



Visual Basic

CHAKRAOUI Mohamed

FPO Ouarzazate

Plan

- Chapitre 1 : Introduction générale
- Chapitre 2 : Premiers éléments (Variables, Opérateurs, tests, Boucles)
- Chapitre 3 : Premiers contrôles (Form, Bouton de commande, Etiquette, Zone de text,..)
- Chapitre 4 : Quelques fonctions
- Chapitre 5 : Contrôles et groupes
- Chapitre 6 : Éléments graphiques
- Chapitre 7 : Complément de code (Gestion des erreurs, fonctions personnalisées)
- Chapitre 8 : Gestion des fichiers
- Chapitre 9 : Les bases de données
- Chapitre 10 : Utilisation de modules externes

Chapitre 1 : Introduction Générale

Chapitre 1 : Introduction Générale

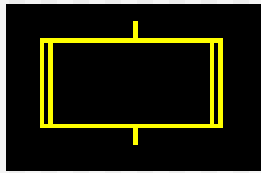
- Notion d'algorithme

Chapitre 1 : Introduction Générale

- **Un algorithme est une règle**, il s'exprime par une suite ordonnée de directives composée d'**actions** et de **décisions** qu'il faut exécuter en séquence suivant un enchaînement strict pour accomplir une tâche donnée, conforme à un cahier des charges.
- ***L'algorigramme*** reproduit dans un langage graphique normalisé tous les cheminements du raisonnement logique qui détermine la composition de l'algorithme.

Chapitre 1 : Introduction Générale

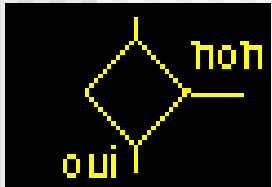
■ Structures algorithmiques :



Sous programme - Portion de programme considéré comme une simple opération.

Chapitre 1 : Introduction Générale

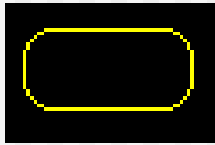
■ Structures algorithmiques :



Embranchement - Exploitation de conditions variables impliquant le choix d'une voie parmi plusieurs.

Chapitre 1 : Introduction Générale

■ Structures algorithmiques :



Début, fin ou interruption d'un organigramme, point de contrôle, etc...

Chapitre 1 : Introduction Générale

■ Sens conventionnel des liaisons

Le sens général des lignes de liaison doit être :

- de haut en bas
- de gauche à droite

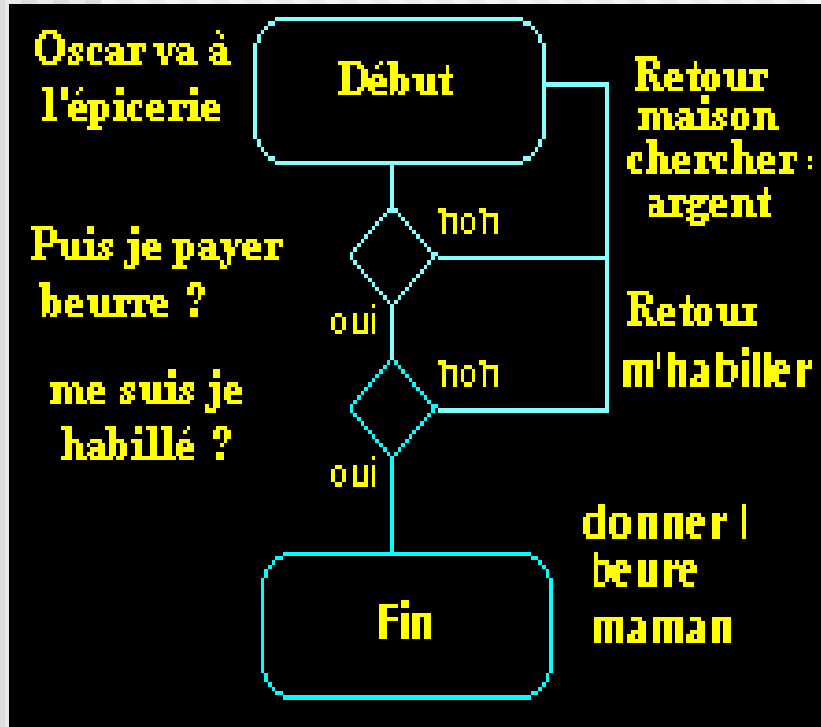
Chapitre 1 : Introduction Générale

■ Exemple

C'est la mère d'Oscar qui l'appelle et qui lui dit :
Oscar cours vite à l'épicerie, achète moi un pain de beurre,
prend de l'argent dans mon porte monnaie, habille toi bien
parce qu'il fait froid.

Si on crée une application en fonction du cahier des charges à
notre disposition, il va s'ensuivre ce qui suit :

Chapitre 1 : Introduction Générale



Oscar va à l'épicerie, demande du beurre,
traitement conditionnel1(Tc1) à t'il de l'argent ?
, non, pas d'argent,
alors il retourne chercher l'argent,
traitement conditionnel1(Tc1), à t'il de l'argent ?
oui, alors il prend le beurre,
(T2c2) est il habillé ?
, non, retour à la maison On recommence, s'habiller,
aller à l'épicerie, premier Tc,
réponse oui, deuxième Tc,
réponse oui, retour à la maison et donner beurre à maman. FIN

Chapitre 1 : Introduction Générale

- Un algorithme sert donc à faire d'un programme une suite logique d'événement qui s'enchaîne les uns aux autres à fin d'éviter de reprendre le même programme tout simplement parce que l'on avait oublié une étiquette, une feuille, ou une réponse à un message d'erreur.

Chapitre 1 : Introduction Générale

- Visual Basic, un langage événementiel (presque) objet

Chapitre 1 : Introduction Générale

■ Particularités d'un langage événementiel:

Un langage événementiel possède toutes les caractéristiques d'un langage traditionnel, avec deux grands aspects supplémentaires:

- 1- Les objets
- 2- Les procédures événementielles

Chapitre 1 : Introduction Générale

- Les objets

Chapitre 1 : Introduction Générale

Définition :

Un **objet** peut être considéré comme une structure supplémentaire d'information, une espèce de **super-variable**.

Dans une variable, on ne peut stocker qu'une information et une seule.

Un objet est un groupe de variables de différents types. Il rassemble ainsi couramment des dizaines d'informations très différentes les unes des autres au sein d'une même structure, rendant ainsi ces informations plus faciles à manier.

Chapitre 1 : Introduction Générale

Les Propriétés d'un objet :

Les différentes variables d'un même objet ne sont pas désignées par un indice, mais par un nom qui leur est propre. En l'occurrence, ces noms qui caractérisent les différentes variables au sein d'un objet s'appellent des **propriétés de l'objet**.

Conséquence : **toute propriété d'objet obéit strictement aux règles qui s'appliquent aux variables dans tout langage (type, taille, règles d'affectation...).**

Chapitre 1 : Introduction Générale

A titre d'exemple, prenons un objet d'usage courant : un ministre.

Les **propriétés** d'un ministre sont : sa taille, son poids, son âge, son portefeuille, le montant de son compte en Suisse, son nom, sa situation par rapport à la justice, etc.

On peut retrouver aisément le type de chacune de ces propriétés :

- le portefeuille, le nom, sont des propriétés de type caractère.
- la taille, le poids, l'âge, le compte en Suisse, sont des propriétés de type numérique.
- la situation judiciaire (mis en examen ou non) est une propriété booléenne.

Chapitre 1 : Introduction Générale

Ministre
le portefeuille (chaine de caractères) le nom (chaine de caractères) la taille(numérique) le poids(numérique) l'âge(numérique) le compte en Suisse(numérique) la situation judiciaire(booléen)

On peut en suite créer plusieurs entités du type "Ministre" comme Duchemol.

Duchemol possède des valeurs pour chacune des propriétés de l'objet "Ministre".

Chapitre 1 : Introduction Générale

La syntaxe qui permet de désigner une propriété d'un objet est : **objet.propriété**

Par exemple si le montant du compte en Suisse du ministre Duchemol s'élève à 100 000 euros et si la propriété désignant ce compte pour les objets de type ministre est la propriété CompteSuisse, on écrira donc l'instruction suivante :

Duchemol.CompteSuisse = 100 000

```
graph TD; A[Duchemol.CompteSuisse = 100 000] --> B[L'Objet]; A --> C[La Propriété]
```

L'Objet

La Propriété

Chapitre 1 : Introduction Générale

Pour affecter à la variable somme le montant actuel du compte en Suisse du ministre Duchemol, on écrira :

somme = Duchemol.CompteSuisse

Pour augmenter de 10 000 euros le montant du compte en Suisse de Duchemol, on écrira :

**Duchemol.CompteSuisse = Duchemol.CompteSuisse +
10 000**

Chapitre 1 : Introduction Générale

Les Méthodes d'un objet :

Les langages objet ont intégré une autre manière d'agir sur les objets : les **méthodes**.

Une méthode est une action sur l'une – ou plusieurs - des propriétés d'un objet.

Une méthode va supposer l'emploi d'un certain nombre d'**arguments**, tout comme une fonction. On trouvera donc des méthodes à un argument, des méthodes à deux arguments, et aussi des méthodes sans arguments.

Chapitre 1 : Introduction Générale

Par exemple, reprenons le cas notre ministre. Une méthode pourrait être **AugmenterPatrimoine**, qui supposerait un argument de type numérique.

Ministre
le portefeuille (chaine de caracteres) le nom (chaine de caracteres) la taille(numérique) le poids(numérique) l'âge(numérique) le compteSuis(numérique) la situation judiciaire(booléen)
AugmenterPatrimoine(numérique)

On pourrait ainsi écrire :

Duchemol.AugmenterPatrimoine(10 000)

Ce qui aurait exactement le même effet que de passer par la propriété correspondante :

Duchemol.CompteSuisse = Duchemol.CompteSuisse +
10 000

Chapitre 1 : Introduction Générale

- Procédures événementielles

Chapitre 1 : Introduction Générale

En PASCAL ou en C, par exemple, une application est constituée d'une procédure principale contenant la totalité du code (y compris par l'appel indirect à des sous-programmes). Les instructions qu'elle contient sont exécutées les unes après les autres, jusqu'à la fin.

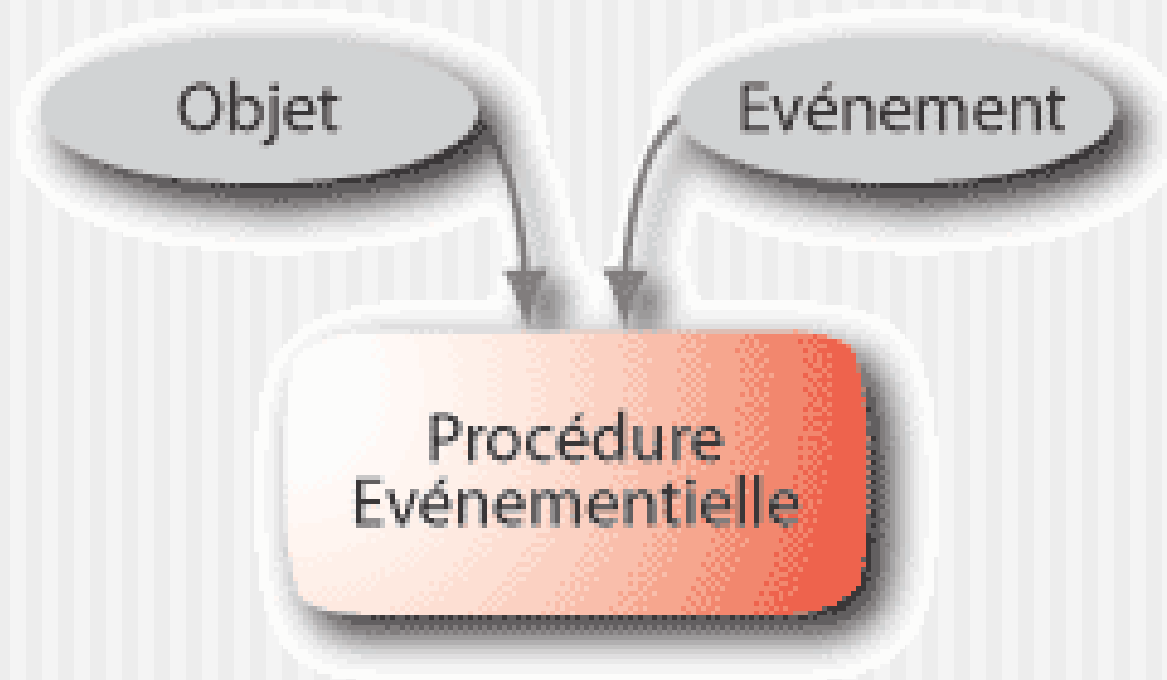
L'ordre d'exécution des procédures et des sous-procédures est entièrement fixé d'avance par le programmeur lui-même, par le biais des instructions d'appel des sous-procédures.

