UNIVERSITE DES SCIENCES SOCIALES DE TOULOUSE
Licence Professionnelles LSi
Master FC IGSI

Programmation Visual Basic

Visite guidée d'un programme Visual Basic 6.0

1. un exemple d'application
a) créer dans votre compte personnel un répertoire nommé "VB"
b) vous connecter sur ftpsip/christophe.sibertin-blanc/VBUV8
c) recopier dans votre répertoire VB le répertoire "Exemple" (Tout le répertoire, pas les fichiers un à un)

2. utiliser cette application
Dans le répertoire Exemple que vous venez de créer,
lancer l'exécution du programme "moyenne.exe" et tester cette application ;
Trouver les "bugs" qu'elle contient

3. étudier la structure de cette application
a) lancer l'application VisualBasic, et ouvrir le projet "moyenne"
b) lancer l'exécution du programme et tester cette application; provoquer une erreur d'exécution.
c) menu Affichage
ouvrir/fermer les principales fenêtres
ne garder que objet, code, boite a outils et propriétés

d) sur la fenêtre objet
sélectionner les différents éléments et voir leurs propriétés
notion de type, caption/name, position, ...
ajouter d'autres éléments à partir de la boite à outil

e) Module de code basic : les fonctions de traitement

f) fenêtre de code de la feuille :
le code des traite-événements associé aux boutons

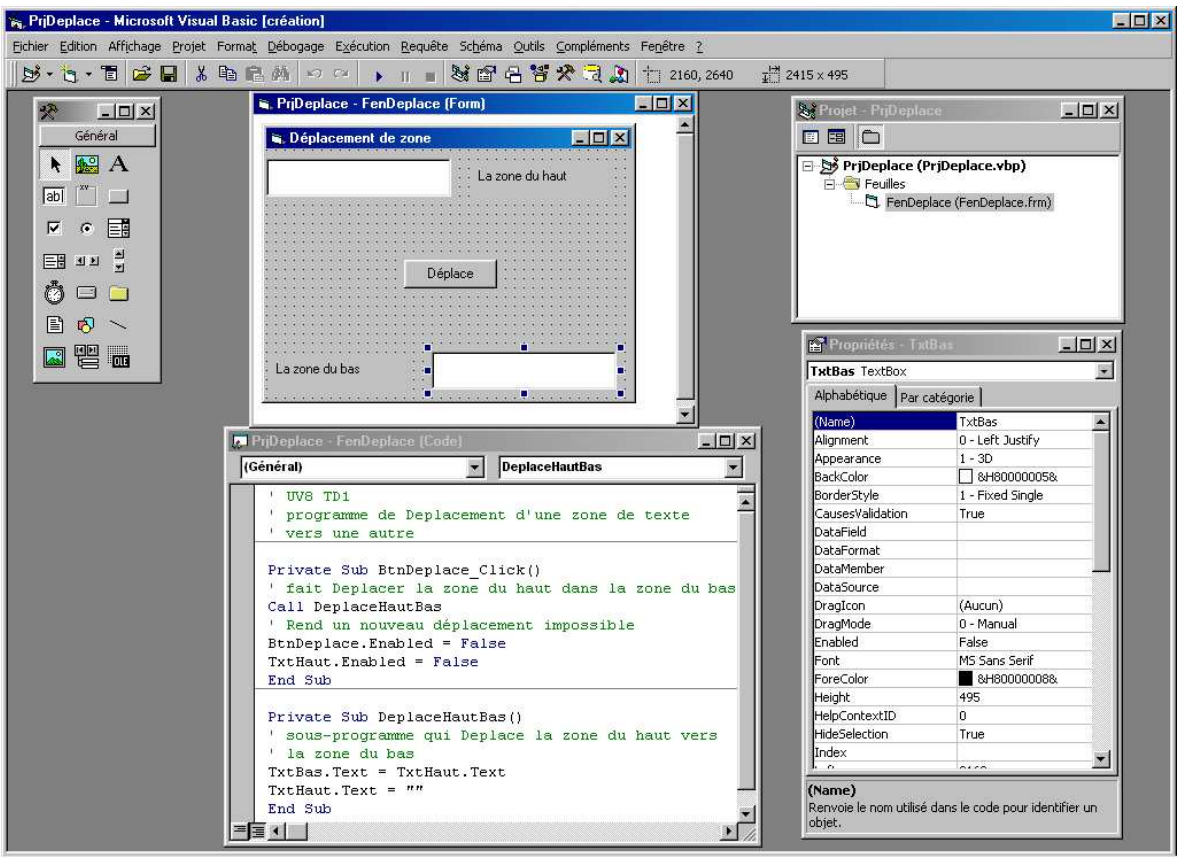
4) Fonctions offertes par l'environnement VB
a) Rôle des différentes fenêtres
savoir organiser son écran pour n'y avoir que ce qui est utile

