DEUST IOSI FORMATION CONTINUE

LE LANGAGE

VISUAL BASIC

SOMMAIRE

1 - GENERALITES

A - PRELIMINAIRE

Communiquer → Langage (parlé, écrit, par signes …)

Langage → ensemble de caractères, de symboles et de règles permettant de les assembler, dans le but de communiquer :

- Langages naturels : celui des hommes, des animaux,…
- Langages artificiels : utilisés pour simplifier la communication (pictogrammes) et surtout en programmation informatique.

Programmer → écrire dans un langage de programmation informatique, une suite d'instructions, organisée en algorithme dans un but précis, exécutable par un ordinateur.

B - LANGAGES DE PROGRAMMMATION

- Langage machine
  Le seul compréhensible par la machine.
  Assemblage de 0 et de 1 (bits).
  Complexe à mettre en œuvre →***Domaine de spécialistes***

- Le langage d'assemblage (ou Assembleur)
  Permet de développer des programmes proches des instructions de base d'un microprocesseur.
  Exemple :
  ```
  Message DB 'Bonjour' , '$';     met la chaîne dans une zone mémoire
  MOV AH,09h;                     charge le registre A pour afficher
  ```

- Langages évolués (*programmation linéaire*)
  Le programmeur écrit des lignes d'instructions proches du langage naturel.
  Ce code source est ensuite :
  - soit exécuté ligne à ligne par un interpréteur,
  - soit traduit en langage machine par un compilateur avant l'exécution.

  Le programme se déroule de façon linéaire en respectant la structure suivante :
  - il y a un point d'entrée pour le programme,
  - il y a un point de sortie,
  - entre les deux, on trouve l'exécution de l'application,
  - des appels à des procédures ou fonctions peuvent être faits.

  Exemple :
  PASCAL : programmation procédurale. (Begin Write ('Bonjour') End.)
  PROLOG : programmation logique.
  LISP : programmation fonctionnelle.

- Langages objets ou orientés objets (*programmation événementielle)*
  Le programmeur écrit des procédures indépendantes les unes des autres.
  Le code source n'est pas compilé mais interprété ().
  Exemple :
  C et C++ : programmation de logiciels.
  VISUAL BASIC : programmation graphique événementielle.
  JAVA : récent, portable, voisin du C++

C - METHODE DE PROGRAMMATION

- **Spécification des besoins** des futurs utilisateurs.
- **Spécifications fonctionnelles** : comment satisfaire aux besoins.
- **Conception générale** : division du logiciel en programmes.
- **Conception détaillée** : algorithme le plus adapté pour chaque programme.
- **Assemblage** des différents programmes.
- **Codage** à l'aide du langage le plus adapté.
- **Validation** et qualification.

La conception est beaucoup plus importante que le codage qui peut être sous-traité dans le cas de gros logiciels.

2 - CARACTERISTIQUES DE VISUAL BASIC

A - PRESENTATION

- Ancien **BASIC** (**B**eginner's **A**ll purpose **S**ymbolic **I**nstruction **C**ode)
- **Programmation par objets** (briques logicielles)
- **Programmation graphique** (fenêtres, icônes, menus, souris…)
- **Programmation événementielle** (sollicitations : souris, clavier, autre événement…)
- **Réutilisable** (modules de code BASIC)

B - STRUCTURE GENERALE

Une application Visual Basic est constituée d'un ensemble de procédures indépendantes les unes des autres. Une procédure comprend des instructions écrites à l'aide du langage BASIC. Elle est associée à un **objet**, c'est-à-dire à un des éléments d'une feuille (bouton, liste, champ de saisie, …).

Une application Visual Basic n'est pas ***compilée***, mais ***interprétée*** (dès qu'une ligne de code a été saisie, il est possible de lancer l'exécution de l'application).

- Un programme VB est appelé **projet** (.VBP) et se compose de plusieurs éléments (réutilisables):
  - **un module** (.BAS) : regroupe les déclarations et les instructions non directement liées à une feuille particulière (facultatif),
  - **une feuille (ou plusieurs)** (.FRM) : composée de
    → d'un outil de création de fenêtre,
    → d'un éditeur pour l'écriture du code.
- Le développement d'une application passe par les étapes suivantes :
  - **dessin de l'interface d'utilisation**, c'est-à-dire les fenêtres et leurs constituants, à l'aide d'un outil interactif de dessin.
  - **valorisation initiale des propriétés** qui sont des attributs ou caractéristiques de chaque élément de l'interface,
  - **écriture du code en Basic.**
- Les objets manipulés sont appelés des **contrôles** (bouton de commande, boîte de dialogue, zone de texte, zone d'image, etc.). Chaque contrôle peut réagir à des événements qui lancent des suites d'instructions codées en BASIC.
- **Conseil**
  Avant de commencer votre projet, il est conseiller de créer **un dossier** sous Windows afin d'y ranger tous les éléments qui le compose : module, forms, images, etc.

C - MODE DE FONCTIONNEMENT

Au lancement → feuille de démarrage.

Puis l'environnement de Visual Basic peut se trouver dans un des trois modes suivants.

- **Le mode création**
  C'est le mode initial pendant lequel on peut construire l'application :
  - créer des feuilles,
  - placer des contrôles,
  - définir les propriétés,
  - écrire le code.

- **Le mode exécution**
  Il permet de tester l'application en la faisant fonctionner.

- **Le mode arrêt**
  C'est le mode dans lequel passe un application interrompue suite à une erreur ou à une demande explicite de l'utilisateur.

  Rq. :Ne pas oublier d'arrêter un programme, sinon toute correction est impossible (notamment pour un programme qui boucle…)

- **Création d'un exécutable**
  Lorsque les tests sont terminés, il est possible de produire un fichier exécutable (.EXE) sous Windows.
  Tout programme VB impose l'emploi d'au moins une bibliothèque de liens dynamiques (.DLL)

D - LES FENETRES DE VISUAL BASIC

1 - Fenêtre principale
- barre de titre,
- barre de menus,
- barre d'outils.

2 – Boite à outils
Accessible uniquement en mode création.

3 - Fenêtre de projet
Contient la liste des divers fichiers d'une application.

4 - Fenêtre de propriétés
Contient la liste des propriétés de l'objet sélectionné ainsi que leurs valeurs.

5 – Palette de couleurs
Accessible uniquement en mode création.

6 - Fenêtre de débogage
Accessible uniquement en mode exécution.

7 – Feuilles de travail
Fenêtres de l'application en cours de conception.
("Form1" est une feuille de travail)

8 - Fenêtre de code
Permet la visualisation ou la saisie d'instructions en Basic.

3 - LE LANGAGE BASIC

A - NOTION DE VARIABLES

- Les variables sont utilisées pour stocker (mémoriser) une valeur réutilisable.

- C'est une zone mémoire qui porte un nom choisi par le programmeur. Ce nom doit commencer par une lettre avec un maximum de 255 caractères (non autorisés espace . ! @ & $ #)

  → utiliser des noms significatifs

  SOM, PRENOM, SALAIRE : valide
  NB DE LIVRE, 1er : non valide

- Le langage BASIC présente une originalité par rapport à de nombreux autres langages : il autorise l'utilisation de variables sans imposer une déclaration préalable. Ainsi la simple utilisation d'une variable dans une ligne de code génère sa déclaration **implicite**.
  Mais il est possible de déclarer explicitement les variables avant leur utilisation (conseillé). Cette déclaration peut se faire n'importe où dans le code d'un programme (il est cependant conseillé de les regrouper).
  Elle se fait à l'aide de l'instruction **Dim** et n'est valide que pour cette procédure. L'existence et la valeur de la variable disparaissent à la fin de la procédure.

