Initiation à Visual Basic

Cours

Public Concerné : Licence Sciences Cognitives

NOM DE L'AUTEUR : A. Belaïd **DATE : 2003**

UNIVERSITE NANCY 2
Département de Math-Info
54000 NANCY cedex
Abdel.Belaid@univ-nancy2.fr

1 Prise en main

1.1 Introduction

Pourquoi Visual Basic

- Il est l'un des langages de programmation les plus performants et les plus simples à utiliser ;
- Le langage Basic a été créé par John G. Kemeny et Thoams E. Kurtz en 1963 et devient vite un langage populaire ;
- Il fût adapté sur PC par Bill Gates, au milieu des années 70 ;
- Depuis, plusieurs versions améliorées pour PC ont vu le jour : Microsoft QuickBasis et MS-DOS Qbasic ;
- La simplicité du langage explique son choix pour le développement d'applications Windows telles que Microsoft Excel.

Notion de programme

- Un programme est un jeu d'instructions conduisant l'ordinateur à exécuter une tâche : gestion d'une paie, calcul scientifique de la solidité d'un pont, etc.
- Le programme est destiné à être exécuté sur Windows ;
- Pour écrire le programme, il faut :
  - Définir l'objectif souhaité ;
  - Concevoir l'écran - interface utilisateur - du programme ;
  - Développer le programme proprement dit avec VB.

Conception du programme

Elle passe par une planification qui consiste à définir la liste chronologique des différentes étapes : on parle d'**algorithme.**

Conception de l'interface

- Il est important ensuite de réfléchir à l'apparence qu'aura le programme ainsi qu'à la manière dont il traitera les informations ;
- L'ensemble des écrans et des images qui apparaissent dans un programme sont regroupés sous le terme **d'interface utilisateur ;**
- L'interface utilisateur comprend tous les menus, boîtes de dialogue, boutons, objets et images que les utilisateurs voient lorsqu'ils exécutent l'application ;
- L'interface utilisateur doit comprendre les éléments standards pour être compréhensible par tous.

Développement d'un programme sous VB

Il implique les trois étapes suivantes :

- Création de l'interface utilisateur à l'aide des contrôles VB
- Définition des caractéristiques ou propriétés des éléments qui composent l'interface
- Ecriture du code de programmation pour un ou plusieurs éléments de l'interface en fonction des besoins

Schéma de réalisation d'un programme

Les étapes de réalisation d'un programme sont décrites dans la figure suivante :

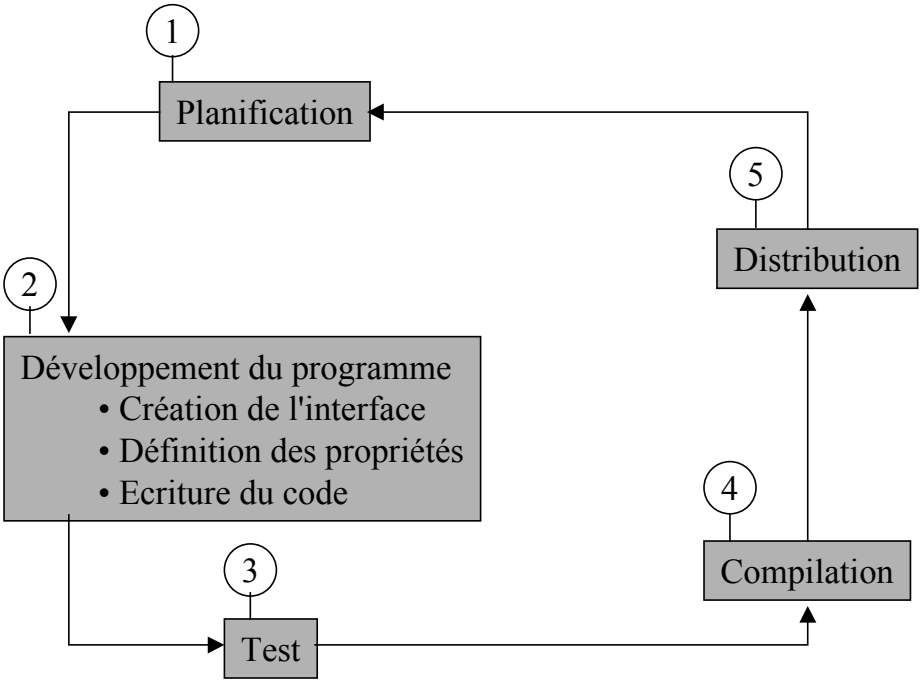


Figure 1: Schéma de réalisation d'un programme

1.2 Faire connaissance avec Visual Basic

Démarrage de VB

1. Cliquer sur « Démarrer », sur « Programmes », puis sur le dossier « Microsoft Basic 6.0 » ;
2. Cliquer sur l'icône du programme VB. La boîte de dialogue « Nouveau projet » s'affiche et un choix se prose pour un type de projet à créer ;
3. Cliquer sur « ouvrir » pour accepter la proposition par défaut. Un nouveau projet s'ouvre accompagné de fenêtres et d'outils**.** La taille et la forme exacte de ces fenêtres dépendent de la configuration du système. (voir Figure 2).

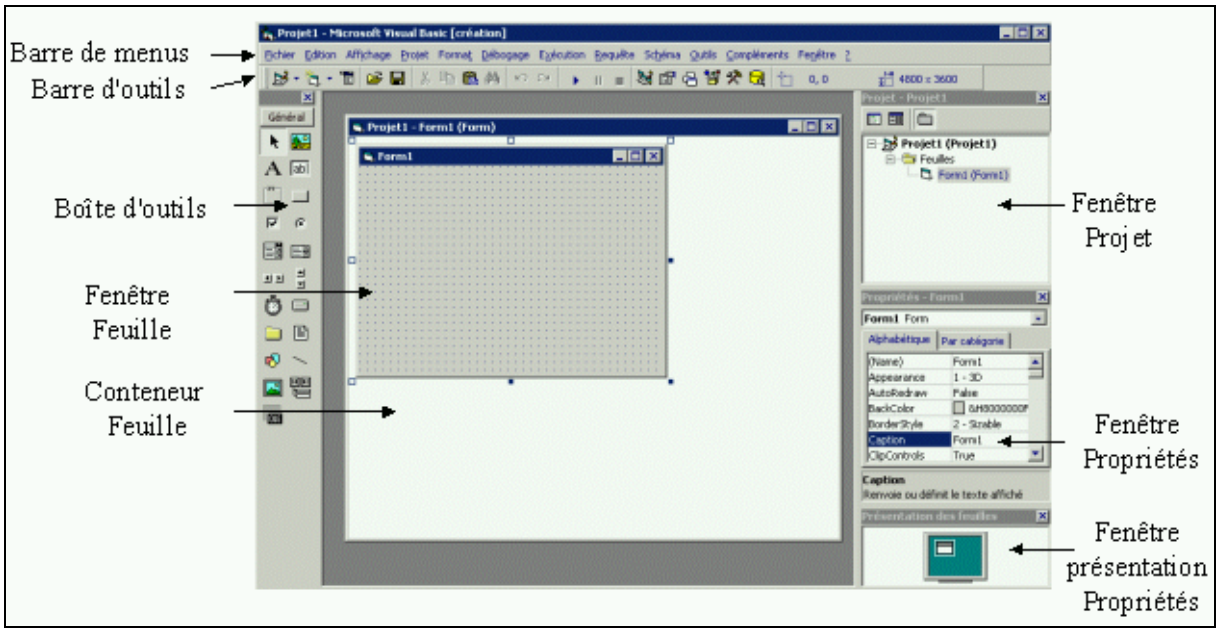


Figure 2: Eléments de la boîte de dialogue VB

Ouverture d'un projet existant

1. Dans Windows, cliquer sur Fichier, puis sur la commande « Ouvrir un projet »
2. Localiser le projet : c'est un fichier d'extension : **.vbp** (Visual Basic Project)

Éléments de l'interface VB

Feuille Interface Utilisateur

- C'est la feuille par défaut (Fenêtre feuille), appelée **Form1**, qui s'affiche au démarrage. Cette feuille présente une grille standard avec des points servant à aligner les éléments créés et composant l'interface Utilisateur ;
- On peut ajuster la taille de l'interface à l'aide de la souris ;
- On peut ajouter des feuilles supplémentaires à l'aide de la commande « Ajouter une feuille » du Menu « Projet ».

Boîte à outils

- Elle contient des **outils** et **contrôles** permettant d'ajouter des éléments à l'interface ;
- Chaque contrôle ajouté à l'interface devient un **objet** , ou élément programmable de l'interface ;
- A l'exécution du programme, ces objets agiront comme tous les objets standards d'une application Windows ;
- La boîte à outils contient également des contrôles que l'on peu exécuter en arrière plan dans un programme VB : leur effet est caché : par ex. manipulation interne de BD, contrôle du temps des programmes, etc.

Fenêtre Propriétés

- Elle répertorie les propriétés possibles des éléments de l'interface et offre la possibilité de les changer ;
- On peut affecter directement des propriétés aux objets sélectionnés ;
- Ces propriétés peuvent être ensuite changées par programme (en agissant sur le code) ;
- Si la fenêtre n'apparaît pas, cliquer sur le bouton correspondant de la barre d'outils.

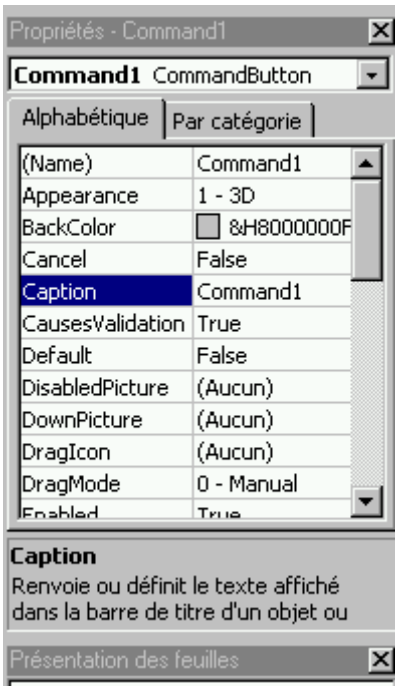


Figure 3: Fenêtre Propriétés

Fenêtre Projet

- Elle répertorie l'ensemble des fichiers utilisés dans la procédure de développement et les rend accessibles grâce à deux boutons spécifiques : « Code » et « Afficher l'objet » ;

- Le fichier projet qui gère la liste de tous les fichiers pris en charge par le projet de développement est appelé fichier de **Projet Visual Basic** (.vbp) ;
- Sous le nom du projet, la fenêtre affiche les composants sous la forme d'une arborescence.
- Cliquer sur le bouton « Explorateur de projet » pour l'afficher.

