Université Lumière – Lyon 2, Faculté de Sciences Économiques et de Gestion

Informatique – Référence du langage Visual Basic
J. Darmont (*http://eric.univ-lyon2.fr/~jdarmont/*), 16/07/2003

1. Stockage des données

Déclarations :

- Variable : `Dim nom_var as type`
- Constante : `Const nom_const = valeur_const`

Types de données :

- Entiers :
  - `Byte` [0, 255]
  - `Integer` [-32768, +32767]
  - `Long` [-2147483648, +2147483647]
- Réels :
  - `Single` [$-3{,}402.10^{38}$, $+3{,}402.10^{38}$]
  - `Double` [$-1{,}797.10^{308}$, $+1{,}797.10^{308}$]
  - `Currency` [-922337203685477,5808, +922337203685477,5807]
- Chaîne de caractères : `String` (jusqu'à environ 2 milliards de caractères)
- Date : `Date` [01/01/100, 31/12/9999]
- Booléen : `Boolean` (True ou False)
- Contrôle : `Object` (adresse mémoire du contrôle)
- Variable : `Variant` (le type change selon le contexte — à utiliser avec modération).

Tableau : `Dim nom_tab(taille) as type`

Type structuré défini par l'utilisateur :

```
Type nom_type
      champ1 as type
      champ2 as type
      …
End Type
```

Ex. Type structuré Personne constitué de Nom (chaîne), Prénom (chaîne), Âge (entier)…

Exemples de déclarations :

```
Const PI=3.1416
Dim i as Integer
Dim ch1, ch2, ch3 as String
Dim tab_de_reels(100) as Single
Dim matrice(10,10) as Double
Type Personne
      nom as String
      prenom as String
      age as Byte
End Type
Dim une_personne as Personne
```


2. Éléments de base du langage

Affectation : `=`

Ex.
```
i = 1
ch1 = "Coucou !"
tab_de_reels(5) = 3.14
matrice(i,4) = PI*2
une_personne.prenom = "Jérôme"
```

Commentaire : `'`

Ex.
```
' Ceci est une ligne de commentaire
```

Conversions de type :

Fonction / Convertit une expression en

`CByte`	/ Byte	`CInt`	/ Integer
`CLng`	/ Long	`CSng`	/ Single
`CDbl`	/ Double	`CCur`	/ Currency
`CStr`	/ String	`CDate`	/ Date
`CBool`	/ Boolean	`CVar`	/ Variant

Ex.
```
i = CInt(3.14) ' i=3
msg_affiche = CStr(i)
```

Opérateurs de comparaison :

Égal : `=`	Différent : `<>`
Inférieur : `<`	Inférieur ou égal : `<=`
Supérieur : `>`	Supérieur ou égal : `>=`

Opérateurs logiques :

- Ou : `or`. La condition (*cond1* `or` *cond2*) est vraie si *cond1* est vraie ou *cond2* est vraie.
- Et : `and`. La condition (*cond1* `and` *cond2*) est vraie si *cond1* et *cond2* sont vraies.
- Non : `not`. La condition (`not` *cond1*) est vraie si *cond1* est fausse.

Composition des opérateurs logiques :

not(*cond1* **or** *cond2*) ⇔ (**not** *cond1*) **and** (**not** *cond2*)

not(*cond1* **and** *cond2*) ⇔ (**not** *cond1*) **or** (**not** *cond2*)

Ex.
```
not(a=1 or b>10)
⇔ (not(a=1)) and (not(b>10))
⇔ (a<>1) and (b<=10)
```

3. Structures de contrôle

Test :
```
If condition then
      ' Instructions si condition vraie
Else
      ' Instructions si condition fausse
End If
```

Ex. Calcul du maximum entre deux nombres
```
If n1<n2 then
      max = n2
Else
      max = n1
End If
```

NB : La clause `Else` est optionnelle dans le cas général.

