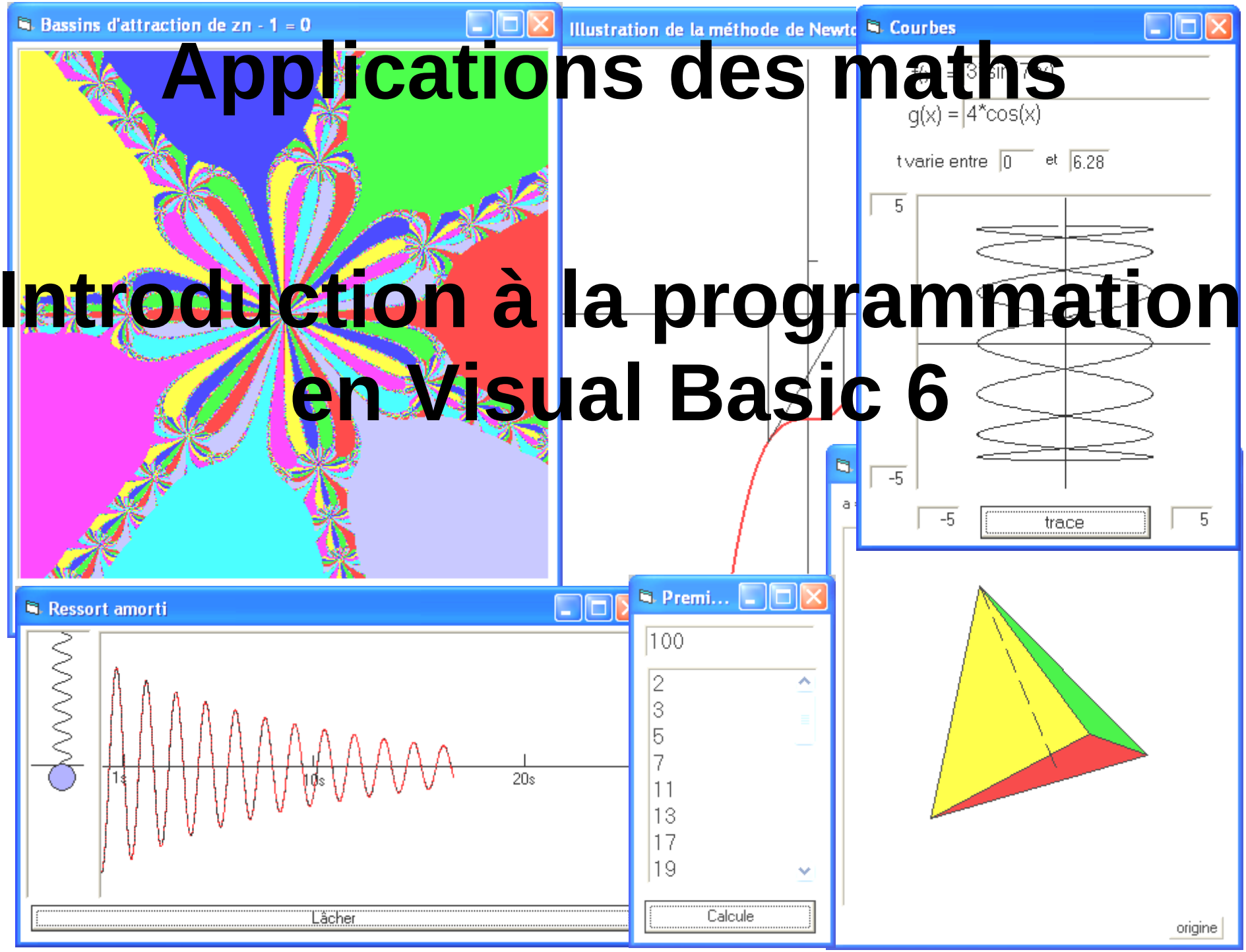
Applications des maths

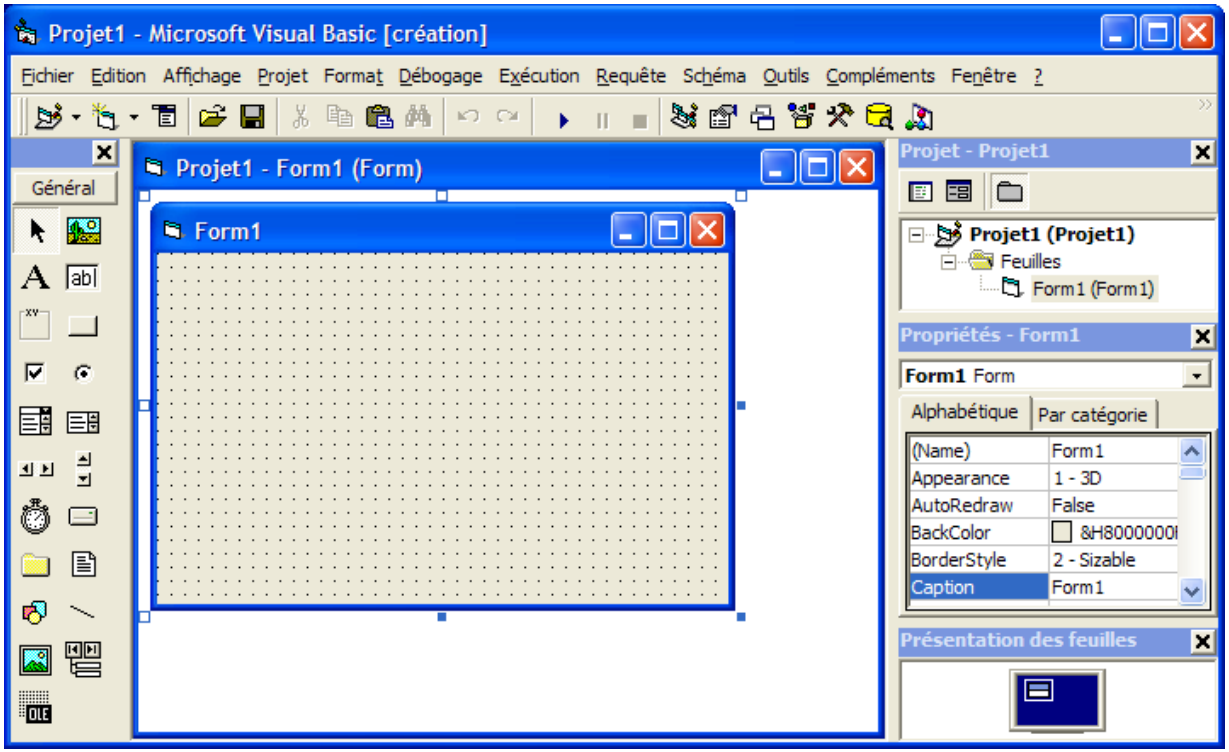
Introduction à la programmation en Visual Basic 6



Visual Basic permet de créer des programmes informatiques en plaçant à l'écran les objets qui devront apparaître durant l'exécution du programme et en écrivant le code Basic qui modifiera les données.
Les objets, appelés ***contrôles*** (étiquettes, zone de texte, boutons de commande, listes, zone d'image, etc.), sont dessinés sur une ***feuille*** (*Form*) qui présente la partie principale de l'écran.
La *feuille* et les *contrôles* possèdent des ***propriétés*** (nom, titre, couleur, etc.) qui définissent leur apparence. Ces propriétés sont définies lors de la création du contrôle mais peuvent varier au cours de l'exécution du programme.

Après avoir chargé Visual Basic, l'écran présente :

- la **barre de menus** qui contient le nom des menus à dérouler.
- La **barre d'outils** qui contient des raccourcis vers les éléments des menus fréquemment utilisés.
- La **feuille** qui est l'écran du programme à réaliser. Elle contiendra les contrôles.
- La **boîte à outils** qui contient les contrôles que l'on peut placer sur la feuille.
  Pour placer un contrôle sur la feuille, il suffit de le sélectionner avec la souris puis de fixer sa position et sa taille sur la feuille.
- La **fenêtre des propriétés** qui donne accès aux propriétés de la feuille et des contrôles dessinés.
  Pour modifier une propriété, il suffit de modifier sa valeur dans la colonne de droite.



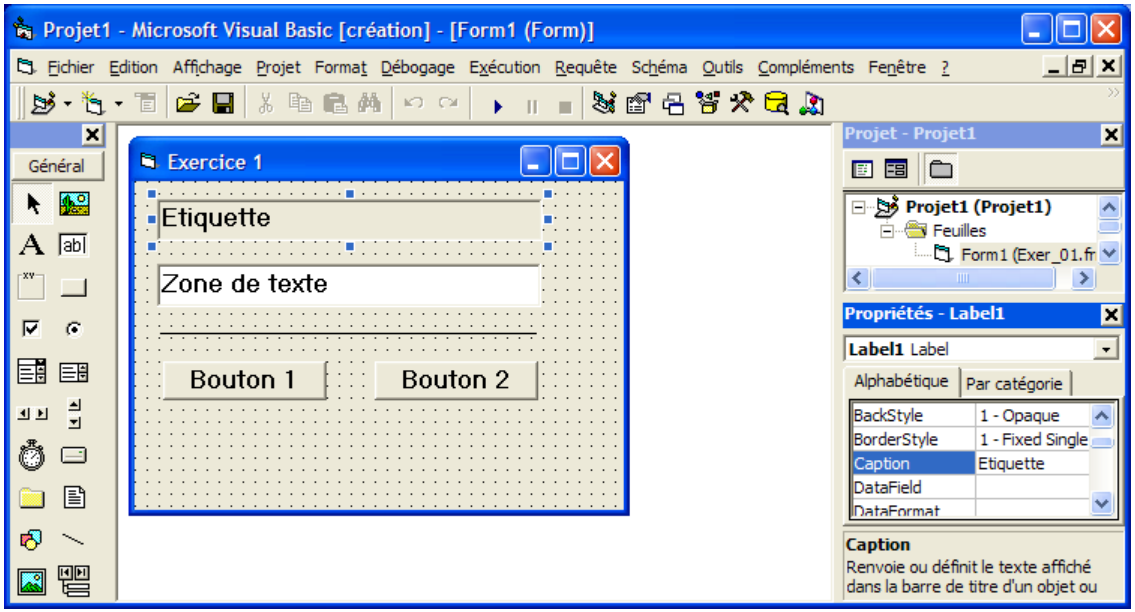
Si l'une ou l'autre de ces parties de l'écran n'est pas affichée, le menu *affichage* permet de la faire apparaître.

Exercice 1

Exécuter Visual Basic puis placer sur la feuille quelques contrôles. Modifier certaines propriétés de la feuille et des contrôles.

- La propriété *Caption* de la feuille détermine le texte affiché dans la barre de titre.
- La propriété *Caption* d'un contrôle ***étiquette*** (*Label*) définit le texte sur l'étiquette.
- La propriété *Text* d'un contrôle ***zone de texte*** (*TextBox*) définit le texte initialement affiché.
- La propriété *Caption* d'un contrôle ***bouton de commande*** (*CommandButton*) définit le texte affiché sur le bouton.

