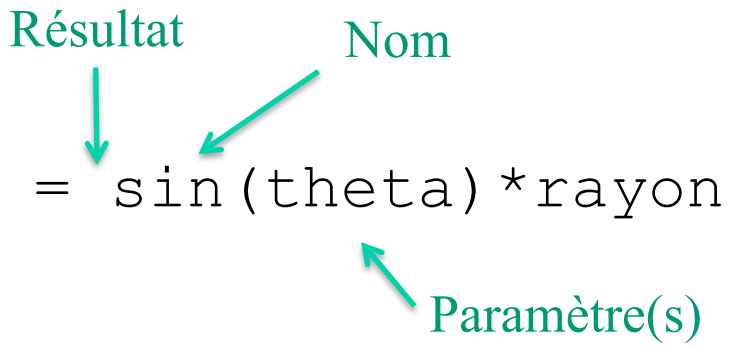
VI. Fonctions et procédures

1. Généralités
2. Ecrire/appeler une fonction
3. Les procédures
4. Paramètres par valeur et par référence
5. tableau en paramètre
6. Variables locales et globales
7. Exercice

1. Généralités



Une fonction ou procédure :

- Action qui porte un nom
- Accepte 0, 1 ou plusieurs paramètres
- Exécute une suite d'instructions
- Une fonction retourne un résultat (i.e. peut apparaître dans une expression)
- Une procédure ne retourne pas de résultat

Intérêt des fonctions et procédures :

➔ Ecrire une seule fois un algorithme utilisé plusieurs fois

➔ Algorithme paramétré

➔ Structurer un programme pour en faciliter la conception (1 tâche -> 1 procédure)

2. Ecrire/appeler une fonction

Ecrire une fonction peut se faire en 3 étapes :

a) concevoir l'algorithme,
b) écrire le code,
c) en faire une fonction.

a) Concevoir l'algorithme

- Quel objectif ? *Exemple : calculer la racine carrée d'un nombre*
- Quelles sont les données ? *un réel positif*
- Quel résultat ? *un réel positif*

Exemple pour le calcul de la racine carrée d'un réel positif

Méthode :

- On encadre la solution entre min et max
- On resserre min et max autours de la solution jusqu'à un écart < epsilon

```
soient les réels x, min, max, milieu, resultat
lire(x)
min = 0
si x<1 alors max = 1
sinon max = x
tant que (max-min)>0.001
    milieu = (min+max)/2
    si milieu*milieu > x alors max = milieu
    sinon min = milieu
fin tant que
resultat = (min+max)/2
ecrire resultat
```

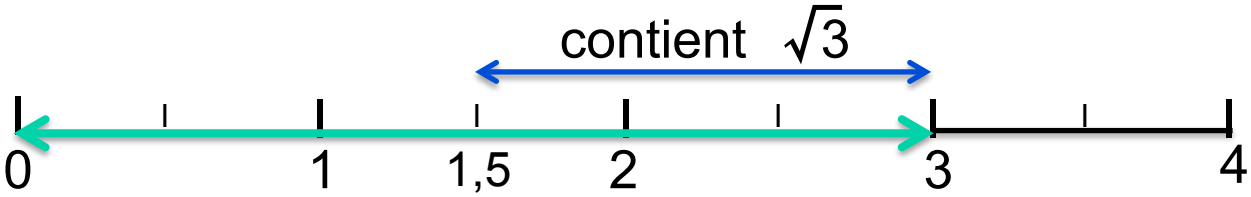
Récupérer la donnée

Réduire l'intervalle jusqu'à encadrer la solution à 10^{-3}

Communiquer le résultat

$\sqrt{3}$?

1,5*1,5 = 2,25 < 3



```
Function racine2(x As Float) As Float
   Dim x As Float, min As Float, max As Float
   Dim milieu As Float, resultat As Float
   x = InputBox("donner x")
   min = 0
   If x<1 Then
       max = 1
   Else
       max = x
   End If
   Do While (max-min)>0.001
       milieu = (min+max)/2
       If milieu*milieu > x Then
              max = milieu
       Else
              min = milieu
       End If
   Loop
   resultat = (min+max)/2
   racine2 = resultat
End Function
```

b) Écrire le code

```
Dim x As Double, min As Double, max As Double
Dim milieu As Double, resultat As Double
x = inputBox("?")
min = 0
If x<1 Then
  max=1
Else
  max=x
End If
Do While(max-min) > 0.001
  milieu = (min+max)/2
  If milieu*milieu > x Then
    max = milieu
  Else
    min = milieu
  End If
Loop
resultat = (min+max)/2
MsgBox resultat
```

