A – Eléments préalables

1 – Environnement de base

Accès à l'Onglet **Développeur** du ruban :

- Obtenu sous Windows par *Fichier / Options / Personnaliser le ruban / onglets principaux / cocher développeur*
- Obtenu sur Mac OS par *Excel / Préférences / Affichage / Dans le Ruban Afficher : Onglet Développeur*

Création d'un module pour créer un programme

Onglet de ruban **Développeur / Visual Basic** puis menu **Insertion / Module**

Exécution avec affichage direct dans la fenêtre d'exécution

Onglet de ruban **Développeur / Visual Basic** puis menu **Affichage / Fenêtre d'exécution**

Sur ordinateur Apple, la fenêtre d'exécution a également le nom de fenêtre **Immédiat**

Conventions de saisie :

Toute ligne débutant par une apostrophe (`'`) ou le mot REM sera un commentaire ignoré à l'exécution

Dans un module, toute instruction doit être encadrée par un `Sub ... End Sub` (pour indiquer qu'elle appartient à une procédure et sera exécutée lors de l'appel de celle-ci) ou par un `Function ... End Function` (pour indiquer qu'elle appartient à une fonction et contribuera au calcul du résultat de celle-ci). En fenêtre d'exécution (ou fenêtre immédiat sur Mac OS), les instructions sont exécutées sitôt validées par la touche Entrée sans nécessité d'encadrement.

2 – Valeurs littérales en VBA

- Valeurs numériques :
 - Utilisation du point comme séparateur de décimales
 - Possibilité de notation Scientifique $3.5e-3 = 3.5 \times 10^{-3} = 0.0035$
- Valeurs textuelles : Encadrées par des guillemets, les éventuels guillemets contenus étant doublés
 - Ex: `"On prête à Jules César la déclaration ""Alea Jacta Est"""`
- Valeur calendaires : En notation anglo-saxonne et entre des # avec un format de type
 - `#MM-JJ-YYYY#`, `#hh:mm:ss#` ou `#MM-JJ-YYYY hh:mm:ss#`
- Valeurs logiques : **True**, **False**

3 – Entrées-sorties en VBA

- Accéder à une cellule ou une plage de cellule : *adresse de la cellule entre crochets*

Exemples : ◦ `[a1]` ◦ `[Feuil1!A1]` ◦ `['[Classeur1 n°1]Feuil1'!$A$1]`

- Afficher un résultat :
 - Ecran surgissant : **MsgBox** *résultat*
Ex: `MsgBox [a1]+10`
 - Affichage des résultats dans la fenêtre exécution :
 - Depuis une *sub* ou une *function* : **Debug.Print** *résultat*
Ex: `Debug.Print [b2]`
Ex: `Debug.Print "Le produit de A1 et A2 est " & [a1]*[a2]`
 - Depuis la fenêtre d'exécution en mode interactif : **? Résultat**
Ex: `? "Le produit de A1 et A2 est " & [a1]*[a2]`
 - Dans la barre d'état : **Application.StatusBar=message**
Ex: `Application.StatusBar= "La cellule A1 vaut " & [a1]`
`Application.StatusBar=false` 'pour effacer le message

- Demander une saisie à l'utilisateur :

InputBox(Message, [Titre], [valeur par défaut]) → *valeur saisie ou valeur par défaut en cas d'appui sur le bouton Annuler.*

Ex: `Debug.print inputBox("Quel est votre nom ?")`
`Debug.print inputBox("Quel est votre age ?", "Enquête", 21)`

B - Les variables élémentaires

Les variables sont utiles pour mémoriser des résultats intermédiaires. Elles associent un espace mémoire à un nom choisi par le programmeur

Les noms de variables ne peuvent contenir d'espaces ou de ponctuation, ils sont formés de lettres, de chiffres et de `_`, et commencent obligatoirement par une lettre ou un `_`.

