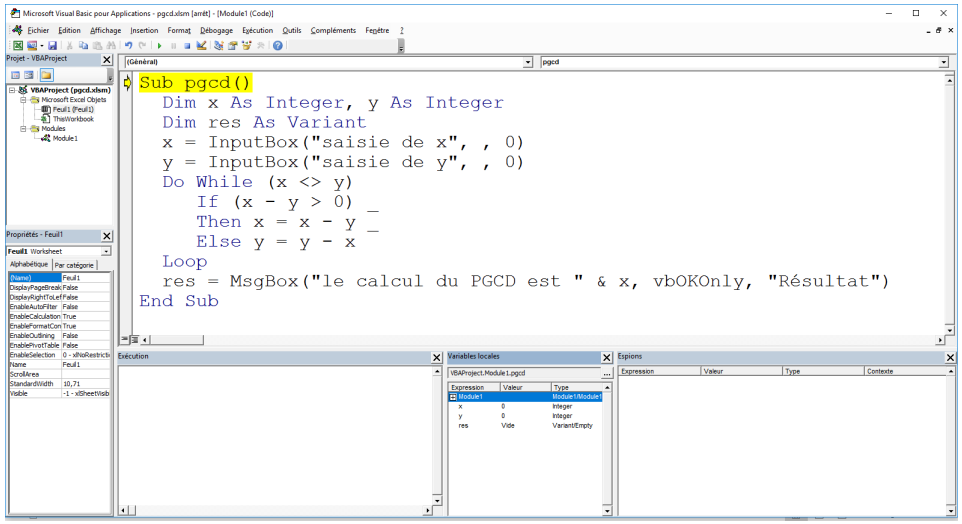
Algorithmique et Langage Visual Basic Application (VBA)

Sommaire



1 Programmation en Visual Basic Application

Le langage de programmation VBA permet d'utiliser du code Visual Basic (VB) pour utiliser les fonctionnalités d'Excel.

Il existe deux modalités de programmation en VBA :

1. **Programmation automatisée** en utilisant l'enregistreur de macro.
2. **Programmation manuelle** en utilisant l'environnement VBE.

Ce cours portera sur la programmation manuelle.

Pour accéder à l'environnement VBE (Visual Basic Editor), nous devons ouvrir Excel, puis cliquer sur la commande Visual Basic de l'onglet "développeur" :

Cet environnement est composé de plusieurs fenêtres :

- **Fenêtre "VBA project"** : elle regroupe de manière hiérarchisée les éléments du classeur (feuilles Excel), les feuilles de dialogue, les modules dans lesquels sont définis les procédures et fonctions.
- **Fenêtre propriétés** : elle présente les propriétés des objets de la fenêtre VBA project, lorsqu'ils sont activés.
- **Fenêtre de code** : elle permet d'afficher le code associé à chacun des objets du projet. Ce code a été créé soit de manière automatisée ou manuelle, il peut à tout moment être modifié et exécuté.
- **Fenêtre d'exécution** : Elle s'active au moment de l'exécution du code d'un objet. Elle permet d'exécuter directement des commandes. Cette fenêtre est très utile lors du débogage.
- **Fenêtre variables locales** : Elle s'active lors de l'exécution du code de l'objet et permet de suivre l'évolution de l'état des variables locales.

- **Fenêtre d'espion** : Elle permet de positionner des points d'inspection lors de l'exécution du programme. Cette fenêtre est très utilisée lors du débogage du programme.

Vous pourrez trouver l'ensemble des éléments du langage VBA en consultant l'aide dans l'environnement VBE (touche F1).

Vous pouvez également consulter le site web suivant :
https://docs.microsoft.com/fr-fr/office/vba/language/

1.1 Les variables

Une variable est un emplacement de stockage d'informations qui porte un nom et une taille et qui peut être modifié lors de l'exécution du programme.

Attention, une variable doit posséder un identifiant unique dans le domaine de sa portée (la notion de portée sera explicitée ultérieurement).

1.1.1 Déclaration de la variable

La déclaration de la variable se fait grâce au mot réservé **Dim**.

Cette déclaration doit se faire prioritairement après l'entête de la fonction ou de la procédure, bien qu'elle soit permise à n'importe quel endroit du code.

```
Dim nom_variable
```

nom_variable est une suite de lettres, chiffres et caractères soulignés.
Le premier caractère est obligatoirement une lettre.