Lecture de la donnée

Communiquer le résultat

c) En faire une fonction

Nom de la fonction

Paramètre formel (donnée)

Type du résultat

```
Function racineCarree(x As Double) As Double
   Dim x As Double, min As Double, max As Double
   Dim milieu As Double, resultat As Double
   x = inputBox("?")
   min = 0
   If x<1 Then
     max=1
   Else
     max=x
   End If
   Do While(max-min)>0.001
     milieu = (min+max)/2
     If milieu*milieu > x Then
       max = milieu
     Else
       min = milieu
     End If
   Loop
   resultat = (min+max)/2
   MsgBox racine
   racineCarree = resultat
End Function
```

Syntaxe d'une fonction :

Function *nom*(**[***listeArguments***]**) **As** *type*

  [*déclaration***]**

  [*instruction***]**

End Function

Résultat transmis via le nom de la fonction

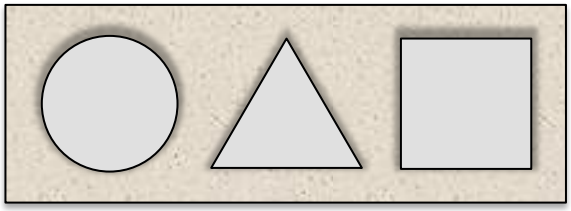
d) Appeler la fonction *exemple : calcul de l'hypoténuse d'un triangle rectangle*

```
Sub Main( )
  Dim larg As Double, haut As Double, H2 As Double, H As Double
  larg = inputBox("donnez la largeur : ")
  haut = inputBox("donnez la hauteur: ")
  H2 = larg*larg+haut*haut
  H = racineCarree(H2)
  MsgBox  "Hypotenuse = " & H
End Sub
```

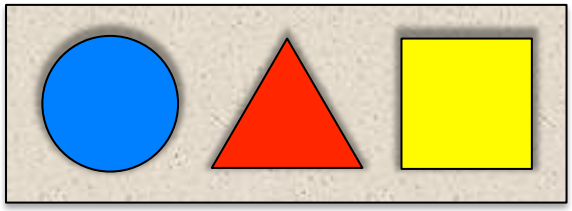
Appel de la fonction

Paramètre effectif

- Une fonction est déclarée après la procédure principale (ici `Main`)
- Elle peut être appelée dans n'importe quelle expression
- Le résultat de la fonction se substitue à l'appel pour l'évaluation de l'expression
- Les **paramètres effectifs** (ou paramètres d'appels) **remplacent** les **paramètres formels** dans l'exécution de la fonction appelée



formels



effectifs

e) Résumé sur les fonctions :

- ✓ Définir le nom, les paramètres avec leurs types et le type de la fonction

  Function *sinus(angle* ***As Double) As Double***

  Erreurs courantes : paramètres sans type, fonction non typée

- ✓ Déclarer les variables nécessaires au code en plus des paramètres

  Erreur courante : Redéclarer les paramètres

- ✓ Ecrire le code correspondant à l'algorithme

  Erreur courante : lire au clavier les paramètres données

- ✓ Affecter le résultat du calcul à l'identificateur de la fonction

  sinus = résultat

  Erreur courante : confondre "retourner le résultat" et "afficher le résultat"

- ✓ Clôturer la fonction

  End Function

- ✓ Appel de fonction

  sinus(a)

  Erreur courante : donner le type du paramètre ex : *sinus(a ~~As Double~~)*

f) Quelques fonctions prédéfinies sous VBA :

- ✓ Fonctions mathématiques

Fonction	Argument	Résultat
Abs	tout type numérique	Id type number
Atn	numérique	Double
Cos	numérique (en radian)	Double
Log	numérique (>0)	Double
Sin	numérique (en radian)	Double
Sqr	numérique (>=0)	Double
Tan	numérique (en radian)	Double

