E.S.L.S.C.A.

Programmation VBA

Thème 3 :
La Programmation structurée (les variables, les structures et les tableaux)

Les variables

Lors d'une procédure, les variables servent à stocker toutes sortes de données (des valeurs numériques, du texte, des valeurs logiques, des dates ...). Elles peuvent également faire référence à un objet.
Suivant les données que la variable recevra, on lui affectera un type différent. Les différents types de variables de VB sont :

Type de données:	*Mot clé :*	Espace occupé	Plage de valeur
Octet	**Byte**	1 octet	Entier de 0 à 255
Logique	**Boolean**	2 octets	True ou False
Entier	**Integer**	2 octets	Entier de -32 768 à 32 768
Entier Long	**Long**	4 octets	Entier de -2 147 483 648 et 2 147 483 647 à 2 147 483 648 et 2 147 483 647
Décimal simple	**Single**	4 octets	-3,402823E38 à -1,401298E-45 pour les valeurs négatives 1,401298E-45 à 3,402823E38 pour les valeurs positives.
Décimal Double	**Double**	8 octets	-1,79769313486231E308 à -4,94065645841247E-324 pour les valeurs négatives 4,94065645841247E-324 et 1,79769313486231E308 pour les valeurs positives
Monétaire	**Currency**	8 octets	de -922 337 203 685 477,5808 et 922 337 203 685 477,5807
Date	**Date**	8 octets	1er Janvier 100 au 31 décembre 9999
Decimal	**Decimal**	12 octets	+/-79 228 162 514 264 337 593 543 950 335 sans point décimal +/-7,9228162514264337593543950335 avec 28 décimales.
Objet	**Object**	4 octets	toute référence à des objets
Chaîne de caractères à longueur variable	**String**	10 octets + longueur de chaîne	de 0 à 2 milliards de caractères
Chaîne de caractères à longueur fixe	**String**	Longueur de la chaîne	1 à 65 400 caractères
Variant avec chiffres	**Variant**	16 octets	Valeur numérique jusqu'au type double.
Variant avec caractères	**Variant**	22 octets + longueur de la chaîne	Même plage que pour un String de longueur variable
Défini par l'utilisateur	**Type**	Variable	Identique au type de données.

Pour rendre obligatoire la déclaration de variables, placez l'instruction "Option Explicit" sur la première ligne du module ou cochez l'option "Déclaration des variables obligatoires" dans le menu "Outils-Options" de l'éditeur de macros.
La déclaration explicite d'une variable se fait par le mot Dim (abréviation de Dimension). Le nombre maximum de caractères du nom de la variable est de 255. Il ne doit pas commencer par un chiffre et ne doit pas contenir d'espaces. La syntaxe est "Dim NomDeLaVariable as Type".

```
Sub Test()
    Dim SommeVal As Integer
    Dim Val1 As Integer
    Dim Val2 As Integer
    Val1 = 5
    Val2 = 2
    SommeVal = Val1 + Val2
    MsgBox Somme
End Sub
```

Vous pouvez également déclarer vos variables sur une même ligne :

```
Sub Test()
    Dim SommeVal As Integer, Val1 As Integer, Val2 As Integer
    Val1 = 5
    Val2 = 2
    SommeVal = Val1 + Val2
    MsgBox SommeVal
End Sub
```

La portée d'une variable est différente suivant l'endroit et la façon dont elle est déclarée.
Une variable déclarée à l'intérieur d'une procédure est dite "Locale". Elle peut être déclarée par les mots Dim, Static ou Private. Dès que la procédure est terminée, la variable n'est plus chargée en mémoire sauf si elle est déclarée par le mot Static. Une variable Locale est généralement placée juste après la déclaration de la procédure.

