

Visual Basic.Net

EISTI – Ing 2 IFI

Olivier Favre

Charles Villette

Sommaire

- I. Visual Studio 2003 – L'EDI
- II. Syntaxe : Variables, Tableaux, Fonctions et procédures
- III. Un peu d'IHM
- IV. Se débrouiller seul
- V. Syntaxe : Boucles, Gestion d'erreurs
- VI. IHM
- VII. Le débogage
- VIII. Exercices
- IX. Accès aux données
- X. Exercices sur les données

Visual Studio 2003 L'EDI

The screenshot displays the Visual Studio 2003 IDE with a Windows Forms application in design mode. The interface is annotated with red numbers 1 through 6, highlighting key components:

- 1**: The central design surface showing a form with a `TextBox1` and a `Calcular fact` button.
- 2**: The Solution Explorer on the right, showing the project structure for `WindowsApplication1`, including `References`, `AssemblyInfo.vb`, and `Form1.vb`.
- 3**: The Properties window on the right, showing the properties for the selected `Button1` control, such as `AccessibleName`, `Appearance`, `Font`, `Image`, `Text`, and `TextAlign`.
- 4**: The Output window at the bottom, which is currently empty.
- 5**: The Toolbox on the left, showing the list of Windows Forms controls available for design.
- 6**: The menu bar and toolbar at the top of the IDE.

Visual Studio 2003

L'EDI

- **1** – Zone Visuel + code
- **2** – Votre projet – La hiérarchie des classes
- **3** – Propriétés des objets
- **4** – Output : débog et code d'erreurs
- **5** – Toolbox (uniquement en mode dessin)
- **6** – Boite générale : Tout est accessible depuis cette boite.

Démarrage rapide : Exemple !

- Création d'un nouveau projet
- Double clic sur la fenêtre → du code !

Syntaxe : Variables

- Règles habituelles :
 - commence par une lettre
 - Pas d'espaces ni caractères spéciaux sauf _
- `Dim` *NomDeVariable* `As` *Type* =
affectation facultative
- `NomDeVariable` = *valeur*
- `NomDeVariable` = `Nothing`

Syntaxe :

Fonctions et procédures

- **Différence** : une fonction renvoie une valeur
- **Public|Private|Protected Sub**
Nom(param AS Type)
' Code
End Sub
- **Public|... Function** *Nom(Optional*
param AS Type = default) AS
TypeDeRetour
' Code
End Function

Syntaxe :

Valeur de retour

- `Public Successeur(n As Integer) As Integer`
 `Successeur = n+1`
 `Successeur = Successeur * 10`
 `Exit Function` ' ou `Return`, sans valeur
 Commandes non exécutées
 `End Function`
- `Public Successeur(n As Integer) As Integer`
 `Return 10*(n+1)`
 `End Function`

Un peu d'IHM

- Dépôt d'un bouton
- Changement d'une propriété
- Double clic → un événement « click »

Se débrouiller tout seul, déjà !

- IntelliSense
 - Ctrl+Space : Auto-complétion
 - Ctrl+Shift+Space
- L'Explorateur d'Objets

Onglet Object Explorer (Ctrl+Alt+J)

The screenshot displays the Visual Studio IDE with the following components highlighted:

- Object Explorer:** Shows the project structure under 'Microsoft.VisualBasic'. The 'Interaction' folder is expanded, and 'MsgBoxStyle' is selected.
- Find Symbol Dialog:** A dialog box titled 'Find Symbol' is open. The 'Find what:' field contains 'MsgBox'. The 'Look in:' dropdown is set to 'Active Project'. The 'Options' section has 'Match whole word' selected. The 'Find' button is visible.
- Members of 'Interaction':** A list of methods and properties is shown. The entry 'MsgBox(ByVal Object, [ByVal Microsoft.VisualBasic.MsgBoxStyle], [ByVal Object]) As Microsoft.VisualBasic.MsgBoxResult' is highlighted with a red box.
- Find Symbol Results:** A window at the bottom shows the search results for 'MsgBox'. It lists one match: 'MsgBox(ByVal Object, [ByVal Microsoft.VisualBasic.MsgBoxStyle], [ByVal Object]) As Microsoft.VisualBasic.MsgBoxResult (Microsoft.VisualBasic.Interaction)'. The text is highlighted with a red box.
- Documentation Panel:** A detailed view of the 'MsgBox' function is shown, including its signature, summary, and parameters. The text is highlighted with a red box.