Exécuter le programme et observer les particularités des contrôles. Interrompre l'exécution en cliquant sur la croix puis enregistrer le projet sous le nom *EXER01*. Lors de l'enregistrement du projet, 2 fichiers au moins sont créés. Le premier (*EXER01.FRM*) contient les caractéristiques des objets de la feuille et le second (*EXER01.VBP*) une description générale du projet.



Exercice 2

Exécuter Visual Basic puis placer sur la feuille un contrôle ***zone d'image*** (*PictureBox)* ainsi qu'un contrôle étiquette.

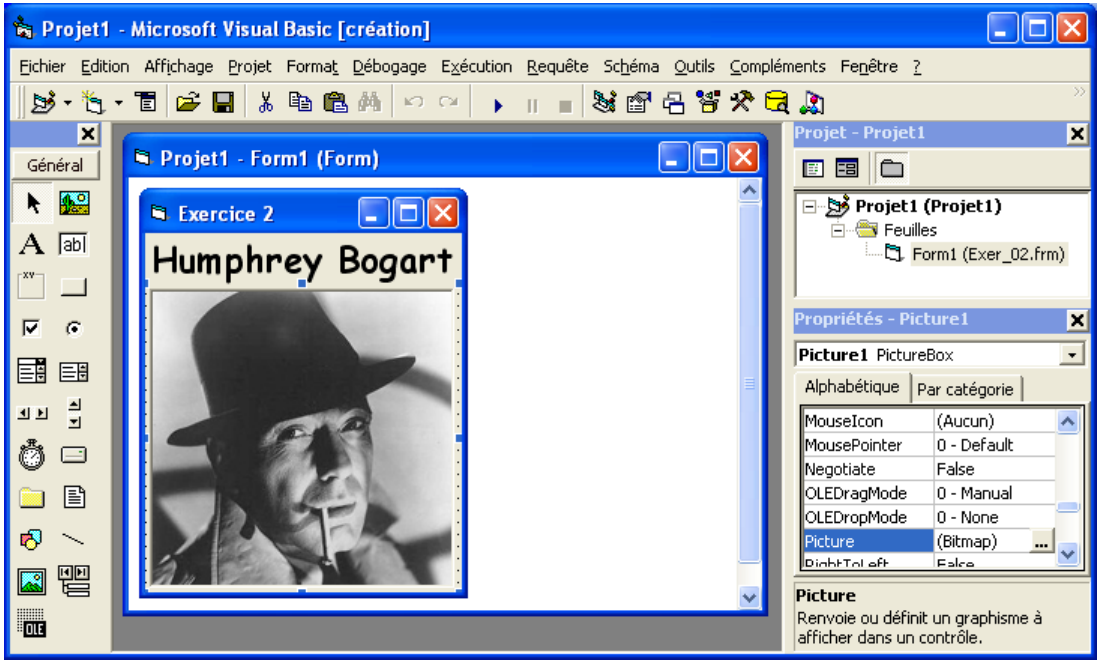
Un contrôle zone d'image permet d'afficher une image dont le nom est précisé par la propriété *picture*. Ce contrôle permettra également d'afficher des graphismes.

Un contrôle étiquette permet d'afficher des textes définis par la propriété *Caption*.

Modifier des propriétés de ces contrôles de sorte qu'ils affichent votre photo ainsi que votre nom.

Essayer de modifier d'autres propriétés et observer l'effet.

Enregistrer le projet sous le nom *EXER02*.



Chaque contrôle porte un nom qui lui est attribué lors de sa création. La première étiquette placée sur la feuille portera le nom ***Label1***, la seconde ***Label2*** et ainsi de suite. Le 1er bouton placé se nommera ***Command1***, le second ***Command2***; la première zone de texte se nommera ***Text1***, la seconde ***Text2*** …

A chaque propriété correspond une **variable** nommée *contrôle.propriété* où *contrôle* représente le nom du contrôle et *propriété* celui de la propriété. Par exemple, la variable correspondante à la propriété *Caption* d'un bouton nommé *Label1* s'appellera *Label1.Caption*. Pour modifier une propriété durant l'exécution du programme, il faut modifier la valeur de cette variable en incorporant une instruction Basic du style :

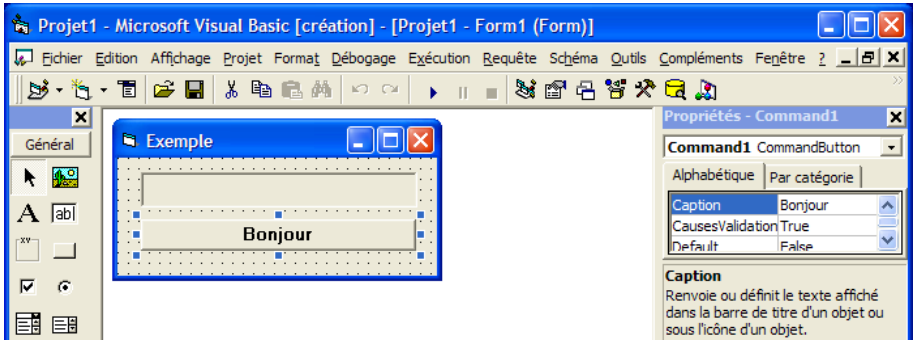
Let contrôle.propriété = valeur ou *valeur* représente la nouvelle valeur que doit prendre la propriété.

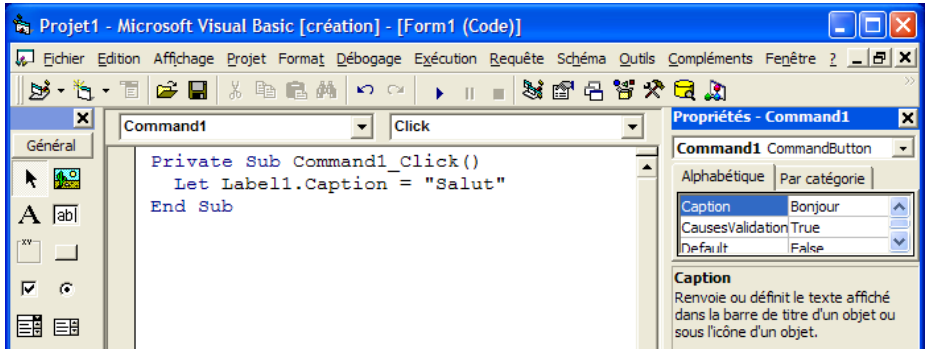
Par exemple, l'instruction basic *Let Label1.Caption="Salut"* permet d'affecter la valeur *Salut* à la propriété *Caption* du contrôle *Label1* donc d'afficher le mot *Salut* à l'emplacement de l'étiquette nommée *Label1*.

Les instructions basic qui forment un programme sont regroupées en **procédures**. Une procédure particulière est activée, durant l'exécution du programme, par un *clic* sur un bouton de commande. Pour créer cette procédure particulière, il faut *double-cliquer* sur le bouton de commande concerné; on accède ainsi à un éditeur qui permet d'écrire le programme.

L'exemple ci-contre illustre la création d'un petit programme qui, lors d'un clic sur le bouton intitulé *Bonjour*, affiche *Salut* sur une étiquette.

1) Dessiner l'étiquette et le bouton sur la feuille puis modifier leurs propriétés.
2) *Double-cliquer* sur le bouton *Command1* (*Bonjour*) pour accéder à l'éditeur de programme qui propose de compléter la procédure *Click()* du contrôle *Command1*.
3) Taper l'instruction Basic *Let Label1.Caption="Salut"* qui permet de modifier la propriété *caption* du contrôle *Label1*.

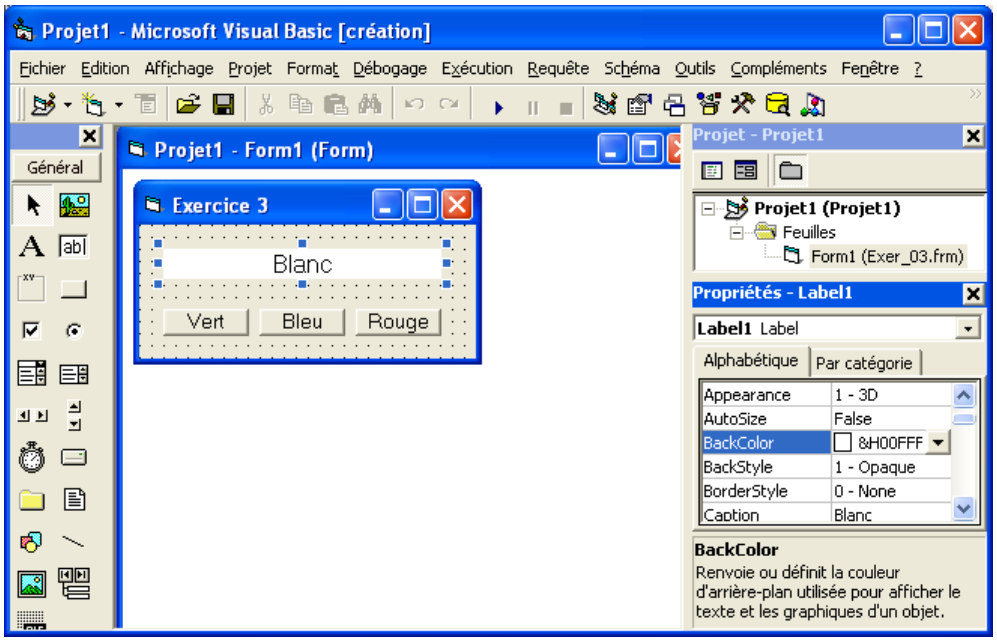




L'éditeur de programme peut aussi être appelé par l'option *Code* du menu *Affichage*. Deux menus permettent alors de choisir le contrôle et la procédure qui doit être créée.

Exercice 3

Créer un programme comprenant une étiquette et 3 boutons. Un clic sur l'un des boutons doit permettre de changer la couleur de fond et le contenu de l'étiquette.



Indications

1) Les instructions

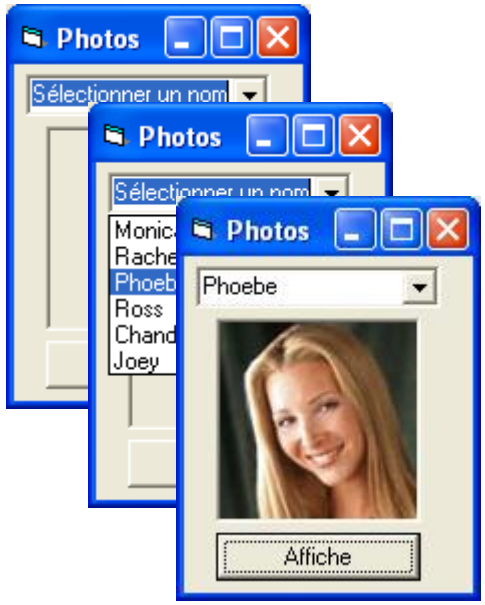
   Let Label1.BackColor=RGB(255,0,0)
   Let Label1.BackColor=RGB(0,255,0)
   Let Label1.BackColor=RGB(0,0,255)

   permettent de colorier le fond de l'étiquette *Label1* en rouge, en vert, en bleu.