Une variable peut être "Locale au module" si celle-ci est déclarée avant la première procédure d'un module. Toutes les procédures du module peuvent alors lui faire appel. Elle est déclarée par les mots Dim ou Private.

```
Option Explicit
'Les variables Val1 et Val2 peuvent être utilisées dans toutes les
procédures du module
Dim Val1 As Integer, Val2 As Integer

Sub Test()
    Static SommeVal As Integer
    Val1 = 3
     Val2 = 5
    SommeVal = Val1 + Val2
```

```
    Msgbox SommeVal
End Sub

Sub Test2()
    Static DivisVal As Integer
    Val1 = 6
     Val2 = 2
    DivisVal = Val1 / Val2
    Msgbox DivisVal
End Sub
```

Un variable peut également être accessible à tous les modules d'un projet. On dit alors qu'elle est publique. Elle est déclarée par le mot Public. Elle ne peut pas être déclarée dans un module de Feuille ou dans un module de UserForm.

```
Option Explicit
'Les variables Val1 et Val2 peuvent être utilisées dans toutes les
procédures de tous les modules du projet.
Public As Val1, Integer, Val2 As Integer
```

Une variable peut garder toujours la même valeur lors de l'exécution d'un programme. Dans ce cas, elle est déclarée par les mots Const ou Public Const.

```
Option Explicit
'La variable Chemin gardera toujours la valeur.
Const Chemin as String = "c:\application\excel\"
```

Il est possible de définir une taille fixe pour une variable de type String par la syntaxe Dim Variable as String * Longueur ou Longueur correspond au nombre de caractère que prend la variable.

```
Option Explicit

Sub Test
    Dim Couleur as String * 5
    Couleur = "Rouge"
    ' Si Couleur était égal à "Orange", la variable Couleur aurait pris
    comme valeur "Orang".
End Sub
```

Il est important de déclarer ses variables par un nom explicite pour rendre le programme plus lisible. Vous pouvez également précéder ce nom par le caractère standard des types de variables. Par exemple, le caractère "i" représente un entier et la variable peut être nommée Dim iNombre as Integer.

Caractère :	*Type de variable* :
b	Boolean
i	Integer
l	long
s	Single
d	Double
c	Currency
dt	Date
obj	Object
str	String
v	Variant
u	Défini par l'utilisateur

Les variables peuvent également faire référence à des objets comme des cellules, des feuilles de calcul, des graphiques, des classeurs ... Elles sont déclarées de la même façon qu'une variable normale.

```
Option Explicit
Sub Test()
    'La variable MaCel fait référence à une plage de cellule
    Dim MaCel As Range
    'Le mot Set lui affecte la cellule "A1"
    Set MaCel = Range("A1")
    'La cellule "A1" prend comme valeur 10
    MaCel.Value = 10
End Sub
```

Les conditions.

Les conditions sont très courantes dans les applications VB. Elles peuvent déterminer la valeur que prennent les variables, arrêter une procédure, appeler une procédure, quitter une boucle, atteindre une étiquette.
Les exemples suivants vont déterminer la valeur que prendra la variable Mention par rapport à des notes. Le tableau des notes est :

Notes :	*Mention* :
0	Nul
1 à 5	Moyen
6 à 10	Passable
11 à 15	Bien
16 à 19	Très bien
20	Excellent

L'instruction la plus courante dans VB est la condition **If** *condition* **Then** *valeur vrai* :

```
'La Note se trouve dans la cellule "A1", la mention est à mettre dans
la cellule "B1"
'Pour trouver la valeur de la mention, on pourrait écrire :
Dim Note As Integer
Dim Mention As String
Note = Range("A1")
If Note = 0 Then Mention = "Nul"
If Note >= 1 And Note <6 Then Mention = "Moyen"
If Note >= 6 And Note <11 Then Mention = "Passable"
If Note >= 11 And Note <16 Then Mention = "Bien"
If Note >= 16 And Note <20 Then Mention = "Très Bien"
If Note = 20 Then Mention = "Excellent"
Range("B1") = Mention
```

Si la valeur vraie possède plusieurs lignes d'instructions, la syntaxe devient **If** *Condition* **Then** *Valeur vraie* **End If**.

```
'Si la note est égale à 0, la mention prend comme valeur "Nul" et la
couleur de la police devient Rouge:
Dim Note As Integer
Dim Mention As String
Note = Range("A1")
If Note = 0 Then
    Mention = "Nul"
    Range("B1").Font.Color = RGB(255, 0, 0)
End If
Range("B1") = Mention
```