1.1.2 Les types de variables

La plupart des langages de programmation imposent de déterminer le type de la donnée qui sera stockée dans la variable lors de la déclaration.

En VBA, le programmeur peut décider de l'omettre, dans ce cas la variable sera de type **Variant** ce qui permettra de stocker n'importe quel type de données.

La déclaration de type se fait de la manière suivante :

```
Dim nom_variable As Type
```

Voici quelques exemples de types de variable :

Type	Contenu	Plage de valeurs
Date	Informations de date et heure	1^{er} janvier 100 au 31 décembre 9999 Taille en octets : 8
Double	Nombre en virgule flottante (double précision)	$\pm\ 5\times10^{-324}$ à $1{,}8\times10^{308}$ Taille en octets : 8
Integer	Entiers	$-$ 32 768 à 32 767 Taille en octets : 2
Long	Entiers	$\pm\ 2\times10^{9}$ Taille en octets : 4
Object	N'importe quelle référence d'objet	
Single	Nombre en virgule flottante (simple précision)	$\pm\ 1{,}4\times10^{-45}$ à $3{,}4\times10^{38}$ Taille en octets : 4
String	Texte	Long. fixe : 1 à 65 400 Long. variable : 0 à 2×10^{9}
Variant	N'importe quels types précédents	Nombres (voir Double) Caractères (voir texte à longueurs variables)

type doit être l'une des catégorie suivantes : boolean, entier (comme byte, integer, long), réels (comme currency, single, double), date, object, string, variant. De plus, l'utilisateur peut définir des types complexes avec le mot réservé ***type***.

La définition d'un nouveau type "personne" se fait de la manière suivante :

```
Private type personne
    nom As String
    prenom As String
    age As Integer
End Type

Dim toto As personne
```

1.2 Les opérateurs

Le Visual Basic comme tous les langages de programmation possède de nombreux opérateurs permettant de construire des expressions.

Nous rassemblons dans le tableau suivant les principaux opérateurs que nous

Arithmétique	Comparaison	Logique
Élévation à une puissance (^)	Égalité (=)	**Not**
Négation (−)	Inégalité (<>)	**And**
Multiplication et division (*,/)	Infériorité (<)	**Or**
Division d'entiers (\)	Supériorité (>)	**Xor**
Modulo arithmétique (**Mod**)	Infériorité ou égalité (<=)	**Eqv**
Addition et soustraction (+, −)	Supériorité ou égalité (>=)	**Imp**
Concaténation de chaînes (&)	**Like**	

1.3 Les structures de sélection

Nous retrouvons en VBA les structures de sélection que nous avons mises en évidence en algorithmique.

1.3.1 Structures de sélection simple

Cette structure de sélection peut prendre deux formes.

If (condition) **Then** <instructions> **Else** <instructions>

If (condition) **Then**
    <instructions>
    ⋮
Elseif (condition_1) **Then**
    <instructions>
    ⋮
Elseif (condition_n) **Then**
    <instructions>
    ⋮
Else
    <instructions>
    ⋮
End if

Comme vous pouvez le constater, il est possible d'exécuter plusieurs instructions après les mots réservés **Then** et **Else**. Dans le cas d'une seule ligne, un bloc d'instructions se définit par des instructions séparées entre elles par " : ".

1.3.2 Structures de sélection multiple

La sélection multiple est obtenue par la mise en œuvre de l'instruction **Select Case**, comme suit :

Select Case (expression numérique)
Case (valeur, ou ensemble de valeurs, ou plage de valeurs) <instructions>
    ⋮
Case (valeur, ou ensemble de valeurs, ou plage de valeurs) <instructions>
Case Else <instructions>
End Select

L'expression numérique doit renvoyer une valeur entière, ou alphabétique.

Si l'expression numérique permet de satisfaire plusieurs clauses (**Case**), seule, la première clause rencontrée sera exécutée.

1.4 Les structures itératives

1.4.1 Structures itératives à bornes fixes

La structure **Pour .... Fin Pour** peut être représentée par la structure suivante :

For variable = valeur de départ **To** valeur de fin [**Step** valeur de l'incrément]
    <instructions>
    ⋮
Next variable

Il est conseillé de donner à la variable un type entier ou chaine de caractère.