- Variables de portées particulières,
  - **Static** garde la valeur de la variable lors du lancement de l'application (à déclarer dans une procédure),
  - **Public** utilisable dans toute les procédures de la feuille (à déclarer dans un module),
  - **Private** utilisable dans une seule procédure.

  Remarque : Si une variable est déclarée dans la section des déclarations d'un module elle est valide dans toutes les procédures du module.

- Variables **globales** ou **locales** :
  - **Globales** : variables disponibles dans toute procédure et tout module d'un programme (doivent être déclarées dans le module),
  - **Locales** : variables connues uniquement à l'intérieure de la procédure où elle est déclarée.

Remarque

Pour obliger la déclaration des variables, il suffit d'ajouter la commande suivante dans le programme :

Option Explicit

B - TYPE DES VARIABLES

Les principaux types de variables sont :

Type	Suffixe	Contenu	Taille
Integer	%	-32768 à 32767	2 octets
Long	&	-2 147 483 648 à 2 147 483 647	4 octets
Single	!	-3.042823E38 à 3.042823E38	4 octets
Double	#	-1.79769313486232E308 à 1.79769313486232E308	8 octets
Currency	@	4 décimales fixes	8 octets
String	$	65 400 octets (16 bits) à 2^{31} octets (32 bits)	1 octet par caractère
Byte		entier de 0 à 255	1 octet
Boolean		True ou False	2 octets
Date		1 janvier 100 à 31 décembre 9999	8 octets
Type		Exemple : Type ETUDIANT NOM as string * 20 PRENOM as string * 15 CLASSE as string * 5 End Type	selon la structure

Les variables les plus utilisées sont :

- le type ***String*** (chaîne de caractères),
- le type ***Integer*** (entier relatif)
- le type ***Single*** (décimal).

C - EXEMPLE DE DECLARATION

Syntaxe : **Dim** <NomVariable> **As** <Type>
Global <NomVariable> **As** <Type>

Exemples
```
Dim I as Integer
Dim Taux As Single
Global Mot-Initial As String
```

Remarque Il est possible d'utiliser les suffixes pour les déclarations.

```
Dim I%              (équivalent à Dim I as Integer)
Dim Taux!           (équivalent à Dim Taux As Single)
Global Mot-Initial$ (équivalent à Global Mot-Initial As String)
```

D - LES CONSTANTES

Si une variable dans le code contient une valeur qui ne change pas, elle sera appelée **constante**

Syntaxe : **Const** <NomVariable> **=** valeur

Exemples
```
Const PI = 3.141592 ' Valable dans une procédure
Public Const PI = 3.141592 ' Valable dans toutes les procédures
                           si placé dans un module
```

E - LES COMMENTAIRES

Pour la lisibilité du code on peut apporter du commentaire après une apostrophe (').
→ en couleur verte dans le code

Exemples

```
Dim Taux As Single   ' Taux de la TVA
Dim Réponse As String   ' Mot proposé par l'utilisateur
Global Mot-Initial As String ' Premier mot à traiter
```

F - INITIALISATION DES VARIABLES

Pour éviter tout problème lors de l'exécution de votre programme, il est préférable d'initialiser les variables déclarées

Exemples

```
COMPTEUR = 0
TAUX = 20.6
NOM = "TOTO"
```

4 - LES OBJETS (OU CONTROLES)

A - OBJETS ET PROPRIETES

- **Un objet** peut posséder un grand nombre de propriétés que l'on peut modifier avec l'outil de création graphique ou par programmation. Par exemple :
  - sa forme,
  - sa couleur,
  - sa position dans la feuille,
  - sa visibilité,
  - etc.

- **Principaux objets**
  - ***Form*** (feuille)
    conteneur graphique des contrôles de l'application,
  - ***CommandButton*** (bouton de commande)
    exécute le code associé à l'événement click sur le bouton
  - ***Label*** (étiquette)
    affiche une information (texte, nombre, date, …) sans permettre la saisie par l'utilisateur
  - **Image** (image)
    affiche des image en mode point (BitMap au format .BMP, .WMF, .ICO)
  - ***TextBox*** (zone de texte)
    champ de saisie de texte
  - ***CheckBox*** (case à cocher)
    permet la saisie d'une option de type binaire (oui/non).
  - ***OptionBox*** (bouton d'option)
    utilisé en groupe pour autoriser le choix d'une option parmi plusieurs possibles.
  - ***ComboBox*** (liste combinée)
    combinaison d'un champ de saisie et d'une liste simple.
  - ***ListBox*** (liste simple),
    choix d'une option parmi plusieurs, en nombre variable.
  - ***Timer*** (minuterie)
    n'est pas visible en mode d'exécution, mais permet de générer des événements à une périodicité donnée.

- **Les propriétés**

  Syntaxe : **Objet.Propriété =** valeur

  La plus importante est la propriété **Name** qui donne un nom au contrôle. Ce nom permet de référencer le contrôle.

  Exemples :
  - Etiquette. BackColor = &HFF0000 ' Bleu
  - Affichage.Caption = "Bonjour"
  - Image.Visible = True
  - Quitter.Enabled = False

- **De bonnes habitudes**
  - Dès qu'un objet est créé sur une feuille, lui donner un nom significatif
    → propriété **Name**
  - Préfixer selon nomenclature suivante :

Objet	Préfixe	
Form	Frm	Frm_Accueil
Command	cmd	Cmd_Oui
Label	lbl	Lbl_Result
Image	img	Img_Result
TextBox	txt	Txt_Nb1

B - OBJETS ET EVENEMENTS

- Les objets réagissent à divers éléments : Click, Change, MouseUp, MouseDown
- Le code d'un événement associé à un objet, forme une procédure événementielle.

Syntaxe :
```
Sub NomContrôle_Evénement()
    Instruction 1
    Instruction 2
    ……
    Instruction N
End Sub
```

Exemple1 :
```
Sub Cmd_Quitter_Click()
  Unload Me
End Sub
```

Exemple 2 :
```
Sub Txt_Nb1_Change()
  NB1 = Text1
End Sub
```

5 - LES INSTRUCTIONS DE BASE

A - ENTREE DES DONNEES

Si l'utilisateur fournit une donnée, il faut la stocker dans une variable afin de la réutiliser autant de fois que l'on veut. Le plus simple est d'utiliser la boite de dialogue prédéfinie **InputBox**.

