



Environnement de Développement graphique

(Visual Basic)

Téléchargement cours + exemples :

Adresse: insea2eme@yahoo.fr

Mot de passe: insea2008

Khalid EL HACHEMI

[Sommaire]

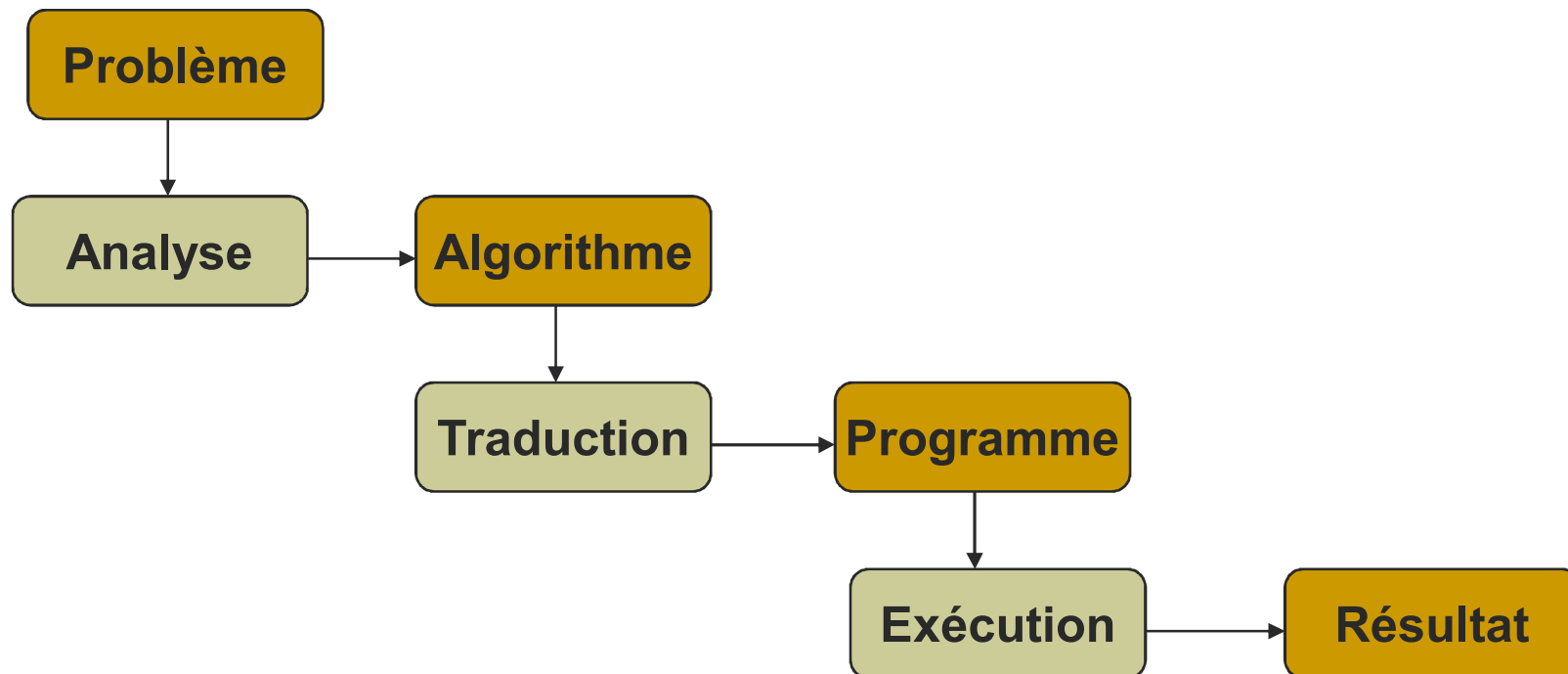
- Introduction
- Notion d'objet
- Présentation de Visual Basic
- Syntaxe de base
- Contrôles et Propriétés
- Événements et Méthodes
- Fonctions prédéfinies
- Fonctions et Procédures utilisateur
- Transmission des paramètres

[Sommaire]

- Types de données utilisateur
- Manipulation des Dates
- Les boîtes de message et la fonction `InputBox`
- Formatage de données (Fonction `Format`)
- Contrôles avancés (menus et menus contextuels – les grilles)
- Gestion des erreurs (L'objet `Err`)
- Accès aux bases de données

[Introduction]

■ La programmation

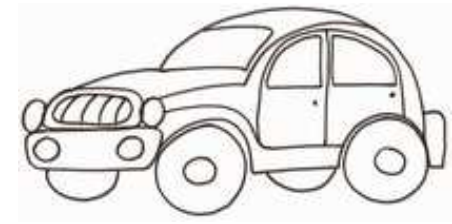


[Notion d'objet]

Objet: Super-variable (groupe de variables de différents types)
Les variables sont désignées par un nom = **Propriétés**

Exemple 1 : l'objet Voiture

Propriétés: couleur, marque, modèle, carburant, puissance fiscale, vitesse Maxi, prix....



Syntaxe : objet • propriété

Voiture.couleur = rouge

Voiture.marque = "Peugeot"

p = Voiture.puissanceFiscale



[Notion d'objet]

Exemple 2 : l'objet Etudiant

Propriétés: nom, classe, adresse, situation familiale, âge, taille, budget, situation/justice....

utilisation :

Etudiant.âge = 23

Etudiant.Budget = Etudiant.Budget + 100

n = Etudiant.nom

[Notion d'objet]

Méthodes = fonctions de l'objet

Exemple : Voiture



Accélérer, ralentir, avancer, reculer, s'arrêter....

Syntaxe : objet • Méthode()

Voiture.accélérer()

Voiture.reculer(10 m)

Voiture.changerCouleur(bleu)

Voiture.couleur = bleu

[Notion d'objet]

Exemple 2 : Méthode de l'objet : Etudiant

Réussir, Redoubler, Tricher, s'absenter,
préparerExamen, AugmenterBudget....

Syntaxe : objet • Méthode()

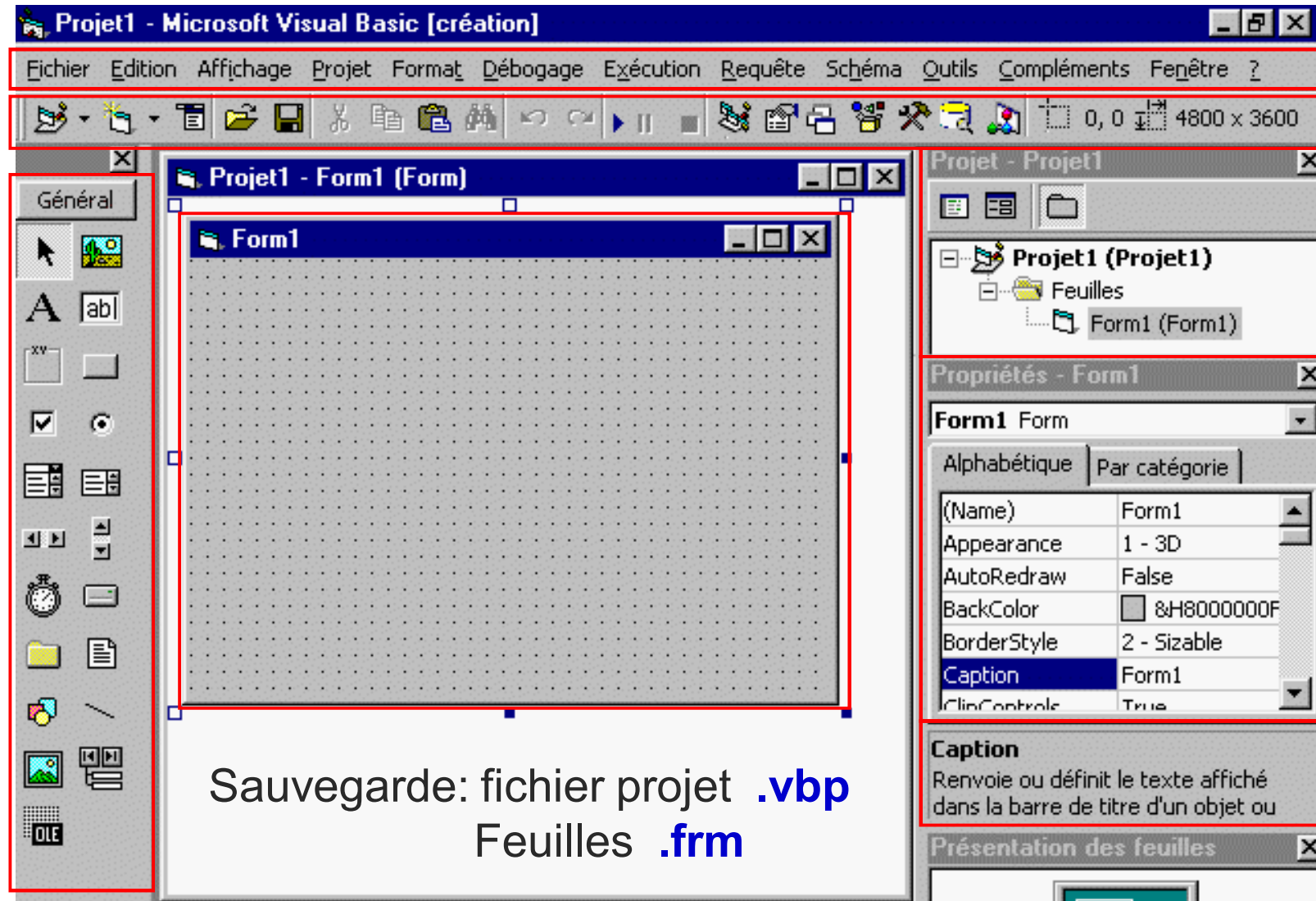
Etudiant.Réussir()

Etudiant.Sabsenter(2)

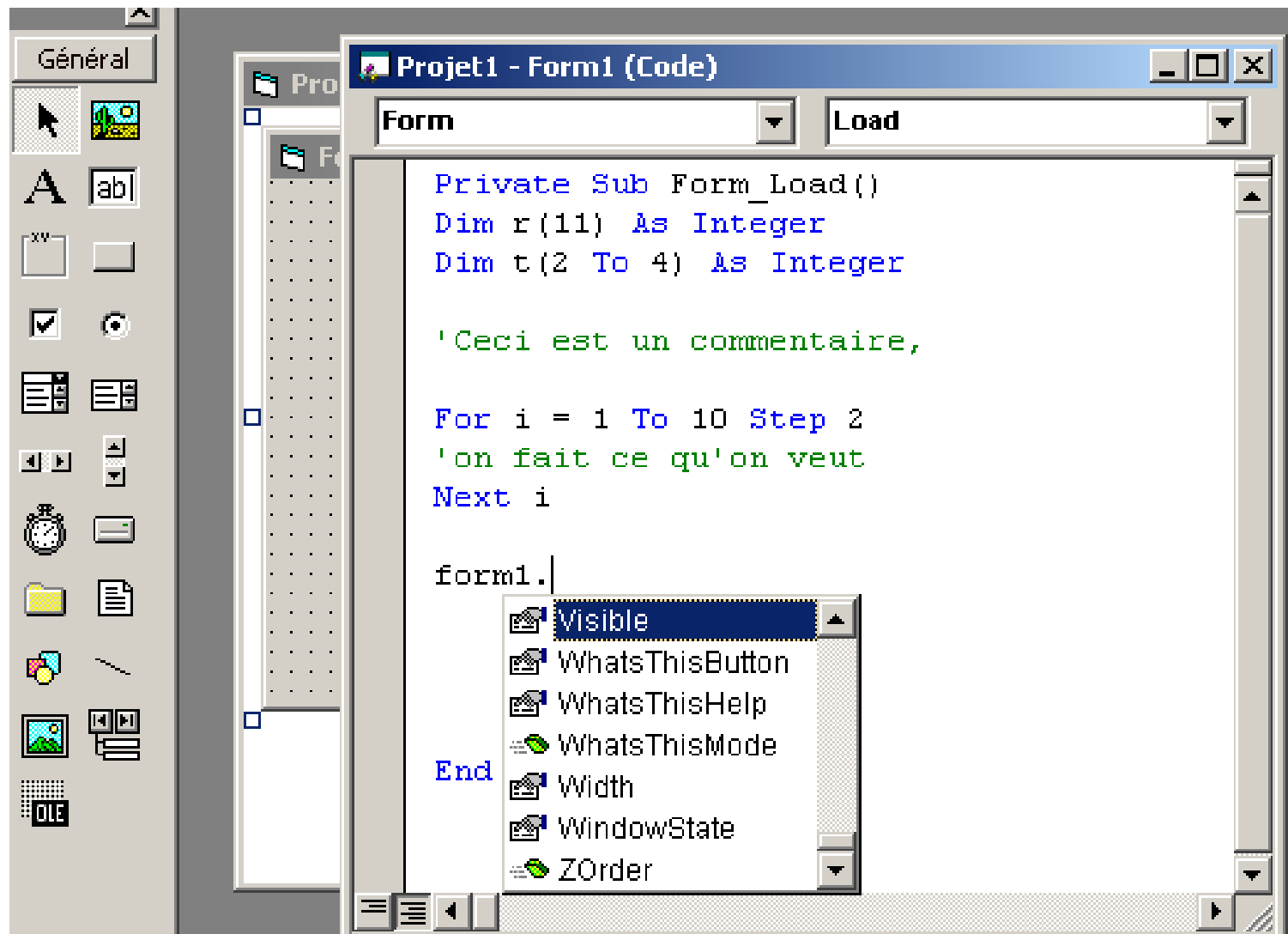
Etudiant.AugmenterBudget(100)

Etudiant.Budget = Etudiant.Budget + 100

Présentation de visual basic



Présentation de visual basic



[Syntaxe de base]

■ Variables:

Noms de variables:

1-Pas de longueur maximale

2-Commence par un caractère: 12~~xy~~

3-Pas d'espace: date_de_naissance ~~date de naissance~~

4-Pas de sensibilité à la casse (toto=Toto=ToTO)

[Syntaxe de base]

■ Variables:

Types:

Boolean : True – False

Byte : 0 à 255

Integer : -32 768 à 32 767

Long : -2 à 2 milliards

Single : réels avec 6 chiffres décimaux ($x=10/3$; 3,333333)

Double : réels avec 14 chiffres décimaux

String : chaînes de caractères

Date : Date et heure

Variant : entier, réel, chaîne, booléen, ...

