Examen d'informatique

Lundi 18 mai 2015

Durée : 2h

Documents, calculettes et téléphones portables interdits
Le barème est donné à titre indicatif, il est susceptible d'être modifié

Remarques :
- vous trouverez à la fin du sujet quelques rappels de classes VBA extraits de votre polycopié de cours qui peuvent vous être utiles pour répondre à certaines questions ;
- la lisibilité et l'efficacité de vos programmes seront pris en compte pour l'évaluation de vos réponses ;
- à chaque question, vous pouvez supposer que les fonctions et procédures des questions précédentes sont définies correctement, vous pouvez donc les utiliser ;
- le barème est donné à titre indicatif, il est susceptible d'être modifié.

Remarques générales sur la notation :
- le barème initial est sur 21, la première partie de l'exercice 1 est sur 9 pts (et non 8 comme indiqué sur le sujet d'examen des étudiants)
- il y a en plus jusqu'à 1.5pt de bonus :
  - 0.5 pt de bonus si l'étudiant pense à utiliser le type Byte pour représenter le chiffre à mettre dans la grille de Sudoku (cf barème Q1.1)
  - 0.5 pt de bonus si l'étudiant pense à ne faire le tri que sur une colonne (tout en modifiant les deux) dans la quesion Q 2.3 (cf barème Q2.3)
  - 0.5 pt de bonus si les programmes sont bien présentés et bien indentés.

Le note maximale est donc 22.5/20. Si la note d'une copie est supérieure à 20, on indique 20 sur la copie et dans la feuille de report de note.

Exercice 1 : Sudoku (15 pts)

Le jeu du *Sudoku* consiste à remplir une grille de taille 9×9 avec les chiffres de 1 à 9 de manière à ce que chaque ligne, chaque colonne et chaque « sous-carré » de taille 3×3 contiennent une seule fois les chiffres de 1 à 9. La grille de jeu est ainsi un carré de 9 cases de côté subdivisé en 9 sous-grilles carrées identiques, appelées *sous-carrés*. Dans cet exercice, on considère qu'une grille de Sudoku est représentée dans la feuille Excel sur la plage de cellules allant de la cellule L1C1 à la cellule L9C9, comme le montre la grille de la figure 1. Les 9 sous-carrés de la grille sont délimités par les bordures noires dans la figure.
La solution de la grille de la figure 1 est donnée dans la figure 2. On peut en effet observer que sur chaque ligne, sur chaque colonne et dans chaque sous-carré de la solution, chaque chiffre de 1 à 9 n'apparaît qu'une seule fois. **Dans tout l'exercice, on suppose que, à tout moment, la grille est valide, c'est à dire que les chiffres dans la grille partiellement (ou complètement) remplie ne sont que des chiffres de 1 à 9 et qu'ils n'apparaissent pas plus d'une fois par ligne, colonne ou sous-carré.**

Partie 1. Possibilités par case (9 pts)

Q 1.1 Ecrire en VBA la fonction `nLigne` qui prend en paramètre un nombre `nb` et un numéro de ligne `li`, et retourne vrai si le nombre `nb` n'est pas déjà présent dans la ligne `li` et faux sinon.

	1	2	3	4	5	6	7	8	9	10
1	7									
2				3	6		9		8	
3	8	3		4			1	2		
4	2					7				
5	3				8				1	
6				1					4	
7		7	3			1		4	2	
8	4		2		3	6				
9									6	
10										

FIGURE 1 – Grille de Sudoku

	1	2	3	4	5	6	7	8	9	10
1	7	9	5	2	1	8	4	6	3	
2	1	2	4	3	6	5	9	7	8	
3	8	3	6	4	7	9	1	2	5	
4	2	8	1	5	4	7	6	3	9	
5	3	4	9	6	8	2	7	5	1	
6	5	6	7	1	9	3	2	8	4	
7	6	7	3	9	5	1	8	4	2	
8	4	1	2	8	3	6	5	9	7	
9	9	5	8	7	2	4	3	1	6	
10										

