Un Langage de programmation

Pour quoi faire ?

Maîtriser les outils informatiques
Mieux comprendre la structure de l'information manipulée par un logiciel.
Pouvoir développer des petites applications personnalisées.

Environnement de travail choisi

Un document Word ou Excel
Les bibliothèques de programmes et d'informations qui permettent de modifier le document ;
Environnement de programmation : débogueur, aide en ligne

Les Macros

Définition :

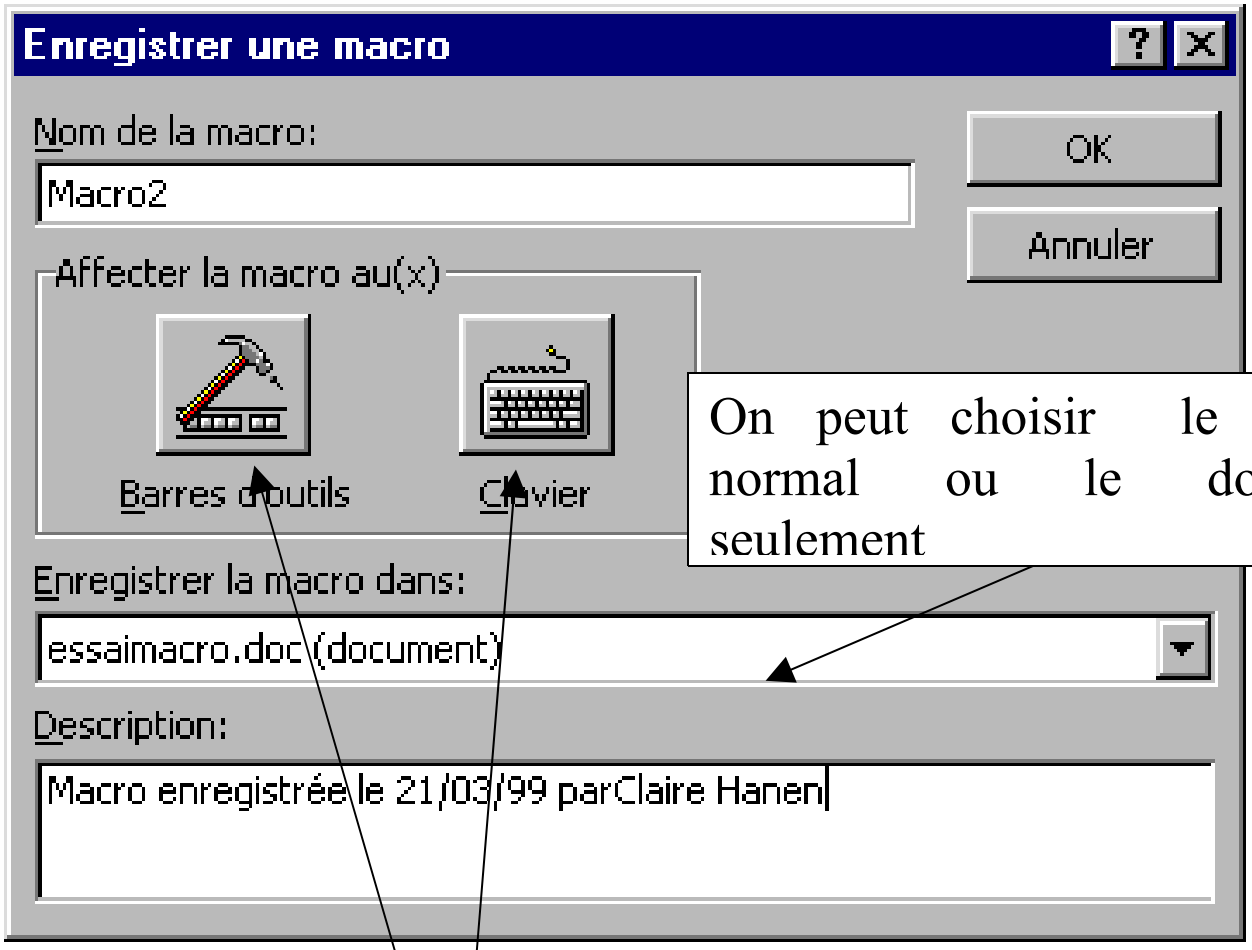
Une macro est la mémorisation d'une suite de commandes d'un logiciel.
Une macro correspond à un sous-programme qui peut être écrit dans un langage de programmation et qui est exécutable à la demande de l'utilisateur.
Ce sous-programme possède un **nom** défini par l'utilisateur.
Il peut être associé à un bouton de la barre d'outils ou à une touche du clavier.
Dans tous les cas il est exécutable par le **menu Outils-Macro**

Enregistrer une macro

Pourquoi faire ? : automatiser des tâches répétitives.
Exemple : enregistrer un document sur la disquette et sur le disque dur.

