Les sous-programmes: exercices corrigés en VBA
Corrigé

Exercice 1 : Livres

L'objectif de cet exercice est d'écrire un programme pour gérer une bibliothèque personnelle de taille réduite. On ne développera qu'une interface homme/machine minimale.

1.1 *Définition d'un livre.* Un livre est caractérisé par son titre, son nombre de pages et son genre. Le genre est soit roman, soit BD, soit art, soit technique, etc. Il faudrait bien sûr bien d'autres informations pour décrire complètement un livre.

1.1.1 Donner le type Livre.

Solution : D'après sa caractérisation, un livre peut être représenté par un type énuméré avec des champs titre (une chaîne de caractères), son nombre de page (entier) et son genre (un type énuméré puisqu'il peut prendre une parmi plusieurs valeurs).

```
Constante
    LG_TITRE = 50       -- Longueur maximal pour le titre d'un livre
Type
    Genre = (ROMAN, BD, ART, TECHNIQUE)

    Livre =
        Enregistrement
            titre: Chaîne[LG_TITRE]
            nb_pages: Entier
            genre: Genre
        FinEnregistrement
```

```
Const ROMAN = 1, BD = 2, ART = 3, TECHNIQUE = 4

Type typ_Livre
    titre As String
    nb_pages As Integer
    genre As Integer
End Type
```

1.1.2 Écrire un sous-programme qui permet d'initialiser un livre à partir de son titre, de son nombre de pages et de son genre.

Solution : Ce sous-programme prend en paramètre :

- le livre à initialiser (**out**) ;
- le titre du livre (**in**) ;
- le nombre de pages (**in**)
- et le genre du livre (**in**).

Comme il s'agit d'un sous-programme d'initialisation, nous en faisons une procédure dont voici la spécification.

```
Procédure initialiser_livre(livre: out Livre;
                            titre: in Chaîne;
                            npage: in Entier;
                            genre: in Genre) Est
    -- Initialiser un livre à partir de son titre, son nombre de pages
    -- et son genre.
    --
    -- Nécessite
    --  npage > 0
    --
    -- Assure
    --  livre.titre = titre
    --  livre.nb_pages = npages
    --  livre.genre = genre
```

Le code du sous-programme est alors immédiat puisqu'il s'agit d'initialiser tous les champs de l'enregistrement.

```
Début
    livre.titre <- titre
    livre.nb_pages <- npages
    livre.genre <- genre
Fin
```

```
Sub initialiser_livre(livre As typ_Livre, _
                      ByVal titre As String, ByVal npages As Integer, ByVal genre As I
'ROLE : Initialise un livre a partir de son titre, son nombre de pages et son genre.
'ENTREE titre, npages (>0), genre
'SORTIE livre
    livre.titre = titre
    livre.nb_pages = npages
    livre.genre = genre
End Sub
```

1.1.3 Écrire un sous-programme qui permet d'afficher un livre.

Solution : Ce sous-programme n'a qu'un seul paramètre, le livre qui est en entrée. De plus il fait de l'affichage. Il s'agit donc nécessairement d'une procédure.

```
Procédure afficher_livre(livre: in Livre) Est
    -- Afficher un livre
```

Le code consiste à afficher chacun des champs.

```
Début
    Afficher livre.titre
    Afficher livre.nb_pages
    Afficher livre.genre
Fin
```

Pour afficher le genre du livre, on peut écrire un nouveau sous-programme.

```
Function libelle_genre(genre As Integer) As String
'ROLE : renvoie sous forme de chaine de caracteres le libelle associe au code de genre
'ENTREE genre
    Select Case genre
        Case ROMAN:     libelle_genre = "ROMAN"
        Case BD:        libelle_genre = "BD"
        Case ART:       libelle_genre = "ART"
        Case TECHNIQUE: libelle_genre = "TECHNIQUE"
        Case Else:      libelle_genre = "ERREUR ''''"
    End Select
End Function

Sub afficher_livre(livre As typ_Livre)
'ROLE : affiche les informations attachées à un livre
'ENTREE livre
    Afficher livre.titre, livre.nb_pages & " pages ", libelle_genre(livre.genre)
End Sub
```

1.2 *La bibliothèque.* Nous nous limitons à une bibliothèque qui ne peut pas contenir plus de 1000 ouvrages qui sont rangés dans l'ordre alphabétique de leur titre.