b) parcourir les différents menus, notamment
l'exécution
l'édition de menus

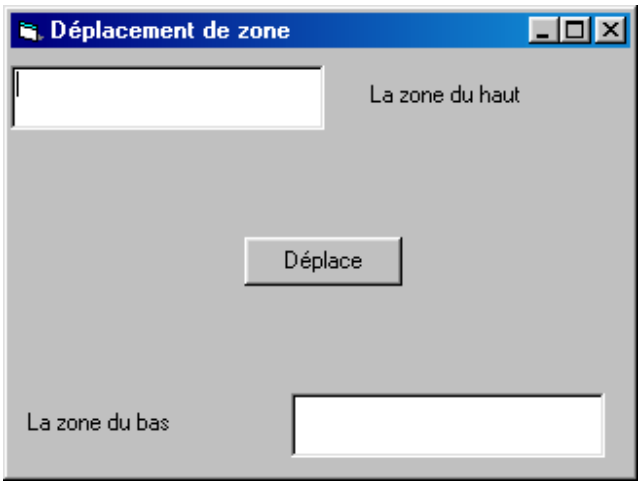
5) Refaire les étapes 1-4 avec le projet "moyenne"

Bien distinguer les rôles développeur / utilisateur de l'application

Les fenêtres de l'environnement de développement

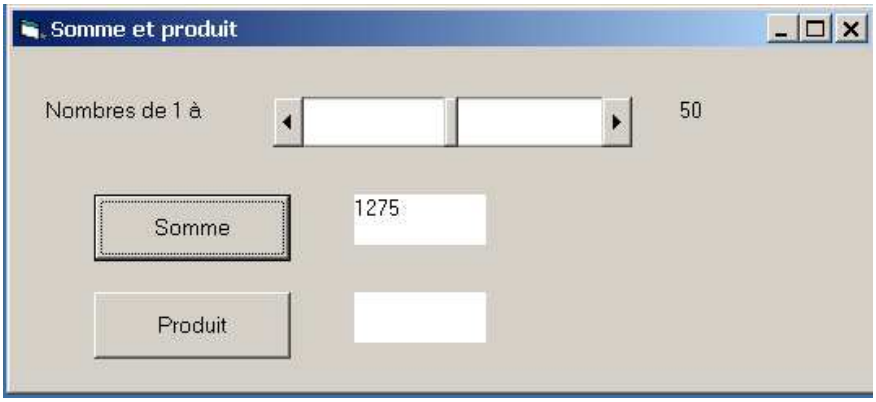


La fenêtre de l'application en cours d'exécution



Somme et produit des N premiers entiers

Les contrôles et leurs propriétés



Spécification

On souhaite créer un programme capable de calculer la somme et le produit des N premiers nombres entiers
(1 < N <= 100), N étant un paramètre fourni par l'utilisateur à l'aide d'une "ScrollBar" (ascenseur horizontal).

Pendant le déplacement du curseur de la barre, la valeur numérique correspondant à la position du curseur doit être continuellement rafraîchie (204 dans l'exemple).

L'activation du bouton Somme affiche la somme des N premiers entiers (ici 20910), l'activation du bouton Produit affiche leur produit.

Indications

Une scrollbar est un contrôle qui associe une valeur entière à la position du curseur ; cette valeur est accédée par les propriétés suivantes :
Value : une valeur entière déterminée par la position du curseur ;
Min : entier qui détermine la valeur minimale de la propriété **value** (curseur à l'extrémité gauche).
Max : entier qui détermine la valeur maximale de la propriété **value** (curseur à l'extrémité droite).