Chapitre 1 : Introduction Générale

- Dans un langage événementiel, il n'y a donc plus à proprement parler de procédure principale ; l'existence d'une procédure principale n'a rien d'obligatoire.
- Chaque procédure est liée à la survenue d'un **événement** sur un objet, et sera donc automatiquement exécutée lorsque cet événement se produit.
- **Le nom de la procédure est alors, de manière obligatoire, le nom de la combinaison objet-événement qui la déclenche.**

Chapitre 1 : Introduction Générale

Résumons par un petit dessin :



Chapitre 1 : Introduction Générale

Un objet est un contrôle.

Un contrôle est un des éléments de l'interface graphique de Windows, éléments que VB met à la disposition du programmeur pour qu'il constitue ses propres applications. Les contrôles les plus fréquents sont : la feuille, le bouton de commande, la liste, la case à cocher, le bouton radio, etc.

Un événement peut être déclenché :

- Par l'utilisateur : la frappe au clavier, le clic, le double-clic, le cliquer-glisser.
- Par déroulement du programme lui-même qui modifie, lors de l'exécution, une caractéristique de l'objet : le redimensionnement, le déplacement, etc.

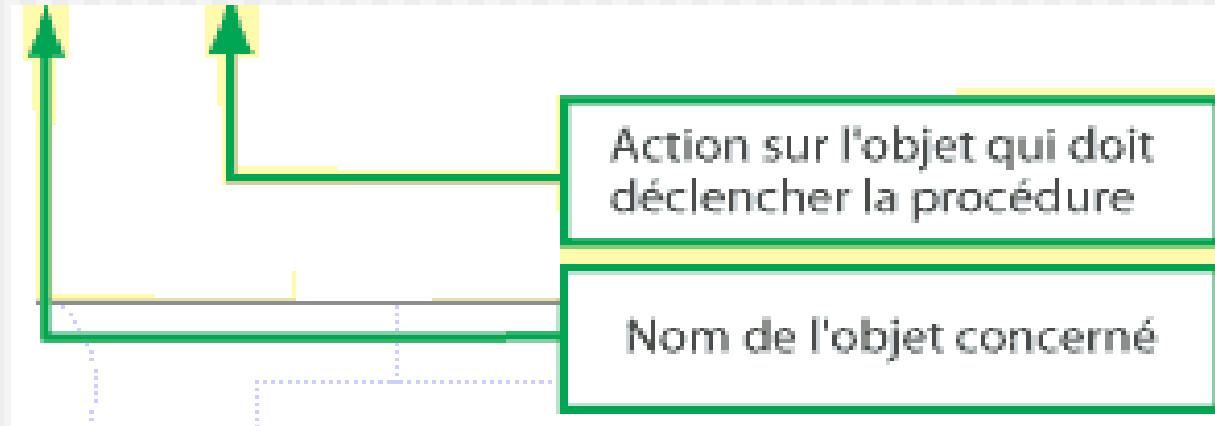
Chapitre 1 : Introduction Générale

Par exemple, considérons la procédure suivante :

```
Private Sub Machin_Click()
```

```
...
```

```
End Sub
```



Chapitre 1 : Introduction Générale

Cette procédure se déclenche si et seulement si l'utilisateur clique sur l'objet dont le nom est "Machin".

Si on prend le problème dans l'autre sens : si je suis programmeur, et que je veux qu'il se passe ceci et cela lorsque l'utilisateur clique sur l'objet appelé "Machin", je dois créer une procédure qui s'appellera obligatoirement :

Sub Machin_Click() et qui contiendra les instructions "ceci" et "cela".

Moralité : Écrire un programme qui exploite l'interface Windows, c'est avant tout commencer par définir :

- quels sont les objets qui figureront à l'écran
- ce qui doit se passer lorsque l'utilisateur agit sur ces objets via la souris ou le clavier.

Chapitre 1 : Introduction Générale

A la différence d'un programme traditionnel, les procédures liées aux différents événements possibles ne seront donc pas toujours exécutées dans le même ordre : tout dépend de ce que fera l'utilisateur.

En pratique, il est rare que l'on doive prévoir tous les événements. Si un événement se produit pour un objet (ex : l'utilisateur clique dessus) et qu'aucune procédure n'est attachée à cet événement, **il ne se passe rien !**

Chapitre 2 : Premiers éléments du code

Chapitre 2 : Premiers éléments du code

- Les variables

Chapitre 2 : Premiers éléments du code

En ce qui concerne les **noms** de variables :

- Les noms de variables n'ont pas de longueur maximale
- Ils doivent commencer par un caractère
- Ils ne doivent pas comporter d'espace
- Ils ne sont pas sensibles à la casse (prénom et PRénom sont la même variable)

Chapitre 2 : Premiers éléments du code

Les types de variables :

- **Boolean** : True – False
- **Byte** : de 0 à 255
- **Integer** : de -32 768 à 32 767
- **Long** : de -2 à +2 milliards environ
- **Single** : virgule flottante simple précision
- **Double** : virgule flottante double précision
- **String** : jusqu'à 65 000 caractères

Chapitre 2 : Premiers éléments du code

Déclaration de variables :

- La **déclaration** des variables est optionnelle.
- Dans le cas de non déclaration d'une variable, cette dernière est automatiquement créée dans un type spécial : le type **Variant**. Ce type possède comme particularité d'être "souple", à savoir de pouvoir s'ajuster à n'importe quel contenu (caractère, booléen, numérique..).

Mais, le type variant a comme inconvénient d'être très gourmand en mémoire vive occupée.

Chapitre 2 : Premiers éléments du code

La Portée des variables :

L'existence d'une variable peut se dérouler sur deux niveaux :

- **Niveau Procédure** : cela veut dire que la variable est **locale**. Dès que l'on quitte la procédure en question, la variable disparaît, et son contenu avec elle. Pour déclarer une variable au niveau procédure, on tape au sein de la procédure considérée :
Dim NomVariable **as** Type
- **Niveau Form** : la variable est disponible pour toutes les procédures de la Form , mais pas pour les procédures se situant sur une autre Form. Pour déclarer une variable au niveau Form, on tape **tout en haut de la Form, à l'extérieur des procédures** :
Dim NomVariable **as** Type

Chapitre 2 : Premiers éléments du code

Variables indicées

- On les appelle aussi des **tableaux** ! Ce peut être des tableaux de nombres, de chaînes, de booléens, bref, de tout ce qu'on veut.
- Tout tableau doit **obligatoirement** être déclaré.
- Quand on crée un tableau, soit on sait d'avance combien d'éléments il va contenir, soit on veut qu'il soit dynamique.
- Pour créer un tableau de 12 entiers, on écrira :
`Dim MonTableau(11) As Integer`
- Pour créer un tableau dynamique, on écrira :
`Dim MonTableau() As Integer`
- Ensuite, dès qu'on veut en fixer la taille, on écrit dans le code :
`Redim MonTableau(11)`
- Pour manipuler les éléments du tableau, il suffit d'écrire :
`MonTableau(1) = 9`

Chapitre 2 : Premiers éléments du code

- Les opérateurs

Chapitre 2 : Premiers éléments du code

Les opérateurs sont absolument standard. On les rappelle rapidement :

- Opérateurs numériques : **+** **-** ***** **/**
- Opérateurs booléens : **And** **Or** **Xor** **Not**
- Opérateur caractères : **&** (concaténation)

Chapitre 2 : Premiers éléments du code

- Les tests

Chapitre 2 : Premiers éléments du code

Les Structures de tests possibles sont :

- If ... Then
 ...
 EndIf
- If ... Then
 ...
 Else
 ...
 EndIf
- If ... Then
 ...
 Elseif ... Then
 ...
 Elseif ... Then
 ...
 Else
 ...
 EndIf

Chapitre 2 : Premiers éléments du code

- Les Boucles

Chapitre 2 : Premiers éléments du code

- While ...
...
Wend
- Do While ...
...
loop
- For x = a to b
...
Next x

Chapitre 3 : Premiers contrôles

Chapitre 3 : Premiers contrôles

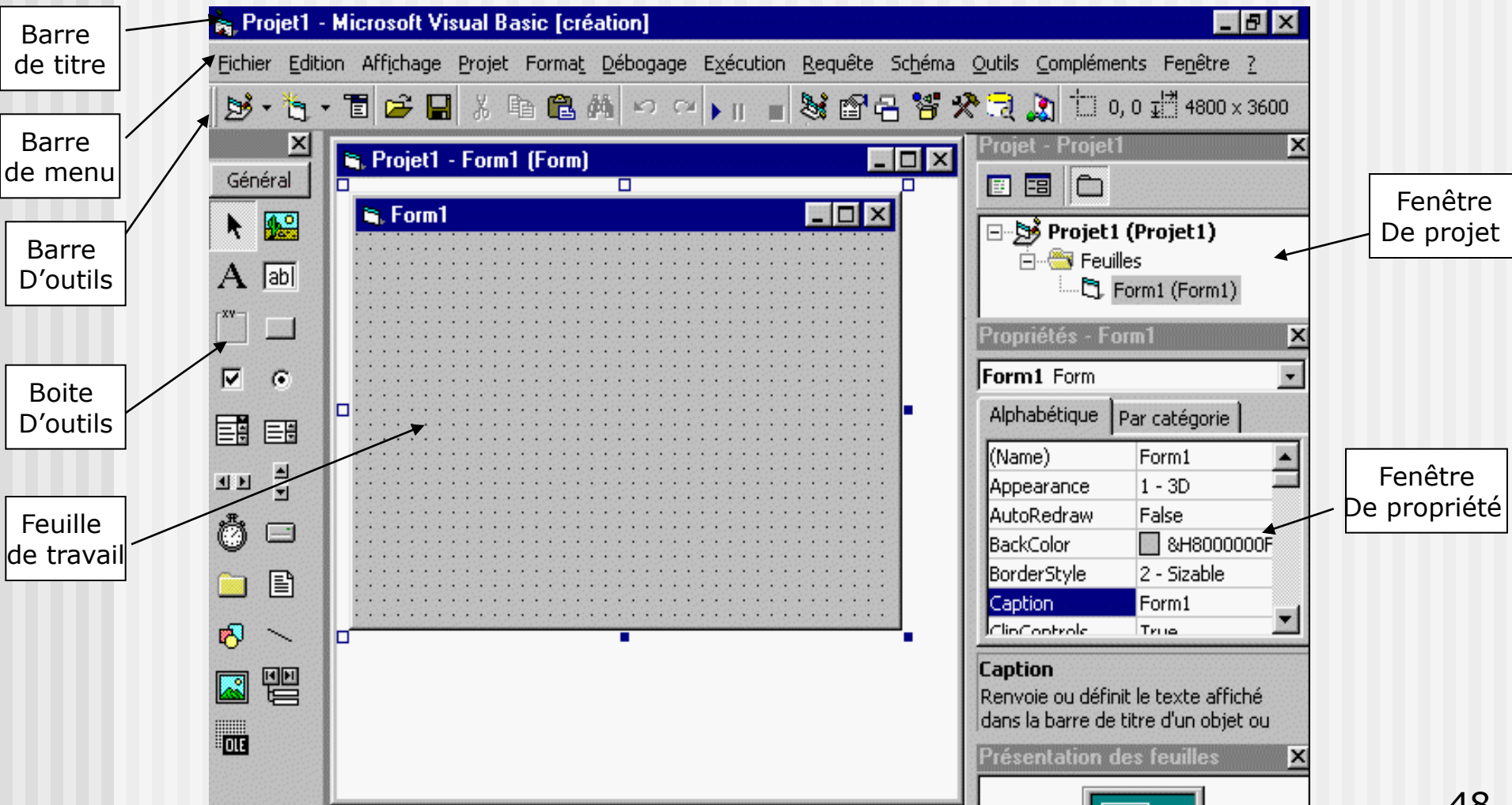
- Démarrer avec l'environnement Visual Basic :



