



Programmation en VBA



Présentation de Visual Basic

- ▶ **Visual Basic :**
 - ▶ Basic : dérivé du langage Basic (Beginner's All purpose Symbolic Instruction Code) des années 60
 - ▶ Visual : dessin et aperçu de l'interface avant l'exécution

Introduction

- ▶ Apprendre à programmer
- ≠
- ▶ Apprendre un langage de programmation
- ▶ → écrire un algorithme avant d'écrire le programme correspondant

Introduction

▶ Définitions :

- ▶ Un algorithme est la description de la suite des opérations élémentaires ordonnées capables de résoudre le problème posé.
- ▶ Un programme est la traduction d'un algorithme en un langage compréhensible par l'ordinateur.

Exemple

- ▶ Un commerçant veut calculer les montants de ses factures.
- ▶ Comment décrire de manière la plus élémentaire possible, la suite d'instructions à exécuter pour résoudre le problème.

Exemple

- ▶ Question 1 : Quelles sont les variables que nous devons manipuler ?
 - ▶ Les prix des produits;
 - ▶ les quantités des produits;
 - ▶ les montants des différentes lignes;
 - ▶ le total des montants.

Exemple

- ▶ **Question 2 : Comment manipuler ces variables pour résoudre le problème ?**
 - ▶ Les montants sont calculés en multipliant les prix unitaires par les quantités.
 - ▶ Le total hors taxe est calculé en effectuant la somme des montants.

Notion de variable

- ▶ En programmation, une variable est un nom qui sert à repérer un emplacement donné de la mémoire centrale.
- ▶ Les variables permettent de manipuler les valeurs sans avoir à se préoccuper de l'emplacement qu'elles occupent effectivement en mémoire.
- ▶ Choix des noms des variables : doit être le plus parlant possible.
- ▶ Exemple : PrixUnitaire, Quantite, Montant, c'est mieux que d'écrire x,y,z.



Programmation

Introduction

- ▶ Un programme VBA s'écrit à l'intérieur de l'instruction
Sub NomProgramme()

End Sub

- ▶ Les instructions à exécuter devront alors être incluses dans le bloc sub ... end sub

Introduction

- ▶ Dans l'éditeur de programmes, on peut saisir le code de plusieurs programmes, chacun dans un bloc sub ... end sub.
- ▶ Si on veut exécuter un des programmes, on place le curseur à l'intérieur du bloc et on clique sur exécuter.

Variables et constantes

- ▶ Une variable = {identificateur, type, valeur}
 - ▶ Identificateur : nom par lequel la variable est manipulée dans le programme
 - ▶ Type : type des valeurs possibles que la variable peut contenir
 - ▶ Valeur : valeur stockée dans la variable

Identificateur d'une variable

- ▶ Elle doit commencer par un caractère alphabétique et ne pas comporter les caractères suivants : . % , + - * ! # @ \$
- ▶ Elle ne peut pas excéder 255 caractères.
- ▶ Pour faciliter la lisibilité des programmes, on s'efforcera d'utiliser des minuscules pour les variables utilisées par le programme.

Déclaration des variables

- ▶ Par défaut, il n'est pas nécessaire de déclarer les variables utilisées. On peut donc utiliser n'importe quelle variable sans se préoccuper de sa déclaration préalable
- ▶ Les variables sont alors de type Variant
- ▶ ☹ pas d'optimisation
- ▶ 😊 déclarer explicitement les variables
 - ▶ Syntaxe Dim identificateur As Type

Les types de données

- ▶ VBA autorise la création de la plupart des types de données classiques :
 - ▶ *Integer* : entier court ;
 - ▶ *Long* : entier long ;
 - ▶ *Single* : réel simple ;
 - ▶ *Double* : réel en double précision ;
 - ▶ *Currency* : nombre monétaire ;
 - ▶ *Date* : date et heure ;
 - ▶ *String* : chaîne de caractères ;
 - ▶ *Boolean* : booléen (*True* ou *False*) ;
 - ▶ *Object* : référence quelconque à un objet ;
 - ▶ *Variant* : type particulier pouvant être n'importe quel autre type.
- ▶ Outre ces types élémentaires, il est également possible de créer des tableaux et des types personnalisés (voir plus loin).