Tout déplacement du curseur, met automatiquement à jour la propriété value.

L'utilisateur dispose de différentes façons pour déplacer le curseur qui correspondent aux d'événements suivants :

- déplacement directe du curseur en le traînant : événement **scroll**
- clic à droite ou gauche du curseur, y compris aux extrémités : événement **change**

Réalisation

1) **Sur papier**, dessiner la feuille correspondante et pour chacun des contrôles, préciser son type et choisir son identificateur (la propriété (name)) de.
2) **Sur machine**, réaliser la feuille puis écrire l'instruction du traite-événement correspondant au déplacement du curseur (affichage de la valeur de N). **Tester** votre programme.
3) **Sur machine**, créer un module de code (commande du menu Projet) dans lequel vous saisissez la fonction suivante :

```
function calc_somme (n as integer) as long
dim resultat as long
dim i as integer
resultat = 0
for i=1 to n
    resultat = resultat + i
next i
calc_somme = resultat
end function
```

4) **Sur papier**, écrire l'instruction du traite-événement correspondant à l'activation du bouton de commande Somme (cette procédure doit appeler la fonction `calc_somme`), puis s**ur machine** saisir le code. **Tester** votre programme.
5) **Sur papier**, écrire le code de la fonction `calc_produit` (en vous inspirant de la fonction `calc_somme`), puis saisissez cette fonction dans le même module de code.
6) **Réaliser** le traite-événement correspondant à l'activation du bouton de commande Produit selon le même principe qu'en 4). **Tester** votre programme.

Compléments

1) La scrollbar est trop sensible ?

Essayons de gérer également des déplacements plus courts qui sont possibles en cliquant avant ou après le curseur ou aux extrémités de la barre de défilement**.** L'événement à gérer est alors **change** et les propriétés **LargeChange** et **SmallChange.** A vous de jouer !

2) On pourrait conserver l'historique des calculs dans deux ListBox, faisant apparaître la valeur de N et la somme ou le produit respectivement. Nous allons le faire dans une nouvelle application.

a) Concevoir la présentation de la nouvelle application, choisir l'identificateur (propriété (`name`)) de ces deux listes.

b) Créer un nouvelle application qui réutilisera l'essentiel de ce que vous venez de faire :
- enregistrer (menu Fichier) la feuille sous un autre nom,
- enregistrer le projet sous un autre nom,
- vérifier, par l'explorateur, que vous avez 2 fichiers **.**frm et 2 fichiers **.**vbp.

La nouvelle application utilisera le même module de code que la précédente.

c) Modifier la présentation de l'application, conformément à ce que vous avez fait en a).

d) Ecrire le nouveau traite-événement des deux boutons en utilisant la méthode additem :

<NameListBox>**.**additem <nameScrollBar>.value & " –> " & calc_somme (<nameScrollBar>.value)

Devine COMBIEN !

La dialogue entre l'Utilisateur et l'application. Maîtriser l'utilisation des fenêtres `Objet` *et* `Code`*.*
Importance des identificateurs : propriété (`name`*) des objets, identificateur des procédures de traîte-événement.*

Spécification

On veut réaliser une application qui permette de faire des parties du jeu suivant :

- l'application choisit un nombre au hasard, entre 1 et 100, que l'utilisateur doit découvrir
- elle demande à l'utilisateur de proposer un nombre. Si l'utilisateur propose le nombre à découvrir, il a gagné; sinon, l'application lui indique si le nombre qu'il a proposé est trop petit ou trop grand.

L'application doit offrir à l'utilisateur les services (ou fonctionnalités) suivants :

1. débuter une nouvelle partie,
2. faire une proposition,
3. connaître le nombre de propositions faites depuis le début de la partie en cours,
4. faire afficher la solution.