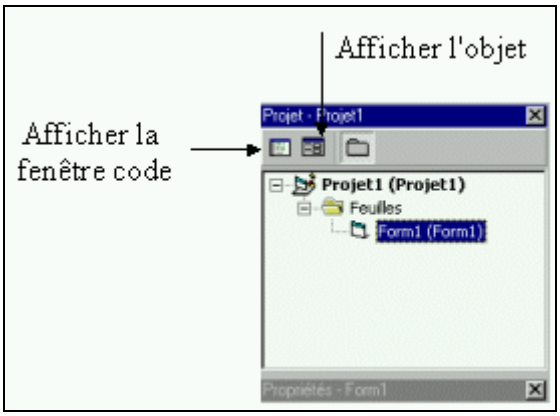


Figure 4: Fenêtre Projet

Enregistrer et quitter le programme

1. Enregistrer la feuille (en Cliquant sur Form1 dans la fenêtre Projet) sous : nom.frm
2. Enregistrer le projet (Barre Outils) en cliquant sur le bouton « Enregistrer le projet » sous le nom nom.vbp
3. Dans le menu Fichier, cliquer sur la commande « Quitter »

1.3 Écriture d'un programme VB

Exemple : Programme de jeu : "Lucky Seven"

Il s'agit de réaliser une interface d'un programme de jeu simulant une machine à sous. La Figure 5 montre le type d'interface à créer. Elle comporte deux boutons de commande, trois fenêtres d'affichages de chiffres, un graphique représentant une pile de pièces de monnaie, et le libellé Lucky Seven (nom du jeu). Le jeu consiste à cliquer sur la commande « Jouez », si un chiffre 7 apparaît dans une des trois fenêtres, alors le programme affiche l'image des piles de pièces, sinon cette image ne s'affiche pas. L'arrêt du jeu se fait en appuyant sur la commande « Arrêtez »

Figure 5 : Interface du programme Lucky Seven

Création de l'interface

1. Ouvrir un nouveau projet ;
2. Créer les deux boutons de commande : cliquer sur le bouton « commande » de la boîte à outils, rester appuyé dessus, le déplacer sur la feuille et le positionner à l'endroit voulu. On peut le re-dimensionner avec la souris (pointer sur le coin inférieur et tirer avec la souris) ;
3. Ajouter les étiquettes chiffres en utilisant le bouton « label » (même principe que pour Commande). Prévoir une zone plus importante pour le label 4 accueillant le texte Lucky Seven ;
4. Introduire une zone « Image » pour y insérer l'image des sous. La Figure 6 donne l'image de l'interface.

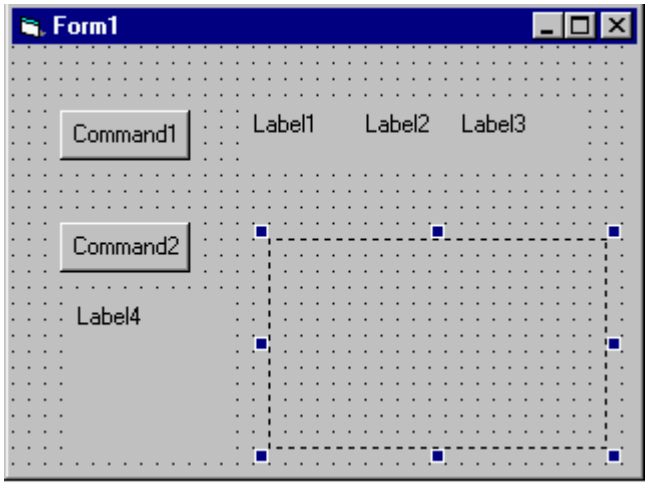


Figure 6: Préparation de l'interface de Lucky Seven

Définition des propriétés des boutons Commande

1. Cliquer sur le bouton Command1 ;
2. Double-cliquer sur la fenêtre propriétés ;
3. Double-cliquer sur Caption ;
4. Saisir « Jouer » ;
5. Faire la même chose avec Command2 en saisissant « Arrêter ». Pour retrouver les commandes, il suffit d'aller dans la zone de liste déroulante Objet située en haut de la fenêtre Propriétés.

Définition des propriétés des étiquettes de chiffres

1. Sélectionner les trois étiquettes de chiffres en cliquant d'abord sur la première puis sur les deux autres en maintenant le bouton MAJ appuyé. Un rectangle de sélection encadre chacune des étiquettes.
   - *Comme plusieurs objets ont été sélectionnés, seules les propriétés susceptibles d'être changées collectivement sont affichées dans la fenêtre Propriétés.*
2. Propriétés à définir :
   - Alignement : choisir 2-center ;
   - BorderStyle : choisir 1-Fixed dans le menu ;
   - Font : Times New Roman, style Gras, taille de caractère de 24 points ;
3. Supprimer les trois libellés afin que les cases soient vides au démarrage du programme :
   - Sélectionner individuellement chacune des trois étiquettes ;
   - Double-cliquer sur la propriété Caption et appuyer sur SUPPR. Le libellé de l'objet Label1 est supprimé. Répéter l'opération pour les deux autres.

Définition de l'étiquette descriptive

1. Cliquer sur l'objet étiquette ;
2. Changer la propriété Caption en Lucky Seven ;
3. Changer la fonte, la taille… comme précédemment ;
4. Changer la couleur en agissant sur ForeColor. Cliquer sur l'onglet "Palette" puis sur la case violet foncé. La couleur est traduite en Hexadécimal dans la fenêtre.

Définition des propriétés de la zone Image

Cette zone est sensée contenir le graphique des pièces. Ce graphique apparaît lorsque l'utilisateur remporte le jackpot (au moins une fois le chiffre 7).

1. Cliquer sur l'objet zone d'image ;
2. Mettre la propriété Stretch à True ;
3. Double cliquer sur la propriété Picture dans la fenêtre Propriétés . La boîte de dialogue "Charger une image" apparaît, puis aller chercher l'image dans la partition Microsoft sous Clipart. Le fichier s'appelle "Pieces.wmf". En l'ouvrant, le métafichier Windows est chargé dans la zone d'image de la feuille ;
4. Mettre la propriété Visible sur False de manière à masquer les pièces au démarrage du programme. (Vous le ferez apparaître ultérieurement dans le programme).

Ecriture du code du programme

Il s'agit du code chargé de calculer les chiffres aléatoires, de les afficher dans les cases correspondantes et de détecter un éventuel jackpot.

Comme le programme est géré par l'activation des boutons "Jouer" et "Arrêter", il suffit d'associer le code approprié à ces boutons.

La fenêtre *Code* est une fenêtre spéciale de l'environnement de programmation permettant d'entrer et d'éditer les instructions de programmation :

> ✈ ***La procédure écrite est une procédure événementielle. Elle s'exécutera quand on aura appuyé sur la commande correspondante.***

Utiliser la fenêtre Code

1. Doubler cliquer sur la fenêtre "Arrêter" sur la feuille. La fenêtre Code apparaît ;
2. Rentrer l'instruction : End ;
3. Doubler cliquer sur la fenêtre "Jouer" sur la feuille. La fenêtre Code apparaît ;
4. Rentrer le code suivant :

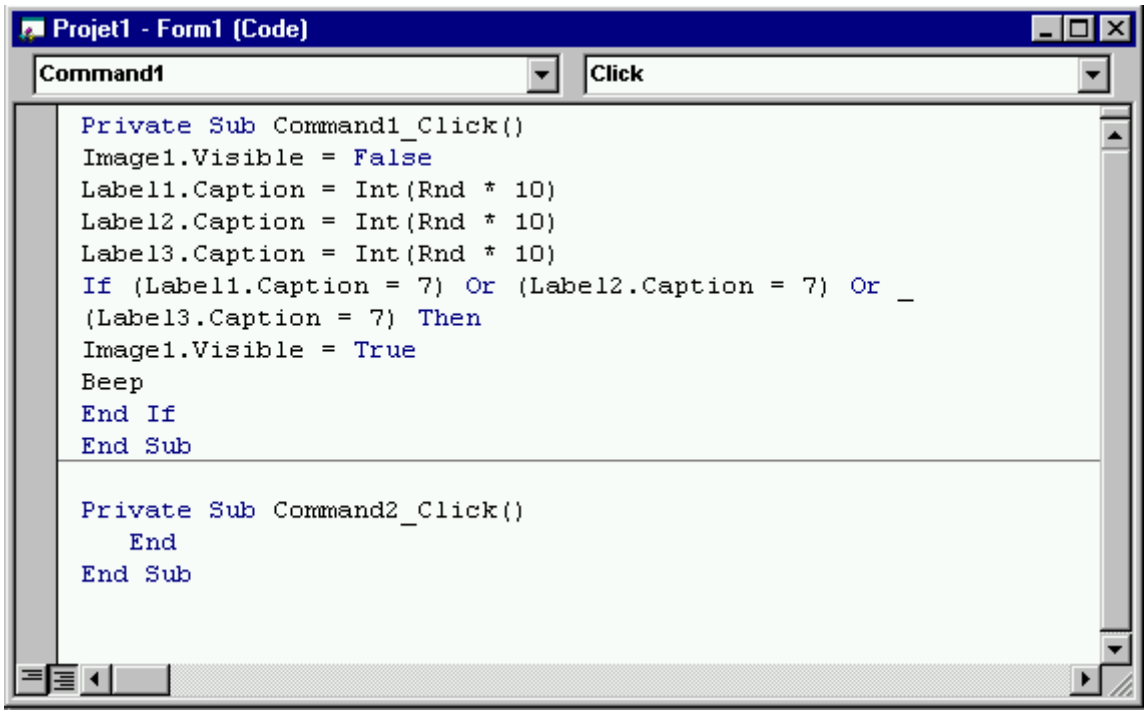


Figure 7: Fenêtre Code

Examen de la procédure Command1_Click

Cette procédure est exécutée quand on appuie sur le bouton « Jouer ». Elle exécute 3 tâches :

1. Masque la pile de pièces ;
2. Crée trois chiffres aléatoires pour les trois fenêtres ;
3. Affiche la pile de pièces si un 7 apparaît.

Enregistrement du programme

Enregistrer le programme et l'exécuter.

1.4 Utilisation des contrôles

Points étudiés

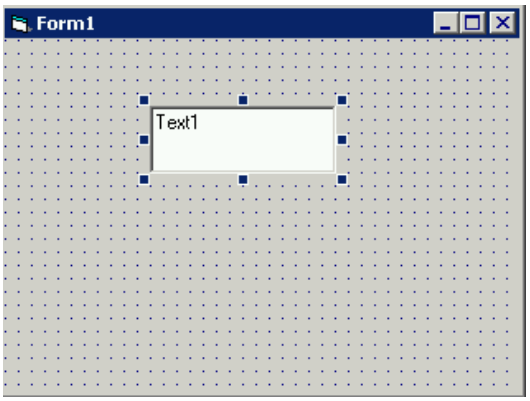
Les contrôles sont les outils graphiques utilisés pour construire l'interface utilisateur. Il s'agit dans cette leçon d'apprendre à utiliser les contrôles pour créer des objets VB recueillant les entrées des utilisateurs durant l'exécution des programmes.

Application 1 : affichage d'un texte

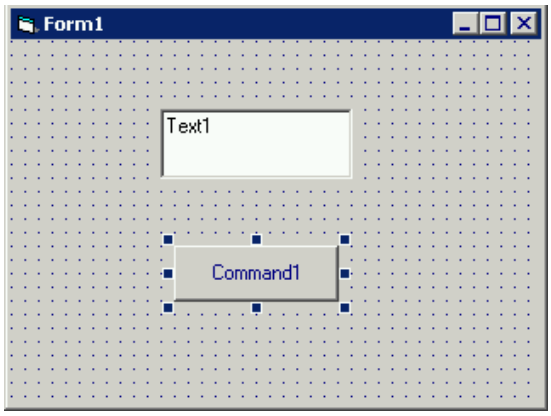
C'est un petit exercice d'affichage d'un message « Hello World », utilisé dans tous les langages comme premier pas vers la programmation.