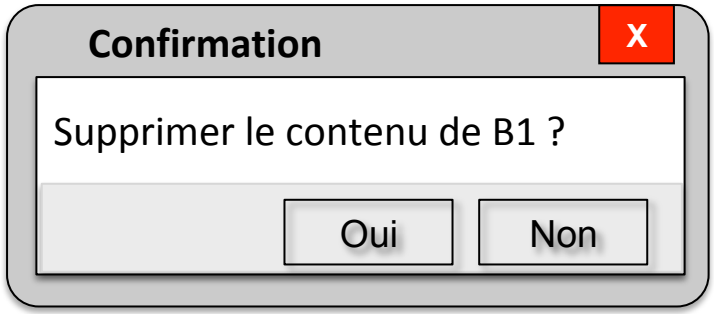
- ✓ Sur les chaînes de caractères (Len, StrComp, Replace, Val, …)
- ✓ Sur les dates (codage spécifique des dates non abordé ici)
- ✓ … *(voir documentation)*

g) Complément sur MsgBox

Exemple :

Message *Types de bouton* *Titre fenêtre*

```
Dim reponse As Integer
reponse = MsgBox("Supprimer le contenu de B1 ?", vbYesNo, "Confirmation")
If reponse = vbYes Then
    Cell(1, 2) = " "
    MsgBox "Le contenu de B1 a été effacé !"
End If
```



Type boutons VBA	Description	réponses		
vbYesNo	*Oui/Non*	vbYes	vbNo	
vbYesNoCancel	*Oui/Non/Annuler*	vbYes	vbNo	vbCancel
vbOkOnly	*Ok*	vbOk		
vbOkCancel	*Ok/Cancel*	vbOk	vbCancel	
vbRetryCancel	*Recommencer/Annuler*	vbRetry	vbCancel	
vbAbortRetryIgnore	*Abandonner/Recommencer/ Ignorer*	vbAbort	vbRetry	vbIgnore

3. Les procédures

- Même principe que les fonctions
- Ne sont pas typées et ne retournent pas de résultats
- Sont appelées avec l'instruction **`Call`**

```
Sub Main( )
  Dim nom As Variant
  nom = inputBox("Comment t'appelles-tu ?")
  Call Hello(nom)
End Sub

Sub Hello(name As Variant)
  msgBox "Hello " & name &   ", comment vas-tu ?"
End Sub
```

Quitter une fonction ou une procédure

Il peut être commode dans certains cas de quitter une procédure en cours d'exécution.

- **Exit Function** (pour les fonctions uniquement)
- **Exit Sub (**pour les procédures uniquement)

auront pour effet de :

1. quitter la fonction ou la procédure en cours
2. poursuivre le programme à la suite de l'appel de cette fonction ou procédure

Remarques :

- ces instructions seront toujours soumises à condition
- l'identificateur de la fonction doit être affecté du résultat avant l'instruction **Exit**

```
Function bidouille( … )
    …
    If condition Then
       bidouille = . . .
       Exit Function
    End If
    …
End Function
```

4. Paramètre par valeur et par référence

Le **passage par référence** est défini par défaut sous VBA

Le paramètre effectif se substitue au paramètre formel :

- il contient éventuellement une donnée nécessaire à la procédure
- et/ou il recevra une nouvelle valeur dans la procédure (paramètre résultat)

MAIS on souhaite parfois communiquer une copie de la donnée

(protection de l'original)

- Il faut dans ce cas faire un **passage par valeur**

Pour déclarer un paramètre **passé par valeur : `ByVal`**

- Syntaxe : **`ByVal`** *`variable`* **`As`** *`type`*
- Le paramètre formel devient une variable locale
- Le paramètre effectif sera recopié dans le paramètre formel (c'est une affectation)
- Le paramètre effectif ne pourra être modifié par l'appel
- **`ByRef`** (passage par référence) est la déclaration par défaut

Exemple :

```
Sub procedurePrincipale()
  Dim nb As Integer
  nb = 10
  Call somme1ersEntiers(nb)
  MsgBox nb
End Sub
```

n est une copie de nb

nb vaut toujours 10

```
Sub somme1ersEntiers(ByVal n As Integer)
  Dim s As Integer
  s = 0
  Do While n>0
    s = s+n
    n = n-1
  Loop
  MsgBox "somme = " & s
End Sub
```

nb n'est pas modifié

Exercices :

1. Ecrire la fonction factoriel(n) qui prend un Integer en argument et retourne un Integer.

Remarque : pour que le paramètre effectif puisse être une expression (voir exo2 (n-p) !), il faut utiliser un passage de paramètre par valeur (ByVal).