Conversion de types

- ▶ Il est parfois nécessaire d'opérer des conversions de types.
- ▶ VBA permet d'effectuer ces opérations grâce à une série d'instructions, dont voici quelques exemples :
 - ▶ *CInt* : conversion en type de données entier ;
 - ▶ *CStr* : conversion en type de données *string* ;
 - ▶ *CBdI* : conversion en type de données *double* ;
 - ▶ *CDate* : conversion en type de données *date* ;
 - ▶ *CVar* : conversion en type de données *variant*.



Structures de contrôle



Structures conditionnelles

▶ **If ... Then ...Else ... Elseif**

- ▶ permet d'écrire une structure conditionnelle
- ▶ syntaxe :

```
If <condition> then
    <instructions>
Else
    <instructions>
End If
```

- ▶ Dans le cas où il est nécessaire d'imbriquer plusieurs structures conditionnelles, on peut utiliser l'instruction Elseif :

```
If <condition> Then
    <instructions>
Elseif <condition> Then
    <instructions>
End If
```

Structures conditionnelles (2)

▶ **Select...case**

- ▶ Permet d'écrire une structure conditionnelle dans laquelle une expression doit être comparée à plusieurs valeurs.
- ▶ La syntaxe est la suivante :

```
Select Case <variable>
  Case valeur1:
    <instructions>
  Case valeur2:
    <instructions>
  ...
  Case Else:
    <instructions>
End Select
```

- ▶ Lorsque la variable est égale à une valeur répertoriée, les instructions correspondantes sont exécutées, et l'instruction *Select Case* se termine. La ligne *Case Else* permet d'inclure toutes les occurrences de la variable non répertoriées auparavant. Elle est facultative.

Boucles

▶ **While Wend**

- ▶ permet de réaliser une boucle conditionnelle.
- ▶ La syntaxe est la suivante :

While <condition>

<instructions>

Wend

- ▶ La condition est évaluée au début du traitement. Si elle est vraie, les instructions sont exécutées. Ensuite la condition est réévaluée, et ainsi de suite.

Boucles (2)

▶ **Do Until Loop**

- ▶ similaire à *Do While Loop*. Cependant, les instructions contenues dans le corps de l'instruction sont d'abord exécutées, à la suite de quoi la condition est évaluée.

- ▶ La syntaxe est donc la suivante :

Do

<instructions>

Loop Until <condition>

Boucles (3)

▶ **For Step Next**

- ▶ utilisée pour répéter une action selon un nombre d'itérations déterminé.

- ▶ La syntaxe est la suivante :

For <variable> = valeur1 To valeur2 Step <pas>

<instructions>

Next <variable>

- ▶ *Step* sert à indiquer le pas d'itération. Cette précision est toutefois facultative : par défaut, la valeur du pas utilisé est 1.

Boucles (4)

▶ **For Each In Next**

- ▶ est une variante de l'instruction *For Step Next*.
- ▶ permet de réaliser une itération en parcourant tous les objets contenus dans une collection.
- ▶ La syntaxe est la suivante :

```
For Each <objet> In <collection>  
    <instructions>  
Next <object>
```

Boucles (5)

▶ **Exemple :**

- ▶ Le code suivant permet de renommer toutes les feuilles de calcul du classeur actif :

```
Dim i As Integer
```

```
Dim feuille As Worksheet
```

```
...
```

```
i = 0
```

```
For Each feuille In ActiveWorkbook.Sheets
```

```
    i = i + 1
```

```
    'Cstr (i) permet de convertir l'entier i en une chaîne de caractères
```

```
    feuille.Name = "Compte rendu Client n°" + CStr(i)
```

```
Next feuille
```

- ▶ Remarque : la ligne de commentaire introduite par une apostrophe.

Arrêt du programme

▶ **Exit**

- ▶ permet de sortir de la séquence en cours.
- ▶ Exemple : pour terminer prématurément une procédure, on écrira :

```
Sub essai()
```

```
    ...
```

```
    If <condition> Then
```

```
        Exit Sub
```

```
    End If
```

```
    ...
```

```
End Sub
```

- ▶ De même, les commandes *Exit Do* et *Exit Next* permettent de quitter une structure conditionnelle et une boucle *For Next*.