Création de l'interface correspondante

- Démarrer VB et cliquer sur « ouvrir » ;
- Sélectionner le contrôle **TextBox** et l'amener au milieu de la feuille ;
- Tracer une zone de texte au milieu de la feuille ;



- On utilise un objet « zone de texte » pour afficher du texte ou recueillir une saisie du texte par l'utilisateur
  - Ajouter la commande associée : Cliquer dans la boîte à outils sur le contrôle **Commandbutton** et créer un bouton commande sur la feuille. Cela veut dire :
    - On utilise un objet « bouton de commande » pour recueillir l'action ;
    - l'utilisateur utilise un bouton de commande pour créer un **événement** devant être traité par le programme.



Définition des propriétés des deux objets créés

- Cliquer sur la zone Text1, annuler la valeur de la propriété Text (de manière à avoir la zone de texte vide au départ) ;
- Double-Cliquer sur la zone de commande « ok » pour rentrer l'instruction dans la fenêtre de code de Command1_Click():

```
Text1.text = "Hello, World ! "
```

  - ✈ Cet exemple modifie une propriété (Text de la zone de texte) en phase d'exécution ;
  - ✈ L'instruction est placée dans une procédure événementielle, elle est exécutée dès que l'on clique sur le bouton de commande.

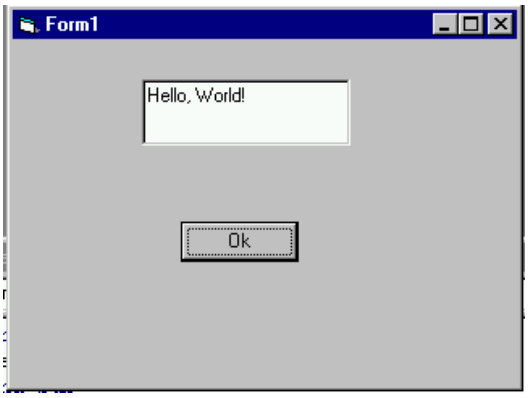


Figure 8: Exécution de la commande "Ok"

Application 2 : réalisation d'un browser

Il s'agit de permettre, lors de l'exécution, la sélection d'un fichier image et d'en afficher le contenu. La sélection concerne :

- La partition : a, c, d… : une fenêtre spéciale sera ouverte à cet effet ;
- Le répertoire : une fenêtre spéciale sera ouverte à cet effet avec un menu déroulant. Le contenu sera mis à jour à partir de la sélection de la partition ;
- Le fichier : une fenêtre spéciale sera ouverte à cet effet avec un menu déroulant. Le contenu sera mis à jour à partir de la sélection du répertoire.

Création de l'interface correspondante

- Ouvrir VB et cliquer sur « ouvrir » ;
- Assurez-vous que dans le menu Options, sous Editeur, la case « Déclaration des variables obligatoire » ne soit pas sélectionnée ;
- Agrandir la feuille de manière à disposer du maximum de place : des ascenseurs peuvent se créer.

Création de la zone du lecteur

- Utiliser le contrôle DriveListBox ;
- Le lecteur actif est positionné.

Création de la zone du répertoire

- Utiliser le contrôle DirListBox ;
- Le répertoire actif est positionné.

Création de la zone du fichier

- Utiliser le contrôle FileListBox ;
- Le fichier actif est positionné.

Création de la zone image

- Utiliser le contrôle Image et ouvrir une grande fenêtre. L'interface doit ressembler à ceci :

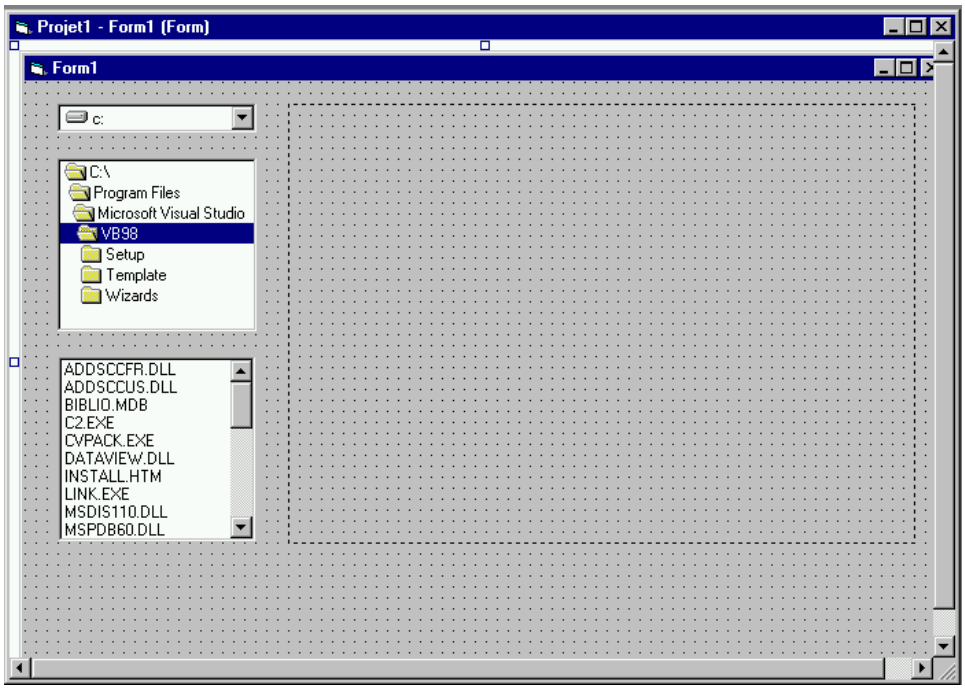


Figure 9: Interface d'un browser

Mise à jour des propriétés

- Le fichier image doit être d'un des types suivants : *.bmp, *.mf, *.ico ;
- Image1 : Stretch = 1-Fixed Single, BorderStyle = 1-Fixed Single.

Création des procédures événementielles correspondantes

Ecrire le code suivant :

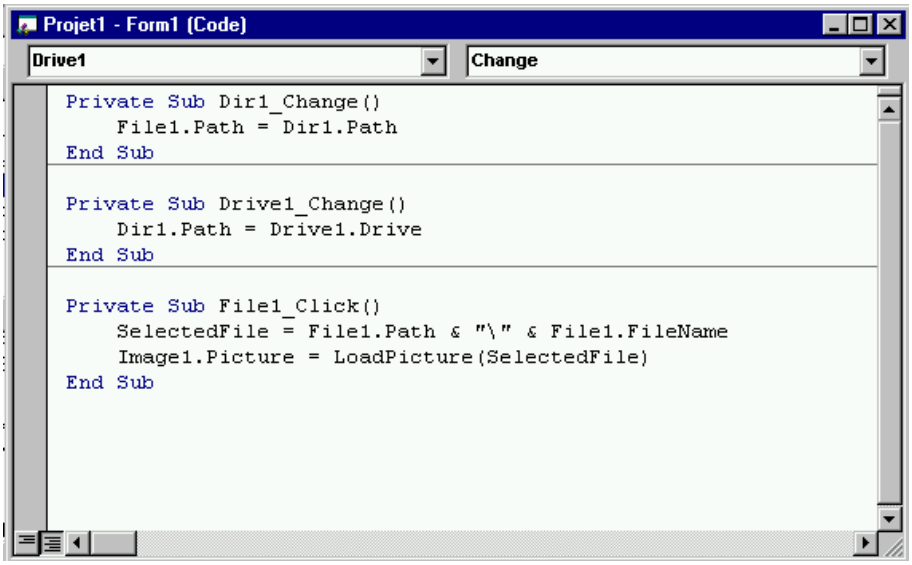


Figure 10: Code de procédures événementielles

Application 3 : Utilisation d'un Objet OLE

Il s'agit de créer une interface de boutons d'applications à l'aide du contrôle OLE. Comme le montre la figure suivante, il s'agit de créer 3 boutons correspondant aux applications Microsoft : Word, Excel et Paint

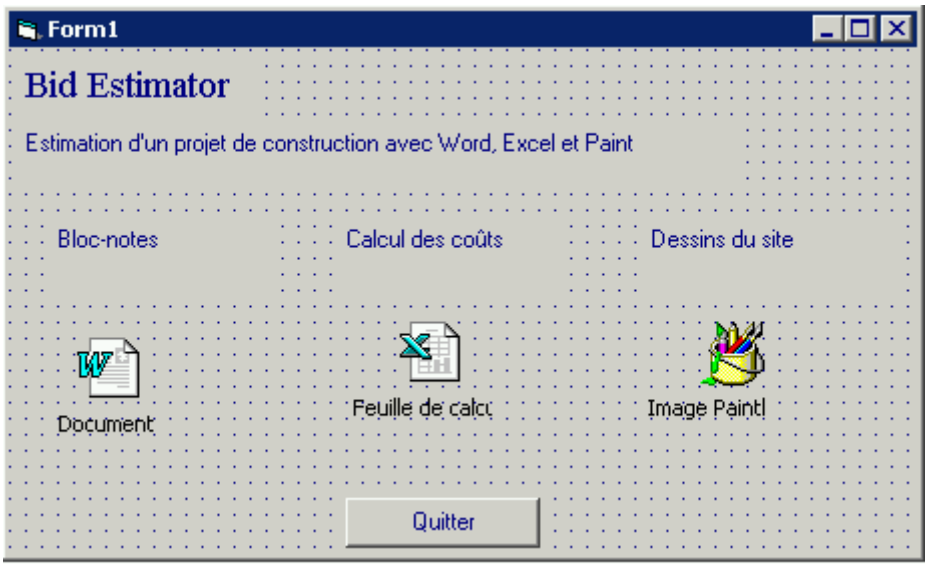


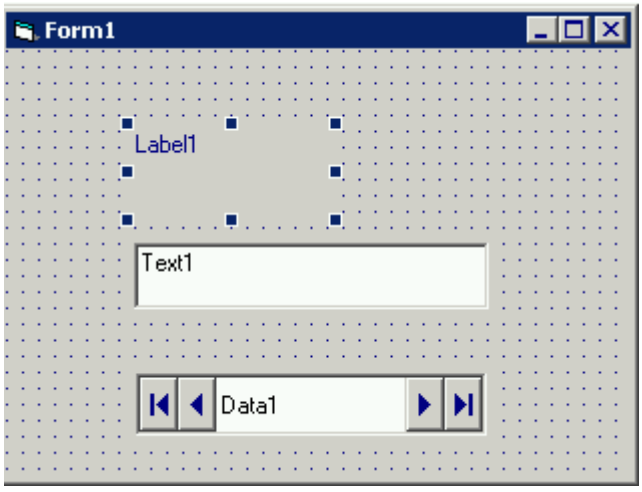
Figure 11: Interface d'applications par l'objet OLE

Construction

- Créer les labels tel que mentionné dans la figure ;
- Utiliser le contrôle OLE pour créer trois rectangles en dessous des labels : bloc-notes, calcul des coûts et Dessins du site ;
- Une fenêtre d'objets s'affiche. Faire défiler les objets et choisir l'application souhaitée.
  - Changer les propriétés des objets :
    - Pour Label 1 : Font : Times…
    - Pour Ole1, Ole2 et Ole3 : BorderStyle (0-None), Appearence (0-Flat) et BlackColor (Gris Clair)
  - Mettre le code End dans la procédure événementielle Command1_Click

Application 4 : création d'un objet donné pour visualiser une base de données

- Utiliser le contrôle Data pour créer une zone Data ;
- Utiliser le contrôle TextBox pour créer une fenêtre de Texte aussi grande que la fenêtre Data ;
- Utiliser le contrôle Label pour introduire Label1.