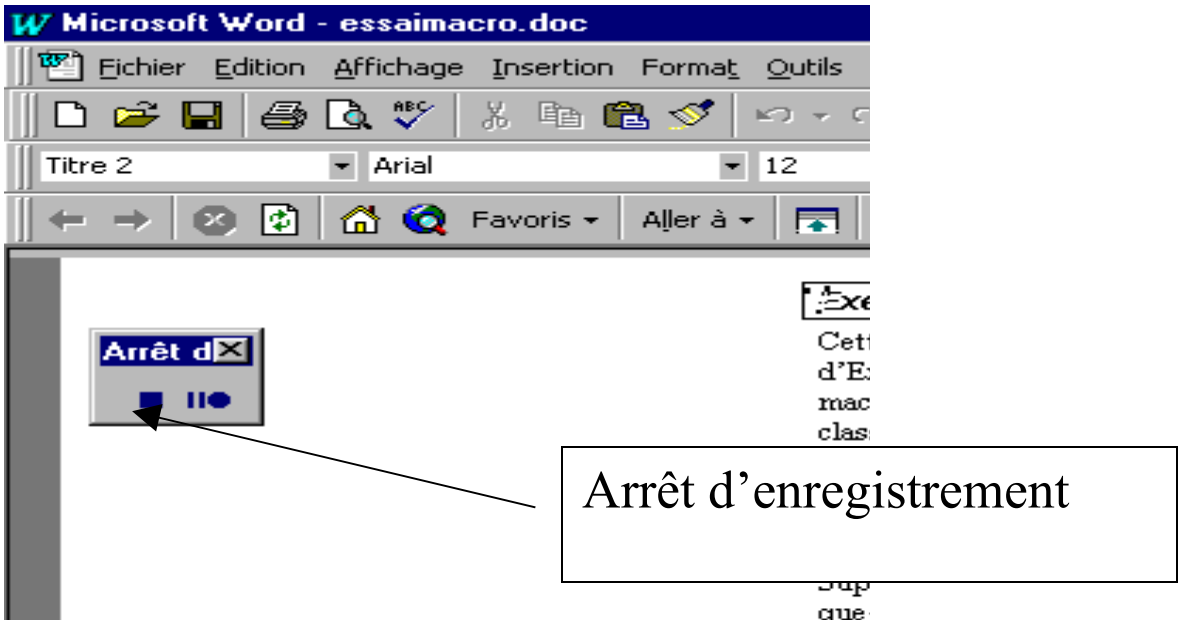
Enregistrement et exécution

Menu Outils-Macro-Nouvelle Macro



On peut choisir le modèle normal ou le document seulement

Associer à un bouton ou à une touche

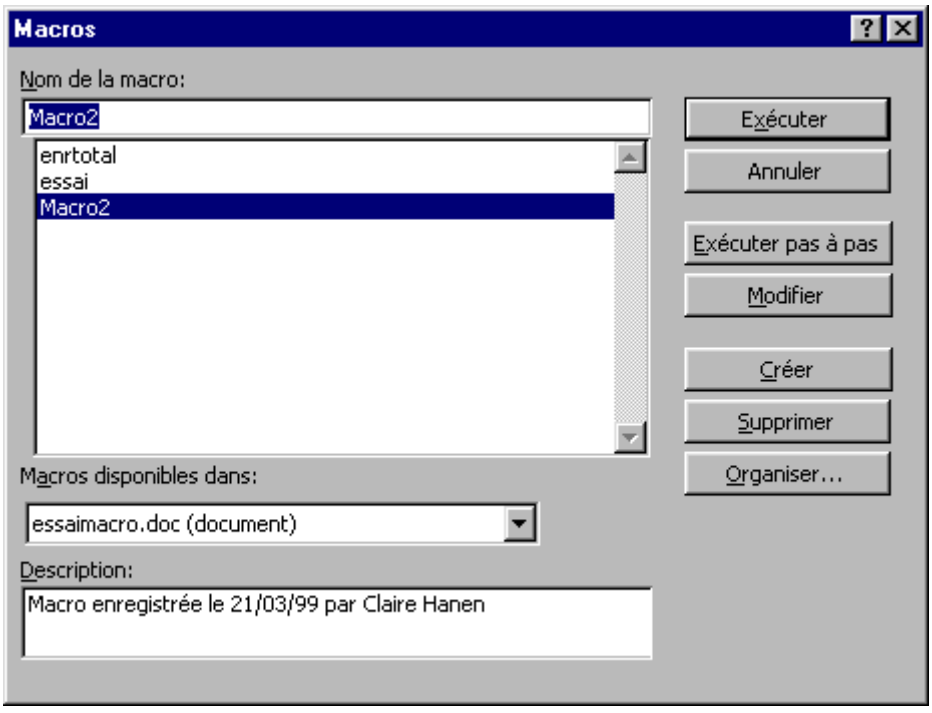


Enregistrement de la suite de commande

2 fois enregistrer sous, en choisissant d'abord la disquette, puis le bon répertoire du disque dur)

Exécution de la macro

(Un peu plus tard)
Menu **Outils-Macro-Macros**…



Environnement VBA

Actions du logiciel

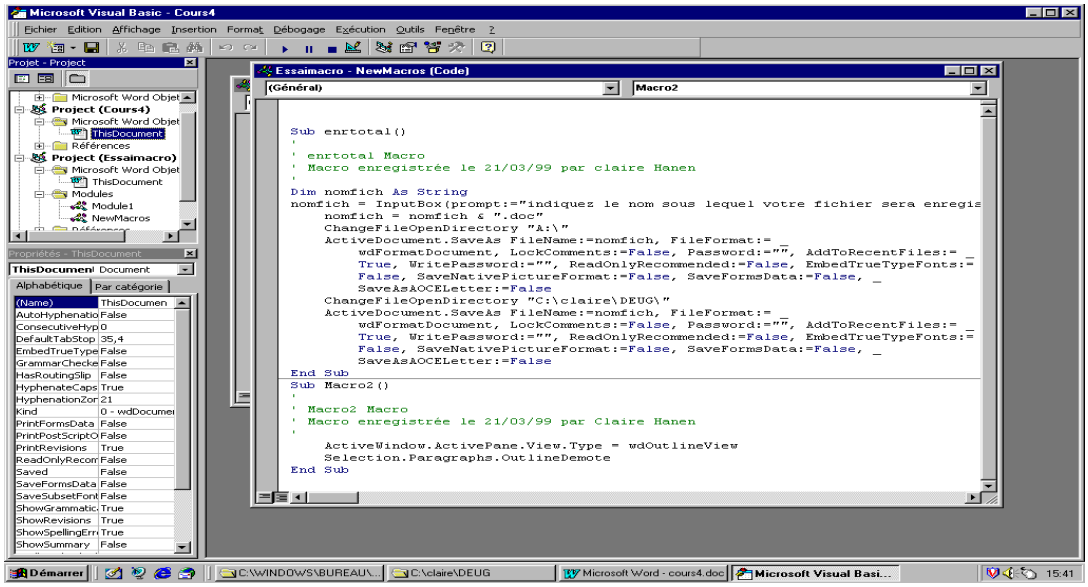
Le logiciel a créé un sous-programme écrit en Visual Basic, à partir de composants prédéfinis fournis avec le logiciel.

Lorsque l'on demande l'exécution de la macro

1) Le sous-programme est compilé (traduit en langage machine)
2) Chargé en mémoire
3) Exécuté.

Maîtrise du sous-programme

L'utilisateur peut visualiser le texte Visual Basic, le modifier, et avoir de l'aide pour corriger ses erreurs.



Menu Outils-Macro-Visual Basic editor

Texte de la macro :
Sub/ End Sub : début/fin de sous-programme
Structure générale : suite d'instructions. Chaque instruction sur une ligne, ou alors sur plusieurs lignes avec un espace souligné en fin de ligne

```
Sub Macro2()
'
' Macro2 Macro
' Macro enregistrée le 21/03/99 par Claire Hanen
'
    ChangeFileOpenDirectory "A:\"
    ActiveDocument.SaveAs FileName:="essaimacro.doc", FileFormat:= _
        wdFormatDocument, LockComments:=False, Password:="", AddToRecentFiles:= _
        True, WritePassword:="", ReadOnlyRecommended:=False, EmbedTrueTypeFonts:= _
        False, SaveNativePictureFormat:=False, SaveFormsData:=False, _
        SaveAsAOCELetter:=False
    ChangeFileOpenDirectory "C:\claire\DEUG\"
    ActiveDocument.SaveAs FileName:="essaimacro.doc", FileFormat:= _
        wdFormatDocument, LockComments:=False, Password:="", AddToRecentFiles:= _
        True, WritePassword:="", ReadOnlyRecommended:=False, EmbedTrueTypeFonts:= _
        False, SaveNativePictureFormat:=False, SaveFormsData:=False, _
        SaveAsAOCELetter:=False
End Sub
```

Eléments de base

Chaque ligne correspond à une instruction. Si une instruction ne tient pas sur une ligne, il faut mettre en fin de ligne un blanc souligné et passer à la ligne suivante.

Instructions de déclaration

Il s'agit d'indications pour le compilateur, lui permettant de traduire les instructions de calcul et leur enchaînement en langage machine.