“Exe.standard” représente le modèle le plus courant pour réaliser la plupart des applications sous Visual basic

Chapitre 3 : Premiers contrôles

■ L'interface de travail Visual Basic :



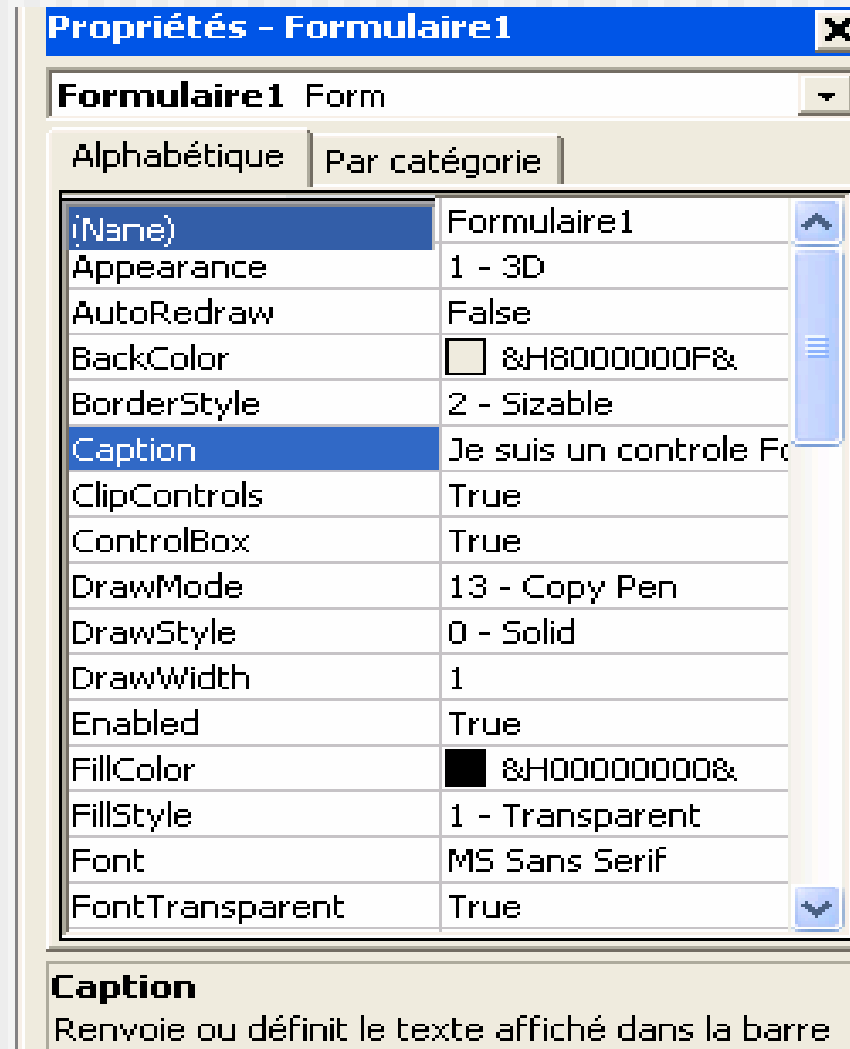
Chapitre 3 : Premiers contrôles

■ Le contrôle Form

C'est le contrôle de base. En Visual Basic c'est la feuille, ou formulaire, en anglais, **Form**. Cet objet est incontournable ; **on ne peut créer et utiliser d'autres objets que si ceux-ci font partie d'une Form.**

Chapitre 3 : Premiers contrôles

■ Fenêtre de Propriétés du contrôle Form

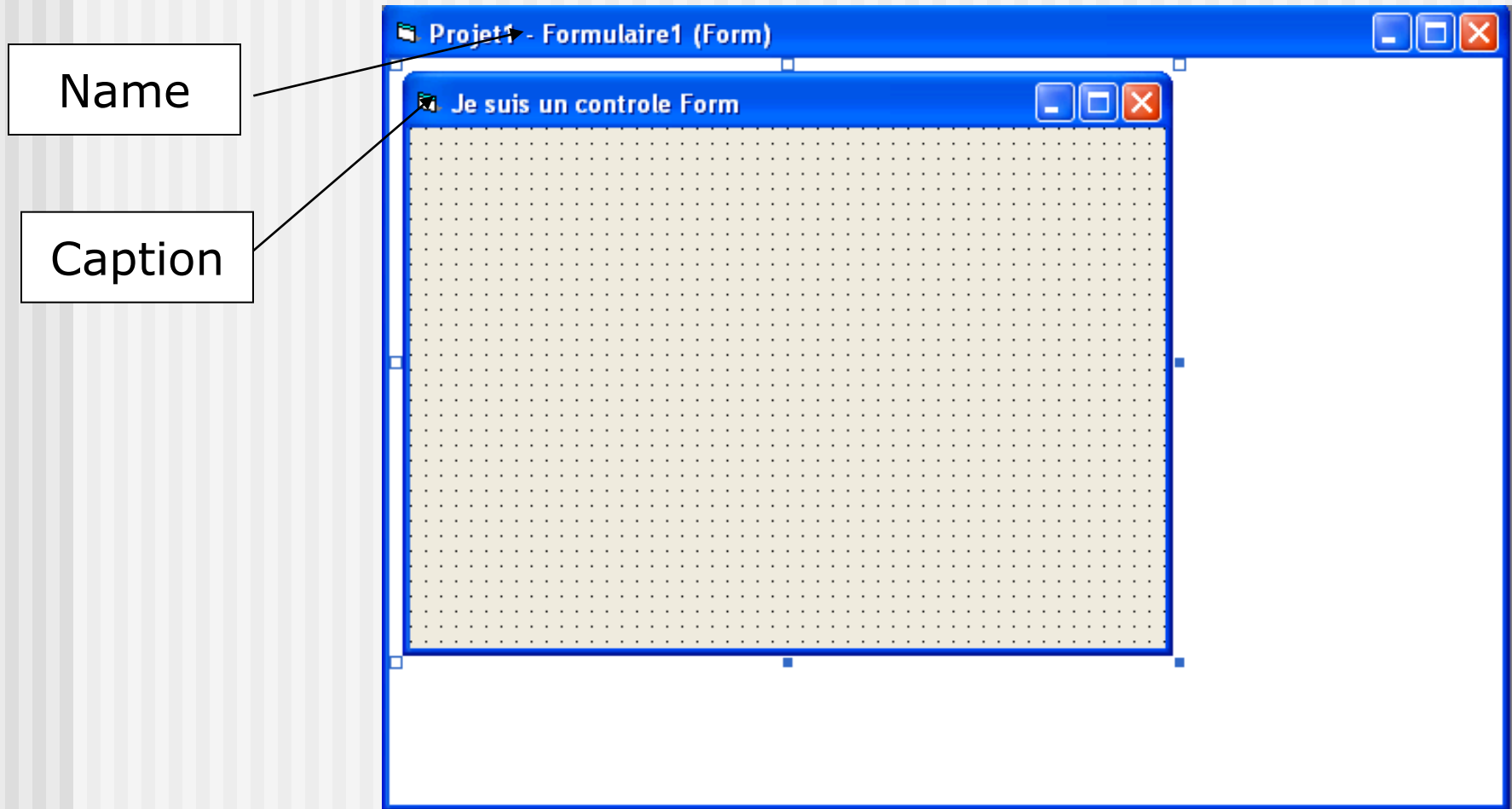


The screenshot shows the 'Propriétés - Formulaire1' window. It has a title bar with a close button. Below the title bar is a dropdown menu showing 'Formulaire1 Form'. There are two tabs: 'Alphabétique' (selected) and 'Par catégorie'. The main area is a list of properties and their values. The 'Caption' property is selected and highlighted in blue. Below the list, there is a section for the selected property, 'Caption', with a description: 'Renvoie ou définit le texte affiché dans la barre'.

Propriété	Valeur
(Name)	Formulaire1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H80000000F&
BorderStyle	2 - Sizable
Caption	Je suis un controle Fo
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	&H00000000&
FillStyle	1 - Transparent
Font	MS Sans Serif
FontTransparent	True

Caption
Renvoie ou définit le texte affiché dans la barre

Chapitre 3 : Premiers contrôles



Chapitre 3 : Premiers contrôles

- **Name** : il s'agit du **nom de l'objet tel qu'il est géré par l'application**. Cela correspond en quelque sorte à un nom de variable (sauf que ce n'est pas une variable, c'est un objet !).
- Par défaut, VB nomme tous les objets que vous créez des noms génériques, comme Form1, Form2, Form3, Text1, Text2, Text3, etc.

Chapitre 3 : Premiers contrôles

- **Caption** : il s'agit du **texte associé à l'objet sur l'écran**. Cette **Caption** est donc très utile pour professionnaliser une application, lui donner un look fini, mais ne joue aucun rôle dans la désignation de l'objet par l'application. C'est un rôle purement décoratif !
- Dans le code, on ne désigne donc jamais un objet par sa **Caption**, mais par son **Name**. Dans le cas de la Form, par exemple, la propriété **Caption** désigne le texte qui vient s'écrire dans la barre de titre.
- Chaque propriété joue un rôle différent.

Chapitre 3 : Premiers contrôles

Les procédures événementielles associées à l'objet Form sont nombreuses, la plus célèbre est la suivante :

Private Sub Form_Load()

End Sub

Qui se déclenche à l'ouverture du Form.

Chapitre 3 : Premiers contrôles

Le contrôle **CommandButton** (Bouton de Commande)

Quelques propriétés intéressantes de l'objet **CommandButton** :

- **Name** : Le nom du bouton
- **Caption** : Le texte qui s'affiche sur le bouton
- **Visible** : Propriété booléenne qui détermine la visibilité du bouton
- **Enabled** : Propriété booléenne permet (valeur **True**) à un contrôle d'être actif, c'est-à-dire de pouvoir recevoir des événements, et donc de déclencher des procédures. Inversement, elle interdit (valeur **False**) à un contrôle de recevoir quelque événement que ce soit de la part de l'utilisateur. Dans ce cas, le contrôle apparaît grisé à l'écran.