FIGURE 2 – Solution de la grille de Sudoku

```
Function nLigne(ByVal nb As Byte, ByVal li As Long) As Boolean
    Dim i As Long
    nLigne = True
    For i = 1 To 9
        If Cells(li, i).Value = nb Then
            nLigne = False
            Exit Function
        End If
    Next i
End Function
```

Barème : /2

- syntaxe : /1
  - -0.25pt si erreur ou oubli de typage
  - -0.25pt si oubli de valeur de retour
  - -0.25pt si oubli de déclarations des variables
  - -0.25pt si mauvaise syntaxe dans l'écriture de la boucle ou de la fonction
  - ....
- parcours des cases : /0.5
  - -0.25pt si pas de sortie de fonction quand la ligne est trouvée (pas de Exit ou Do Loop)
- test et valeur de retour (vrai ou faux) : /0.5
- **Bonus :** +0.5 pt s'ils pensent à utiliser le type Byte pour les chiffres de la grille

La fonction `nColonne` prend en paramètre un nombre `nb` et un numéro de colonne `co`, et retourne vrai si le nombre `nb` n'est pas déjà présent dans la colonne `co` et faux sinon. On suppose que cette fonction est définie, elle peut donc être utilisée par la suite (on ne vous demande pas de l'écrire).

```
'Non demandée
Function nColonne(ByVal nb As Byte, ByVal co As Long) As Boolean
    Dim i As Long
    nColonne = True
    For i = 1 To 9
        If Cells(i, co).Value = nb Then
            nColonne = False
            Exit Function
        End If
    Next i
End Function
```

Q 1.2 Ecrire en VBA la fonction `nCarre` qui prend en paramètre un nombre `nb`, un numéro de ligne `li` et un numéro de colonne `co`, et retourne vrai si le nombre `nb` n'est pas déjà présent dans le sous-carré auquel appartient la case `(li,co)`. L'appartenance d'une case à un sous-carré dépend de la position de la case : par exemple la case `(2,4)` appartient au sous-carré allant de la cellule L1C4 à la cellule L3C6, et la case `(8,8)` appartient au sous-carré allant de la cellule L7C7 à la cellule L9C9.

```
Function nCarre(ByVal nb As Byte, ByVal li As Long, ByVal co As Long) As Boolean
    Dim i As Long, j As Long, res As Boolean, li_initiale As Long, co_initiale As Long
    li_initiale = ((li - 1) \ 3) * 3 + 1
    co_initiale = ((co - 1) \ 3) * 3 + 1
    nCarre = True

    For i = li_initiale To li_initiale + 2
    For j = co_initiale To co_initiale + 2
        If Cells(i, j).Value = nb Then
            nCarre = False
            Exit Function
        End If
        Next j
    Next i
End Function
```

Barème : /2.5
- syntaxe : /1
  - -0.25pt si erreur ou oubli de typage
  - -0.25pt si oubli de valeur de retour
  - -0.25pt si oubli de déclarations des variables
  - -0.25pt si mauvaise syntaxe dans l'écriture de la boucle ou de la fonction
  - ....
- parcours des cases du sous-carré : /0.5
  - -0.25pt si pas de sortie de fonction quand la ligne est trouvée (pas de Exit ou Do Loop)
- calcul des valeurs limites du sous-carré (calul ou if ou case...) : /1

Q 1.3 Ecrire en VBA la fonction `nbPossibilites` qui prend en paramètre une numéro de ligne `li` et

un numéro de colonne `co`, et retourne le nombre de chiffres (entre 1 et 9) qu'il est possible de mettre dans la case `(li,co)` (en fonction des nombres déja présents sur la ligne `li`, sur la colonne `co` et dans le sous-carré auquel la case `(li,co)` appartient). On suppose que la case `(li,co)` est vide. Par exemple, l'instruction `nbPossibilites(4,7)` sur la grille de la figure 3 retournerait 4 car les quatre chiffres 3, 5, 6 et 8 sont les seuls chiffres qu'il est possible de placer dans la case `(4,7)`.

```
Function nbPossibilites(ByVal li As Long, ByVal co As Long) As Byte
Dim i As Byte, cpt As Byte
cpt = 0
    For i = 1 To 9
        If nLigne(i, li) And nColonne(i, co) And nCarre(i, li, co) Then cpt = cpt + 1
    Next i
    nbPossibilites = cpt
End Function
```