Fixer les propriétés des objets

- Pour Data

- **Connect** : Access : enregistre le format de base de données ou du tableur. Les formats susceptibles d'être lus par VB sont : Access, Excel, Lotus 1-2-3, dBase, FoxPro, et Paradox ;
- **DataBaseName** : une fenêtre s'ouvre : choisir un fichier base de données existant, par ex. Comptoir : "C:\Program Files\Microsoft Office\Office\Exemples\comlptoirs.mdb" ;
- **RecordSource :** Employés
- **Caption :** Comptoir.mdb

- Pour Texte
  - **DataSource :** Data1
  - **DataField :** Nom
- Pour Etiquette :
  - **Caption :** Nom

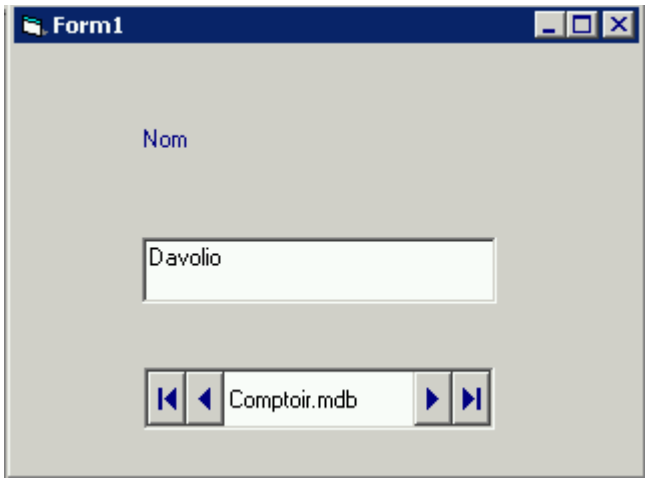


Figure 12: Interface de modification de la base de données

Modifier la base de données

Il s'agit de changer un nom, pour cela :

- Se placer sur un des noms à remplacer ;
- Le noircir avec la souris ;
- Le changer ;
- Passer au nom suivant et revenir pour vérifier le changement ;
- Le nom est enregistré ;
- Arrêter l'exécution.

2 Programmation en VB

2.1 Principes de base de la programmation

Anatomie d'une instruction VB

C'est un ordre donné à la machine pour réaliser un travail donné. Cet ordre peut être de différentes natures : lecture d'une donnée, affichage d'un résultat, ouverture d'un fichier, calcul d'un résultat, action système, etc. L'instruction a une syntaxe qui dénote une phrase dans le langage d'expression des ordres.

Exemples :

```
Beep     // un seul mot clé dont l'effet est d'émettre un son

Label1.Caption = Time  // syntaxe plus développée conduisant à affecter par le signe = la valeur de la fonction de VB Time à la propriété Caption de l'objet Label1.
```

Notion de variable

C'est un conteneur de valeur.

- Déclaration :

```
DIM Nom
```

- Réserve un espace mémoire. En l'absence de déclaration de type, Nom est déclaré de type variant et peut contenir aussi bien des nombres que du texte.

- Déclaration implicite sans DIM

```
Nom = "Toto"          Valeur1 = 24          Valeur2 = 38.5
```

Exercice : écrire une interface VB affichant le contenu d'une variable Nom, contenant successivement le texte "smart" et le nombre 99.

Correction :

1. Créer un nouveau projet VB (commande ouvrir) ;
2. Créer deux boutons : Afficher et Arrêter ;
3. Changer Caption de Command1 en Afficher ;
4. Changer Caption de Command2 en Arrêter ;
5. Créer deux labels : Label1 et Label2 associés au bouton Afficher ;
6. Changer les propriétés des labels : Caption =vide, BorderSTyle = 1-Fixed Single, Font = MS sans sérif, normal, 12 ;
7. Introduire comme code dans la procédure événementielle Command1_Click associé à Label1 et Label 2:

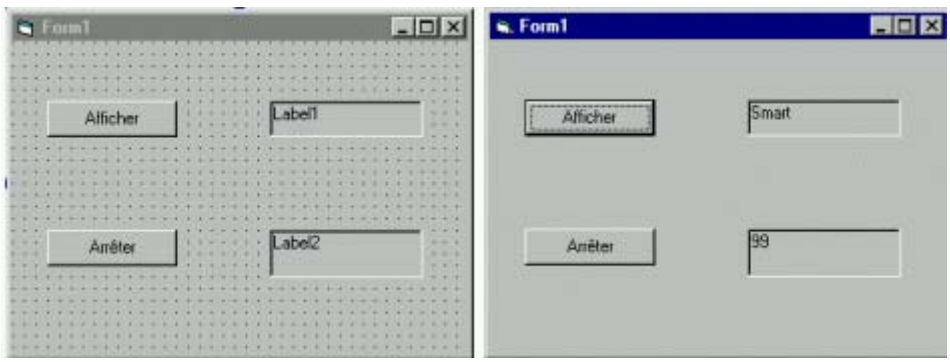
```
Dim Nom
Nom = "Smart"
Label1.Caption = Nom

Nom = 99
Label2.Caption = Nom
```

8. Introduire comme code dans la procédure événementielle Command2_Click :

```
End
```

9. Exécuter le programme



Utiliser une variable pour l'entrée : InputBox

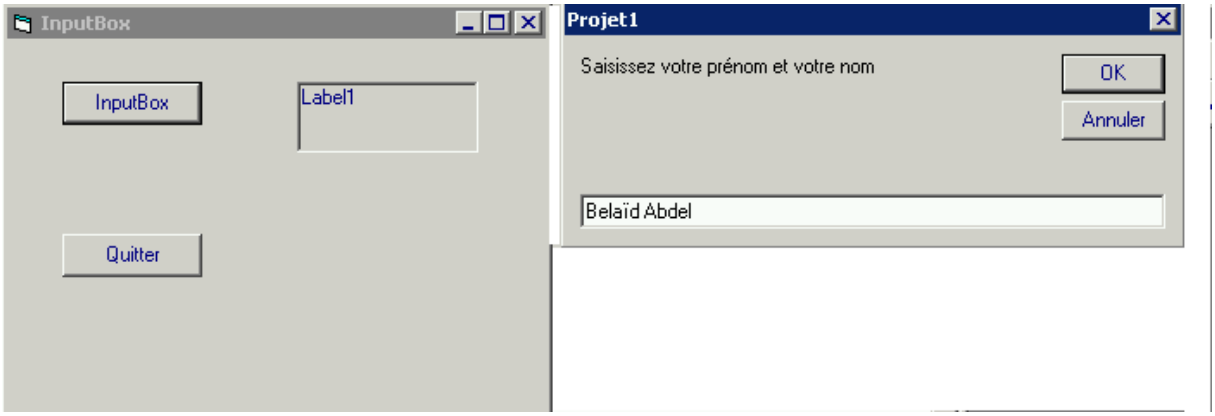
InputBox est une fonction de lecture VB qui renvoie une valeur. Cette valeur est affectée à une variable.

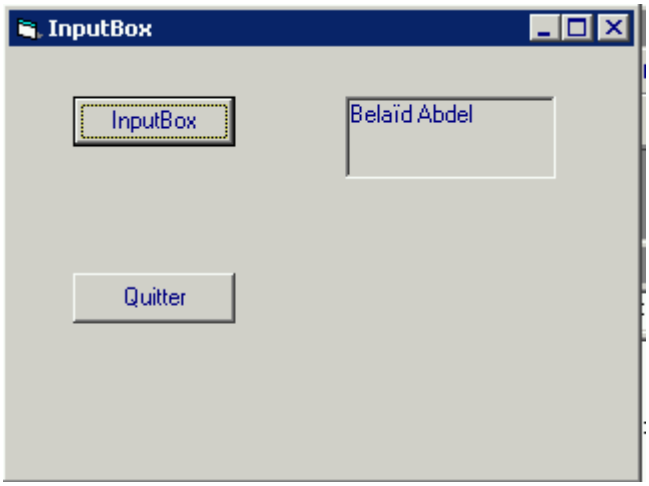
Pratique :

1. Créer un nouveau projet VB ;
2. Créer deux boutons commandes, l'un appelé InputBox et l'autre Quitter (agir sur leur Caption pour indiquer ces noms) ;
3. Créer un bouton label, le creuser en agissant sur BorderStyle ;
4. Mettre End dans le code de la commande Quitter ;
5. Inscrire le code suivant pour la commande InputBox.

```
Dim Invite, NomComplet
Invite = "Saisissez votre prénom et votre nom"
NomComplet = InputBox$(Invite)
Label1.Caption = NomComplet
```

La variable Invite est utilisée comme argument de la fonction InputBox, c.à.d. que la fonction InputBox affiche d'abord le message passé dans la variable Invite, lit le nouveau message et le range dans la variable NomComplet.





Utiliser une variable pour la sortie : MsgBox

Il est possible d'afficher le contenu d'une variable en l'affectant à une propriété (par ex.. caption d'un objet étiquette) ou en le passant comme argument à une fonction de boîte de dialogue.

La syntaxe de MsgBox est :

```
BoutonCliqué = MsgBox(Message, NumeroDeBouton, Titre)
```

- Message : texte affiché à l'écran ;
- BoutonCliqué : est un numéro de style de bouton (de 1 à 5) ;
- Titre : texte affiché dans la barre de titre de la boîte de dialogue ;
- La variable BoutonCliqué est affectée du résultat livré par la fonction MsgBox, c'est le bouton sur lequel l'utilisateur a cliqué dans la boîte de dialogue.