Public Function `MsgBox`(ByVal *Prompt* As **Object**, Optional ByVal *Buttons* As **Microsoft.VisualBasic.MsgBoxStyle** = 0, Optional ByVal *Title* As **Object** = Nothing) As **Microsoft.VisualBasic.MsgBoxResult**
Member of: **Microsoft.VisualBasic.Interaction**

Summary:
Displays a message in a dialog box, waits for the user to click a button, and then returns an integer indicating which button the user clicked.

Parameters:
Prompt: String expression displayed as the message in the dialog box. The maximum length of Prompt is approximately 1024 characters, depending on the width of the characters used. If Prompt consists of more than one line, you can separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or a carriage return-linefeed character combination between each line.
Buttons: Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If you omit Buttons, the default value is zero.
Title: String expression displayed in the title bar of the dialog box. If you omit Title, the application name is placed in the title bar.

Find Symbol Results - 1 match found
◆ `MsgBox`(ByVal Object, [ByVal Microsoft.VisualBasic.MsgBoxStyle], [ByVal Object]) As Microsoft.VisualBasic.MsgBoxResult (Microsoft.VisualBasic.Interaction)

Gestion d'erreurs : On Error

- On Error Resume Next
- On Error Goto EtiquetteGestionErreur
' Code
Exit Function|Sub ' à ne pas oublier !
EtiquetteGestionErreur:
If MsgBox("Erreur n°" & Err.Number & " : "
& Err.Description, MsgBoxStyle.Critical +
MsgBoxStyle.AbortRetryIgnore, "Erreur !")
= MsgBoxResult.Ignore Then
Resume Next
End If

Gestion d'erreurs : Try – Catch, Throw

- Plus rapide que `On Error`, mais pas utilisables dans une même méthode

- `Try`
`y = 1 / x` ' attention à la division par 0
`Catch ex As ArithmeticException`
`MsgBox("Erreur : " & ex.Message())`
`Catch ex As Exception`
`MsgBox("Erreur Inconnue : " & ex.Message())`
`Finally`
' Code toujours exécuté !
`End Try`

- `Throw New Exception("Mon exception !")`

Éléments d'IHM

- Button
- Label
- TextBox
- Menu
- CheckBox, RadioButton
- ListBox
- ProgressBar
- ...

Se débrouiller seul bis : Le Débogage

→ Breakpoints

→ Définition, Désactivation, Conditions

→ Watches

→ Auto

→ Locals

→ Watch

→ Call Stack

Exercice 1

- Calculatrice de factorielles

- TextBox, NumericSpinner, TrackBar

- Button de calcul

- Labels d'affichage

- MsgBox de bienvenue avec l'heure

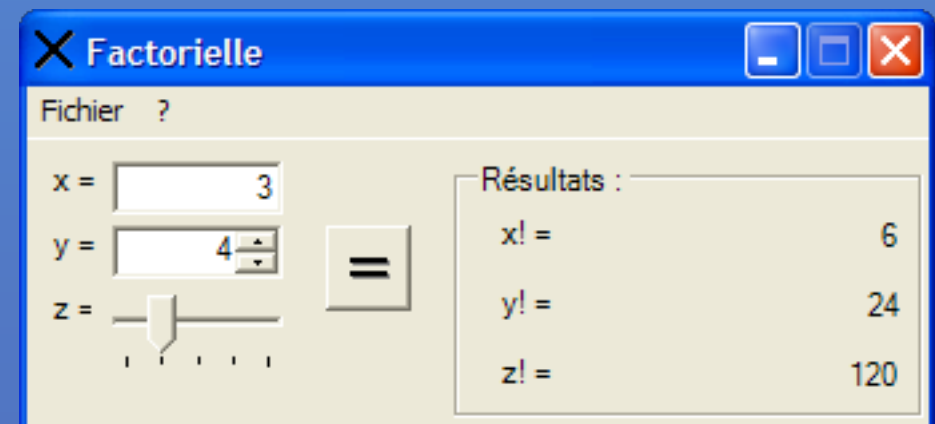
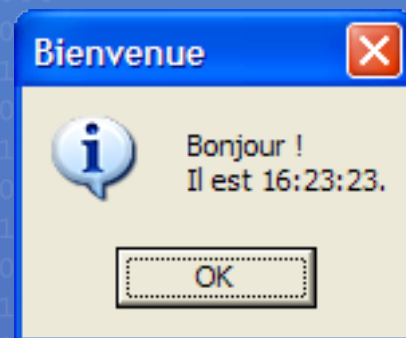
- Menu Fichier/Quitter (Ctrl+Q) et ?/À propos (F1)