Dans notre exemple, l'instruction peut être mieux structurée. La couleur de la police de la mention est rouge si la note est inférieure à 10 et verte si la note est supérieure à 10 en utilisant la syntaxe **If** *condition* **Then** *valeur vrai* **Else** *valeur fausse* **End If**.

```
If Note < 10 Then
    Range("B1").Font.Color = RGB(255, 0, 0)
Else
    Range("B1").Font.Color = RGB(0, 255, 0)
End If
```

Pour calculer la valeur de la mention, on utilisera plus facilement la syntaxe **If** *condition* **Then** *valeur vraie* **ElseIf** *condition* **Then** *valeur vrai* **Else** *valeur fausse* **End If** en ajoutant autant de fois que nécessaire l'instruction **ElseIf.**

```
Dim Note As Integer
Dim Mention As String
Note = Range("A1")
If Note = 0 Then
    Mention = "Nul"
ElseIf Note >= 1 And Note <6 Then
    Mention = "Moyen"
ElseIf Note >= 6 And Note <11 Then
    Mention = "Passable"
ElseIf Note >= 11 And Note <16 Then
    Mention = "Bien"
ElseIf Note >= 16 And Note <20 Then
    Mention = "Très Bien"
Else
    Mention = "Excellent"
End If
Range("B1") = Mention
```

Dans le cas de conditions multiples, comme dans notre exemple, on préférera le bloc d'instruction **Select Case** *expression* **Case** *valeur expression* **Case Else End Select**.

```
Dim Note As Integer
Dim Mention As String
Note = Range("A1")
Select Case Note
    Case 0
        Mention = "Nul"
    Case 1 To 5
        Mention = "Moyen"
    Case 6 To 10
        Mention = "Passable"
    Case 11 To 15
        Mention = "Bien"
    Case 16 To 19
        Mention = "Très Bien"
    Case Else
        Mention = "Excellent"
End Select
Range("B1") = Mention
```

Une condition peut appeler une étiquette. Une étiquette représente un endroit de la procédure. Elle se déclare par un nom suivi du signe ":".

Reproduire l'exemple suivant : rentrer des instructions et faire en sorte que si la variable i prend la valeur 10 , la procédure va directement à la ligne Msgbox "Fin du programme".

```
    Dim i As Integer
    I= inputbox (“donner la valeur de I”)
    If  i = 10 Then GoTo Fin
    Msgbox "la valeur donnée est différente de 10"
    Msgbox "Le Goto n’a pas donc opéré puisque je m’affiche”
Fin:
    Msgbox "Fin du programme"
```

Les boucles (structures itératives).

Les boucles le plus souvent utilisés sont les boucles **For** ... **Next.** Elles permettent de répéter un nombre de fois défini un bloc d'instructions. Elles utilisent une variable qui est incrémentée ou décrémentée à chaque répétition

```
Dim i As Integer
'La boucle suivante va écrire les chiffres de 1 à 10 'dans la plage de
cellule "A1:A10". La variable i 's'incrémente de 1 à chaque fois
For i  = 1 To 10
    Range("A1").Offset(i - 1) = i
Next i
```

La variable peut être incrémentée d'une valeur différente de 1 par le mot **Step**.

```
Dim i As Integer, j As Integer
'La boucle suivante va écrire les chiffres pairs
'dans la plage de cellule "A1:A10". La variable i 's'incrémente de 2 à
chaque fois
j = 0
For i  = 2 To 20 Step 2
    Range("A1").Offset(j) = i
    j = j + 1
Next i
```

La variable peut également être décrémentée. Dans ce cas, le mot **Step** est obligatoire.

```
Dim i As Integer, j As Integer
'La boucle suivante va écrire les chiffres de 20 à 10
'dans la plage de cellule "A1:A10". La variable i
'se décrémente de 1 à chaque fois
j = 0
For i  = 20 To 10 Step -1
    Range("A1").Offset(j) = i
    j = j + 1
Next i
```

A l'intérieur d'un bloc d'instruction **For Next**, l'instruction **Exit For**, peut quitter la boucle avant que la variable n'est atteint sa dernière valeur. Dans le tableau suivant se trouve une liste d'élèves avec leurs notes.