Syntaxe : Variable = **InputBox (Prompt, Titre_fenêtre, Val_Défaut, X, Y)**

Prompt : texte
Titre-Fenêtre : titre de la fenêtre (facultatif)
Val_Défaut : valeur par défaut (facultatif)
X : position de la fenêtre en abscisse (facultatif)
Y : position de la fenêtre en ordonnée (facultatif)

Exemples :

- NOMBRE = **InputBox (**"Entrez un nombre", "Addition"**)**
- NOM = **InputBox (**"Indiquez votre nom",,"DUPONT",1,1**)**

B - AFFICHAGE DES DONNEES

Pour afficher un message non interactif, on utilise la boite de dialogue prédéfinie **MsgBox**. C'est un simple message affiché dans une boîte agrémentée d'un bouton OK

Syntaxe : **MsgBox (Prompt, VbMsgBoxStyle, Titre-Fenêtre)**

Prompt : texte, variable, "texte"&variable
(**&** symbole de la **concaténation)**
VbMsgBoxStyle : élément de présentation de la boite de dialogue (facultatif)
Titre-Fenêtre : nom de la fenêtre (facultatif)

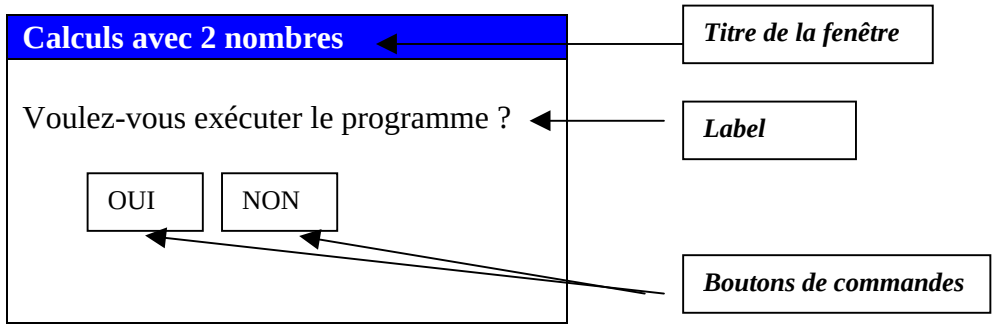
Exemples :

- MsgBox (TOTAL)
- MsgBox ("Bienvenu à tous", VbInformation, "Accueil IOSI")
- MsgBox ("Vous avez "&AGE&" ans")

C – PREMIER PROGRAMME

Utilisation des Inputbox

1. **Création de la Form** (Label + CommandButton) suivante



Le bouton de commande **OUI** permet d'exécuter le programme :

- saisie des données (par des InputBox),
- calcul des résultats,
- affichage des résultats (par des MsgBox).

Le bouton de commande **NON** permet d'arrêter le programme.

2. **Ecriture du code**

```
Dim NB1, NB2 As Integer
Option Explicit
```

```
Private Sub OUI_Click()
    NB1 = Val(InputBox("1er nombre"))
    NB2 = Val(InputBox("2ème nombre"))
    MsgBox ("L'addition est : " & (NB1 + NB2))
    MsgBox ("La soustraction est : " & (NB1 - NB2))
End Sub
```

```
Private Sub NON_Click()
    Unload Me            'Permet de décharger la Form
End Sub
```

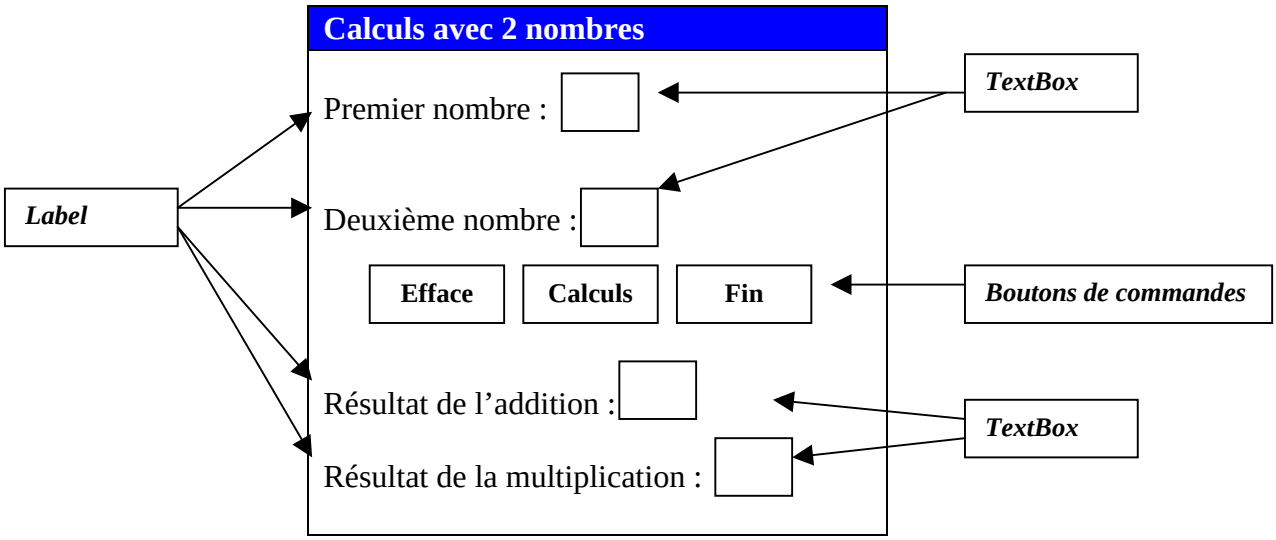
3. **Fonction utilisée**

Val (chaîne) : convertit une chaîne de caractères contenant des nombres en une valeur numérique.

D – AUTRE METHODE

Utilisation des TextBox

1. Création de la Form (Label + TextBox + CommandButton) suivante



La saisie des données se fait avant l'exécution du programme grâce à l'utilisation des *TextBox*.
Le bouton de commande **Efface** permet d'effacer les valeurs de la form.
Le bouton de commande **Calculs** permet d'exécuter le programme :

- calcul des résultats,
- affichage des résultats dans des *Textbox* qui seront verrouillées.

Le bouton de commande **Fin** permet d'arrêter le programme.

2. Ecriture du code

```
Dim NB1, NB2 As Integer
Option Explicit
```

```
Private Sub Calcul_Click()
   Text3 = NB1 + NB2
   Text4 = NB1 * NB2
End Sub
```

```
Private Sub Efface_Click()
   Text1 = ""
   Text2 = ""
   Text3 = ""
   Text4 = ""
End Sub
```

```
Private Sub Fin_Click()
   Unload Me
End Sub
```

```
Private Sub Text1_Change()
   NB1 = Val(Text1)
End Sub
```

```
Private Sub Text2_Change()
   NB2 = Val(Text2)
End Sub
```

E - STRUCTURES CONDITIONNELLES

1. La structure If

Les instructions à exécuter peuvent dépendre d'une (ou plusieurs) condition (s). Il faut alors utiliser une structure décisionnelle qui oriente le déroulement du programme vers des blocs d'instructions déterminés.
C'est la structure **If … Then … Else … End**.

- Syntaxe sur une seule ligne: **If** condition **Then** instruction1 **Else** instruction2
  condition : expression dont le résultat est vrai ou faux
  (**True** ou **False**)
  Else instruction2 : facultatif

  Exemple :
  If Moyenne >=10 **Then** Résultat = "Admis" **Else** Résultat = "Ajourné"
  If Moyenne >=10 **Then** Résultat = "Admis"

- Syntaxe sous forme de bloc

```
If condition Then
    Bloc d'instruction1
Else
    Bloc d'instruction2
End If
```

  Exemples :

  -
```
If Moyenne >=10 Then
    Admis = Admis + 1
    MsgBox ("Etudiant Admis")
Else
    Ajourné = Ajourné + 1
    MsgBox ("Etudiant Ajourné")
End If
```

  -
```
If Moyenne >=10 Then
    Admis = Admis + 1
        If Moyenne > 14 Then
            MsgBox ("Etudiant Admis avec mention")
        Else
            MsgBox ("Etudiant Admis")
        End If
Else
    Ajourné = Ajourné + 1
    MsgBox ("Etudiant Ajourné")
End If
```

2. La structure Case

Au-delà de 3 possibilités on a besoin d'une autre structure qui peut gérer plusieurs cas. C'est la structure **Select … Case …**
C'est une extension du **If**. Elle permet une programmation plus claire en évitant une trop grande imbrication de If successifs.