Chapitre 3 : Premiers contrôles

Les procédures événementielles associées à l'objet Bouton sont nombreuses, la plus célèbre est la suivante :

Private Sub Command1_Click()

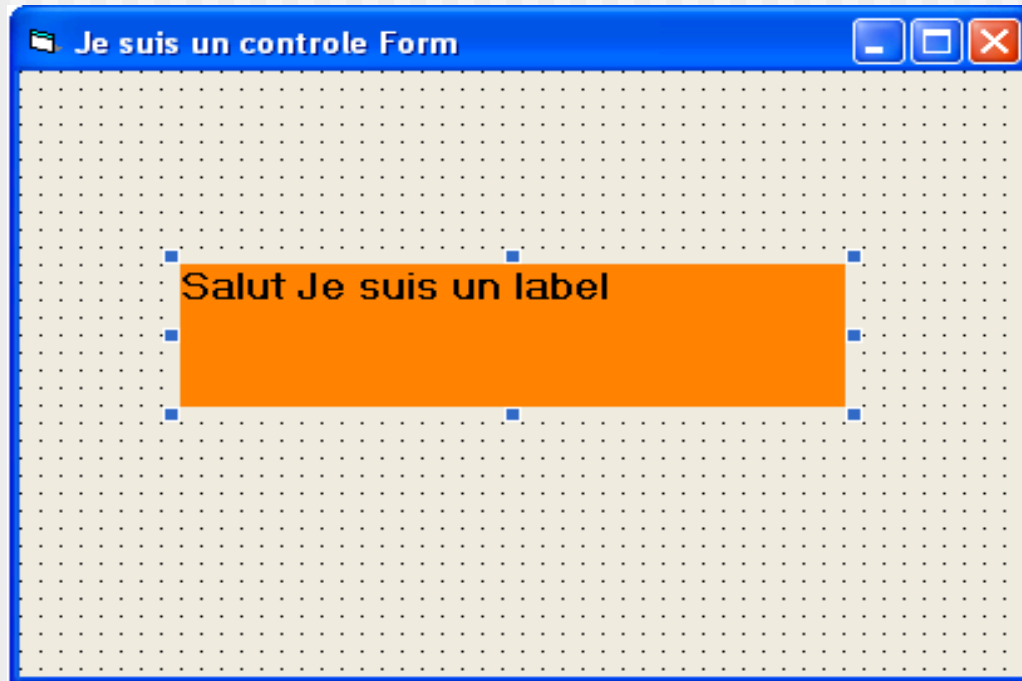
End Sub

Qui se déclenche quand l'utilisateur clique sur le bouton.

Chapitre 3 : Premiers contrôles

Le contrôle **Label** (**Etiquette**)

Un **Label** est un contrôle "inerte", qui sert à afficher un texte sur une Form. Son aspect peut varier quelque peu selon les styles adoptés :



Chapitre 3 : Premiers contrôles

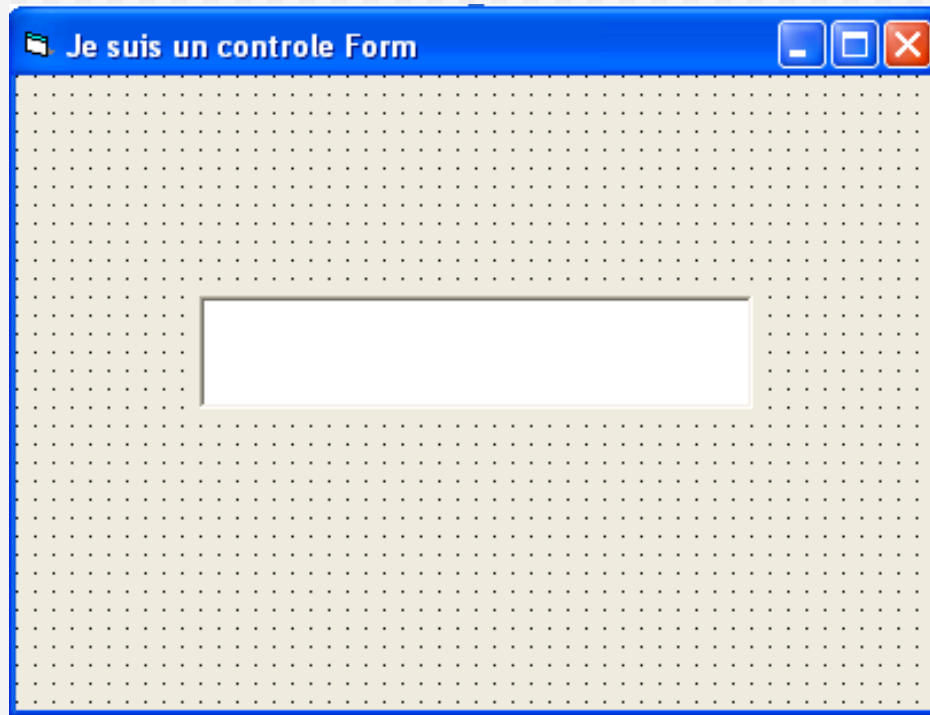
Quelques propriétés :

- **Name**
- **Caption** : Détermine le texte s'affichant sur le label
- **Font** : Détermine la police du label
- **BackColor** : Détermine le fond de couleur du label

Chapitre 3 : Premiers contrôles

Le contrôle Zone de Texte (TextBox)

Il peut servir à **saisir une information**. Il s'agit du seul contrôle permettant une saisie au clavier par l'utilisateur.



Chapitre 3 : Premiers contrôles

Quelques propriétés :

- La propriété essentielle d'une Zone de Texte est **Text**. C'est la propriété qui désigne son contenu.
- Si nous voulons mettre une zone de texte appelé "NomDeFamille" à blanc à l'affichage de la feuille, on écrira tout simplement :

Nomdefamille.Text = ""

- Si l'on veut récupérer le texte saisi par l'utilisateur dans la variable texte1, on passera l'instruction suivante :

texte1 = **Nomdefamille.Text**

Chapitre 3 : Premiers contrôles

■ Compilation et interprétation

Lorsqu'on écrit une application Visual Basic, on crée donc un ensemble d'objets et on définit les procédures qui se rapportent à ces objets.

Lorsqu'on sauvegarde cette application, Visual Basic va créer un certain nombre de fichiers. Pour l'essentiel :

Chapitre 3 : Premiers contrôles

- Un fichier dit **Projet** comportant l'extension ***.vbproj** (qui reprend l'acronyme de *Visual Basic Project*). Ce fichier rassemble les informations générales de votre application (en gros, la structure des différents objets Form, qui sont le squelette de toute application)
- Un fichier par objet **Form** créé, fichier portant l'extension ***.frm**. Donc si votre application comporte six Form, vous aurez en plus du fichier "projet", six fichiers "Form" à sauvegarder. Chacun de ces fichiers comporte les objets contenus par la "Form", ainsi que tout le code des procédures liées à ces objets.
- Éventuellement, d'autres fichiers correspondant à d'autres éléments de l'application.

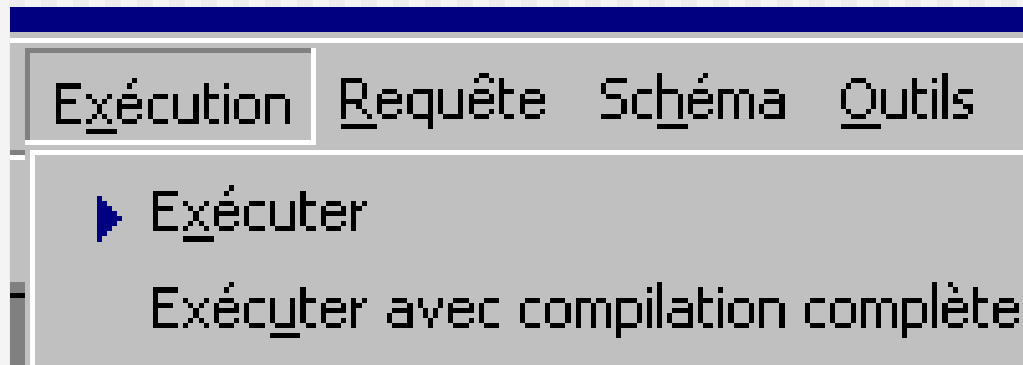
Chapitre 3 : Premiers contrôles

- Tant que le projet est ouvert sous la forme d'une collection de fichiers vbp et frm, vous pouvez l'exécuter afin de le tester.
- Lors de l'exécution, le langage est alors ce qu'on appelle « compilé à la volée ». C'est-à-dire que VB traduit vos lignes de code au fur et à mesure en langage machine, puis les exécute.

Chapitre 3 : Premiers contrôles

■ Exécution du projet :

Dans la barre d'outils, cliquez sur le bouton exécuter formé d'un petit triangle.



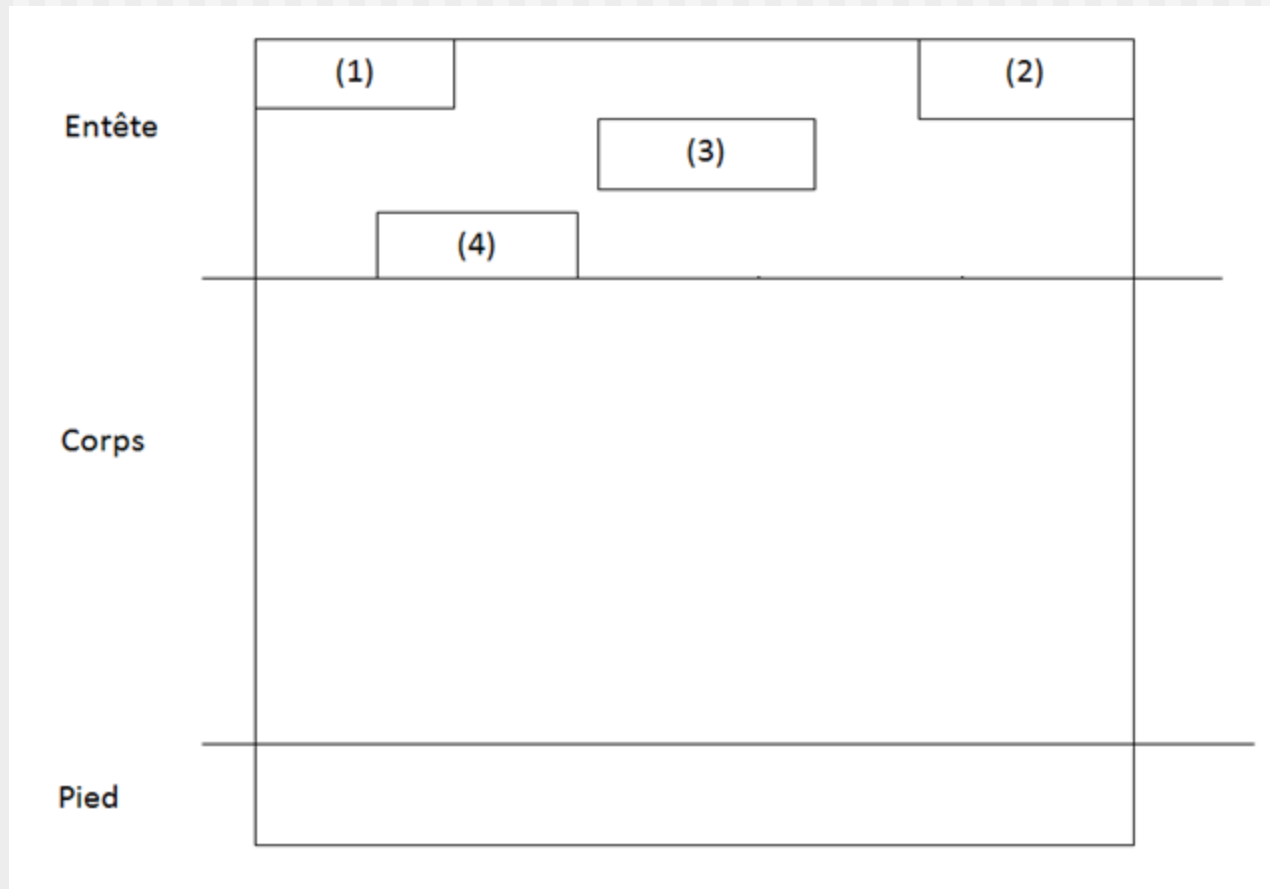
La feuille doit s'afficher, et lorsque vous cliquez sur le bouton "Quitter", l'application doit se fermer.