Objet : objets VB (Contrôles, feuilles,...)

[Syntaxe de base]

■ **Déclaration des Variables:**

Déclaration optionnelle (! Mémoire vive)

■ **Portée des Variables:**

1-Niveau Procédure : **Dim** NomVariable **As Type**

2-Niveau Form : **Dim** NomVariable **as Type**

3-Niveau Projet (module) : **Public** NomVariable **As Type**

Exemple: **Dim** i **As integer**, **Public** b **As boolean**

[Syntaxe de base]

■ Variables indicées (Tableaux): Déclaration obligatoire!!

- 1- Taille connue à l'avance : `Dim MonTableau(n) As Integer`
`Dim MonTableau(11) As Integer` (indices : de 0 à 11 => 12 éléments)
- 2- Tableau dynamique : `Dim MonTableau( ) As Integer`
- 3- Mes propres indices : `Dim MonTableau(2 to 5) As Integer`
- 4- Taille fixée ultérieurement : `Redim MonTableau(10)`
Préserver les valeurs existantes :
`Redim Preserve MonTableau(11)`

■ Les Matrices (mêmes règles que les tableaux!)

`Dim Mat(2,3) As Integer`

[Syntaxe de base]

■ Opérateurs

Opérateurs numérique : + - * /

Opérateurs booléens : And Or Xor Not

Opérateur caractère : & (concaténation)

**** : Division entière

^ : puissance

Mod : reste de la division entière

[Syntaxe de base]

■ L'alternative (si.....sinon.....)

If (Conditions) Then
Liste d'instructions
End If

If (Conditions) Then
Liste d'instructions
Else
Liste d'instructions
End If

If (Conditions) Then
Liste d'instructions
Elseif (Conditions) Then
Liste d'instructions
Elseif (Conditions) Then
Liste d'instructions
.....
Else
Liste d'instructions
End If

[Syntaxe de base]

- **Select case :**

```
Select case x  
  case 0  
    instructions  
  case 1 to 6  
    .....  
  case else  
    instructions  
End Select
```

[Syntaxe de base]

■ Les Boucles

```
For x = a to b Step c  
Liste d'instructions  
Next x
```

Exemple :

```
For i = 1 to 10  
s = s + i  
Next i
```

■ Fin prématurée (*Exit For*)

```
For i = 1 to 10  
  If t(i) = c then  
    Trouvé = true  
    Exit For  
  End if  
Next i
```

[Syntaxe de base]

■ Les Boucles (test avant exécution)

While (Conditions) Liste d'instructions Wend	Do While (condition de continuité) instructions Loop
---	--

Do Until (condition **d'arrêt**)
instructions
Loop

[Syntaxe de base (*complément*)]

- Les Boucles (test après exécution)

Do

instructions

Loop While (condition de continuité)

Do

instructions

Loop Until (condition d'arrêt)

Exemple

Contrôles et Propriétés

■ Le contrôle Form

Propriétés	Signification
BackColor	Définit la couleur d' arrière plan
BorderStyle	Définit le style de la bordure d' un objet
Caption	Définit le texte dans la barre de titre de la feuille
ControlBox	Détermine si les boutons du menu système en haut à droite dans la barre de titre de la feuille sont affichés.
Enabled	Détermine si l' objet peut répondre aux événements, c'est à dire s'il est utilisable.
Font	Définit une police, un style et une taille de caractère
ForeColor	Définit la couleur de premier plan
Height	Définit la hauteur de l' objet à partir du haut
Icon	Indique l' icône qui est affichée quand la feuille est réduite

[Contrôles et Propriétés]

■ Le contrôle **Form**

Exemple ▮

Propriétés	Signification
Left	Indique la position horizontale gauche à l' exécution de la feuille
MaxButton	Définit si une feuille contient un bouton "Agrandissement"
MidiChild	Définit si une feuille est considérée comme feuille enfant MIDI
MinButton	Définit si une feuille contient un bouton "Réduction"
MouseIcon	Définit une icône de souris personnalisée
MousePointer	Définit le type de pointeur de souris
Picture	Définit un graphisme à utiliser
tag	Permet de stocker des informations supplémentaires
Top	Détermine la position verticale de l' objet à exécution
Width	Indique la largeur de l' objet

[Contrôles et Propriétés]

Contrôles visuels standards

The screenshot displays a Windows form titled "je suis une feuille (Form)". The form contains several standard visual controls:

- A red text label at the top center: "je suis une etiquette avec bordure et en alignement centré".
- A frame control labeled "je suis un Frame" containing:
 - A multi-line text box labeled "je suis une zone de text multilignes (TextBox)".
 - A sub-frame labeled "Frame2" containing:
 - Two radio buttons: "Option1" (unselected) and "Option2" (selected).
 - Two checked checkboxes: "Check1" and "Check2".
- A ComboBox control on the right with the text "ComboBox" and a dropdown arrow. The dropdown menu is open, showing three items: "élément 1", "élément 2" (selected), and "élément 3".
- A list box control below the ComboBox, also showing "élément 1", "élément 2" (selected), and "élément 3". It has a vertical scrollbar on its right side.
- A horizontal scrollbar at the bottom right labeled "HScrollbar".

[Contrôles et Propriétés]

- Exemples :

[textBox](#)

[textBox \(calculatrice\)](#)

[AutresExemples...](#)

[Principales Propriétés]

Propriété	Description
Name	permet d'identifier un contrôle dans le code. VB attribue des nom par défaut exemple: Label1, Label2,..., Text1,....
Caption	Etiquette qui accompagne un objet (Label, CommandButton, OptionButton, CheckBox, Frame)
BackColor	couleur de fond d'un contrôle Remarque: pour définir une couleur on utilise la fonction RGB(rouge, vert, bleu)
ForeColor	Couleur du text d'un contrôle
Height	hauteur d'un contrôle en twip (T wentieth of a P oint) = 1cm/567
Width	largeur
Top	distance en twip qui sépare le bord sup du contrôle du bord sup de la feuille
Left	----- gauche du contrôle du bord gauche - - - -
Alignement	Alignement du text dans un contrôle: 0 = gauche, 1 = droite, 2 = centre
Enabled	permet d'autoriser un contrôle à réagir ou non à un événement, (= true -->réagit) Exemple□

[Principales Propriétés]

Propriété	Description
Visible	permet de rendre un contrôle visible ou non
Text	détermine le contenu d'une zone de texte pour : textBox et ComboBox
MaxLength	détermine la longueur maximale d'un texte dans un TextBox
PasswordChar	détermine le caractère masque des caractères tapés dans un TextBox :*****
Locked	permet de verrouiller une zone de texte quand elle prend la valeur true
TabIndex	détermine l'ordre de tabulation d'un contrôle dans la feuille (0, 1, 2,...)
AutoSize	détermine si un contrôle Label est automatiquement redimensionné pour afficher la totalité de son contenu
Value	détermine la valeur d'un contrôle (true, false 0,1,2 ...)
Min	Valeur minimale pour une barre de défilement
Max	Valeur maximale pour une barre de défilement
SmallChange	détermine le pas de déplacement de l'assenseur lorsqu'on clique sur les flèches
LargeChange	détermine le pas de déplacement de l'assenseur lorsqu'on clique sur une zone de la barre

[Principales Propriétés]

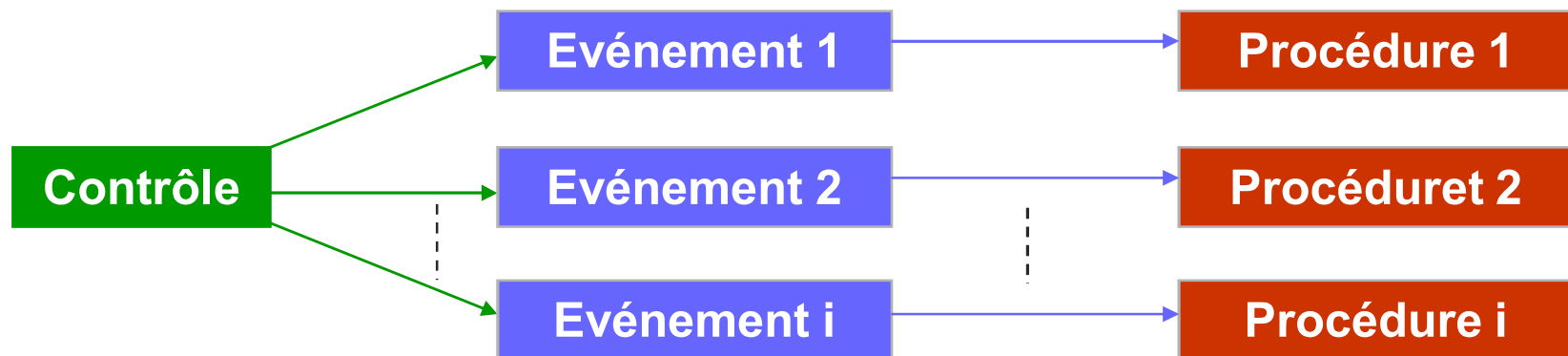
Propriété	Description
FontSize	détermine la taille des caractères : Label, TextBox, CommandButton, OptionButton, Frame, CheckBox
FontBold	Caractères en gras si true
FontName	nom de la police de caractères d'un text
UnderLine	Soulignement si true
FontStrike	barré si true
FontItalic	caractères en italique si true
MultiSelect	mode de selection de éléments d'une liste (0 = 1 seul, 1 = plusieurs éléments, 2 = plusieurs blocs d'éléments)
ListCount	nombre total d'éléments dans une liste
SelCount	nombre d'éléments sélectionnés dans une liste
ListIndex	position de l'élément sélectionné (-1 si aucun)
Selected	renvoi true si un élément est sélectionné (List1.Selected(2))
List	Contenu d'un élément de la liste spécifié par sa position (List1,List (10))
Sorted	permet le remplissage d'élément avec tri automatique (ordre croissant) exemple

[Principales Propriétés]

Propriété	Description
Rows	nombre total de lignes dans une grille
Cols	nombre total de colonnes dans une grille
FixedRows	nombre de lignes fixes dans une grille
FixedCols	nombre de colonnes fixes dans une grille
Row	position de la ligne sélectionnée à l'exécution
Col	position de la colonne sélectionnée à l'exécution
TextMatrix	contenu d'une case de la grille (grille1.TextMatrix(num ligne, num colonne)
Interval	nombre de millisecondes entre les appels à un événement Timer du contrôle Timer exemple

[Événements & Méthodes associées]

Événement = Action produite par l'utilisateur (click, survol) ou par le système (Timer) au moment de l'exécution sur un contrôle



Syntaxe : Private Sub Objet_événement([arguments])

exemple : Private Sub Command1_Click()

Private Sub Check1_GotFocus(Index As Integer)

[Principaux Événements]

1- Click

Se produit lorsque l'utilisateur appuie puis relâche le bouton gauche de la souris (ou entrée si le contrôle a le focus)

```
Private Sub Objet_Click( [Index As Integer])
```

2- DbClick

Se produit lorsque l'utilisateur réalise un double click

```
Private Sub Objet_DbClick( [Index As Integer])
```

3- GotFocus

Se produit lorsqu'un contrôle reçoit le focus (click ou tabulation ou par code : Objet.SetFocus)

```
Private Sub Objet_GotFocus( [Index As Integer])
```

[Principaux Événements]

4- LostFocus >>>

Se produit lorsqu'un contrôle perd le focus

```
Private Sub Objet_LostFocus( [Index As Integer])
```

5-KeyPress >>>

Se produit lorsque l'utilisateur appuie et relâche une touche du clavier

```
Private Sub Objet_KeyPress( [Index As Integer], KeyAscii As Integer)
```

Codes : 0 : 48 | 9 : 57 | A : 65 | Z : 90 | a : 97 | z : 122 | entrer : 13 ... >>>

6- Change

Se produit lorsque le contenu d'un contrôle change

```
Private Sub Objet_Change( [Index as Integer])
```

[Principaux Événements]

7- Timer

Se produit à chaque fois que la période de temps fixée par la propriété « interval » d'un contrôle Timer s'est écoulée

```
Private Sub Objet_Timer( [Index As Integer])
```

8- Autres événements liés à la souris

MouseDown

MouseUp

MouseMove >>>

[Méthodes associées]

Méthode = sous programme propre au contrôle qui, en s'exécutant, permet d'effectuer une action donnée

Utilisation : Objet.Méthode [Liste d'arguments]

□ Quelques méthodes :

1- SetFocus : envoie le focus à un contrôle qui peut le recevoir (Enabled = True et Visible = True) appel : **Objet.SetFocus**

2- AddItem : ajoute un élément dans un contrôle List ou ComboBox

Appel : **List1.AddItem** élément

3- RemoveItem : supprime un élément d'une liste, l'élément est spécifié par sa position Appel : **List1.RemoveItem** position

[Méthodes associées]

□ Quelques méthodes :

4- Clear : supprime tous les éléments d'une liste

>>>

appel : **List1.Clear**

5- Move : déplace un contrôle vers de nouvelles coordonnées (gauche, haut)

Appel : **Objet.Move** 100, 500

>>>

[Fonctions Prédéfinies]

Fonction : sous programme qui reçoit un ensemble de données d'entrée, effectue un traitement et qui renvoie **un et un seul** résultat

Données d'entrée : Arguments

Résultat calculé : retourné par le nom de la fonction

Fonctions prédéfinies : offertes par Visual Basic
(mathématique, chaînes, booléennes, conversion, fichiers ...)