Syntaxe :

```
Select Case Expression
        Case ListeValeurs1
                Bloc d'instruction1
        Case ListeValeurs2
                Bloc d'instruction2
        …….
        [Case Else
                Bloc d'instruction N]
End Select
```

Expression : ou variable
ListeValeurs : peut être :
- une suite de valeurs : 1, 3, 5, 7, 9
- une fourchette : 0 To 9
- une plage : Is >= 10

Case Else
Bloc d'instruction N : facultatif

Une seule Expression (ou Variable) est testée au début, puis est comparée avec les listes de valeurs. A la première concordance, le bloc d'instruction correspondant est exécuté, puis le programme sort de la structure.
Si aucune concordance n'est trouvée les instructions placées après le Else sont exécutées.

Exemples :

```
Select Case CodeASCIICaractère
        Case 65, 69, 73, 79, 85
                MsgBox(" C'est une voyelle ")
        Case 66 To 90
                MsgBox(" C'est une consonne ")
        Case Else
                MsgBox(" Ce n'est pas une lettre ")
End Select
```

3. La structure IIf

C'est exactement la fonction IF d'EXCEL.

Syntaxe :

```
IIf (Condition, ValeurSiVrai, ValeurSiFaux)
```

Exemple :

```
Dim Note As Single
Dim Réponse As String
Note = InputBox (" Tapez votre note ")
Réponse = IIf (Note >= 10, " Admis ", " Ajourné ")
MsgBox (Réponse)
```

F - COMPLEMENTS SUR MSGBOX

Lorsque plusieurs boutons de commande sont affichés par la MsgBox, il est possible de faire un traitement en fonction du bouton de commande choisi. Pour cela, il est nécessaire d'utiliser une variable qui recevra le résultat du clic.

Exemples :

- ```
  Dim REP As String
  REP = MsgBox("Confirmation ?", vbYesNo)
  If  REP = vbYes  Then
      MsgBox ("Traitement OUI")
  Else
      MsgBox ("Traitement NON")
  End If
  ```

- ```
  Dim REP As String
  REP = MsgBox("Confirmation ?", vbYesNoCancel)
  Select Case REP
     Case vbYes
          MsgBox ("Traitement OUI")
     Case vbNo
          MsgBox ("Traitement NON")
     Case vbCancel
          MsgBox ("Traitement ANNULE")
  End
  ```

G - LES BOUCLES

1. La structure itérative

Le programme va exécuter un bloc d'instructions de façon indéfinie. L'arrêt se fera lorsque la condition ne sera plus vérifiée.

Syntaxe : **While** condition
Bloc d'instructions
Wend
La condition est testée au début, c'est à dire à l'entrée de la boucle.
La boucle sera répétée tant que la condition est vraie.
Si la condition n'est pas vraie au départ, le bloc d'instructions ne sera pas exécutée.

Exemple :
```
Mot = InputBox("Donnez votre mot de passe")
While Mot <> MotDePasse
    Mot = InputBox("Donnez votre mot de passe")
Wend
```

2. La structure répétitive

Le programme va exécuter un bloc d'instructions de façon indéfinie. L'arrêt se fera lorsque la condition sera vérifiée.

Syntaxe : **Do**
Bloc d'instructions
Loop Until condition
La condition est testée après avoir exécuté une fois le bloc d'instruction.
La boucle sera répétée tant que la condition est fausse.

Exemple :
```
Do
    Mot = InputBox("Donnez votre mot de passe")
Loop Until Mot = MotDePasse
```

3. La structure Pour

Le programme va exécuter un bloc d'instructions de façon définie. L'arrêt se fera lorsque la valeur finale sera atteinte.

Syntaxe : **For variable =** valeur_initial **To** valeur_finale **[Step** incrémentation]
Bloc d'instructions
Next
La variable est incrémentée à chaque fin de boucle du nombre indiqué par l'incrément. Si l'incrément n'est pas spécifié, il est fixé à 1.
Si la valeur finale est inférieure à la valeur initiale, l'incrément est négatif.

Exemple :
```
For Ind = 1 to 25
    MsgBox(Ind)
Next
```

6 - TRAITEMENT DES CHAINES DE CARACTERES

Une variable chaîne de caractères se déclare de type String..

Exemples :

- Dim MotProposé As String
  → la variable contient alors une chaîne de longueur variable
- Dim Lettre As String * 1
  → la variable contient alors une chaîne de longueur 1 (1 caractère)
- Dim Adresse As String * 30
  → la variable contient alors une chaîne de longueur 30 :
  - si 18 caractères dans la variable, le reste est à blanc,
  - si plus de 30 caractères, la variable est tronquée

A - COMPARAISON DES CHAINES DE CARACTERES

- La longueur d'une chaîne est donnée par la fonction **Len**.

  Exemple :

  ```
  Dim Phrase as String
  Phrase = "Université de Valenciennes"

  L'instruction Len(Phrase) retournera la valeur _____.
  ```

- On compare deux chaînes de caractères par la fonction **StrComp**. Elle renvoie
  la **valeur 0** si les deux chaînes sont rigoureusement identiques,
  la **valeur 1** si les chaînes diffèrent même par un seul octet.

  Exemple :

  ```
  Dim Phrase1, Phrase2, Phrase3 as String
  Phrase1 = "Université de Valenciennes"
  Phrase2 = "Universite de Valenciennes"
  Phrase3 = "Université de Valenciennes"

  L'instruction StrComp(Phrase1,Phrase2) retournera la valeur _____.
  L'instruction StrComp(Phrase1,Phrase3) retournera la valeur _____.
  ```

B - RECHERCHE D'UNE CHAINE DE CARACTERES

- La fonction **InStr** permet de rechercher si une chaîne de caractères existe à l'intérieur d'une autre chaîne de caractères. Cette fonction retourne la position de la première occurrence de la chaîne recherchée.

  Syntaxe: **InStr**(Chaîne1, Chaîne2)
  Chaîne1 : chaîne de caractères à traiter (sur laquelle porte la recherche),
  Chaîne2: chaîne de caractères recherchée dans Chaîne1.

  Exemple :

```
Dim Chaine1, Chaine2, Chaine3 as String
Chaine1 = "Université de Valenciennes"
Chaine2 = "IUT"
Chaine3 = "Val"
```

  L'instruction InStr(Chaîne1,Chaîne2) retournera la valeur _____.
  L'instruction InStr(Chaîne1,Chaîne3) retournera la valeur _____.

- La fonction **Ucase** met une chaîne de caractères en majuscules.

  Syntaxe: **Ucase**(Chaîne)

  Exemple :

```
Dim Phrase as String
Phrase = "Université de Valenciennes"
```

  L'instruction Ucase(Phrase) retournera la valeur : _____

- La fonction **Lcase** met une chaîne de caractères en minuscules.

  Syntaxe: **Lcase**(Chaîne)
  Exemple :

```
Dim Phrase as String
Phrase = "Université de Valenciennes"
```

  L'instruction Lcase(Phrase) retournera la valeur : _____

C - EXTRACTION D'UNE CHAINE DE CARACTERES

- La fonction **Right** donne la partie droite d'une chaîne de caractères. Le nombre de caractères de cette partie doit être précisé.

  Syntaxe: **Right**(Chaîne, Nombre)
  Chaîne : chaîne de caractères à traiter,
  Nombre: nombre de caractères à partir de la droite.