Chapitre 3 : Premiers contrôles

- Une fois l'application (le "projet") mis au point définitivement, vous devez la **compiler** une bonne fois pour toutes, créant ainsi un unique fichier ***.exe**. Ce fichier contient cette fois à lui seul l'ensemble de votre projet, form, code. Et il peut être exécuté sans l'ouverture – donc la possession - préalable de Visual Basic.
- Cependant, ce fichier ne peut être exécuté que sur les systèmes Windows (non portabilité des applications VB)

Éléments de modélisation d'interfaces

- Une interface graphique (objet) est composée de 3 parties : entête, corps et pied.



Éléments de modélisation d'interfaces

- L'entête :
 - (1) : Raison sociale de l'expéditeur. Exemple : « FPO-Ouarzazate ».
 - (2) : Représente la date d'édition correspondant à la date courante du système → il faut récupérer la date système avec les mot clé : Date et Time, et en utilisant le contrôle « Timer ».
 - (3) : Contient le titre de ce qui est en cours. Exemple : « Gestion des examens ».
 - (4) : Raison sociale du destinataire, elle représente la personne physique ou morale pour qui l'application actuelle a été réalisée.

Éléments de modélisation d'interfaces

- Le corps :
 - Dans cette partie de l'objet, l'utilisateur sera amené à définir les différentes entités en entrée et en sortie nécessaires à la résolution du problème.
 - Le contenu du corps n'est jamais standard, il varie d'un exercice à l'autre et il est impossible de lui donner un format standard.

Éléments de modélisation d'interfaces

- Le pied :
 - C'est une partie réservée à la totalisation ou synthèse de cette interface graphique.

Les boîtes de dialogues

■ Inputbox :

Variable\$ = InputBox\$(Prompt\$, Titre\$, Defaut\$, X%, Y%)

Prompt\$: le texte saisi par l'utilisateur.

Titre\$: contient le titre de la boîte de dialogue.

Defaut\$: est proposé au moment de l'ouverture de la boîte de dialogue.

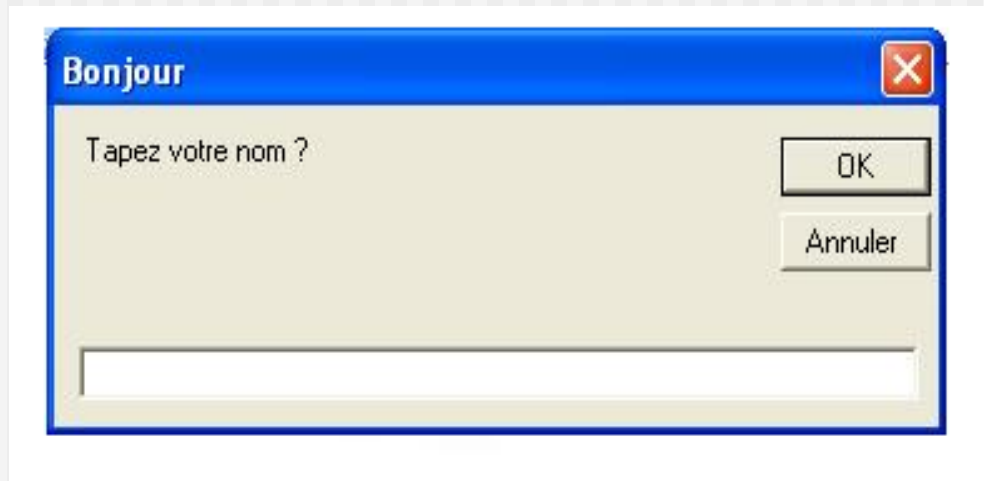
X% et Y% permettent de positionner la boîte, en l'absence de ces valeurs, la boîte de dialogue est centrée.

Les boites de dialogues

■ Exemple :

Nom = InputBox("Bonjour","Tapez votre nom ?")

Ca donne :



Les boites de dialogues

■ MsgBox :

Variable = MsgBox (texte1, integer, texte2, ...)

La fonction MsgBox comporte donc trois arguments essentiels

- le texte du message à envoyer
- le style des boutons et de l'icône éventuelle à faire figurer sur le message
- le texte de la barre de titre

Les boites de dialogues

■ Exemple :

```
Texte = "Enregistrer les modifications (...)"  
Titre = "Microsoft FrontPage"  
Toto = MsgBox ( Texte, 51, Titre )
```

Ou

```
Texte = "Enregistrer les modifications (...)"  
Titre = "Microsoft FrontPage"  
Toto = MsgBox (Texte, vbExclamation + vbYesNoCancel, Titre)
```

Les boites de dialogues

➔ On peut tester la valeur renvoyée par la fonction MsgBox.

■ Exemple :

```
Msg = "Alors, tu t'es décidé ? C'est oui ou c'est non ?"
```

```
Btn = vbYesNo + vbQuestion
```

```
Tit = "Alors ?..."
```

```
toto = MsgBox(Msg, Btn, Tit)
```

- ' Test de la valeur répondue, et branchement sur deux autres msgbox
- ' En cas de réponse OUI, Toto vaut la constante VbYes, mais aussi 6

```
If toto = vbYes Then
```

```
    tutu = MsgBox("C'est Oui !!!", vbExclamation, "Super")
```

```
Else
```

```
    tutu = MsgBox("Quel dommage...", vbCritical, "Aaaargh !")
```

```
End If
```

Exercices sur les boucles

- Demander la saisie d'une note tant qu'elle n'est pas comprise entre 5 et 20.
- Calculer et afficher la somme des entiers de 1 à x (x est un entier saisi par l'utilisateur).
- Calculer et afficher la factorielle de x (x est un entier saisi par l'utilisateur).
- Calculer et afficher la moyenne de 5 valeurs saisies par l'utilisateur.
- Calculer et afficher le max et le min de 5 valeurs saisies par l'utilisateur.
- Afficher tous les nombres inférieurs à x par ordre décroissant (x est un entier saisi par l'utilisateur).
- Afficher pour x donné, la somme des carrés des n premiers entiers jusqu'à x .

Quelques fonctions numériques

- `Int (nombre)` : renvoie la partie entière de ce nombre
- `Rnd ()` : renvoie un nombre pseudo-aléatoire compris entre 0 (inclus) et 1 (exclu).
- `Val (Chaîne)` : renvoie un nombre si Chaîne est composée de chiffres
- `Str (Nombre)` : renvoie Nombre sous forme de chiffres (c'est-à-dire de caractères)

Les chaînes de caractères

- Mid (Nomdechaîne, nombre1, nombre2) : renvoie une chaîne, extraite de Nomdechaîne, commençant au caractère numéro nombre1 et faisant nombre2 caractères de long
- Len (Nomdechaîne) : renvoie le nombre de caractères de Nomdechaîne.
- LTrim (Nomdechaîne) : renvoie la chaîne Nomdechaîne, débarrassée de tous les espaces se trouvant à gauche.
- Rtrim (Nomdechaîne) : renvoie la chaîne Nomdechaîne, débarrassée de tous les espaces se trouvant à droite.
- AllTrim (Nomdechaîne) : renvoie la chaîne Nomdechaîne, débarrassée de tous les espaces se trouvant à droite et à gauche.

Exercices sur les chaînes de caractères

- Ch1 et Ch2 sont deux chaînes de caractères. Vérifier que Ch1 et Ch2 comportent le même nombre de caractères, et qu'elles commencent par le même caractère et se terminent par le même caractère.

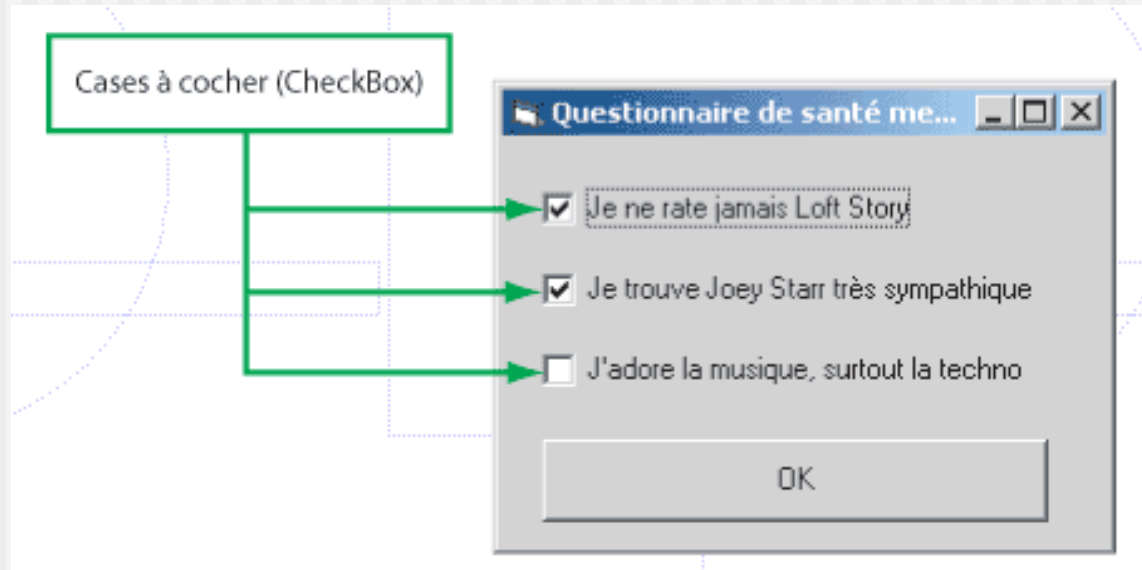
Utiliser Len et Mid

- Palindrome
- Transformer une chaîne de caractères en un tableau de caractères.

Autres contrôles : Les cases

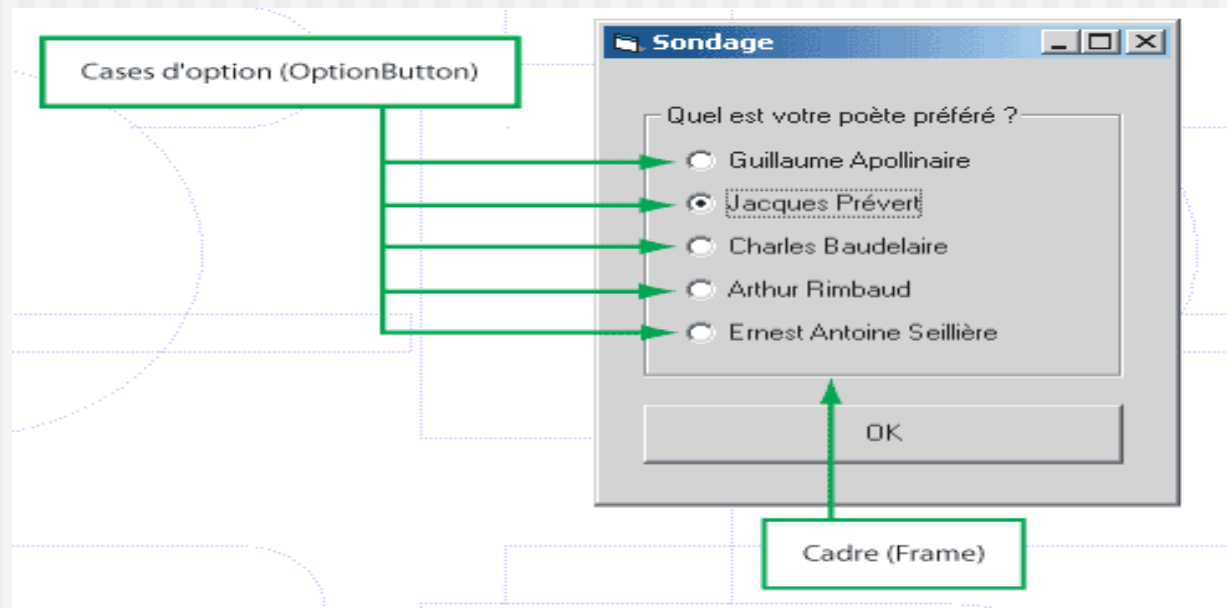
Il existe sous Windows deux sortes de cases :

- les cases dites "cases à cocher" (Checkbox): Elles sont carrées, et indépendantes les unes des autres, même si elles sont regroupées dans un cadre pour faire plus joli.



