Les types utilisateurs (VBA)
Corrigé

Résumé

Ce document décrit comment traduire en VBA les types utilisateur du langage algorithmique.

Table des matières

Liste des exercices

1 Types énumérés

Les types énumérés n'existent pas en Visual BASIC. On utilise donc des constantes de valeurs différentes, chaque constante ayant le nom d'une valeur symbolique du type énuméré algorithmique. Le type des constantes est généralement entier, mais on peut aussi utiliser des chaînes de caractères.

```
1 Const JANVIER=1, FEVRIER=2, MARS=3, AVRIL=4, MAI=5, JUIN=6, JUILLET=7, AOUT=8,
2 Const SEPTEMBRE=9, OCTOBRE=10, NOVEMBRE=11, DÉCEMBRE=12
```

Remarque : Lorsqu'on veut scinder en 2 lignes « physiques » une longue ligne « logique », on peut utiliser les 2 caractères successifs Espace et Souligné à la fin de la 1re ligne pour signifier la continuité sur la suivante :

```
1 Const JANVIER=1, FEVRIER=2, MARS=3, AVRIL=4, MAI=5, JUIN=6, JUILLET=7, AOUT=8, _
2       SEPTEMBRE=9, OCTOBRE =10, NOVEMBRE=11, DÉCEMBRE=12
```

2 Les tableaux

2.1 Motivation

2.2 Définition

2.3 Tableaux à une dimension

Les tableaux en Visual BASIC sont beaucoup plus contraignants qu'en notation algorithmique : on ne peut nommer un type tableau ; le type des indices est forcément entier.

Pour déclarer un tableau, on précise le type des éléments et la plage de variation des indices :

```
Dim vect(0 To 9) As Double   ' vect est un tableau de 10 réels indicés de 0 à 9
```

Les indices valides sont donc 0, 1, 2, ..., 8 et 9.

C'est équivalent à la déclaration suivante en notation algorithmique :

```
Variable  vect : Tableau [0..9] De Réel
```

L'accès à un élément se fait en utilisant des parenthèses :

```
vect(5) = 10
vect(1) = vect(5) + 1
```

Lors de la déclaration, il est possible de ne préciser que la borne supérieure des indices, la borne inférieure étant alors considérée par défaut à 1 :

```
Dim vect (10) As Double   ' vect est un tableau de 10 réels :
                                          ' vect (1), vect (2), ... vect (10)
```

Remarque : Il est recommandé d'utiliser une constante symbolique à la place de valeurs numériques, car celle-ci sera réutilisée dans les boucles qui permettront de manipuler le tableau :

```
Const N_MAX = 10
Dim vect(N_MAX) As Double
```

Attention : L'affectation entre tableaux est impossible en Visual BASIC. Il est nécessaire de faire une affectation élément par élément avec une boucle For ; par exemple, si VectA et VectV sont 2 tableaux de N_MAX éléments :

```
For n = 1 to N_MAX
        vectB(n) = vectA(n)
Next n
```

Attention : De même, la comparaison directe entre 2 tableaux est impossible et nécessite de passer par une boucle.

Remarque : L'utilisation d'une valeur d'indice débordant de celles définies dans la déclaration du tableau provoque une erreur d'exécution.

2.4 Exemples

Exercice 1 : Initialiser un tableau

Initialiser astucieusement un tableau de 10 entiers pour qu'il contienne les valeurs suivantes :

1	2	3	4	5	6	7	8	9	10

Solution : Si on considère un tableau définit par :

```
Type
    Vecteur = Tableau [1..10] De Entier
```

On constate que la valeur de la case d'indice i est justement i. Pour initialiser un vecteur, il suffit donc de parcourir les entiers de 1 à 10 pour remplir la case correspondante.

```
Variable
    tab: Vecteur
    indice: Entier
Début
    Pour indice <- 1 JusquÀ indice = 10 Faire
        tab[indice] <- indice
    FinPour
Fin
```

```
Attribute VB_Name = "Exo1_tab_initialiser"
'*********************************************************************
'* Auteur : Claude Monteil <monteil@ensat.fr>
'* Version : 1.0
'* Objectif : Initialiser un tableau avec dans l'ordre les valeurs :
'* 1, 2, 3, 4, 5, 6, 7, 8, 9 et 10.
'*********************************************************************

Option Explicit

Sub tab_initialiser()
    Dim tableau(10) As Integer  ' le tableau à initialiser
      'NOTA : la variable Tab n'est pas utilisable
      '       car c'est une fonction predefinie
    Dim indice As Integer   ' parcourir les entiers de 1 à 10

    For indice = 1 To 10 ' le pas est omis quand il est égal à un
        tableau(indice) = indice
    Next indice
End Sub
```

Exercice 2 : Afficher un tableau d'entier

Écrire un programme qui affiche tous les éléments d'un tableau de capacité MAX (égale à 10) mais dont la taille effective (c'est-à-dire le nombre d'éléments utiles) est donné par la variable `nb`.