- Tester avec « toto » et « 100 »

- Gestion des erreurs

Bonus :

- Icône dans un Button
- Icône de fenêtre
- Icône d'exécutable



Syntaxe : Tableaux

- **Dim** MonTableau(n) *As Type*
- ' **De 0** à **MonTableau.Length-1**
- **Dim** DimensionImpliciteObligatoirement()
As Type = { liste, de, valeurs, facultatives }
- MonTableau(x) = MonTableau(y)
- **Redim** [**Preserve**] MonTableau(x)
- **Erase** MonTableau
- **Dim** MultiDimensionnel(n1, n2) *As Type*

Encore un peu de syntaxe :

While, Do – Loop

- `while` *Condition*
' Code
Exit while
' Non exécuté !
End while
- `Do while|until` *Condition*
' Code
Exit Do 'pour l'exemple
Loop
- `Do`
' Code au moins exécuté une fois
Exit Do 'pour l'exemple
Loop `while|until` *Condition*

Encore un peu de syntaxe :

For, For Each

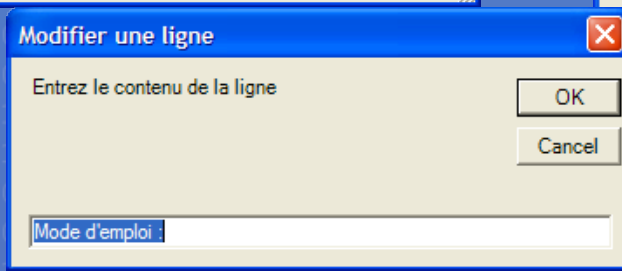
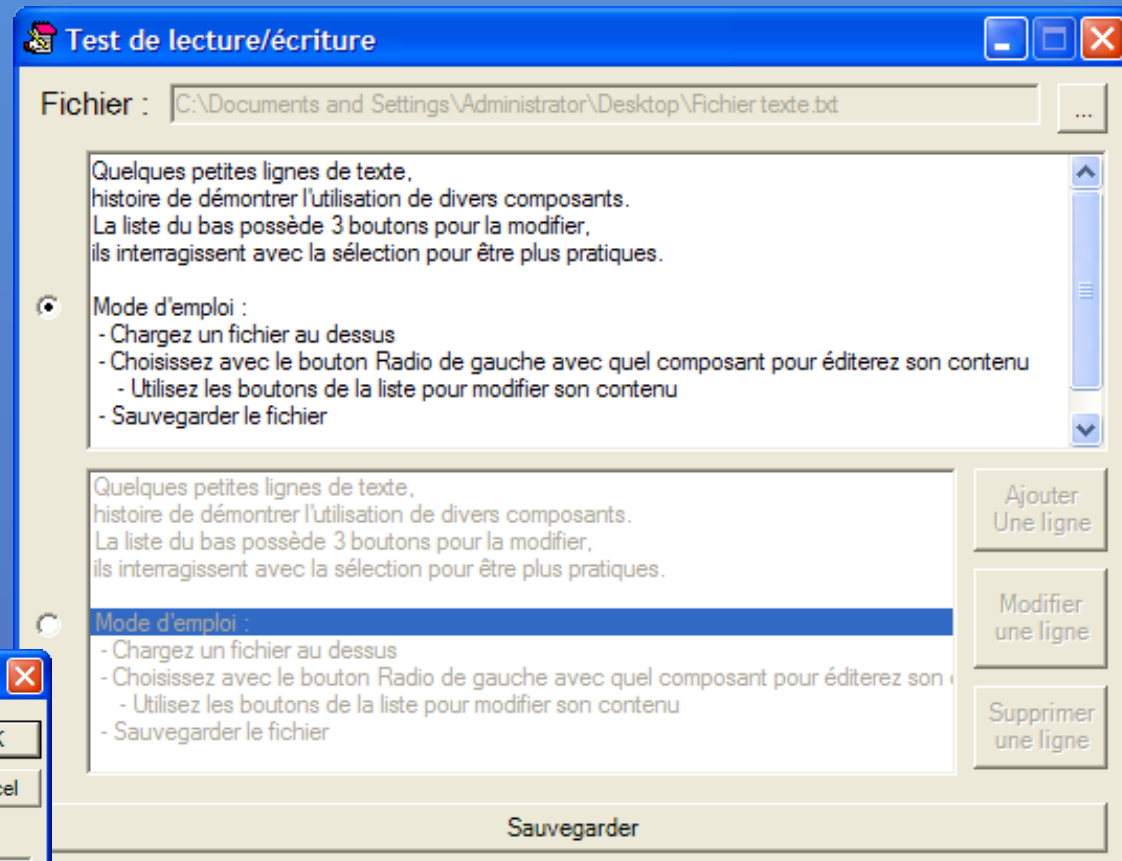
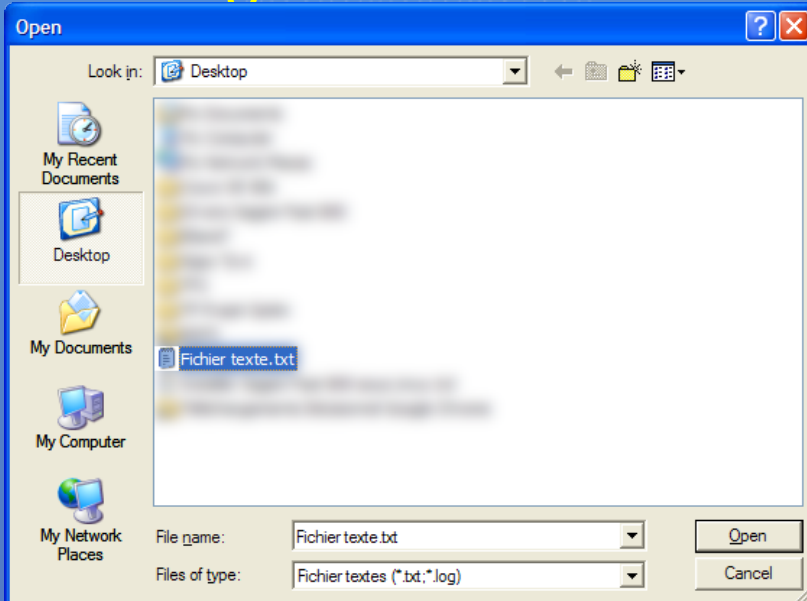
- `Dim i As Integer`
`For i = 0 To 10`
`' Exit For, encore pour`
`l'exemple`
`Next [i]` 'on peut remettre le nom
de la variable
- `Dim e As TypeElement`
`For Each e In collectionDElements`
`' Exit For, idem`
`Next [e]` 'nom facultatif aussi