La valeur de **Step** est optionnelle, par défaut elle vaut 1.

1.4.2 Structures itératives conditionnelles

La structure **Tant que .... Fin Tant que** peut être représentée par les structures suivantes :

While (condition logique)
    <instructions>
    ⋮
Wend

```
Do While (condition logique)
    <instructions>
        :
Loop
```

Ces deux structures s'exécutent tant que la condition est vraie.

La structure **Répéter .... Fin Répéter** peut être représentée par les structures suivantes :

```
Do
    <instructions>
        :
Loop Until (condition logique)
```

Cette structure s'exécute jusqu'à ce que la condition devienne vraie.

1.5 Les commentaires

Les commentaires sont mis en place par le développeur afin d'apporter de la lisibilité à son code, faciliter la maintenance.

Les commentaires ne sont pas interprétés par VBA, donc leur forme n'est pas soumise à une syntaxe particulière.

Pour mettre un commentaire dans le code, il faut commencer la ligne par le mot réservé **Rem** ou par le caractère réservé guillemet simple : (').

1.6 Les données de type tableau

Les tableaux sont des zones de stockage structurées dans lesquelles sont rangées des collections de données d'un même type.

L'accès aux données de la collection se fait par un ou plusieurs index.

1.6.1 Déclaration d'un tableau

Il existe deux procédures de déclaration de tableau :

```
Dim liste_ref(350) As String
Dim liste_ref(1 to 350) As String
Dim liste_ref() As String
```

Pour les deux premiers exemples, la déclaration du tableau `liste_ref` précise que nous réservons 350 emplacements pour stocker des chaines de caractères.

Dans la première déclaration l'index d'accès aux données varie entre [0 et 349], dans le second cas nous précisons la plage de variation entre [1 et 350].

Pour la troisième déclaration, la réservation de place n'est pas faite, il faudra qu'ultérieurement une instruction **ReDim** précise la dimension du tableau, comme dans l'exemple suivant :

```
ReDim liste_ref(350)
```

Il est possible de redimensionner le tableau plusieurs fois, mais chaque redimensionnement provoque l'effacement de l'ensemble des données du tableau.

1.6.2 Initialisation des valeurs du tableau

Deux méthodes permettent d'affecter des valeurs à un tableau. Si la déclaration a été faite par l'instruction **Dim** comme ceci :

```
Dim val(2) As Integer
val(0) = 10
val(1) = 0
```

1.6.3 Modification et accès aux valeurs d'un tableau

La modification d'un élément du tableau s'effectue simplement en affectant (avec l'opérateur =) une nouvelle valeur, comme ceci :

```
val(1) =100
```

L'accès aux valeurs du tableau par appel direct de l'élément du tableau comme dans l'exemple suivant :

```
x = val(1)
```

1.7 Les Objets en VBA

1.7.1 La notion d'bjet en informatique

Un objet (que vous étudierez davantage en S3) est un ensemble d'instructions et de données, il se comporte comme une entité unique et cohérente.

Un objet possède :

- **des propriétés** qui définissent son état courant,
- **des méthodes** qui sont ses fonctionnalités,
- **un type** issu de la classe abstraite qui le définit.

1.7.2 La manipulation des objets

La manipulation des objets se résume en deux types d'actions : l'une permet de créer ou de détruire l'objet, la seconde permet d'accéder à ses propriétés ou à ses méthodes.

La création et destruction d'un objet :

Il existe différentes manières pour créer un objet (en jargon informatique nous disons instancier). Nous ne verrons que celle qui nous sera utile dans le contexte de ce cours.

La création d'un objet se fait grâce au mot réservé **New**, sa destruction se fait par le mot **Nothing** comme dans l'exemple suivant :

```
Sub main()
    Dim rep As Integer

    ' Création d'un objet du type my_obj
    Dim mon_objet As my_obj

    ' Création de l'instance de l'objet
    ' (c.a.d réservation de l'espace mémoire)
    Set mon_objet = New my_obj

    ' Mise à jour du contenu de la propriété message de l'objet
    mon_objet.message = "coucou"

    ' Exécution de la méthode envoi_message
    mon_objet.envoi_message

    ' Exécution de la méthode longueur_message
    rep = mon_objet.longueur_message

    ' Destruction de l'objet
    Set mon_objet = Nothing' détruit l'instance

End Sub
```

Manipulation des membres d'un objet

Nous entendons par manipulation des membres l'accès aux propriétés et méthodes d'un objet. L'accès aux membres se fait avec l'opérateur point (.).