1.2.1 Définir le type bibliothèque.

Solution : Nous représentons la bibliothèque par un tableau de capacité 1000. Cependant, pour connaître le nombre d'ouvrages effectivement présents, il faut gérer la taille (entier). La bibliothèque est donc un type enregistrement.

```
Constante
    MAX_LIVRES = 1000   -- nb maximal de livres dans la bibliothèque
Type
    Bibliothèque =
        Enregistrement
            livres: Tableau [1..MAX_LIVRES] De Livre
            nb_livres: Entier
        FinEnregistrement
```

```
Const MAX_LIVRES As Integer = 1000 ' nombre maximal de livres dans la bibliotheque

Type typ_Bibliotheque
    Livres(MAX_LIVRES) As typ_Livre
    nb_livres As Integer
End Type
```

1.2.2 Écrire un sous-programme pour initialiser une bibliothèque à vide.

Solution : Ce sous-programme prend en paramètre la bibliothèque (**out**). Il s'agit d'une procédure.

```
Procédure initialiser_bibliothèque(b: out Bibliothèque) Est
    -- Initialiser la bibliothèque à vide
    --
    -- Assure
    --  b.nb_livres = 0 -- bibliothèque vide
```

Le code est donc immédiat.

```
Début
    b.nb_pages <- 0
Fin
```

Notons qu'il n'est pas nécessaire d'initialiser les livres du tableau puisque la champs `nb_pages` mis à zéro indique qu'on ne peut accéder à aucune case du tableau `livres`.

```
Sub initialiser_bibliotheque(b As typ_Bibliotheque)
'ROLE : initialise la bibliotheque specifiee comme vierge (0 livres)
'SORTIE b
    b.nb_livres = 0
End Sub
```

1.2.3 Écrire un sous-programme pour afficher le contenu de la bibliothèque.
Solution : Ce sous-programme prend un seul paramètre, la bibliothèque (en **in**). Il s'agit donc d'une procédure.

```
Procédure afficher_bibliothèque(b: in Bibliothèque) Est
    -- Afficher les livres de la bibliothèque.
```

Pour l'implémentation, nous choisissons d'utiliser une répétition **Pour** puisque nous connaissons le nombre de livres à afficher.

```
Variable
    i: Entier   -- parcourir les livres
Début
    Pour i <- 1 JusquÀ i = b.nb_livres Faire
        afficher_livres(b.libres[i])
    FinPour
Fin
```

```
Sub afficher_bibliotheque(b As typ_Bibliotheque)
'ROLE : afficher les livres contenus dans la bibliotheque specifiee
'ENTREE b
    Dim i As Integer ' index pour parcourir les livres

    For i = 1 To b.nb_livres
        afficher_livre b.Livres(i)
    Next i
End Sub
```

1.2.4 Écrire un sous-programme pour ajouter un ouvrage dans la bibliothèque.
Solution : Ce sous-programme prend en paramètre :

- la bibliothèque dans laquel le livre doit être ajouté (**in out**) ;
- le livre à ajouter (**in**).

Il s'agit donc d'une procédure.

```
Procédure ajouter(b: in out Bibliothèque; l: in Livre) Est
    -- ajouter le livre l dans la bibliothèque b
```

Les livres sont rangés dans la bibliothèque dans l'ordre lexicographique des titres. Il s'agit donc de faire un tri par insertion pour trouver la place de du nouveau livre. Nous optons pour une insertion séquentielle dont voici le premier niveau de raffinage.

```
R1 : Raffinage De « ajouter »
  | Déterminer la position du livre     position: out Entier
  | Décaler les livres compris entre position et b.nb_livres
  | Ranger le nouveau livre
```

Ces étapes correspondant à des algorithmes classiques sur les tableaux, il ne sont pas détaillés en algorithmique.

```
Sub ajouter(b As typ_Bibliotheque, l As typ_Livre)
'ROLE : ajouter le livre l dans la bibliotheque b. L'ajout s'effectue par ordre
'       alphabetique du titre, en decalant les livres situés après
'HYPOTHESE : b.nb_livres < MAX_LIVRES  (bibliotheque non pleine)
'ENTREE l
'M.A J. b
    Dim i As Integer        ' index pour parcourir les livres
    Dim position As Integer ' position où doit être range le livre l
    Dim AInserer As Boolean ' indique si le livre doit etre insere avant d'autres
                            ' sinon il sera ajoute apres le dernier deja present

   '1.Determiner la position d'insertion du livre
    position = 1
    AInserer = False 'a priori
    Do While Not AInserer And (position <= b.nb_livres)
        If l.titre < b.Livres(position).titre Then
            AInserer = True
        Else
            position = position + 1
        End If
    Loop

   '2.Decaler les livres si besoin est
    If AInserer Then
        For i = b.nb_livres To position Step -1 'attention : boucle en sens rétrograde
            b.Livres(i + 1) = b.Livres(i)
        Next i
    End If

  '3.Ranger le livre
    b.Livres(position) = l
    b.nb_livres = b.nb_livres + 1
End Sub
```

1.2.5 Écrire un sous-programme pour indiquer combien il y a d'ouvrages d'un genre donné.