Barème : /1.5
- syntaxe : /0.5
- validité de la fonction (parcours + gestion du compteur + tests) : /1

	1	2	3	4	5	6	7	8	9	10
1	7									
2				3	6		9	7	8	
3	8	3		4	7		1	2		
4	2					7				
5	3				8		7		1	
6				1					4	
7		7	3			1		4	2	
8	4		2		3	6			7	
9				7					6	
10										

FIGURE 3 – Grille partiellement remplie

Q 1.4 Ecrire en VBA la macro `unePossibilite` qui colore en gris (`RGB(220,220,220)`) l'intérieur des cellules vides de la grille de sudoku pour lesquelles il n'y a qu'une possibilité. On rappelle qu'une cellule vide a pour valeur `""`.

```
Sub unePossibilite()
    Dim i As Long, j As Long
    For i = 1 To 9
        For j = 1 To 9
            If Cells(i, j).Value = "" And nbPossibilites(i, j) = 1 Then
                Cells(i, j).Interior.Color = RGB(220, 220, 220)
            End If
        Next j
    Next i
End Sub
```

Barème : /1
- syntaxe de Cells(i, j).Interior.Color = RGB(220, 220, 220) : /0.5
- validité de la macro (parcours + tests) : /0.5
- -0.25pt par faute de syntaxe non pénalisée dans les questions précédentes
- -0.25pt si paramètres

Q 1.5 On s'intéresse dans cette question à l'estimation du niveau de difficulté de la grille. On considère que le niveau de difficulté de la grille dépend du nombre moyen de possibilités par case vide. La grille est considérée comme *facile* si le nombre moyen de possibilités par case vide est inférieur à 2, *moyen* s'il est compris entre 2 et 4, *difficile* s'il est entre 4 et 6, et *diabolique* sinon. Ecrire en VBA la macro `difficulte` qui détermine le nombre moyen de possibilités par case vide et affiche par `MsgBox` le niveau de difficulté de la grille.

```
Sub difficulte()
    Dim i As Long, j As Long, somme As Integer, moyenne As Double, cpt As Byte
    somme = 0
    cpt = 0
    For i = 1 To 9
        For j = 1 To 9
            If Cells(i, j).Value = "" Then
                somme = somme + nbPossibilites(i, j)
                cpt = cpt + 1
            End If
        Next j
    Next i
    moyenne = somme / cpt
    Select Case moyenne
        Case 0 To 2
            MsgBox "Niveau facile"
        Case 2 To 4
            MsgBox "Niveau moyen"
        Case 4 To 6
            MsgBox "Niveau difficile"
        Case Else
            MsgBox "Niveau diabolique"
    End Select
End Sub
```

Barème : /2
- calcul de la moyenne (gestion des compteurs + boucles) : /1
- bonne gestion de l'affichage (MsgBox + tests) : /1
- -0.25pt par faute de syntaxe non pénalisée dans les questions précédentes
- -0.25pt si paramètres

Partie 2. Indications par chiffre (5 pts)

Dans cette partie de l'exercice, on s'intéresse à l'affichage d'un indice pour l'utilisateur qui cherche à résoudre la grille de sudoku. Puisque chaque chiffre de 1 à 9 apparaît exactement 9 fois dans une grille de sudoku résolue, l'indice consiste à déterminer si un chiffre x est déjà placé dans 8 cases de la grille, car dans ce cas la place du 9e x est évidente. Par exemple, dans la figure 3 le chiffre 7 est déjà présent 8 fois dans la grille, le 9e 7 de cette grille doit donc être placé dans la case `(6,3)`.

