Sciences de l'informatique - ALGORITHMIQUE ET PROGRAMMATION

Correction Epreuve principale - Session de juin 2011

Première partie : (10 points)

Exercice n°1 : (4 points)

- ***Trace de la fonction Inconnue pour le couple de valeurs (2,13):***

b	*n*	*ch*	*r*	*s*	*inconnue*
2	*13*	""	*1*		
	6	"1"	*0*	"1"	
	3	"01"	*1*	"0"	
	1	"101"	*1*	"1"	
	0	"1101"		"1"	**"1101"**

Résultat de la fonction

- ***Trace de la fonction Inconnue pour le couple de valeurs (16,163):***

b	*n*	*ch*	*r*	*s*	*inconnue*
16	*163*	""	*3*		
	10	"3"	*10*	"3"	
	0	"A3"		"A"	**"A3"**

Résultat de la fonction

- Le type de la fonction **inconnue** est : ***Chaîne de*** caractères
  Tableau de déclaration d'objets :

Objet	Type/Nature
ch, s	Chaîne de caractères
r	Entier

- La fonction retourne le résultat de la **conversion** d'un entier **n** dans la base **b**.

Exercice n°2 : (3 points)

Dans cet exercice il est demandé **l'algorithme** d'une **fonction récursive**.

1) D'après la définition de la suite, on déduit qu'elle est définie selon 3 cas :

- Cas 1, **n=0**, le résultat (U) est égal à **0**, puisque la valeur du résultat est fixée, il s'agit d'un **cas d'arrêt** du traitement récursif.
- Cas 2, n=1, le résultat (U) est égal à **-9**, puisque la valeur du résultat est fixée, il s'agit d'un **cas d'arrêt** du traitement récursif

- Cas 3, n≥ **2**, le résultat est défini en fonction de 2 autres termes. Puisque la valeur du résultat n'est pas fixée, il ne s'agit pas d'un cas d'arrêt du traitement. De plus, comme le résultat est défini en fonction de **n** termes précédents, on en déduit qu'il s'agit de la définition de l'étape **d'avancement dans le traitement récursif**

On obtient alors l'algorithme suivant pour **la fonction récursive U** :

0/ DEF FN U (n : Entier) : Entier Long
1/ **Si** (n = 0) **Alors** U ← 0
Sinon Si (n = 1) **Alors** U ← -9
Sinon U ← 6 * **FN** U(n-1) – 9 * **FN** U(n-2)
FinSi
2/ **FIN** U

2) L'ordre de récurrence de la fonction est égal à **2** car le terme **Un** est défini à partir des termes **Un-1** et **Un-2**

Solutions équivalentes possibles :

- car le terme **Un** est défini à partir des deux termes précédents.
- car le terme **Un** est défini à partir de deux termes.

Exercice n°3 : (3 points)

Analyse de la fonction Div_huit

```
DEF FN Div_huit (Ch : chaîne) : Booléen                          (0.25 pt)
5/ Résultat = Div_huit ← (Cas1 OU Cas2)                          (0.25 pt)
4/ (Cas1, Cas2) = [Cas1 ← Faux, Cas2 ← Faux]
        Si (c MOD 2 = 0) ET ((d*10 + u) MOD 8 = 0) Alors         (0.75 pt)
            Cas1 ← Vrai
        Sinon
          Si (c MOD 2 = 1) ET ((d*10 + u - 4) MOD 8 = 0) Alors
            Cas2 ← Vrai
          FinSi                                                  (0.75 pt)
        FinSi
1/ c = Val (ch[Long(ch)-2],c,e)
2/ d = Val(ch[Long(ch)-1],d,e)                                   (3*0.25 = 0.75
3/ u = Val(ch[Long(ch)],u,e)                                      pt)
6/ FIN Div_huit
```

Explications :

- ***Séquences 1, 2 et 3 :*** Puisque la valeur numérique à traiter contient un nombre de chiffres compris entre 20 et 200, il est impossible d'utiliser :

1. Un type numérique prédéfini
2. Les opérations DIV et MOD pour extraire le chiffre des unités **(u)**, des dizaines **(d)** et des centaines **(c)**.

Il faudra donc :

1. Pour contenir les chiffres qui composent le nombre, utiliser le type chaîne de caractères et la fonction de conversion entre les chaînes et les entiers **(Val)**.
2. Pour obtenir :

a. le chiffre des unités (**u**), extraire le dernier caractère de **ch** (noté, **ch[longueur(ch)]**)

b. le chiffre des dizaines (**d**), extraire le dernier caractère de ch (noté, **ch[longueur(ch)] - 1**)

c. le chiffre des centaines (**c**), extraire le dernier caractère de ch (noté, **ch[longueur(ch) - 2]**)

- ***Séquence 4 :*** Traduction des 2 cas de divisibilité :

1er cas : Le chiffre des centaines (c) est **pair** (c MOD 2 = 0) et le nombre formé par les 2 derniers chiffres les plus à droite(du) est multiple de **8**. ((d*10 + u) MOD 8 = 0)