Sélection :

```
Select case variable
      Case valeur
            ' Instructions
            …
      Case Else
            ' Traitement par défaut (optionnel)
End Select
```

Ex.

```
Select case num_mois
  Case 1
    nom_mois = "janvier"
  Case 3, 4, 5
    saison = "printemps"
  Case Else
    message = "cas non prévu"
End Select
```

Test « court » : `Iif condition, si_vrai, si_faux`

Ex. Calcul du maximum entre deux nombres

```
Iif n1<n2, max = n2, max = n1
```

Choix d'index : `var = Choose (index,expr1,expr2…)`

Ex. Calcul de taux

```
taux = Choose (i, 0.05, 0.07, 0.1, 0.15)
```

Si i=1 alors taux= 5 % Si i=2 alors taux= 7 %

Si i=3 alors taux=10 % Si i=4 alors taux=15 %

Boucle « pour » :

```
For var=min to max step pas
      ' Instructions
Next var
```

Ex. Calcul de factorielle (10! = 1*2*3*4*5*6*7*8*9*10)

```
fact = 1
For i = 1 to 10   ' step optionnel si 1
      fact = fact*i
Next i
```

Ex. Initialisation à rebours d'un tableau

```
For i = 100 to 1 step -1
      tab_de_reels(i) = i*3.14
Next i
```

Boucle « pour chaque élément de tableau » :

```
For each var in tableau
      ' Instructions
Next var
```

Ex.

```
Dim nom_mois(12) as String
Dim mois as String
nom_mois(1) = "janvier"
…
For each mois in nom_mois
      ' Traitement
Next
```

Boucle « tant que » :

```
While condition
      ' Instructions
Wend
```


Ex. Arrêt du calcul de factorielle quand le résultat dépasse 100

```
fact = 1
i = 1
While (i<=10) and (fact<100)
  fact = fact*i
  i = i+1
Wend
```

NB : Le test sur la condition étant placé en début de boucle, on peut ne pas « entrer » dans la boucle.

Boucle « répéter jusqu'à » :

```
Do
      ' Instructions
Loop until condition
```

Ex. Arrêt du calcul de factorielle quand le résultat dépasse 100

```
fact = 1
i = 1
Do
  fact = fact*i
  i = i+1
Loop until (i>10) or (fact>=100)
```

NB : Le test sur la condition étant placé en fin de boucle, les instructions sont au moins exécutées une fois.

Comment choisir la « bonne » boucle ?

	Nombre d'itérations connu	Nombre d'itérations inconnu
La boucle doit être exécutée au moins une fois	FOR NEXT	DO LOOP UNTIL
La boucle peut ne pas être exécutée	×	WHILE WEND

Structures imbriquées

Ex. 1. Condition « complexe » pour calculer une remise

```
If bon_client then                  ' ⇔ bon_client=true
      If montant>1000 then
            taux_remise = 0.1       ' 10 %
      Else
            taux_remise = 0.05      ' 5 %
      End If
Else                                ' bon_client=false
      If montant>2000 then
            taux_remise = 0.05      ' 5 %
      Else
            taux_remise = 0.025     ' 2.5 %
      End If
End If
```

Ex. 2. Initialisation d'une matrice

```
For i = 1 to 10
      For j = 1 to 10
            matrice(i,j) = i*j
      Next j
Next i
```

Ex. 3. Comptage d'une valeur dans un tableau

```
c = 0
For i = 1 to 100
      If tab_de_reels(i) = 3.14 then c = c+1
Next i
```

4. Sous-programmes

Définition : Programme autonome dédié à une tâche précise, de préférence de taille réduite. Un sous-programme peut recevoir des paramètres, ou arguments.

Deux types de sous-programmes :
- *Procédures* : Sous-programmes proprement dits
- *Fonctions* : Sous-programmes qui renvoient un résultat
  Ex. Fonction de calcul du sinus d'un angle

Structure d'une procédure :
```
Public|Private Sub nom_proc (paramètres)
      ' Déclarations
      ' Instructions
End Sub
```

Structure d'une fonction :
```
Public|Private Function nom_fn (paramètres) as type_val_retour
      ' Déclarations
      ' Instructions
      nom_fn = val_retour
End Function
```

NB : *Private :* Appel au sous-programme possible uniquement depuis la même feuille ou le même module / *Public :* depuis toute l'application.