Instructions de calcul

Il s'agit d'instructions demandant l'exécution d'un calcul ou d'un programme déclaré au préalable (par une instruction de déclaration) et modifiant l'état de la mémoire.

Les instructions sont composées à partir de mots du langage (dits ***mots réservés***) et de noms définis par le programmeur à l'aide des instructions de déclaration.

Tout programme est une ***suite d'instructions***, avec une syntaxe précise et dépourvue de toute ambiguïté.

Une macro exécutable par le menu Macro est une procédure sans paramètres :

Sub nomdeproc() (Instruction de déclaration)
Suite d'instructions
End Sub

Notion de variable

Le programmeur peut associer un nom de son choix à un espace mémoire, dans lequel il pourra stocker des informations. C'est une variable

Types de base

Pour des raisons de codage, ce nom est associé à un type d'information bien précis. En VBA , les principaux types d'information de base sont :

Byte : nombre entier de 0 à 255
Integer : nombre entier de –32768 à 32767
Boolean : True ou False
String : chaîne de caractères
Double : nombre décimal en double précision
Variant : lorsqu'il peut y avoir plusieurs types

Déclaration de variables locales

Se fait en début de procédure.
Syntaxe :
Dim NomdeVar **As** TypedeVar

NomdeVar est un nom choisi par le programmeur (sans espace, sans point)

TypedeVar est un nom de type, parmi ceux existant.

Effet de la déclaration : le compilateur réserve un espace en mémoire centrale pour les futures valeurs de la variable, correspondant à la taille du codage correspondant à son type.

Utilisation de variables locales :

Une fois déclarée, la variable peut être utilisée dans n'importe quelle instruction de la macro dans laquelle elle est déclarée. Elle n'est pas connue des autres macros.

Instruction élémentaire: l'affectation

Définition : ranger une information (valeur) dans l'espace mémoire associé à une variable, et donc modifier la valeur de la variable.

Syntaxe :

NomDeVariable = Expression

Il faut qu'il y ait concordance de type entre l'expression et la variable (soit elles sont de même type, soit VBA sait comment transcrire un type dans un autre).

Effet : Calcule la valeur de l'expression et range en mémoire à l'emplacement occupé par la variable le code de cette valeur. Après cette instruction, la variable a changé de valeur.

Exemple :

```
Sub essai()
Dim a As Integer
Dim b As Integer
a=15          met la valeur entière 15 dans case a.
b=a+2         met la valeur entière 17 dans case b.
b=b*2 +a      met la valeur entière 49 dans la case b
End Sub
```

Entrées-Sorties

Définition : moyen de communiquer avec l'utilisateur d'un document Excel d'échanger des données.

VBA offre de nombreuses manières d'inscrire des informations directement dans des cellules et de faire des calculs à partir d'un tableau Excel.

Dans un premier temps, on s'en affranchira, en utilisant des outils très rudimentaires (pour mieux comprendre les éléments du langage).

Entrée d'information :

Utilisateur vers macro
Utilisation de la Fonction prédéfinie **InputBox**.

Syntaxe :

NomdeVar=InputBox(message)
NomdeVar est une variable préalablement déclarée
message est une expression de type String, par exemple " entrez un entier "

Effet :

-Affichage d'une boite de dialogue devant le classeur Excel avec le message.
-L'utilisateur tape ensuite l'information demandée dans la zone de dialogue.
- Cette information est rangée dans la variable NomdeVar

Sortie d'information

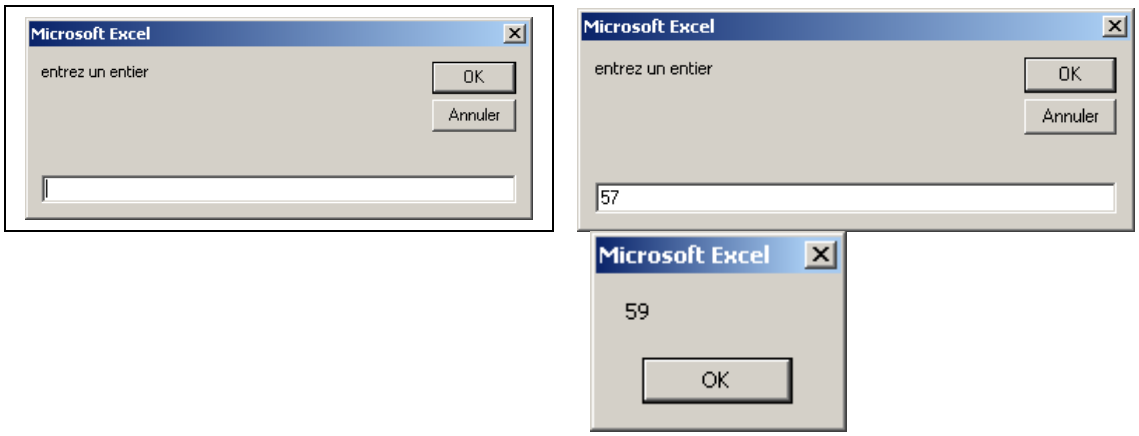
Programme vers utilisateur.
Utilisation de la procédure prédéfinie **MsgBox**

Syntaxe : MsgBox(Expression)

Effet :
Calcul de la valeur de l'expression puis affichage de cette valeur dans une boite d'information à l'utilisateur du document Excel.

Exemple :

Sub essai()
Dim x **As** Integer
x = **InputBox**("entrez un entier")
MsgBox (x+2)
End Sub



Sub carre()
Dim z **As** double
z=**Inputbox**(" entrez un nombre ")
z=z*z
MsgBox(z)

End Sub

Expressions en VBA

Expressions de base

Constantes

Une **constante** est une expression.
Constantes String: une suite de caractères entre guillemets..:" ceci est une phrase "
Constantes entières (Byte ou integer): écriture habituelle d'un nombre entier: 25, -68
Constantes Boolean: True, False
Constantes double: écriture habituelle d'un nombre décimal.
Il existe des constantes prédéfinies dans les applications qui possèdent des noms
Exemple : vbOK de type Byte, qui vaut 1

Variables

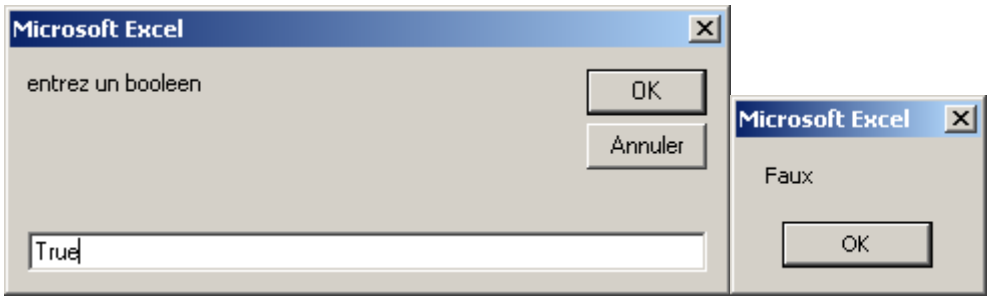
Un nom de variable (ou **identificateur**) est une expression. La valeur de l'expression est alors la valeur de la variable au moment du calcul.