[1- Fonctions Numériques]

Fonction	Description	Type résultat	exemple
Abs(x)	Valeur absolue	numérique	a=Abs(-3,4) □ 3,4
Sqr(x)	Racine carrée	numérique	a=Sqr(25) □ 5
Fix(x)	Partie entière	entier	a=Fix(3,62) □ 3 a=Flx(-3,21) □ -3
Int(x)	Plus grand entier < x	entier	a=Int(3,62) □ 3 a=Int(-3,21) □ -4
Sgn(x)	Signe de x	entier	a=Sgn(3) □ 1 a=Sgn(0) □ 0 a=Sgn(-3) □ -1
Rnd	Nombre aléatoire ∈[0, 1[	réel	x=Rnd □ ?

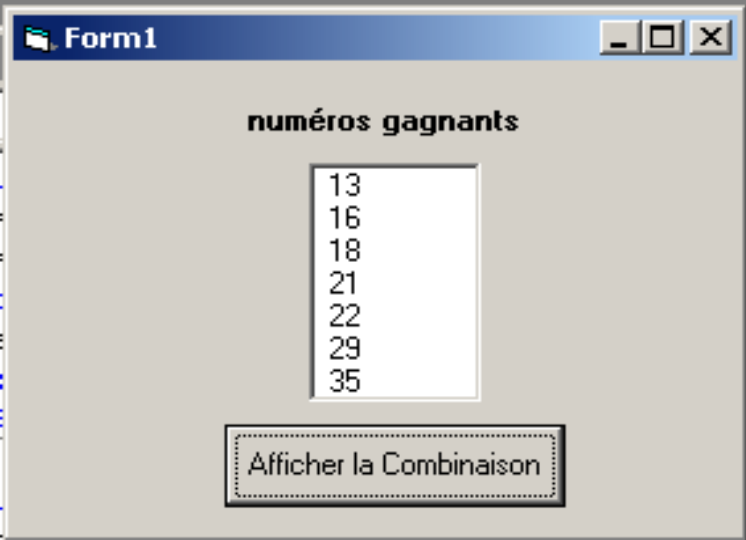
[1- Fonctions Numériques]

Générer un nombre entier aléatoire entre a et b:

$$X = \text{Fix} (\text{Rnd} * (b - a + 1) + a)$$

Changer la valeur de départ ☐ **Randomize Timer** >>>

Exercice : Loto



The screenshot shows a Windows application window titled "Form1". Inside the window, the text "numéros gagnants" is displayed above a list box. The list box contains the following numbers: 13, 16, 18, 21, 22, 29, and 35. Below the list box is a button labeled "Afficher la Combinaison".

[2- Fonctions chaînes]

Fonction	Description	Type résultat	exemple
Len(ch)	Longueur de ch	entier	a=Len("Insea") = 5
Asc(ch)	Code Ascii du 1 ^{er} caractère de ch	entier	a=Asc("Insea") = 73
Chr(x)	Caractère dont le code Ascii = x	string	a=Chr(103) = g
Ucase(ch)	conversion en majuscules	string	
Lcase(ch)	Conversion en miniscules	string	
Space(x)	Chaîne formée de x espaces	string	a=Space(3) = " "

2- Fonctions chaînes

Fonction	Description	Type résultat	exemple
Ltrim(ch)	Élimine les espace de gauche	string	Ltrim(" salut") → "salut "
Rtrim(ch)	Élimine les espace de droite	string	Rtrim("Insea ") → "Insea"
Trim(ch)	Élimine les espace de gauche et de droite	string	Trim (" Insea ") → "Insea"
String(n,c)	Chaîne formée de n fois le caractère c	string	String(3, "A") → "AAA"
Left(ch,n)	Chaîne formée de n caractères de gauche	string	Left("insea",2) → "in"
Right(ch,n)	Chaîne formée de n caractères de gauche	string	Right("insea",3) → "sea"
Mid(ch,p,n)	Chaîne formée de n caractères à partir de p	string	Mid("abcdef",4,2) → "de"

[3- Fonctions de conversion]

Fonction	Description	Type résultat	exemple
Str(x)	Convertit x en chaîne	string	Str(123) □ "123"
Val(ch)	Convertit ch en entier	entier	Val("123") □ 123
CSng(ch)	Convertit ch en réel	réel	CSng("103,41") □ 103,41 CSng("2,5e3 ") □ 2500
DateValue(ch)	Convertit ch en date	date	DateValue("05/02/1976") □ 05/02/1976
CVDate(ch)	Convertit ch en moment (date et heure)	date	CVDate("14/02/2006 3:40:00") □ 14/02/2006 3:40:00
Date	Date système	date	
Time	Heure système	date	
now	Date et heure système	date	

Autres Fonctions de conversion

Les fonctions ci-dessous convertissent une **expression** en un **type de données** spécifique.

Syntaxe :

CBool(*expression*)
CByte(*expression*)
CDate(*expression*)
CDBl(*expression*)
CDec(*expression*)
CInt(*expression*)
CLng(*expression*)
CSng(*expression*)
CStr(*expression*)
CVar(*expression*)

[Opérations sur les dates]

⊙ **Nombre de jours qui sépare deux dates:**

$$\text{Nbj} = \text{Date1} - \text{Date2}$$

⊙ **Durée en jours entre deux horaires**

$$H1 = \text{TimeValue}("11:00:00")$$

$$H2 = \text{TimeValue}("17:00:00")$$

$$X = H2 - H1 = 0,25$$

⊙ **déterminer une nouvelle date à partir d'une ancienne date + n jours**

- **Quelle date seront nous après 20 jours?**

$$D2 = \text{Date} + 20$$

[Fonctions Utilisateur]

☉ Syntaxe:

```
[Public | Private] Function NomFonction([Liste arguments]) As Type
```

```
Instructions
```

```
NomFonction = expression
```

```
End Function
```

☉ Exemple

```
Function Racine (N As Double) As Double
```

```
If N < 0 Then ' Évalue l'argument.
```

```
Exit Function ' Quitte pour revenir à la ' procédure appelante.
```

```
Else ' Renvoie la racine carrée.
```

```
Racine = Sqr(N)
```

```
End If
```

```
End Function
```

[Procédures Utilisateur]

☉ Syntaxe:

```
[Public | Private] Sub NomProcedure([Liste arguments])  
Instructions  
End Sub
```

☉ Exemple

```
Private Sub Init( )  
Nom.Text = ""  
Prénom.Text = ""  
...  
End Sub
```

☉ Appel

```
Call Init >>>
```

TP

[Transmission des paramètres]

Deux modes de transmission :

1- Par référence (ByRef) □ Mode par défaut

```
Private Sub Toto ( ByRef Y as Integer)
```

```
Private Sub Toto ( Y as Integer)
```

```
Private Function Carre(X as Double)
```

```
    Carre = X * X
```

```
End Function
```

2- Par valeur (ByVal)

```
Private Sub Dodo ( ByVal Y as Integer)
```

```
>>>
```

[Types définis par l'utilisateur]

Déclaration : (dans un module !!!)

```
[Private | Public] Type NomType  
    Var1 as type  
    Var2 as type  
    .....  
End Type
```

Utilisation :

```
Public n As NomType  
n.Var1 = expression
```

Manipulation de Date

DateAdd("m",3,"31-jan-95")
DateDiff("d","01/01/2005","16/05/2006")
DatePart("q",Date)

Year(Date)
Month(Date)
MonthName(2, true)
WeekDay(Date)
weekdayName(3, false, VbSunday)
Day(Date)

Hour(Time)
Minute(Time)
Second(Time)

>>>

Valeur	Description
yyyy	Année
q	Trimestre
m	Mois
y	Jour de l'année
d	Jour
w	Jour de la semaine
ww	Semaine
h	Heure
n	Minute
s	Seconde

[MsgBox - Instruction ou fonction ?]

Cette instruction très utilisée dans Visual Basic permet de poser une question à l'utilisateur au travers d'une boîte de dialogue.

Syntaxe:

Variable = MsgBox("Message", Type, "Titre de la fenêtre")

Variable : Variable de type Entier(Integer) recevant la valeur du bouton

Message : Chaîne de caractères affichée comme message dans la boîte de dialogue

Type : Expression numérique qui contrôle les boutons et les icônes à afficher.

Titre : Expression chaîne affichée dans la barre de titre de la boîte de dialogue.

MsgBox (type)

■ Les constantes pour les boutons

Constante VB	Chiffre	Affichage
vbOKOnly	0	le bouton OK uniquement
vbOKCancel	1	les boutons Ok et Annuler
vbCancelRetryIgnore	2	les boutons Abandonner, Répéter et Ignorer
vbYesNoCancel	3	les boutons Oui, Non et Annuler
vbYesNo	4	les boutons Oui et Non
vbRetryCancel	5	les boutons Répéter et Annuler

MsgBox (type)

■ Les constantes pour les icônes

Constante VB	Chiffre	Affichage
vbCritical	16	l'icône erreur critique 
vbQuestion	32	l'icône question 
vbExclamation	48	l'icône message d'avertissement 
vbInformation	64	l'icône message d'information 



MsgBox "Erreur fatale, division par 0", vbCritical

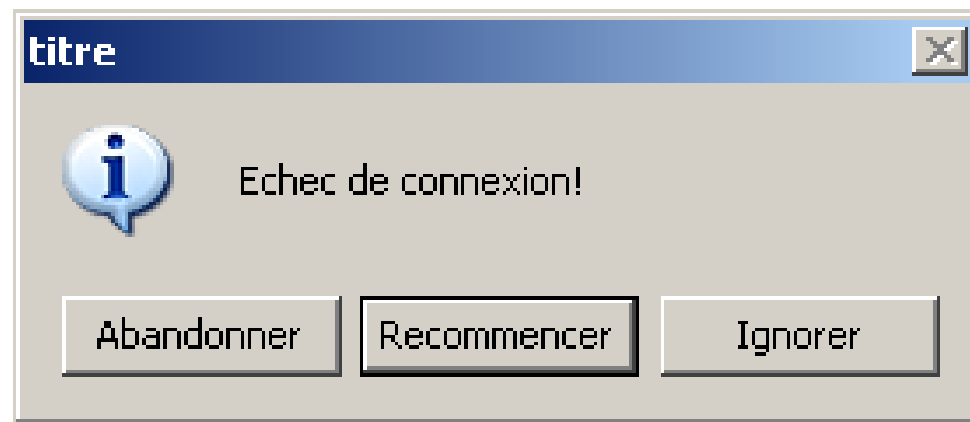
[MsgBox (type)]

■ Les constantes pour le bouton par défaut

Constante	Chiffre	Description
vbDefaultButton1	10	Le 1er bouton est le bouton par défaut
vbDefaultButton2	256	Le 2ème bouton est le bouton par défaut
vbDefaultButton3	512	Le 3ème bouton est le bouton par défaut

[MsgBox]

Exemple



```
x = MsgBox("Echec de connexion!", vbDefaultButton2 + vbAbortRetryIgnore + _  
          vbInformation, "titre")
```

[MsgBox (constantes renvoyées)]