Exercice 2

- Ouvrir un fichier
- Visualisation et modification dans un :
 - TextBox : Activer le MultiLine et le ScrollBars Both
 - List : Utiliser 3 boutons : Ajouter/Modifier/Supprimer une ligne
- Enregistrer le fichier

Bonus :

- Mettre une icône à la fenêtre et à l'application
- Faire une procédure Main
- Dossier par défaut : Desktop



Accès aux données : Connexion

- Se connecter à une BDD Access :

```
Dim cn As New OleDbConnection
```

```
cn.ConnectionString =
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;"
```

```
cn.ConnectionString &= "DataSource=" & _  
& "C:\maBdd.mdb"
```

```
cn.Open()
```

Accès aux données : Requête

- Lire une source de données :
- ```
Dim Reader As OleDbDataReader
Reader = New OleDbCommand(_
 "SELECT * FROM client",cn _
).ExecuteReader()
Do While Reader.Read
 MsgBox(Reader.GetValue(0))
Loop
Reader.Close()
```

# Accès aux données : Requête

- Modifier/Ecrire une source de données

```
New OleDbCommand("INSERT INTO
Client VALUES(...)",
cn).ExecuteNonQuery()
```

- N'ayant pas besoin de retour (hormis une erreur éventuelle), il n'y a pas plus à faire pour modifier/écrire.

# Accès aux données : Fermeture

- **NE JAMAIS OUBLIER** de fermer la connexion à la base de données :
- `cn.Close()`

# Exercice 3

- Connexion à une base Access :
  - Connexion à une base de données depuis un fichier
  - Création d'une table
  - Ajout de données
  - Exécution d'une requête

## Bonus :

- Traitement dans une procédure déléguée
- Barre d'état
- Icônes à partir des ressources