Démarche

1. Pour chacun de ces services, déterminer les entrées à fournir par l'utilisateur et les résultats à afficher par l'application.
2. Sur papier, dessiner la présentation de l'application ;
   bien préciser l'identificateur de chaque objet, c'est-à-dire la valeur de sa propriété (`name`).
3. Réaliser la feuille VB correspondante, la tester.
4. Sur papier, déterminer ce que doit faire l'application lorsque l'utilisateur commande :
   a) de faire une nouvelle partie
   b) d'afficher la solution.
5. Programmer le code des traîte-événements correspondants. Exécuter et vérifier
6. Sur papier, déterminer ce que doit faire l'application lorsque l'utilisateur fait une nouvelle proposition.
7. Coder le traîte-événement correspondant. Exécuter et vérifier.
8. même démarche pour la fonctionnalité permettant de connaître le nombre de propositions
9. créer un fichier .exe.

Indications

1) l'expression `Int(Rnd*100)` retourne un nbre aléatoire compris entre 0 et 100
2) le générateur de nbres aléatoires est initialisé par l'instruction `randomize`
3) la fonction `isnumeric` teste si une suite de caractères correspond à une valeur numérique
4) l'affichage de la valeur d'une variable numérique `xxx` peut être réalisée
   - par l'instruction `msgbox ("la solution est " & str$(xxx))`
   - avec une zone de texte, par l'instruction `unezonedetexte.text = str$(xxx)`
5) Saisir une valeur donnée par l'utilisateur dans une variable numérique `xxx` :
   a) dans une boîte de dialogue : `xxx = val(Inputbox$("Quelle est votre proposition ?"))`
   b) dans une zone de texte : `xxx = val(unezonedetexte.text)`

Compléments

1) Au démarrage de l'application,
   initialiser automatiquement une nouvelle partie, en provoquant l'occurrence de cet événement.
2) Après affichage de la solution, provoquer une nouvelle partie.
3) Si la saisie se fait dans une zone de texte, effacer automatiquement la valeur précédente
4) Limiter les mouvements que l'utilisateur doit faire avec la souris
   (utiliser la méthode `setfocus`, l'événement `KeyPressed`)
6) Permettre à l'utilisateur de choisir les bornes entre lesquelles le nombre est à chercher
7) afficher l'encadrement courrant du nombre à découvrir (le plus grand des trop petits - le plus petit des trop grands)
   ou bien la liste des propositions précédentes avec la réponse correspondante
8) Gérer les scores réalisés par l'utilisateur pendant une suite de parties

Un logiciel pédagogique

Utilisation des menus. Evolution de la présentation d'une application.

Description de l'application

On se propose de réaliser un petit logiciel pédagogique d'entraînement aux mathématiques.
Ce logiciel permet à l'utilisateur :

- de répondre à des questions sur les multiplications
- de répondre à des questions sur les nombres premiers
- d'obtenir la solution à une question qui lui a été posée.
- de connaître son score (nbre de questions posées, nbre de réponses correctes)
- de déterminer la borne supérieure des nombres figurant dans les questions posées

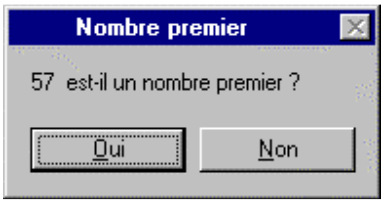
Lorsque l'utilisateur sélectionne la fonction "multiplication", il voit apparaître la fenêtre suivante :

Les deux nombres à multiplier sont choisis au hasard par le programme, entre 1 et la borne supérieure, par exemple 20.

Lorsque, après avoir saisi un résultat, l'utilisateur clique sur "OK" une étiquette colorée apparaît alors dans la fenêtre principale ; dans cette étiquette est inscrit "BRAVO !" si la réponse était correcte, et "Pas Fameux !" sinon.

Remarque : cette étiquette disparaît dès que l'utilisateur clique l'un des deux boutons "multiplication" ou "nbre premier".

Lorsque l'utilisateur sélectionne la fonction "nbre premier", il voit apparaître la fenêtre suivante:
Le nombre affiché est choisi au hasard par le programme entre 1 et la borne supérieure.



Lorsque l'utilisateur clique sur l'un des boutons Oui ou Non, apparaît dans la fenêtre principale une étiquette dans laquelle est inscrit "BRAVO !" si la réponse était correcte, et "Pas Fameux !" sinon.