	A	B
1	ELEVE	NOTE
2	PIERRE	5
3	JACQUES	15
4	JEAN	10
5	VALERIE	12
6	PAUL	18
7	DELPHINE	13
8	ANTOINE	0
9	JEANNE	6
10	ANDRE	19
11	JOCELYNE	8
12	FELIX	12
13	JUSTIN	15
14	ETIENNE	6
15	MARIE	5

Pour connaître la note de Paul, on pourrait utiliser :

```
Dim i As Integer
Dim NbreEleve As Integer, NoteEleve As integer
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
'La dernière ligne - 1 correspond au nombre d'élèves
NbreEleve = Cel.End(Xldown).Row - 1
For i  = 1 To NbreEleve
    If Cel.Offset(i) = "PAUL" Then
        'On récupère la note
        NoteEleve = Cel.Offset(i, 1)
        'puis on sort de la boucle
        Exit For
    End If
Next i
Msgbox "La note de Paul est " & NoteEleve
```

Pour répéter un bloc d'instructions pour chaque objet d'une collection ou pour chaque élément d'un tableau, on utilisera le bloc d'instruction **For Each** *Objet* **In** *Collection* **Next.**
L'exemple suivant mettra la police de couleur rouge si les notes sont inférieures à 10 et de couleur verte si les notes sont supérieures à10.

```
Dim Cel As Range, Cel2 As Range
'On affecte la plage de cellules "B2:B15"
'à la variable Cel
Set Cel = Range("B2:B15")
'Pour chaque cellule de la plage de cellule
For Each Cel2 In Cel
    If Cel2 < 10 Then
        'Police de couleur rouge
        Cel2.Font.Color = RGB(255, 0, 0)
    Else
        'Police de couleur verte
        Cel2.Font.Color = RGB(0, 255, 0)
    End If
Next
```

On peut également utiliser l'instruction **Exit For** pour sortir d'un bloc d'instruction **For Each ... Next**.

Les boucles conditionnelles:
Les boucles **While** *condition* **Wend** exécutent un bloc d'instruction tout pendant que la condition est vraie.

```
Dim Calcul As Integer, Compteur As Integer
Compteur = 1
'Le bloc d'instruction suivant va additionner les
' nombres de 1 à 10 (1+2+3+4+5+6+7+8+9+10).
'Tant que la valeur de Compteur est inférieur 11
While Compteur < 11
    Calcul = Calcul + Compteur
    'Ne pas oublier d'incrémenter le compteur sinon
    'la boucle ne pourra pas s'arrêter.
    Compteur = Compteur + 1
Wend
Msgbox Calcul
```

Les boucles **Do Loop** sont mieux structurées que les boucles **While Wend**. On peut à tout moment sortir d'une boucle **Do Loop** par l'instruction **Exit Do**.
La boucle **Do While** *condition* **Loop** exécute un bloc d'instruction tout pendant que la condition est vraie. Dans l'exemple suivant, on veut ajouter l'élève Annie à la liste des élèves.

```
Dim Compteur As Integer
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
Compteur = 1
'Le bloc d'instruction suivant va se répéter
'tant que la cellule n'est pas vide
Do While Cel.Offset(Compteur) <> ""
    'Ne pas oublier d'incrémenter le compteur sinon
    'la boucle ne pourra pas s'arrêter.
    Compteur = Compteur + 1
Loop
Cel.Offset(Compteur) = "ANNIE"
```

