Exercices des chapitres 7 et 8

Sommaire

01-**-ADN

Les bases d'un brin d'ADN sont codées par l'un des quatre caractères 'A', 'C', 'G', ou 'T'. On peut représenter un brin d'ADN par un tableau contenant une séquence de bases, apparaissant dans un ordre quelconque. On peut alors calculer le brin complémentaire sachant que, dans deux brins complémentaires, les bases 'A' et 'T' se correspondent ainsi que les bases 'C' et 'G'
Ecrire un algorithme qui value un tableau avec les bases d'un brin d'ADN données par l'utilisateur (saisies contrôlées), construit, puis affiche le tableau représentant le brin complémentaire. La fin de saisie sera marquée par l'entrée d'un caractère spécial, par exemple le caractère 'X'.

Exemple :

Brin initial :	A	T	G	A	T	C	C	G	.....
	1	2	3	4	5	6	7	8	.....
Brin complémentaire :	T	A	C	T	A	G	G	C	.....

02-**-Classes d'âge

On veut étudier la répartition d'un échantillon de population dans les neuf classes d'âge définies ci-dessous.

age dans	classe
[0, 10[	0
[10, 20[	1
[20, 30[	2
[30, 40[	3
[40, 50[	4
[50, 60[	5
[60, 70[	6
[70, 80[	7
80 et plus	8

Ecrire un algorithme qui :
- saisit une liste d'entiers représentant les âges sans les mémoriser et construit, au fur et à mesure de la saisie, un tableau contenant l'effectif de chaque classe. Une valeur négative terminera la saisie.
- affiche une représentation de ce tableau en affichant une ligne par classe, et, sur chaque ligne un nombre d'étoiles égal à l'effectif de la classe.

Exemple avec la série :
```
68 92 60 24 71 14 52 12 16 40 80 18 20
40 10 6 48 43 25
```
on obtiendra la représentation ci-contre :

```
classe 0 : *
classe 1 : *****
classe 2 : ***
classe 3 :
classe 4 : ****
classe 5 : *
classe 6 : **
classe 7 : *
classe 8 : **
```

03-**-Pluviométrie 1

On dispose d'un ensemble de relevés pluviométriques réalisés sur un territoire donné.

Les relevés ont lieu en différents points de ce territoire. Un relevé est constitué de 3 données inscrites sur une fiche :

le lieu	le numéro du mois	la hauteur de précipitation en ce lieu et pour le mois donné (en mm)

Exemple de relevé :

DIJON	10	150

L'objectif de cet exercice est :

- de faire effectuer la saisie des relevés pour les stocker dans trois vecteurs nommés *Lieux*, *Mois*, *Hauteurs*. Cette saisie se fera par une procédure nommée *SaisirReleves* qui remplira les trois vecteurs et calculera le nombre de relevés *Nbsaisis*.
- d'exploiter ces données pour remplir un vecteur nommé *Somme* qui contiendra pour chaque mois la somme des hauteurs de précipitation enregistrées dans tous les lieux. Cette exploitation se fera à l'aide d'une procédure nommée *ExploiterReleves*.
- d'afficher le contenu du vecteur *Somme* avec une procédure *AfficherSomme*.

L'action *SaisirReleves* demande à l'utilisateur une suite de relevés (lieu, n° du mois, hauteur) et range ces relevés dans 3 vecteurs *Lieux*, *Mois*, *Hauteurs* de sorte que ∀i, *Lieux[i]*, *Mois[i]*, *Hauteurs[i]* relèvent de la même fiche.
Le nombre de relevés n'est pas connu a priori par l'utilisateur, celui-ci devra donc arrêter la saisie par une chaîne de caractères fictive, par exemple 'Z'. L'action devra aussi, en cours de saisie, compter le nombre de relevés pour valuer la variable *Nbsaisis*..
De plus la saisie d'un numéro de mois d'une part et celle de la hauteur d'autre part devront faire l'objet d'une procédure de saisie avec contrôle de réponse (entre 10 et 1000 pour la hauteur).
Exemple :

	Lieux
1	DIJON
2	NANTES
3	NANTES
4	DIJON
5	LYON
6	VIERZON

	Mois
1	10
2	5
3	10
4	5
5	1
6	8

	Hauteurs
1	150
2	80
3	200
4	50
5	100
6	80

	Somme
1	100
2	0
3	0
4	0
5	130
6	0
7	0
8	80
9	0
10	350
11	0
12	0

L'action *ExploiterReleves* prend les données nécessaires dans les vecteurs *Mois*, *Hauteurs* et dans la variable *Nbsaisis* et remplit le vecteur *Somme* comme indiqué ci-dessus.

04-**-Pluviométrie 2

Refaites l'exercice précédent (pluviométrie 1) en utilisant la notion de structure.