Dans l'exemple précédent, l'objet `mon_objet` possède une propriété et deux méthodes de sa classe `my_obj`. Pour accéder à la propriété comme aux méthodes, il suffit de mettre le nom de l'objet puis celui de la propriété ou des méthodes séparés par un point (voir le paragraphe précédent).

Dans le monde d'Excel, les choses sont plus complexes de par la nature des objets manipulés. En général les membres des objets sont eux-mêmes des objets, possédant des membres qui sont à leur tour des objets ...

L'accès au membre désiré peut être compliqué car il nécessite de connaitre la structure des objets Excel.

Par exemple, si nous voulons lire le contenu de la cellule A1 de la feuille "Feuil1", il faudrait faire :

```
Sub main()
    Dim My_val As Variant

    My_val=ThisWorkbook.Worksheets("Feuil1").Cells(1, 1)
End Sub
```

1.7.3 L'instruction With ... End With

Cette instruction permet de définir un bloc dans lequel un objet sera référencé une seule fois de manière à simplifier l'écriture du code, comme le montre l'exemple suivant :

```
Sub main ()
    Dim My_val_1 As Variant, My_val_2 As Variant
    Dim My_val_3 As Variant
    With ThisWorkbook.Worksheets("Feuil1")
        My_val_1= .Cells(1, 1)
        My_val_2= .Cells(1, 2)
        My_val_3= .Cells(2, 1)
    End With
End Sub
```

Pour réaliser la même fonction, nous pouvons également créer une variable **My_feuil**.

```
Sub main ()
    Dim My_val_1 As Variant, My_val_2 As Variant
    Dim My_val_3 As Variant
    Dim My_feuil As Worksheet
    Set My_feuil = ThisWorkbook.Worksheets("Feuil1")
    My_val_1 = My_feuil.Cells(1, 1)
    My_val_2 = My_feuil.Cells(1, 2)
    My_val_3 = My_feuil.Cells(2, 1)
End Sub
```

1.7.4 Les Objets Cellules

Une cellule est un objet dont le nom est **Range**.

Les principales propriétés concernant les cellules sont :

- Sélectionner la cellule A1 : `Range("A1").Select`
- Sélectionner les cellules A1 à B3 : `Range("A1:B3").Select`
- Cellules sélectionnées : `Range("A1:B3").Select`
- Cellule active : `ActiveCell`
- Saisir une donnée dans la cellule A1 :
  `Range("A1").Value = "Texte à saisir"`
- Copier la cellule A1 en A2 : `Range("A1").Copy Range("A2")`
- Couper la cellule A1 en A2 : `Range("A1").Cut Range("A2")`
- Effacer la cellule A1 : `Range("A1").Clear`
- Effacer la cellule A1 en conservant la mise en forme :
  `Range("A1").ClearContents`

Les propriétés concernant les formats des cellules sont :

- Gras : `Range("A1").Font.Bold = True`
- Italique : `Range("A1").Font.Italic = True`
- Souligné : `Range("A1").Font.Underline = True`
- Police de caractère :
  `Range("A1").Font.Name = "Times New Roman"`
- Taille de la police : `Range("A1").Font.Size = 14`
- Couleur de la police : `Range("A1").Font.ColorIndex = 2`
- Couleur de remplissage : `Range("A1").Interior.ColorIndex = 1`
- Centrer horizontalement :
  `Range("A1").HorizontalAlignment = xlCenter`
- Gauche horizontalement :
  `Range("A1").HorizontalAlignment = xlLeft`
- Droite horizontalement :
  `Range("A1").HorizontalAlignment = xlRight`

Les couleurs des index sont :

- Noir : 1
- Blanc : 2
- Rouge : 3
- Vert : 4
- Bleu : 5
- Jaune : 6
- Violet : 7
- Turquoise : 8
- Marron : 9

1.7.5 Les Objets Bordures

Pour affecter l'extérieur d'une cellule, nous faisons appel à la propriété **Borders**.