Expressions avec Opérateurs

Les constantes et les variables peuvent être combinées au moyen d'opérateurs (comme en Excel)
Opérateurs logiques: not, and, or
Syntaxe:
not expr expr1 and expr2 expr1 or expr2

expr, expr1,expr2 doivent être des expressions de type boolean

```
Sub test1()
Dim a As Boolean
a = InputBox(" entrez un booleen")
MsgBox (Not a)
End Sub
```



Opérateurs de comparaison: >,<=,=,>=,>

Syntaxe:

Expr1 op expr2

Expr1 et Expr2 sont des expressions de même type, ou peuvent être converties dans un même type, op est un opérateur. Le type de l'expression est Boolean.

Exemple:

MsgBox (x<12)

La boite affiche faux si la variable x est de valeur supérieure ou égale à 12, vrai sinon.

Opérateurs arithmétiques: +,-,*,/

Attention: toute expression de type double, si elle est affectée à une variable de type integer ou Byte, est tronquée (on en prend la partie entière inférieure).

Exemple du carré.

Opérateur sur les chaînes: &

C'est l'opérateur de concaténation.

Exemple:

```
Sub concat()
Dim chaine As String
chaine=  InputBox( "  entrez votre nom ")
chaine= "Bonjour  " & chaine &" ! "
MsgBox(chaine)
End Sub
```

Structures de contrôle

Objectifs :

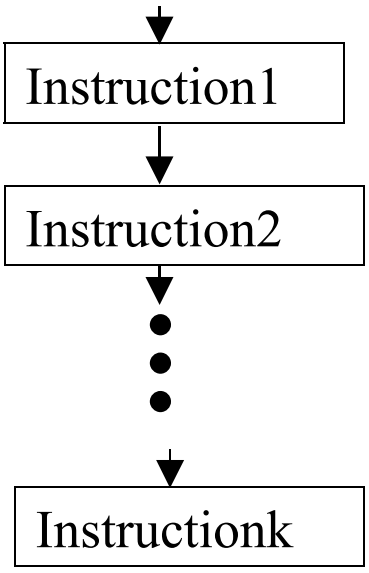
Donner au programmeur des outils pour définir les instructions effectivement exécutées, et leur ordre d'exécution, en fonction du contexte de l'exécution.

Deux types principaux :

Structures de contrôle ***alternatives*** (choix entre plusieurs séquences d'instructions)
Structures de contrôle ***répétitives*** (possibilité de répéter, sous conditions, une séquence d'instruction).

Déroulement d'un programme sans structures de contrôle :

```
Sub toto()
Instruction1
Instruction2
.
.
.
Instructionk
End Sub
```



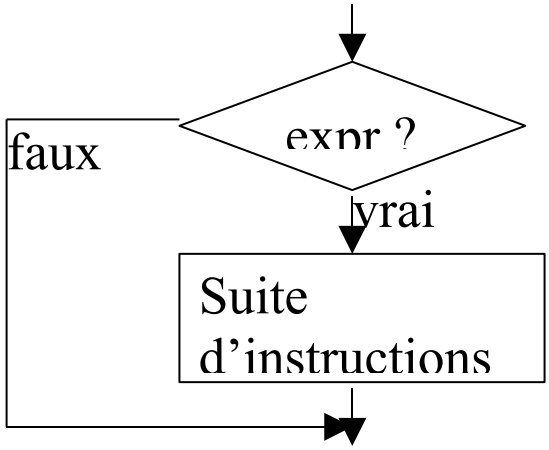
Structures Alternatives

Définition :

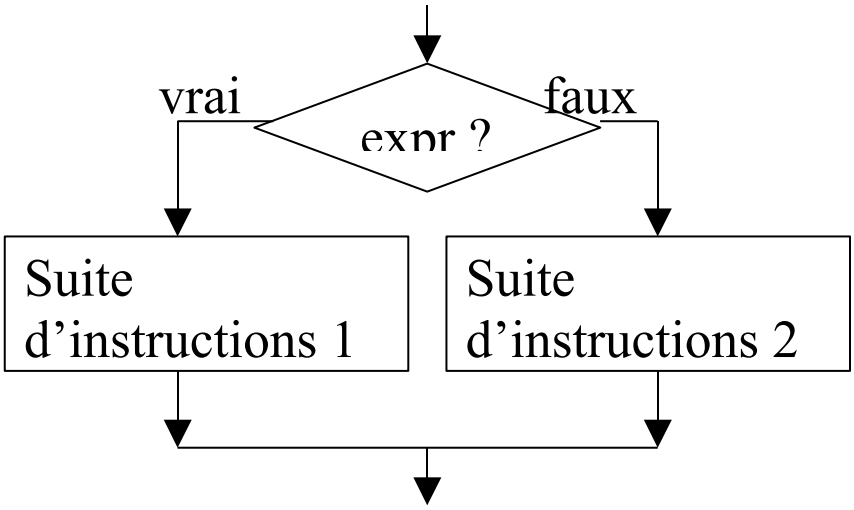
Choix entre plusieurs séquences d'instructions selon la valeur d'une expression.

If Then Else

Syntaxe :
If expr **Then**
Suite d'instructions
End If



Ou
If expr **Then**
Suite d'instructions 1
Else
Suite d'instructions 2
End If



expr est une est une expression à de type booléen (vraie ou fausse)

Exemple If

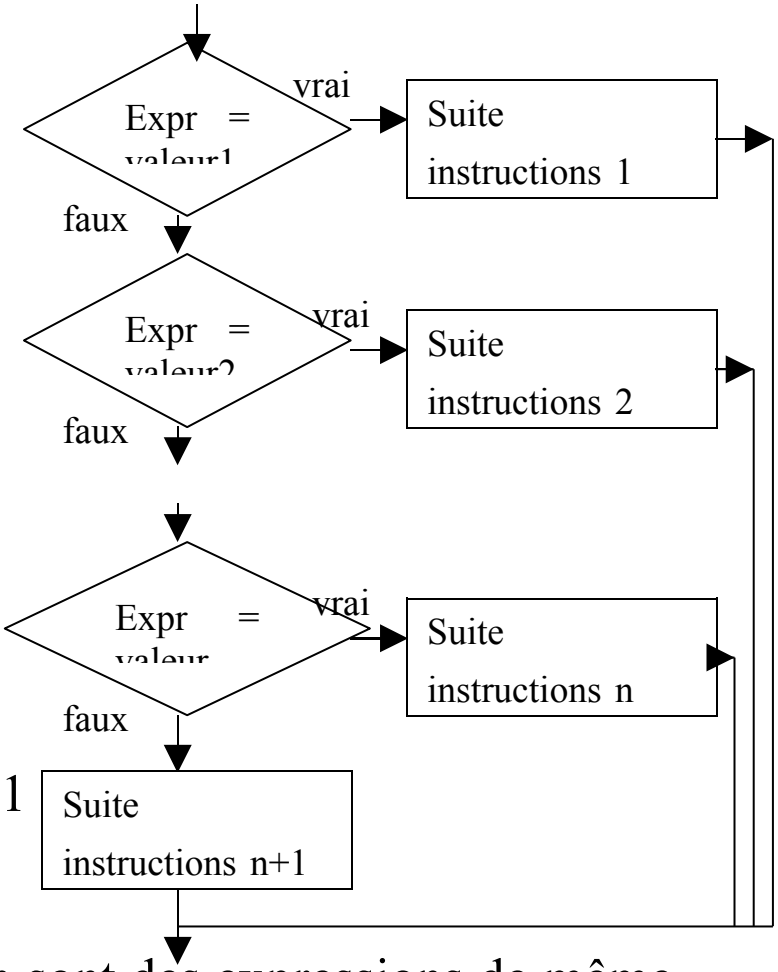
```
Sub moyen ()
Dim note1,note2, moy As double
note1=InputBox("entrez votre première note")
note2=InputBox("entrez votre seconde note")
moy=(note1+note2)/2
If moy<10 Then
MsgBox("eliminé")
Else
MsgBox("Reçu")
End If
End Sub
```