La fonction "solution" fait afficher la réponse à la question qui vient d'être posée.

La fonction score permet à l'utilisateur de faire apparaître ou disparaître un cadre dans lequel figurent le nombre de questions posées (c'est à dire le nombre de fois que l'utilisateur a activé "multiplication" ou "nbre premier"), ainsi que le nombre de bonnes réponses fournies.

Démarche

1• Réalisation de la présentation de l'interface

On associera les fonctions "multiplication" et "nbre premier" à des boutons, et les autres à un menu. Concevoir et réaliser la présentation.

2• Réalisation du traite-événement associé au bouton "multiplication"

a) Ecrire sur papier l'algorithme de la procédure (indication 1, 2) :
- poser la question et obtenir la réponse de l'utilisateur (utiliser la fonction `InputBox`)
- traiter la réponse

b) Saisir le code de ce traite-événement, et le tester.

3• Réalisation de la fonction associé au bouton "solution"

Pourquoi faut-il placer dans général/(déclaration) la déclaration des 2 variables qui mémorisent les nombres figurant dans la question ?

4**• Gestion du score**

a) quelles sont les données nécessaires ? Où les déclarer ? Où les initialiser ?
b) présentation : définir les contrôles permettant à l'utilisateur de voir ces données
c) comment mettre à jour ces données ? Comment mettre à jour leur présentation ?
d) affichage : On souhaite que les éléments de la présentation concernant le score s'affichent lorsque la commande correspondante du menu est cochée, et disparaissent dans le cas contraire. Faire un menu, associer le code à la commande score.

5• Réalisation du traite-événement associé au bouton "nbre premier"

a) Ecrire sur papier l'algorithme de la procédure qui s'exécute lorsque l'utilisateur clique sur ce bouton. (indication 3, 4).
b) Saisir le code basic de ce traite-événement, et le tester.

Indications

1) Pour obtenir un nombre aléatoire entre 1 et bornesup, utiliser l'expression `Int(rnd * bornesup)`.
2) La propriété "visible" d'un contrôle permet de le faire afficher ou non sur la feuille.
3) Cet algorithme peut être basé sur la propriété suivante : un nombre inférieur à 100 est premier si et seulement si il n'est pas divisible 2, 3, ..., racine(100). N est divisible par x ssi (N mod x = 0).
4) Regarder l'aide en ligne concernant la fonction MsgBox pour faire afficher une telle fenêtre et savoir si l'utilisateur a cliqué sur le bouton Oui ou sur le bouton Non.

Compléments

Paramétrage

Ajouter une commande au menu pour permettre à l'utilisateur de déterminer la borne supérieure des nombres intervenant dans les questions posées (dans ce qui précède, cette valeur était égale à 20; il s'agit donc d'en faire une variable).

Une fonction d'aide

Ajouter au logiciel un autre élément de menu, qui permet à l'utilisateur de faire apparaître ou disparaître l'affichage d'une aide.

• Pour les nombres premiers, affichage d'une étiquette indiquant l'ensemble des nombres premiers inférieurs à 100. Créer un menu "aide", comportant un sous-menu "nbres premiers" tel que, lorsque l'utilisateur le sélectionne,

Si l'élément de menu n'est pas coché,
alors il devient coché, la procédure est appelée, et l'étiquette s'affiche
Si l'élément de menu est coché,
alors il n'est plus coché et l'étiquette disparaît.

• Pour les multiplications, affichage d'une étiquette montrant la table de multiplication des nombres de 1 à 10. Créer, dans le menu "aide", un sous-menu "multiplication" qui fonctionne comme le précédent.

Plan d'amortissement d'un bien

Analyser un pb, et concevoir un algorithme le plus simple possible. Chargement dynamique de contrôles
L'environnement de mise au point (pt d'arrêt, espions, fenêtre des variables)

Spécification

Pour un bien dont la valeur est donnée par l'utilisateur, l'application affiche un tableau (ou plan) d'amortissement indiquant, pour chaque année, le montant de l'amortissement et la valeur résiduelle du bien en fin d'année.