Déclaration implicite : Tout nouveau nom recevant une valeur entraine automatiquement la déclaration d'une variable de ce nom. La déclaration implicite est interdite si au sommet du module est présente l'option `Explicit`

Déclaration facultative : **Dim** *nom de variable*

Dim *nom de variable* **As type**

Affectation de valeur : *nom de variable* = valeur

Lors d'une affectation de valeur, la nouvelle valeur écrase l'ancienne, laquelle sera disponible pour toute lecture ultérieure.

Toute citation d'un nom de variable en dehors d'une affectation de valeur se traduit par l'obtention de la dernière valeur mise dans une variable.

C - Les variables tableau

Leur particularité est de pouvoir contenir une série de valeurs et non une valeur unique.

Déclaration obligatoire !

Dim <nom de variable>(bornes dim 1, bornes dim 2,...)

Bornes : Soit <indice de départ> **To** <Indice de fin>

 Soit <Taille> équivalent à **0 To Taille**

Accès aux éléments :

<nom de variable>(indice dim 1, indice dim 2,...)

sachant que pour chaque dimension il convient de respecter :

borne inférieure ≥ indice ≥ borne supérieure

Les bornes inférieures et supérieures peuvent être connues à tout moment par appel des fonctions VBA **Lbound**(variable tableau, numéro de dimension avec 1 si omis) pour une borne inférieure et **Ubound**(variable tableau, numéro de dimension avec 1 si omis) pour une borne supérieure.

Tableaux dynamiques

Utilisés lorsque l'on ne connaît pas la taille d'un tableau au moment de la programmation des traitements

Déclaration : **Dim** <nom de variable>()

Dès lors que la taille est connue et à chaque changement de taille :

Redim [Preserve] <nom de variable>(<bornes obtenues de variables>)

Le mot clé facultatif `Preserve` indique que l'on souhaite conserver les éventuelles valeurs du tableau contenues avant redimensionnement.

Tableaux stockés dans un variant

Le type Variant est le type universel pouvant recevoir n'importe quelle valeur, y compris un tableau. Ce type est retenu par défaut lors d'une déclaration **Dim** sans type. Un tableau dans un variant peut être initialisé par la fonction **Array**(énumération de valeurs séparées par des virgules) qui créera un tableau dont le premier élément est à la position 0.

Dim liste as Variant

Liste=Array(1,2,3)

Il est possible d'utiliser ensuite Redim [Preserve] pour redimensionner un tel tableau.

D / Echanges variables VBA ↔ cellules du tableur

2 notations pour accéder en VBA aux cellules du tableur :

1. [*<adresse ou nom tableur>*]
2. **Workbooks**(*Nom du classeur*).**Worksheets**(*Nom de la feuille*).**Range**(*Adresse dans la feuille*)

Sachant que le **Workbooks**(*Nom du classeur*). Peut être omis dans le cas du classeur courant et **Worksheets**(*Nom de la feuille*). peut être omis dans le cas de la feuille courante

La notation avec des [] ne peut recevoir que des adresse fixes (qui ne peuvent changer à l'exécution du programme). En revanche la notation longue peut recevoir des variables contenant un texte ou plus largement toute expression retournant un texte, donc dépendre des conditions d'exécution.

2^e distinction : **Cellule unique ou plage de cellule ?** Une cellule unique correspondra à une seule valeur tandis qu'une plage de cellules correspondra toujours à un tableau à deux dimensions dont la première position dans chacune des deux dimension est 1. Ainsi, le contenu d'une plage de cellule (*plage.value*) transmise à une fonction VBA donne toujours un tableau à 2 dimensions (1 to *NbLignes*, 1 to *NbColonnes*). Une simple ligne aura les dimensions (1 TO 1, 1 to *NbColonnes*) et une simple colonne aura pour dimensions (1 TO *NbLignes*, 1 TO 1)

Application : A partir de la connaissance des points C et D, proposez plusieurs méthodes pour déclarer 5 cashflows progressant de 1000€ à 1400 € associés à des dates de versement tous les 1er du mois à compter du 1er octobre 2024.