Dans l'exemple précédent, la condition est testée à l'entrée de la boucle. Dans la boucle **Do Loop While** *condition*, le bloc d'instruction est exécuté une fois avant que la condition soit testée.

```
Dim Compteur As Integer
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
Compteur = 1
'Le bloc d'instruction suivant va se répéter
'tant que la cellule n'est pas vide
Do
    'Ne pas oublier d'incrémenter le compteur sinon
    'la boucle ne pourra pas s'arrêter.
    Compteur = Compteur + 1
Loop While Cel.Offset(Compteur) <> ""
Cel.Offset(Compteur) = "ANNIE"
```

Pour sortir d'une boucle, on utilise l'instruction **Exit Do**. Pour recherche la note de André, on pourrait utiliser :

```
Dim Compteur As Integer, NoteEleve As integer
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
Compteur = 1
'Le bloc d'instruction suivant va se répéter
'tant que la cellule n'est pas vide
Do While Cel.Offset(Compteur) <> ""
    'Si la valeur de la cellule est "ANDRE", on sort
    'de la boucle
    If Cel.Offset(Compteur) = "ANDRE" Then
        Exit Do
    End If
    'Ne pas oublier d'incrémenter le compteur sinon
    'la boucle ne pourra pas s'arrêter.
    Compteur = Compteur + 1
Loop
NoteEleve = Cel.Offset(Compteur, 1)
Msgbox "La note de André est " & NoteEleve
```

Les boucles **Do Until** sont identiques aux boucles **Do While**, seulement le bloc d'instruction est répété tout pendant que la condition n'est pas vraie. La syntaxe est exactement la même, il y a juste le mot **Until** qui remplace le mot **While.** Si on reprend l'exemple précédent, la procédure deviendrait :

```
Dim Compteur As Integer, NoteEleve As integer
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
Compteur = 1
'Le bloc d'instruction suivant va se répéter
'tant que la cellule n'est pas vide
Do Until Cel.Offset(Compteur) = "ANDRE"
    'Ne pas oublier d'incrémenter le compteur sinon
    'la boucle ne pourra pas s'arrêter.
    Compteur = Compteur + 1
Loop
NoteEleve = Cel.Offset(Compteur, 1)
Msgbox "La note de André est " & NoteEleve
```

Les tableaux.

Contrairement aux variables classiques qui contiennent une seule valeur, un tableau est une variable qui peut contenir un ensemble de valeurs de même type. Prenons comme exemple notre liste d'élève :

	A	B
1	ELEVE	NOTE
2	PIERRE	5
3	JACQUES	15
4	JEAN	10
5	VALERIE	12
6	PAUL	18
7	DELPHINE	13
8	ANTOINE	0
9	JEANNE	6
10	ANDRE	19
11	JOCELYNE	8
12	FELIX	12
13	JUSTIN	15
14	ETIENNE	6
15	MARIE	5

Pour mettre en mémoire le nom de tous les élèves, déclarons un tableau plutôt que de déclarer autant de variables que d'élèves. La syntaxe pour déclarer un tableau est "**Dim** *Variable*(Nbre éléments) **As** *Type*"

```
    'Déclaration du tableau
    Dim MesEleves(14) As String
    Dim i As Integer
    Dim Cel As Range
    'On affecte la cellule "A1" à la variable Cel
    Set Cel = Range("A1")
  'Boucle pour remplir le tableau
    For i  = 1 To 14
        MesEleves(i) = Cel.Offset(i)
    Next i
```

Dans cet exemple, la variable MesEleves(1) est égale à "PIERRE", MesEleves(2) à "JACQUES",... , MesEleves(15) à "MARIE".