CORRIGES

01-**-ADN

```
Algorithme ADN
Constantes
  TAILLE_MAX = 100  /* taille maximale d'un brin d'ADN */

Types
  t_ADN = tableau[1..TAILLE_MAX] de caractere

Variables
  brinInitial     : t_ADN   /* le brin initial d'ADN */
  brinComplement  : t_ADN   /* le complémentaire du brin initial */
  nbBases         : entier  /* nombre de bases dans les brins */

Début
  remplir(brinInitial, nbBases)
  construire(brinInitial, nbBases, brinComplement)
  afficher(brinInitial, nbBases)
  afficher(brinComplement, nbBases)
Fin

/*
* Value un brin d'ADN
*/
procédure remplir(  sortie brin : t_ADN,
                    sortie nb : entier)
Variables locales
  base  : caractère
Début
  nb ← 0
  obtenirBase(base)     /* on arrête avec 'X' */
  tantque base != 'X' et nb < TAILLE_MAX
    nb ← nb + 1
    brin[nb] ← base
    obtenirBase(base)
  fintantque
Fin

/*
* Saisit un caractère et le redemande jusqu'à ce qu'il soit un 'X'
* ou corresponde à une base
*/
procédure obtenirBase (sortie base : caractère)
Début
  écrire('Base ou X pour terminer : ')
  lire(base)
  tantque base != 'A' et base != 'T' et base != 'G' et base != 'C'
      et base != 'X'
    écrire ('erreur : répondre par A, T, G, C ou Z pour terminer')
    lire(base)
  fintantque
Fin
```

```
/* NB : on aurait pu utiliser un tableau pour stocker les bases et
le caractères d'arrêt X */

/*
* Construit le brin complémentaire de brin qui possède nb bases
*/
procédure construire (  entrée brin : t_ADN,
                        entrée nb : entier,
                        sortie compl : t_ADN)
Variables locales
  i : entier
Début
  pour i de 1 à nb
    selon brin[i] dans
      'A' : compl[i] ← 'T'
      'T' : compl[i] ← 'A'
      'C' : compl[i] ← 'G'
      'G' : compl[i] ← 'C'
    finselon
  finpour
Fin

/*
* Affiche un tableau de caractères = un brin d'ADN
*/
procédure afficher (  entrée  t : t_ADN,
                      entrée nb : entier)
Variables locales
  i : entier
Début
  pour i de 1 à nb
    écrire(t[i], " , ")
  finpour
  écrire(CRLF)    /* passage à la ligne */
Fin
```

02-**-Classes d'âge

```
/*
* on suppose que les classes d'âge sont des intervalles qui vont
* de 10 en 10. Il y a 9 classes d'âge
*/
Algorithme ClassesDAge

Constantes
  NB_CLASSES  = 9
  TAILLE_CLASSES  = 10
Types
  t_classes = tableau[1..NB_CLASSES] d'entier
Variables
  classes : t_classes
Début
  calculer(classes)
```

```
  afficher(classes)
Fin

/*
* saisit un certain nombre d'âges entiers jusqu'à une saisie
* négative et calcule les effectifs des classes d'âge au fur et à
* mesure
*/
procédure calculer(sortie t : t_classes)
Variables locales
  i : entier
  age : entier
Début
  lire(age)
  tantque age >= 0
    classe ← classement(age)
    t[classe] ← t[classe] + 1
  fintantque
Fin

/*
* retourne l'indice de la classe d'âge en fonction de l'âge passé
* en paramètre
*/
fonction classement(age : entier) : entier
Variables locales
  i : entier
  clas : entier
Début
  clas ← age div 10
  si clas = 0 alors
    clas ← 1
  sinon
    si clas > 9 alors
      clas ← 9
    finsi
  finsi
  retourner clas
Fin

/*
* retourne une chaine composée de n étoiles
*/
fonction etoiles(n : entier) : chaine
Variables locales
  ch : chaine
  i : entier
Début
  pour i de 1 à n
    ch ← ch + "*"
  finpour
  retourner ch
Fin
```

```
/*
* affiche les classes d'âge
*/
procédure afficher(entrée t : t_classes)
Variables locales
  i : entier
Début
  pour i de 1 à 9
    écrire("classe ", i-1, " : ", etoiles(classes[i]))
  finpour
Fin
```