Autres contrôles : Les cases

- les cases dites "cases d'option", voire "boutons radio" (OptionButton). Elles sont rondes et font toujours partie d'un ensemble (dessiné par l'objet Frame). Au sein d'un ensemble de cases d'option, jamais plus d'une seule case ne peut être cochée à la fois.
- Il faut toujours créer le Frame avant de poser des cases à l'intérieur de ce Frame !! Dans le cas contraire, le langage ne reconnaîtra pas les cases comme faisant partie du même ensemble.



Autres contrôles : Les cases

■ Moralité !!

Avant de mettre des cases et d'écrire le code qui s'y rapporte, il faut bien se demander de quel type de cases on a besoin.

Autres contrôles : Les cases

La propriété la plus intéressante d'une case est celle qui nous permet de savoir si elle est cochée ou non. Cette propriété s'appelle **Value**.

- Value prend la valeur True ou False lorsqu'elle concerne les cases d'option.
- Value prend la valeur 1 ou 0 lorsqu'elle s'applique aux cases à cocher.

Value peut être utilisée aussi en écriture (pour initialiser telle ou telle case au moment où elle arrive à l'écran).

Les cases : Exercice 1

- Proposez une interface qui propose de choisir des options supplémentaires d'une voiture qui coûte au départ 150 000dhs. Les prix des options sont :
 - Option 1 : 10000dhs
 - Option 2 : 9000dhs
 - Option 3 : 12000dhs
 - Option 4 : 17000dhs
 - Option 5 : 20000dhs
- L'utilisateur peut choisir 0 ou plusieurs options.
- Calculez et affichez le prix total de la voiture :
 - Après validation des choix
 - Au fur et à mesure de la sélection des options

Les cases : Exercice 2

- Proposez une interface qui propose à l'utilisateur de choisir 3 livres à acheter (chacun avec son prix). Chaque livre est choisi parmi une liste de livres. De chaque liste, l'utilisateur ne peut choisir qu'un seul et unique livre.
- Affichez les livres que l'utilisateur a sélectionné, le prix de chaque livre et le prix total :
 - Après validation des choix
 - Au fur et à mesure de la sélection des livres

Les cases : Exercice 3

- Une calculette qui permet de faire le choix de l'opération (Addition, Soustraction, Produit, Quotient) en cochant une case.

Autres contrôles : Les listes

VB fournit deux contrôles de liste :

- une liste dite simple : contrôle ListBox
- une liste dite modifiable : contrôle ComboBox

Ces listes peuvent posséder ou non deux caractéristiques :

- Elles peuvent être modifiables : c'est-à-dire que l'utilisateur a la possibilité d'entrer un élément qui ne figure pas au départ dans la liste.
- Elles peuvent être déroulantes : c'est-à-dire qu'on ne voit qu'un seul élément de la liste à la fois, et qu'il faut cliquer sur la flèche du côté pour "déplier" la liste.

Autres contrôles : Les listes

■ Tableau récapitulatif :

LISTE	Non déroulante	Déroulante
Non modifiable	ListBox	ComboBox Style = 2
Modifiable	ComboBox Style = 1	ComboBox Style = 0 (défaut)

Autres contrôles : Les listes

Propriétés indispensables :

- **ListIndex** : renvoie ou définit l'indice de l'élément actuellement sélectionné.
VB gère les listes à la manière des tableaux. Il attribue à chaque élément d'une liste un indice, cet indice commençant toujours à zéro. Si aucun élément n'est sélectionné dans la liste, la propriété Listindex vaut -1.
- **List** : Cette propriété renvoie sous forme de texte un élément d'une liste en fonction de son indice. Il s'agit donc d'une propriété qui nécessite un argument (le numéro d'indice).

Exemple : Pour récupérer le libellé du produit sélectionné parmi la liste « Produit » :

```
NomProduit = Produits.List(Produits.ListIndex)
```


Autres contrôles : Les listes

- Toutefois, pour les comboBox on peut utiliser directement la propriété « Text » pour récupérer l'élément sélectionné.

Exemple :

```
NomProduit = Produits.Text
```

Autres contrôles : Les listes

Autres propriétés intéressantes :

- **ListCount**, qui renvoie le nombre d'éléments d'une liste (propriété numérique)
- **Multiselect**, qui permet la sélection multiple (propriété booléenne)
- **Sorted**, qui trie automatiquement les éléments d'une liste (propriété booléenne)

Autres contrôles : Les listes

Méthodes à connaître pour gérer les listes:

- **AddItem** Chaîne : ajoute l'élément Chaîne à une liste.
- **RemoveItem** (indice) : supprime de la liste l'élément possédant l'indice spécifié.
- **Clear** : efface tous les éléments d'une liste

Les listes : Exercices

- Une interface permet à son utilisateur de faire un seul choix parmi 2 auteurs.
- Au choix d'un auteur, sont affichés sur une liste ses œuvres littéraires pour que l'utilisateur sélectionne les livres qu'il veut acheter.
- Quand le choix de l'auteur change, la liste des livres est mise à jour également.

Les listes : Exercices

- Une interface propose ce qui suit :
- Une liste listant un ensemble de produits
- Une liste vierge qu'on peut alimenter à partir de la première liste avec les actions:
 - A droite
 - A gauche
 - Tout à droite
 - Tout à gauche
- Avant de lancer une de ces action, l'utilisateur doit sélectionner un produit de la première liste.

Les objets systèmes de fichiers

- La liste de lecteurs
- La liste de répertoires
- La liste de fichiers

Les groupes de contrôles

Les groupes de contrôles

Jusqu'à présent, nous avons toujours considéré que chaque contrôle était un objet complètement indépendant des autres contrôles. Cela se manifestait de deux manières :

- lors de la création des contrôles, nous cherchons toujours l'objet dans la boîte à outils, en évitant de faire des copier coller.
- par conséquent, chaque objet possède une propriété « Name » bien à lui, qui nous permet de le désigner sans ambiguïté en écrivant le code. Et donc, chaque objet possède ses propres procédures événementielles, distinctes de celles liées aux autres objets.

Les groupes de contrôles

- Cette stratégie peut provoquer dans certains programmes une lourdeur :
Lorsque plusieurs objets de même type remplissent des tâches semblables , on se retrouve avec une multitude de procédures événementielles (une par objet) qui se ressemblent beaucoup, voire qui sont carrément semblables.
- ➔ La création d'un groupe de contrôles a pour but d'alléger tout ceci.

Comment faire ?

Les groupes de contrôles

- 1- Lors de la création des contrôles (exemple : champs texte), on procède par copier-coller.
- 2- A la question "souhaitez-vous créer un groupe de contrôles?", on répond par l'affirmative.

Les quatre objets créés seront des membres d'un même groupe : ces quatre objets porteront le même nom (Name) mais auquel est ajouté un Index qui servira à les identifier individuellement.

➔ Donc, Si on a créé un groupe de 4 contrôles de TextBox le nom par défaut de ce groupe sera Text1. On aura alors Text1(0), car l'Index commence à zéro , Text1(1), Text1(2) et Text1(3).

Les groupes de contrôles

- Si on a un traitement à effectuer sur les quatre champs texte, on pourra écrire une boucle, par exemple pour effacer tout le texte dans les champs :

```
For i = 0 to 3
```

```
Text1(i).text=« »
```

```
Next i
```

- Toutes les structures de boucles déjà vu peuvent être utilisées : for next, while ou do while

Les groupes de contrôles

- Quand on crée la procédure associée à ce groupe on obtient la fonction suivante :

```
Private Sub Text1_Change(Index As Integer)  
  
End Sub
```

- La variable Index, que Visual Basic crée automatiquement en tant que paramètre en entrée de cette procédure, contient à chaque exécution de la procédure, l'indice de l'élément du groupe qui vient de déclencher l'événement : si au sein de cette procédure, on a besoin de savoir quel champs dont la valeur a été changée, il suffit de tester la variable Index, qui peut valoir de zéro à trois.

Application des groupes de contrôles : La gestion dynamique des contrôles

- La gestion classique des contrôles consiste à créer au départ tous les contrôles nécessaires, quitte à en cacher provisoirement quelques uns hors de la vue de l'utilisateur.
Ensuite on permet à l'utilisateur d'effectuer certaines actions sur ces contrôles qui permettront de les rendre visible.

Exemple : Le jeu du démineur : l'utilisateur a le droit de choisir entre plusieurs tailles de damier.

Une manière classique de programmer cela, serait de créer au départ le plus grand damier autorisé pour le jeu ; puis ensuite, à la lumière du choix effectué par le joueur, de masquer les cases qui n'ont pas lieu d'être.

Inconvénients : Cette solution est :

- Fastidieuse
- Très lourde en termes de ressources mémoire, puisqu'on va mobiliser de la place pour gérer des dizaines de contrôles.

La gestion dynamique des contrôles

- Autre solution : Utilisation des groupes de contrôles

Procédure à suivre :

- On crée à la main l'élément numéro zéro d'un groupe de contrôles (quitte à le masquer provisoirement), et on définit ses propriétés.
- On écrit les procédures événementielles liées à ce groupe
- On engendre de manière dynamique les autres éléments du groupe, dans la quantité souhaitée.
- Pour créer un élément supplémentaire d'un groupe de contrôles, on emploiera le code suivant :

Load NomduGroupe(i)

"i" étant l'index de l'élément qui sera créé.

La gestion dynamique des contrôles

- Remarque importante :

Tout nouvel élément, créé par **Load**, d'un groupe, est situé par défaut à l'emplacement exact de l'original, et invisible. Il faudra donc le plus souvent modifier ses propriétés **Top** et **Left**, ainsi que sa propriété **Visible**.

- Pour supprimer un élément d'un groupe de contrôles :

Unload NomduGroupe(i)

Les groupes de contrôles : Exercices

Exercice 1 :

- Faîtes une interface qui réalise la table de multiplication

Les groupes de contrôles : Exercices

Exercice 2 :

- Proposez une interface qui présente trois listes de produits : Produits cosmétiques, produits alimentaires et vêtements.
- Une quatrième liste contiendra les produits choisis par le client
- Un label affichera le prix total des produits à acheter

Les groupes de contrôles : Exercices

Exercice 3 : (Gestion dynamique des contrôles)

- Proposez une interface où l'utilisateur va choisir le nombre de modules qu'il veut suivre.
- Proposez une deuxième interface qui va offrir à l'utilisateur des contrôles au nombre du chiffre saisi avant, et des contrôles pour choisir la période d'inscription au module.

Les groupes de contrôles : Exercices

Pour un hôtel, on dispose de N chambres. Chaque chambre a un prix pour passer une nuit. Pendant le mois de Juillet, chaque chambre a été réservée pendant un nombre de jours précis.

Faites une application qui permet de :

- Saisir le nombre N de chambres dans l'hôtel
- Saisir le prix pour chaque chambre
- Saisir le nombre de nuits réservées pendant Juillet pour chaque chambre
- Calculer le revenu de l'hôtel pendant Juillet (la somme des prix payés par les personnes ayant réservé les chambres)
- Avoir la chambre qui a rapporté le maximum de revenu en Juillet
- Avoir la chambre qui a rapporté le minimum de revenu en Juillet

Les tableaux

Les tableaux (variables indicées)

- Ce peut être des tableaux de nombres, de chaînes, de booléens, bref, de tout ce qu'on veut.
- Quand on crée un tableau, soit on sait d'avance combien d'éléments il va contenir, soit on veut qu'il soit dynamique.
- Tout tableau doit obligatoirement être déclaré.
- Pour créer un tableau de 12 entiers, on écrira :
`Dim MonTableau(11) As Integer`
- Pour créer un tableau dynamique, on écrira :
`Dim MonTableau() As Integer`
- Ensuite, dès qu'on veut en fixer la taille, on écrit dans le code :
`Redim MonTableau(11)`

Les tableaux : Représentation sur écran

Exemple :

- Un tableau de 5 éléments de type String qui va contenir les libellés des produits alimentaires saisis par l'utilisateur sera représenté par un groupe de 5 champs textes : Text1(0), Text1(1), Text1(2), Text1(3), Text1(4).