Par défaut, la valeur de l'index inférieur d'un tableau est 1.Pour la changer, on va utiliser le mot **To**. L'exemple suivant va créer un tableau du 5ème au 10ème élève :

```
'Déclaration du tableau
Dim MesEleves(5 To 10) As String
Dim i As Integer
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
'Boucle pour remplir le tableau
For i  = 5 To 10
    MesEleves(i) = Cel.Offset(i)
Next i
```

Vous pouvez créer des tableaux contenant des éléments de types différents de deux façons. La première consiste à déclarer la variable de type variant :

```
'Déclaration du tableau
Dim CeJour(4) As Variant
CeJour(1) = 15
CeJour(2) = "Septembre"
CeJour(3) = 2003
CeJour(4) = "Roland"
```

La seconde utilise la fonction **Array**.

```
'Déclaration du tableau
Dim CeJour As Variant
CeJour = Array(15, "Septembre", 2003, "Roland")
```

Dans le 1er cas, CeJour(1) contient 15, CeJour(2) contient "Septembre", CeJour(3) contient 2003 et CeJour(4) contient "Roland".
La valeur du premier index de la fonction Array est 0, donc, dans le second cas, CeJour(0) = 15, CeJour(1) = "Septembre", CeJour(2) = 2003 et CeJour(3) = "Roland".

Vous pouvez créer des tableaux à plusieurs dimensions. Pour mettre en mémoire le nom des élèves avec leurs notes, nous allons créer un tableau à 2 dimensions.

```
'Déclaration du tableau
'14 représente le nombre d'enregistrements a traiter, 2 le nombre de
'champs (Elèves, Notes).
Dim MesEleves(1 To 14, 1 To 2) As Variant
Dim i As Integer
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
'Boucle pour remplir le tableau
For i  = 1 To 14
    MesEleves(i, 1) = Cel.Offset(i) 'Elèves
    MesEleves(i, 2) = Cel.Offset(i, 1) 'Notes
Next i
'Ou alors :
Dim j As Integer
For i  = 1 To 14
    For j  = 1 To 2
        MesEleves(i, j) = Cel.Offset(i, j - 1)
    Next j
Next i
```

Dans cet exemple, MesEleves(5, 1) contient "PAUL" et MesEleves(5, 2) la note 18. Notez que la variable a été déclarée de type Variant étant donné qu'elle recevait des données de type String et des données de type Integer. Si elle était déclarée de type String, les notes seraient en mode texte.

Il est possible de redimensionner un tableau par le mot **Redim**. En effet, le nombre d'éléments ou de dimensions que doit contenir un tableau n'est pas toujours connu. Pour conserver les éléments d'un tableau redimensionné, utilisez le mot **Preserve**.

Dans l'exemple suivant, le tableau va recevoir le nom des élèves dont la note est supérieure ou égale à 10.

```
'Déclaration du tableau
Dim MesEleves() As String
Dim i As Integer
Dim j As Integer 'Nbre éléments du tableau
Dim Cel As Range
'On affecte la cellule "A1" à la variable Cel
Set Cel = Range("A1")
'Boucle pour remplir le tableau
For i  = 1 To 14
    If Cel.Offset(i, 1) >= 10 Then 'Si la note >=10
        j = j + 1
        'Redimension du tableau en conservant
        'ses éléments
        ReDim Preserve MesEleves(j)
        MesEleves(j) = Cel.Offset(i)
    End If
Next i
```

Le tableau contient 8 éléments et, par exemple, la valeur de MesEleves(5) est "DELPHINE".

Exercices :
1. Ecrire le programme VBA qui doit permettre de saisir les coordonnées (Nom, Prénom et Tél) dans un tableau mémoire de 10 éléments. La structure de type de données personnalisé sera utilisée. Une fois rempli, le tableau sera ensuite copié sur une feuille que vous sélectionnerez.

2. Vous travaillez dans une agence immobilière. Vous disposez de la base de données suivante concernant des villas de haut standing.

Villa	*Ville*	*Nbre de pièces*	*Prix*
Flairsou	St Tropez	10	10 000 000 €
Picsou	St Barth	14	19 000 000 €
Tirsou	St Jacques	7	13 000 000 €
Rentresou	St Simpson	6	14 000 000 €

Des personnes intéressées par l'achat d'une des villas viennent vous voir. Ces personnes ne disposent pas des fonds nécessaires à un tel achat. Ils vous demandent donc, à chaque fois, de leur faire des simulations afin de voir les mensualités auxquelles elles seront soumises. Devant la répétition de ces demandes, vous décidez de créer un modèle pour gérer ce type de calcul en respectant les contraintes suivantes :

- La personne **saisit** le nom de la villa qui l'intéresse, le système affiche automatiquement la ville, le nombre de pièces et le prix de la villa, en s'aidant de la base ci-dessus.
- Le taux d'intérêt annuel varie de 1% à 20% par variation de 0,10%.
- Le nombre d'annuités est fixé à 12.
- Aucun apport initial.

