Les sous-programmes : exercices résolus en VBA
Corrigé

Objectifs

- Savoir écrire des sous-programmes ;
- Comprendre les modes de passage de paramètres ;
- Faire la différence entre fonction et procédure ;
- Continuer à appliquer les principes des semaines 1 & 2.

1 Édition simplifiée d'une facture

L'objectif de ces exercices est de réaliser l'édition d'une facture dans une version relativement simplifiée puisqu'il s'agit de pouvoir ajouter et/ou supprimer des lignes sur une facture et de l'éditer à l'écran.

Nous ne traiterons pas l'interface homme/machine (IHM) et nous nous contenterons donc de développer les types et sous-programmes qui constituent ce modèle.

Exercice 1 : Notion de produit

Commençons par définir le type Produit et les opérations associées.

1.1 Écrire le type Produit sachant qu'un produit est caractérisé par un code (sur 3 caractères), un libellé (9 caractères) et un prix hors taxe.

Solution : Le type Produit est un enregistrement composé de trois champs un code de type chaîne de 3 caractères, un libellé (chaîne de 9 caractères) et un prix hors taxe (**Réel**).

```
1 Type Produit
2     code As String
3     libelle As String
4     prix_ht As Double
5 End Type
```

1.2 Écrire un sous-programme pour initialiser un produit à partir de son code, son libellé et son prix hors taxe.

Solution : Le but du sous-programme est « initialiser un produit à partir de son code, son libellé, et son prix hors taxe ».

Les paramètres sont donc :
- le produit à initialiser : **out** ;
- le code du produit : **in** ;
- le libellé du produit : **in** ;
- le prix du produit : **in**.

On constate qu'il n'y a qu'un seul paramètre en sortie et tous les autres sont en entrée. Nous en faisons quand même une procédure car il s'agit d'un sous-programme d'initialisation. Il est préférable dans ce cas que la mémoire soit fournie par le programme appelant (ceci évite par exemple de faire des copies inutiles lors du retour de la fonction).

On peut donc en déduire la spécification.

```
Procédure initialiser_produit(p: out Produit) ;
            c: in Code ; l: in Libellé ; prix_ht: in Réel) Est
    -- Initialiser le produit p à partir de son code c, de son libellé l
    -- et de son prix hors taxe.
```

On peut maintenant en déduire le code.

```
    Début
        produit.code <- c
        produit.libellé <- l
        produit.prix_ht <- prix_ht
    Fin
```

```
Sub initialiser_produit(p As Produit, _
                        ByVal c As String, ByVal l As String, ByVal prix_ht As Double)
'ROLE : Initialiser un nouveau produit p avec un code, un libellé et un prix hors taxe
'ENTREE c, l, prix_ht : code, libellé et prix à affecter au produit
'SORTIE p
    p.code = c
    p.libelle = l
    p.prix_ht = prix_ht 'pas d'ambiguïté entre les 2 identificateurs prix_ht :
      'le 1er est forcément un champ de l'enregistrement, le second une variable ou
      'un paramètre formel (ici, c'est un paramètre formel)
End Sub
```

Exercice 2 : Notion de ligne de facture

Le type LigneFacture permet de représenter une ligne d'une facture. Il est caractérisé par le produit concerné, la quantité commandée et le prix total hors taxe de la ligne.

2.1 Définir le type LigneFacture.

Solution : Le type LigneFacture est également un enregistrement. Notons la redondance entre le prix hors taxe de la ligne et des informations prix hors taxe d'un produit et quantité commandée.

```
Type LigneFacture
    produitAchete As Produit
    quantite As Integer ' > 0
    prix_ht As Double   ' prix_ht = quantite * produitAchete.prix_ht
End Type
```

2.2 Définir une opération qui permet d'initialiser une ligne de facture à partir du produit concerné et de la quantité commandée. Le prix total hors taxe de la ligne peut être déduit de ces

deux informations puisqu'il correspond au prix hors taxe du produit multiplié par la quantité commandée.

Solution : L'initialisation d'une ligne de facture est réalisée sur le même principe que le produit.

```
Sub initialiser_ligne(ligne As LigneFacture, p As Produit, ByVal quantite As Integer)
'ROLE : Initialiser une ligne de facture a partir du produit concerne
'       et de la quantite commandee.
'ENTREE p, quantite
'SORTIE ligne
    ligne.produitAchete = p
    ligne.quantite = quantite
    ligne.prix_ht = p.prix_ht * quantite
End Sub
```

2.3 Écrire une opération qui affiche une ligne de facture. Les informations sont affichées de manière tabulée avec dans l'ordre le code du produit, son libellé, son prix hors taxe, la quantité commandé, le prix total hors taxe et le prix total TTC. On supposera le taux de TVA constant et indépendant des produits.