Case Select

Choix d'une séquence d'instruction parmi plusieurs selon la valeur d'une expression.

Syntaxe :
Select Case expr
Case valeur1
    Suite instructions 1
Case valeur 2
    Suite instructions 2

Case valeur n
    Suite instructions n
Case Else *´(facultatif)*
    Suite instructions n+1
End Select



Expr, valeur1, .. ;valeur n sont des expressions de même type.

Exemple Case

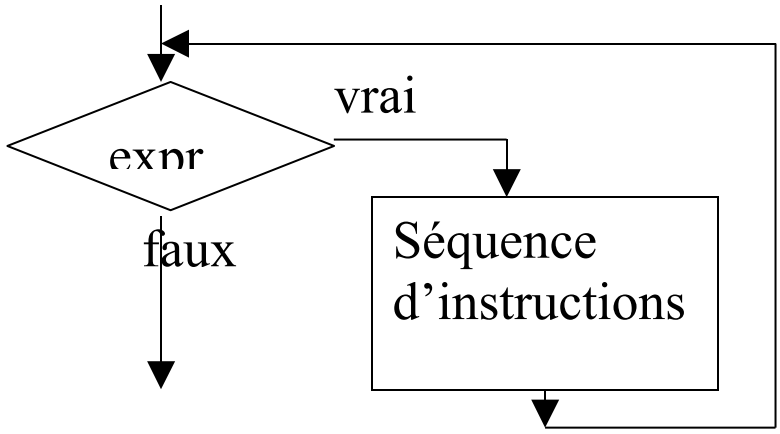
```
Sub departement()

Dim Deptnum As Integer
Dim Deptnom As String
Deptnum = InputBox(" entrez un numéro de département ")
Select Case reponse
Case 75
Deptnom="Paris"
Case 91
Deptnom=" Essonne "
Case 77
Deptnom= "Seine et Marne"
Case 78
Deptnom="Yvelines"
Case 92
Deptnom= "Hauts de Seine"
Case 93
Deptnom= "Seine Saint Denis"
Case 94
Deptnom= "Val de Marne"
Case 95
Deptnom= "Val d'oise"
Case Else
      Deptnom= "ce n'est pas en ile de France"
End Select
MsgBox(Deptnom)
End Sub
```

Structures répétitives

Do While…Loop

Syntaxe :
Do While expr
Séquence d'instructions
Loop



expr est une expression à valeur booléenne (vraie ou fausse)
Pour utiliser une boucle While, il faut que l'expression devienne fausse à un moment de l'exécution. (sinon le programme ne s'arrête pas)

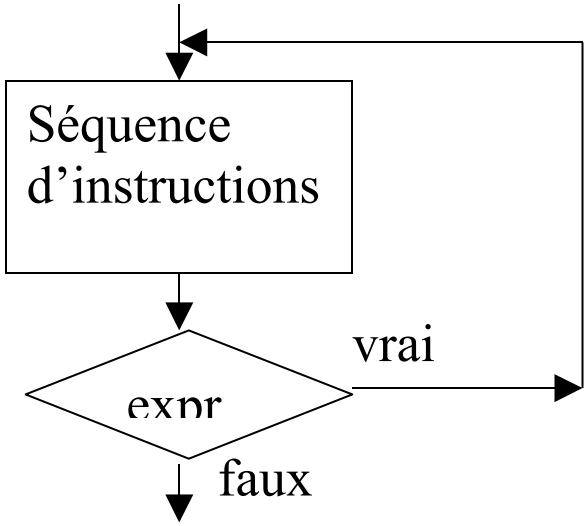
La séquence d'instruction doit donc être susceptible de modifier la valeur de vérité de expr.

Boucle infinie :
```
Dim n as integer
Dim somme as integer
n=1
somme=0
Do while n<10
somme =somme +n
Loop
```

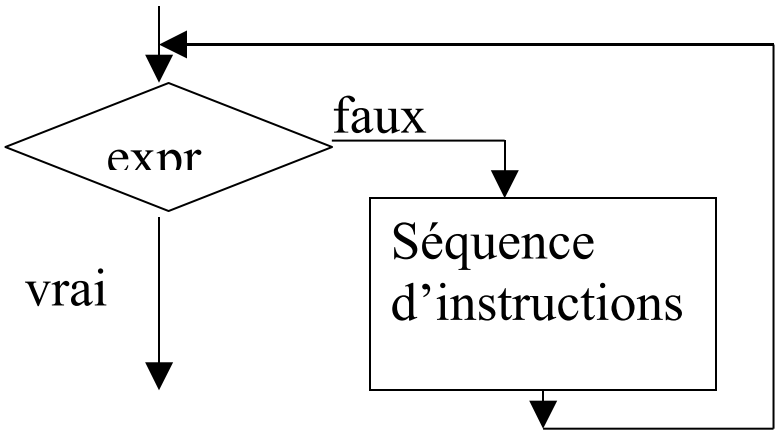
Boucle correcte :
```
Dim n as integer
Dim somme as integer
n=1
somme=0
Do while n<10
somme =somme +n
n=n+1
Loop
```

Variantes de la boucle while :

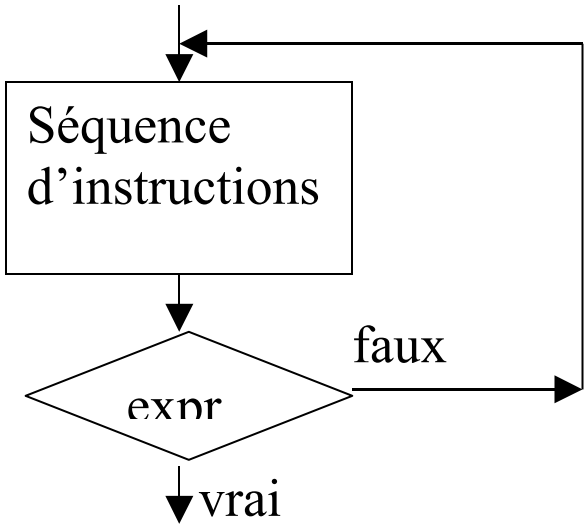
Do
Séquence d'instructions
Loop While expr



