CYCLE SUPERIEUR
1ère Année
2009- 2010

VBA & EXCEL & ACCESS
pour la Statistique

Annick VALIBOUZE

QuickTime™ et un décompresseur
BMP sont requis pour visualiser
cette image.

Annick Valibouze
ISUP
Université Pierre et Marie Curie
4, place Jussieu
F-75252 Paris cedex 05
http://www.isup.upmc.fr

VBA & EXCEL & ACCESS

Pour la statistique

TABLE DES MATIERES

Chapitre 1 Excel : Premiers Pas

Lancement :
Une **feuille de calcul** dans l'environnement Excel.

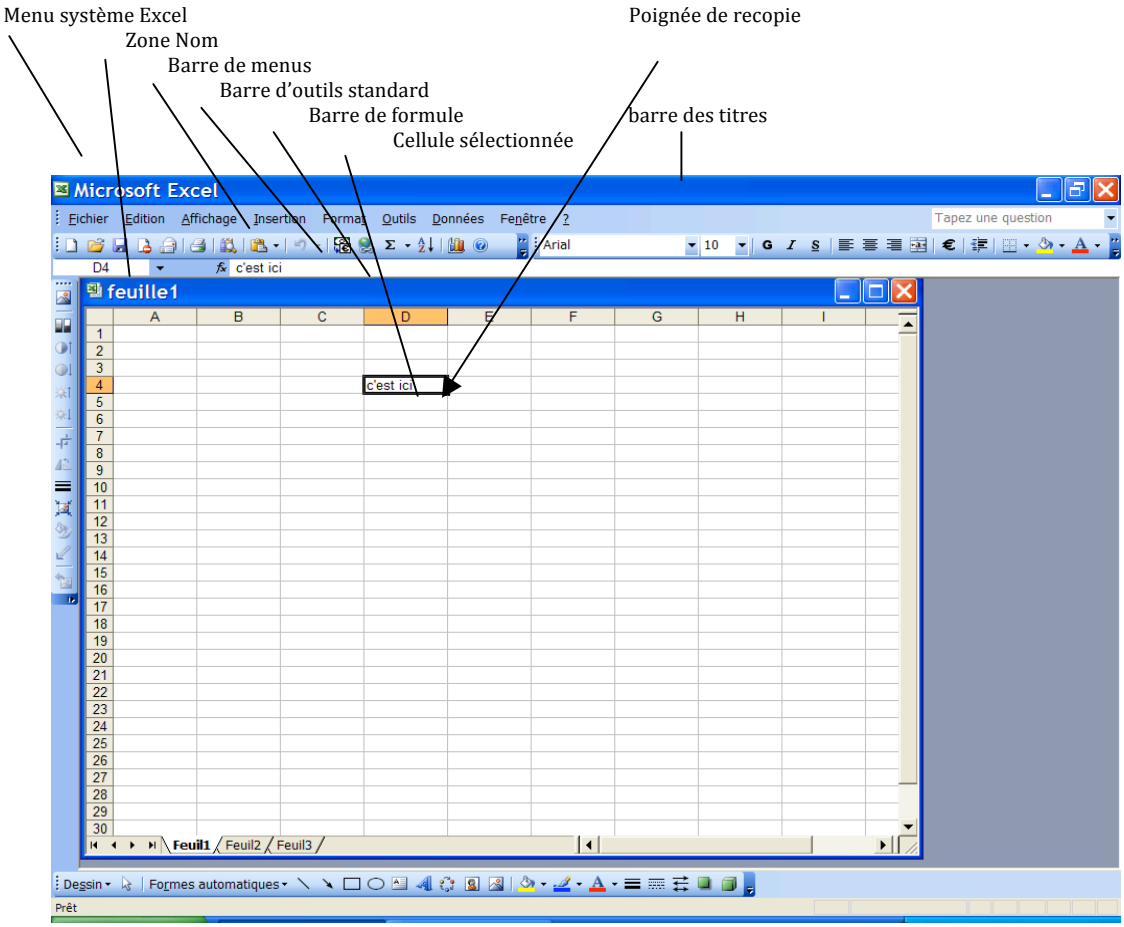


Figure 1

C'est un tableau à double entrée.
Les colonnes sont indicées par des lettres (A, B,…) et les lignes par des entiers (1, 2,…).