  Exemple :
  Dim Chaîne as String
  Chaîne = "Université de Valenciennes"

  L'instruction Right(Chaîne,5) retournera la valeur _____.

- La fonction **Left** donne la partie gauche d'une chaîne de caractères. Le nombre de caractères de cette partie doit être précisé.

  Syntaxe: **Left**(Chaîne, Nombre)
  Chaîne : chaîne de caractères à traiter,
  Nombre: nombre de caractères à partir de la gauche.

  Exemple :
  Dim Chaîne as String
  Chaîne = "Université de Valenciennes"

  L'instruction Left(Chaîne,7) retournera la valeur _____.

- La fonction **Mid** extrait une partie d'une chaîne de caractères. Le nombre de caractères de cette partie doit être précisé ainsi que la position du premier caractère extrait.

  Syntaxe: **Mid**(Chaîne, Position, Nombre)
  Chaîne : chaîne de caractères à traiter,
  Position : position dans la chaîne, à partir de laquelle il faut extraire des caractères,
  Nombre: nombre de caractères à extraire.

  Exemple :
  Dim Chaîne as String
  Chaîne = "Université de Valenciennes"

  L'instruction Mid (Chaîne, 4, 3) retournera la valeur _____.
  L'instruction Mid (Chaîne, 25, 7) retournera la valeur _____.
  L'instruction Mid (Chaîne, 30, 2) retournera la valeur _____.

D - APPLICATIONS

1. Exemple1

Que fait ce programme le programme ci-dessous ?
Remplacer les éléments de l'InputBox et des MsgBox par des textes significatifs.

```
Dim T1, T2 As String
Dim I As Integer

T1 = "Pour qui sont ces serpents qui sifflent sur vos têtes ?"
T2 = InputBox ("              ")
I = InStr(T1, T2)
If  I = 0  Then
     MsgBox("Pas OK !")
Else
     MsgBox("OK en" & I)
End If
```

2. Exemple2

Que fait ce programme le programme ci-dessous ?
Compléter la MsgBox de la fin de programme.

```
Dim Phrase, C As String
Dim Cp, L, I As Integer

Phrase = InputBox("Tapez votre phrase")
Cp = 0
L = Len (Phrase)
For I = 1 To L
     C = Mid(Phrase, I, 1)
     If C = " " Then Cp = Cp + 1
Next
MsgBox("Cette phrase _____ " & Cp & "_____.")
```

7 - GESTION DES FEUILLES

A - CHARGEMENT

- **LOAD** : permet de charger une feuille sans l'activer. Elle n'est pas visible, il faut ensuite réaliser un **SHOW**.

  Syntaxe : **Load** nom de la feuille

  Exemple : Load Form1

- **SHOW** : permet d'activer et de rendre visible la feuille.

  Syntaxe : nom de la feuille.**Show**

  Exemple : Form1.Show

  Si l'on désire charger et activer en même temps, le SHOW seul est suffisant.

B - DECHARGEMENT

- **UNLOAD** : permet décharger la feuille et désactiver en même temps.

  Syntaxe : **Unload** nom de feuille

  Exemple : **Unload** Form1

  A noter que la feuille active peut être déchargée par **Unload Me** (Me désignant la feuille active).
  Set Form1 = **Nothing** permet en plus de libérer la mémoire.

- **HIDE** : permet de cacher la feuille sans la désactiver.

  Syntaxe : nom de feuille.**Hide**

  Exemple : Form1.**Hide**

C - DIMENSIONS

Exprimées par les propriétés ci-dessous, en **Twips** (1cm = 567 Twips) ou en **cm** en modifiant l'unité par la propriété ***ScaleMode***

- **Left** : distance au bord gauche de l'écran.
- **Top** : distance au bord supérieur de l'écran.
- **Height** : hauteur de la feuille.
- **Width** : largeur de la feuille.

Exemples

```
Form1.Left = 100
Form1.Top = 200
Form1.Height = Screen.Height/2      la moitié de l'écran
Form1.Width = Screen.Width/3        un tiers de l'écran
Form1.Height = Screen.Height        taille de l'écran
Form1.Width = Screen.Width          taille de l'écran
```

D - PROPRIETES

- **BorderStyle** : définit le type de bordure.
  0 = aucune bordure
  1 = bordure simple
  2 = bordure de changement de taille
  3 = bordure à double filet
  4 = boite à outils sans changement de taille
  5 = boite à outils avec changement de taille

- **Caption** : Texte de la barre de titre.

- **MinButton** : bouton de réduction.
  MinButton = False -→ bouton inaccessible
  MinButton = True -→ bouton apparent

- **MaxButton** : bouton de réduction.
  MaxButton = False -→ bouton inaccessible
  MaxButton = True -→ bouton apparent

- **ControlBox** : existence des boutons systèmes.
  ControlBox = False -→ pas de boutons
  ControlBox = True -→ boutons apparents

- **Moveable** : feuille déplaçable.
  Moveable = False -→ pas déplaçable
  Moveable = True -→ déplaçable

- **BackColor** : Couleur de remplissage du fond

- **Picture** : dessin du fond.

E - EVENEMENTS CARACTERISTIQUES

- **LOAD** : affichage au chargement de la feuille.

```
Private Sub Form_Load()
    Load Form2
    Form2.Height = Screen.Height
    Form2.Width = Screen.Width
    Form2.Show
End Sub
```

- **ACTIVATE** : activation de la feuille à l'affichage.

```
Private Sub Form_Activate()
    Load Form2
    Form2.Height = Screen.Height
    Form2.Width = Screen.Width
    Form2.Show
End Sub
```

- **CLICK** : activation de la feuille par un clic de la souris.

```
Private Sub Form_Click()
        Load Form2
        Form2.Height = Screen.Height
        Form2.Width = Screen.Width
        Form2.Show
End Sub
```

- **KEYPRESS** : activation de la feuille par appui sur une touche du clavier

```
Private Sub Form_KeyPress(KeyAscii As Integer)
        Load Form2
        Form2.Height = Screen.Height
        Form2.Width = Screen.Width
        Form2.Show
End Sub.
```

8 - LES TABLEAUX

A - STRUCTURE D'UN TABLEAU

On a souvent besoin de travailler sur un ensemble de données de même type.

Exemples :

- Températures moyennes des 12 mois de l'année
  → on pourrait déclarer 12 variables identiques
- Numéros du loto
  → on pourrait déclarer 7 variables identiques (6 + 1 pour le compl.)

Pour éviter des déclarations multiples, on peut utiliser une structure de données appelée **tableau** qui permet de conserver dans une seule structure plusieurs valeurs de même type. Celle-ci sera parcourue par une variable de type entier, appelée **indice**.

Exemples : soit un tableau TAB de longueur 10

1	2	3	4	5	6	7	8	9	10
Val1	Val2	Val3	Val4	Val5	Val6	Val7	Val8	Val9	Val10

L'accès à la case numéro 7 se fait par TAB(7) qui vaut Val7

Déclarations :

- **Dim** Nom(Taille) **As** Type
  → 1ère case indice 0, dernière case indice Taille
- **Dim** Nom(1 **To** Taille) **As** Type
  → 1ère case indice 1, dernière case indice Taille

La déclaration d'un tableau avec **Public** est incompatible, il faut utiliser **Dim**.