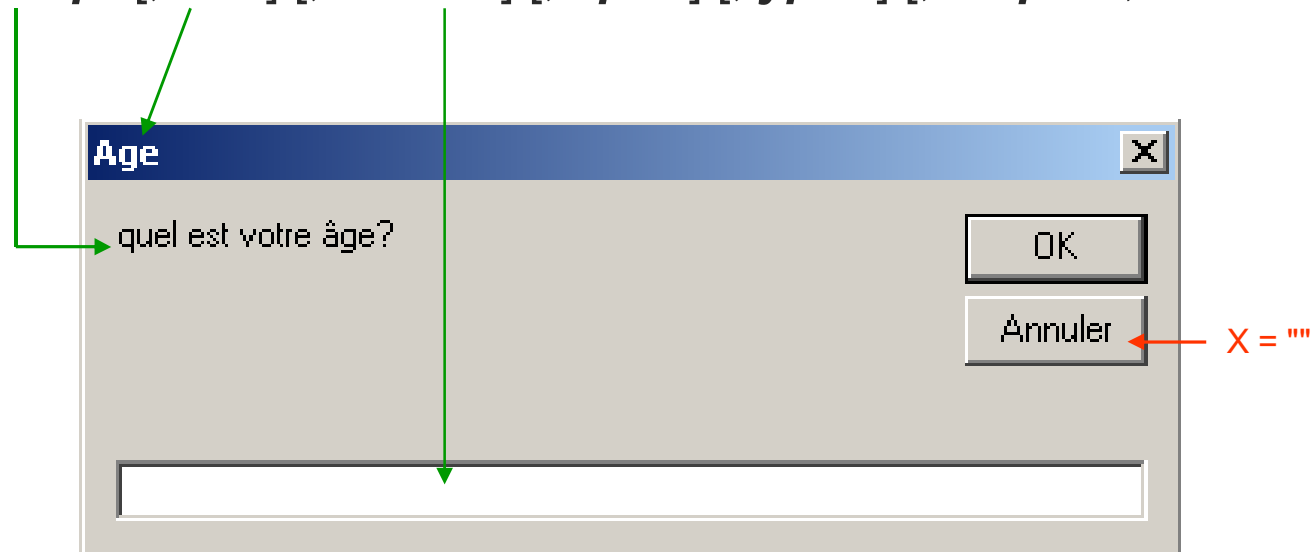
■ Les constantes renvoyées par MsgBox

Constante	Chiffre	Description
vbOk	1	OK
vbCancel	2	Annuler
vbAbort	3	Abandonner
vbRetry	4	Réessayer
vbIgnore	5	Ignorer
vbYes	6	Oui
vbNo	7	Non

[La fonction : **InputBox**]

■ Syntaxe

InputBox (*prompt* [, *title*] [, *default*] [, *xpos*] [, *ypos*] [, *helpfile*, *context*])



```
x = InputBox("quel est votre âge?", "Age", , 1000, 2000)
```

[La fonction `Format( )`]

`Format(expression_à_formater, format_donné)`

`expression_à_formater :`

- 1- chaîne de caractère
- 2- numérique
- 3- date

[La fonction `Format( )`]

1- chaîne de caractère

Caractère de Formatage	Description
@	Un caractère apparaît à la position @
&	Un caractère s'il existe apparaît à la position &
<	Convertit en minuscules
>	Convertit en majuscules

Important !! : les @ et & s'appliquent de droite à gauche

Exemples:

`Format("022554466", "@@@-@@-@@-@@")` □ 022-55-44-66

`Format("022554466", "&&&&-&&&-&&&")` □ 022-554-466

`Format("Intéressant", ">")` □ INTERESSANT

[La fonction **Format()**]

1- numérique

Caractère de Formatage	Description
0	Un chiffre apparaît à la position 0
#	Un chiffre s'il existe apparaît à la position #
.	Point décimal
,	Séparateur de milliers

Exemples:

Format (3456.199, "# # #,# # #.00") □ 3 456.20

Format(3456.199, "000,000.00") □ 003 456.20

Format(0.199, "# #.# #") □ .2

Format(0.199, "#0.00") □ 0.20

[La fonction **Format()**]

1- Date

Caractère de Formatage	Description
d	Affiche le jour sans zéro non significatif (1 à 31)
dd	Affiche le jour avec un zéro non significatif (01 à 31)
ddd	Affiche le jour en lettre sous sa forme abrégée (lun-dim)
dddd	Affiche le jour en lettre sous sa forme complète (lundi...)
m	1-2-3 -.....12
mm	01-02-03 -12
mmm	jan-fév- déc
mmmm	Janvier-décembre
y	Jour de l'année (1-366)
yy	Année avec 2 chiffres (00-99)
yyyy	Année avec 4 chiffres (100-9999)

[L'instruction With]

With Text1

.BackColor = vbRed

.BorderStyle = 1

.Alignment = 2

With .Font

.Bold = True

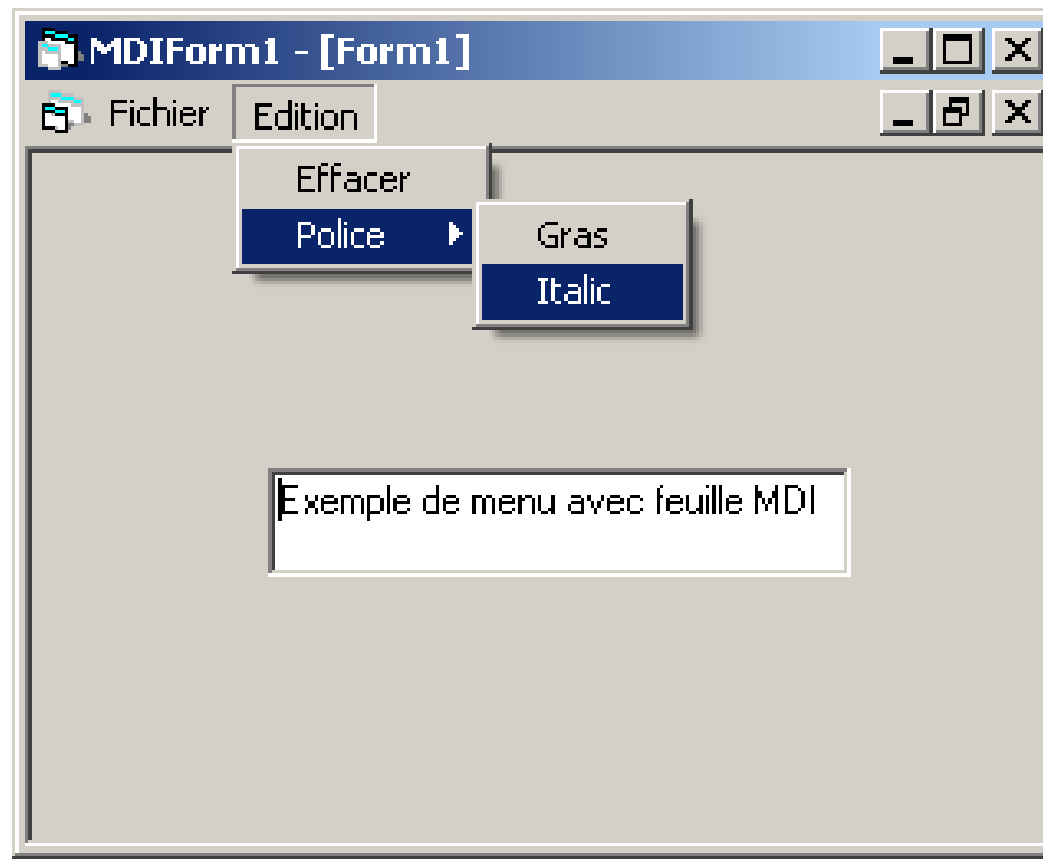
.Size = 14

End With

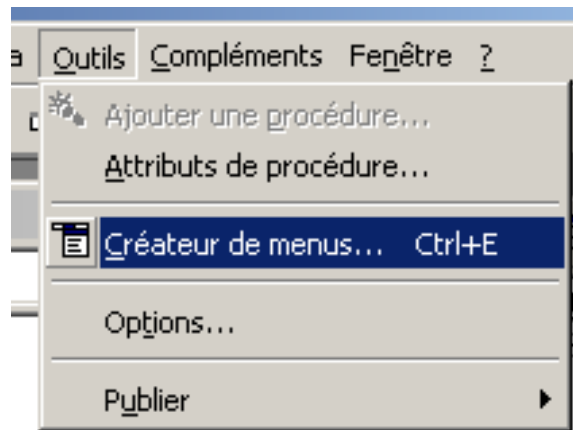
End With

[Contrôles avancés]

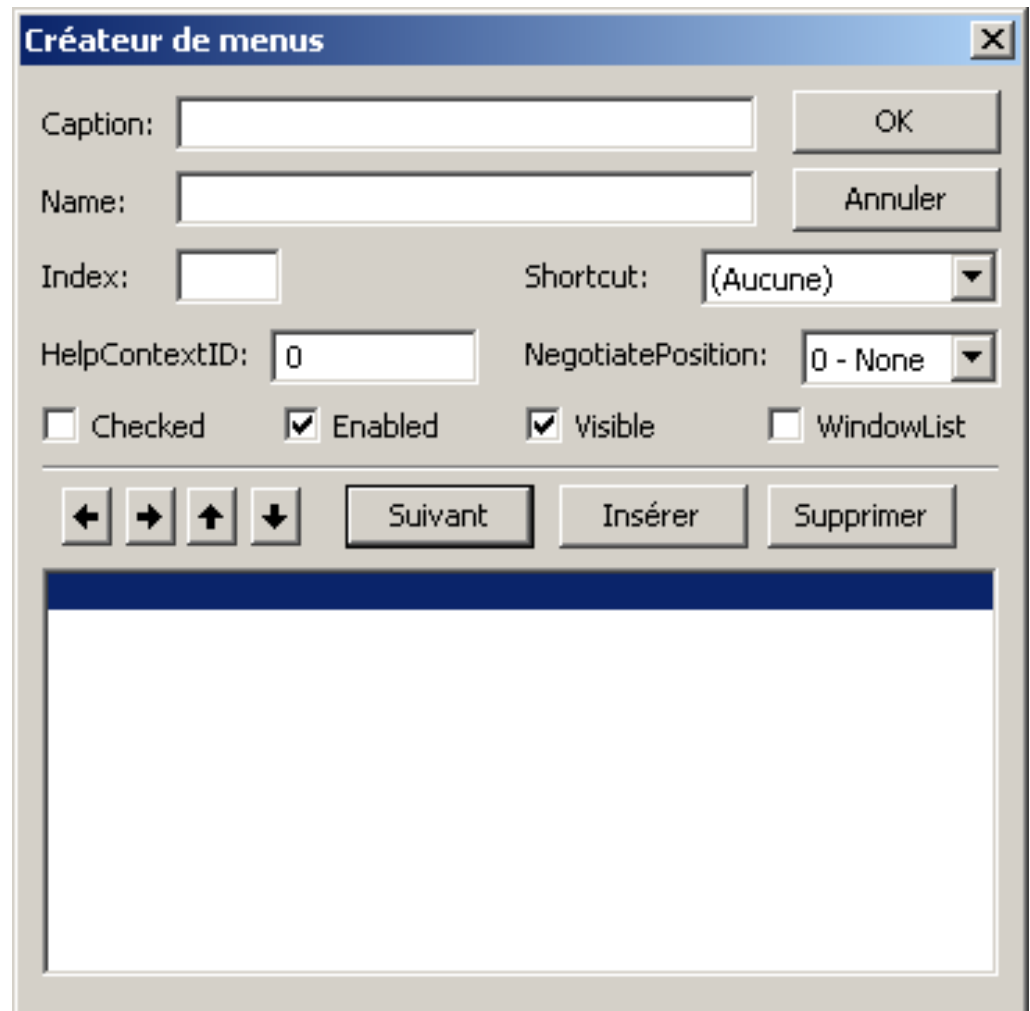
Les menus :



[Contrôles avancés (les menus)]



Application à réaliser



[Contrôles avancés]

Les menus contextuels (PopupMenu) :

Un menu contextuel est un élément du menu
mais invisible!

Démonstration

Gestion des erreurs

■ L'instruction On Error

Exemple :

```
Public Sub Divise()  
    On Error GoTo Oops  
    Tata = 12 / 0  
    Exit Sub  
Oops:  
    MsgBox("Diviser par zéro, c'est pas malin !")  
End Sub
```

[*Récupération des erreurs*]

■ L'objet **Err**

Cet objet, qui n'étant pas un contrôle, ne possède pas d'existence graphique, est utilisé par toute application pour stocker l'état des Erreurs qu'elle a éventuellement provoquées. On pourra, dans le code, avoir recours à différentes **propriétés** de l'objet Err :

Number : désigne le code de cette erreur, permettant d'identifier son type

Description : il s'agit d'un rapide bla-bla sur la nature de l'erreur

Source : qui indique l'objet à l'origine de l'erreur

[Récupération des erreurs]

■ L'objet Err

Exemple:

```
Public Sub Divise()  
    On Error GoTo Oops  
    Tata = 12 / 0  
    Exit Sub
```

```
    Oops:  
    MsgBox "Erreur numéro: "& Err.Number & Err.Description  
End Sub
```

[Gestion des fichiers]

Etape 1: Ouverture

Le système réserve une place en mémoire pour permettre l'accès au fichier par l'application. Elle est réalisée par l'instruction : **Open** (si le fichier n'existe pas, elle le crée !!)