2) La procédure *Command1_Click* sera exécutée lors d'un clic de l'utilisateur sur le bouton *Command1* alors que la procédure *Command2_Click* sera exécutée lors d'un clic de l'utilisateur sur le bouton *Command2* ...

Exercice 4

Créer un programme qui permette de sélectionner le nom d'un élève de la classe dans une ***zone de liste*** (*ComboBox*) puis, par un clic sur un bouton de commande, qui affiche dans une zone d'image, une photo de l'élève sélectionné.



Remarques :

1) La propriété *Text* d'une zone de liste contient le texte qui est affiché à l'écran.
2) La propriété *List* d'une liste contient les éléments qui apparaissent dans le menu.
3) La propriété *Picture* d'une zone d'image définit l'image à afficher.
4) L'instruction Basic *Let Picture1.Picture = LoadPicture("phoebe.jpg")* permet par exemple d'afficher, dans la zone d'image nommée *Picture1*, l'image *phoebe.jpg*.
5) L'instruction Basic *Let Picture1.Picture = LoadPicture(*Combo1.Text & *".jpg")* permet d'afficher, dans la zone d'image nommée *Picture1* l'image sélectionnée dans le menu.
6) Dans le programme à réaliser, l'instruction Basic à insérer dans la procédure *Click* du contrôle *Command1* est :

   Let Picture1.Picture = LoadPicture(Combo1.Text & ".jpg")

   Attention, l'image doit se trouver dans le même dossier que le programme.

Une ***variable*** est une zone de mémoire repérée par un nom et pouvant mémoriser une valeur. On connaît déjà les variables qui contiennent les propriétés des contrôles. En plus de ces variables dont le nom est de la forme *Contrôle.Propriété*, le programmeur peut utiliser d'autres variables dont il décide lui-même du nom. Pour assigner une valeur à une variable nommée par exemple *Var*, on utilise l'instruction *Let Var = valeur*.
La valeur peut être un nombre ou une chaîne de caractères (mot ou phrase) délimitée par des guillemets. A la place d'une valeur spécifiée, on peut mettre une formule de calcul. Dans ce cas, la valeur mémorisée est le résultat du calcul indiqué.
Pour écrire une formule de calcul numérique, on utilise les mêmes opérateurs arithmétiques qu'une calculatrice : +, -, *, /, ^ (élévation à une puissance) et les fonctions **Sqr( )** (racine carrée) et **Int( )** (partie entière).
Pour manipuler des chaînes de caractères, on utilise l'opérateur **&** qui colle deux mots, les fonctions **Left(***X*,*N***)** et **Right(***X*,*N***)** qui prennent *N* caractères à gauche ou à droite de la chaîne *X* et la fonction **LEN(***X***)** qui donne la longueur de la chaîne *X*.
Exemple :

```
Let Nom = "Dupont"
Let L = Len(Nom)
Let Affichage = Nom & " comporte " & L & "caractères"
Let Label1.Caption = Affichage
```

Les ***zones de texte*** (*TextBox*) sont des contrôles qui permettent d'introduire des données (nombres ou chaînes de caractères) dans un programme. Elles affichent un texte qui peut être modifié par l'utilisateur durant l'exécution du programme. La propriété *Text* contient le texte affiché et la variable *TEXT1.TEXT* permet donc de connaître ou de modifier le contenu d'une zone de texte appelée *TEXT1*.

Les ***étiquettes*** (*Label*) ont des contrôles qui permettent d'afficher du texte ou le contenu d'une variable. Pour afficher le contenu d'une variable, il faut donner la valeur de la variable à la propriété *Caption* de l'étiquette.

Exemple

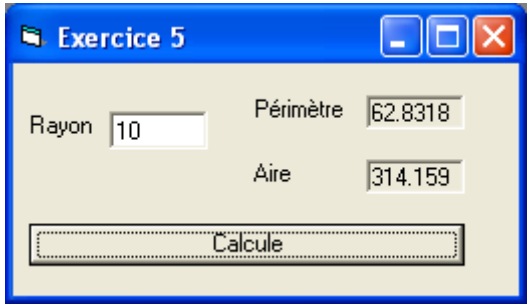
Cet exemple consiste en un programme de conversion des francs suisses en euros. La feuille contient une zone de texte pour saisir une somme en francs suisses, une étiquette pour afficher la valeur en euros, deux étiquettes pour les libellés CHF et EUR ainsi qu'un bouton de commande pour afficher la somme et un autre pour quitter le programme.
La procédure activée lors d'un *clic* sur le bouton nommé *Command1* (*Affiche*) est la suivante :

```
Private Sub Command1_Click()        début de la procédure Click
  Let Chf = Text1.Text              assigne à la variable Chf le contenu de la zone de texte Text1
  Let Euro = Chf / 1.67             assigne à la variable Euro le résultat de l'opération
  Let Label1.Caption = Euro         affiche le contenu de la variable Euro sur l'étiquette Label1
End Sub                             fin de la procédure Click
```

La procédure liée au bouton de commande *Fin* ne contient que l'instruction basic *End* qui termine le programme.

Exercice 5

Créer un programme qui permette de donner le rayon d'un cercle puis, par un clic sur un bouton de commande, affiche son périmètre et son aire.



Déroulement du programme :

- Une zone de texte permet à l'utilisateur de donner le rayon.
- Le rayon donné dans la zone de texte sera, suite à un clic sur *calcule*, enregistré dans une variable nommée *Rayon*.
- Le périmètre est calculé et enregistré dans variable nommée *Perimetre*.
- L'aire est calculée et enregistrée dans une variable nommée *Aire*.
- Le périmètre et l'aire sont affichés sur les étiquettes correspondantes.
  Pour afficher le périmètre, il suffit d'assigner à la variable *Label1.Caption* la valeur de la variable *Perimetre*.
  L'instruction *Let Label1.Caption = Perimetre* permet cette assignation.

Exercice 6

Créer un programme qui permette de donner le rayon et la hauteur d'un cylindre puis, par un clic sur un bouton de commande, affiche son volume.

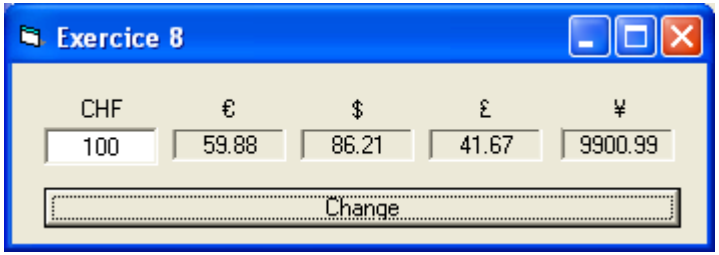
Exercice 7

Créer un programme qui permette de donner le nom de l'utilisateur puis, par un clic sur un bouton de commande affiche, sur une étiquette, *Salut* suivi du nom donné.

Exercice 8

Créer un programme qui permette de donner une somme en francs suisses puis qui affiche l'équivalent dans différentes monnaies étrangères.
Pour améliorer l'affichage des montants, on pourra recourir à une commande de formatage comme ***Format***(*Var*,"0.00") qui force l'écriture de la variable *Var* avec au moins un chiffre avant la virgule et de 2 après.

Pour les cours du change, voir, par exemple, "http://www.swissinfo.org"



Le ***type*** d'une variable est le genre de contenu qu'elle est susceptible de recueillir. Les types les plus utilisés pour les variables sont les entiers, les réels et les chaînes de caractères. Par défaut Visual Basic considère que chaque variable qui n'a pas été *déclarée* est de type *variable* (*variant)* c'est à dire qu'elle peut changer de type lors de l'exécution du programme.
Exemple de types variables :

Let a = 12	*a contient un nombre entier*
Let b = a & a	*les contenus de a et de b sont ici considérés comme des chaînes de caractères*
*Let c = 2 * b* $\Rightarrow$ *c = 2424*	*les contenus de b et de c sont ici considérés comme des entiers*

Pour rendre le code plus clair et pour éviter les erreurs, il est préférable et parfois indispensable de préciser quel type de donnée sera enregistré dans les variables employées. Cette ***déclaration*** se fait dans la partie ***Déclaration*** située au début du programme à l'aide de l'instruction ***Dim*** dont la syntaxe est la suivante :

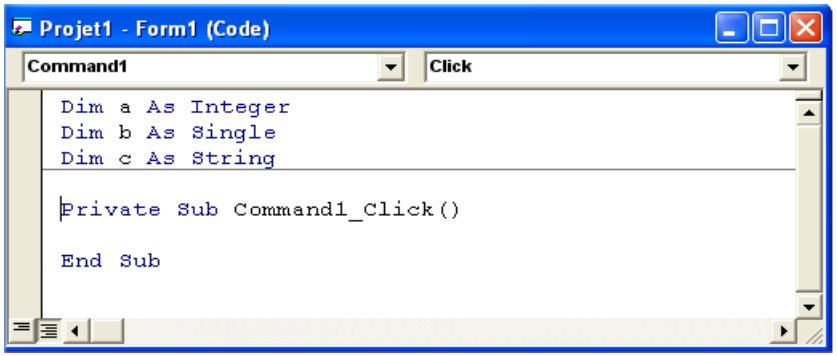
Dim Nom de variable As Type

Les principaux types sont :

Integer	nombre entier compris entre -32768 et 32767
Long	nombre entier compris entre -2 147 483 648 et 2 147 483 647
Single	nombre réel à virgule flottante en simple précision
Double	nombre réel à virgule flottante en double précision
String	chaîne de caractères

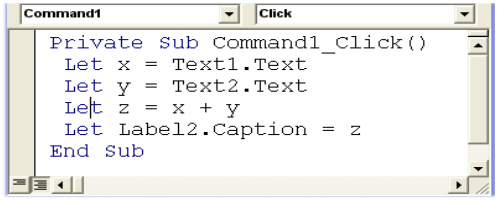
Exemple :

Dim a As Integer	*déclare la variable a comme entière*
Dim b As Single	*déclare la variable b comme réelle*
Dim c As String	*déclare la variable c comme chaîne de caractères*



Remarques :

1. Il est indispensable de déclarer une variable si sa valeur doit être utilisée dans plusieurs procédures du projet ou dans plusieurs exécutions de la même procédure car, au début de l'exécution d'une procédure, **toutes les variables sont automatiquement mises à zéro**.
2. En exécutant le programme ci-contre, avec deux chiffres dans les zones de texte, on constate que ces chiffres sont considérés comme des textes et que le résultat affiché n'est pas celui que l'on attend. Pour corriger ce défaut, il faut soit déclarer les variables *x* et *y*, soit compléter les affectations des zones de texte par : "Let x = Val(Text1.Text)", qui transforme le contenu Text1.Text en une valeur numérique.