Exemples :

- Dim TabTemp(1 To 12) As Single

1	2	3	4	5	6	7	8	9	10	11	12
6	5,5	7	11,5	15	17	22	21	19	14	10	8

L'accès à la case numéro 5 se fait par TabTemp(5) qui vaut 15

- Dim TabLoto (1 To 7) As Integer

1	2	3	4	5	6	7
6	49	25	14	47	17	22

B - REMPLISSAGE D'UN TABLEAU

Pour remplir un tableau, on le balaye avec une boucle For … To … Next, car le nombre de cases à parcourir est connu d'avance.

Exemple 1 : Températures moyennes

```
Dim TabTemp(1 To 12) As Single
Dim Cpt As Integer

For Cpt = 1 To 12
    TabTemp(Cpt)=InputBox("Température N° " & Cpt)
Next
```

Exemple 2 : Tirage du loto

```
Dim TabLoto(1 To 6) As Integer
Dim Cpt As Integer

For Cpt = 1 To 6
      TabLoto (Cpt)=Rnd * 49
Next
' On supposera que tous les nombres tirés sont différents !
```

C - TRAITEMENT DES VALEURS D'UN TABLEAU

On suppose saisies les 12 températures dans le tableau TabTemp. On veut rechercher la température maximale dans ce tableau de 12 valeurs.

Il s'agit donc de balayer ce tableau et de conserver la valeur maximale dans une variable. Au départ, on suppose que la température maximale est la première du tableau.

```
Dim Cpt As Integer
Dim Maxi As Single

Maxi = TabTemp(1)
For Cpt = 2 To 12
     If TabTemp(Cpt)>Maxi Then Maxi = TabTemp(Cpt)
Next
```

A la fin du processus la variable Maxi contiendra la valeur recherchée.

D - AFFICHAGE D'UN TABLEAU

Il est possible d'afficher un tableau grâce à une **ListBox**

Exemple : Affichage de 10 prénoms saisis

```
Dim TabNom(1 To 10) As String
Dim I As Integer

For I = 1 To 10
    TabNom(I)=InputBox("Entrer prénom N° " & I)
Next
LstNom.Clear                          'liste vide
For I = 1 To 10
    LstNom.AddItem TabNom(I)          'ajout d'un élém. dans la liste
Next
```

9 - DATE ET HEURE

A - FONCTIONS DE DATE

- La fonction **Date** donne la date système sous la forme jj/mm/aa.
- La fonction **Day(Variable)** donne le numéro du jour de la date contenue dans la variable.
  → si VarDate = "02/11/01" alors Day(VarDate) = 2
  → Day(Date) donne le numéro du jour d'aujourd'hui
- La fonction **Month(Variable)** donne le numéro du mois de la date contenue dans la variable.
  → si VarDate = "02/11/01" alors Month(VarDate) = 11
  → Month(Date) donne le numéro du mois d'aujourd'hui
- La fonction **Year(Variable)** donne le numéro de l'année de la date contenue dans la variable.
  → si VarDate = "02/11/01" alors Year(VarDate) = 01
  → Year(Date) donne le numéro l'année d'aujourd'hui
- La fonction **WeekDay(Variable)** donne le numéro du jour de la semaine de la date contenue dans la variable (Dimanche porte le numéro 1)
  → si VarDate = "02/11/01" alors WeekDay(VarDate) = 6 (Vendredi)
  → WeekDay(Date) donne le numéro du jour d'aujourd'hui : 4 (Mercredi)

B - FONCTION D'HEURE

- La fonction **Time** donne l'heure système sous la forme hh:mm:ss.

C - L'OBJET TIMER

L'objet **Timer** (minuterie) permet de tester un événement qui se produit à des intervalles de temps fixe.

Il apparaît en mode construction, mais il est invisible à l'exécution. La propriété **Enabled** permet de le mettre en marche (True) ou de l'arrêter (False).

Lorsqu'il est en marche, l'événement Timer se produit à intervalles réguliers. Cet intervalle de temps est indiqué dans la propriété **Interval** de l'objet (exprimé en millisecondes). C'est un entier positif de 0 à 64767 (1 minute maxi).

Exemple :

```
Private Sub Timer1_Timer()
    Label1.Caption = "Minuteur en cours " & Time
End Sub
```

Le programme affichera "Minuteur en cours" dans un Label au bout de 2000 millisecondes (à préciser dans les propriétés)

Il peut y avoir au maximum **16 minuteries**.

10 - LES SOUS-PROGRAMMES

A - LES PROCEDURES - LES ACTIONS NOMMEES

Une procédure est un ensemble d'instructions qui participent à une même tâche.

Elle débute par le mot réservé **Sub** et se termine par **End Sub**.

Exemple de procédure événementielle :

```
Private Sub CmdQuitter_Click
    Unload Me
End Sub
```

Si un bloc d'instructions doit être utilisé à plusieurs endroits (par exemple dans plusieurs procédures événementielles) il est préférable d'en faire une procédure publique qui sera utilisable dans toute la feuille.

Exemple de procédure publique :

```
Public Sub SaisieNote()
        Do
            Note = InputBox("Tapez une note")
        Loop Until Note>=0 And Note<=20
End Sub
```

Pour utiliser cette procédure il suffira de l'appeler par son nom : SaisieNote

B - LES FONCTIONS

Une fonction est aussi un ensemble d'instructions mais qui retourne une valeur contenue dans le nom même de la fonction. Cette valeur doit être affectée au nom de la fonction avant la fin du bloc d'instructions.

Elles débutent par le mot réservé **Function** et se terminent par **End Function**. Il faut aussi préciser le type de la valeur retournée.

Exemple de fonction :

```
Public Function Carré(x) As Single
        Carré = x * x
End Function
```

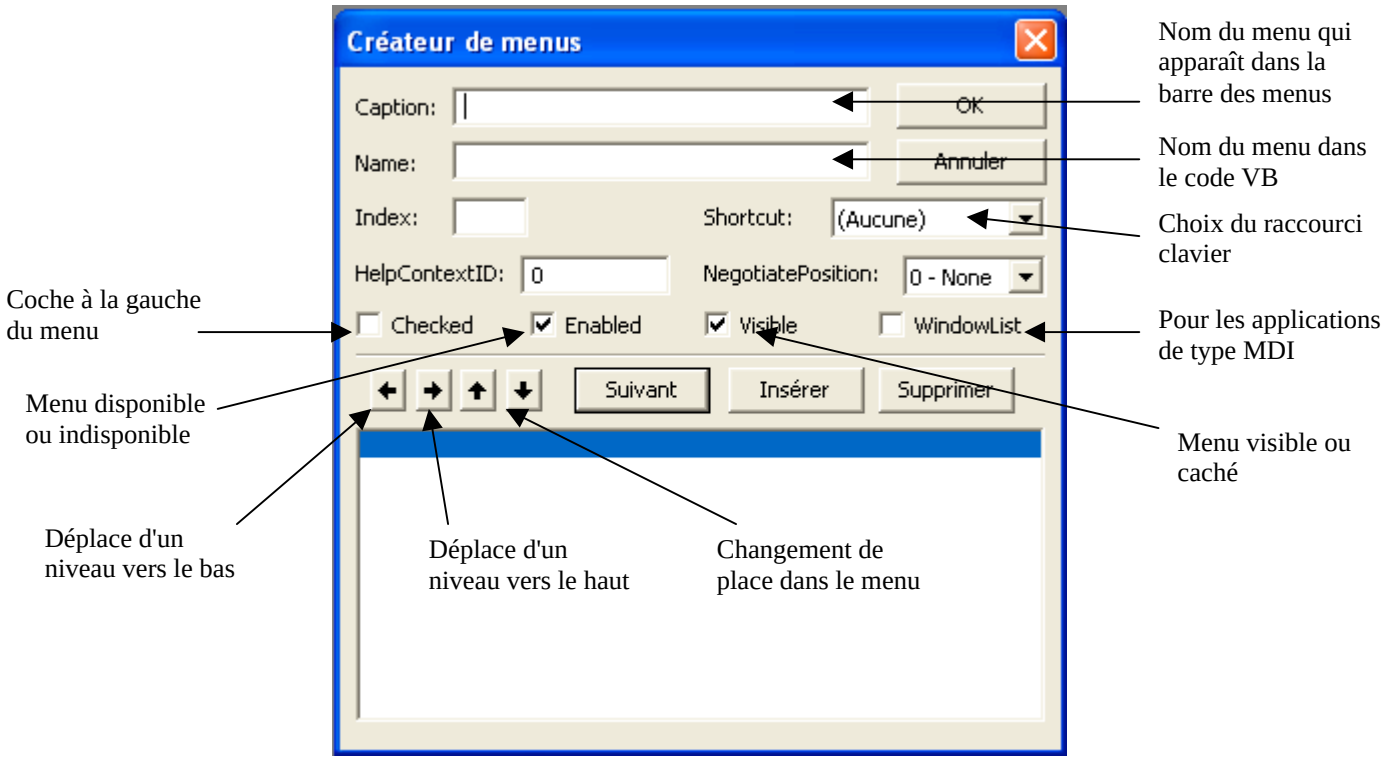
Par exemple Carré(7) retournera la valeur 49.