Interaction entre VBA et EXCEL

- ▶ Première solution :
 - ▶ La feuille est un objet.
 - ▶ Constituée de plusieurs cellules.
 - ▶ → `Feuil1.Cells(i,j)` désigne la cellule de coordonnées (i,j) dans la feuille de nom Feuil1.
 - ▶ Tous les objets Excel (classeur, feuilles, fonctions, ...) correspondent à des objets (au sens programmation du terme) et peuvent être manipulés dans VBA (à voir dans le chapitre suivant).



Procédures et fonctions



Procédures

- ▶ Une procédure est un ensemble d'instructions qui participent à une même tâche.
- ▶ Syntaxe :
[Public/Private][Static] Sub nom([liste_arguments])
 déclarations
 instruction
End Sub
- ▶ Exemple de procédure événementielle :
Private Sub cmdQuitter_Click()
 instructions
End Sub

Procédures : exemple

- ▶ Exemple de procédure publique :

```
Public Sub SaisieNote()
```

```
    Do
```

```
        Note = InputBox("Tapez une note")
```

```
        `demande la note à l'utilisateur
```

```
        Loop Until Note >= 0 And Note <= 20
```

```
End Sub
```

- ▶ Pour utiliser cette procédure, il suffira de :

- ▶ l'appeler par son nom : SaisieNote

- ▶ utiliser l'instruction Call :

```
...
```

```
Call SaisieNote `appelle la procédure de saisie
```

```
...
```

Procédures : exemple

- ▶ Les arguments/paramètres : une procédure peut prendre des arguments (utiliser des données en entrée)
- ▶ Déclaration dans les parenthèses:
Private Sub nom (nom1 As type1, ...)
 Instructions
End sub
- ▶ À l'appel :
 Call nom(expr_du_bon_type,...)
- ▶ Dans le code de la procédure, nom1 à la valeur de expr_du_bon_type

Procédures : exemple

- ▶ Exemple de passage de paramètre :
Private Sub Affiche(Message As String)
 MsgBox(Message)

End Sub

- ▶ À l'appel :

...

Dim news As String

news=« cool ! »

Affiche(«ift-20403»)

Affiche (news)

Affiche news

Call Affiche(«news»)

Call Affiche news

...

‘ affiche ift-20403

‘ affiche cool !

‘ affiche cool !

‘ affiche news

‘erreur compilation

Appel / invocation de procédure

- ▶ **Différents types d'appel :**
 - ▶ Call Nom-proc(argument1, ...)
 - ▶ Nom-proc(arg1, ...) ou Nom-proc arg1,...
 - ▶ Variable = NomFonct(argument1, ...)

- ▶ **Deux types de passages de paramètre à l'appel :**
 - ▶ Passage par référence
 - ▶ Passage par valeur
 - ▶ Distinction importante, commune à une majorité de langage de programmation haut niveau...

Passage par référence

- ▶ **C'est le mode par défaut :**
 - ▶ C'est la variable qui est passée.
 - ▶ Le nom de l'argument n'est alors qu'un alias (pseudonyme), c'est la variable qui est manipulée.
 - ▶ Toute affectation sur le paramètre modifie la variable.
 - ▶ Côté déclaration : rien ou ByRef.
 - ▶ Côté appel : normal
 - ▶ Nom(argument1, ...) ou Nom argument 1,...
 - ▶ Call Nom(argument1, ...)
 - ▶ Call Nom(argument) ou Nom argument si un seul argument

Passage par valeur

- ▶ Seule la valeur de l'argument est passée (pas la variable).
- ▶ Donc seul le paramètre local peut être modifié, pas la variable
 - ▶ Côté déclaration : `Byval`
 - ▶ Côté appel : `nom((argument), ...)`
 - ▶ On ajoute des parenthèses au paramètre passé.

Exemple

```
Private sub CallVal (Byval y As integer)
```

```
    y=y+1
```

```
End Sub
```

```
Private sub CallRef (y As integer)
```

```
    y=y+1
```

```
End Sub
```

Appels :

```
Dim num As integer
```

```
num = 2
```

```
CallVal (num)
```

‘après, num vaut toujours 2

```
CallRef num
```

‘après, num vaut 3

```
CallRef (num)
```

‘après, num vaut toujours 3

```
CallRef ((num))
```

‘après, num vaut toujours 3

Fonctions : un premier exemple

- ▶ Rajouter une fonction à l'ensemble des fonctions Excel :
 - ▶ Une fonction qui prend en arguments deux cellules et qui calcule la plus grande des deux valeurs.
 - ▶ Une fonction qui prend en arguments trois cellules et qui calcule la plus grande des trois valeurs.