Exercice 9

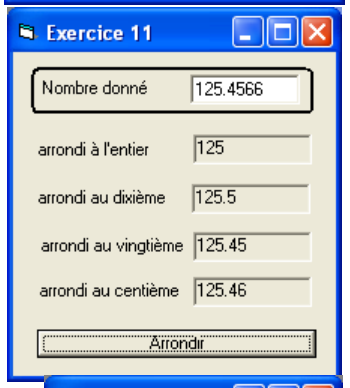
Créer un programme qui demande la longueur des côtés d'un rectangle puis qui affiche le périmètre et l'aire de ce rectangle.

Exercice 10

Créer un programme qui demande un verbe régulier en "er" puis qui affiche, à la première personne du singulier, le présent, l'imparfait, le futur et le passé simple de ce verbe.

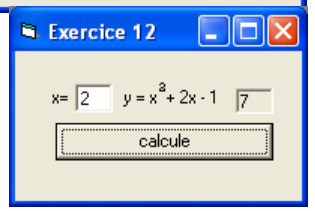
Exercice 11

Créer un programme qui demande un nombre, puis lors d'un clic sur un bouton de commande, calcule puis affiche le nombre arrondi à l'entier, au dixième, au vingtième et au centième. Soigner la présentation…
Remarque : *INT(X + 0.5)* produit un entier en arrondissant *X* à l'entier le plus proche.



Exercice 12

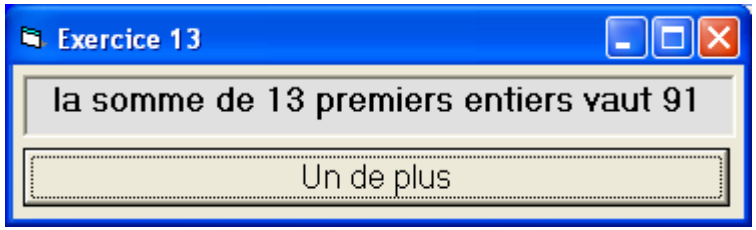
Créer un programme qui calcule l'image de x par la fonction donnée par $f(x) = x^2 + 2x - 1$.
La valeur de x doit pouvoir être donnée par l'utilisateur puis, après un clic sur un bouton, l'image de x par f doit être affichée.



Exercice 13

a) Créer un programme qui affiche le nombre de clics que la souris effectue sur un bouton de commande
Remarque : L'instruction *Let N=N+1* permet de modifier la variable *N* en additionnant 1 à sa valeur.

b) Créer un programme qui calcule la somme des premiers nombres entiers.
Remarque : L'instruction *Let S=S+N* permet d'ajouter *N* à la variable *S*.



On a vu que, lors de l'exécution d'un programme, un *clic* sur un bouton de commande nommé par exemple *Command1* a pour effet d'exécuter une procédure appelée *command1_Click( ).*
Le *clic* de la souris est un ***événement*** et la procédure *Command1_Click( )* est une ***procédure événementielle***, elle s'exécute lorsque l'événement a lieu.

D'autres événements peuvent provoquer l'exécution d'une procédure événementielle, par exemple le déplacement de la souris sur un contrôle nommé *Command1* a pour effet d'exécuter une procédure appelée *Command1_MouseMove(. . . .).*

Outre les boutons de commandes, les autres contrôles possèdent également des procédures événementielles, par exemple :

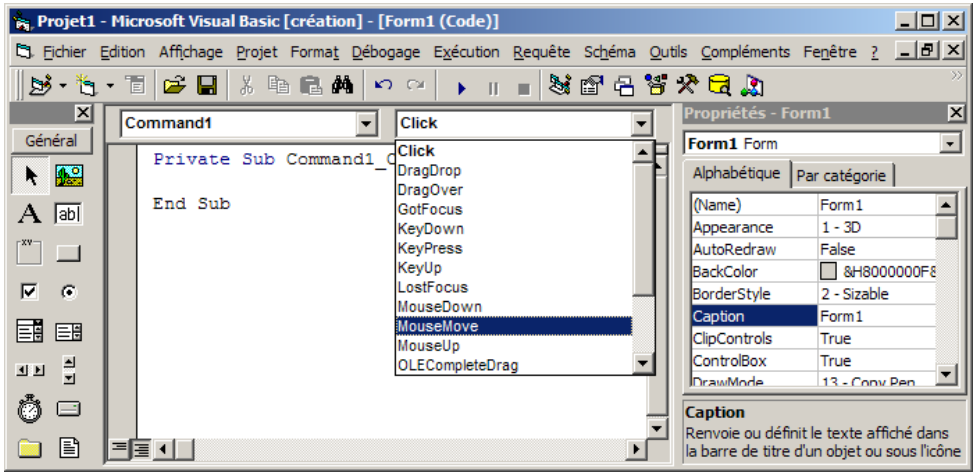
- chaque contrôle possède une procédure événementielle nommée *Click* qui est exécutée lors d'un clic de la souris sur le contrôle. La procédure *Label1_Click( )* est par exemple exécutée lors d'un clic de la souris sur une étiquette nommée *Label1*.
- un contrôle *zone de texte* possède une procédure événementielle nommée *Change* qui est exécutée lors de chaque modification du contenu de la zone de texte.

  La procédure ci-dessous permet par exemple d'afficher sur une étiquette nommée *Label1* les caractères tapés dans une zone de texte nommée *Text1*.

```
Private Sub Text1_Change()
  Label1.Caption = Text1.Text
End Sub
```

- un contrôle *zone de texte* possède une procédure événementielle nommées *KeyPress* qui est exécutée lorsqu'une touche du clavier est pressée. Cette procédure *KeyPress* attribue encore à une variable nommée *KeyAscii*, le code ascii de la touche préssée.
- la feuille (*Form1*) possède une procédure événementielle nommée *Form1_Load* qui est exécutée au chargement de la feuille donc lorsque le programme démarre. Cette procédure est utile pour attribuer des valeurs par défaut aux variables.

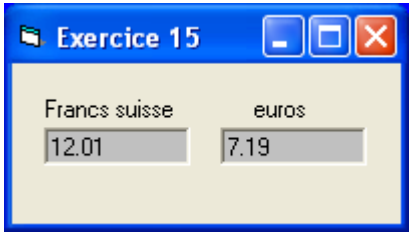
Pour écrire une procédure événementielle, il faut la sélectionner ainsi que le contrôle correspondant dans les menus déroulants de l'éditeur de programme.



Exercice 14

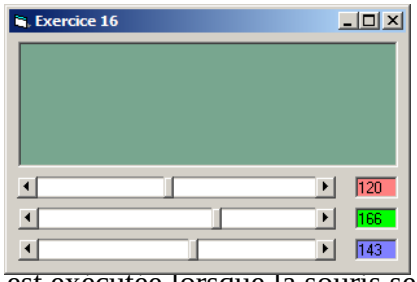
A chaque caractère correspond un code, le code ascii de ce caractère. Par exemple le code ascii du caractère "A" est 65.
Écrire un programme qui permet de taper du texte dans un contrôle zone de texte, et affiche (sur une étiquette) le code ascii de chaque caractère pressé.
La procédure d'événement *Text1_KeyPress(KeyAscii)* d'un contrôle zone de texte *Text1* est exécutée lors de chaque pression sur une touche du clavier. La variable KeyAscii prend la valeur du code ascii correspondant à la touche pressée.
Remarque : la fonction *Chr(N)* rend le caractère dont le code ascii est *contenu* dans la variable N et la fonction *Asc(C)* rend le code ascii du caractère *contenu* dans la variable C.

Exercice 15



Écrire un programme qui permette de donner une somme en francs suisses et qui comporte 2 zones de texte. Si l'utilisateur tape une somme en francs suisse dans la zone de texte de gauche, le programme affiche la somme correspondante en euro dans celle de droite et si l'utilisateur tape une somme en euro dans la zone de texte de droite, le programme affiche la somme correspondante en francs suisses dans celle de gauche. (les sommes sont affichée avec 2 décimales).
Remarque : l'utilisation d'un contrôle zone de texte pour afficher des résultat est a éviter, pour s'en convaincre, observer le déroulement du programme avec un change fixé à 1.67 CHF pour 1 €.

Exercice 16



Écrire un programme qui permette d'afficher une couleur dont l'utilisateur précise ses composantes de rouge de vert et de bleu dans 3 barres de défilement horizontales.
Consulter l'*aide en ligne* de *Visual Basic* pour découvrir les contrôles barres de défilement horizontal.

Exercice 17

Écrire un programme qui permette de déplacer une image avec la souris.

- La procédure événementielle *Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)* est exécutée lorsque la souris se déplace sur la feuille. Les coordonnées du pointeur de la souris sont enregistrées dans les variables *X* et *Y*. L'origine est le coin supérieur gauche de la feuille.
- La procédure événementielle *Picture1_MouseMove(...)* est exécutée lorsque la souris se déplace sur le la zone d'image *Picture1*. Les coordonnées du pointeur de la souris sont enregistrées dans les variables *X* et *Y*. L'origine est le coin supérieur gauche de la zone d'image.
- Pour déplacer la zone d'image *Picture1*, il suffit de modifier ses propriétés *Picture1.Left* et *Picture1.Top*.

Pour exécuter certaines instructions lorsqu'une condition est satisfaite et d'autres instructions si cette condition n'est pas satisfaite on utilise une structure de sélection :

If condition **Then**
partie du programme étant exécutée
si la condition est remplie
End If

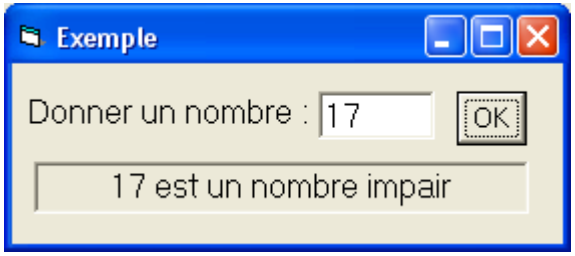
ou

If condition **Then**
partie du programme étant exécutée
si la condition est remplie
Else
partie du programme étant exécutée
si la condition n'est pas est remplie
End If

Les conditions sont des égalités (=) ou inégalités (<, >, <=, >=, <>) entre des variables ou des expressions numériques ou alphabétiques.

