

SLFD1

Informatique

Enseignant responsable : Maria ZIMINA

INTRODUCTION A LA PROGRAMMATION EN VISUAL BASIC

Programmer - c'est écrire dans un langage informatique une suite d'instructions organisées en algorithme dans un but précis.

Langage de programmation VISUAL BASIC, caractéristiques :

- Ancien BASIC (Beginner's All purpose Symbolic Instruction Code)
- Programmation par objet (briques logicielles)
- Programmation graphique (fenêtres, icônes, menus, souris)
- Programmation événementielle (solicitation : souris, clavier, autre événement)
- Réutilisable (modules de code BASIC)

Implantation du code

Le code en programmation est une suite d'instructions prédéfinies.

Le programme est constitué de mots reliés par des *séparateurs*. Le séparateur le plus important est le blanc. Les autres séparateurs sont (), { }, " " et les opérateurs arithmétiques : +, -, *, : . Comme dans la plupart des autres langages, les chaînes de caractères constantes sont entourées des guillemets. Attention, le langage Visual Basic fait la différence entre les majuscules et les minuscules.

Exemple 1:

```
Sub Bonjour ( )  
    MsgBox ("Salut tout le monde !")  
End Sub
```

Analyse :

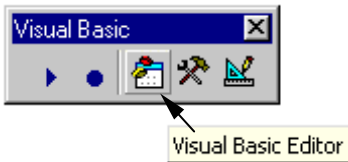
La première ligne définit une *procédure* appelée *Bonjour*. Une procédure Visual Basic est un groupe d'instructions reliées logiquement entre elles, auquel est attribué un nom. Une procédure se compose d'un en-tête (première ligne terminée par la parenthèse fermante) et d'un corps terminé par la ligne *End Sub*. Le corps contient les instructions à exécuter.

Les parenthèses situées à la suite du nom de la procédure contiennent la liste des données soumises à la procédure. On les appelle "paramètres" ou "arguments".

Ci-dessus, la procédure *Bonjour()* s'exécute sans paramètres et les parenthèses restent vides.

Le corps de la procédure *Bonjour()* ne comporte qu'une seule instruction. C'est un appel à la *fonction prédéfinie MsgBox*, avec comme argument, la chaîne de caractères constante *"Bonjour tout le monde !"*. La fonction *MsgBox* est chargée d'afficher à l'écran des messages dans une boîte de dialogue.

- **Créer un nouveau répertoire qui porte votre nom dans D:\Deug TAL** (Fichier -> Nouveau -> Dossier...).

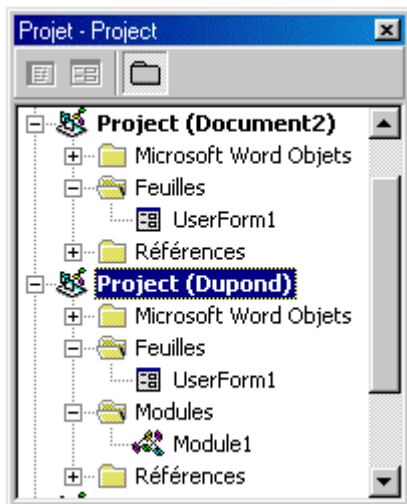



- **Exécuter Microsoft Word et créer un nouveau document Word dans votre répertoire de travail.**
- **Afficher la barre d'outils Visual Basic** (Affichage -> Barre d'outils -> Visual Basic).

- **Accéder à Visual Basic Editor.**

Dans la partie gauche de la fenêtre, **sélectionner** le projet qui porte le nom du document Word que vous venez de créer :

- **Insérer un nouveau module dans votre projet VB** (Insertion -> Module).



- **Sélectionner le module inséré. Saisir le code indiqué sur la 1^{ère} page dans la fenêtre de droite.**
- **Pour l'instant, le programme Visual Basic saisi n'est qu'un texte placé sur le disque. Exécuter le programme en appuyant sur**  **.**

Notion d'une variable

Les constantes ne suffisent pas à assurer le fonctionnement d'un programme. Par exemple, une application chargée de compter le nombre de mots d'un texte devra incrémenter un compteur après chaque mot. Cette opération nécessite une

donnée susceptible de prendre plusieurs valeurs successives, autrement dit une *variable*. Une variable est donc un emplacement de mémoire, d'un ou de plusieurs octets, qui permet de stocker ou de modifier des valeurs.