Fonctions

- ▶ Une fonction encapsule aussi un ensemble d'instructions, mais retourne une valeur (désignée par le nom même de la fonction).
- Cette valeur doit être affectée au nom de la fonction avant la fin du bloc d'instructions.
- Syntaxe :
Function nom(arg1 As type, ...) As Type
 Instructions
 nom = exp_du_bon_type
 Instructions
End Function
- ▶ Il faut préciser le type de la valeur retournée.

Fonctions : exemple

- ▶ Exemple de fonction :

```
Public Function Carré(x As Single) As Single
```

```
    Carré = x * x
```

```
End Function
```

- ▶ Exemples d'utilisation :

- ▶ X=Carré(7) ' affecte 49 à x

- ▶ X=Carré(Carré(7)) ' affecte 2401 à x

- ▶ X=Carré(2+Carré(7)) ' affecte 2601 à x

Optional : exemple

- ▶ Le mot clé `Optional` permet de définir des arguments optionnels :
 - De type `Variant` (automatique)
 - Tous les arguments suivants seront optionnels
 - La fonction `IsMissing()` permet de tester si l'argument a été donné ou non
 - Exemple :

```
Function CalculTVA(MontantHT As Single, Optional Taux) As single
    If IsMissing(Taux) Then Taux=20.6
    End If
    CalculTVA = MontantHT * Taux/100
End Function
```

Appels: `MontantTTC = MontantHT+CalculTVA 100`

Ou : `MontantTTC = MontantHT+CalculTVA 100, 5.5`

Structures de données

Les tableaux

Les tableaux

- ▶ Il arrive fréquemment que l'on doive manipuler une série de données de même nature :
 - des résultats financiers mensuels
 - des vecteurs d'indices
 - des matrices
 - des températures
 - ...
- ▶ Dans ce cas, on peut définir et manipuler des tableaux d'une ou plusieurs dimensions (jusqu'à 60 en VB)

Déclaration des tableaux

Syntaxe des déclarations de tableaux :

- Variable : `Dim <nom> As Type`
- Constantes : `Const <nom> As Type`
- Tableaux de variables :
`Dim <nom>(L1 To U1, ...) As Type`
- Variante :
`Dim <nom>(N1, ..., Nn) As Type`

Exemples

Dim Mois(1 To 12) As String

Mois(1)=« janvier »

Mois(2)=« février »

...

Dim Jours(7) As String

Jours(0)=« lundi »

Jours(1)=« mardi »

...

txtJour.Text = Jours(1)

‘affiche mardi dans la zone de texte txtJour

‘déclaration

‘initialisation

‘déclaration

‘initialisation

Passage des tableaux en paramètre

- ▶ Pour passer un tableau, il faut :
 - ▶ Call `nomproc(nomTab())`
 - ▶ `Nomproc nomTab()`
 - ▶ `Nomproc nomTab(6)`
 - ▶ `Nomproc (nomTab(6))`

Algorithmes sur les tableaux

- ▶ Saisie des éléments d'un tableau
- ▶ Comparaison de deux tableaux
- ▶ Recherche d'un élément dans un tableau
- ▶ Ajout d'un élément dans un tableau
 - ▶ Dans une position p
 - ▶ À la fin du tableau
- ▶ Suppression d'un élément du tableau
 - ▶ La position dans le tableau est connue
 - ▶ L'élément à supprimer est connu
- ▶ Tri des éléments d'un tableau

Structures de données

Les enregistrements

Les enregistrements

- ▶ Contrairement aux tableaux, ce type structuré permet de regrouper des données de types différents.
- ▶ Exemple : on identifie un ouvrage par un code, un titre, un ou plusieurs auteurs, un éditeur et éventuellement la date de parution.
- ▶ Ouvrage est une variable de type enregistrement; chacune de ces cinq données est un champ pouvant être simple ou structuré.

Les enregistrements

- ▶ Les enregistrements sont déclarés en VB avec le mot Type.