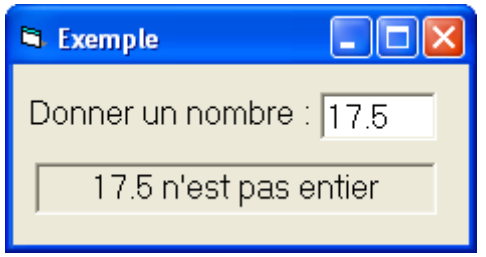
Exemple
détection de la parité d'un nombre entier

```
Private Sub Command1_Click()
    Let n = Val(Text1.Text )
    If Int(n/2) = n/2 Then
        Let Label1.Caption = n & " est pair"
    Else
        Let Label1.Caption = n & " est impair"
    End If
End Sub
```



Remarque
Il peut être utile de pouvoir déterminer si une valeur donnée dans une zone de texte est un nombre ou non. La fonction **IsNumeric(x)** donne vraie si x est un nombre et faux sinon.

Exemple




```
Dim n as Single
Private Sub Text1_Change()
    If IsNumeric(Text1.Text) Then
        Let n = Val(Text1.Text)
        If n = Int(n)  Then
            If Int(n / 2) = n / 2 Then
                Let Label2.Caption = n & " est un nombre pair"
            Else
                Let Label2.Caption = n & " est un nombre impair"
            End If
        Else
            Let Label2.Caption = n & " n'est pas  entier !"
        End If
    Else
        Let n = Text1.Text
        Let Label2.Caption = n & " n'est pas un nombre !"
    End If
End Sub
```

Exercice 18

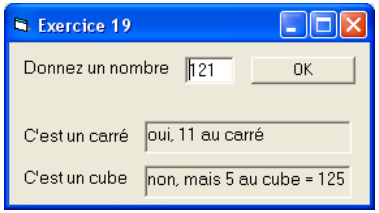
Écrire un programme qui demande un nom puis qui affiche « Bonjour » si le nom donné est le vôtre ou « vous n'avez pas accès à cet ordinateur » si c'est un autre nom.

Exercice 19

Écrire un programme qui détermine si un nombre donné est un carré parfait (1, 4, 9, 16, …) ou un cube parfait (1, 8, 27, …).

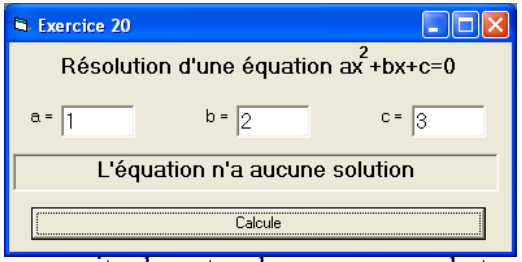
Remarque

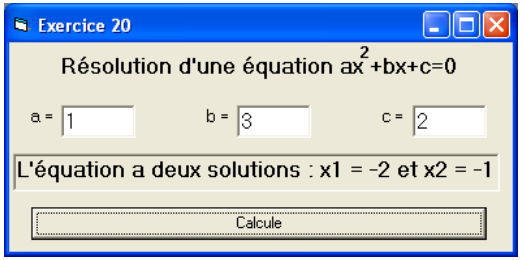
- **Sqr(*x*)** calcule une estimation de la racine carrée de *x*
- ***x*^(1/3)** calcule une estimation de la racine cubique de *x*.



Exercice 20

Programmer la résolution d'une équation du $2^{ème}$ degré.

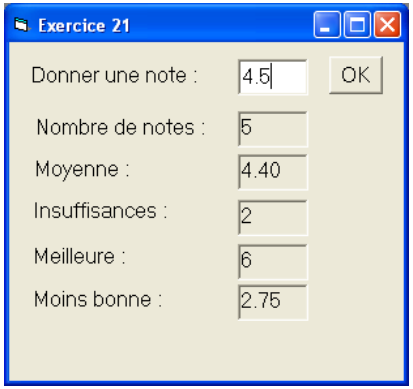




Exercice 21

a) Écrire un programme qui permette à l'utilisateur d'introduire une suite de notes dans une zone de texte et qui affiche le nombre de notes données ainsi que leur moyenne.
Seules les notes comprises entre 1 et 6 et données au quart de point doivent être acceptées.

b) Modifier le programme de sorte qu'il affiche encore la meilleure et la moins bonne des notes données ainsi que le nombre de notes inférieures à 4.

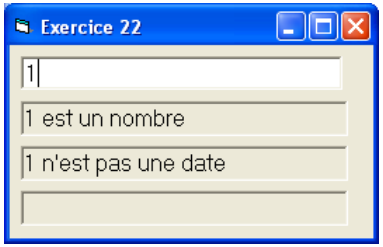


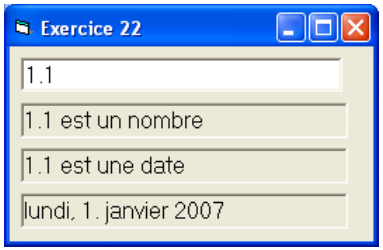
Exercice 22

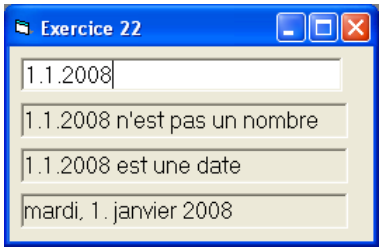
Écrire un programme qui détermine si une donnée tapée par l'utilisateur dans une zone de texte et un nombre ou une date.

Remarques :

- la fonction **IsDate(x)** permet de déterminer si x contient une date.
- une date est donnée sous la forme 01.01.2000 ou sous la forme 1 janvier 2000
- La fonction **Format(x, "Long Date**") permet d'afficher la date x en *toutes* lettres.



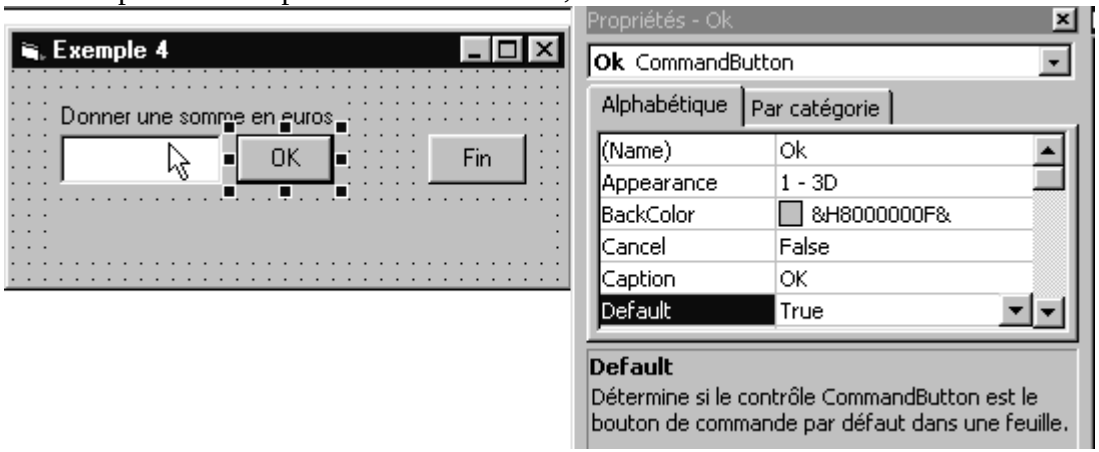




Si plusieurs contrôles pouvant réagir aux actions de l'utilisateur (zones de texte, boutons, …) sont sur une feuille, un seul de ces contrôles est actif à la fois. On dit que le contrôle actif possède le *focus*.
Pour sélectionner un contrôle (lui donner le *focus*), il faut, soit *cliquer* sur ce contrôle, soit presser sur la touche de tabulation jusqu'à ce que le contrôle soit sélectionné; la touche de tabulation permettant de passer le *focus* d'un contrôle au suivant.
L'ordre dans lequel les contrôles sont activés par la touche de tabulation est défini par la propriété ***TabIndex*** des contrôles. La valeur 0 donnée à cette propriété indique que le contrôle est le premier sélectionné, les valeurs 1, 2, 3 … sont donnés aux contrôles suivants.
Il est possible de préciser qu'un contrôle ne fait pas partie de la liste des contrôles à sélectionner par pression de la touche de tabulation, il faut pour cela donner la valeur *False* à la propriété ***TabStop*** de ce contrôle.

Si un *bouton de commande* possède le *focus*, une pression sur la touche "Entrée" est équivalente à un clic sur ce bouton et entraîne donc l'exécution de la procédure *Clic* associé à ce contrôle.

Si une *zone de texte* possède le *focus* et s'il existe, sur la feuille, un *bouton de commande* dont la propriété ***default*** est égale à la valeur *True*, alors une pression sur la touche "Entrée" est équivalente à un *clic* sur ce bouton, et entraîne donc l'exécution de la procédure *Click* associée au *bouton de commande*.
Le recours à "*default=True*" est utilisé pour valider par la touche "Entrée", des données introduites dans une zone de texte.

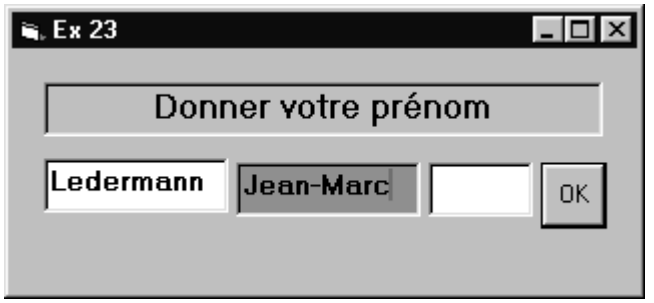


Le bouton OK étant le bouton par défaut, une pression sur "Entrée" est équivalent à un *clic* de la souris sur ce bouton.

Chaque contrôle pouvant avoir le *focus* possède 2 procédures événementielles, l'une ***gotfocus*** est exécutée chaque fois que le contrôle prend le *focus*, et l'autre, ***lostfocus***, est exécutée chaque fois que le contrôle perd le *focus*.

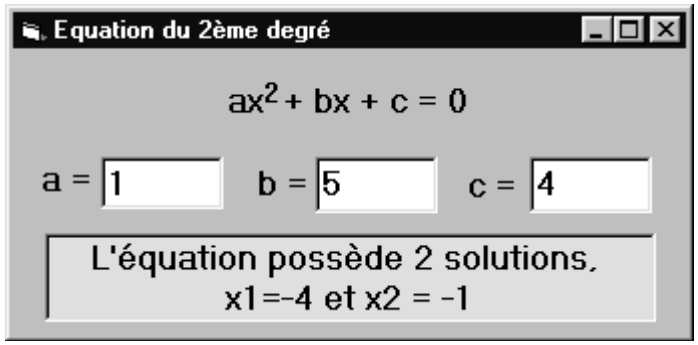
Remarque : L'instruction *Text1.SetFocus* permet de donner le focus à la zone de texte *Text1*.