Pratique :

1. Réutiliser le projet précédent ;
2. Ajouter dans le code de la procédure Command1_Click l'instruction suivante avant Label1.Caption = NomComplet : `MsgBox (NomComplet), , "Résultat de l'entrée"`
3. Exécuter : une boîte de dialogue apparaît pour confirmer l'entrée du label avant de l'afficher dans label1, comme montré ci-après :

Principaux types de données VB

Type de données	Taille	Plage	Exemple
Entier(Integer)	2 octets	-32768 à 32768	Dim Oiseaux% Oiseaux%=37
Entier long (Long Integer)	4 octets	-2147483648 à +2147483648	Dim Salaire& Salaire& = 350000
Virgule flottante simple précision	4 octets	-3,402823 ^{E}38 à 3,402823 ^{E}38	Dim Prix! Prix = 899,90
Virgule flottante double précision	8 octets		Dim Pi# Pi# = 3,1415926535
Monétaire (Currency)	8 octets		Dim Debit@ Debit@ = 7600300,50
Chaîne (String)	1 octet par caractère	De 0 à 65535 caractères	Dim Chien$ Chien$ = "pointer"
Booléen (Boolean)	2 octets	Vrai ou Faux	Dim Flag as Boolean Flag = True
Date	8 octets	1 janvier 100 au 31 décembre 9999	Dim Anniv as Date Anniv = #1/3/63#
Variant	16 octets (pour les nombres), 22 octets + 1 octet par caractère (pour les chaînes)	Toutes les plages des autres types	Dim Total Total = 289,13

Type de données personnalisé

VB permet de créer des types de données personnalisés grâce à l'instruction **Type**

Exemple :

```
Type Personnel
Nom As String
DateDeNaissance As Date
DateEmbauche As Date
End Type
```

Après la création du type, on peut l'utiliser dans le programme :

Exemple :

```
Dim ChefProduit As Personnel
ChefProduit.Name = "Eric Cody"
```

Portée d'une variable

- **Variable locale** : Une variable déclarée par Dim dans une **procédure** n'a d'existence que dans la procédure où elle est déclarée ;
- **Variable globale** : Une variable déclarée par Dim dans la section générale d'un **module**. Elle est utilisée dans toutes les procédures événementielles et toutes les procédures de feuille. Un module est un fichier spécial contenant des déclarations et des procédures qui ne sont pas associées à une feuille spécifique.

Notion de constante

Une constante est une variable qui ne varie pas
Exemple :

```
Const Pi = 3,14159265
```

Elle est utilisée à l'endroit où elle est déclarée.
Pour qu'une constante soit disponible dans l'ensemble des feuilles et modules d'un programme, créer la constante dans un module standard, précédée du mot-clé **Public.**

Exemple :

```
Public Const Pi = 3,14159265
```

Notion d'Opérateurs

Une formule est une instruction combinant des nombres, des variables, des opérateurs et des mots-clés pour créer une nouvelle valeur.

Opérateur	Opération mathématique
+	addition
-	soustraction
*	multiplication
/	division
\	Division entière
Mod	Modulo (reste de la division entière)
^	Puissance
&	Concaténation de chaînes

Pratique : créer l'interface suivante permettant de réaliser les opérations mathématiques binaires, à la manière de l'interface de la Figure 13.

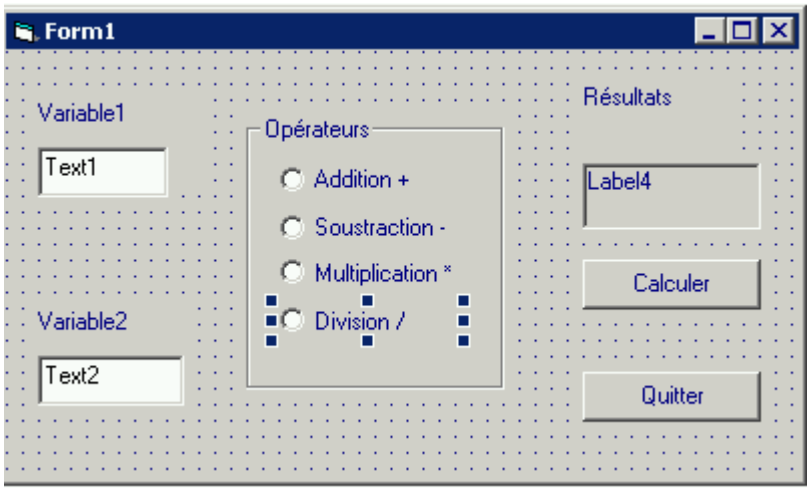


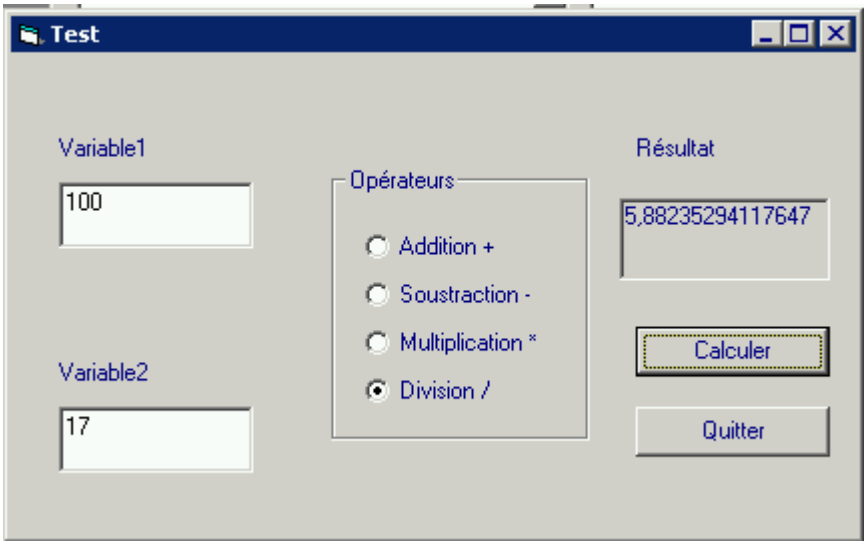
Figure 13: Interface de calcul

1. Deux contrôles label, Label1 et Label2 portant le texte Variable 1 et Variable 2 dans Caption ;
2. Deux contrôles TextBox, Text1 et Text2. Effacer leur Caption ;
3. Un contrôle Frame pour créer la zone des opérateurs, changer sa Caption ;
4. Utiliser la commande OptionButton avec comme Caption le texte de l'opérateur pour introduire les opérateurs ;
5. Un contrôle Label pour Label3 portant le texte Résultat dans la propriété Caption ;
6. Un contrôle Label, Label4, dont la propriété Caption est vide et Borderstyle à 1-Fixed Single ;
7. Un contrôle CommandButton portant le texte Calculer dans la propriété Caption et un autre portant le texte Quitter dans Caption ;
8. Mettre End comme Code à la procédure de Quitter ;
9. Mettre le code suivant dans la procédure de Calculer.

```
Dim Premier, Second 'Déclaration de variables
Premier = Val(Text1.Text) 'Conversion en nombre
Second = Val(Text2.Text) 'Conversion en nombre

    'Si le premier cercle est activé
    If Option1.Value = True Then
    Label4.Caption = Premier + Second
    End If
    'Si le deuxième cercle est activé
    If Option2.Value = True Then
    Label4.Caption = Premier - Second
    End If
    'Si le troisième cercle est activé
    If Option3.Value = True Then
    Label4.Caption = Premier * Second
    End If
    'Si le quatrième cercle est activé
    If Option4.Value = True Then
    Label4.Caption = Premier / Second
    End If
```

10. Exécuter, en mettant des valeurs aux variables et en sélectionnant l'opérateur.



Notion de fonctions mathématiques

Visual Basic met à disposition plusieurs fonctions mathématiques utilisables dans les programmes, comme Abs (calculant la valeur absolue d'un nombre), des fonctions trigonométriques : Cos, Tan, Sin…

Programmation pilotée par les événements

La méthodologie apprise jusqu'à maintenant et qui consiste à passer par les menus et attendre l'action de l'utilisateur, s'appelle **programmation pilotée par les événements**. Le programme est construit à partir d'objets intelligents qui savent se comporter lorsque l'utilisateur les active.

La Figure 14 montre comment un programme pilotée par les événement fonctionne.

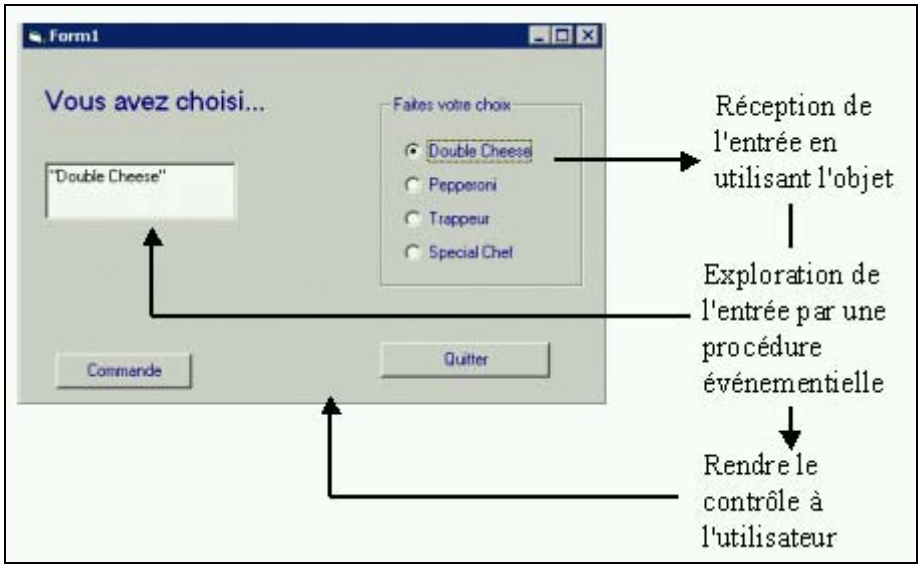


Figure 14: programmation événementielle

Evénements pris en charge

- Chaque objet VB dispose d'un jeu prédéfini d'événements auquel il est susceptible de répondre. Ces événements sont répertoriés dans la liste déroulante des procédures de la fenêtre Code ;

- Pour chacun de ces événements, on a la possibilité d'écrire une procédure événementielle. Si l'événement intervient au cours de l'exécution du programme, la procédure événementielle sera exécutée ;
- Ainsi, un objet liste déroulante prend en charge les événements Clik, DblClick, DragDrop, GotFocus, ItelChek, KeyDown, KeyPress, KeyUp, LostFocus, MouseDown, MouseMove, MouseUp, PLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedBack, OLESetData, OLEStartDrag, Scroll et Validate ;
- En principe, on ne peut programmer que deux ou trois de ces événements ;
- La Figure 15 montre un extrait de la liste des événements pour un objet de liste déroulante dans la fenêtre Code

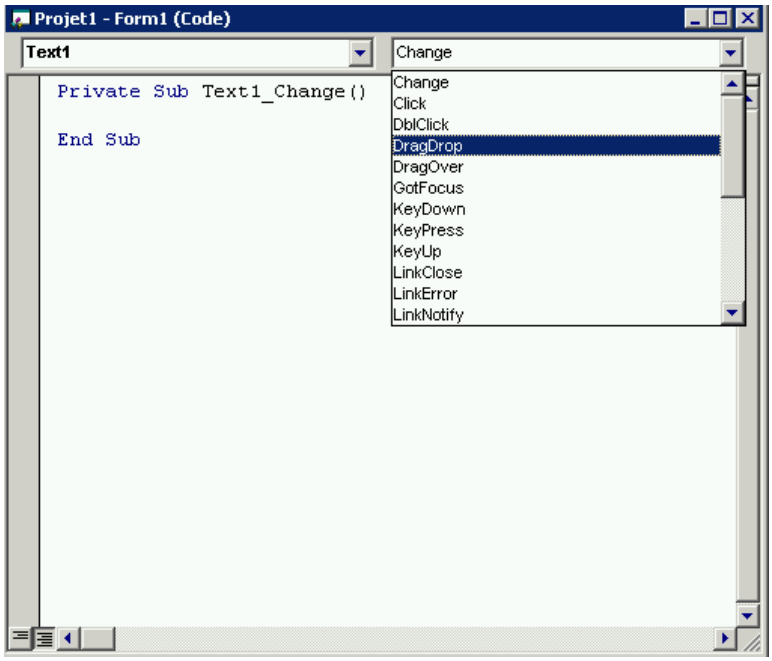


Figure 15: liste d'événements

Les expressions conditionnelles

Opérateurs de comparaison

Exemples :

- 10 <> 20 donne vrai,
- Score < 20 donne vrai si la valeur de score est inférieure à 20
- Score = Label1.Caption donne vrai …
- Text1.text = "Jean" donne vrai si …

Structures de décision If…Then

Forme 1: If Condition Then Instruction

Exemple : If Score>=20 Then Label1.Caption = "Vous avez gagné !"