- ▶ Syntaxe :

```
Type NomEnregistrement  
  Champ1 As type1  
  Champ2 As type2  
  ...  
End Type
```



- ▶ Exemple :

```
Type ouvrage  
  code as Integer  
  titre As String*40  
  auteur As String*50  
  editeur As String*50  
  dateparution As Date  
End Type
```

```
Type Date  
  jour As Integer  
  mois As Integer  
  annee As Integer  
End Type
```



Les enregistrements

▶ Exemple :

Type ouvrage

code As Integer

titre As String*40

auteur As String*50

editeur As String*50

dateparution As Date

End Type

Type Date

jour As Integer

mois As Integer

annee As Integer

End Type

▶ Pour accéder à un champ :

Dim livre As ouvrage

livre.auteur = "Durand "

livre.dateparution.annee = 1980

‘on s’aperçoit ici que l’on pourrait remplacer livre par un tableau dans le type ouvrage...Dim livre(1 To 10000) as ouvrage...

livre(9).auteur = "Durand" s’il s’agit du neuvième livre de la liste...

...

Les enregistrements – Exemple

Un étudiant est défini par son nom, son prénom, sa date de naissance et sa note :

```
Private Type Etudiant
    nom As String * 40
    prenom As String * 40
    dateNaissance As Date
    note As Double
End Type
```

Une classe peut contenir au plus 30 étudiants :

```
Const NbMax = 30                                     'pour le nombre limite d'étudiants
Private Type Classe
    liste(NbMax) As Etudiant                         'la liste est un tableau d'étudiants
    nbr As Integer                                    'le nombre réel des étudiants
End Type
```

On déclare ensuite la classe d'étudiants :

```
Dim c As Classe
```

Les enregistrements – Exercice

- ▶ L'exemple précédent sera complété dans le prochain cours sur les interfaces graphiques...
- ▶ Comment définir une matrice ?
- ▶ Créer un programme qui affiche le nombre de lignes et de colonnes d'une matrice saisie sur la Feuille du classeur.

Dans le module...

Type matrice

m(100, 100) As Integer

nc As Integer

nl As Integer

End Type

Les enregistrements : saisie matrice 1

Sub saisie(m As matrice)

Dim i As Integer

Dim j As Integer

i = 1

While Feuille.Cells(i, 1) <> "" And i <= 100 'Empty pose
certains problèmes...

i = i + 1

Wend

m.n1 = i - 1

Les enregistrements : saisie matrice 1

j = 1

While Feuil1.Cells(1, j) <> "" And j <= 100 j = j + 1

Wend

m.nc = j - 1

For i = 1 To m.n1

 For j = 1 To m.nc

 m.m(i, j) = Feuil1.Cells(i, j)

 Next j

Next i

End Sub

Les enregistrements : saisie matrice 2

Sub saisie2(m As matrice)

Dim i As Integer

Dim j As Integer

i = 1

While Feuil2.Cells(i, 1) <> "" And i <= 100

i = i + 1

Wend

m.n1 = i - 1

Les enregistrements : saisie matrice 2

j = 1

While **Feuil2**.Cells(1, j) <> "" And j <= 100 j = j + 1

Wend

m.nc = j - 1

For i = 1 To m.n1

 For j = 1 To m.nc

 m.m(i, j) = **Feuil2**.Cells(i, j)

 Next j

Next i

End Sub

Les enregistrements : Affichage

Sub affichage(m As matrice)

Dim i As Integer

Dim j As Integer

Feuil3.Cells.Clear

For i = 1 To m.n1

For j = 1 To m.nc

Feuil3.Cells(i, j) = m.m(i, j)

Next j

Next i

Feuil3.Activate

End Sub

Les enregistrements : somme

Sub somme(A As matrice, B As matrice, C As matrice)

Dim i As Integer

Dim j As Integer

If A.ni <> B.ni Or A.nc <> B.nc Then

MsgBox ("Calcul impossible")

Else

For i = 1 To A.ni

For j = 1 To A.nc

C.m(i, j) = A.m(i, j) + B.m(i, j)

Next j

Next i

C.ni = A.ni

C.nc = A.nc

End If

End Sub

Les enregistrements : produit par une constante

Sub produitscalaire(A As matrice, l As Double, C As matrice)

Dim i As Integer

Dim j As Integer

For i = l To A.ni

 For j = l To A.nc

 C.m(i, j) = l * A.m(i, j)