Pour modifier une bordure, il faut appeler une des constantes de `XlBordersIndex` pour identifier un élément de la bordure : `XlDiagonalDown`, `XlDiagonalUp`, `XlEdgeBottom`, `XlEdgeLeft`, `XlEdgeRight` et `XlEdgeTop`.

Les propriétés de l'objet **Borders** sont :

- `ColorIndex` : couleur de la bordure.
- `LineStyle` : style de ligne `xlContinuous`, `xlDash`, `xlDashDot`, `xlDashDotDot`, `xlDot`, `xlDouble`, `xlLineStyleNone` ou `xlSlantDashDot`.
- `Weight` : épaisseur `xlHairLine`, `xlMedium`, `xlThick`, ou `xlThin`,

1.7.6 Les Objets Lignes et Colonnes

Une colonne est un objet dont le nom est **Columns**.
Voici les principales propriétés concernant les colonnes.

- Sélectionner la colonne A : `Columns("A").Select`
- Sélectionner les colonnes A à C : `Columns("A:C").Select`
- Largeur de la colonne D à 25 : `Columns("D").ColumnWidth = 25`

Une ligne est un objet dont le nom est **Rows**.
Voici les principales propriétés concernant les lignes.

- Sélectionner la ligne 1 : Rows("1").Select
- Sélectionner les lignes 1 à 3 : `Rows("1:3").Select`

- Hauteur de la ligne 5 à 30 : `Rows("5").RowHeight = 30`

1.7.7 Les Objets Classeurs

Le classeur (fichier Excel) est un objet dont le nom est **Workbooks**.
Voici les principales propriétés de l'objet Workbooks.

- Ouvrir un classeur :
  `Workbooks.Open "C:\Chemin\NomDuFichier.xlsm"`
- Activer un classeur : `Workbooks("NomDuFichier.xlsm").Activate`
- Créer un nouveau classeur : `Workbooks.Add`
- Enregistrer un classeur :
  `ActiveWorkbook.SaveAs Filename:="NomDuFichier.xlsm"`
- Fermer un classeur : `Workbooks("NomDuFichier.xlsm").Close`
- Fermer le classeur actif : `ActiveWorkbook.Close`

1.7.8 Les Objets Feuilles

Une feuille de calcul est un objet dont le nom est **Sheets**.

Voici les principales propriétés de l'objet Sheets.

- Sélectionner la feuille 1 : `Sheets("Feuil1").Select`
- Feuille active : `ActiveSheet`
- Ajouter une feuille :
  `Sheets.Add.Name = "Nom de la nouvelle feuille"`

- Renommer une feuille :
  `Sheets("Feuil1").Name = "Nouveau nom de la feuille"`
- Copier la feuille 1 après la feuille 3 :
  `Sheets("Feuil1").Copy After:=Sheets("Feuil3")`

- Déplacer la feuille 1 après la feuille 3 :
  `Sheets("Feuil1").Move After:=Sheets("Feuil3")`

- Masquer la feuille 1 : `Sheets("Feuil1").Visible = False`
- Supprimer la feuille 1 : `Sheets("Feuil1").Delete`

1.7.9 Les Objets Graphiques

Notre objectif est ici de créer un graphique via le code VBA.
Nous avons deux possibilités :

- Créer un graphique sur une nouvelle feuille (feuille de graphique)
- Créer un graphique sur notre feuille de calcul (feuille des données)

```
Const NomGraphique As String = "Graphique"

Dim Graph As Chart
Dim Plage As Range

' Définition de la plage des données source du graphique
' Par exemple A1 à B6
Set Plage = Sheets("Feuil1").Range("A1:B6")

' Suppression du graphique si déjà existant
For Each Graph In Charts
   Graph.Delete
Next Graph

' Création du graphique
Set Graph = Charts.Add
With Graph
    ' Type histogramme
    .ChartType = xlColumnClustered
    ' Source du graphique
    .SetSourceData Source:=Plage, PlotBy:=xlColumns
    ' Affichage du titre
    .HasTitle = True
    ' Intitulé
    .ChartTitle.Characters.Text = Plage.Cells(1, 1)
    ' Nom de la feuille recevant le graphique
    .Name = NomGraphique
End With
```

Il y a plusieurs types de graphiques (`ChartType`) : `xlArea` (Aires), `xBarClustered` (Barres groupées), `xlColumnClustered` (Histogramme groupé), `xlLine` (Ligne), `xlPie` (Secteur) ...