Une fois obtenu l'ensemble de ces infos, le système calcule le montant de la mensualité (il s'agit de mensualités constantes).
Il vous est demandé de faire le modèle générique répondant à ce cahier de charge.

3. On se propose de réaliser un formulaire VBA répondant au besoin suivant :
Afin d'offrir un plus grand choix à sa clientèle, l'agence «Tour Operator » propose l'organisation d'un voyage touristique. L'agence propose 3 forfaits s'appliquant à la même destination :

- **Forfait 1 : Achat billet : 1000 euros**
- **Forfait 2 : Achat billet + Logement inclus : 1500 euros**
- **Forfait 3 : Achat billet + Logement + Visite guidée : 2200 euros.**

Le formulaire doit renseigner, outre les trois forfaits disponibles, des informations sur le client : Nom, Prénom, N° Tél., Adresse, Destination, Type de forfait (numéroter de 1 à 3), Mode de paiement (Chèque, CB ou cash), quantité de forfait.
Une fois ces informations saisies, le prix total de l'ensemble des forfaits pris par le client doit être affiché automatiquement sur le formulaire.

Proposer un formulaire répondant au besoin de cette agence.

4. Un magasin de sport voudrait automatiser l'émission de ses devis et de ses factures clients. Pour cela, une partie de son stock a été saisie dans le tableau 1 sous forme de base de données.
Le tableau 1 contient la désignation de l'article, le code, le prix unitaire et la quantité dans le stock.
La disposition et l'ordre des champs (colonnes) peuvent être modifiés selon les préférences du programmeur.
Le format du devis est donné par le tableau 2.
Il suffit de saisir le ***code*** de l'article pour que ***la désignation*** et le ***prix unitaire*** s'affichent.
La **quantité** est saisie directement à partir du clavier. L'achat de 10 articles ou plus permet une réduction de 25 % sur le prix unitaire de l'article.
La saisie d'un devis doit être interactive avec la quantité disponible dans le stock.

Il vous est demandé de donner le mode opératoire et d'établir les formules dans les cellules spécifiques du tableau 2 afin d'automatiser l'impression des devis.

5. Que font les procédures suivantes :

```
Sub toto1()
    Dim Cel As Range, Cel2 As Range
    Set Cel = Range("B2:B15")
    For Each Cel2 In Cel
        If Cel2 < 10 Then
           Cel2.Font.Color = RGB(255, 0, 0)
        Else
           Cel2.Font.Color = RGB(0, 255, 0)
        End If
    Next
End Sub ()
```

```
Sub toto2()
    Dim i As Integer
    Dim Liste(1 To 14, 1 To 2) As String
    For i = 1 To 14
        Liste(i, 1) = Range("A1").Offset(i)
        Liste(i, 2) = Range("B1").Offset(i)
    Next i
    MaBoite.ListBox1.List = Liste
End Sub
```

```
Sub toto3()
    Dim F As Worksheet
    Dim i As Integer
    For Each F In Worksheets
        Range("A1").Offset(i) = F.Name
        i = i + 1
    Next
End Sub
```

```
Sub toto4()
    Dim i As Integer
    i = 1
    Do While Range("A1").Offset(i) <> ""
        Rows(i + 1).Insert
        i = i + 2
    Loop
```

```
End Sub
```

```
Sub toto5()
    Dim i As Integer
    Dim Cel As Range
    Set Cel = Range("A1")
    With UserForm1
        For i = 0 To Cel.End(xlDown).Row - 1
            .Controls("Label" & i + 1) = Cel.Offset(i)
        Next i
        .Show
    End With
End Sub
```