Voici un exemple de ligne affichée.

```
| 001 |       Truc |   99.00 | 10 |  990.00 | 1184.04 |
```

Solution : Ce sous-programme réalise un affichage sur l'écran. Il convient donc d'en faire une procédure. Notons qu'il n'a qu'un seul paramètre en entrée, la ligne de facture, ce qui nous aurait aussi conduit à en faire une procédure.

```
Sub afficher_ligne(ligne As LigneFacture)
'ROLE : Afficher une ligne de facture
'ENTREE ligne
    Dim prix_ttc As Double
   ' calculer le prix ttc
    prix_ttc = ligne.prix_ht * (1 + TVA)
   ' afficher la ligne
    Afficher ligne.produitAchete.code, ligne.produitAchete.libelle, ligne.produitAchet
             ligne.quantite, ligne.prix_ht, prix_ttc
End Sub
```

Exercice 3 : Notion de facture

Définissons enfin le type Facture. Une facture contient plusieurs lignes (20 au maximum). Elle est caractérisée par un numéro de facture.

3.1 Définir le type Facture.

Solution : Le type Facture est un tableau de lignes de facture. On ne sait pas à priori combien il y aura de lignes mais on sait qu'il y en a moins de 20. On peut donc prendre un tableau de capacité 20 et il faudra gérer la taille effective. Ces deux informations sont donc regroupées dans un enregistrement

```
Const NB_LIGNES As Integer = 20 ' nombre maximal de lignes sur une facture
Type Facture
    numero As Integer
    lignes(NB_LIGNES) As LigneFacture
    nb_lignes_eff   As Integer  ' nombre effectif de lignes sur la facture
```

```
End Type
```

3.2 Écrire une opération qui permet d'initialiser une facture à partir de son numéro. Une telle facture est bien entendu vierge et ne contient donc aucune ligne.

Solution : Ce sous-programme a un seul paramètre, la facture à initialiser. C'est un paramètre en sortie. On fait donc une procédure.

```
Sub initialiser_facture(une_facture As Facture, ByVal son_numero As Integer)
'ROLE : Initialiser une facture vierge (aucune ligne) en lui affectant le numero spéci
'ENTREE  son_numero
'SORTIE une_facture
    une_facture.numero = son_numero
    une_facture.nb_lignes_eff = 0
End Sub
```

3.3 Écrire un sous-programme qui permet de calculer le montant total hors taxe de la facture. Il s'agit de faire la somme des totaux hors taxe des lignes de la facture.

Solution : Ce sous-programme prend en paramètre une facture (entrée) et fournit le total hors taxe de la facture (sortie). Il y a donc un seul paramère en sortie, d'autres qui sont tous en entrée. On en fait donc une fonction.

```
Fonction montant_total_ht_facture(une_facture: Facture): Réel Est
    -- Le montant total hors taxe de la facture une_facture.
```

```
Function montant_total_ht_facture(une_facture As Facture) As Double
'ROLE : renvoie le montant total HT d'une facture
'ENTREE une_facture
    Dim l As Integer ' index pour parcourir les lignes de la facture
    Dim resultat As Double
    resultat = 0
    For l = 1 To une_facture.nb_lignes_eff
        resultat = resultat + une_facture.lignes(l).prix_ht
    Next l
    montant_total_ht_facture = resultat
End Function
```

Notons qu'en F, on a défini une variable locale `resultat` qui sert à accumuler le total des lignes.

3.4 Écrire un sous-programme pour ajouter une ligne à une facture, c'est-à-dire un produit et la quantité commandée.

Solution : Ce sous-programme prend en paramètre la facture à modifier (**in out**), le produit à ajouter (**in**) et la quantité commandée (**in**). On fait donc une procédure.

Notons que pour l'implantation de cette procédure, on peut utiliser l'opération `initialiser_ligne` déjà définie.

```
Sub ajouter_ligne(une_facture As Facture, p As Produit, ByVal quantite As Integer)
'ROLE : Ajoute une ligne a une facture
'HYPOTHESE : une_facture.nb_lignes < NB_LIGNES
'ENTREE p, quantite
'M.A J. une_facture
    une_facture.nb_lignes_eff = une_facture.nb_lignes_eff + 1
```

```
7       initialiser_ligne une_facture.lignes(une_facture.nb_lignes_eff), p, quantite
8   End Sub
```

3.5 Écrire un sous-programme qui permet d'éditer une facture. Il s'agit d'afficher un entête portant le numéro de la facture. Ensuite sont affichées toutes les lignes et enfin le total hors taxe et TTC de la facture.