Etape 2: Traitement du fichier

Lectures ou écritures de données. Il est possible aussi de se positionner dans le fichier à un endroit quelconque

Etape 3: Fermeture du fichier

Le système libère la mémoire réservée. Elle est réalisée par l'instruction : **Close**

Gestion des fichiers

■ Accès séquentiel (ouverture)

1- En lecture seule : **Input**

2- En écriture : **Output**

Attention ! : le fichier est réinitialisé à vide et son contenu précédent est perdu

3- En Mise à jour (ajout): **Append**

Le contenu précédent est conservé, et l'écriture dans le fichier ajoute de nouveaux enregistrement

Exemple: **Open** "Monfich.txt" For **Input** As #1
 Open "Monfich.txt" For **Append** As #1

Gestion des fichiers

■ Accès séquentiel

Exemple: `Open "Monfich.txt" For Input As #1`
 `Open "Monfich.txt" For Append As #500`

Remarque :

De 1 à 255 : accès exclusif au fichier, les autres applications ne peuvent pas y accéder tant qu'il est ouvert

De 256 à 511 : fichier partagé avec d'autres applications

Gestion des fichiers

■ Accès séquentiel

Lecture : `Open "C:\contact.txt" For Input As #1`
 `Line Input #1, ligne`

Ecriture : `Open "C:\contact.txt" For Append As #1`
 `nom = txtnom`
 `prenom = txtprenom`
 `tel = txttel`
 `Print #1, nom & prenom & tel`

Important : un enregistrement est composé de plusieurs champs (variables String) de taille fixe :

```
Dim nom As String * 20
Dim prenom As String * 20
Dim tel As String * 9
```

Gestion des fichiers

■ Lecture séquentielle

Line Input #1, ligne

toto	tata	111111111
fofo	fifi	222222222
salah	salim	333333333
mohamed	mahmoud	444444444

EOF(1) = True

>>>

Gestion des fichiers

■ Accès Direct (Random)

`Open "Monfich.txt" For Random As # 1 Len = tailleEnregistrement`

Exemple : `Open App.Path & "\test.txt" For Random As #1 Len = Len(contact)`

Lecture : `Get numFichier, numEnregistrement, variable`

Exemple : `Get 1, 3, contact`
(retourne le 3^{ème} enregistrement et l'affecte à la variable contact)

Ecriture : `Put numFichier, numEnregistrement, variable`

Exemple : `Put 1, 5, contact`

Gestion des fichiers

■ Accès Direct (Random)

Remarque: La variable contact est une variable de type composé

```
Public Type enregistrement (déclaration dans un module !!)
    nom As String * 20
    prenom As String * 20
    tel As String * 9
End Type
```

Ensuite: quelque part dans le code ...

```
Dim contact As enregistrement
Open App.Path & "\test.txt" For Random As #1 Len = Len(contact)
Get 1, 1, contact
txtnom = Trim(contact.nom)
txtprenom = Trim(contact.prenom)
txttel = Trim(contact.tel)
```

>>>

Accès aux bases de données

Le langage SQL

LMD : Langage de Manipulation de Données

Extraction de données

Regroupement et synthèse de données

Jointure des tables

Modification de données (mise à jour et suppression)

[Le langage SQL]

1. Extraction de données

```
SELECT champ [, champ2]...  
FROM table  
WHERE condition  
ORDER BY champ [ASC | DESC]
```

Utilisation de la clause SELECT

Exemple: 1-

```
SELECT CodeFact, DateFact, Montant * 1.2  
FROM Facture  
ORDER BY DateFact
```

2-

```
SELECT *  
FROM Facture
```

[Le langage SQL]

Utilisation de la clause WHERE

Exemple: SELECT CodeFact, DateFact, Montant * 1.2
 FROM Facture
 WHERE Montant > 10 000 And DateFact < # 31/12/2005 #
 ORDER BY DateFact

Opérateurs: = < > <= >= <> Between a And b
 And Or Xor Not
 IN
 Like

[Le langage SQL]

Utilisation de la clause WHERE

Exemple: SELECT Personnes.Nom, Personnes.Prénom
 FROM Personnes
 WHERE Personnes.Nom Like "c*"

* : designe 0 ou plusieurs caractères

? : Désigne un caractère

Question : Personnes dont le nom est composé de 5 caractère ?

 SELECT Personnes.Nom, Personnes.Prénom
 FROM Personnes
 WHERE Personnes.Nom Like "?????"

Au moins 5 caractère : Like "?????*"

Contient la lettre b : Like "*b*"

Ne contient pas la lettre b : Not Like "*b*"

Le langage SQL

Élimination des doublons : (DISTINCT)

```
SELECT Distinct Type  
FROM Produit
```

Limitation de nombre de lignes renvoyées par une requête :

```
SELECT TOP 10 nomProduit  
FROM Produit
```

Question : nom des 3 étudiants les mieux classés ainsi que leurs moyennes?

```
SELECT TOP 3 nom, moyenne  
FROM Resultat  
ORDER BY moyenne DESC
```

Question : nom et moyenne du dernier de la classe ?

```
SELECT TOP 1 nom, moyenne  
FROM Resultat  
ORDER BY moyenne
```

[Le langage SQL]

2. Regroupement et synthèse de données

Fonctions d'agrégation :

SUM : somme

AVG : moyenne

MIN : minimum

Max : maximum

COUNT : nombre de lignes

SELECT **MAX**(moyenne) (ou **MAX**(**DISTINCT**(moyenne)))
FROM Resultat

SELECT **COUNT**(*)
FROM Employe (nombre d'employés dans l'entreprise)

SELECT **SUM**(Salaire)
FROM Employe (masse salariale)

Le langage SQL

- Regroupement avec GROUP BY

```
SELECT Type, MAX(Prix), MIN(Prix)
FROM Produits
GROUP BY Type
ORDER BY Type
```

Important : tout champ qui n'est pas contenu dans une fonction d'agrégation doit figurer dans la clause GROUP BY

Question : types de produits dont le prix maximum dépasse 1000 DH?

```
SELECT Type, MAX(Prix)
FROM Produits
GROUP BY Type
HAVING MAX(Prix) > 1000
```

Important : la clause WHERE exprime des conditions sur les champs (application avant regroupement) alors que la clause HAVING exprime des condition sur le groupe (application après regroupement)

3. Jointure des tables: jointures internes

Pere

numPere	Nom	prenom
1	Salmi	Mohamed
2	Bahi	Rida
3	Alami	Khalil
4	Rizki	Rachid

Fils

numPere	prenom
1	Slimane
1	salima
3	Laila

SELECT Pere.Nom, Fils.Prenom
FROM Pere Join Fils
On Pere.numPere = Fils.numPere

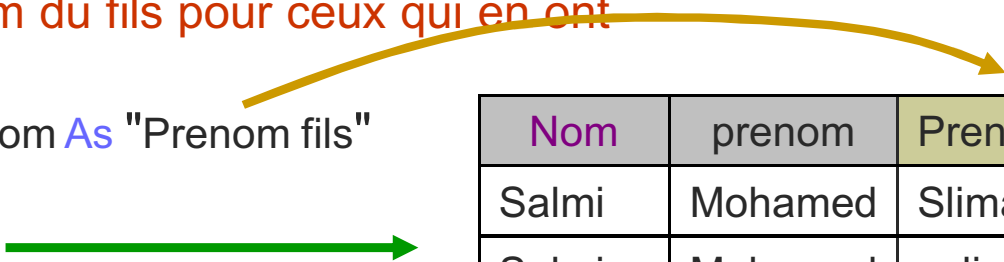
Nom	prenom
Salmi	Slimane
Salmi	salima
Alami	Laila

La requête retourne uniquement les enregistrements qui respectent la condition,
on parle de **Jointure interne**

3. Jointure des tables: jointures externes

Question : Liste de tous les pères (nom et prenom)
ainsi que le prénom du fils pour ceux qui en ont

```
SELECT p.Nom, p.Prenom, f.Prenom As "Prenom fils"  
FROM Pere p Left Join Fils f  
On p.numPere = f.numPere
```



Nom	prenom	Prenom fils
Salmi	Mohamed	Slimane
Salmi	Mohamed	salima
Bahi	Rida	NULL
Alami	Khalil	Laila
Rizki	Rachid	NULL

1- La requête retourne tous les enregistrements de la table de gauche ainsi que les enregistrements de la table de droite qui respectent la condition

2- L'usage des alias (p, f) permet de réduire le code et le rendre plus lisible. Dans le cas où les tables jointes possèdent des noms de champs identiques, cette recommandation devient obligatoire

3. Jointure des tables: jointures externes

Remarque : On dispose également de jointure externes à droite: **Right Join**.
Le principe est le même, il suffit d'inverser l'ordre des tables

```
SELECT p.Nom, p.Prenom, f.Prenom  
FROM Fils f Right Join Pere p  
On p.numPere = f.numPere
```



Nom	prenom	Prenom fils
Salmi	Mohamed	Slimane
Salmi	Mohamed	salima
Bahi	Rida	NULL
Alami	Khalil	Laila
Rizki	Rachid	NULL

Question: Résultats de la requête suivante ?

```
SELECT p.Nom, p.Prenom, f.Prenom  
FROM Fils f Left Join Pere p  
On p.numPere = f.numPere
```



Nom	prenom	Prenom fils
Salmi	Mohamed	Slimane
Salmi	Mohamed	salima
Alami	Khalil	Laila

3. Jointure des tables: jointures externes

Employe

nume	Nom	nums
1	Salmi	20
2	Bahi	10
3	Alami	20
4	Rafik	

Service

nums	nomService	ville
10	Rehcerche	Rabat
20	Vente	Kénitra
30	Achat	Fès

Question : noms de tous les employés ainsi que le nom du service où ils travaillent ?

```
SELECT e.Nom, s.nomService
FROM Employe e Left Join Service s
On e.num = s.num
```



Nom	nomService
Salmi	Vente
Bahi	Recherche
Alami	Vente
Rafik	

3. Jointure des tables: jointures externes

Employe

nume	Nom	nums
1	Salmi	20
2	Bahi	10
3	Alami	20
4	Rafik	

Service

nums	nomService	ville
10	Rehcerche	Rabat
20	Vente	Kénitra
30	Achat	Fès

Question : nombre d'employés par service ?

```
SELECT s.nomService, COUNT(e.nume) As "nbrEmploye"  
FROM Service s Left Join Employe e  
On e.nums = s.nums  
GROUP BY s.nomService
```



nomService	nbrEmploye
Recherche	1
Vente	2
Achat	0

4. Ajout de données: INSERT [INTO]

INSERT Etudiant (nom, prenom, note)

VALUES ("Carlos", "Santana", 14)

INSERT Etudiant

VALUES (Liste valeurs dans l'ordre des champs de la table)

5. Modification de données: UPDATE

UPDATE Etudiant

SET note = note+2

WHERE Classe = 3 And Option = "Actuariat"

Remarque :

La clause WHERE est utilisée de la même manière qu'avec la clause SELECT

6. Suppression de données: DELETE

Supprimer l'étudiant Carlos

```
DELETE Etudiant  
WHERE nom = "Carlos"
```

Vider la table Etudiant :

```
DELETE Etudiant
```

Accès aux bases de données

■ Le contrôle Data

Mise à jour Client

Ajouter Mettre à jour Supprimer Fermer

Client n° :

Nom et Prénom :

Adresse :

Téléphone :

Navigation:

[Utilisation du contrôle Data]

1-Initialiser les principales propriétés

DatabaseName = C:\magasin.mdb

RecordSource = Nom Table (*ou requête accès*)

2- Lier les différentes zones de texte au contrôle Data

DataSource = nom du contrôle Data

3- Lier la zone de texte à un champ de la table

DataField = nom du champ

[Utilisation du contrôle Data]

Les contrôles pouvant être liés à un contrôle Data:

Label (étiquette)

TextBox (chamm de saisie)

CheckBox (case à cocher)

PictureBox (zone d'image)

ListBox (liste)

ComboBox (liste combinée)

Il est également possible d'utiliser d'autres contrôles :

DBCombo (liste combinée liée)

DBList (liste liée)

DBGrid (grille liée)

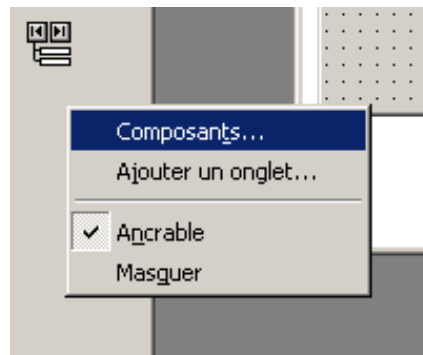
[Utilisation du contrôle Data]