Une variable possède toujours un *type*, qui fixe la manière dont son contenu est à interpréter. Le type d'une donnée définit la taille de la mémoire occupée et la forme de sa mémorisation. Pour que la variable concernée soit localisable à tout moment dans la mémoire, elle possède une *adresse*. Chacune des cellules d'un octet qui forme la mémoire centrale de l'ordinateur porte un numéro. L'adresse d'une variable est le numéro de la première cellule occupée par la variable. Une variable porte aussi un nom permettant d'y accéder. Un nom de variable peut contenir toutes les lettres de l'alphabet, les chiffres, la lettre "_". Ainsi, *toto*, *grand_toto*, *toto1* sont des noms de variables correctes ; *!toto*, *grand-toto* sont des noms incorrects.

Dans l'exemple suivant *a\$* est une variable de type *string*. C'est un type de donnée composé d'une séquence de caractères contigus interprétés en tant que caractères et non en tant que valeurs numériques. Une donnée de type string peut inclure lettres, nombres, espace et signes de ponctuation. Dans Visual Basic, le signe dollar (\$) est le caractère de déclaration du type *string*.

Exemple 2 :

```
Sub Bonjour()  
    a$ = InputBox$("Entrez votre nom", "Bonjour", "Dupond")  
    MsgBox ("Bonjour" + Space(1) + a$)  
End Sub
```

Analyse :

L'utilisateur fournit une information qu'il faut stocker dans une variable afin de pouvoir la réutiliser autant de fois que l'on veut. On utilise une fonction prédéfinie **InputBox\$** qui retourne une valeur que l'on peut affecter à une variable déclarée.

Structures conditionnelles

Les instructions à exécuter peuvent dépendre d'une condition. Dans ce cas, il faut utiliser une *structure décisionnelle* qui oriente le déroulement du programme vers des blocs d'instructions déterminés.

L'instruction **if** ne permet l'exécution d'une ou de plusieurs instructions que si la vérification d'une ou de plusieurs conditions préalablement énoncées a permis de constater que cette dernière était remplie.

```
If condition Then  
    Instructions  
Else  
    Instructions  
End If
```

```
If b > 5 Then  
    a$ = 0  
    b$ = 3  
Else  
    a$ = 1  
    b$ = 2  
End If
```

Boucle en nombre défini

Cette boucle est utilisée si l'on connaît à l'avance le nombre de fois qu'elle sera parcourue. L'exécution d'une boucle **for** commence par l'attribution des valeurs initiales à la variable (ou aux variables) utilisée(s) dans la boucle. Cette étape n'a lieu qu'une fois au début de l'instruction **for**. Si la condition de la boucle est remplie (**i = 1 To 6**), les instructions dans le corps de la boucle sont exécutées. Dès que la partie exécution est terminée, une actualisation ou une modification des valeurs des variables de contrôle de la boucle et d'autres variables a lieu dans la partie **Instructions**, suivie d'une nouvelle analyse de la condition de boucle, etc. En revanche, si le résultat de l'analyse est égal à 0, la condition de boucle est considérée comme étant FALSE est la boucle se termine sans autre exécution du corps de la boucle. La variable numérique **Compteur** est incrémentée à chaque fin de boucle.

```
For Compteur = début to fin  
    Instructions  
Next Compteur
```

```
For i = 1 To N  
    Somme = Somme + i  
Next i
```

Travail à faire :

(I) Créer un programme qui permet de réaliser les étapes suivantes :

- inciter l'utilisateur à saisir un nombre N et le stocker dans une variable ;
- calculer la somme des N premiers nombres. *Par exemple :*
le nombre saisi est "8" ($N = 8$) ; la somme des N premiers nombre est calculée de la manière suivante : $Somme = 1+2+3+4+5+6+7+8$ ($SommeN = 36$).
- générer le message suivant "La somme des [*nombre N saisi*] premiers nombres est égale à [*Somme*]".