Définition des paramètres : Séparés par des virgules, en spécifiant le type de chacun
Ex. `poct as Byte, pch as String, preel as Single`

Modes de passage :
- *Par valeur* (`ByVal`) : Si le paramètre est une variable existante, son contenu est recopié. Il ne sera pas modifié en sortie de sous-programme.
- *Par référence* (`ByRef`) : Si le paramètre est une variable existante, son adresse est utilisée. Il pourra être modifié en sortie de sous-programme. Par défaut (si l'on n'indique rien), les paramètres sont passés par référence.

Ex. `ByVal poct as Byte, ByRef preel as Single`

Exemple de fonction : Calcul de maximum
```
Public Function Max(ByVal n1 as Single, ByVal n2 as Single) as Single
      If n1<n2 then
            Max = n2 ' Valeur de retour : n2
      Else
            Max = n1 ' Valeur de retour : n1
      End if
End Function
```

Exemple de procédure : Échange de deux variables
```
Public Sub Swap(v1 as String, v2 as String)
' v1 et v2 sont passées par référence
      Dim temp as String ' Variable temporaire
      temp = v1
      v1 = v2
      v2 = temp
End Sub
```


Appel de fonction depuis un autre sous-programme **:**
Ex. `un_reel = Max (var1, 32767)`

Appel de procédure depuis un autre sous-programme **:**
Ex. 1 : `Call Swap (chaine1, chaine2)`
Ex. 2 : `Swap chaine1, chaine2`

5. Portée des variables

Variable déclarée dans un sous-programme ⇒ **variable locale au sous-programme** accessible uniquement dans ce sous-programmeInitialisée à chaque appel
- Détruite en fin de sous programme
- Sauf si déclarée en temps que **variable statique** (valeur conservée entre deux appels au sous-programme)
  Ex. `Static compteur as Integer`

Variable déclarée dans la section *Général / Déclarations* d'une feuille ou d'un module ⇒ **variable locale à la feuille / au module** accessible par tous les sous-programmes de la feuille / du module, mais pas à l'extérieur

Variable déclarée publique dans la section *Général / Déclarations* d'un module ⇒ **variable globale** accessible par toutes les feuilles et tous les modules de l'application
Ex. `Public Score as Integer`

6. Éléments visuels

Contrôle : Objet prédéfini (type `Object`) au sens de la programmation orientée-objet
- *Propriétés :* Définition de son aspect
- *Événements :* Définition de son comportement

Contrôles courants :
- Étiquette (*Label*)
- Zone de Texte (*TextBox*)
- Cadre (*Frame*)
- Bouton de commande (*CommandButton*)
- Case à cocher (*CheckBox*)
- Bouton d'option (*OptionButton*)
- Zone de liste (*ListBox*)
- Zone de liste modifiable (*ComboBox*)

Quelques propriétés communes :
- `Name` : Nom interne de l'objet qui peut être utilisé dans un programme
- *Appearance* : Apparence du contrôle (3D ou plat)
- *BackColor* : Couleur du fond
- *Caption* : Texte affiché à l'écran (sauf pour les champs de saisie ⇒ *Text*)
- *Enabled* : Contrôle activé ou non (True/False)
- *Font* : Police de caractères
- *ForeColor* : Couleur de l'écriture
- *TabIndex* : Ordre d'accès avec la touche TAB

- *Visible* : Contrôle visible ou non (True/False)

Quelques événements communs :*Change* : Modification du contenu du contrôle
- *Click* : Clic de souris sur le contrôle
- *DblClick* : Double clic de souris sur le contrôle
- *Drag…* : Glisser/déposer
- *GotFocus* : Gain du focus
- *Key…* : Événements provenant du clavier*LostFocus* : Perte du focus
- *Mouse…* : Événements provenant de la souris autres que le clic

Manipulation de contrôle par programme :
Ex. Procédure générique d'affichage d'un texte dans une étiquette (*label*).

```
Public Sub Etiq_Affiche (etiq as Object, ByVal msg as String)
      etiq.Caption = msg
End Sub
```

Ex. d'appels : `Etiq_Affiche label1, "coucou"`
`Etiq_Affiche label2, message`