Exercice 23



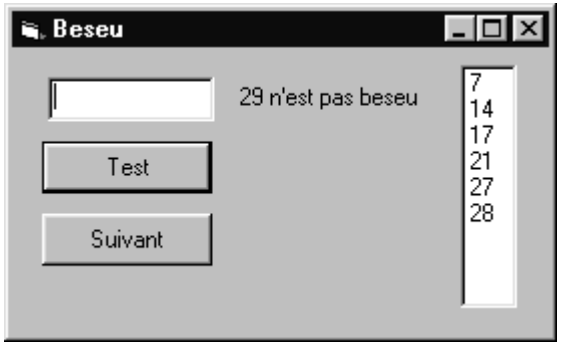
Écrire un programme qui présente 3 zones de texte. Le fond de la zone de texte en édition est rouge alors que le fond des autres zones de texte est blanc. Un texte, affiché sur une étiquette, dépend de la zone de texte en édition. Un bouton de commande permet de quitter le programme. Une pression sur la touche "Entrée" est équivalente à un *clic* sur le bouton de commande.

Exercice 24



Modifier l'exercice de résolution des équations du 2ème degré de sorte que l'on puisse donner (dans l'ordre) les 3 valeurs des coefficients, en passant d'une zone de texte à l'autre en pressant sur la touche "*Tab"*.
Les solutions doivent s'afficher au moment ou l'on presse sur la touche "*Entrée*" (procédure *Click* d'un bouton de commande par défaut placé à l'extérieur de la fenêtre d'affichage pour le rendre invisible.
Remarque
Pour valider une entrée donnée dans une zone de texte et passer à la zone de texte suivante il faut donc presser pur la touche "*Tab*" et non sur la touche "*Entrée*"

Exercice 25



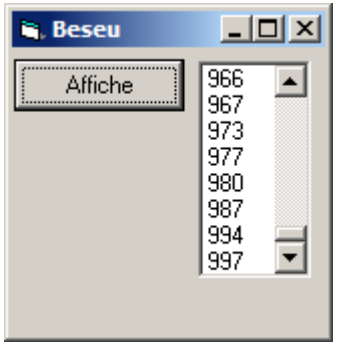
Un nombre entier positif est dit "beseu" s'il est divisible par 7 ou s'il se termine par 7.
Les premiers nombres "beseu" sont ainsi 7, 14, 17, 21, 27, 28, …
Écrire un programme qui permette à l'utilisateur de faire tester par l'ordinateur si un nombre donné dans une zone de texte est un nombre "beseu". Une pression sur la touche "Entrée" permet de tester le nombre donné.
Indication : un nombre entier *n* se termine par 7 si le nombre *n* – 7 est divisible par 10.
Compléter le programme en ajoutant un bouton "suivant" qui passe au nombre suivant et le teste. Une pression sur la touche "Entrée" sans avoir introduit de nombre dans la zone de texte permet de passer au suivant.
Compléter encore le programme de sorte qu'il affiche les nombres "beseu" trouvés dans une ***liste***.

Une ***liste*** est un contrôle dans lequel viendront s'inscrire des éléments, l'un en dessous de l'autre.
La commande **List1.AddItem(e)** permet d'ajouter un élément ***e*** dans la liste ***List1***.
La commande **List1.Clear** supprime tous les éléments de la liste ***List1***.

Pour exécuter automatiquement plusieurs fois une partie d'un programme on utilise une *structure d'itération*.

Exemple 1

Le programme suivant affiche dans une liste, après pression sur un bouton de commande, tous les nombres *Beseu* compris entre 1 et 1000

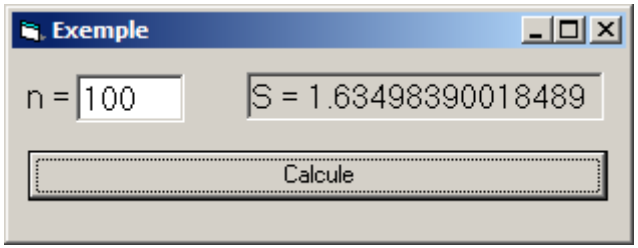



```
Private Sub Command1_Click()
For  n = 1  To  1000
    If  n / 7 = Int(n / 7)  Or  (n - 7) / 10 = Int((n - 7) / 10) Then
        List1.AddItem(n)
    End  If
Next  n
End  Sub
```

Toutes les instructions comprises entre "*For n=1 to 1000*" et "*Next n*" sont répétées 1000 fois.
La variable *n* est le *compteur de boucle* et prend successivement les valeurs 1, 2 , 3, …, 1000.

Exemple 2

Le programme calcule la somme $S = 1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\frac{1}{5^2}+\cdots+\frac{1}{n^2}$, ou *n* est donné par l'utilisateur.




```
Dim  s As  Single
Private Sub Command1_Click()
If  IsNumeric(Text1.Text)  Then
    Let  n = Text1.Text
    For  i = 1 To n
        Let  s = s + 1 / i ^ 2
    Next  i
    Let  Label2.Caption = "S = " & s
End  If
End  Sub
```

Il est possible de préciser le *pas* en utilisant l'instruction *Step :*
Par exemple l'instruction *For i = 1 To 100 Step 2* crée une boucle qui sera répétée 50 fois, *n* prenant successivement les valeurs 1, 3, 5, …, 99.

Exercice 26

Écrire un programme qui affiche, dans une liste, tous les diviseurs d'un nombre donné dans une zone de texte.
Compléter le programme de sorte qu'il affiche encore le nombre de diviseurs et précise si le nombre donné est premier ou non.
Compléter encore le programme de sorte qu'il affiche la somme des diviseurs et précise si le nombre donné est parfait ou non.
Remarque : un nombre parfait est égal à la moitié de la somme de ses diviseurs

Exercice 27

Écrire un programme qui affiche dans une liste tous les nombres premiers inférieurs à un nombre donné.

Exercice 28

Écrire un programme qui simule 10'000 lancers d'une pièce de monnaie puis qui affiche le nombre de *pile* et le nombre de *face*
Remarque : la fonction ***Rnd*** rend un nombre aléatoire compris entre 0 et 1.

Exercice 29

Deux nombres $n1$ et $n2$ sont des *nombres amis* si $n2$ est égal à la somme des diviseurs de $n1$ (diviseurs inférieurs à $n1$) et que $n1$ est égale à la somme des diviseurs de $n2$ (diviseurs inférieurs à $n2$).
Les nombres 220 et 284 sont, par exemple, des nombres amis car :
Les diviseurs de 220 sont : 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110 et 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284
Les diviseurs de 284 sont : 1, 2, 4, 71, 142 et 1 + 2 + 4 + 71 + 142 = 220

Écrire un programme qui affiche tous les couples de nombres amis inférieurs à 10'000.
Marche à suivre :
Pour chaque nombre n_1 compris entre 1 et 10'000
    Calculer la somme n_2 des diviseurs de n_1
    Calculer la somme s des diviseurs de n_2
    Si $s = n_1$ alors les nombres n_1 et n_2 sont des nombres amis.

Exercice 30

Écrire un programme qui affiche les triplets de Pythagore inférieurs à 100.
Exemple de triplet de Pythagore : (3;4;5) car $3^2 + 4^2 = 5^2$

Alors que la structure d'itération "**For …. Next"** permet de créer une boucle parcourue un nombre connu de fois, la structure d'itération "**Do … Loop Until …"** permet de créer une boucle qui sera parcourue **jusqu'à ce qu'une condition soit remplie.**

Exemple 1

Le programme ci-dessous calcule la somme $S=\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots$ jusqu'à ce qu'elle atteigne la valeur 100 puis affiche le nombre de terme nécessaire pour atteindre cette valeur.

```
Private Sub Command1_Click()
    Let s = 0
    Let i = 0
    Do
        Let i = i + 1
        Let s = s + 1 / i
    Loop Until s >= 10
    Let Label1.Caption = i
End Sub
```

Exemple 2

Le programme ci-dessous simule le lancer de 2 dés jusqu'à ce que les dés affichent chacun la face 6. Il indique alors le nombre de lancers nécessaires.

```
Private Sub Command1_Click()
    Let n = 0
    Do
        Let d1=Int(6*Rnd)+1
        Let d2=Int(6*Rnd)+1
        Let n=n+1
    Loop Until d1=6 and d2=6
    Let Label1.Caption = n
End Sub
```

Remarque :

La boucle "For …. Next" peut être réalisée par une boucle "Do … Loop Until …"

```
For I=1 to 100
    …
    …
    …
    …
Next I
```

*I est le **compteur** de la boucle*

Peut être remplacé par

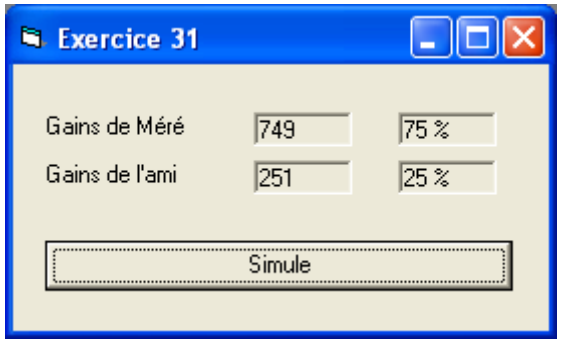
```
Do
    Let I=I+1
    …
    …
    …
Loop Until I=100
```

*I est le **compteur** de la boucle*

Exercice 31

L'étude des probabilités a débuté au 17ème siècle par un problème posé par le Chevalier Méré à Blaise Pascal.
Le Chevalier Méré avait l'habitude de jouer avec un ami. Le jeu consistait à miser sur l'issue d'un jet de dés. Méré parie sur le 6, son ami sur le 4. Le premier qui totalise 3 points gagne.
Une partie de dés fut interrompue alors que Méré avait 2 points et son ami 1 point. La question posée à Pascal est de savoir comment rétablir équitablement la mise et revient à calculer les probabilités de gagner de chacun des 2 joueurs,

a) Écrire un programme qui simule des fins de parties
b) Écrire un programme qui simule 1000 fins de parties et affiche le nombre de parties gagnées par Méré et par son ami en nombres et en pour-cent.



Exercice 32

Combien faut-il, en moyenne, lancers un dé, jusqu'à ce qu'il présente deux fois consécutivement la face 6.
Pour répondre à cette question (on doit trouver 42), écrire un programme qui simule 10'000 suites de lancers de dés…

Exercice 33

On appelle **série**, une somme (infinie) de termes.