2ème cas : Le chiffre des centaines (c) est **impair** (c MOD 2 = 1) et le nombre formé par les 2 derniers chiffres les plus à droite(du) diminué de 4 est multiple de **8**. ((d*10 + u - 4) MOD 8 = 0)

TDO

Objet	Nature / Type	Rôle
Cas1, Cas2	Booléen	Variable intermédiaire
c	entier	Chiffre des centaines
d	entier	Chiffre des dizaines
u	entier	Chiffre des unités
e	entier	Variable intermédiaire

Partie 2 : (10 points)

Analyse du programme principal

```
DEBUT Zone_Accumulation
4/ Résultat = PROC Afficher_NbZones (Nb_Zones, Recap, Nb_Recap, Deg_Min)
3/ (Recap, Nb_Recap, Nb_Zones) = Proc Déterminer_Zones (Espace, N, Recap, Nb_Recap, Nb_Zones, Deg_Min, DN)
1/ (Espace, N) = Proc Remplir_Matrice (Espace, N)
2/ (Deg_Min, DN) = Proc Valider (Deg_Min, DN)
5/ FIN Zone_Accumulation
```

Tableau des nouveaux types

Type
Tab_Mat = Tableau de 100x100 d'entiers Enreg = enregistrement Lig : entier Col : entier Nb_Uns : entier **FIN** Enreg Tab_Recap = Tableau de 100x100 d'enreg (récapitulatif des subdivisions de la matrice)

Tableau des objets globaux

Objet	Type	Rôle
Recap	Tab_Recap	Tableau des zones de concentration
Nb_Recap,	Entier	Contient le nombre d'éléments du tableau
Nb_zones	Entier	Contient le nombre de zones de concentration
Espace	Tab_Mat	Matrice carrée contenant des 0 et des 1
N	Entier	Contient l'ordre de la matrice Espace
DN	Entier	Contient le nombre de zones
Deg_Min	Entier	Degré minimum de concentration

Afficher_NbZones	Procédure	Permet d'afficher les zones de concentration min
Déterminer_Zones	Procédure	Permet de déterminer les zones et leur concentration
Remplir_Matrice	Procédure	Permet de remplir la matrice Espace
Valider	Procédure	Permet de valider la valeur de Deg_Min et DN

Analyse de la procédure Remplir_Matrice

Rôle du module : La saisie de **N**, la taille de la matrice **Espace** et son remplissage de **0** et de **1**, d'une manière aléatoire

DEF PROC Remplir_Matrice (Var Espace : Tab_Mat ; Var N : entier)
Résultat = (Espace, N)
1/ (Espace , N) = [Répéter
N = Donnée (″Introduire N : ″)
Jusqu'à (N dans [1..100])

Pour l de 1 à N faire
Pour c de 1 à N faire
Espace [l, c] ← hasard(2)

FIN pour
FIN Pour
2/ FIN Remplir_Matrice

- Validation de la dimension **N** de la matrice **Espace**
- Parcours de chaque ligne et chaque colonne d'**Espace**.
- Remplissage de chaque élément de **Espace** par une valeur au hasard comprise entre **0** et **2-1** (c'est-à dire entre **0** et**1**).

Tableau des objets locaux

Objet	Nature / Type	Rôle
L	Entier	Compteur
C	Entier	Compteur

Analyse de la procédure Valider

Rôle du module :

- Saisir et validation de **DN**, un diviseur de **N**
- Saisie et validation de **Deg_Min**, le degré de concentration minimum (**Deg_Min** ∈ [1,(**DN*DN**)]).

DEF PROC Valider (VAR Deg_Min , DN : entier)
Résultat = (Deg_Min , DN)
2/ Deg_Min = [] Répéter
Deg_Min = Donnée
Jusqu'à (Deg_Min DANS [1..DN*DN])
1/ DN = [] Répéter
DN = Donnée
Jusqu'à (N MOD DN = 0) ET (DN > 0)
3/ **FIN** Valider

- Validation de **deg-min qui doit être compris entre 1 et DN ^{2}.**
- Validation de **DN** qui doit être un diviseur de N e positif.

Analyse de la procédure Afficher_NbZones

Rôle du module : Affichage des résultats
DEF PROC Afficher_NbZones (Recap : Tab_Recap ; Nb_Recap, Nb_Zones, Deg_Min : entier)

```
Résultat = Tab-Affiché
1/ Tab-Affiché = [ Ecrire (« Le nombre de zones de concentration est : » , Nb_Zones) ]
        Pour i de 1 à Nb_Recap faire
            Si recap[i]. Nb_uns >= Deg_Min alors
              écrire (" Zone n° :" ,i, " : ligne : ", recap[i].lig , " , colonne : ", recap[i].col , " .
                    Le nombre de 1 dans cette zone est : ", recap[i].nb_uns, ". ")
            FinSi
        FinPour
2/  FIN Afficher_NbZones
```