Voici un exemple d'édition d'une facture.

```
1
2   Édition de la facture 1
3   ---------------------------------------------------------
4   | 001 |       Truc |   99.00 | 10 |  990.00 | 1184.04 |
5   | 020 |      Chose |   15.00 |  2 |   30.00 |   35.88 |
6   | 163 |     Bidule |  250.00 |  1 |  250.00 |  299.00 |
7   ---------------------------------------------------------
8                                     | 1270.00 | 1518.92 |
```

Solution : Ce sous-programme prend un seul paramètre, la facture à éditer (**in**). Son objectif est de faire un affichage sur l'écran (une sortie sur l'écran). Nous en faisons donc une procédure.

```
1   Sub editer_facture(une_facture As Facture)
2   'ROLE : Affiche a l'ecran la facture
3   'ENTREE une_facture
4       Dim l As Integer        ' index pour parcourir les lignes de la facture
5       Dim total_ht As Double  ' total ht de la facture
6       Dim total_ttc As Double ' total ttc de la facture
7
8      '1.Calculer les totaux
9       total_ht = montant_total_ht_facture(une_facture)
10      total_ttc = total_ht * (1 + TVA)
11
12     '2.Afficher l'entete
13      Afficher "Facture n° ", une_facture.numero
14      Afficher "CODE", "LIBELLE", "PRIX HT", "QUANTITE", "TOTAL HT", "TOTAL TTC"
15
16     '3.Afficher les lignes
17      For l = 1 To une_facture.nb_lignes_eff
18          afficher_ligne une_facture.lignes(l)
19      Next l
20     '4.Afficher les totaux
21      Afficher , , , "TOTAL", total_ht, total_ttc
22  End Sub
```

3.6 Écrire un sous-programme pour supprimer une ligne d'une facture en fonction de son numéro. Bien sûr, la première ligne porte le numéro 1.

Solution : Ce sous-programme a pour paramètres la facture à modifier (**in out**) et le numéro de la ligne à supprimer (**in**). On fait donc une procédure.

```
1   Sub supprimer_ligne(une_facture As Facture, ByVal n_ligne As Integer)
2   'ROLE : supprime de la facture spécifiée la ligne de numero numero n_ligne
3   'HYPOTHESE : n_ligne est compris entre 1 et une_facture.nb_lignes inclus
4   'M.A J. une_facture
```

```
'ENTREE n_ligne
    Dim l As Integer ' index pour parcourir les lignes a decaler
   ' Decaler les lignes
    For l = n_ligne To une_facture.nb_lignes_eff
        une_facture.lignes(l) = une_facture.lignes(l + 1)
    Next l
    une_facture.nb_lignes_eff = une_facture.nb_lignes_eff - 1
End Sub
```

Exercice 4 : Programme de test

Écrire un programme de test pour les types et opérations définis dans les exercices précédents.

Remarque : Même si c'est le dernier exercice, il est souhaitable (et nécessaire) de le faire en même temps que les autres, au fur et à mesure que les sous-programmes sont écrits. En effet, tester au fur et à mesure permet de tester de manière plus exhaustive (donc de détecter plus d'erreurs), de localiser plus vite la cause de l'erreur et donc de la corriger plus facilement.

Solution :

```
Sub TesterFacture()
'ROLE : tester l'utilisation d'une facture
'       Attention : ce programme de test n'est absolument pas complet '
    Dim f1 As Facture
    Dim p1 As Produit, p2 As Produit, p3 As Produit

    EffacerEcran "Test␣des␣procédures␣de␣gestion␣d'une␣facture"
   '1.Creation de 3 produits
    initialiser_produit p1, "001", "Truc", 99
    initialiser_produit p2, "020", "Chose", 15
    initialiser_produit p3, "163", "Bidule", 250

   '2.Creation de la facture en ajoutant les lignes
    initialiser_facture f1, 1
    editer_facture f1
    ajouter_ligne f1, p1, 10
    editer_facture f1
    ajouter_ligne f1, p2, 2
    editer_facture f1
    ajouter_ligne f1, p3, 1
    editer_facture f1

   '3.Suppression de la deuxieme ligne
    supprimer_ligne f1, 2
    editer_facture f1

End Sub
```