Utiliser la structure *for ... next* pour effectuer le calcul de la somme.

Voici les fonctions nécessaires à la réalisation de ce programme :

- **MsgBox()** }
• **InputBox()** } cf. pp. 1-2
- **Val()**
n = Val(a\$) 'renvoie la valeur numérique de a\$'
- **Str()**
a\$=Str(Nombre) 'renvoie une chaîne de caractères représentant un nombre.'

(II) Créer un programme qui permet de réaliser les étapes suivantes :

- inciter l'utilisateur à saisir son nom et le stocker dans une variable ;
- lui demander son age et le mémoriser ;
- identifier si l'utilisateur du programme est mineur (<18 ans) ou majeur (18 ans ou >18ans);
- générer les messages suivants "Bienvenu [*nom de l'utilisateur*]." (si l'utilisateur est majeur) et "Vous êtes mineur. Au revoir [*nom de l'utilisateur*]." (si l'utilisateur est mineur).

Comment peut-on modifier ce programme pour que les questions soient posées soit anglais soit en français (en fonction de la langue maternelle de l'utilisateur) ?

Utiliser les structures imbriquées *If... Then... Else... End If* pour détecter l'âge de l'utilisateur ainsi que sa langue maternelle.

Fonctions de manipulation de chaînes de caractère en Visual Basic :

- Les fonctions **Right/Left** permettent de renvoyer un nombre de caractères spécifié en partant de l'extrémité droite/gauche de la chaîne, par exemple :
AnyString\$ = "Bonjour à tous"
MyStr\$ = Right\$(AnyString, 6) 'Renvoie "à tous" '

- La fonction **Len (string | varname)** donne la longueur d'une chaîne. *Exemple :*

```
b$ = "manger"
c$ = Len(b$) - 2           'c$ prend la valeur de quatre'
```

- La fonction **InStr** renvoie une valeur de type indiquant la position de la première occurrence d'une chaîne à l'intérieur d'une autre chaîne. Syntaxe :

```
InStr(string1, string2)    'string 1 – expression de chaîne
                           dans laquelle la recherche est effectuée'
                           'string 2 – expression de chaîne recherchée'

n$ = Instr([Index,] Source$, Rechercher$)
Index           La position du caractère où doit commencer la recherche
Source$         Le texte dans lequel la recherche doit avoir lieu.
Rechercher$    Le texte à chercher
```

La fonction retourne la position du premier caractère du texte à rechercher, ou zéro si le texte Rechercher\$ ne se trouve pas dans le texte Source\$.

Exemple : Pos\$ = Instr("Corbeau", "beau") 'donne à la variable Pos\$ la valeur 4.'

- **Mid\$ ()**

```
n = Mid(Source$, Index [,Nombre])
```

Retourne le nombre de caractères indiqué par *Nombre* à partir de Source\$, en commençant au caractère *Index*. Si *Nombre* n'est pas indiqué, le reste de la chaîne de caractères est retourné.

- **Asc ()**

```
n = Asc (a$)
```

Retourne le code caractère du premier caractère compris dans a\$. *Asc* est une abréviation du jeu de caractères ASCII utilisé dans les premières versions du BASIC et ainsi désigné pour des raisons de compatibilité. Le code retourné est, en réalité, celui que Microsoft Windows utilise ; le code ANSI.'

- **Chr ()**

```
a$ = Chr(CodeCar)          'retourne le caractère dont le code ANSI correspond à CodeCar.'
```

Le tableau suivant liste quelques-uns des caractères spéciaux que l'on peut obtenir grâce à Chr() :

Chr(9)	Tabulation
Chr(11)	Caractère de nouvelle ligne (MAJ + ENTREE)
Chr(13) + Chr(10)	Marque de paragraphe
Chr(34)	Guillemets