 Next j

Next i

C.ni = A.ni

C.nc = A.nc

End Sub

Les enregistrements : produit matriciel

Sub produit(A As matrice, B As matrice, C As matrice)

Dim i As Integer, j As Integer, k As Integer

If A.nc <> B.nl Then MsgBox ("Calcul impossible") Else

For i = 1 To A.nl

For j = 1 To B.nc

C.m(i, j) = 0

For k = 1 To A.nc

C.m(i, j) = C.m(i, j) + A.m(i, k) * B.m(k, j)

Next k

Next j

Next i

C.nl = A.nl

C.nc = B.nc

End If

End Sub

Les enregistrements : matrice transposée

Sub transposée(A As matrice) 'on veut une matrice carrée

Dim i As Integer, j As Integer, temp As Integer

If A.n1 <> A.nc Then

MsgBox ("Calcul impossible")

Else

For i = 1 To A.n1

 For j = i + 1 To A.nc

 temp = A.m(i, j)

 A.m(i, j) = A.m(j, i)

 A.m(j, i) = temp

 Next j

Next i

End If

End Sub

Les enregistrements : main

Sub tout()

Static m As matrice

Static p As matrice

Static s As matrice

Dim choix As Integer

Dim l As Double

Do

choix = InputBox("1. Entrer une matrice et l'afficher" & Chr(13) & "2. Entrer deux matrices et afficher la somme" & Chr(13) & "3. Entrer une matrice et afficher le produit entre la matrice et un réel" & Chr(13) & "4. Entrer deux matrices et afficher le produit" & Chr(13) & "5. Entrer une matrice et afficher sa transposée")

Loop Until choix = 1 Or choix = 2 Or choix = 3 Or choix = 4 Or choix = 5

Les enregistrements : main

Select Case choix

Case 1:

Call saisie(m)

Call affichage(m)

Case 2:

Call saisie(m)

Call saisie2(p)

Call somme(m, p, s)

Call affichage(s)

Les enregistrements : main

Case 3:

Call saisie(m)

l = InputBox("entrer l")

Call produitscalaire(m, l, p)

Call affichage(p)

Case 4:

Call saisie(m)

Call saisie2(p)

Call produit(m, p, s)

Call affichage(s)

Les enregistrements : main

Case 5:

Call saisie(m)

Call transposée(m)

Call affichage(m)

End Select

End Sub



Eléments visuels



Les feuilles

- ▶ La feuille : c'est le cadre dans lequel tous les autres éléments (visuels) d'une application VB sont placés :
 - ▶ Contrôles (boutons, zones de texte, listes, ...),
 - ▶ Affichage de texte ou dessin d'images,...
- ▶ La feuille, tout comme les autres contrôles, dispose de propriétés et d'événements.
- ▶ Les éléments systèmes d'une feuille sont gérés par des propriétés. Ils comprennent :
 - ▶ Titre (Caption)
 - ▶ Menu Système (ControlBox)
 - ▶ Bordure (BorderStyle)
 - ▶ La propriété Name permet d'identifier la feuille.

Les feuilles

- ▶ **Les fenêtres prédéfinies : permettent de saisir ou d'afficher du texte:**

- ▶ Saisie de texte : c'est la fonction `InputBox` qui affiche une boîte de saisie et retourne une chaîne de caractères :

- ▶ Syntaxe : `var l = InputBox("message pour l'utilisateur ", "Titre de la fenêtre ", "valeur par défaut ")`

- ▶ Exemple : `dim n as Integer`

...

```
n = InputBox("Donner un entier ", "Exemple", 0)
```

...

- ▶ Affichage de message : c'est la fonction `MsgBox` qui affiche une boîte avec un texte comme message, un ou plusieurs boutons et éventuellement une icône.

- ▶ Syntaxe : `MsgBox("Le message", "Titre de la fenêtre")`

- ▶ Exemple : `MsgBox("Bonjour", "Fenêtre de test")`

Les contrôles

- ▶ Les contrôles sont des objets d'interaction grâce auxquels l'utilisateur construit un dialogue. Ce dernier s'articule sur des affichages et/ou des saisies de données, des ordres de calculs...
- ▶ Comme pour les feuilles, des propriétés et des événements sont associés aux contrôles.
- ▶ Il existe différents types de contrôles :
 - ▶ de texte pour l'affichage ou la saisie de données : zone de texte, étiquette.
 - ▶ les boutons de commande pour le déclenchement d'actions, de radio pour, les choix d'options, les cases à cocher pour les réponses oui/non.
 - ▶ les listes, sous différentes formes : simples, modifiables.
 - ▶ personnalisés : grille, boîte de dialogue, OLE....