Remarque : pour travailler avec les listes, les listes combinées et les grilles, il faut les ajouter à la boîte à outils :

Click bouton droit dans la boîte à outils , puis **Composant**

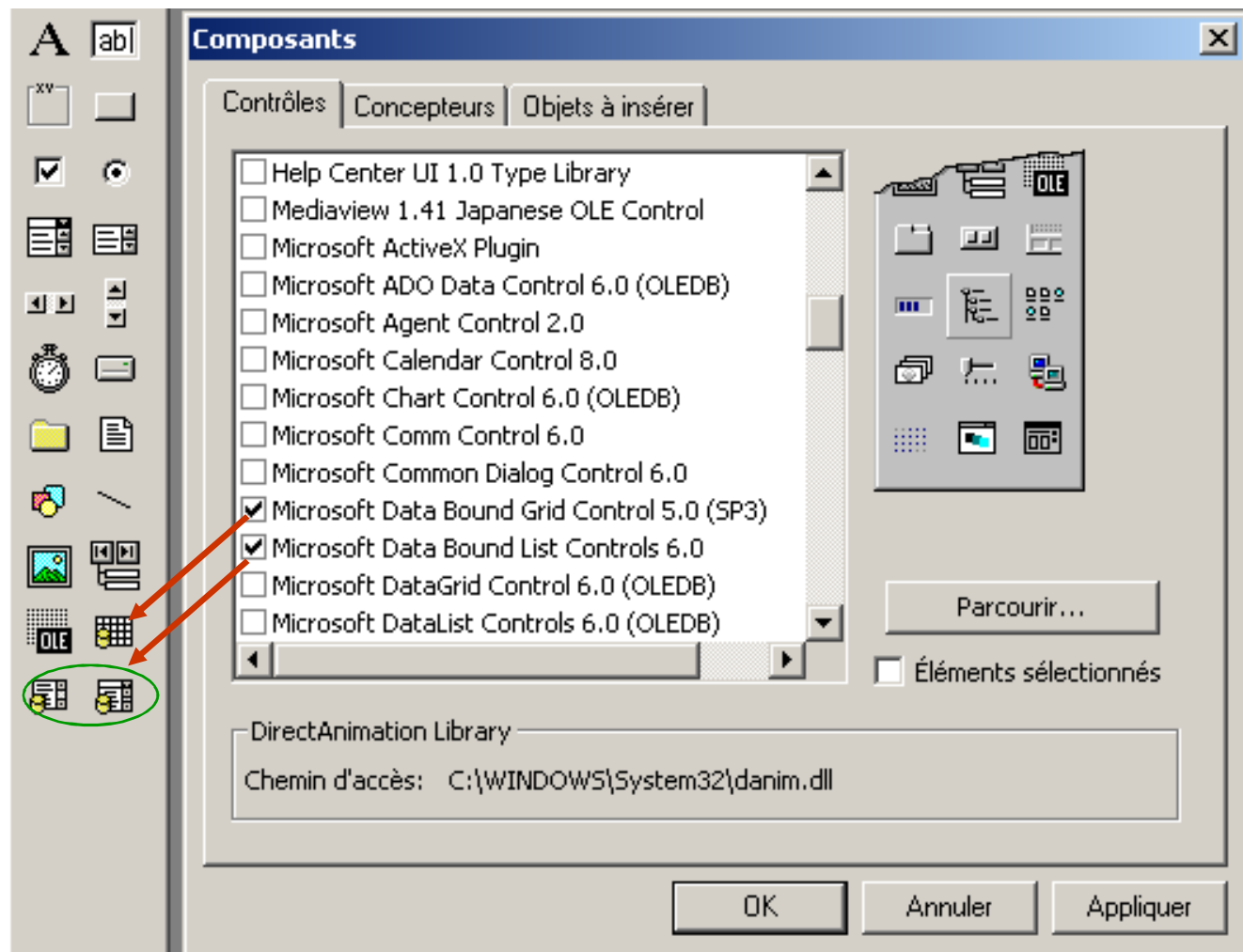
(ou bien : Menu Projet □ Composants,

Ou utiliser directement le raccourcis clavier : **Ctrl + t**)



Utilisation du contrôle Data

Cocher Microsoft Data Bound Grid et List



[Utilisation du contrôle Data]

La propriété Recordset : Créée automatiquement à partir de la propriété RecordSource (table ou requête), elle donne accès aux enregistrements

a- Ajout d'un nouvel enregistrement:

1. Utiliser la méthode **AddNew** (Data1.Recordset.**AddNew**)
2. **Modifier** le contenu des contrôles liées
3. Appeler la méthode **Update** pour enregistrer leur contenu
(Data1.Recordset.**Update**)

[Utilisation du contrôle Data]

b- Modification de l'enregistrement courant :

1. **Modifier** le contenu des contrôles liées (enregistrement courant)
2. Appeler la méthode **Update**

c- Suppression de l'enregistrement courant :

Appeler la méthode **Delete** du Recordset :

Data1.Recordset.**Delete**

Data1.Recordset.**MoveNext**

'pour se placer sur l'enregistrement suivant

Les propriétés **BOF** et **EOF** du Recordset peuvent être utilisées pour savoir si la position courante se trouve au début ou à la fin du Recordset

Exemple >>>

Utilisation du contrôle Data

Les contrôles liés Complexes :

Affichage du contenu de toute une table (ou requête) avec le contrôle **DBGrid** :



The screenshot shows a Delphi application window titled "Personnel". Inside the window is a DBGrid control displaying data from a table named "EMPLOYES". The table has five columns: "NumEmployé", "NomFamille", "Prénom", "Ville", and "Salaire". The data is as follows:

	NumEmployé	NomFamille	Prénom	Ville	Salaire
	2	dodo	didi	dakhla	6000
	6	kaka	koko	kénitra	6000
*					

Below the grid, there is a status bar with navigation buttons (first, previous, next, last) and a text field containing the word "Personnel".

[Utilisation du contrôle Data]

Le contrôle DBGrid :

Pour les différentes manipulations d'enregistrements, il suffit de mettre à **True** les propriétés correspondantes

a- Ajout d'un nouvel enregistrement :

AllowAddNew = True *(saisie dans la dernière ligne)*

b- Modification :

AllowUpdate = True

c- Suppression : *(sélection d'une ligne puis touche suppr)*

AllowDelete = True

Exemple >>>

[Utilisation du contrôle Data]

Les contrôles liés **DBList** et **DBCombo** présentent des propriétés complémentaires permettant un fonctionnement plus complexe: **ils autorisent la visualisation simultanée de toutes les lignes d'un champ du Recordset**

Propriétés :

RowSource : nom du contrôle Data

ListField : nom du champ de la table affiché dans la liste

Exemple >>>

[Utilisation du contrôle Data]

Aide à la saisie :

Employés

NumEmployé: 1

NomFamille: koto

Prénom: titi

Ville: tetouan

Salaire: 10000

numService: recherche

Saisie très difficile

Ajouter + Enregistrer Supprimer

Grille -->

[Utilisation du contrôle Data]

Aide à la saisie: Réalisation

Exemple >>>

1- Ajouter un 2eme contrôle Data

name = **dataSencondaire**

DatabaseName = **Entreprise** (la même base)

RecordSource = **Service** (table liée)

2- Remplacer le TextBox par un ComboBox

RowSource = **dataSecondaire** (source de données du contrôle)

ListField = **nomService** (champ qui contient le nom du service)

DataSource = **Data1** (contrôle qui est mis à jour (table employé))

DataField = **numService** (champ devant être mis à jour)

BoundColumn = **numService** (lien entre les deux tables)

[Utilisation du contrôle Data]

Les filtres (listes dépendantes)

Exemple >>>

Le principe est de changer la propriété **RecordSource** en cours d'exécution, actualiser le contrôle Data en appelant la méthode **Refresh** et actualiser le 2^{ème} DBCombo en appelant la même méthode

```
Private Sub DBCombo1_Click(Area As Integer)
```

```
    'Construction de la requête
```

```
    requete = "Select * FROM Ville WHERE numPays = ("
```

```
    requete = requete + "SELECT numPays FROM Pays WHERE  
                        nomPays = '" & DBCombo1.Text & "'")"
```

```
    Data2.RecordSource = requete
```

```
    Data2.Refresh
```

```
    DBCombo2.Refresh
```

```
End Sub
```

[Accès aux données avec ADO]

Pour pouvoir utiliser ADO dans un projet Visual Basic vous devez ajouter la bibliothèque [Microsoft ActiveX Data Objects 2.0 Library](#)

menu Projets □ Références,
puis cocher [Microsoft ActiveX Data Objects 2.0 Library](#)

ADO propose les objets suivants :

Recordset : jeu d'enregistrements retournés lors de l'exécution d'un
SELECT

Connection : connexion à une source de données (fichier texte,
fichier Excel, ou une base de données)

Error : ensemble des erreurs retournées par le SGBD

Command : permet d'exécuter des requêtes

Parameter : permet de définir un paramètre d'une requête

[Accès aux données avec ADO]

1- Création d'un DSN (Data Source Name)

- Dans le panneau de configuration de windows, sélectionnez *outils d'administration*, puis *Sources de données (ODBC)*
- Dans l'administrateur de source de données ODBC, sélectionnez *Sources de données utilisateur* (le 1^{er} onglet)
- Cliquez sur le bouton **ajouter**, puis sélectionner *Microsoft Access Driver(*.mdb)* et cliquer sur **terminer**
- La boîte de dialogue *installation ODBC Microsoft Access* est affichée. Entrez le nom de la source de données : *Gestion entreprise*
- Cliquez sur le bouton **sélectionner** et sélectionnez le fichier de votre base de données, le DSN *Gestion entreprise* est ajouté à la liste, cliquez ensuite sur **ok** *(Démonstration □)*

[Accès aux données avec ADO]

L'objet Recordset

La propriété **Fields** étant la propriété par défaut, les lignes suivantes sont équivalentes:

rs.Fields("nomService")

rs.Fields(1) *'le 1^{er} champ est d'indice 0*

rs ("nomService")

rs.Fields(1)

rs![nomService]

Exemple>>>

[Accès aux données avec ADO]

L'objet Recordset

L'exemple précédent n'est pas optimisé. En effet, une connexion à la base de données est établie chaque fois qu'une requête est générée : lors du remplissage initial de la liste, puis à chaque sélection d'un élément de la liste.

Problème : temps d'établissement d'une connexion peut devenir long, surtout si la base se trouve sur une autre machine

Solution : Etablir une connexion permanente indépendamment des requêtes en utilisant l'objet **Connection**

[Accès aux données avec ADO]

L'objet [Connection](#) : (déclaration dans le volet général !!)

```
Private cn As Connection
```

```
Private Sub Form_Load()
```

```
'création d'une connexion
```

```
Set cn = New Connection
```

```
'etablisement de la connexion
```

```
cn.Open "DSN=gestion entreprise"
```

```
'création du Recordset
```

```
Dim rs As New Recordset
```

```
'lien avec la connexion cn
```

```
Set rs.ActiveConnection = cn .....
```

Exemple>>>

[Accès aux données avec ADO]

A- Ajout d'un nouvel enregistrement :

```
dim rs As New ADODB.Recordset
rs.Open "Select * from Service ","DSN=gestion entreprise",_
        adOpenDynamic, adLockOptimistic
rs.AddNew
rs![numService] = 50
rs![nomService] = "Ressources humaine"
rs.Update
```

Remarque : **adOpenDynamic** c'est pour permettre l'écriture sur la base (*le mode par défaut étant la lecture seule*)

[Accès aux données avec ADO]

B- Modification de tous les enregistrements :

```
dim rs As New ADODB.Recordset
rs.Open "Select * from Article ", "DSN=gestion entreprise", _
        adOpenDynamic, adLockOptimistic
```

```
dim stock as integer
Do While Not rs.EOF
    stock = rs![stock]
    rs![stock] = stock - 10
    rs.Update
    rs.MoveNext
Loop
```

[Accès aux données avec ADO]

C- Suppression de tous les enregistrements :

```
dim rs As New ADODB.Recordset
rs.Open "Select * from Article ","DSN=gestion entreprise",_
        adOpenDynamic, adLockOptimistic

Do Until rs.EOF
    rs.Delete
    rs.MoveNext
Loop
```

Remarque : pour supprimer un enregistrement particulier il faut ajouter un filtre dans la requête (clause WHERE)

[Accès aux données avec ADO]

Le contrôle ADODC (ADO DataControl) :

Le contrôle ADODC est semblable au contrôle Data présenté précédemment

Pour l'ajouter au projet, il faut ajouter le composant :
Microsoft ADO Data Control 6.0 (OLEDB)