Le bien est amorti de façon dégressive durant les premières années, de façon linéaire ensuite, et de sa valeur résiduelle la dernière année. Le montant annuel de l'amortissement est donc calculé en appliquant les règles suivantes :

r1 : les premières années, l'amortissement est égal à 20% de la valeur résiduelle du bien
r2 : l'amortissement ne peut pas être inférieur à 10% de la valeur initiale du bien
r3 : la dernière année, l'amortissement est égal à la valeur résiduelle du bien

Réalisation

1) Traitez un exemple, pour vous assurer que vous avez bien compris le principe, et dans quel ordre les règles doivent être appliquées (on ne peut programmer un algo pour résoudre un pb. si on ne sait pas résoudre ce problème).

2) Dessiner la Présentation (interface utilisateur) de l'application, par exemple :

• une zone de texte, `txvalinit`, dans laquelle l'utilisateur saisit la valeur du bien.
• un bouton de commande, `bttableau`, qui provoque l'affichage du tableau d'amortissement.
• pour l'affichage du tableau d'amortissement : une ListBox comportant une ligne de donnée pour chaque année

3) Codage et test de cette Présentation

4) Codage de la structure de données globale :

L'analyse du pb fait apparaître les données suivantes à déclarer dans (général)/déclaration

```
Const CoeffADeg = 0.2          'coefficient pour la période d'amortissement dégressif
Const CoeffALin = 0.1          'coefficient pour la période d'amortissement linéaire
```

5) Conception de l'algorithme de calcul du tableau, puis codage de `bttableau_click()`

Algorithme de la procédure `bttableau_click()`

```
                    'calcule et affiche le plan d'amortissement
Dim valinit as currency          'la valeur initiale du bien
Dim an as integer          'l'année courante
Dim amort as currency      'le montant de l'amortissement pour l'année courante
Dim valres as currency           'la valeur résiduelle du bien à l'issue de l'année courante
                    'initialisation
valinit = val(txvalinit.text)
                    'calcul et affichage pour chacune des années
an = 0
valres = valinit
while (valres >0)
   an = an + 1
                'calcul de l'amortissement de l'année
      . . .
                'calcul de la valeur résiduelle en fin d'année
   valres = valres - amort
                'affichage du résultat
      . . .
wend
```

Indication

Pour ajouter une ligne dans une ListBox, utiliser la méthode A*ddItem*.

Compléments

1. Affichage du résultat dans une TextBox différente pour chaque année :

Créer dans la présentation une zone de texte `txresultat`, ayant la valeur 0 pour sa propriété `index`.

Pour afficher le résultat d'une année,

1. créer une copie de `txresultat` par l'instruction `load txresultat(an)`
2. rendre visible et positionner ce nouveau contrôle : `txresultat(an).visible = 1`, `txresultat(an).top = ...`
3. `txresultat(an).text =` l'année, le montant de l'amortissement, et la valeur résiduelle en fin d'année

2. Affichage du résultat dans une seule TextBox dont la propriété `multiline` est mise à vrai ;
`chr(13)` permet d'ajouter un saut de ligne dans une chaîne de caractères.

3. au lieu d'afficher l'amortissement et la valeur résiduelle pour chaque année au fur et à mesure de leur calcul, on mémorise ces valeurs dans un tableau. On introduit une procédure `calc_tableau` qui remplit ce tableau ; le traite-événement `bttableau_click` se contente alors d'appeler cette procédure puis d'afficher le résultat.

Gérer une liste

Spécification

On veut réaliser une application qui permette à l'utilisateur de gérer une liste de valeurs numériques (par exemple des notes comprises entre 0 et 20). On suppose que la taille maximum de cette liste est fixée par une constante NBMAXINOTES.
On ne cherche pas à lier ces valeurs à un fichier dans lequel elles pourraient être lues et enregistrées.
L'application doit permettre à l'utilisateur de :

1. saisir une nouvelle note;
2. connaître la note minimale et la note minimale;
3. calculer la moyenne des notes;
4. rechercher le nombre d'occurrences d'une note donnée par l'utilisateur;
5. calculer la distribution des notes, c'est à dire le nombre de notes supérieures à la moyenne;

Présentation

On fait les choix suivants pour la présentation de l'application :

1. la saisie d'une nouvelle note se fait par l'intermédiaire d'un contrôle de type `text` que l'on nomme `txnote`;
2. l'ensemble des notes saisies sont affichés dans un contrôle de type ListBox ;
3. l'utilisateur active les services offerts par l'application par l'intermédiaires de boutons de commande.