Forme 2 :
```
If Condition1 Then Instructions1
ElseIf Condition2 Then  Instruction2 …
Else
Instructions-finales
End If
```

Condion1 est évaluée en premier, puis Instructions1 sont évaluées si Condion1 est vraie …

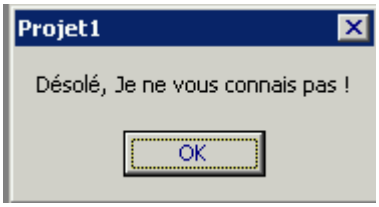
Pratique : écrire une interface qui valide à la connexion, pour une machine, le nom de son utilisateur

1. Démarrer un nouveau projet
2. Utiliser CommandButton pour créer une commande dont la Caption sera Ouverture de session
3. Double-cliquer sur cette commande et rentrer le code suivant :

```
NomUtilisateur = InputBox("Saisissez votre nom")
If NomUtilisateur = "Laura" Then
MsgBox ("Bonjour, Laura ! En forme pour attaquer le travail ? ")
Form1.Picture = _
LoadPicture("C:\Program Files\Microsoft Office\Clipart\Popular\Approuv.wmf")
ElseIf NomUtilisateur = "Marc" Then
MsgBox ("Bonjour, Marc ! Prêt à attaquer le travail ? ")
Form1.Picture = _
LoadPicture("C:\Program Files\Microsoft Office\Clipart\Popular\Approuve.wmf")
Else
MsgBox ("Désolé, Je ne vous connais pas !")
   End 'Abandon du programme
End If
```

4. Exécuter.
5. Le résultat positif est illustré ci-après :

6. Le résultat négatif est illustré ci-après :



Opérateurs logiques

On peut écrire des expressions conditionnelles complexes en utilisant les opérateurs logiques suivants : and, or, not, xor

Pratique : compléter l'exercice précédent par le contrôle du mot de passe. Pour cela : il suffit d'ajouter après l'entrée du nom :

```
Pass = InputBox ("Saisissez votre mot de passe")
```

```
If NomUtilisateur = "Laura" And Pass ="May17" Then …
```

Select Case

C'est une sélection à choix multiple :

Forme :

```
Select  Case variable
        Case Value 1
                Instruction1
        Case value2
                Instruction2
        …
End Select
```

Les boucles et le Timer

Boucle For … Next

Elle permet d'exécuter un groupe d'instruction un certain nombre de fois dans le cadre d'une procédure événementielle.

Syntaxe :

```
For variable = start To end
        Instructions à exécuter répétitivement
Next Variable
```

Exemple :

```
For i = 1 To 4  {step j}
        Beep
Next i
```

Application 1 : affichage du compteur de la boucle par Print

…

1. Sélectionner Form1
2. Mettre AutoRedraw à True
3. Double-cliquer sur la commande Boucle et mettre le code suivant…
4. Exécuter

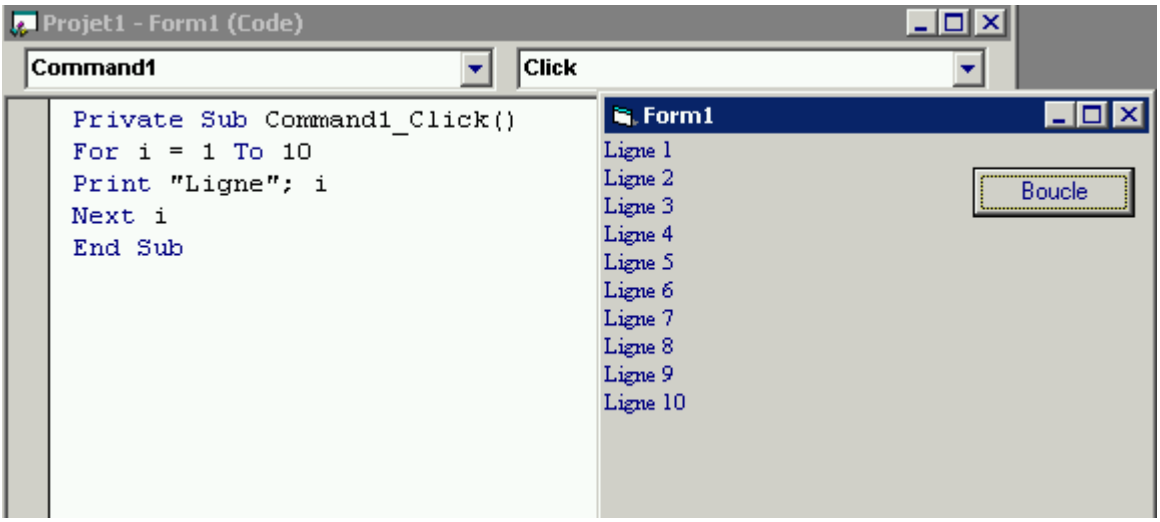
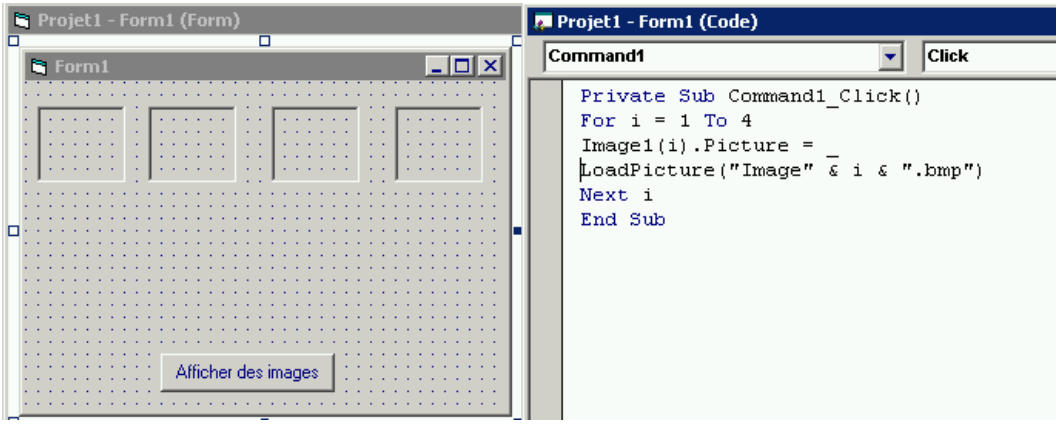


Figure 16: Affichage du comptage

Application 2 : affichage d'images

1. Préparation d'imagettes :
   - Ouvrir la fenêtre Poste de travail de Windows et passer en mode d'affichage Grandes icônes par le menu Affichage
   - Appuyer sur les touches ALT+IMPR ECRAN : cette commande a pour effet de copier la fenêtre active dans le Presse-papiers de Windows
   - Charger Paint
   - Appeler la commande Coller du menu Edition
   - Activer l'outil Sélection et tracer un cadre de sélection autour de la première icône
   - Appeler Copier du menu Edition pour placer une copie de cette icône dans le Presse-papiers
   - Appeler la commande "Copier vers" du menu Edition. Dans la boîte de dialogues, localiser le répertoire Exercices et ranger l'image sous le nom Image1.bmp
   - Refaire la m^me chose pour les 3 autres images

2. Création de l'application
   - Utiliser le contrôle Image et créer un petit contrôle image en haut de la feuille
   - Dans le menu Edition, Cliquer sur la commande Copier, une copie du contrôle Image est placée dans le presse Papiers de Windows. On va s'en servir pour créer trois nouveaux contrôles Image sur la feuille.
   - Appeler la commande Coller du menu Edition, répondre oui à la création d'un groupe de contrôle. Un deuxième contrôle Image apparaît, le glisser pour le superposer au rectangle de l'image.
   - Appeler Coller du menu Edition … et faire cela pour toutes les images
   - Créer un bouton de commande, appelée "Afficher les images" et lui affecter le code mentionné dans la figure.
   - Enregistrer la feuille et le projet sous les MonGroupeContrôles, puis exécuter.



Instruction EXIT FOR

Cette instruction permet d'interrompre la boucle et de sortir :

Exemple :

```
For i = 1 To 10
        Nom = InputBox("Saisissez votre nom ou tapez Fini pour quitter")
        If Nom ="Fini" Then Exit For
        Print Nom
Next i
```

Boucle Do…Loop

Elle offre une alternative aux boucles For…Next. Elle exécute le bloc jusqu'à ce qu'une condition définie devienne True

Syntaxe :

```
Do While condition
        Instructions à exécuter répétitivement
Loop
```

Exemple :

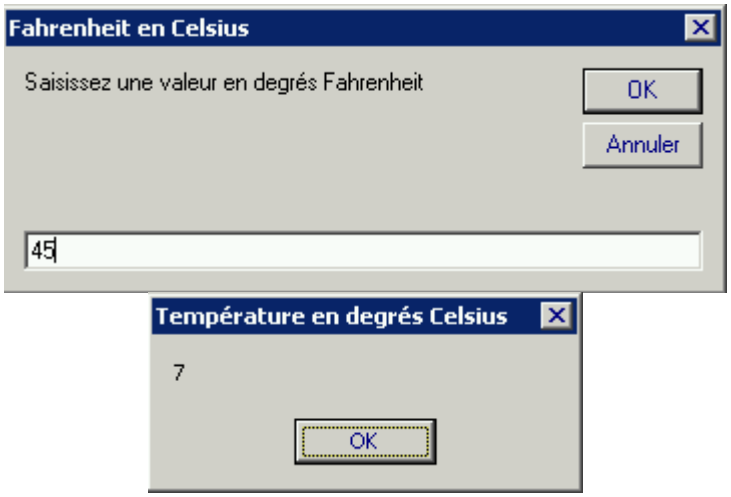
```
Do While Nom <> "Fini"
        Nom = InputBox(("Saisissez votre nom ou tapez Fini pour quitter")
        If Nom <> "Fini" Then Print Nom
Loop
```

Application :

- Créer un nouveau projet ;
- Ouvrir la fenêtre Propriétés de la feuille, mettre Visible à False (exécution en arrière plan) ;
- Double-cliquer sur la feuille et entrer le code suivant :

```
Prompt = "Saisissez une valeur en degrés Fahrenheit"
Do
    Ftemp=InputBox(Prompt, "Fahrenheit en Celsius")
    If  Ftemp <> "" Then
            Celsius = Int(Ftemp + 40) * 5/9-40)
            MsgBox (Celsius),  , "Température en degrés Celsius"
    End If
Loop While Ftemp <> ""
End
```

Le résultat est :



On peut utiliser le Until :

Exemple :

```
Do Nom = InputBox(("Saisissez votre nom ou tapez Fini pour quitter")
        If Nom <> "Fini" Then Print Nom
Loop Unitil Nom = "Fini"
```

Objet Timer

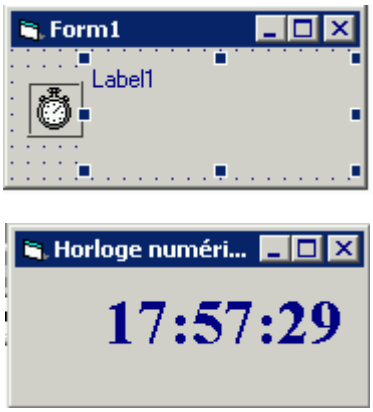
- VB permet d'exécuter un groupe d'iunstructions pendant un Laps de temps déterminé en utilisant un objet Timer.

