DS2 - Algorithmes et Structures de Données
Jeudi 8 Janvier 2026

Durée 3h – Documents non autorisés

Exercice 1 : valeur décimale d'un nombre écrit en chiffres romains (6 pts)

Les conventions d'écriture des nombres en chiffres romains permettent de représenter les entiers de 1 à 3999.
Les chiffres sont les suivants : I = 1 ; V = 5 ; X = 10 ; L = 50 ; C = 100 ; D = 500 ; M = 1000.
Le zéro n'existe pas. La règle de base est de faire la somme des chiffres.
Exemples :
III = 1+1+1 = 3 ; VI = 5+1= 6 ; ; XIII = 10+1+1+1 = 13 ;
MMXXVI = 1000+1000+10+10+5+1 =2026
Cependant, on n'écrit jamais plus de 3 chiffres identiques consécutifs. On a donc recours aux 6 écritures particulières suivantes :
IV = -1+5 = 4 ; IX = -1+10 = 9 ; XL = -10+50 = 40 ; XC = -10+100 = 90 ;
CD = -100+500 = 400 ; CM = -100+1000 = 900
Autres exemples :
LXXIII = 50+10+10+1+1+1 = 73 ; CDXLIII = -100+500-10+50+1+1+1 = 443 ;
DCCIII = 500+100+100+1+1+1 = 703 ; CMIV = -10+1000-1+5 = 904 ;
MMMCMXCIX = 1000+1000+1000-100+1000-10+100-1+10 = 3999
Un nombre écrit en chiffres romains est représenté par une chaîne de caractères de longueur maximale égale à 15.
On souhaite écrire une fonction qui, étant donnée une chaîne représentant un nombre écrit (correctement) en chiffres romains, le convertit en un entier (valeur décimale). On dispose d'un tableau contenant chaque chiffre romain avec sa conversion dans le système décimal :

```
Type  valeur = Enregistrement
                   rom : car
                   dec : entier
               FinEnregistrement
      tab-val = tableau [1..7] de valeur
```

1.1. Donner le principe itératif de la méthode utilisée : discuter notamment la valeur (positive ou négative) du caractère courant en fonction du caractère suivant.
1.2. Ecrire une fonction `donneval` qui renvoie la valeur décimale d'un **chiffre** romain :

```
Fonction donneval(cr : car, t : tab-val) : entier
```

1.3. Ecrire une fonction `traduit` qui renvoie la conversion d'un **nombre** écrit en chiffres romains dans le système décimal et qui utilise `donneval` :

```
Fonction traduit(nb : chaîne, t : tab-val) : entier
```

Remarque : on n'utilisera pas de test sur les valeurs explicites des chiffres romains ; on cherchera plutôt à utiliser la logique de la représentation.

Exercice 2 : dessin récursif d'un arbre binaire (4 pts)

On souhaite dessiner récursivement un arbre binaire `A` dont la racine est de coordonnées `x` et `y` (donnés en nombre de pixels). On dispose des outils suivants :
`Procédure va-en(E x,y : entier)` qui permet de déplacer le crayon au point de coordonnées `(x,y)`.
`Procédure trace-ligne(E x,y : entier)` qui permet de tracer un trait partant du point courant du crayon jusqu'au point de coordonnées `(x,y)`.

Pour ce dessin récursif, on suppose qu'à chaque niveau, la distance `d` (horizontale entre chaque noeud) est divisée par 2 et la hauteur `h` (verticale entre chaque niveau) est multipliée par 2/3.

2.1. Expliquer en français la méthode **récursive** utilisée.

2.2. Ecrire en pseudo-langage une procédure **récursive** qui réalise ce dessin.

```
Procédure dessine-ArbreBinaire (E A : ArbreBinaire ; x,y,d,h : entier)
```

Utiliser les opérations du TAD `ArbreBinaire`.

Exercice 3 : structure de données dynamique (10 pts)

On souhaite représenter une discographie en utilisant la structure de données dynamique décrite par la figure ci-dessous : une liste de catégories de musique (pointée par d) et pour chacune d'elles, la liste des artistes présents avec le nombre d'albums correspondants.

3.1. Ecrire en pseudo-langage les types pour représenter cette structure de données.

3.2. Ecrire en pseudo-langage une procédure d'ajout d'une catégorie de musique sans artiste (la liste est triée dans l'ordre alphabétique). Faire des dessins selon les cas particuliers.

3.3. Ecrire en pseudo-langage une procédure d'ajout d'un artiste et du nombre d'album (liste triée dans l'ordre alphabétique) dans une catégorie musicale donnée (existante). Faire des dessins selon les cas particuliers.

3.4. Ecrire en pseudo-langage une fonction qui retourne le nombre total d'albums.

3.5. Ecrire en pseudo-langage une fonction qui retourne la catégorie musicale d'un artiste donné (catégorie unique).

3.6. Ecrire en pseudo-langage une fonction qui retourne la catégorie ayant le plus d'artistes (de manière arbitraire s'il y en a plusieurs).