E / Portée des variables

Les variables peuvent être déclarées :

a / au sommet d'un module : elles sont partagées par toutes les procédures et fonctions du module (voire d'autres modules).

b / Dans des procédures et fonctions : Elles n'existent que durant l'exécution de la procédure ou fonction qui les déclare. Deux fonctions peuvent déclarer des variables de même nom qui correspondent à des emplacement mémoire séparés. Les variables non déclarées sont considérées déclarées au niveau de la procédure ou de la fonction.

F - Les fonctions et procédures

Définition : Les fonctions et procédures hébergent tous les traitements réalisables en VBA. Les fonctions retournent un résultat tandis que les procédures se contentent de réaliser des actions ou traitements.

a/ Les traitements simples par les opérateurs

Opérateurs arithmétiques : + - * / ^ \ (division entière) **mod** (reste de la division entière). L'opérateur ^ pouvant être également utilisé comme suffixe précisant le type d'une variable, il est recommandé, si l'on veut réaliser une puissance, de le séparer par un espace d'un éventuel nom de variable qui le précéderait.

Opérateurs sur les textes : & + (mise bout à bout)

Opérateurs de comparaison : < > <= >= = <> **LIKE** <motif de comparaison>

Le motif de comparaison qui suit l'opérateur LIKE est un texte (variable ou constante entre guillemets) dans lequel tout caractère précisé doit être trouvé à la même position dans le texte comparé pour que la

comparaison réussisse hormis 2 caractères spéciaux : * qui autorise n'importe quelle suite de caractères (y compris vide) et ? qui autorise n'importe quel caractère mais un et un seul à sa position.

Opérateur logiques (entre 2 conditions) : **And**, **Or**, **Xor** (conditions affichant des résultats différents)

Priorités de calcul : ()

Question : Comment vérifier en VBA qu'une variable nommée *Année* contient une année bissextile ?

Une année bissextile est une année divisible par 4 mais pas divisible par 100 (à l'exception toutefois des années divisibles par 400 qui sont bissextiles).

b/ Les fonctions du langage VBA

Elles sont définies au niveau du langage et prennent en charge les calculs élémentaires.

Elles sont accessibles directement en indiquant leur nom, ou , en cas de recherche, en bénéficiant de la complétion automatique (CTRL+Espace) après saisie de **VBA.** .

Conversions

Conversion entre les types de données. CBool, CByte, CCur, CDate, CDbl, CDec, Clnt, CLng, CLngLng, CLngPtr, CSng, CStr, CVar, CVer, Str, Val, Fix, Int

Test des types de données. IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric, IsObject

Traitement de tableaux

Test d'un tableau. IsArray

Renvoi des limites d'un tableau. Lbound, Ubound

Traitement de tableaux Array, Filter (sous-ensemble d'après une valeur), Join (création de chaîne de caractère à partir d'un tableau), Split (décomposition de chaîne en tableau), Erase

Prise de décision et test

Prise de décisions. Choose, iif, Switch

Traitement de chaînes de caractères (textes)

Comparaison de deux chaînes. StrComp

Conversion de chaînes. StrConv

Mise en forme de valeurs sous forme de texte Format, FormatCurrency, FormatDateTime, FormatNumber, FormatPercent, Hex, Oct

Mise en majuscules, minuscules Lcase, Ucase

Création de chaînes répétant un même caractère. Space, String

Calcul de la longueur d'une chaîne. Len

Manipulation de chaînes. InStr, InStrRev, Left, Ltrim, Mid, Right, RTrim, Trim, Replace, StrReverse, Lset, RSet

Utilisation des codes ASCII et ANSI, ou Unicode Asc, Chr, ChrW

Gestion des erreurs d'exécution

Récupération des messages d'erreur.	Error
Informations sur les erreurs.	Err
Création d'une erreur à partir d'un code	CVErr
Vérification de type.	IsError

Calculs mathématiques

Fonctions trigonométriques.	Atn, Cos, Sin, Tan
Calculs usuels.	Exp, Log, Sqr
Génération de nombres aléatoires.	Randomize, Rnd
Renvoi de la valeur absolue.	Abs
Renvoi du signe d'une expression.	Sgn
Conversions numériques.	Fix, Int, Round