2. Ecrire la fonction Cnp(p,n) qui utilise la fonction factoriel pour calculer le nombre de combinaisons de p éléments parmi n.

Le résultat sera de type Integer.

On rappelle que : $C^p_n = n ! / (p ! (n-p) !)$

Remarque : ce n'est pas la bonne manière de calculer C^p_n *car les valeurs générées par la fonction factoriel dépassent rapidement la capacité d'un Integer . Mais l'objectif de cet exercice est seulement d'utiliser des fonctions.*

5. Tableau en paramètres

```
Sub procedurePrincipale()
  Dim tab(10) As Integer
  Call initTableau(tab, 10)
  MsgBox  "tab contient les 11 premiers entiers"
End Sub

Sub initTableau(t() As Integer, ByVal n As Integer)
  'remplit t avec les n premiers entiers.
  Dim i As Integer
  For i=0 To n
    t(i)= i
  Next i
End Sub
```

Tableau en paramètre effectif

Précise que `t` est un tableau

Dans la liste des paramètres de la procédure `initTableau` :

- `t()` désigne un tableau
- il faut aussi donner les bornes de `t` (en l'occurrence `n` dans l'exemple)

6. Variables locales et globales

a) Variables locales

- Les variables déclarées à l'intérieur d'une procédure sont dites **locales**
- Elles sont invisibles aux autres procédures
- Il n'y a pas de conflit de nom entre des variables locales de procédures différentes
- Ces propriétés sont vraies aussi pour les paramètres des procédures

```
Function max( t( ) As Integer, n As Integer)  As integer
   Dim i As Integer, m As Integer
   m = t(0)
   For i=1 To n
      If m < t(i) Then
         m = t(i)
      End If
   Next i
   max = m
End Function
```

```
Function min( t( ) As Integer, n As Integer)  As Integer
   Dim i As Integer, m As Integer
   m = t (0)
   For i=1 To n
      If m > t(i) Then
         m = t(i)
      End If
   Next i
   min = m
End Function
```

b) Variables globales

- Déclarées en dehors de toutes procédures
- De préférence en tête du module
- Accessibles par toutes les procédures (zones mémoires partagées)
- Elles sont occultées dans les procédures ayant des variables locales de même nom
- Les cellules de la feuille de calcul courante sont des variables globales (**Cells**)

Conseils :

✓ Utilisez de préférence les variables locales

✓ Réservez les variables globales pour des rôles importants dans le pb à traiter

✓ Commentez systématiquement le rôle des variables globales

7. Exercices

- **Fonctions avec paramètres résultats**

Ecrivez la fonction `Equa2 (a, b, c, x1, x2)` qui résout $ax^2 + bx + c = 0$

- Cette fonction **retournera** 0, 1 ou 2 suivant le nombre de solutions dans R
- Dans le cas d'une solution unique, x1 sera affecté de cette solution
- Dans le cas de deux solutions, x1 et x2 seront affectés par ces 2 solutions

Ecrivez la procédure principale qui

- lit les 3 coefficients de l'équation
- appelle la fonction `Equa2` avec ces paramètres
- affiche une réponse adaptée au nombre de solutions

7. Exercices

Nombres premiers

Soit la variable globale :

```
Dim tab(2 To 100) As Integer
```

- Ecrivez la procédure initTab( ) qui initialise tab avec les entiers de 2 à 100.
  Remarque : la valeur i se trouve à l'indice i dans tab

- Ecrivez la procédure suprimMultiples(n) qui « élimine » tous les multiples de n dans tab . On éliminera ces multiples en affectant la valeur 0 à la place.

- Ecrivez la procédure recopie1er (i) qui recopie les valeurs non nulles de tab dans la i-ème ligne de la feuille excel en commençant par la cellule (i, 1)

- En utilisant ces 3 fonctions, écrivez la procédure nombre1er( ) qui « filtre » tous les nombres premiers entre 2 et 100 et les recopie dans la 1ère ligne de la feuille de calcul

Remarque : cette solution n'est pas optimale car la suppression des multiples de n ne tient pas compte des étapes précédentes (par exemple que les nbres pairs ont été effacés)