Cellules

La feuille de calculs est constituée de **cellules** notées
Ai, Bi, Ci, …. (représentées par Ai, Bi, Ci , ...)
où i est un entier ; c'est le format $A$1.

Une cellule est dite **active** si elle est entourée d'un cadre noir gras avec un carré noir en bas à droite.

Pour activer une cellule :
- cliquer dessus
- taper sa notation dans la **zone nom**
- s'y déplacer avec les flèches (ou TAB ou CTRL)

Saisie du contenu :

- dans la cellule elle-même
- **dans la barre des formules**

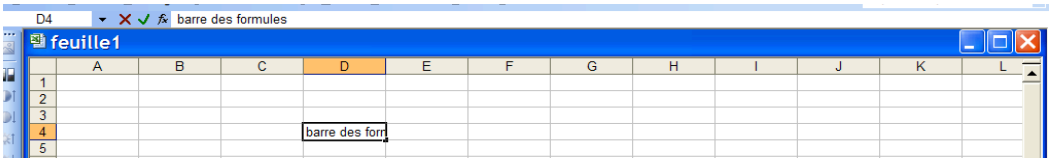


Figure 2

Validation avec : Entrée, Y , cliquer sur une autre cellule

Correction avec : la barre des formules

Annulation de la dernière modification : cliquer sur la croix rouge

Suppression : - sélectionner les cellules (touche CTRL enfoncée si non connexes) puis
- utiliser la touche clavier Suppr ou bien
- Dans la barre Menu : Edition/Effacer (tout ou formules)

Fusion : Format/cellule/Alignement/fusionner

Déplacer une plage de cellules : sélectionner la plage ; cliquer avec le bouton droit sur l'un des quatre bords ; déplacer la plage bouton enfoncé ; lacher le bouton à l'emplacement désiré.

Copier une plage de cellules : utiliser la **poignée de copie** (voir plus loin, **la propagation**) ou sélectionner la plage avec la souris, taper la donnée dans l'une des cellules et faire CTRL + ENTREE.

Intersection de plages de cellules : utiliser la touche espace du clavier comme opérateur d'intersection. Elles ne fonctionnent pas avec les références 3D (voir plus loin).

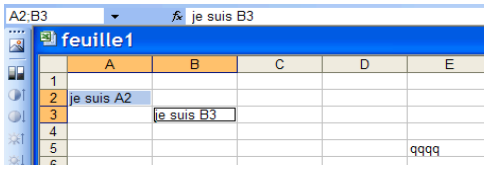


Figure 3

Note. Le signe « : » désigne un intervalle et le signe « ; » sépare des éléments.

Menu Contextuel et Format :

Affichage du **menu contextuel** de la cellule avant et après validation.

Afin qu'apparaisse le menu contextuel, se placer à l'intérieur de la cellule et cliquer sur le bouton droit de la souris.
Ci-dessous, le contenu de la cellule n'est pas encore validé.

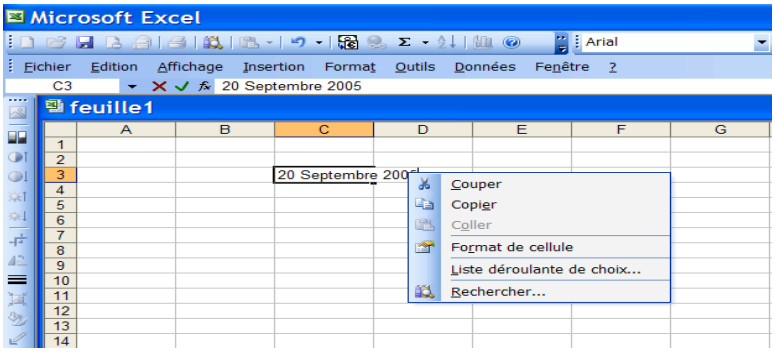


Figure 4

Dans cette figure, le contenu de la cellule a été validé. Le menu contextuel a été modifié et le contenu de la cellule a été interprété par Excel. On note la différence entre l'affichage dans la cellule et le contenu de la barre des formules.