Solution : Ce sous-programme a pour paramètres :

- la bibliothèque dans laquelle on veut recenser les ouvrages (**in**) ;
- le genre cherché (**in**) ;
- le nombre d'ouvrages du genre cherché (**out**).

Un seul paramètre en sortie, deux en entrée. On peut donc en faire une fonction.

```
Fonction nb_livres_genre(b: in Bibliothèque; genre: in Genre): Entier Est
    -- Nombre de livres de la bibliothèque b qui sont du genre spécifié.
```

Concernant l'implantation, il faut parcourir tous les ouvrages (boucle **Pour**) et comptabiliser ceux qui sont du genre demandé.

```
Variable
    i: Entier   -- parcourir les livres
Début
    Résultat <- 0
    Pour i <- 1 JusquÀ i = b.nb_livres Faire
        Si b.livres[i].genre = genre Alors
            Résultat <- Résultat + 1
        FinSi
    FinPour
Fin
```

```
Function nb_livres_genre(b As typ_Bibliotheque, genre As Integer) As Integer
'ROLE : renvoie le nombre de livres de la bibliotheque b qui sont du genre specifie.
'ENTREE b, genre
    Dim resultat As Integer
    Dim i As Integer ' index pour parcourir les livres

    resultat = 0
    For i = 0 To b.nb_livres
        If b.Livres(i).genre = genre Then resultat = resultat + 1
    Next i
    nb_livres_genre = resultat
End Function
```

1.3 *Programme de test.* Écrire un programme de test des sous-programmes écrits ci-dessus.
Solution : Voici un programme de test minimal qui ajoute quelques livres dans une bibliothèque et affiche le nombre de livres des différents genres.

```
Sub tester_bibliotheque()
'ROLE : tester les procedures et fonctions de gestion d'une bibliotheque
    Dim MaBiblio As typ_Bibliotheque

    EffacerEcran "Test␣des␣procédures␣de␣gestion␣d'une␣bibliothèque"

    initialiser_bibliotheque MaBiblio
    afficher_bibliotheque MaBiblio

    AjouterLivreEtAfficher MaBiblio, "Peinture", 120, ART
    AjouterLivreEtAfficher MaBiblio, "Tintin", 54, BD
    AjouterLivreEtAfficher MaBiblio, "Algorithmique", 540, TECHNIQUE
    AjouterLivreEtAfficher MaBiblio, "Apprendre␣VBA", 140, TECHNIQUE

    Afficher "Nb␣livres␣ART␣:␣" & nb_livres_genre(MaBiblio, ART)
    Afficher "Nb␣livres␣BD␣␣:␣" & nb_livres_genre(MaBiblio, BD)
    Afficher "Nb␣livres␣ROMAN␣␣:␣" & nb_livres_genre(MaBiblio, ROMAN)
    Afficher "Nb␣livres␣TECHNIQUE␣␣:␣" & nb_livres_genre(MaBiblio, TECHNIQUE)

End Sub

Sub AjouterLivreEtAfficher(b As typ_Bibliotheque, _
```

```
23              ByVal titre As String, ByVal npages As Integer, ByVal genre As Integer)
24  'ROLE : ajoute un livre du titre, nombre de pages et genre specifies dans la bibliothe
25  '       indiquee et affiche le contenu de la bibliotheque a l'issue de cet ajout
26  '       (programme appelé pour tester l'ajout successif de livres dans la bibliotheque
27  'ENTREE titre, npages (>0), genre
28  'M.A J. b
29      Dim LivreCourant As typ_Livre
30      Afficher "-> Ajouter le livre :"
31      initialiser_livre LivreCourant, titre, npages, genre
32      afficher_livre LivreCourant
33      Afficher "-> La bibliotheque contient :"
34      ajouter b, LivreCourant
35      afficher_bibliotheque b
36      Afficher "--------"
37    End Sub
```