- C'est un objet horloge invisible permettant d'accéder à l'horloge système à partir d'un programme

Application 1 : création d'une horloge numérique

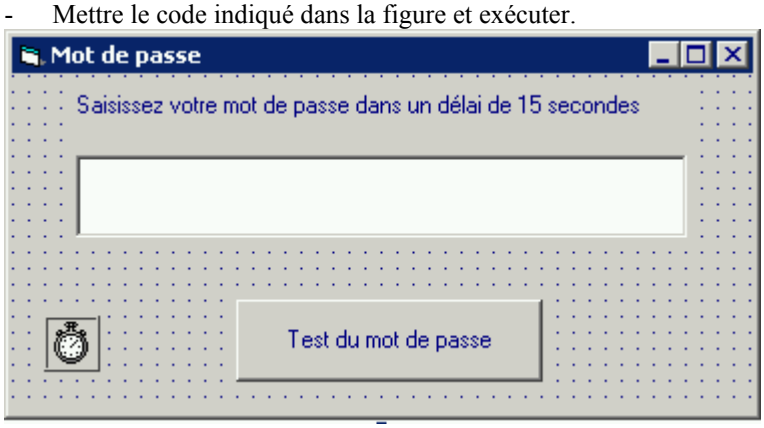
- Créer un nouveau projet, redimensionner la feuille ;
- Introduire le contrôle Timer ;
- Cliquer sur le contrôle Label ;
- Donner les propriétés suivantes :
  - Label 1 : Caption (vide), Font (Times, gras, 24), Alignement (2-Center) ;
  - Timer1 : Interval (1000), Enabled (True);
  - Form1 : Caption (Horloge numérique) ;
- Double-Cliquer sur Timer et rentrer l'instruction suivante :

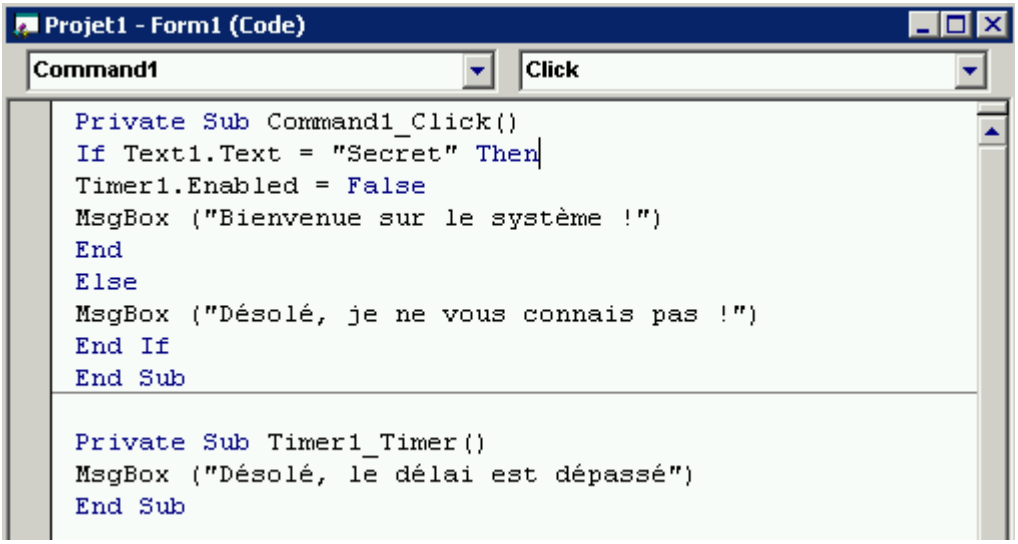
  Label1.Caption = Time



Application 2 : contrôler le délai de saisie du mot de passe

- Créer un nouveau projet ;
- Créer une zone de texte, une commande, un label et un Timer ;
  - Text1 : Text (vide), PasswordChar (*) ;
  - Form1 : Caption (Mot de passe) ;
  - Label1 : Caption (Saisissez…) ;
  - Command1 : Caption (Test…);
  - Timer1 : Interval (15000), Enabled (True);
- Mettre le code indiqué dans la figure et exécuter.





Modules et procédures

Notion de module strandard

C'est un conteneur isolé qui contient :

- Des variables globales ou publiques
- Des procédures Function et Sub

Il permet de piloter plusieurs procédures, plusieurs feuilles partageant des variables globales. Il est suffixé par **.bas**

Création et enregistrement

Dans le menu **Projet**, cliquer sur **Ajouter un module**, puis cliquer sur **Ouvrir**. VB ajoute un module standard appelé Module1. Dans la fenêtre du projet, les modules apparaissent avec les feuilles. La figure suivante montre un exemple d'un tel module:

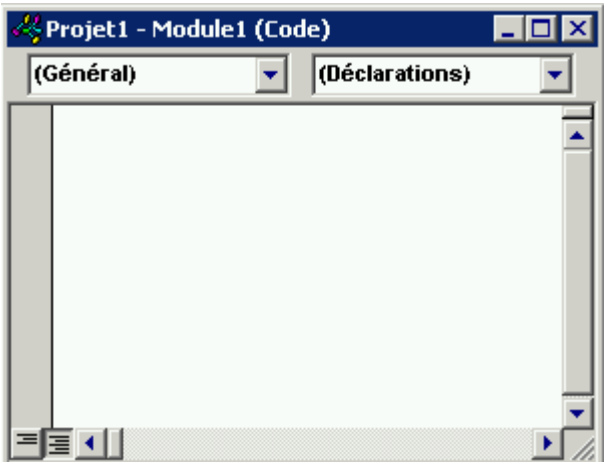


Figure 17: Module standard

- La partie Général donne la liste des objets et procédures déclarés dans le module
- Comme un module standard ne contient pas d'objet, sa seule propriété est **Name**, Name permet de spécifier le nom du module, ce qui permet d'en créer plusieurs.
- Renommer Name en modVariables
- Par convention, les noms de modules sont précédés du préfixe **mod**

Variables publiques

Public nom_variable (déclaration par défaut)
Public nom_variable As string

Exemple : révision du projet **LuckySeven**

1. Ouvrir le projet et afficher Lucky.frm
2. L'enregistrer sous les noms Gains.frm et Gains.vbp
   - Créer avec le contrôle Label une étiquette sous Lucky Seven
   - Définir les propriétés suivantes : label5 : Alignment (2-center), Caption (Gains:0), Font (Arial, Gras Italique, 12 pts), ForeColor (Vert), Name (IblGains), Form1 : Caption (Lucky Seven)
3. Ajouter un module standard
   - Saisir : **Public Gains**, puis **Entrée**
   - Enregistrer le module sous **Gains.bas**
4. Ouvrir la procédure événementielle correspondant au bouton "Jouer" et ajouter après l'instruction Beep :
   Gains = Gains + 1; // incrémente la variable publique Gains chaque fois qu'un 7 apparaît
   IblGains.Caption = "Gains :" & Gains // sert à l'affichage

   L'intérêt de cette déclaration est d'éviter d'initialiser Gains chaque fois qu'on joue et qu'on appelle la procédure de jeu.

Figure 18: Affichage du résultat de la variable publique

Notions de procédures et de fonction

Dans VB, il y a deux types de procédures, les procédures événementielles et les procédures à caractère général.

Les **procédures événementielles** sont associées à un événement se produisant au cours de l'exécution, ou à un objet créé depuis la boîte à outils.

Les **procédures à caractère général** sont des tâches (blocs d'instruction) appelées de partout. Elles ont un nom et des arguments. Parmi ces procédures, on trouve : les procédures de fonction, les procédures Sub et les procédures Property.

les procédures de fonction

Elles ont un nom, comprennent des arguments et retournent un résultat.

Syntaxe :

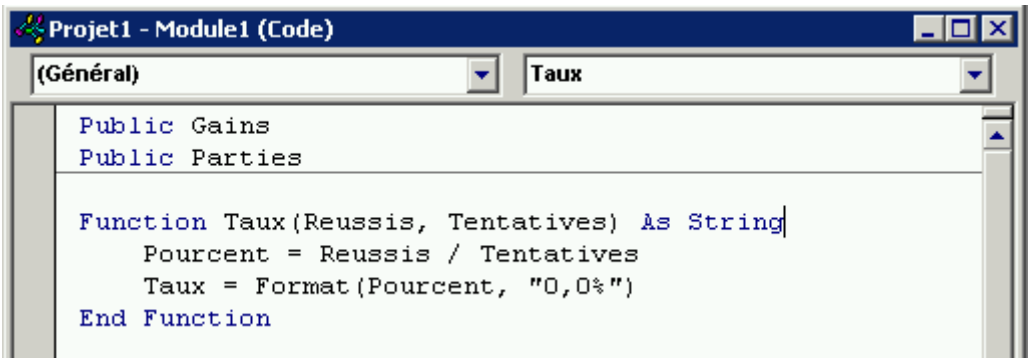
Function NomFunction ([arguments]) [As Type]
    Instructions de la fonction
End Function

Les arguments sont séparés par des virgules. La fonction retourne toujours une valeur portée par son nom.

Exemple : ajouter une fonction au programme Lucky Seven pour calculer le taux de réussite. Pour cela mettre en place une fonction taux et une variable publique appelée Jeux dans le module standard. Cette fonction sera appelé chaque fois que le bouton "jouer" est activé. Le résultat doit apparaître dans un nouveau label à placer sur la feuille.