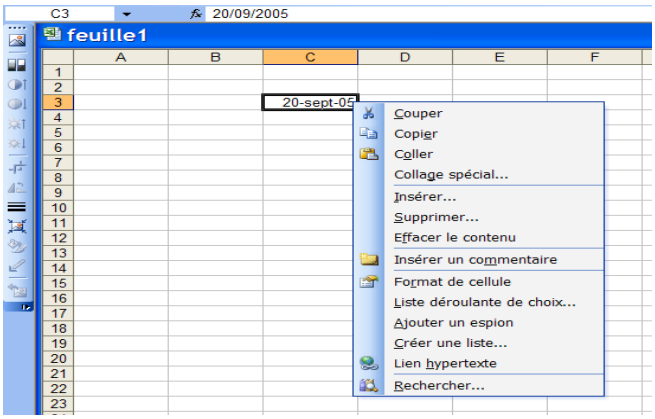


Figure 5

En demandant le **Format** dans le menu contextuel, nous observons celui choisit par défaut par Excel. Il est possible alors de le modifier.

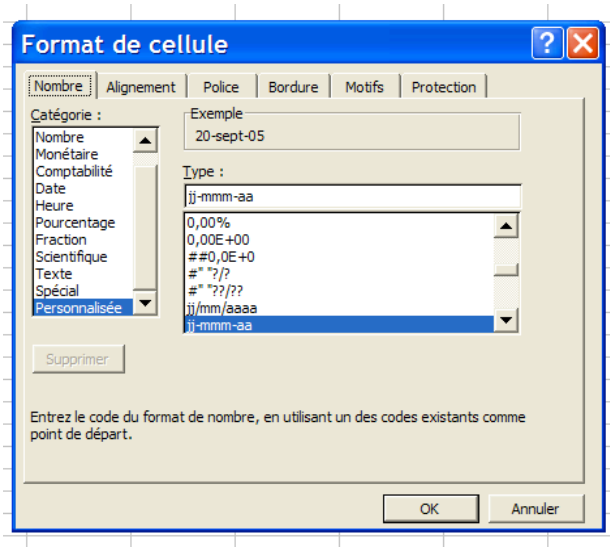


Figure 6

Changer d'environnement : Référence absolue, relative ou mixte

Dans Outil/Options/Général, nous changeons pour adopter un autre système que $A$1. Nous lui préfèrerons le système matriciel « L1C1 » (voir Figure 7).

Dans ce système matriciel, sont définis trois types de références :

Référence **absolue :** LiCj désigne la cellule ligne i colonne j

Si la cellule active est LiCj alors pour deux entiers relatifs k et l

Référence **relative** : L(k)C(l) désigne la cellule ligne i+k colonne j+l
Référence **mixte** : L(k)Cl désigne la cellule ligne i+ k colonne l.

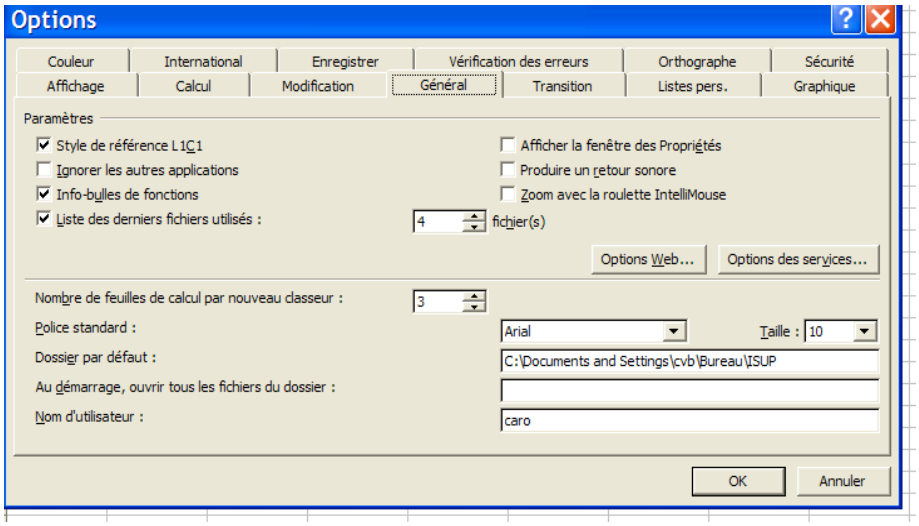


Figure 7

Astuce : Pour passer d'une référence à l'autre automatiquement, tapez sur F4 lorsque La cellule ou le groupe de cellules est sélectionné.