Les éléments du tableau seront écrits entre crochets, dans l'ordre croissant de leur indice et séparés par des points-virgules. Voici quelques exemples :

```
[]              -- un tableau vide (de taille effective nulle)
[ 1; 2; 3]      -- tableau contenant les 3 valeurs 1, 2 et 3.
[ 421 ]         -- tableau contenant la seule valeur 421
```

Solution : On constate que l'on ne peut pas afficher directement chaque élément du tableau car le point-virgule ne doit pas être mis après le dernier élément. le principe est donc d'afficher d'abord

le premier élément puis, pour tous les éléments suivants, d'afficher le point-virgule séparateur et l'élément lui-même. Ceci ne peut bien sûr être fait que si le tableau contient au moins un élément.

```
Attribute VB_Name = "Exo2_tab_afficher"
'*****************************************************************
'* Auteur : Claude Monteil <monteil@ensat.fr>
'* Version : 1.0
'* Objectif : afficher un tableau d'entiers de taille effective nb
'*****************************************************************

Sub tab_afficher()
    Const MAX As Integer = 10
    Dim tableau(MAX) As Integer  ' le tableau à afficher
    Dim nb As Integer 'taille effectivement utilisée du tableau
    Dim i As Integer ' indice pour parcourir les entiers de 1 à nb
    Dim chaine As String ' chaine correspondant à l'affichage du tableau

    EffacerEcran "Affichage d'un tableau"

   '1.initialiser le tableau
    tableau(1) = 4: tableau(2) = 2: tableau(3) = 1
     'en Visual Basic, il n'y pas d'instruction standard pour définir
     'une constante de type tableau, du style (4,2,1), par exemple
     ' => necessite d'initialiser chaque élément 1 à 1
     '    (ou de se définir ses propres fonctions : on verra plus tard)
    Afficher "Taille effective du tableau"
    Saisir nb

   '2.Afficher le tableau
     ' puisque l'instruction Afficher réalise un saut de ligne implicite,
     ' on construit l'information à afficher dans un chaine
     ' sur laquelle on exécute ensuite l'instruction : Afficher chaine
    chaine = "["
    If nb > 0 Then chaine = chaine & tableau(1) '1er élément
    For i = 2 To nb 'éléments suivants
        If i <= MAX Then chaine = chaine & "; " & tableau(i)
    Next i
    chaine = chaine & "]"
    Afficher chaine
End Sub
```

2.5 Chaînes de caractères

En Visual BASIC, les chaînes de caractères ne sont pas considérées comme des cas particuliers de tableaux, mais comme un type spécifique. Il est donc possible d'affecter une chaîne à une autre, de tester l'égalité, et même de comparer 2 chaînes entre elles (c'est l'ordre alphabétique qui est pris en compte). On peut connaître la longueur d'une chaîne ch (nombre de caractères qui la composent) avec la fonction **Len**(ch), ou accéder au n-ème caractère de la chaîne avec la fonction **Mid**(ch,n,1).

Voici quelques fonctions spécifiques concernant les chaînes de caractères (voir l'aide en ligne de l'éditeur VBA sur chacune de ces fonctions pour plus de détails) :

```
ch = "Bon" & "jour"
   ' concatène 2 chaînes : résultat = "Bonjour"
ch = Left("Bonjour", 3)
   ' renvoie les 3 caractères de gauche : résultat = "Bon"
ch = Mid("Bonjour", 5, 2)
   ' renvoie la sous-chaîne démarrant à 5 de longueur 2 : résultat = "ou"
ch = Right("Bonjour", 4)
   ' renvoie les 4 caractères de droite : résultat = "jour"
ch = Replace("Bonjour", "jour", "␣appétit")
  ' renvoie une chaîne où une sous-chaîne est remplacé par une autre :
  ' résultat = "Bon appétit"
pos = InStr("Bonjour", "ou")
   'indique la position de la sous-chaine "ou" dans la chaîne "Bonjour" :
   'résultat = 5 (renvoie 0 si la sous-chaîne est absente)
ch = UCase("Bonjour")
   ' renvoie la chaîne en majuscules (Upper Case) : résultat = "BONJOUR"
ch = LCase("Bonjour")
   ' renvoie la chaîne en minuscules (Lower Case) : résultat = "bonjour
ch = String(5, "-")
   ' renvoie une chaîne constituée de caractères identiques :
   ' résultat = "-----"
ch = LTrim("␣␣bon␣jour␣␣")
  ' renvoie une chaîne privée de ses espaces de tête (Left) :
  ' résultat = "bon␣jour␣␣"
ch = RTrim("␣␣bon␣jour␣␣")
  ' renvoie une chaîne privée de ses espaces de queue (Right) :
  ' résultat = "␣␣bon␣jour"
ch = Trim("␣␣bon␣jour␣␣")
  ' renvoie une chaîne privée de ses espaces d'extrémités :
  ' résultat = "bon␣jour"
ch = Format(0.196, "0.00%")
  ' renvoie une chaîne correspondant au formatage d'un nombre :
  ' résultat = "19.60%"
```

2.6 Tableaux à plusieurs dimensions

Un tableau à plusieurs dimensions se déclare de la façon suivante :

```
Const NB_LIGNES = 5, NB_COLONNES = 10
Dim m1(NB_LIGNES, NB_COLONNES) As Double
Dim m2(NB_LIGNES, NB_COLONNES) As Double
Dim m3(NB_LIGNES, NB_COLONNES) As Double
```

L'accès à un élément du tableau utilise les parenthèses : `m1(i, j)`.

3 Les enregistrements

Les enregistrements sont déclarés avec la séquence **`Type`** ... **`End Type`**. L'accès aux champs utilise la même notation pointée qu'en algorithmique. On peut affecter une variable d'un type enregistrement à une autre variable du même type.
Attention : Le mot-clé **`Type`** sert exclusivement à la déclaration de types-enregistrements : il n'a aucune autre utilisation en Visual BASIC.

```
1 Type DatePerso ' on ne peut utiliser l'identificateur Date qui est prédéfini
2      jour As Integer  ' numero du jour dans le mois
3      mois As Integer  ' numero du mois dans l'annee
4      annee As Integer ' numero de l'annee
5 End Type
```