Q 1.6 Ecrire en VBA la fonction `verifieParChiffre` qui retourne un chiffre de 1 à 9 qui est déjà présent 8 fois dans la grille. De plus, la fonction donne aussi le numéro de ligne et le numéro de colonne de la case dans laquelle le 9e chiffre doit être placé (précisons qu'il n'est pas demandé ici d'utiliser de boîtes de dialogues). Si aucun des chiffres de 1 à 9 n'est déjà présent 8 fois dans la grille, la fonction `verifieParChiffre` retourne 0.
NB : la notation de cette question tient compte de l'efficacité de votre algorithme.

```
'Cette version est sur 2.5 (une version sans exit function sera /2)
Function verifParChiffre(ByRef li As Long, ByRef co As Long) As Byte
    Dim nb As Byte, i As Long, nbL As Byte
    nbL = 0
    For nb = 1 To 9
        For i = 1 To 9
            If nLigne(nb, i) Then
                li = i: nbL = nbL + 1
            End If
            If nColonne(nb, i) Then
                co = i
            End If
        Next i
        If nbL = 1 Then
            verifParChiffre = nb
            Exit Function
        End If
    Next nb
    verifParChiffre = 0
End Function

'Cette version est /3
Function verifParChiffreBis(ByRef li As Long, ByRef co As Long) As Byte
    Dim nb As Byte, i As Long, nbL As Byte, nbC As Byte
    nbL = 0: nbC = 0
    For nb = 1 To 9
        i = 1
        Do While i < 10 And nbL < 2 And nbC < 2
            If nLigne(nb, i) Then
                li = i: nbL = nbL + 1
            End If
            If nColonne(nb, i) Then
                co = i: nbC = nbC + 1
            End If
            i = i + 1
        Loop
        If nbL = 1 And nbC = 1 Then
            verifParChiffreBis = nb
            Exit Function
        End If
    Next nb
    verifParChiffreBis = 0
End Function
```

Barème : /3
- ByRef : /0.5
- validité : /1.5
- sortie quand un chiffre peut être retourné (Do Loop ou Exit) : /0.5
- arrêt d'examen d'une ligne ou colonne dès qu'on sait qu'il y a moins de 7 fois le chiffre dans la grille (i.e. chiffre absent plus de 2 fois) : /0.5
- -0.25pt par faute de syntaxe non pénalisée dans les questions précédentes

Q 1.7 On appelle n la taille de la grille de dimension $n \times n$ (dans cet exercice, on a donc $n = 9$). Donner l'ordre de grandeur du nombre d'opérations de votre fonction `verifieParChiffre` dans le pire cas et dans le meilleur cas en fonction de n. Préciser brièvement ce que sont le meilleur cas et le pire cas.

Pire cas : le dernier chiffre testé (9 dans la correction) est présent 8 fois dans la grille : $O(n^3)$
Meilleur cas : le premier chiffre testé (1 dans la correction) est présent 8 fois dans la grille : $O(n^2)$

Barème : /1
- description des cas : /0.5 (0.25 par cas)
  - les pire et meilleur cas doivent bien correspondre à ce qu'ils ont écrit
- complexité : /0.5 (0.25 par cas)
  - la complexité doit bien correspondre à ce qu'ils ont écrit. Tout résultat de type $n \times (2n^2) + Cte$ ou $n \times (2(n-1)^2) + Cte$ (où Cte est une constante) est accepté.

On peut être souple dans la notation de cette question : si l'on sent que l'étudiant a la bonne intuition on peut donner les points, même si la réponse n'est pas juste ou si elle est imprécise ou incomplète

Q 1.8 Ecrire en VBA la macro `testVPC` qui affiche le message « Le nombre x peut être placé dans la case (l,c) » lorsqu'un nombre x (entre 1 et 9) est déjà présent 8 fois dans la grille et que la seule place possible pour le 9e x est la case (l,c), et n'affiche rien sinon.

```
Sub testVPC()
    Dim li As Long, co As Long, nb As Byte
    nb = verifParChiffre(li, co)
    If nb > 0 Then
        MsgBox "le nombre " & nb & " peut être placé dans la case (" & li & "," & co & ")"
    End If
End Sub
```

Barème : /1
- -0.25 par erreur

Partie 3. Evènements (1 pt)