Note : - Par défaut, les noms utilisent toujours des références absolues aux cellules.
- Dans l'autre mode, $A$1 est une référence absolue et A1 est une référence relative.
- voir Outils/Options/Formules.

Référence à une feuille et à un classeur

[NomClasseur]NomFeuille !PlageCellules

Exemple : [NotesPremierSemestre]Feuil2 !L1C1:L4C4

La mise à jour est réalisée si le classeur contenant la liaison est ouvert.
Si le classeur source (contenant les données) n'est pas ouvert, il faut référencer en donnant le chemin : "C:\...\...\ [NomClasseur]NomFeuille !PlageCellules" .

Référence 3D : NomFeuille1 : NomFeuille2 !

Contenu d'une cellule

- ✓ Les **valeurs**

Les différents types sont :

- Booléen : VRAI ou FAUX avec un affichage centré.
- Un nombre entier ou « réel » : 12 ou 12,7 (voir les formats) avec affichage à droite.
- Une date, une heure (voir formats) avec affichage à droite.
- Un texte avec affichage à gauche.

✓ Les **formules**

Chaque cellule peut être le résultat d'une formule.

Une formule commence par le signe =.

Elle peut utiliser

- Des opérateurs arithmétiques : +, -, *, /
- Des opérateurs de comparaisons : <, <=, >=, =, <>
- Des opérateurs booléens : ET, OU, NON
- Des fonctions :

Des fonctions prédéfinies apparaissent dans la zone nom si on clique sur sa flèche.

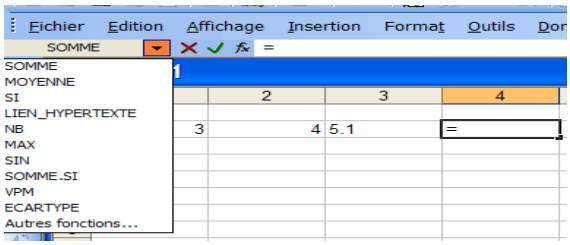


Figure 8

La moyenne est sélectionnée et Excel nous propose la liste des cellules sur lesquelles s'opèrera cette moyenne : de la cellule C(-3) à la cellule C(-1) de la même ligne. Astucieusement, Excel propose les formules avec des références relatives.

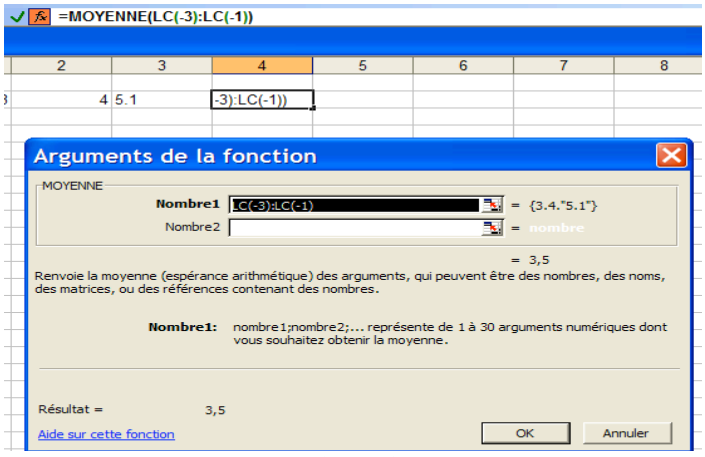


Figure 9

Une fois validé : le contenu de la cellule affiche l'**évaluation** de la fonction alors que la barre des formules affiche son **corps**.