La procédure associée à ce groupe de 5 champs texte est :

1- En considérant l'événement: changer le texte :

```
Private Sub Text1_Change(Index As Integer)
```

```
End Sub
```

2- En considérant l'événement: cliquer sur le champs :

```
Private Sub Text1_Click(Index As Integer)
```

```
End Sub
```

Les matrices

- Les matrices sont des variables indicées de dimension 2. C'est un ensemble fini de même type où l'accès est direct et nécessitant 2 indices de parcours tous les deux de type entier, le premier est appelé indice ligne, le deuxième est appelé indice colonne.
- Pour déclarer une matrice de 6 lignes et 5 colonnes, soit 30 éléments de type entiers :
`Dim MaMatrice(6,5) as Integer`
- Pour créer une matrice dynamique, on écrira :
`Dim MonMatrice() As Integer`
- Ensuite, dès qu'on veut en fixer la taille, on écrit dans le code :
`Redim MonMatrice(6,5)`
- Un élément de la matrice est noté `MaMatrice(i,j)`.

i représentant l'indice ligne de l'élément, et j représente l'indice colonne.

Les matrices

- Le parcours de la matrice peut se faire de 2 manières différentes :
 - 1- On fixe la ligne et on fait varier la colonne (parcours horizontal). La boucle correspondante est :

```
for i=0 to 5
  for j=0 to 4
    .....
  next j
next i
```

- 2- On fixe la colonne et on fait varier la ligne (parcours vertical). La boucle correspondante est :

```
for j=0 to 4
  for i=0 to 5
    .....
  next i
next j
```


Les matrices

- Représentation sur écran :

La représentation des matrices sur écran est impossible.

Cependant nous allons simuler cette représentation à travers un tableau qui n'a besoin que d'un seul indice, et on établira la relation entre cet indice nommé k avec les indices i et j de la matrice.

■ Exemple : $M(2,3)$

1- Représentation horizontale sur écran :

T0	T1	T2
T3	T4	T5

i : l'indice ligne allant de 0 à 1

j : l'indice colonne allant de 0 à 2

k : l'indice du groupe de contrôle allant de 0 à 5

Alors : $k = (i * 3) + j$

■ De façon générale : $M(L1, L2)$ est la matrice.

Pour i et j correspond k tel que :

$$k = (i * L2) + j$$

2- Représentation verticale sur écran :

T0	T2	T4
T1	T3	T5

i : l'indice ligne allant de 0 à 1

j : l'indice colonne allant de 0 à 2

k : l'indice du groupe de contrôle allant de 0 à 5

Alors : $k = (j * 2) + i$

- De façon générale : $M(L1, L2)$ est la matrice.

Pour i et j correspond k tel que :

$$k = (j * L1) + i$$

Les tableaux : Exercice

Exercice :

- Considérez 5 produits. L'utilisateur doit saisir pour chaque produit le libellé, le prix unitaire et la quantité à travers des fenêtres de saisie.

Utilisez des inputbox personnalisés pour chaque produit.

- Après avoir stocké toutes ces informations en mémoire en choisissant les types de données adéquats, calculez le prix total hors taxe, et le prix total TTC.

Les tableaux : Exercice

Exercice :

- Considérez 7 étudiants : Nom, Prénom, classe, note de maths, note de physique, note d'informatique, note de français.
- L'utilisateur doit saisir toutes les données relatives aux 7 étudiants. Utilisez des inputbox personnalisés pour chaque étudiant.
- Après avoir stocké toutes ces informations en mémoire en choisissant les types de données adéquats :
 - Calculez la moyenne des notes pour chaque étudiant
 - Affichez le nom, prénom de l'étudiant qui a la meilleure note en maths, en physique, en informatique et en français.

Les matrices : Exercices

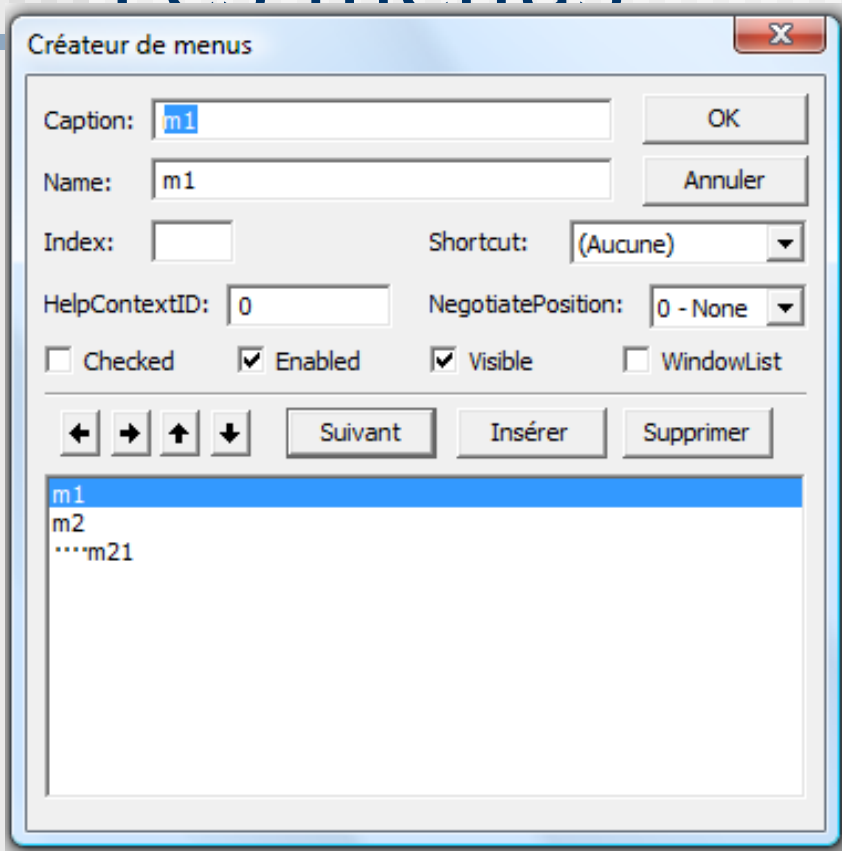
Exercice : Statistiques Marjane

- Considérez 5 produits. L'utilisateur doit saisir pour chaque produit le libellé, le prix unitaire et la quantité vendue pour chaque mois sur un semestre, à travers des fenêtres de saisie.
- Après avoir stocké toutes ces informations en mémoire en choisissant les types de données adéquats, calculez et affichez :
 - Le produit le mieux vendu sur chaque mois.
 - Le produit le mieux vendu sur tout le semestre.

Les menus

- Concevoir des menus pour une application VB se fait en utilisant un outil spécial : le créateur de menus.
- Cet outil fait apparaître un certain nombre de propriétés pour gérer les menus :
 - **Caption** : le texte du menu, tel qu'il apparaît à l'écran (ex : Fichier, Edition, Affichage, etc.)
 - **Name** : le nom de cet élément de menu pour l'application.
 - **Checked** : indique si l'élément de menu est coché par un petit "v" à sa gauche
 - **Enabled** : indique si l'élément de menu est actif ou inactif

Les menus



Ensuite, chaque élément du menu va pouvoir déclencher un traitement par la programmation de la procédure qui lui sera associée :

```
Private Sub NameElementMenu_Click()  
    ...  
End Sub
```

La gestion des erreurs

La gestion des erreurs

L'objet Err :

L'objet Err, 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.

Cet objet possède différentes propriétés que l'on pourra utiliser dans le code pour faire la gestion des erreurs :

- **Number** : désigne le code de cette erreur, permettant d'identifier son type
- **Description** : il s'agit d'une rapide description de la nature de l'erreur
- **Source** : qui indique l'objet à l'origine de l'erreur (le nom du projet)

La gestion des erreurs

- Toute erreur provoque un événement appelé **>Error**.
- Il faudra donc, au début d'une procédure susceptible d'engendrer des erreurs, taper la ligne suivante :
On Error GoTo Etiquette
- Une étiquette est une sorte de sous-titre à l'intérieur de la procédure, se terminant par le signe ":".
- L'idée est donc que si une erreur survient au cours de cette procédure, au lieu de se planter, l'application ira rechercher l'étiquette que vous avez désignée, et exécutera son code.
- **!!** Pour éviter que le déroulement normal de la procédure n'exécute lui aussi ce qui se trouve sous l'étiquette, il faudra forcer la sortie de la procédure avant cette étiquette par l'instruction **Exit Sub**.

La gestion des erreurs

■ Exemple :

```
Public Sub MaFonction()
```

```
On Error GoTo Etq1
```

```
Dim x, y as Integer
```

```
x = inputBox("Veuillez saisir un entier")
```

```
y = 12 / x
```

```
exit sub
```

```
Etq1:
```

```
Msgbox("Attention : Division par zéro !")
```

```
End Sub
```

Gestion de plusieurs erreurs

```
Public Sub MaFonction()  
    On Error GoTo Etq1  
    Dim x, y as Integer  
    x = inputBox("Veuillez saisir un entier")  
    y = 12 / x  
    exit sub
```

Etq1:

```
if(Err.Number=1235) then  
    MsgBox("Attention : Division par zéro !")  
elseif (Err.Number= 147) then  
    MsgBox(" .... ")  
else  
    MsgBox(Err.Description)  
endif  
End Sub
```

Gestion des erreurs : Exercices

Exercice 1:

- Demandez à l'utilisateur de saisir son nom, son prénom, son âge et sa date de naissance.
- Faites la gestion des erreurs relative à une mauvaise saisie de la date, à un dépassement de capacité et à une non compatibilité de types.

Les fonctions et sous-procédures

Les fonctions et sous-procédures

- Jusqu'ici, toutes les procédures que nous avons utilisées étaient des procédures événementielles :
contrôle + événement → procédure événementielle
- Mais VB permet comme tout langage de créer des procédures traditionnelles.
- Leur exécution sera alors déclenchée non pas par une action de l'utilisateur (événement), mais par une instruction de votre programme (appel à partir d'une procédure événementielle ou d'autre procédure).

- Exemple :

l'initialisation d'une Form (toutes les zones de textes doivent être vierges).

Si on a besoin de procéder à l'initialisation à plusieurs moments différents de notre application, au lieu de répéter à chaque fois les mêmes codes, on va créer une procédure Init() qui comportera les lignes de code nécessaires :

Les fonctions et sous-procédures

```
Private Sub Init()
```

```
Nom.Text = ""
```

```
Prénom.Text = ""
```

```
Etc.
```

```
End Sub
```

Pour appeler cette procédure, il suffira ensuite d'écrire l'instruction suivante :

```
Call Init
```

Les fonctions et sous-procédures

■ Portée des procédures :

Une procédure **Private** n'est disponible que si l'appel se situe dans la Form qui la contient.

Pour qu'une procédure puisse être appelée de n'importe quelle Form, il faut qu'elle porte comme nom **Public**. Dans ce cas, lors de l'appel, la procédure doit être désignée comme faisant partie d'une autre Form.

Exemple : Si l'on souhaite appeler dans Form1 une procédure appelée Proc est qui est déclarée dans Form2, il faudra écrire :

Call Form2.Proc

Les fonctions et sous-procédures

- Passage de paramètres à une procédure :

Si l'on veut passer des paramètres à une procédure, on aura recours à deux instructions : **ByVal** pour le passage par valeur(paramètre en entrée), et **ByRef** pour le passage par référence.