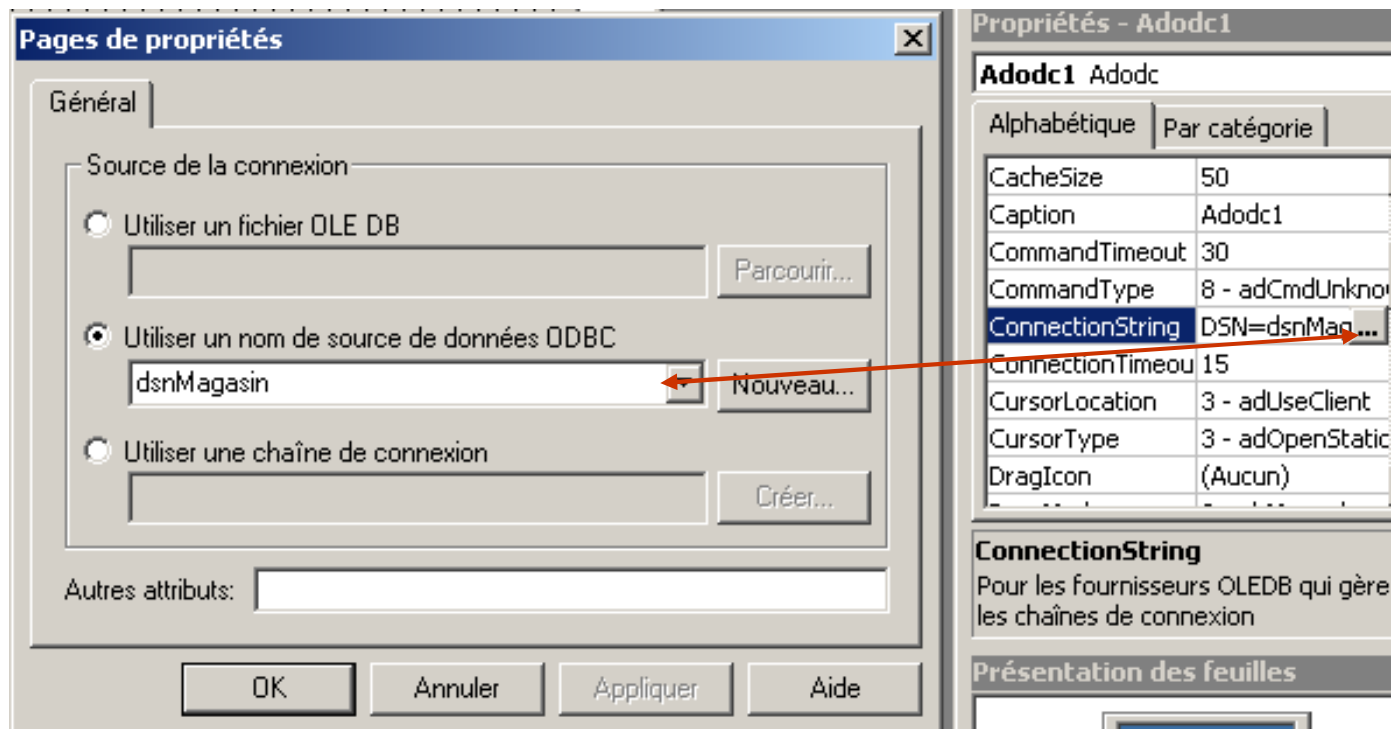


[Accès aux données avec ADO]

Le contrôle ADODC :

Initialisation :

ADODC1.ConnectionString = "DSN=gestion entreprise"



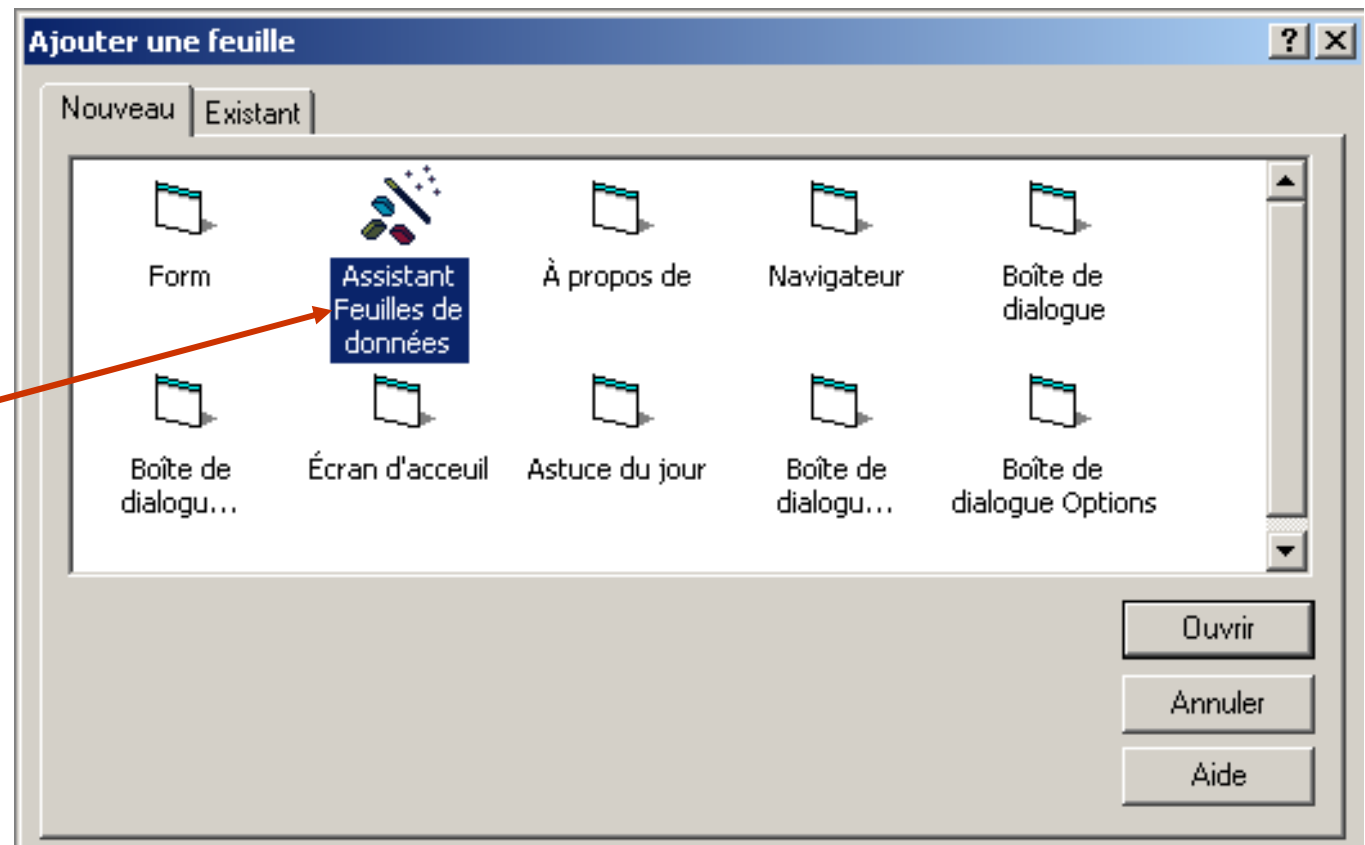
[Accès aux données avec ADO]

Utilisation de l'*Assistant Feuille de données*

Menu fichier,
puis nouvelle
feuille.

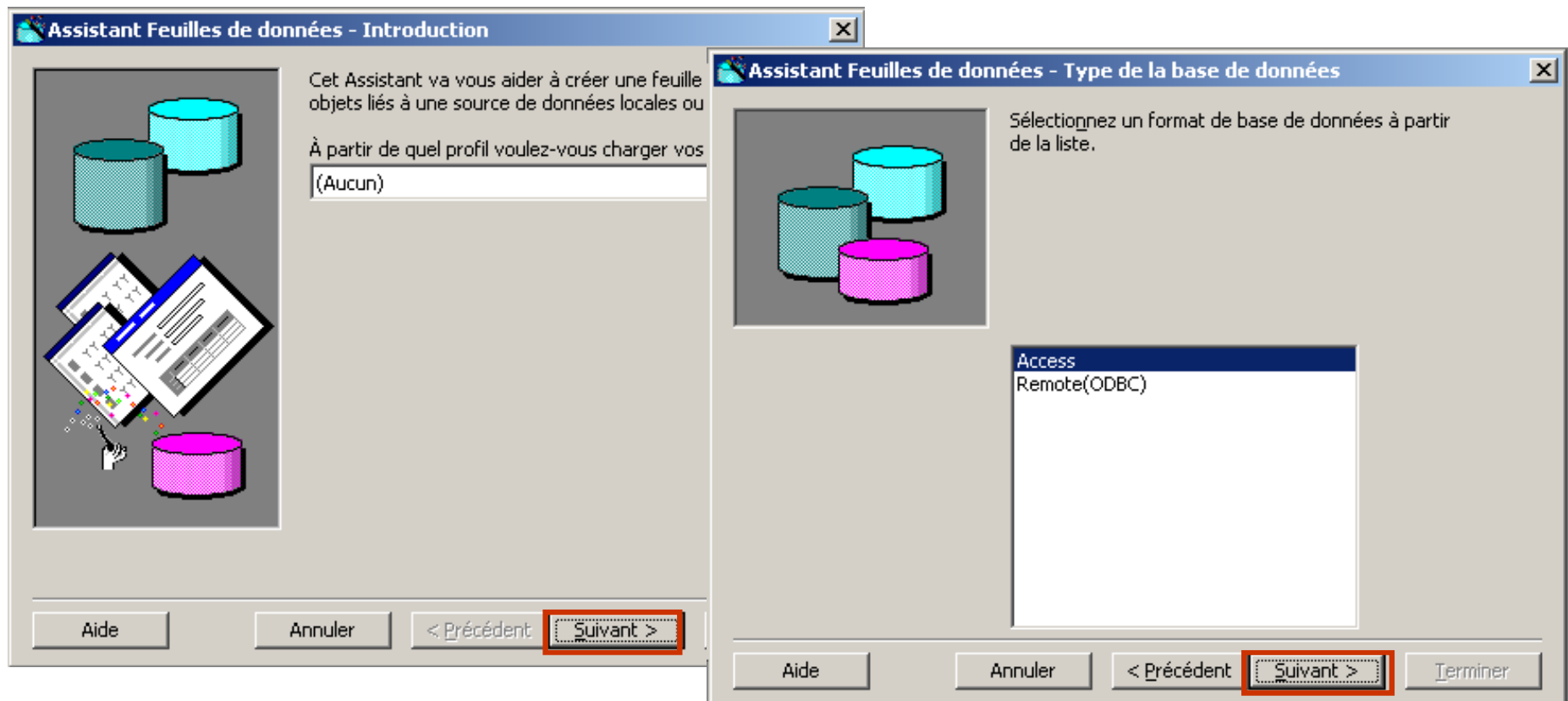
Choisissez
*Assistant Feuille
de données*

Cliquez sur *Ouvrir*



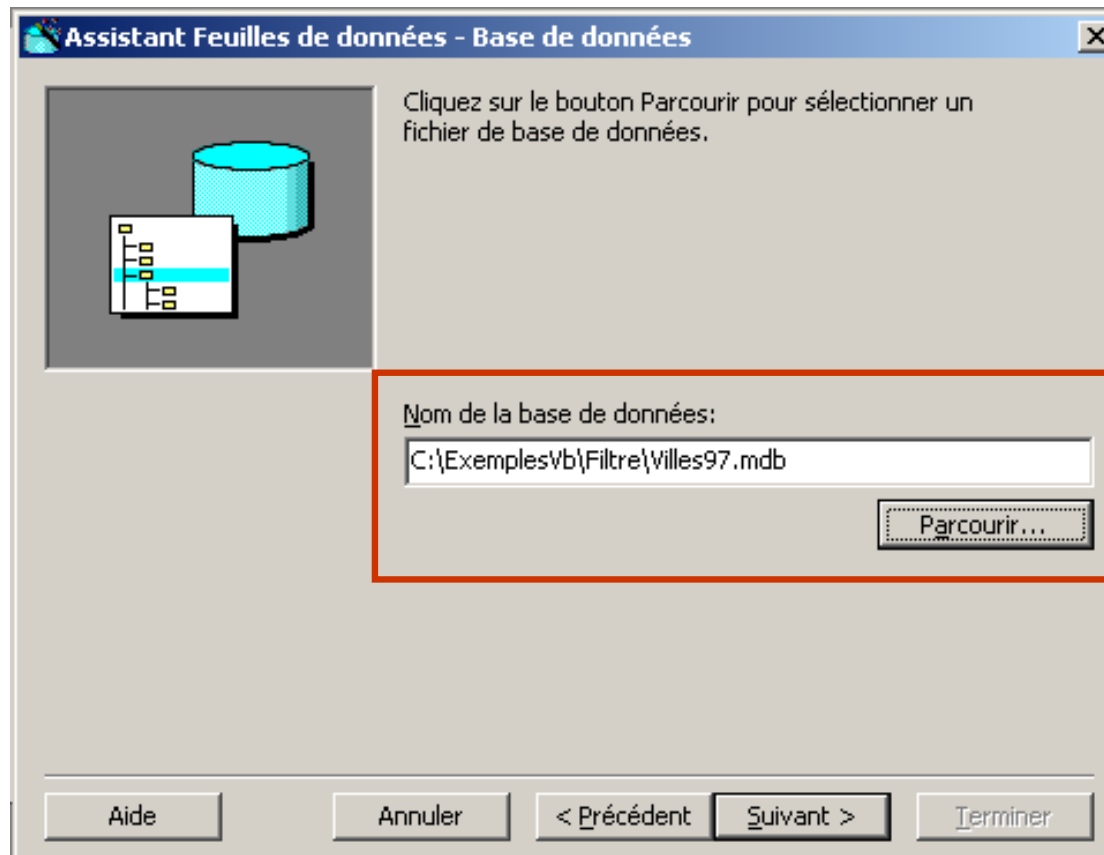
[Accès aux données avec ADO]