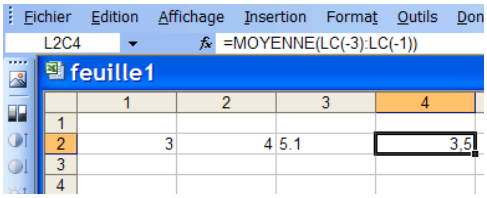


Figure 10

L'instruction **SI-ALORS-SINON**

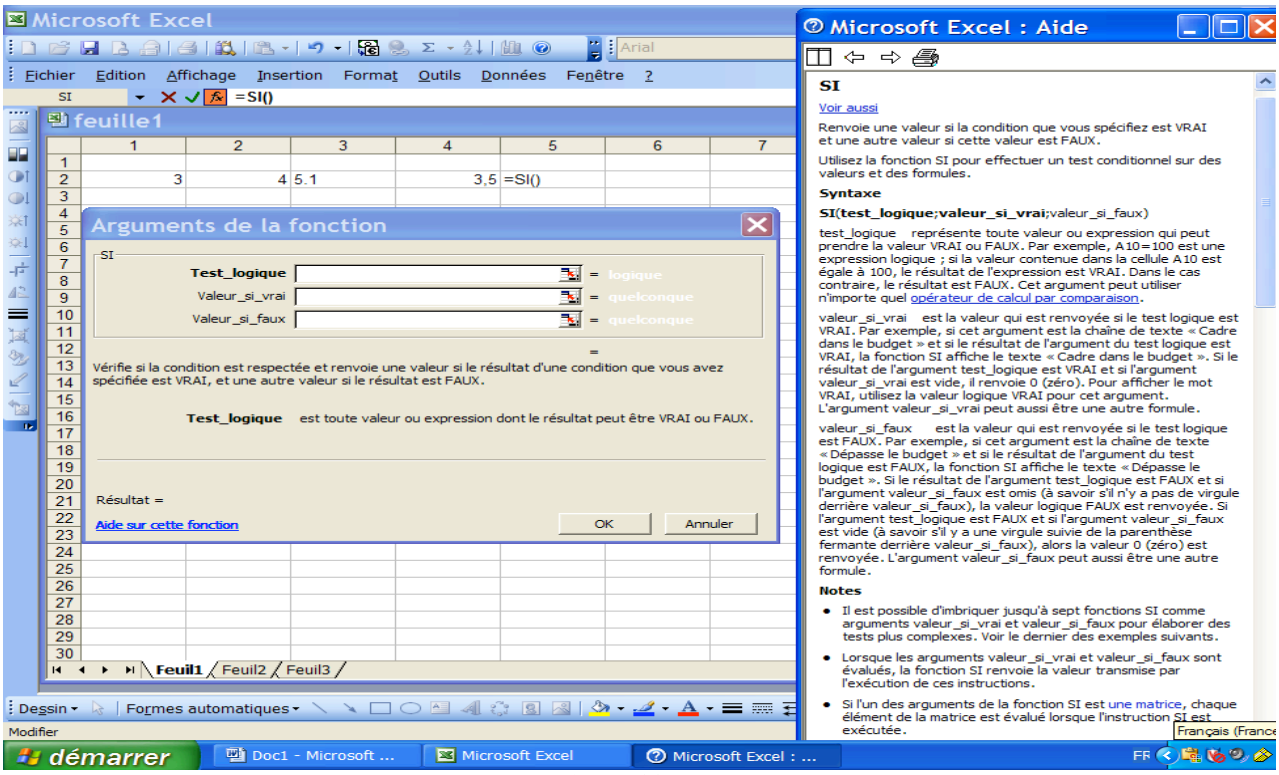


Figure 11

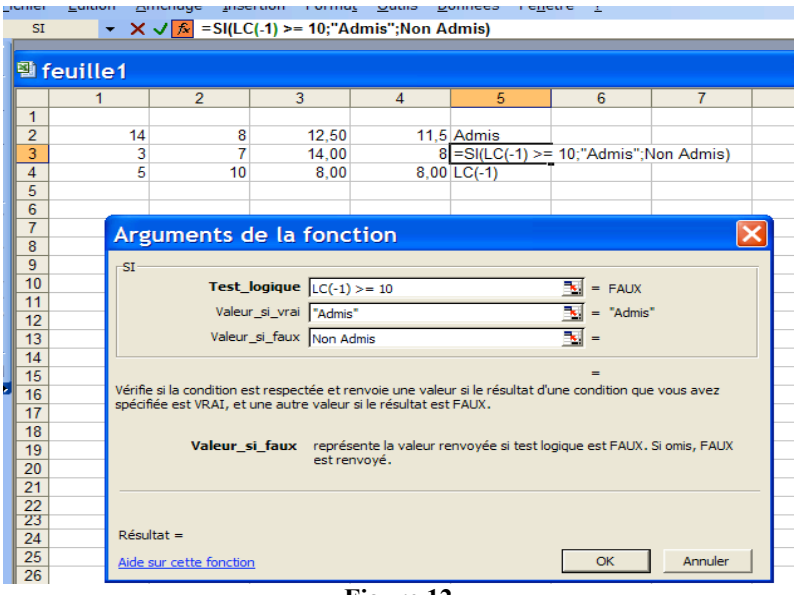


Figure 12

La propagation

Pour éviter de répéter la formule sur toute la colonne, on utilise la **propagation** de valeurs ou de formules en activant la cellule où est enregistrée la formule et en définissant la **zone** de propagation en cliquant sur le carré noir en bas à droite de la cellule (la **poignée de copie**) et en propageant en tenant le bouton gauche appuyé ; puis « Copier les cellules ».

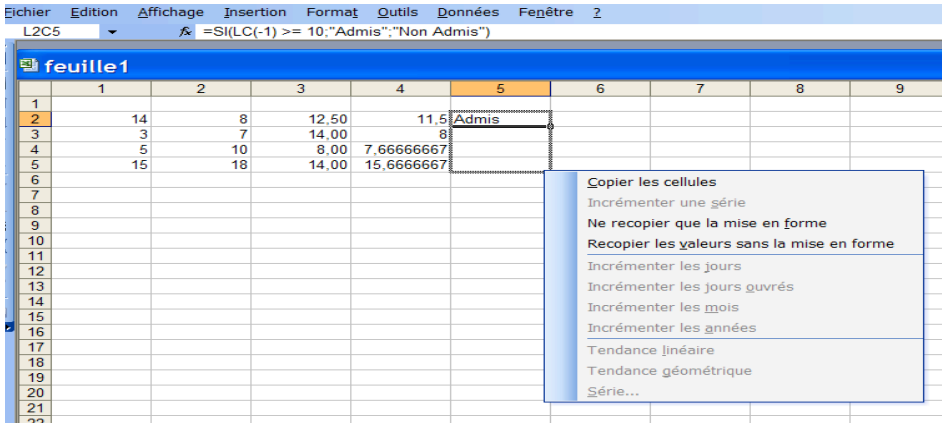


Figure 13

Note : pour faire apparaître le menu de copie, il faut cliquer sur la poignée avec le bouton droit et tirer dessus.