1. Enregistrer le projet précédent sous le nom Reussite
2. Créer une nouvelle étiquette Label sous Gains avec comme propriétés pour Label5 : Alignment (2-center), Caption (0,0%), Font (Arial, Gras Italique, 12 points), ForeColor (Rouge), Name(lblTaux)
3. Dans la fenêtre Projet, double-cliquer sur Reussite.bas pour l'ouvrir dans la fenêtre code, saisir:

   Public Parties
4. Saisir la déclaration de fonction suivante dans le module standard



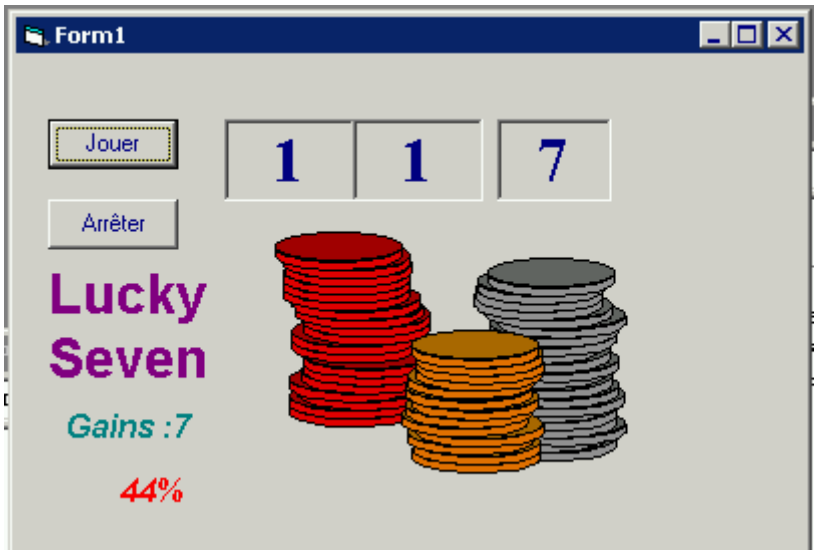
5. Ajouter les 2 instructions suivantes :
   - Celle là après la dernière instruction comprenant Rnd :

     Parties = Parties + 1

   - Celle là entre End If et End Sub :

     lbltaux.Caption = Taux(Gains, Parties)

6. Exécuter le programme



Les procédures Sub

Syntaxe :

```
Sub NomProcedure ([Argument])
Instructions
End Sub
```

Les noms et nombre d'arguments doivent correspondre aux noms et nombre à l'appel.

Exemple :

```
Sub AjouteNomAliste(personne$)
        If personne$ <> "" Then
                Form1.List1.AddItem personne$
                Msg$ = personne$ & "ajouté à la liste"
        Else
                Msg$ = "Nom indéfini"
        End If
                MsgBox (Msg$), ,  "Ajoutde nom"
        End Sub
```

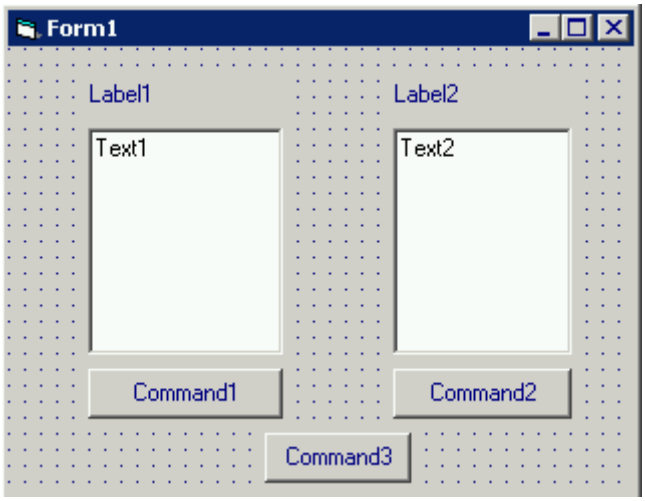
Appel :

Il suffit d'écrire le nom de la procédure et de répertorier ses arguments :

```
AjouteNomAliste "Mariane"
Ou
AjouteNomAliste NouveauNom$
```

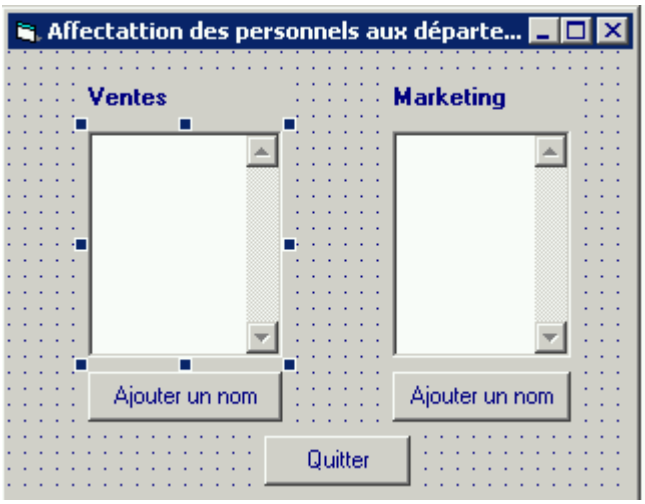
Exemple :

1. Créer un nouveau projet
2. Créer l'interface dessinée ci-après. Les zones de texte contiendront les noms de salariés dans deux départements de l'entreprise :



3. Définir les propriétés suivantes :
   - Text1 et Text2 : Text(vide), Multiline (True), Scrollbars (2-vertical), Tabstop (False), Locked (True), Name (txtVentes pour Text1 et txtMarketing pour Text2)
   - Label1 : caption(Ventes, Font (Gras), Name (lblVentes)
   - Label2 : caption(Marketing, Font (Gras), Name (lblMarketing)
   - Command1 : Caption(Ajouter un nom), Name(cmdVentes)
   - Command2 : Caption(Ajouter un nom), Name(cmdMarketing)
   - Command3 : Caption(Quitter), Name(cmdQuit)
   - Form1 : Caption (Affectation des personnels aux départements)

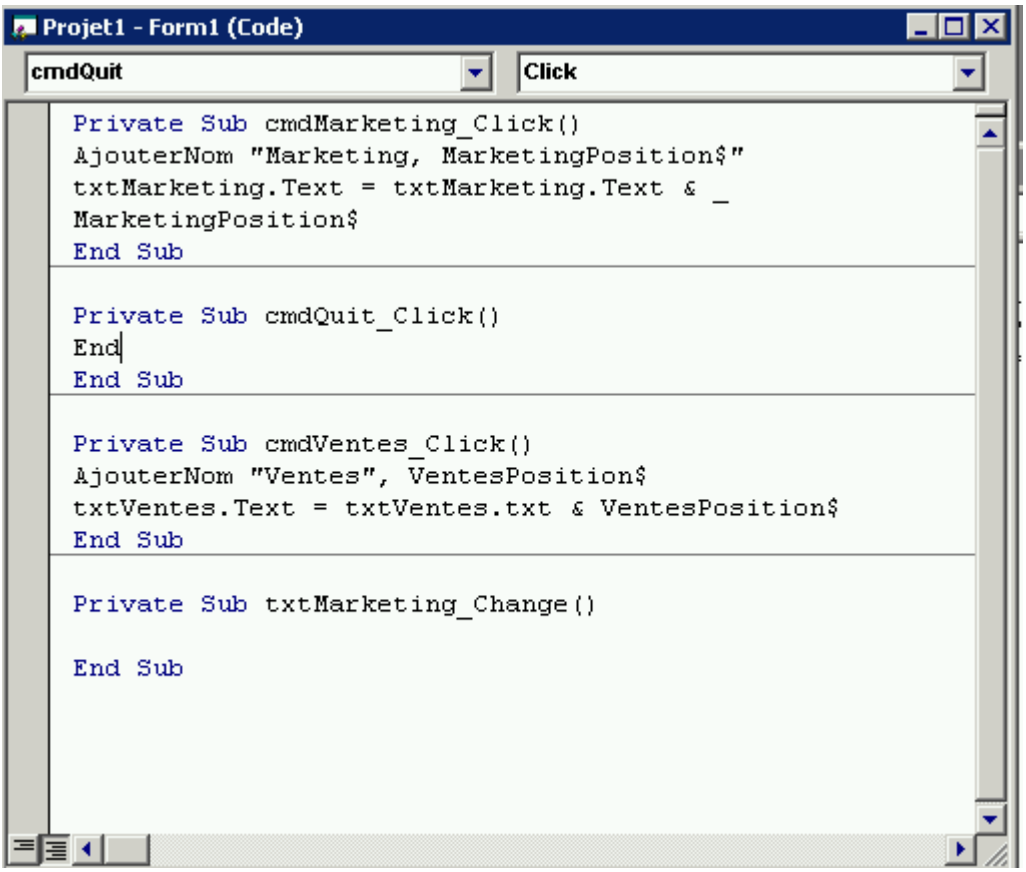
L'interface ressemble à celle dessinée ci-après.



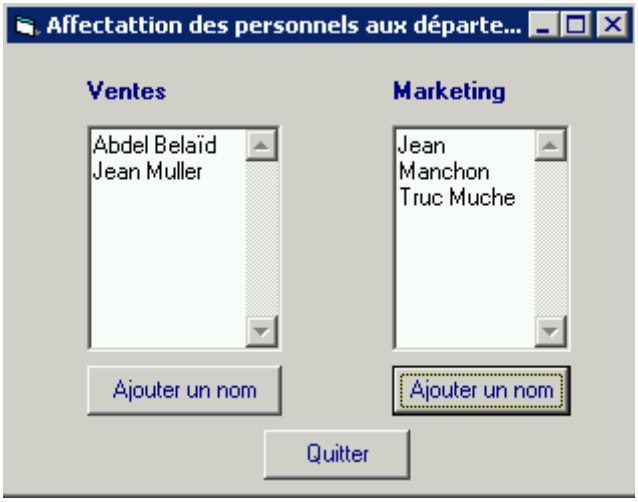
4. Ajouter un module standard, et saisir le code suivant :

```
Sub AjouterNom(Team$, ChaineRetour$)
   Prompt$ = "Saisissez un salarié de " & Team$
   Nm$ = InputBox(Prompt$, "Boîte de saisie")
   WrapCharacter$ = Chr(13) + Chr(10)
   ChaineRetour$ = Nm$ & WrapCharacter$
End Sub
```

5. Ouvrir la fenêtre code et ajouter les codes suivants :



6. Enregistrer le projet sous le nom : Départements et Exécuter



Transmission des arguments

- **Par référence :** tout changement effectué dans la procédure sur la variable est retourné à la procédure appelante. On peut accompagner une variable par le mot-clé ByVal pour forcer le passage par valeur.

  Sub CoutPlusInteret(ByVal Cout, Total)

- **Par valeur :** il suffit de placer la variable entre parenthèses

  CoutPlusInteret(Cout), Total)

Les tableaux

Déclaration d'un tableau statique

```
Public NomTableau(Dim1Eléments, Dim2Eléments, …) As TypeDeDonnées
```

Exemple :

```
Public Employés (9) As String
```

Lors de la création, VB réserve de l'espace en mémoire. La référence aux éléments se fait de 0 à 8. Cependant, si l'on veut que la référence se fasse à partir de 1, il suffit d'ajouter l'instruction suivante dans unmodule standard :

```
Option Base 1
```

Référence au tableau :

```
Employés(5) = "Leslie"
```

Déclaration d'un tableau dynamique

La déclaration en statique empêche l'extension du tableau en cas d'enregistrements de valeurs supplémentaires. Pour résoudre ce problème, on déclare le tableau en dynamique, et on le redimensionne une fois connu le nombre d'éléments

```
Public Températures () As Variant
…
Days = InputBox("Combien de jours ,", "Créer le tableau")
….
Redim Températures(Days)
```

Exemple :

1. Ouvrir un nouveau projet ;
2. Créer trois boutons comme illustrés ci-après avec les propriétés suivantes : Command1 : Caption(Entrer températures), Name (cmdEntrerTemps), Command2 : Caption(Afficher Températures), Nmae (cmdAfficherTemps), Command3 : Caption (Quitter), Name(cmdQuitter), Form1 : Caption (Températures), AutoRead(True) ;
3. Taper le code aux endroits indiqués ci-après.