Les contrôles

- ▶ La zone de texte : (de la classe « TextBox » pour VB) peut servir à **saisir** (ou à la rigueur, à afficher) une information.
- ▶ **Il s'agit du seul contrôle permettant une saisie au clavier par l'utilisateur.**
- ▶ En Visual Basic, il n'y a donc plus à proprement parler d'instruction **Lire**. A la place de cette instruction, on est contraint de passer par de telles zones.
- ▶ Propriétés : La **propriété essentielle** d'une zone **Text** est... **Text**. C'est la propriété qui désigne son **contenu**. Comme toute propriété, elle va pouvoir être utilisée tant en lecture qu'en écriture.

Les boutons

- ▶ Les boutons : Il s'agit du bouton type OK, Annuler, mais dont le texte apparent (en Anglais, **Caption**) et le rôle dans une application peuvent varier à l'infini.
- ▶ Les boutons permettent de déclencher une action à exécuter.
- ▶ Quelques propriétés intéressantes de la classe **CommandButton** :
 - ▶ **Name** : nom du bouton
 - ▶ **Caption** : texte qui va figurer sur le bouton

Les cases

- ▶ Les cases :
 - ▶ Boutons d'option :



- ▶ Valeurs possibles : True si cochée et False sinon.

Les cases

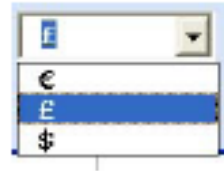
▶ Les cases à cocher

Montant H.T.

Montant T.T.C.

Les listes

► Les listes



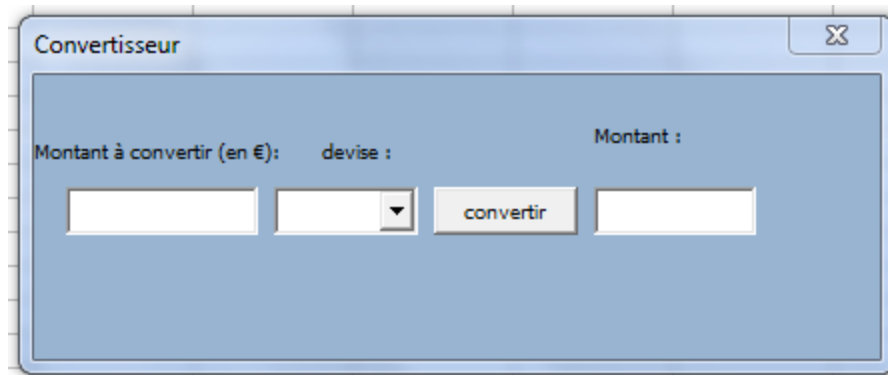
- Le rajout des éléments dans la liste doit se faire « à la main » en utilisant `AddItem`.

Exemple

The image shows a software dialog box titled "Calcul" with a close button in the top right corner. The dialog is divided into several sections:

- Saisir le montant :** A text input field for entering the amount.
- Taux de TVA :** A group box containing three radio buttons for selecting the VAT rate: 19,6%, 11,06%, and 5%.
- Montant à calculer :** A group box containing two radio buttons for selecting the calculation type: Montant H.T. and Montant T.T.C.
- Calculer :** A button to perform the calculation.
- Résultat :** A text input field to display the result.

Exemple 2 : un convertisseur



The image shows a window titled "Convertisseur" with a close button in the top right corner. The window contains three labels: "Montant à convertir (en €):", "devise :", and "Montant :". Below these labels are three input elements: a text box for the amount, a dropdown menu for the currency, and a button labeled "convertir". To the right of the "convertir" button is another empty text box for the result.

Exemple 3 : Base de données sous Excel

- ▶ → Interaction Excel, VBA et UserForm
- ▶ Objectif : montrer
 - ▶ 1. les possibilités qu'offre Excel pour manipuler « des bases de données »
 - ▶ 2. les limites du logiciel
- ▶ → pourquoi doit-on utiliser Access ?...