Exemples : $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{k}+\cdots$ est appelée série harmonique. Cette série **diverge** (elle tend vers l'infini)

$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-\frac{1}{6}\pm\cdots$ est appelée série harmonique alternée.. Cette série **converge** (elle tend vers un nombre fini)

a) Écrire un programme qui calcule une estimation de la somme de la série $1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\cdots+\frac{1}{k^2}+\cdots$

Indication : effectuer les sommes des termes jusqu'à ce que le terme $\frac{1}{k^2}$ soit très petit …

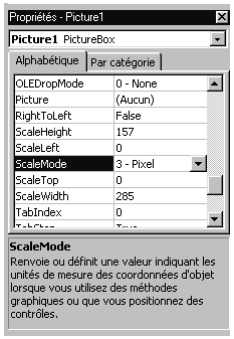
b) Écrire un programme qui calcule une estimation de la somme de la série $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-\frac{1}{6}\pm\cdots$

c) Écrire un programme qui calcule une estimation de la somme de la série $1+\frac{1}{1\cdot 2}+\frac{1}{1\cdot 2\cdot 3}+\frac{1}{1\cdot 2\cdot 3\cdot 4}+\frac{1}{1\cdot 2\cdot 3\cdot 4\cdot 5}+\cdots+\frac{1}{k!}+\cdots$

$n!=1\cdot 2\cdot 3\cdot 4\cdot\ \ \cdot n$ est appelée n factoriel.

d) Écrire un programme qui calcule une estimation de la somme de la série $8\cdot(\frac{1}{1\cdot 3}+\frac{1}{5\cdot 7}+\frac{1}{9\cdot 11}+\frac{1}{13\cdot 15}+\cdots)$

Pour réaliser des graphiques sur une feuille il faut préalablement définir la surface qui devra contenir les dessins. Cette surface est définie par un contrôle PictureBox  (zone de d'image).



Une zone d'image possède un système de coordonnées dont l'origine est située au coin supérieur gauche de la zone.
Il est nécessaire d'attribuer la valeur *Pixel* aux propriétés ***ScaleMode*** de la zone d'image et de la feuille (*form1*) pour que l'unité corresponde à un point graphique.
Les propriétés ***Picture1.width*** et ***Picture1.heigth*** décrivent la largeur et la hauteur de la zone d'image *Picture1*.

- L'instruction ***Picture1.Pset(x,y)*** dessine le point de coordonnées (*x*;*y*) sur la zone d'image *Picture1*.
- L'instruction ***Picture1.Circle(x,y),r*** dessine, sur la zone d'image *Picture1*, un cercle de centre (*x*;*y*) et de rayon *r*.
- L'instruction ***Picture1.Line(x1,y1) – (x2,y2)*** trace, sur la zone d'image *Picture1*, un segment d'extrémités (*x*1;*y*1) et (*x*2;*y*2).
- L'instruction ***Picture1.Line(x1,y1) – (x2,y2) , , B*** trace, sur la zone d'image *Picture1*, un rectangle de sommets opposés (*x*1;*y*1) et (*x*2;*y*2).
- L'instruction ***Picture1.Cls*** efface le contenu de la zone d'image *Picture1*.

Si une instruction de dessin est suivie d'une virgule puis de l'instruction ***RGB(r,g,b)***, alors les traits ont la couleur précisée par la proportion de rouge (*r*), la proportion de vert (*g*) et la proportion de bleu (*b*). (*r*, *g* et *b* peuvent prendre des valeurs comprises entre 0 et 255).
Pour définir la couleur d'un tracé il est également possible de modifier la valeur de la propriété ***ForeColor*** (***Let Picture1.ForeColor = RGB(r,g,b)***).
La largeur des traits est définie par la propriété ***DrawWidth*** de la zone d'image (***Let Picture1. DrawWidth = 2***)

Si la propriété ***FillStyle*** de la zone d'image possède la valeur 0 (***Let Picture1.FillStyle = 0***), alors les cercles et les rectangles sont remplis de la couleur définie par la propriété ***FillColor*** de la zone d'image (***Let Picture1.FillColor = RGB(r,g,b)***).
Si cette propriété *FillStyle* possède la valeur 1 alors les cercles et les rectangles ne sont pas remplis.

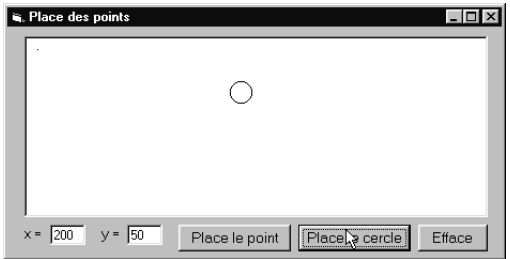
Exemple

Les instructions ci-dessous permettent de tracer un carré rouge à bords bleus de 100 pixels de côté au milieu de la zone d'affichage.

```
Let xc=Picture1.width/2
Let yc=Picture1.height/2
Let Picture1.FillStyle = 0
Let Picture1.FillColor = RGB(255,0,0)
Let Picture1.ForeColor = RGB(0,0,255)
Picture1.Line(xc-50,yc-50) – (xc+50,yc+50),,B
```

Exercice 34

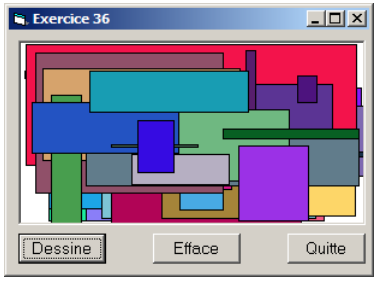
Créer le programme illustré ci-contre qui permet de donner dans des zones de texte l'abscisse et l'ordonnée d'un point puis, par un *clic* sur un bouton de commande, de dessiner ce point dans une zone d'image.



Un autre bouton de commande doit permettre de dessiner un cercle de rayon 10 (10 points graphiques).
Un troisième bouton permet d'effacer le dessin.

Exercice 35

Écrire un programme qui dessine un échiquier. L'utilisateur doit pouvoir préciser le nombre de cases.

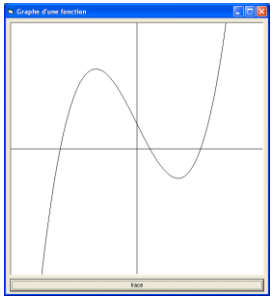


Exercice 36

Écrire un programme qui trace des rectangles de dimensions et de couleurs choisies aléatoirement.

Exercice 37

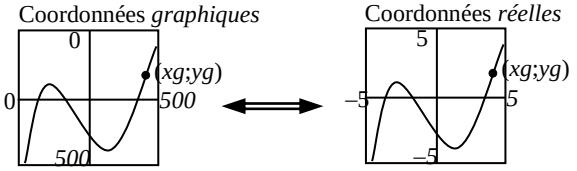
Écrire un programme qui dessine le graphe de la fonction $y = \frac{1}{4}x^3 - 2x + 1$ dans une zone d'image de 500 x 500 pixels, pour des valeurs de x et de y comprises entre –5 et 5.



Remarque :

Pour dessiner un point de coordonnées x et y, il faut effectuer un *changement de coordonnées* pour obtenir des coordonnées *graphiques* xg et yg à partir des coordonnée réelles x et y.

La transformation entre une coordonnée réelle x et la coordonnée graphique xg est donnée par une fonction affine, $xg = a\,x + b$. Pour trouver a et b, il faut poser $xg = 0$ si $x = -5$ et $xg = 500$ si $x = 5$. On obtient ainsi un système de 2 équations à 2 inconnues à résoudre. De la même façon on trouve la transformation entre y et yg ainsi que les réciproques de ces 2 fonctions, elles permettent de déterminer les coordonnées réelles $(x;y)$ d'un point donné par ses coordonnées graphiques $(xg;yg)$.



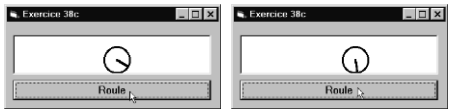
On obtient : $xg = 50 \cdot x + 250\,;\ yg = -50 \cdot y + 250$ et $x = \frac{xg}{50} - 5\,;\ y = -\frac{yg}{50} + 5$

Exercice 38

Écrire un programme qui dessine la courbe donnée par l'équation paramétrique $\begin{cases} x = \sin(2t) \\ y = \cos(5t) \end{cases}$ pour $t \in [0;7]$ et $-2 \le x \le 2$ et $-2 \le y \le 2$

Exercice 39

a) Écrire un programme qui dessine un cercle qui se déplace sur une droite horizontale.
b) Écrire un programme qui dessine un cercle qui se déplace horizontalement et rebondi sur les bords de la zone d'image.



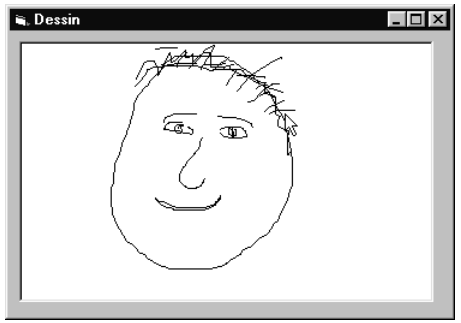
Diverses procédures événementielles permettent d'utiliser la souris pour réaliser des programmes graphiques :

- *Picture1_MouseDown* est exécutée lorsqu'un bouton de la souris est pressé sur le contrôle *Picture1*.
  La variable *Button* permet de savoir quel bouton a été pressé, *Button* prend la valeur 1 si le bouton de gauche est pressé et la valeur 2 pour le bouton de droite (4 pour le bouton du milieu).
  Les variables *X* et *Y* décrivent les coordonnées du pointeur de la souris dans le contrôle *Picture1*.
- *Picture1_MouseUp* est exécutée lorsqu'un bouton de la souris est relâché. Les variables *Button, X* et *Y* sont mises à jour.
- *Picture1_MouseMove* est exécutée à chaque mouvement de la souris. Les variables *Button, X* et *Y* sont misent à jour. (*Button* prend la valeur 0 si aucun bouton n'est pressé)

Exemple 1 :

Le programme ci-dessous permet de dessiner avec la souris :

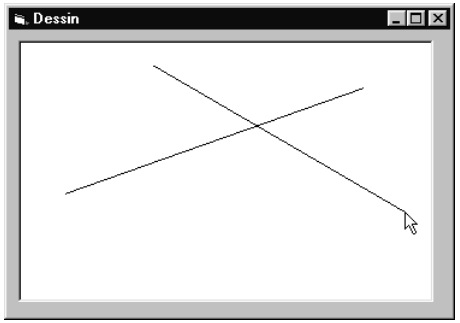
```
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Picture1.PSet (X, Y)
End Sub
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 Then
        Picture1.Line -(X, Y)
    End If
End Sub
```



Exemple 2 :

Le programme ci-dessous permet de dessiner des segments de droites avec la souris :

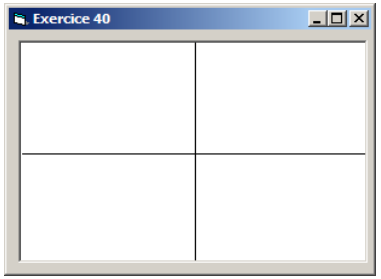
```
Dim X0, Y0, AX, AY As Integer
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Picture1.PSet (X, Y)
    Let X0 = X: Let Y0 = Y: Let AX = X: Let AY = Y
End Sub
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 Then
        Picture1.Line (X0, Y0)-(AX, AY)
        Picture1.Line (X0, Y0)-(X, Y)
        Let AX = X: Let AY = Y
    End If
End Sub
```

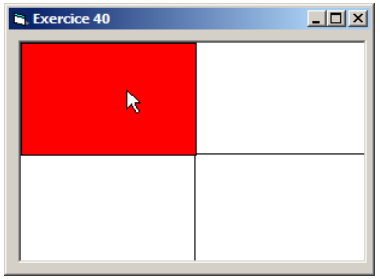


Remaque : la propriété *DrawMode* définit le type de points avec lesquels les objets sont dessinés. Si *DrawMode* a la valeur 13 (*copy pen*) alors les points sont *allumés*, si *DrawMode* a la valeur 10 (*not xot pen*) alors les points sont *inversés.*

Exercice 40

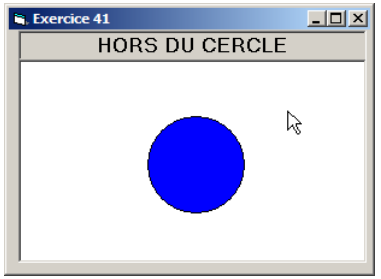
Créer un programme qui trace un échiquier formé de 4 cases blanches puis qui permette de cliquer une case avec la souris pour la colorier. Le programme doit déterminer quelle case a été désignée par la souris.