Calculs calendaires et temporels

Renvoi de la date ou de l'heure en cours.	Date, Now, Time
Calculs de date.	DateAdd, DateDiff, DatePart, Year, Month, Day, WeekDay, Hour, Minute, Second, MonthName, WeekDayName
Renvoi d'une date.	DateSerial, DateValue
Renvoi d'une heure.	TimeSerial, TimeValue
Définition de la date ou de l'heure.	Date, Time
Chronométrage d'un traitement.	Timer

Une liste des fonctions VBA conduisant à l'aide en ligne de chacune est disponible à l'adresse :

<https://learn.microsoft.com/fr-fr/office/vba/language/reference/functions-visual-basic-for-applications>

c / Les fonctions tableur

Les fonctions de feuille de calcul disponibles sous Excel s'appellent par :

Application.WorkSheetFunction.nom anglais de la fonction(arguments dans l'ordre de ceux des feuilles de calcul)

Si le nom anglais de la fonction vous est inconnu, vous pouvez l'obtenir en enregistrant une macro dans laquelle vous inscrivez un appel à la fonction au sein d'une formule.

Exemple : Macro enregistrée ayant mémorisé l'insertion dans la cellule active d'une formule faisant appel à la fonction de feuille de calcul DROITEREG

```
Sub Macro1 ()
ActiveCell.FormulaR1C1 = "=LINEST( , , , )"
End Sub
```

Nom anglais de la fonction DROITEREG,
pouvant être appelée en VBA par
Application.WorksheetFunction.LinEst(...)

d / Les fonctions et procédures définies par l'utilisateurLes procédures :

```
Sub nom ([noms des paramètres éventuels séparés par des virgules])
...
[Exit Sub]
...
End Sub
```

Les Fonctions :

```
Function nom ([noms des paramètres éventuels séparés par des virgules])
...
nom = résultat retourné par la fonction
[Exit Function]
...
End Function
```

Les mots clés **Exit Sub** et **Exit Function** déclenchent une fin prématurée de l'exécution, généralement contrôlée par une condition.

Les procédures sont appelées ensuite par la notation :

<Nom de la procédure> <valeur pour le paramètre n°1>, <valeur pour le paramètre n°2>, ...

ou par

<Nom de la procédure> <nom d'un paramètre> :=<valeur de ce paramètre>, <nom d'un autre paramètre> := <valeur de ce paramètre>, ...

Lorsque l'on appelle une fonction la liste des paramètres (passés par position ou par nom) est obligatoirement encadrée par des parenthèses : <nom de la fonction> (<liste des paramètres>)

G – Les instructions de test

Elles permettent des exécutions différenciées selon la valeur d'une condition

- Fonction **iif**(*condition* , *valeur si la condition est vérifiée* , *valeur si la condition n'est pas vérifiée*)

valeurAbsolue=iif(valeur>=0,valeur,-valeur)

- Instruction **If** :

If *condition* **Then** *instruction unique* [**Else** *instruction unique*]

If *condition* **Then**

instruction unique ou instructions multiples à exécuter si la condition est vérifiée.

[**Elseif** *condition subsidiaire*

instruction unique ou instructions multiples à exécuter si la condition subsidiaire est vérifiée]

[**Else**

instruction unique ou instructions multiples à exécuter si la condition n'est pas vérifiée]

Endif

Exemple :

```
If valeur>=0 Then valeurAbsolue=valeur Else valeurAbsolue=-valeur
If valeur>=0 Then
    valeurAbsolue=valeur
Else
    valeurAbsolue=-valeur
Endif
```

Application : Faire une fonction d'actualisation d'un montant M à partir d'un taux r , d'une fréquence de composition des intérêts m et d'une durée n exprimée en nombre de périodes sachant que si la fréquence de composition n'est pas un nombre strictement positif, une actualisation en taux continu doit être conduite.