Code ASCII

La mémoire de l'ordinateur conserve toutes les données sous forme numérique. Il n'existe pas de méthode pour stocker directement les caractères. Chaque caractère possède donc son équivalent en code numérique: c'est le code ASCII (American Standard Code for Information Interchange - traduisez " Code Américain Standard pour l'Echange d'Informations"). Le code ASCII de base représentait les caractères sur 7 bits (c'est-à-dire 128 caractères possibles, de 0 à 127). Le code ASCII a été mis au point pour la langue anglaise, il ne contient donc pas de caractères accentués, ni de caractères spécifiques à une langue. Pour coder ce type de caractères il faut recourir à un autre code. Le code ASCII a donc été étendu à 8 bits (un octet) pour pouvoir coder plus de caractères (on parle d'ailleurs de code ASCII étendu...). Ce code attribue les valeurs 0 à 255 ([donc codées sur 8 bits, soit 1 octet](#)) aux lettres majuscules et minuscules, aux chiffres, aux marques de ponctuation et aux autres symboles (caractères accentués dans le cas du code *iso-latin1*).

Travail à faire :

Complétez les blancs " _____ "

1)

AnyString\$ = "Bonjour à tous"
MyStr\$ = Left\$(AnyString\$, 7)

'Renvoie " _____ " '

2)

AnyString\$ = "Bonjour à tous"
MyStr\$ = Right\$(AnyString\$, 6)

'Renvoie " _____ " '

3)

MyString\$ = "Démon Fonction Mid"
FirstWord\$ = Mid\$(MyString\$, 1, 4)
LastWord\$ = Mid\$(MyString\$, 6, 8)
MidWords\$ = Mid\$(MyString\$, 6)

"Renvoie " _____ " "

"Renvoie " _____ " "

"Renvoie " _____ " "

4)

Pos = Instr("Corbeau", "beau")

"Renvoie " _____ " "

5)

b\$ = "manger"
c\$ = Len(b\$) - 2

"Renvoie " _____ " "

Structure de données : tableau

Un tableau (parfois aussi appelé *vecteur*) est une zone continue en mémoire dont les cellules d'un octet peuvent être rassemblées en unités plus grandes, identiques et logiques. Ces unités sont les éléments du tableau. La taille d'un élément de tableau (en octets) dépend du type des objets destinés à y être stockés. Par exemple, pour conjuguer un verbe du premier groupe on fait un tableau avec 6 pronoms personnels :

je	e
tu	es
il	e
nous	ons
vous	ez
ils	ent

déclaration informatique

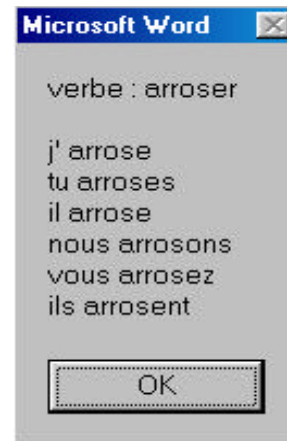
```
Dim Pp$(6)
Dim Tb$(6)
REM : le nombre de valeurs de type identique est à
déclarer entre parenthèses
Pp$(1) = "je"
Pp$(2) = "tu"
Pp$(3) = "il"
Pp$(4) = "nous"
Pp$(5) = "vous"
Pp$(6) = "ils"
Tb$(1) = "e"
Tb$(2) = "es"
Tb$(3) = "e"
Tb$(4) = "ons"
Tb$(5) = "ez"
Tb$(6) = "ent"
```

Une instruction *Dim* permet de déclarer un tableau.

Travail à faire :

1. **Créer** un programme qui permet de conjuguer un verbe du premier groupe. Le programme doit pouvoir reconnaître le radical du verbe pour faire la conjugaison : aimer = aim + er (a\$ = aimer ; radical\$ = aim).

Lorsque vous appuyez sur le bouton OK, après avoir introduit l'infinitif du verbe *arroser* (par exemple), le programme affichera la réponse suivante :



2. Essayer de tenir compte des verbes qui présentent quelques exceptions [ex. manger -> nous mangEons].

Voici les fonctions nécessaires à la réalisation de ce programme :

- **MsgBox** ()
 - **InputBox** ()
 - **InStr** ()
 - **Len** ()
 - **Left** ()
- Pour réserver de la place pour un tableau on utilise la déclaration Dim Tab(6)
 - Pour effectuer la conjugaison, la syntaxe de la boucle est :

```
For i=1 to N
    instructions
next i
```