03-**-Pluviométrie 1

```
Algorithme Pluviométrie1
Constantes
  TAILLE_MAX  = 500 /* choix arbitraire */
Types
  t_tabLieu   = tableau[1.. TAILLE_MAX] de chaînes de caractères
  t_tabEntier = tableau[1.. TAILLE_MAX] d'entier
  t_tabMois   = tableau[1..12] d'entier
Variables
  lieux     : t_tabLieu   /* les lieux des relevés */
  mois      : t_tabEntier /* les mois des relevés */
  hauteurs  : t_tabEntier /* les hauteurs des relevés */
  nbSaisis  : entier      /* nombre de relevés */
  somme     : t_tabMois   /* somme des hauteurs par mois */

Début
  saisirReleves(lieux, mois, hauteurs, nbSaisis)
  exploiterReleves(mois, hauteurs, nbSaisis, somme)
  afficherSomme(somme)
Fin

/*
* saisit une séquence de relevés et value les vecteurs. Le dernier
* relevé saisi n'est pas mémorisé. La procédure obtenirEntre
* saisit un nb compris entre deux bornes
*/
procédure saisirReleves(  sortie lieu : t_tabLieu,
                          sortie mois : t_tabEntier,
                          sortie hauteurs : t_tabEntier,
                          sortie nbSaisis : entier)
Variables locales
  lieuSaisi : chaine
Début
  nbSaisis ← 0
  lire(lieuSaisi)                  /* acquérir premier élément */
  tantque lieuSaisi != 'Z' et nbSaisis < TAILLE_MAX
    nbSaisis ← nbSaisis + 1        /* traiter élément courant */
    lieu[nbSaisis] ← lieuSaisi
    obtenirEntre(mois[nbSaisis], 1, 12)
```

```
    obtenirEntre(hauteur[nbSaisis], 10, 1000)
    lire(lieuSaisi)        /* acquérir élément suivant */
  fintantque
Fin

/*
* renvoie un entier obtenu par saisie, compris entre inf et sup
*/
procédure obtenirEntre( sortie x : entier,
                        entrée inf : entier,
                        entrée sup : entier)
Début
  répéter
    lire(x)
  jusqu'à (x >= inf) et (x <= sup)
Fin

/*
* parcourt les tableaux mois et hauteur et value le vecteur somme
* en cumulant les hauteurs par mois
*/
procédure exploiterReleves(entrée mois : t_tabEntier,
                           entrée hauteurs : t_tabEntier,
                           entrée nbSaisis : entier,
                           sortie somme : t_tabMois)
Variables
  i : entier
Début
  /* initialisation du vecteur somme */
  pour i de 1 à 12
    somme[i] ← 0
  finpour
  /* parcours de tous les relevés */
  pour i de 1 à nbSaisis
    somme[mois[i]] ← somme[mois[i]] + hauteur[i]
  finpour
Fin

/*
* affiche le contenu du vecteur somme
*/
procédure afficherSomme(entrée somme : t_tabMois)
Variables
  i : entier
Début
  pour i de 1 à 12
    écrire("mois n° ", i, somme[i])
  finpour
Fin
```

04-**-Pluviométrie 2

```
Algorithme Pluviométrie2
Constantes
  TAILLE_MAX  = 500 /* choix arbitraire */
Types
  t_releve = enrg   /* un relevé */
    lieu : chaine
    mois : entier
    hauteur : entier
  finenrg
  t_tabReleves  = tableau[1.. TAILLE_MAX] de t_releve
  t_tabMois     = tableau[1..12] d'entier
Variables
releves   : t_ tabReleves/* les relevés */
nbSaisis  : entier       /* nombre de relevés */
somme     : t_tabMois    /* somme des hauteurs par mois */

Début
  saisirReleves(releves, nbSaisis)
  exploiterReleves(releves, nbSaisis, somme)
  afficherSomme(somme)
Fin

/*
* saisit une séquence de relevés.
*/
procédure saisirReleves(  sortie releves : t_tabReleves,
                          sortie nbSaisis : entier)
Variables locales
  lieuSaisi : chaine
Début
  nbSaisis ← 0
  lire(lieuSaisi)                  /* acquérir premier élément */
  tantque lieuSaisi != 'Z' et nbSaisis < TAILLE_MAX
    nbSaisis ← nbSaisis + 1        /* traiter élément courant */
    releves[nbSaisis].lieu ← lieuSaisi
    obtenirEntre(releves[nbSaisis].mois, 1, 12)
    obtenirEntre(releves[nbSaisis].hauteur, 10, 1000)
    lire(lieuSaisi)          /* acquérir élément suivant */
  fintantque
Fin

/*
* renvoie un entier obtenu par saisie, compris entre inf et sup
*/
procédure obtenirEntre( sortie x : entier,
                        entrée inf : entier,
                        entrée sup : entier)
Début
  répéter
    lire(x)
  jusqu'à (x >= inf) et (x <= sup)
```

```
Fin

/*
* parcourt les tableaux mois et hauteur et value le vecteur somme
* en cumulant les hauteurs par mois
*/
procédure exploiterReleves(entrée releves : t_tabReleves,
                           entrée nbSaisis : entier,
                           sortie somme : t_tabMois)
Variables locales
  i : entier
Début
  /* initialisation du vecteur somme */
  pour i de 1 à 12
    somme[i] ← 0
  finpour
  /* parcours de tous les relevés */
  pour i de 1 à nbSaisis
    somme[releves[i].mois] ← somme[releves[i].mois] +
        releves[i].hauteur
  finpour
Fin

/*
* affiche le contenu du vecteur somme
*/
procédure afficherSomme(entrée somme : t_tabMois)
Variables locales
  i : entier
Début
  pour i de 1 à 12
    écrire("mois n° ", i, somme[i])
  finpour
Fin
```