Si m strictement positif :

$$V_0 = \frac{M}{\left(1 + \frac{r}{m}\right)^{n \times m}}$$

Si m non numérique ou ≤ 0

$$V_0 = M \times e^{-r_c \times n}$$

H – Les instructions itératives (boucles)

a / Boucle générique Do ... Loop :

Cette boucle répète l'exécution d'une liste d'instructions encadrées par les mots **Do** et **Loop** tant qu'une condition est vérifiée (ou jusqu'à ce qu'elle soit vérifiée).

Les instructions de la boucle doivent avoir une influence sur la condition.

La position en amont ou en aval de la condition n'est pas neutre: si la condition est en aval, on a l'assurance d'au moins une exécution des instructions de la boucle, ce qui n'est pas le cas si la condition est en amont.

Do [While | Until condition]

Instructions à répéter pouvant appeler la commande EXIT DO dans un bloc conditionnel en vue d'une condition d'arrêt supplémentaire

Loop [While | Until condition]

Exemple :

```
Function sommeTableauDo (tableau)
    Dim indice
    Dim résultat
    résultat = 0
    indice = LBound(tableau) ' la fonction LBound retourne l'indice du
                            ' premier élément du tableau
    Do While indice <= UBound(tableau) ' la fonction UBound retourne l'indice
                                        ' du dernier élément du tableau
        résultat = résultat + tableau(indice)
        indice = indice + 1
    Loop
    sommeTableauDo = résultat
End Function
```

Application : Soit un montant M emprunté de 100 000 euros dans le cadre d'un emprunt bancaire au taux fixe r de 5%, augmentez le nombre n d'annuités en partant de 1 jusqu'à obtenir une annuité a intérieure ou égale à celle proposée par l'utilisateur. Un essai peut être conduit avec 10000 euros.

$$a = M \times \frac{r}{1 - (1 + r)^{-n}}$$

b / Boucle avec compteur For ... Next:

Cette boucle répète l'exécution d'une liste d'instructions en fonction de l'avancement d'une variable compteur. Le compteur est automatiquement augmenté de 1 (ou de la valeur précisée après **Step**) à chaque passage.

La variable compteur gagne à être exploitée dans les instructions à répéter au sein de la boucle pour nuancer les différentes exécutions.

For *variable compteur* = valeur de départ **To** Valeur de fin [**Step** pas]

Instructions à répéter

[Exit For]

Next [*variable compteur*]

Exemple :

```
Function sommeTableauFor(tableau)
    Dim indice
    Dim resultat
    resultat = 0
    For indice=lbound(tableau) To Ubound(tableau) Step 1
        resultat=resultat+tableau(indice)
    Next indice
    sommeTableauFor = resultat
End Function
```

Application : Sachant un taux d'actualisation de 4%, calculez, en pourcentage de sa valeur nominale, la valeur de marché d'une obligation au pair et à remboursement in fine sur 10 ans proposant un taux de coupon variable débutant à 1% et augmentant de 0,1% chaque année.

c / Boucle avec compteur For Each ... Next:

Cette boucle répète l'exécution d'une liste d'instructions pour chacun des éléments d'un tableau ou d'une collection. Lors de chaque itération, l'élément courant est accessible au travers de la variable précisée après les mots **For Each**. Cette variable doit obligatoirement être de type **Variant** (type par défaut lorsqu'aucun type n'est déclaré).

For Each *variable* **In** *Tableau ou collection*

Instructions à répéter

[Exit For]

Next [*variable*]

Exemple :

```
Function sommeTableauForEach(tableau)
    Dim élément
    Dim résultat
    résultat = 0
    For Each élément In tableau
        résultat=résultat + élément
    Next élément
    sommeTableauForEach = résultat
End Function
```

Application :

Créer une fonction qui calcule une valeur actuelle nette à partir d'un taux d'actualisation et d'une plage de cellule contenant des cash-flows en considérant que le premier est à la date courante et que deux cash-flows successifs sont séparés d'un an.