Utilisation de l'Assistant Feuille de données



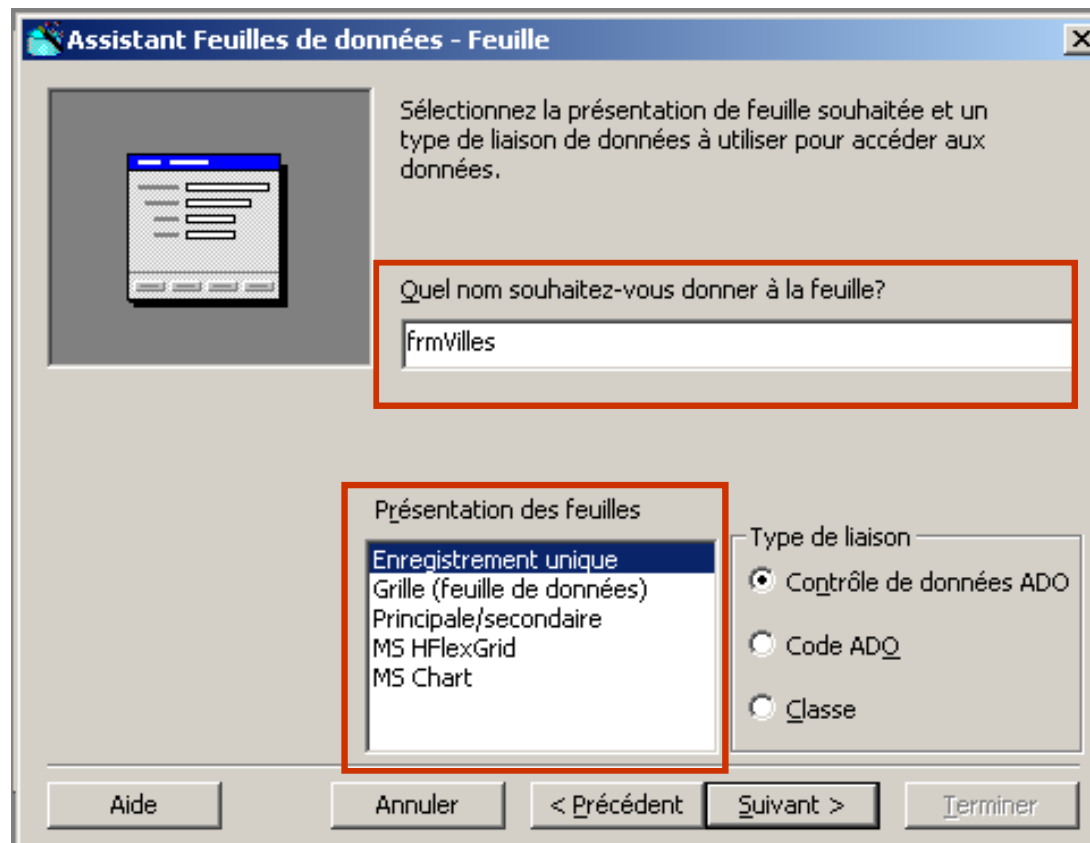
[Accès aux données avec ADO]

Utilisation de l'*Assistant Feuille de données*



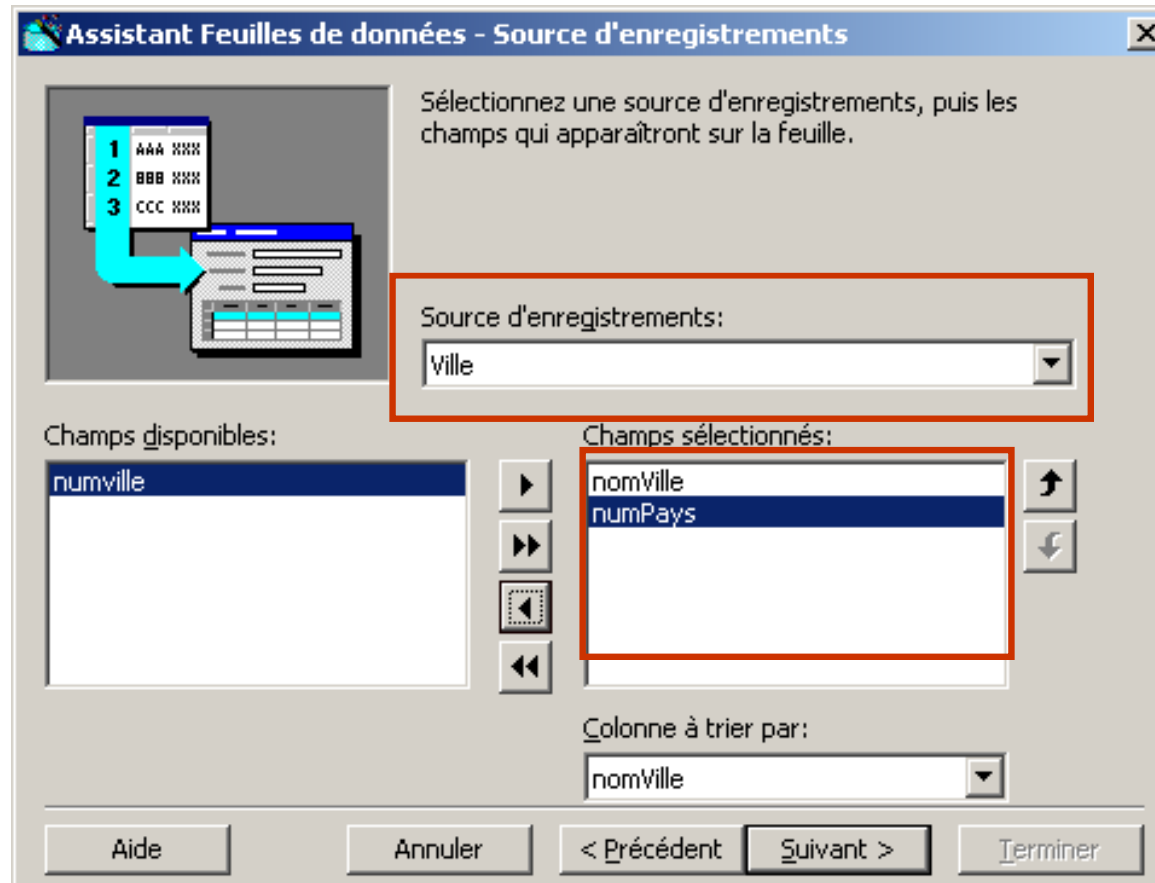
[Accès aux données avec ADO]

Utilisation de l'Assistant Feuille de données



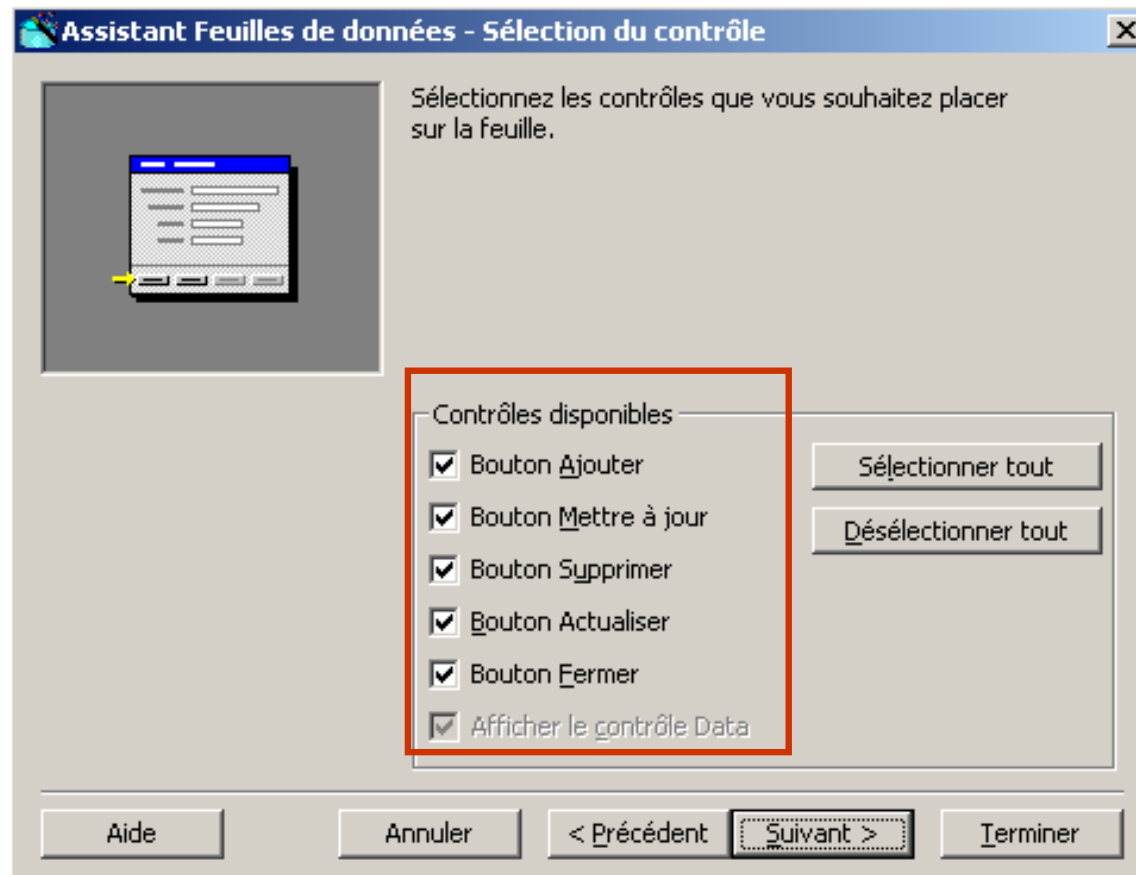
[Accès aux données avec ADO]

Utilisation de l'Assistant Feuille de données



[Accès aux données avec ADO]

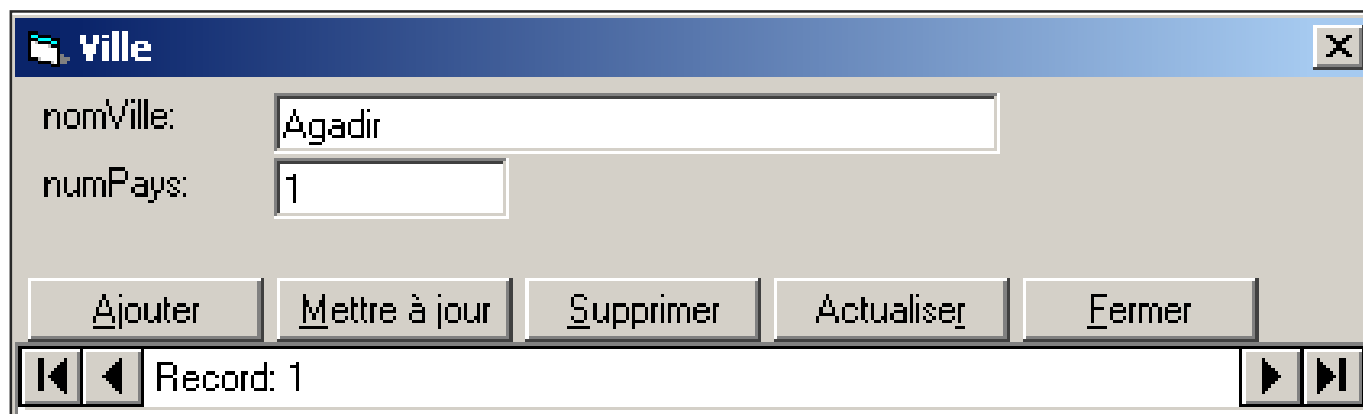
Utilisation de l'Assistant Feuille de données



[Accès aux données avec ADO]

Utilisation de l'*Assistant Feuille de données*

Résultat :



The screenshot shows a Windows-style dialog box titled "Ville". It contains two input fields: "nomVille:" with the text "Agadir" and "numPays:" with the value "1". Below these fields are five buttons: "Ajouter", "Mettre à jour", "Supprimer", "Actualiser", and "Fermer". At the bottom, there is a record navigation bar with left and right arrow buttons, the text "Record: 1", and additional right and left arrow buttons.

Environnement de Développement graphique

(Visual Basic)

FIN

Téléchargement cours + exemples + TP :

Adresse: etudiants3eme@yahoo.fr

Mot de passe: insea2006