ou fonctions permettant :

- de créer un `Point2D` à partir d'une abscisse et d'une ordonnée ;
- d'obtenir l'abscisse d'un `Point2D` ;
- d'obtenir l'ordonnée d'un `Point2D` ;
- de multiplier un `Point2D` par un réel ;
- de calculer la rotation d'un `Point2D` ;
- de calculer la translation d'un `Point2D` ;
- de calculer l'homothétie d'un `Point2D`.

Solution proposée :

- **fonction** point2D (x,y : **Reel**) **:** Point2D
- **fonction** abscisse (p : Point2D) **: Reel**
- **fonction** ordonnee (p : Point2D) **: Reel**
- **fonction** multiplication (p : Point2D, a : **Reel**) **:** Point2D
- **fonction** rotation (c : Point2D, alpha : **Reel**, p : Point2D) **:** Point2D
- **fonction** translation (v : Point2D, p : Point2D) **:** Point2D
- **fonction** homothétie (c : Point2D, k : **Reel**, p : Point2D) **:** Point2D

3.2 Courbe de Koch

FIGURE 1 – Courbe de Koch de niveau 4

La courbe de Koch est une fractal (un dessin récursif) dont le détail varie en fonction d'un niveau donné n. La figure 1 (issue de `http://www.mathcurve.com/fractals/koch/koch.shtml`) présente la courbe de Koch de niveau 4. En fait, dessiner la courbe de Koch de niveau n entre deux points A et E revient à :

- dessiner le segment de droite $[A, E]$ si $n = 0$
- dessiner 4 courbes de Koch de niveau $n - 1$ entre les points A,B,C,D et E (la première entre A et B, la deuxième entre B et C, la troisième entre C et D, et la dernière entre D et E) telles que (Cf. figure 2) :
  - $\overrightarrow{AB} = \frac{1}{3}\overrightarrow{AE}$,
  - $\overrightarrow{ED} = \frac{1}{3}\overrightarrow{EA}$,
  - les points B,C et D forment un triangle équilatéral.

La figure 3 présente quelques courbes de Koch.

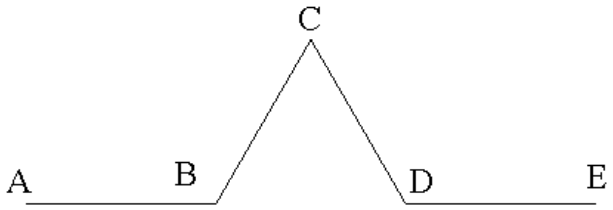

FIGURE 2 – Position des points B,C et D

(a) $n = 1$ (b) $n = 2$ (c) $n = 3$

FIGURE 3 – Quelques courbes de Koch

Types

On suppose posséder le type `Graphique`, le type `Couleur` et la procédure suivante qui permet de dessiner un segment de droite de couleur c entre deux points $pt1$ et $pt2$ sur un graphique e :

— **procédure** droite **(E/S** e : Graphique, **E** pt1, pt2 : Point2D, c : Couleur)

3.2.1 Analyse (3 points)

1. Quelles sont les opérations mathématiques sur les `Point2D` nécessaires au dessin d'une courbe de Koch ?
2. Donnez l'analyse descendante de l'opération `dessinerCourbeKoch` qui permet de dessiner une courbe de Koch d'une certaine couleur, de niveau n entre deux points sur un graphique.

Solution proposée :

1. Nous avons besoin des fonctions de rotation et d'homothétie
2. L'analyse descendante est :
   - dessinerCourbeKoch : Graphique × `Point2D` × `Point2D` × **Naturel**× Couleur → Graphique (récursif)
     - calculerPtB : `Point2D` × `Point2D` → `Point2D`
       - homothetie : `Point2D` × **Reel** → `Point2D`
     - calculerPtD : `Point2D` × `Point2D` → `Point2D`
       - homothetie : `Point2D` × **Reel** → `Point2D`
     - calculerPtC : `Point2D` × `Point2D` → `Point2D`
       - rotation : `Point2D` × **Reel** × `Point2D` → `Point2D`
     - droite : Graphique × `Point2D` × `Point2D` × Couleur → Graphique

3.2.2 Conception préliminaire (1 point))

Donnez les signatures des fonctions et procédures de votre analyse descendante.

Solution proposée :

procédure dessinerCourbeKoch **(E/S** e : Graphique, **E** pta,pte : Point2D, n : **Naturel**, c : Couleur)

⌊**précondition(s)** pta ≠ pte

fonction calculerPtB **(E** pta, pte : Point2D**) :** Point2D

fonction calculerPtD (**E** pta, pte : Point2D) **:** Point2D
fonction calculerPtC (**E** ptb, ptd : Point2D) **:** Point2D

3.2.3 Conception détaillée (3 points)

Donnez le corps des fonctions ou procédures, exceptées celles identifiées dans la partie 3.1 et la procédure `droite`.

Solution proposée :

procédure dessinerCourbeKoch (**E/S** e : Graphique, **E** pta, pte : Point2D, n : **Naturel**, c : Couleur)

⌊**précondition(s)** pta $\neq$ pte

Déclaration ptb,ptc,ptd : Point2D

debut
  si n=0 **alors**
    dessinerDroite(e,pta,pte,c)
  sinon
    ptb $\leftarrow$ calculerPtB(pta, pte)
    ptd $\leftarrow$ calculerPtD(pta, pte)
    ptc $\leftarrow$ calculerPtC(ptb, ptd)
    dessinerCourbeKoch(e, pta, ptb, n-1, c)
    dessinerCourbeKoch(e, ptb, ptc, n-1, c)
    dessinerCourbeKoch(e, ptc, ptd, n-1, c)
    dessinerCourbeKoch(e, ptd, pte, n-1, c)
  finsi
fin
fonction calculerPtB (**E** pta, pte : Point2D) **:** Point2D
debut
  retourner homothetie(pta,1/3,pte)
fin
fonction calculerPtD (**E** pta, pte : Point2D) **:** Point2D
debut
  retourner homothetie(pta,2/3,pte)
fin
fonction calculerPtC (**E** ptb, ptd : Point2D) **:** Point2D
debut
  retourner rotation(ptb,60,ptd)
fin