Do Until expr
Séquence d'instructions
Loop



Do
Séquence d'instructions
Loop Until expr



Boucle For …Next

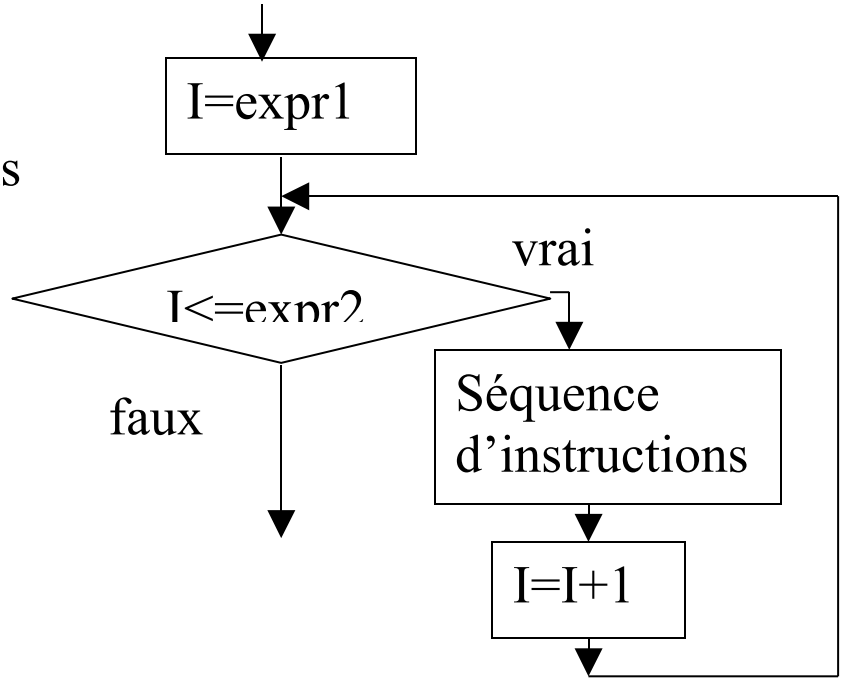
Pour utiliser cette structure de contrôle, on doit avoir une variable de type numérique (Byte, integer, Simple, Double) qui sert à compter le nombre d'itérations de la boucle

Syntaxe simplifiée:

(si I est la variable compteur)

expr1et expr2 sont deux expressions de même type que I (ou convertissable)

For I=expr1 To expr2
Séquence d'instructions
Next

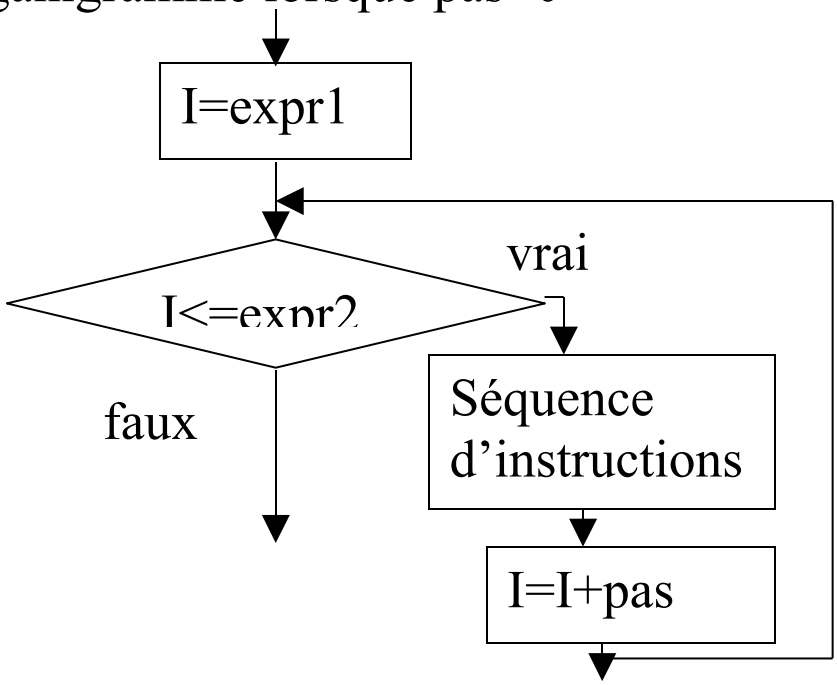


Si au cours de la séquence d'instructions (par exemple dans l'une des alternatives d'un If ou d'un Select)

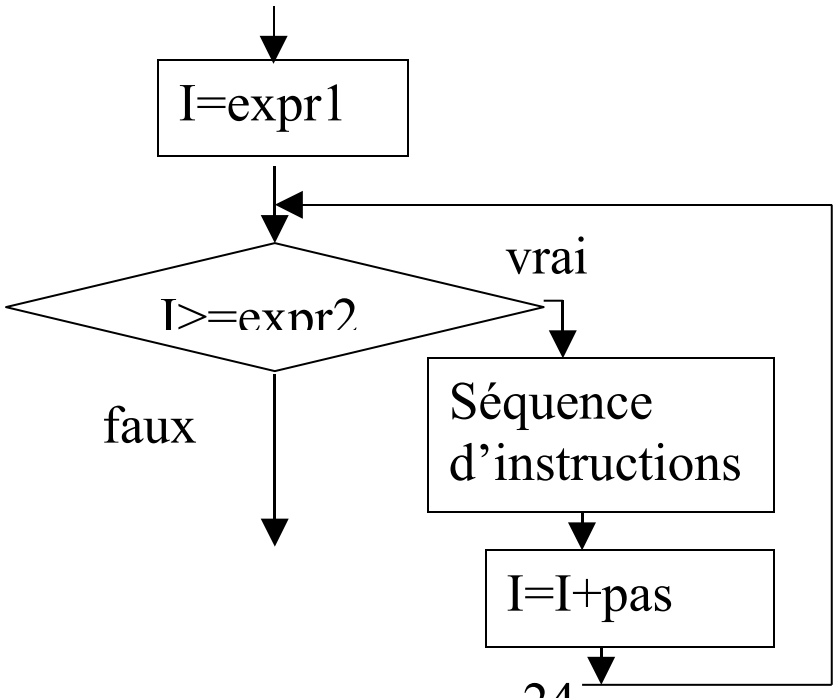
On insère une instruction **Exit For,** cela revient à interrompre la boucle et à passer à l'instruction suivante.

Syntaxe générale :
For I = expr1 To expr2 [Step pas]
Séquence d'instructions
Next

Pas est une expression de même type que I
Si elle n'est pas indiquée, par défaut elle vaut 1
Organigramme lorsque pas>0



Organigramme lorsque pas est <0



Appels de fonctions

Les fonctions considérées ici peuvent-être:
Des fonctions prédéfinies
Des méthodes-fonctions d'objets du logiciel
Des fonctions définies par le programmeur

L'appel à une fonction est une expression. La valeur de l'expression est calculé par le programme qui définit la fonction au moment du traitement de l'appel. Comme toute expression, elle a un type, qui fait partie de la définition de la fonction. Une fonction peut éventuellement posséder des paramètres.