Tableau des objets locaux

Objet	Nature / Type	rôle
i	Variable / Entier	Compteur

Analyse de la procédure Déterminer_Zones

Rôle du module : Détermination du nombre de zones

```
DEF PROC Determiner_Zones (Espace : tab_Mat ; N : entier ; VAR Recap : Tab_Recap ;
VAR Nb_Recap, Nb_Zones : entier ; Deg_Min, DN : entier)
Résultat = (Recap , Nb_Recap, Nb_Zones)
1/ (Recap, Nb_Recap , Nb_Zones) =
        [cpt ←0, Nb_Zones ← 0]
    Pour cptlig de 1 à DN faire
        Pour cptcol de 1 à DN faire
            Nbun ← 0
            Cpt ← cpt + 1
        Recap[cpt].lig ← DN * cptlig – DN + 1
        Recap[cpt].col ← DN * cptcol – DN + 1
        Pour lig de (DN * cptlig – DN + 1) à (DN * cptlig) faire
            Pour col de (DN * cptcol – DN + 1) à (DN * cptcol) faire
                Si (Espace [lig, col] = 1) alors
                    Nbun ← Nbun+1
                FinSi
            FinPour
        FinPour
        Recap[cpt].nb_uns ← nbun
    Si nb_uns ≥ Deg_min alors
        nb_zones ← nb_zones +1
```

Division horizontale de ESPACE en DN bandes, et division verticale de chaque bande en DN subdivisions

Parcours horizontal des DN lignes d'une subdivision

Sauvegarde des coordonnées de la 1ère cellule

Détermination du nombre de « 1 »

Parcours vertical des DN colonnes de la subdivision

Détermination du nombre de zones de concentration

```
            FinSi
        FinPour
    FinPour
    Nb_Recap ← cpt
2/ FIN Determiner-Zones
```

Tableau des objets locaux

Objet	Nature / Type	rôle
Cptlig , cptcol , nbun , cpt , col , lig	Variable / Entier	Compteurs

Algorithme programme principal

```
DEBUT Zone_Accumulation
1/ Proc Remplir_Matrice (Espace, N)
2/ Proc Valider (Deg_Min, DN)
3/ Proc  Déterminer_Zones (Espace, N, Recap, Nb_Recap, Nb_Zones, Deg_Min, DN)
4/ Proc Afficher_NbZones (Nb_Zones, Recap, Nb_Recap, Deg_Min)
5/ FIN Zone_Accumulation
```

Algorithme de la procédure Remplir_Matrice

```
0/ DEF PROC Remplir_Matrice (Var Espace : Tab_Mat ; Var N : entier)
1/ Répéter
                    Lire(N)
   Jusqu'à (N dans [1..100])
   Pour l de 1 à N faire
        Pour c de 1 à N faire
            Espace [l, c] ← hasard(2)
        FIN pour
   FIN Pour
2/ FIN Remplir_Matrice
```

Algorithme de la procédure Valider

```
0/ DEF PROC Valider (VAR Deg_Min : entier ; DN : entier)
1/ Répéter
            DN = Donnée
       Jusqu'à (N MOD DN = 0) ET (DN > 0)
2/ Répéter
        Lire(Deg_Min)
   Jusqu'à (Deg_Min DANS [1..DN*DN])
3/  FIN Valider
```

Algorithme de la procédure Afficher_NbZones

```
0/ DEF PROC Afficher_NbZones (Recap :Tab_Recap ; Nb_Recap, Nb_Zones, Deg_Min :
entier)
```

```
1/   Ecrire (« Le nombre de zones de concentration est : » , Nb_Zones)
   Pour i de 1 à Nb-Recap faire
              Si recap[i]. Nb_uns >= Deg_Min alors
                Ecrire (" Zone n° :" ,i, " : ligne : ", recap[i].lig , " , colonne : ", recap[i].col , " .
Le nombre de 1 dans cette zone est : ", recap[i].nb_uns, ". ")
              FinSi
    FinPour
3/ FIN Afficher_NbZones
```

Algorithme de la procédure Déterminer_Zones

```
0/ DEF PROC Determiner_Zones (Espace : tab_Mat ; D : entier ; VAR Recap : Tab_Recap ;
VAR Nb_Recap, Nb_Zones : entier ; Deg_Min, DN : entier)
1/ cpt ←0 Nb_Zones ← 0
        Pour cptlig de 1 à DN faire
            Pour cptcol de 1 à DN faire
                Nbun ← 0
                Cpt ← cpt + 1
                Recap[cpt].lig ← DN * cptlig – DN + 1
                Recap[cpt].col ← DN * cptcol – DN + 1
                Pour lig de (DN * cptlig – DN + 1) à (DN * cptlig) faire
                      Pour col de (DN * cptcol – DN + 1) à (DN * cptcol) faire
                            Si (Espace [lig, col] = 1) alors
                               Nbun←Nbun+1
                            FinSi
                      FinPour
                FinPour
                Recap[cpt].nb_uns ← nbun
                Si nb_uns >= Deg_min alors
                      nb_zones ← nb_zones +1
                FinSi
            FinPour
        FinPour
        Nb_Recap ← cpt
2/ FIN Determiner-Zones
```