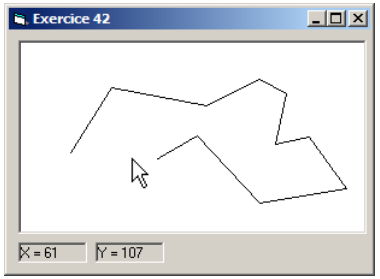
Exercice 41

Créer un programme qui dessine un cercle de rayon aléatoire et centré au milieu d'une zone d'image puis qui affiche si le pointeur de la souris est situé dans ce cercle ou non.



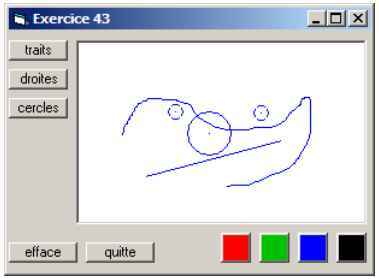
Exercice 42

Créer un programme qui permet de placer des points par des clics de la souris sur une zone d'image, qui trace une ligne brisée qui relie tous les points placés et qui affiche les coordonnées du pointeur de la souris.



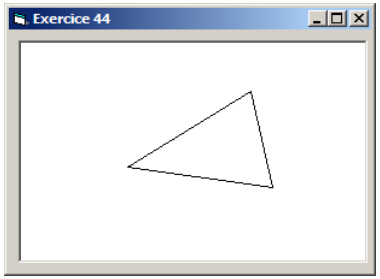
Exercice 43

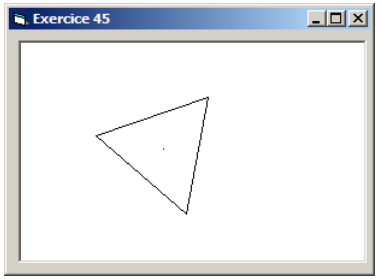
Créer un programme qui permet de dessiner des traits à main levée, des segments de droites et des cercles et de choisir la couleur du dessin.



Exercice 44

Créer un programme qui permet de placer 3 points avec la souris, puis qui dessine un triangle dont les sommets sont les 3 points donnés.





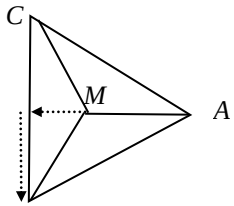
Exercice 45

Créer un programme qui dessine un triangle équilatéral donné par son centre et par un de ses sommets.

Indications : démontrer puis utiliser :

$$\overrightarrow{MB} = -\frac{1}{2}\overrightarrow{MA} + \frac{\sqrt{3}}{2}\vec{n} \quad \text{et} \quad \overrightarrow{MC} = -\frac{1}{2}\overrightarrow{MA} - \frac{\sqrt{3}}{2}\vec{n}$$

où $\vec{n}$ est un vecteur normal à $\overrightarrow{MA}$ et de même norme que $\overrightarrow{MA}$.



Procédures

En plus des procédures événementielles qui sont automatiquement exécutées lorsqu'un événement se produit, il est possible de créer des procédures qui regroupent une partie d'un programme et qui sont exécutées par appel.

Une procédure doit être écrite dans la partie *Générale* du programme, elle débute par *Sub Nom_de_la_procédure* et se termine par *End Sub*.
L'instruction *Call Nom_de_la_procédure* permet d'exécuter la procédure. Il est possible de faire suivre le nom de la procédure par une liste de paramètres.

Exemple

La procédure suivante affiche sur l'étiquette *Label1* les valeurs de 2 variables.

```
Sub Affiche(x,y)
    Let Label1.caption="x = " & x & " - " & " y = " & y
End Sub
```

L'instruction *Call Affiche(4,6)* appelle cette procédure.

Fonctions

Une fonction est une procédure qui *rend* une valeur.

Une fonction doit être écrite dans la partie *Générale* du programme, elle débute par *Function Nom_de_la_procédure* et se termine par *End Function*.

Pour exécuter une fonction, il suffit de l'appeler par son nom.

Exemple

La fonction suivante calcule la norme d'un vecteur.

```
Function Norme(X,Y)
    Let Norme = Sqr(X^2 + Y^2)
End Function
```

L'instruction *Let Label1.Caption=Norme(3,4)* affiche 5 sur l'étiquette Label1.

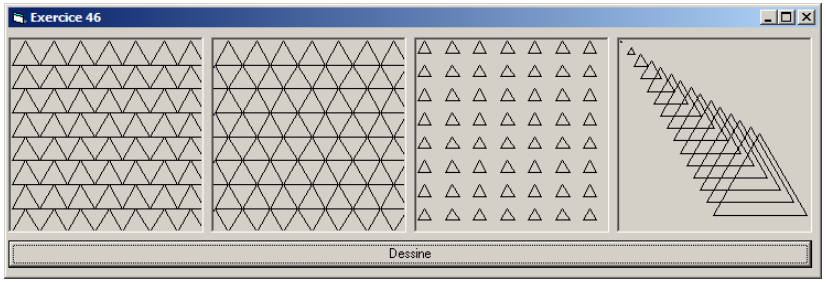
Il est possible de préciser les types de variables utilisées.
Exemple : *Function Norme(X As Single,Y As Single) As Single*

Les procédures et les fonctions sont utilisées pour rendre la structure d'un programme plus claire en regroupant et nommant de petites actions.

Exercice 46

a) Écrire une procédure *Triangle(CONTROLE,X1,Y1,R)* qui trace, sur une zone d'image nommée *CONTRÔLE*, un triangle équilatéral de coté *R*, de base horizontale et dont l'un des sommet est (*X*1;*Y*1*)*

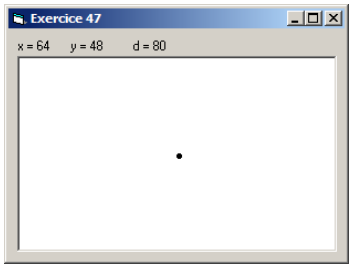
b) Reproduire les dessins ci-dessous en utilisant la procédure *Triangle*.



Exercice 47

a) Écrire une fonction *Distance(X1,Y1,X2,Y2)* qui rend la distance (en pixels et arrondi au dixième) qui sépare les points *(X1,Y1)* et *(X2,Y2).*

b) Écrire un programme qui affiche en permanence la distance entre le centre d'une zone d'image et le pointeur de la souris.



Les **variables indicées** permettent de mémoriser un ensemble de valeurs. Par exemple, pour traiter un polynôme du 3ème degré, il est évidemment possible d'utiliser 4 variables *A*, *B*, *C* et *D* pour mémoriser les coefficients, mais il est plus pratique d'utiliser des **variables indicées** *A*(0), *A*(1), *A*(2) et *A*(3). Le polynôme s'écrira ainsi *Y* = *A*(3)**X*^3 + *A*(2)**X*^2 + *A*(1)**X* + *A*(0).

Les variables indicées peuvent comporter 1 indice (tableaux à une dimension ou liste de nombres), à 2 indices (tableau à 2 dimensions) ou plus.
Il est nécessaire de déclarer les variables indicées (Exemple : *Dim A(10,10) As Interger*).

Exemples

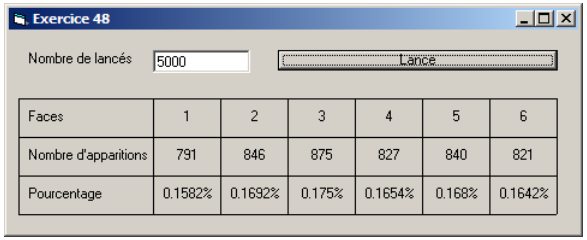
```
Dim A(5) As Integer
Sub Exemple
  For I = 1 to 5
    Let A(I) = I^2
  Next I
End Sub
```

```
Dim A(10,10) As Single
Sub Exemple
  For I = 1 to 10
    For J = 1 to 10
      Let A(I,J) = I/J
    Next J
  Next I
End Sub
```

Les contrôles de Visual Basic peuvent également être indicés, il suffit pour cela de placer un premier contrôle sur la feuille, par exemple une étiquette nommée *Label1*, de copier, puis de coler ce contrôle. Label1 devient ainsi Label1(0) et un nouveau contrôle label1(1) a été créé. En réitérant ce procédé il est possible de créer une suite de controles indicés.

Exercice 48

Écrire un programme qui simule *N* lancers d'un dé (*N* donné par l'utilisateur) et qui affiche le nombre d'apparition de chaque face ainsi que le pourcentage de ces apparitions.



Faces	1	2	3	4	5	6
Nombre d'apparitions	791	846	875	827	840	821
Pourcentage	0.1582%	0.1692%	0.175%	0.1654%	0.168%	0.1642%

Exercice 49

Écrire un programme qui permet de mémoriser une suite de points *cliqués* à l'écran.
Un *clic* du bouton de gauche de la souris permet de définir un nouveau point.
Un *clic* du bouton de droite de la souris permet de terminer l'introduction des points. Le programme doit alors construire un polygone dont les sommets sont les points donnés.

Exercice 50

Écrire un programme qui crée une variable indicée nommée *Premier(..)* qui contient tous les nombres premier plus petits ou égal à *N* (*N* donné par l'utilisateur) puis qui affiche cette liste dans une zone de liste.

Exercice 51

La conjecture de Goldbach affirme que tout entier pair *n*, *n*>=4, est somme de 2 nombres premiers.
Écrire un programme qui permette de vérifier la conjecture de Goldbach pour les nombres pairs inférieurs à 10'000.

Exercice 52

Ecrire un programme qui permet d'écrire un nombre en base 2, enregistré dans une variable indicée, et convertit ce nombre en base 10.

Exercice 53

Après avoir créé 81 étiquettes, écrire le programme qui permet de les aligner comme ilustré ci-contre, puis qui créée une table de multiplication.