En-tête de fonction

Les informations de base concernant le mode d'emploi d'une fonction sont résumées dans l'en-tête de la fonction (une description qui peut se trouver dans l'aide en ligne pour les fonctions existantes).

Syntaxe employée pour décrire l'en-tête:
Function nomfonc(paraform1 As type1,…,paraformk As typek) As typefonc
paraform1, … paraformk sont des noms dits noms de **paramètres formels**. type1,…typek sont des noms de types, tout comme typefonc.
Parfois une fonction n'a pas de paramètres.
Parfois certains paramètres sont facultatifs (ils apparaissent entourés de crochets dans l'en-tête).

Syntaxe(s) d'un appel.

Syntaxe 1: paramètres listés:

Nomfonc(parareel1,parareel2,…,parareelk)
Parareel1,… sont dits paramètres réels de cet appel.
Parareel1 est une expression de type type1
Parareel2 est une expression de type type2
…
parareelk est une expression de type typek

l'appel lui-même est une expression de type typefonc.
Attention: *les paramètres réels sont séparés par des virgules et non des points-virgules, contrairement à Excel.*
Exemple:
fonction existante sqr (calcule la racine carrée)
En-tête : function sqr(Number As Double) As Double
Appel:
x=sqr(2)
Est une instruction qui donne la valeur racine de 2 à la variable x. 2 est le paramètre effectif de cet appel.

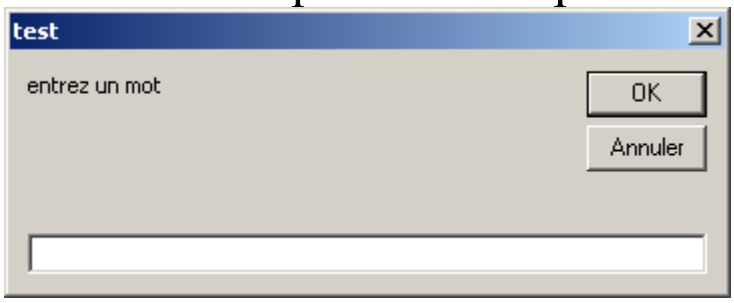
Syntaxe 2: paramètres nommés

Nomfonc (paraform1**:**=parareel1,…,paraformk**:**=parareelk)
On peut parfois se passer des parenthèses
Exemple:

x= sqr (Number:=2)

Même effet que la syntaxe 1

la fonction InputBox complète :



En-tête (trouvée dans l'aide)

Function InputBox(prompt As string[, title As string] [, default As string] [, xpos As integer] [, ypos As integer] [, helpfile, context])
prompt Expression de chaîne affichée comme message dans la boîte de dialogue
title Facultatif. Expression de chaîne affichée dans la barre de titre de la boîte de dialogue.
default Facultatif. Expression de chaîne affichée par défaut dans la zone de texte en l'absence de toute autre valeur
….
Les autres arguments sont également facultatifs.

Exemple :

```
Sub test()
Dim reponse As String
reponse = InputBox(prompt:="entrez un mot",_
Title:="test")
MsgBox (reponse)
End Sub
```

MsgBox complète

(jusque là utilisée comme une procédure)
La **fonction** MsgBox, qui affiche un message et renvoie une valeur indiquant le bouton de commande choisi par l'utilisateur.
Paramètres principaux :
Prompt : Message affiché (seul paramètre obligatoire)
Buttons : Identification des boutons à afficher

Pour afficher les boutons	**Constante prédéfinie**	**Valeur numérique de la constante**
OK	vbOKOnly	0
OK/Annuler	vbOKCancel	1
Oui,Non, Annuler	vbYesNoCancel	3
Oui, Non	vbYesNo	4
Réessayer, Annuler	vbRetryCancel	5

Title :titre de la boite de dialogue

Selon le bouton cliqué par l'utilisateur, la valeur renvoyée est la suivante :

Bouton choisi	**Constante renvoyée**	**Valeur numérique de la constante**
OK	VbOK	1
Annuler	VbCancel	2
Réessayer	VbRetry	4
Oui	VbYes	6
Non	VbNo	7

Appels de procédures

Les procédures sont soit:
Des procédures prédéfinies
Des méthodes d'objets du logiciel
Des procédures (macros) définies par le programmeur

L'appel à une procédure est une **instruction** (et pas une expression)

En-tête d'une procédure

Comme pour les fonctions le mode d'emploi d'une procédure est donné par son en-tête.

Syntaxe employée pour décrire l'en-tête:
sub nomproc(paraform1 As type1,…,paraformk As typek)

paraform1, … paraformk sont des noms dits noms de **paramètres formels**. type1,…typek sont des noms de types.

Parfois une procédure n'a pas de paramètres.
Parfois certains paramètres sont facultatifs (ils apparaissent entourés de crochets dans l'en-tête).

Syntaxe(s) d'un appel.

Syntaxe 1: paramètres listés:

Call Nomproc(parareel1,parareel2,…,parareelk)

Parareel1,… sont dits paramètres réels de cet appel.

Parareel1 est une expression de type type1

Parareel2 est une expression de type type2

…

parareelk est une expression de type typek

Attention: *les paramètres réels sont séparés par des virgules et non des points-virgules, contrairement à Excel.*

Syntaxe 2: paramètres nommés

Nomproc paraform1**:=**parareel1,…,paraformk**:=**parareelk

La liste des paramètres peut éventuellement être entourée de parenthèses.

Déclaration de fonctions et procédures

On peut définir soi même des fonctions et procédures

Procédures

Sub nomproc(liste de paramètres formels)
Instructions
End Sub

Fonctions :

Function nomfonc(liste de paramètres formels) **As** nomtype
Instructions permettant de calculer le résultat de la fonction, avec une instruction (ou plusieurs) du type : nomfonc = expression
End Function
Les noms des paramètres formels peuvent être utilisés comme des valeurs dans des expressions.

Remarque : sous Excel, **il est possible d'utiliser une fonction programmée comme une fonction de cellule,** avec la même syntaxe d'appel que les autres fonctions d'Excel (paramètres effectifs séparés par des points-virgule).

Ecriture sous forme fonction d'exemples :

```
Function conveuro( F As Double) As Double
Conveuro=F/6,559
End Function
```

```
B1=CONVEUR0(A1)
```

```
Sub dialogue
Dim S As double
S=InputBox(« entrez une somme en francs »)
MsgBox(conveuro(S))
End sub
```

Une fonction Excel qui calcule la puissance nième d'un nombre :

```
Function puissance (x As double,n As Byte) As Double
Dim res As double
Dim compteur As Byte
Res = 1
For Compteur = 1 to n
res = res*x
Next
Puissance = res
End Function
```

Utilisation dans une autre fonction VB :

```
a = puissance (x :=3.5, n :=3)
```

Utilisation dans la feuille de calcul :

```
=puissance(A1 ;B1)
```

Objets

Les logiciels Word et Excel permettent au programmeur d'accéder à des informations concernant le document et de les modifier au travers du langage VBA.