La déclaration d'une procédure Proc1 avec un paramètre numérique X en entrée et un paramètre Y en sortie donnera ainsi :

```
Private Sub Proc1(ByVal X as Integer, ByRef Y as Integer)
```

Les fonctions et sous-procédures

Les fonctions :

- Une fonction est une procédure qui retourne une valeur.
- Pour déclarer une Fonction, on utilisera les instructions Function et End Function.
- Pour retourner une valeur, vous devez écrire :
Nom de la fonction = valeur.
- Exemple :

```
Private Function Carre(X as Double)
Carre = X * X
End Function
```

Fonctions et sous-procédures : Exercices

Exercice 1 :

- Ecrire un programme PREMIERS affichant les nombres premiers inférieurs à 100. On utilisera une fonction PREMIER prenant un nombre entier et retournant true si le nombre est premier, false sinon.

Fonctions et sous-procédures : Exercices

Exercice 2 :

- Réalisez une fonction ou procédure de test de validité d'une valeur saisie.

Fonctions et sous-procédures : Exercices

Exercice 3 :

- Ecrire un programme permettant de calculer la somme :

i variant de a à b :

$(x \text{ puissance } i) / (\text{factorielle } i)$

Les fichiers

Les fichiers

- On s'intéresse aux fichiers texte.
- Ouvrir et fermer un fichier :

Pour ouvrir un fichier, vous devez :

- 1- connaître son nom (y compris le chemin d'accès)
- 2- dire ce que vous voulez en faire (lire, écrire, ou ajouter ?)
- 3- lui attribuer un numéro arbitraire, dit numéro de canal.

Open "C:\TextFile.txt" **For Input** As #1

Les fichiers

Remarques :

- Si vous utilisez plusieurs fichiers simultanément, il est impératif de leur attribuer chacun un numéro de canal différent
- **For Input** signifie qu'on ouvre le fichier en lecture. Pour l'ouvrir en écriture, on utilisera **For Output**, et pour l'ouvrir en ajout, **For Append**.
- A l'ouverture, il faut choisir une seule opération, on ne peut faire qu'une seule chose à la fois avec un fichier.
- L'instruction de fermeture d'un fichier est :
Close #i

où i est le numéro de canal du fichier à fermer.

Les fichiers

- Lire un fichier : il y'a plusieurs façons de lire un fichier

1- Procéder par **lecture séquentielle** (caractère par caractère):
NomDeVariable = Input (100, #i)

Recopie dans NomDeVariable les 100 caractères suivants du fichier numéro i, à partir de la dernière lecture.

Les fichiers

- Recopie de tout le fichier :

Nbr= LOF(#i)

NomDeVariable = Input (Nbr, #i)

Nbr récupère ici le nombre de caractères du fichier texte, via la fonction LOF. On recopie ensuite tout le fichier d'un seul coup d'un seul dans la variable NomDeVariable.

Les fichiers

- 2- Tirer avantage des fichiers texte qui sont organisés par lignes :

Line Input #i, NomDeVariable

chaque lecture avec Line Input copie la ligne suivante du fichier dans NomDeVariable.

Si vous avez construit votre fichier sous forme de champs de largeur fixe, vous pouvez récupérer, au sein de chaque ligne, les différents champs en découpant NomDeVariable en différents morceaux via la fonction Mid.

Les fichiers

- Exemple :

```
Dim T() as string
```

```
...
```

```
Open "C:\Monfichier.txt" For Input As #1
```

```
i = -1
```

```
While Not Eof(1)
```

```
    i = i + 1
```

```
    Redim T(i)
```

```
    Line Input #1, T(i)
```

```
Wend
```

On crée un tableau dynamique. On parcourt le fichier ligne après ligne tant qu'il en reste (boucle While ... Wend). A chaque ligne, on actualise l'indice i, on redimensionne le tableau, et on recopie la ligne dans la nouvelle case ainsi créée.

Les fichiers

- Ecrire dans un fichier : L'écriture se fait systématiquement ligne par ligne. L'instruction à utiliser est :

`Print #i, NomDeVariable`

- L'instruction écrit le contenu de la variable `NomDeVariable` dans le fichier de numéro `i`.

Pour être certain que les différentes informations sont rangées en respectant les champs de largeur fixe, le plus simple est de déclarer certaines variables `String` comme possédant toujours un nombre fixe de caractères.

Exemple :

```
Dim NomDeVariable as String*25
```


Les fichiers

Modification d'un fichier:

1- Ajouter des lignes à la fin d'un fichier :

```
Open FileName For Append As #10  
    Print #10, NewLine  
Close #10
```

Si le fichier n'existe pas, il est créé.

2- supprimer une ligne d'un fichier, connaissant le numéro de ligne :

- On considère que la première ligne est la ligne numéro 1.
- Le principe est de lire tout le fichier, de stocker les lignes du fichier dans un tableau. Ensuite, on ouvre le fichier en écriture et on écrit toutes les lignes sauf celle dont le numéro a été passé en paramètre

Le principe est le même pour les fonctions suivantes :

- supprimer une ligne d'un fichier, sur base de la présence d'une sous-chaîne"
- modifier une ligne d'un fichier, sur base de la présence d'une sous-chaîne"
- insérer une ligne à une position donnée dans un fichier

Les fichiers

- Exercice :

- 1- Créer deux fichiers structurés :

- Fichier 1 contient la liste de 5 modules :

- nom module de 20 caractères
- nombre de matières du module d'1 caractère

- Fichier 2 contient la liste de toutes les matières :

- Nom matière de 25 caractères
- Prof enseignant de 15 caractères

- 2- Créer une interface qui comporte deux listes et un bouton :

La première liste affiche les noms des matières

La deuxième liste affiche les nom des profs enseignants

Le bouton permet de passer au module suivant

Les bases de données

Les bases de données

- Visual Basic est un outil de développement qui offre plusieurs fonctionnalités, parmi lesquelles pouvoir se connecter à une base de données, et interagir avec elle (Ajout, suppression, modification, consultation, etc) .

Création de la base de données Access

- **Conseils :**

- Avant de créer votre base de données il faut bien réfléchir à sa structure ! Sinon, vous oublierez certainement certaines informations.
- Chaque table que vous créez doit contenir une clé !

- **NB :**

- A la fin de la création de la base de données, il faut la convertir au format Access 97, pour s'assurer que Visual Basic communique parfaitement avec la base de données.

Utilisation d'une base Access avec Visual Basic

1. Cliquez successivement sur projet, références, puis sélectionnez Microsoft DAO 2.5/3.21 compatibility library. Cliquez sur OK.
2. Déclarez une variable comme base de données, une variable comme « enregistreuse », et une variable de type String pour donner la requête à exécuter dans votre base de données.

Public db as Database

Public rs as recordset

Public sql as String

3. Indiquez le chemin de la base de données à laquelle le logiciel doit se connecter

```
Set db=OpenDatabase("C\....\votrebases.mdb")
```

NB: Pour être connecté dès l'ouverture du programme, mettez ce code dans la Form load

Ajout d'un enregistrement

1. Créer un nouvel enregistrement :

- ❑ `Sql="select * from NomDeVotreTable"`
on sélectionne tous les champs de la table
- ❑ `Set rs=db.OpenRecordset(sql,dbOpenDynaset)`
on sélectionne le mode écriture
- ❑ `rs.addnew`

Pour ajouter un enregistrement

2. Remplir les champs :

- ❑ `rs.fields("Nom")=txtNom.text`
- ❑ `rs.fields("Prenom")=txtPrenom.text`
- ❑ `rs.fields("Salaire")=txtSalaire.text`

3. Une fois les valeurs définies, on met à jour :

- ❑ `rs.update`
- ❑ `rs.close`

Consulter un enregistrement

- ❑ `Sql="select * from NomDeVotreTable where Nom='&txtNom.text & " " "`
 - ❑ `Set rs=db.OpenRecordset(sql,dbOpenSnapshot)`
 - Afficher les informations que vous voulez dans les textBox déjà créé :
 - ❑ `txtNom.text=rs.fields("Nom")`
 - ❑ `txtPrenom.text=rs.fields("Prenom")`
 - ❑ `txtSalaire.text=rs.fields("Salaire")`
- NB** : Il faut faire attention à sélectionner une valeur de champs où il n'y a pas de doublons.

Afficher un champs de tous les enregistrements dans une listbox

- ❑ `Sql="select * from NomDeVotreTable"`
- ❑ `Set rs=db.OpenRecordset(sql,dbOpenSnapshot)`
- ❑ `While not rs.EOF`
- ❑ `List1(0).addItem rs.Fields("Nom")`
- ❑ `rs.MoveNext`
- ❑ `Wend`

Modifier un enregistrement

- ❑ `Sql="select * from NomDeVotreTable where Nom='&txtNom.text & " ` "``
- ❑ `Set rs=db.OpenRecordset(sql,dbOpenDynaset)`
- ❑ `rs.edit`
- ❑ `rs.fields("Salaire")=txtModifsalaire`
- ❑ `rs.update`
- ❑ `rs.close`

Supprimer un enregistrement

- ❑ `Sql="delete from NomDeVotreTable where Nom=' '&txtNom.text & " ` "``
- ❑ `Set rs=db.OpenRecordset(sql,dbOpenDynaset)`
- ❑ `rs.execute`
- ❑ `rs.close`

- ❑ `chemin_base11 = "J:\test.mdb"`
- ❑ `Set base111 = Workspaces(0).OpenDatabase(chemin_base11)`

- ❑ `Set etd = base111.OpenRecordset("select * from Enfants WHERE Enfants.doti = '" & g & "' ")`
- ❑ `etd.MoveFirst`
- ❑ `Do While Not etd.eof`
- ❑ `etd.Delete`
- ❑ `etd.MoveNext`
- ❑ `Loo`

Ajouter une nouvelle table

- `Sql="CREATE TABLE NomDeVotreNouvelleTable (Nom Text (5), Ville Text (5)) "`
- `db.execute sql`

Supprimer une table existante

- `Sql="DROP TABLE NomDeLaTableASupprimer "`
- `db.execute sql`

Ajouter un champs à une table

- `Sql="ALTER TABLE NomDeVotreTable ADD COLUMNS Nom TEXT (10) "`
- `db.Execute sql`

Supprimer un champs à une table

- `Sql="ALTER TABLE NomDeVotreTable DROP COLUMNS Nom "`
- `db.Execute sql`

NB: A la fin, il faut toujours fermer la connection avec :
`db.close`

Projet à rendre le 13/01

- Créez une base de données pour la gestion d'une clinique multi-spécialités.
- Créez une application qui va communiquer avec la base créée et qui assurera les fonctionnalités suivantes (à étendre !!) :
- Pour les médecins :
 - Ajouter un nouveau médecin (nom, prénom, spécialité, date de début de fonction)
 - Supprimer un médecin
 - Modifier les informations sur un médecin après une recherche par clé
 - Faire une recherche du médecin par :
 - Nom
 - Spécialité
 - Date de début de fonction
- Pour les patients :
 - Ajouter un nouveau patient (nom, prénom, numéro de CIN, date d'entrée à la clinique, maladie, médecin traitant, date de sortie)
 - Supprimer un patient
 - Modifier les informations sur un patient après une recherche par clé
 - Faire une recherche du patient par :
 - Nom
 - CIN
 - Date d'entrée à la clinique
 - Maladie
 - Date de sortie
 - Chaque patient doit avoir une fiche où se trouve toutes les traitements, les interventions chirurgicales, les médicaments prescrits, ...

Projet (Suite)

- On doit pouvoir afficher la liste des patients qu'un médecin donné a traité durant une période donnée
- Créez un menu pour l'application
- Faites la gestion des erreurs