Q 1.9 On suppose qu'en plus de la grille, la feuille Excel est dotée d'un bouton nommé « Arreter » et d'un bouton nommé « Lancer ». Expliquer brièvement et le plus simplement possible ce que fait le programme suivant.

```
Dim tps As Date

Private Sub Arreter_Click()
    End
End Sub

Private Sub Lancer_Click()
    tps = Now()
    Do
        Cells(2, 11).Value = Now() - tps
        DoEvents
    Loop
End Sub
```

Barème : /1
- /0.5 : chronomètre le temps
- /0.5 : affiche le temps qui défile

Exercice 2 : Représentation binaire (6 pts)

Q 2.1 (2 pts) Ecrire en VBA la fonction `binToDec` qui prend en paramètre un nombre représentant un entier positif en binaire, et retourne sa valeur en base 10. Par exemple `binToDec(111001)` retournerait 57.

```
Function binToDec(ByVal nb As Long) As Long
Dim n As Long, puiss As Long, res as Long
n = nb: puiss = 1 : res = 0
Do while n > 0
   res = res + (n Mod 10)*puiss
   puiss = puiss * 2
   n = n \10
Loop
binToDec = res
End Function
```

Barème : /2
- gestion de l'accès à chaque chiffre du nombre passé en paramètre : /1
  - on considère que c'est correct si `nb` est de type chaîne de caractères et qu'ils utilisent `Mid` et `Len` pour accéder au chiffre de `nb`
  - Rq : n Mod 2 est aussi correct pour récupérer le chiffre des unités, ainsi que n Mod x pour tout x tel que $10 > x > 1$
- gestion du calcul du nombre en base 10 : /1
- -0.25pt par faute de syntaxe non pénalisée dans les questions précédentes

Q 2.2 (1 pt) Ecrire en VBA la macro `base10` qui parcourt des nombres binaires listés dans la colonne 1 de la feuille Excel en inscrivant leur valeur en base 10 dans la colonne 2 à la ligne correspondante. On ne sait pas à l'avance combien de nombres binaires sont présents dans la colonne 1. On arrête le parcours des nombres binaires dès que l'on rencontre une cellule vide. On rappelle qu'une cellule vide en VBA a la valeur `""`.

```
Sub base10()
Dim i as Long
i = 1
Do until cells(i,1).value = ""
  cells(i,2).value = binToDec(cells(i,1).Value)
  i= i+1
Loop
End sub
```

Barème : /1
- parcours valide (boucle) : /0.5
- remplissage de la colonne 2 : /0.5
- -0.25pt par faute de syntaxe non pénalisée dans les questions précédentes

Q 2.3 (3 pts) On suppose que la macro `base10` a été exécutée. Ecrire en VBA la macro `tri` qui trie en ordre croissant les nombres listés dans les colonnes 1 et 2 de la feuille Excel. Dans cette question, la notation ne tient pas compte du choix de l'algorithme de tri.

```
sub tri()
'tri par sélection
dim i as long, indT as long, mini as long, temp as Long
indT = 1
Do until cells(indT+1,1).value = ""
  'recherche du min entre indT et la fin
  mini = indT : i = indT+1
  Do until cells(i,1).value = ""
    if cells(i,1).value < cells(mini,1).value Then
      mini = i
    end if
    i = i+ 1
  Loop

  'Echange de mini et indT pour les 2 colonnes
  temp = cells(mini,1).value : cells(mini,1).value = cells(indT,1).value
  cells(indT,1).value = temp
  temp = cells(mini,2).value : cells(mini,2).value = cells(indT,2).value
  cells(indT,2).value = temp

  'On recommence sur le reste du tableau
  indT = indT + 1
Loop
end sub
```

Barème : /3
- tri : /2
  - recherche du min à chaque itération (tri sélection ou tri bulle) : /1
  - placement du min à la bonne position (échange des valeurs correct) (tri sélection ou tri bulle) : /1
- parcours des valeurs (="") : /0.5
- gestion des deux colonnes (tri sur une seule suffisant) : /0.5
- -0.25pt par faute de syntaxe non pénalisée dans les questions précédentes
- **Bonus : +0.5pt s'ils pensent à ne faire le tri que sur l'une des deux colonnes**