Pour cela, il faut comprendre comment les informations sont structurées dans le logiciel.

Fonctionnement général:
En mémoire vive se trouvent les données et les programmes associés au logiciel et aux documents ouverts.
Chaque commande du logiciel, ou chaque intervention de l'utilisateur modifie les informations en mémoire.

Puis l'un des composants du logiciel utilise ces informations pour afficher le document en visualisant éventuellement les modifications qui en résultent.

La représentation en mémoire des données et des programmes du logiciel repose sur la notion d'objet:

Notion d'objet

Un **objet** est soit un **objet terminal**,
Soit un **ensemble** d' attributs/**propriétés**, et/ou **ensemble** de **méthodes**.
Physiquement, chaque objet est associé à un espace bien défini en mémoire centrale (dont l'emplacement est connu du logiciel, et défini lors de la traduction du programme en langage machine).

Objet terminal: variable d'environnement

Un objet terminal est un espace mémoire destiné à contenir une information d'un type fixé.
Dans la suite, les objets terminaux seront désignés par le terme de **variables d'environnement**.

Propriétés et Méthodes

Les propriétés sont des objets.
Les méthodes sont des programmes permettant de modifier l'objet ou son environnement, ou de fournir des informations calculées à partir de l'objet.

Désigner un objet

Chaque objet possède un nom. Un objet qui est propriété d'un autre objet, ou un programme qui est méthode d'un objet ne peut être désigné indépendamment de l'objet qui le contient.

Ainsi, si un objet nommé O contient une propriété/méthode nommée A, le terme O**.**A désigne l'objet propriété/méthode A de l'objet O.

Notion de classe

Certains objets possèdent des attributs et des méthodes semblables (c'est à dire avec les même caractéristiques-même noms, mêmes actions pour les méthodes, même types pour les sous-objets terminaux). On dit qu'ils appartiennent à la même classe. Chaque classe d'objet prédéfinie par le logiciel cible possède également un nom.

Exemple : les classeurs Excel sont des objets de la classe Workbook.
Les feuilles d'un classeur sont des objets de la classe Worksheet
Les cellules ou plages de cellules sont des objets de la classe Range.
L'objet Selection désigne la plage de cellules sélectionnée. C'est un objet de classe Range.
Lorsque l'utilisateur sélectionne une plage avec la souris, les informations en mémoire concernant l'objet Selection sont mises à jour par le logiciel.

Notion de Collection:

Certaines propriétés sont des collections d'objets de même classe. (une collection d'objet est elle même un objet).
Exemples : les feuilles d'un classeur Excel. Les cellules, ou les plages de cellules d'une feuille
Pour pouvoir manipuler les objets d'une collection, on a recours à une **indexation**.
Exemples : Sheets est un objet collection qui désigne l'ensemble des feuilles du document actif.
Sheets(2) désigne la feuille numéro 2 du document actif.
Sheets(2). Cells désigne l'ensemble des cellules de la feuille 2 du document actif.
Sheets(2).Cells(3,5) désigne l'objet cellule située à la troisième ligne et la cinquième colonne de la feuille 2 du document actif.

Exemples avec Excel

Exemple de procédures:
Toutes les macros enregistrées sont des procédures.
Les procédures avec paramètres ne peuvent être appelées que par d'autres procédures ou fonctions.

```
Sub enregistre(Nomfichier As String)
Dim chemin As String
Dim i As Byte
For i=1 to 2
    If i=1 Then
        Chemin="A:\"
    Else
        Chemin="C:\Ecolang\"
    End If
    Chemin=Chemin & Nomfichier
    ActiveWorkbook.SaveAs Filename:=Chemin
Next
End sub

Sub demandesauve()
Dim nomfich As String
Dim reponse As Byte
reponse = MsgBox(Prompt:=" voulez-vous enregistrer votre fichier?",
Buttons:=vbYesNo)
If reponse = vbYes Then
    Nomfich= InputBox Prompt:= "nom de fichier:"
    Enregistre(Nomfich)
    Else
    reponse = MsgBox(Prompt:="votre fichier n'a pas été
enregistré")
    End If
End Sub
```

Désigner la représentation mémoire d'une cellule ou d'une plage de cellules:

ActiveCell: objet représentant la cellule active.
Selection: objet représentant la plage de cellules sélectionnées
Cells(l,c) où l,c sont des expressions de type entier désigne la cellule ligne l colonne c de la feuille active.
Sheets(x).Cells(l,c) où x désigne la cellule au croisement de la ligne l et de la colonne c de la feuille x.
Range("ref") où ref est une référence excel de type A1 (exemple A2:C7) désigne la plage de cellule correspondante de la feuille active.

Exemples de variables d'environnement.

Sheets(1).Cells(2,1).Font.Size est une variable d'environnement de type Byte. Réprésente la taille de la police de caractères de la cellule ligne 2 colonne 1 de la feuille 1.
Sheets(1).Cells(2,1).Font.Bold de type Boolean. Réprésente l'aspect gras ou non de la police de caractères de cette même cellule (contient True si c'est en gras, False sinon)
ActiveCell.Font.Name de type String: nom de la police de caractères de la cellule active.
ActiveCell.FormulaR1C1 de type String: texte de la formule en mode LC associé à la cellule active

ActiveCell.Value de type Variant: valeur de la cellule active (représentation interne)

Exemples de méthodes sans paramètres :

Rendre une cellule active :
Méthode Activate de la classe Range.
Sheets(1).Cells(3,5).Activate
Cette action met à jour l'objet ActiveCell, et l'objet Selection.
Effacer le contenu d'une cellule :
Méthode Clearcontents de la classe Range
Sheets(1).Cells(3,5).Clearcontents

Exemple2 : une fonction de plage de calcul qui calcule la somme des valeurs des cellules colorées en jaune de la plage.

```
Function sommejaune(maplage As Range)
Dim ligne,colonne,lmax,colmax As integer
Dim somme As double
lmax=maplage.Rows.Count
colmax=maplage.Columns.Count
somme=0
For ligne = 1 to lmax
For colonne=1 to colmax
If maplage.cells(ligne,colonne).Interior.ColorIndex= 6 Then
Somme=somme+ maplage.cells(ligne,colonne).value
Next colonne
Next ligne
Sommejaune=somme
End Function
```

Appels de fonctions-objets

Certaines méthodes-fonctions renvoient des objets. C'est le cas de la méthode Find de tout objet de classe Range (c'est à dire qui correspond à une plage de cellule).
Un appel à cette méthode renvoie un objet cellule (lui-même de classe Range).
Pour utiliser le résultat d'une méthode-fonction de ce type, on doit définir une variable objet et lui affecter la référence de l'appel à la méthode.
Exemple.
Le paramètre obligatoire de la méthode Find est la valeur recherchée.

```
Sub tesmeth()
Dim c As Range
Set c=selection.Find("bien")
c.Font.size=20
End Sub
```

Selection.find(10) renvoie la première cellule de la sélection contenant le texte "bien". L'instruction suivante met la police de cette cellule en taille 20.