11 - CREATEUR DE MENUS

PRINCIPE

Le créateur de menus est un outil graphique gérant les menus de vos programmes. Pour afficher le créateur de menu, il faut être sur la fenêtre de création de la Form et utiliser :

Le bouton "**Créateur de menu**" dans la barre d'outils
ou
Le menu **Outils**
boîte de dialogue **Créateur de menus**



Remarques

- Si vous voulez créer une barre séparatrice dans votre menu, tapez un simple trait d'union (-) dans la zone Caption
- Pour ajouter une touche d'accès rapide à un menu, il suffit d'ajouter un & devant la lettre choisie dans le mot inscrit dans Caption (&Fichier, Fe&nêtre, …)

12 - TABLEAU A 2 DIMENSIONS DANS VB

A – OUTIL FLEXGRID

Pour afficher un tableau à deux dimensions dans VB, il faut ajouter un composant à la boite à outils :

menu **Projet**
boîte de dialogue **Composants**
choisir le contrôle **Microsoft FlexGrid Control**

Dessiner le tableau dans la form en utilisant l'outil FlexGrid

Utilisation des paramètres suivants :

- MSFlexGrid1.Rows : nombre total de lignes du tableau
- MSFlexGrid1.Cols : nombre total de colonnes du tableau
- MSFlexGrid1.Row : indice de ligne du tableau (commence à 0)
- MSFlexGrid1.Col : indice de colonne du tableau (commence à 0)
- MSFlexGrid1.Text : ajoute une valeur à la position MSFlexGrid1.Row, MSFlexGrid1.Lig

B – SAISIE ET AFFICHAGE D'UN TABLEAU

```
Const N As Integer = 3
Const M As Integer = 4
Dim TABLO(1 To N, 1 To M)

Private Sub Form_Load()
    'FixedRows = 0 : enlève le "noir" de la première ligne
    'FixedCols = 0 : enlève le "noir" de la première colonne
    MSFlexGrid1.Rows = N
    MSFlexGrid1.Cols = M
End Sub

Private Sub Command1_Click()
    For I = 1 To N
        MSFlexGrid1.Row = I - 1
        For J = 1 To M
            MSFlexGrid1.Col = J - 1
            TABLO(I, J) = InputBox("Entrer une valeur")
            MSFlexGrid1.Text = TABLO(I, J)
        Next
    Next
End Sub
```

13 - FICHIERS SEQUENTIELS

3 types de fichiers :

Séquentiel : fichier texte terminé par **CR LF** (Chr$(13) et Chr$(10));
Direct : fichier constitué d'enregistrements de longueur fixe;
Binaire : fichier constitué d'une suite d'octets sans lien logique

A – LES FICHIERS SEQUENTIELS

- **Ouverture**

  Syntaxe : **Open** "chemin et nom de fichier" **For** accès **As** Numéro Fichier

  Accès : **Input** → en lecture
  Output → en écriture (si le fichier existe, il sera écrasé)
  Append → en écriture à la fin du fichier déjà existant
  Numéro Fichier : permet de s'adresser au fichier concerné (valeur de 1 à 255)

  Si le fichier n'existe pas, il sera créé automatiquement en suivant le chemin indiqué (en mode *Output* ou *Append*).

  Exemples : *Open "C:\Istv\Iosifc\fichier.txt" For Input As 1*
  > Si le fichier n'existe pas, il sera créé automatiquement en suivant le chemin indiqué (en mode *Output* ou *Append*).

  Open "C:fichier.txt" For Input As 2
  > Si le fichier n'existe pas, il sera créé dans le dossier courant.

- **Lecture**

  Syntaxe : **Line Input #**Numéro Fichier, Enregistrement

  Enregistrement : variable recevant un enregistrement du fichier

  Si le fichier est vide, la lecture ne peut se faire et VB renvoie un message d'erreur (le programme se "plante")

  Exemple :
  ```
  Dim Enr as String
  Open "C:\Istv\Iosifc\fichier.txt" For Input As 1
  Line Input #1, Enr
  ```

- **Ecriture**

  Syntaxe : **Print #**Numéro Fichier, Enregistrement

  Enregistrement : variable recevant un enregistrement du fichier

  Exemple :
  ```
  Dim Enr as String
  Open "C:\Istv\Iosifc\fichier.txt" For Output As 1
  Enr = "Dernier enregistrement"
  Print #1, Enr
  ```

- **Fermeture**

  Syntaxe : **Close** Numéro Fichier

  Exemple :
  ```
  Close 1
  ```

- **Détection fin de fichier**

Syntaxe : **Eof** (Numéro Fichier)

Exemple :
```
Dim Enr as String
Open "C:\Istv\Iosifc\fichier.txt" For Input As 2
While Not Eof (2)
    Line Input #2, Enr
    Msgbox (Enr)
Wend
Close (2)
```

B – LES FICHIERS DIRECTS

- **Ouverture**

Open "chemin et nom de fichier" **For Random As** Numéro Fichier **Len** = lg

Lg = longueur d'enregistrement du fichier

Exemple
```
Open "C:\Mes Documents\Temp\fichier.txt" For Random As 2 Len = 10
```

- **Lecture**

Get #Numéro Fichier, Numéro Enregistrement, Enregistrement

Exemple
```
Dim Enr As String * 10
Open "C:\Mes Documents\Temp\fichier.txt" For Random As 2 Len = 10
Get #2, 2, Enr
Close (2)
```

- **Ecriture**

Put #Numéro Fichier, Numéro Enregistrement, Enregistrement

Exemple
```
Dim Enr As String * 10
Open "C:\Mes Documents\Temp\fichier.txt" For Random As 2 Len = 10
Enr = "dernier enregistrement"
Put #2, 1, Enr                    'Enr sera tronqué à 10 caractères
Close (2)
```

C – LES FICHIERS BINAIRES

- **Ouverture**

Open "chemin et nom de fichier" **For Binary As** Numéro Fichier

- **Lecture**

Get Numéro Fichier, Position octet départ, Enregistrement

Exemple
```
Dim Enr As String * 15
Open "C:\Mes Documents\Temp\fichier.txt" For Binary As 1
Get #1, 4, Enr              'Lecture de 15 octets à partir du 4ème
Close (1)
```

- **Ecriture**

Put Numéro Fichier, Numéro Enregistrement, Enregistrement

14 - BASES DE DONNEES

A - ACCES A UNE BASE EXTERNE

1. Généralités

Visual Basic permet d'accéder à différents formats de bases de données (Access, Paradox, dBASE, …). Il est possible également d'utiliser des bases de données en architecture client-serveur via des logiciels équipés de fonctionnalités ODBC (Open DataBase Connectivity).