Structure de l'application

On souhaite bien distinguer ce qui relève de la présentation et ce qui relève des traitements et calculs.
Donc on introduit dans l'application un module de code, de telle sorte que les procédures de traite-événements ne fassent que

1. si besoin est, lire les valeurs saisies par l'utilisateur
2. appeler des fonctions du modules de code, avec leurs paramètres
3. mettre à jour la présentation.

Le module de code pourra contenir les déclarations suivantes :

```
Option Explicit
Public Const NBMAXINOTES = 10
dim tb (1 to NBMAXINOTES) as single     'mémorisation des notes
Public nbnotes as integer               'le nombre de notes déjà saisies
```

Indications

1. Pour inclure un module de code dans un projet : voir le menu projet

2. Les ListBox :

méthodes : `addItem, removeItem` : ajout, retrait d'éléments dans la liste.
`list (i%)` : pour accéder à l'élément d'indice i% de la liste
propriétés : `listcount` : nombre d'éléments de la liste

Compléments

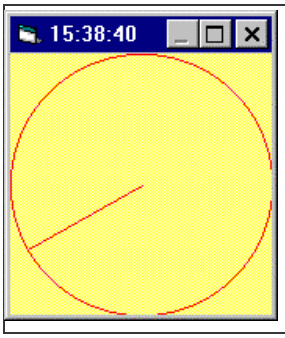
1. Validation de la saisie d'une note par la touche Entrée :

```
Private Sub Txnote_KeyPress(KeyAscii As Integer)
If (KeyAscii = 13) Then Call BtnAjout_Click
End Sub
```

2. Trier la liste des notes par valeur croissante

Programmation d'une horloge

La dimension et le positionnement des contrôles. L'aide en ligne.



Cette horloge est à la fois numérique (dans la barre de titre) et analogique (dessin dans le corps de la fenêtre).
On pourra ne programmer que l'aiguille des secondes, le principe étant le même pour les autres.
Colorer l'aiguille.
Faire un fichier .exe de l'application, qu'un utilisateur pourra lancer depuis Windows.

Un contrôle de type Timer
Ce contrôle génère automatiquement un événement toute les N milisecondes, N étant la valeur de sa propriété `interval`. En lui donnant le `(name)` minuterie1, son traîte-événement est :

```
sub minuterie1_timer ()
  form1.caption = time$     'affiche l'heure (numérique) dans la barre de
titre de la fenêtre
  dessine          'dessine le cercle et l'aiguille dans le corps de la fenêtre
end sub
```

les données globales dans (générale)/déclarations

```
dim cx as integer      'abcisse du centre du cercle
dim cy as integer      'ordonnées du centre du cercle
dim r as integer       'le rayon du cercle
```

Initialisation

```
sub form_load ()
    'assure que la fenêtre est un carré
  form1.width = form1.scaleheight
    'positionner le centre du cercle est au centre de la fenêtre
  cx = form1.scalewidth / 2
  cy = form1.scaleheight / 2
  r = cx
end sub

sub dessine ()
        'dessine l'horloge analogique, en appliquant les rudiments de trigonométrie
  Const Pi =  3.1416
  Dim angle as single
     'traduit le nb de secondes en angle, 60s =2.pi (radian)
  angle = Pi / 2 - (second (now) * 2 * Pi )/ 60
     'la méthode cls efface ce qui était précédemment affiché dans cadran
  form1.cls
     'la méthode circle dessine un cercle de centre (cx, cy) et rayon r
  form1.circle (cx, cy), r
     'dessine l'aiguille,  REGARDER L'AIDE EN LIGNE
 form1.line (cx, cy) ......
end sub
```

Compléments
Permettre à l'utilisateur d'ajuster la taille de la fenêtre.