Le **Contrôle Data** permet de considérer ces bases différentes suivant une même vue par l'intermédiaire d'un **objet Data Base**.

Le Contrôle Data fournit 3 objets qui facilitent la manipulation et l'utilisation d'une base de données (en particulier Access) :

- DBList ..... liste liée
- DBCombo..... liste combinée liée
- DBGrid ..... grille liée

Pour que ces composants soient disponibles dans la boite à outils, il faut les ajouter de la manière suivante:

Menu Projet
→ Composants…
cocher Microsoft Data Bound List Control 6.0
cocher Microsoft DataGrid Control 6.0

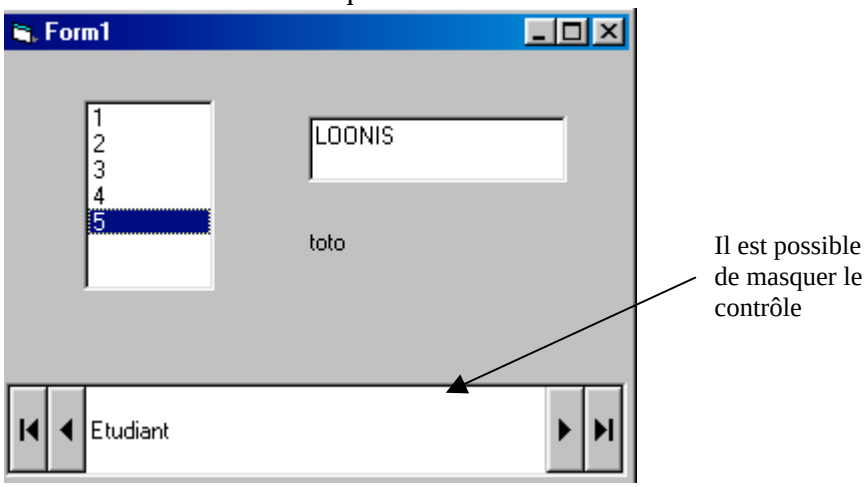
2. Utilisation avec une DBList

- Dans la feuille, insérer un objet **Contrôle Data** et initialiser les 3 propriétés suivante :
  - DatabaseName : chemin et nom de la base
  - Recordset Type : type d'utilisation de la base
    - 0 - table → utilisation en table et index
    - 1 - Dynaset → permet la Maj et le travail sur N tables (le + utilisé)
    - 2 - SnapShot → Photo instantané
  - RecordSource : nom de la table

- Insérer maintenant un objet **DBList** et initialiser les 3 propriétés suivantes :
  - DataSource : nom du contrôle Data de référence (ex. : Data1)
  - ListField : nom du champ qui sera listé dans la DBList (ex. : Num_Cli)
  - RowSource : nom du contrôle Data lié
    - → s'il n'y a qu'une seule table, idem DataSource
    - → si 2 tables reliées, il faut un 2ème contrôle Data

- Insérer autant d'objets **textes** (TextBox si on désire modifier la table, Label pour uniquement visualiser la table) que nécessaire dans la feuille et initialiser les propriétés suivantes :
  - DataSource : nom du contrôle Data (ex. : Data1)
  - DataField : nom du champ qui sera listé

- Au démarrage, la feuille va afficher dans le DBList tous les codes de la table et l'utilisateur va sélectionner un code en cliquant dessus.

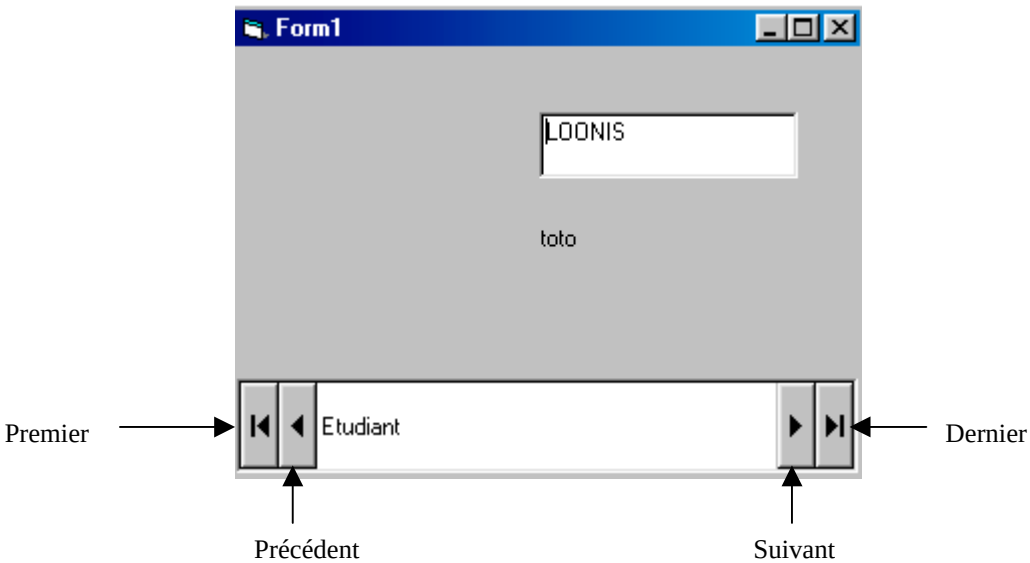


La propriété SelectedItem contient le signet (Bookmark) de l'élément concerné. En attribuant ce bookmark au bookmark du Recordset, on modifie l'enregistrement courant du RecordSet ainsi que le contenu des contrôles liés (en particulier les contrôles textes qui lui ont été attachés).
Les modifications introduites par l'utilisateur seront répercutées dans la base de données si l'affectation est indiquée dans le code VB :

```
Private Sub DBList1_Click()
Data1.Recordset.Bookmark = DBList1.SelectedItem
End Sub
```

3. Utilisation sans DBList

- Dans la feuille, insérer un objet **Contrôle Data** (cf. §2).
- Insérer autant d'objets **textes** que nécessaire (cf. §2).
- Au démarrage, la feuille va afficher dans les objets textes le premier enregistrement de la table choisie. Pour visualiser les autres enregistrements, il suffit d'utiliser les boutons du contrôle Data :



4. Création dans la table

```
Data1.Recordset.AddNew          ' Ajoute un enregistrement dans la table
    With Data1.Recordset        ' With permet de regrouper les Data1.Recordset
        ![champ1] = Text1
        ![champ2] = Text2
        etc.
        ![champN] = TextN
        .Update                 ' Maj dans la table Access
    End With
```

5. Modification dans la table

```
Data1.Recordset.Edit            ' Modifie un enregistrement dans la table
    With Data1.Recordset
        ![champ1] = Text1
        ![champ2] = Text2
        etc.
        ![champN] = TextN
        .Update                 ' Maj dans la table Access
    End With
```

6. Suppression dans la table

```
Data1.Recordset.Delete          ' Supprime un enregistrement dans la table
```