Utilisations de la zone :

- supprimer tout son contenu
- propager une formule ou une valeur
- propager linéairement (i.e. le long d'une ligne ou le long d'une colonne) une série ; Excel interprète 2 puis 4 comme la suite $u_n = u_{n-1} + 2$;

Note. Vous pouvez renommer cette feuille 1 en NotesPremierSemestre : cliquez à gauche sur le nom Feuille 1 en bas de la feuille puis choisir « Renommer ».

Donner et utiliser un Nom

Montrons ci-dessous comment nommer « Admissibilité » toutes les cellules de la colonne C5. Sélectionner la colonne puis Aller dans Insertion/Nom/Définir (ou bien CTRL+F3).

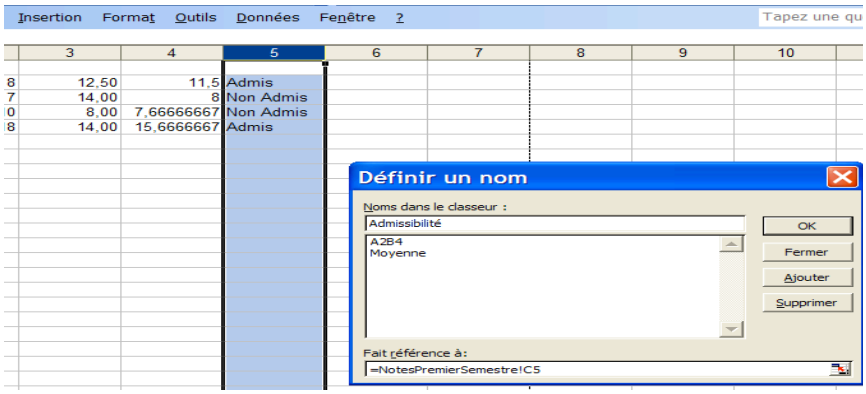


Figure 14

De la même manière, les cellules de la colonne C4 sont définies avec le nom « Moyenne ». Nous pouvons alors utiliser ce nom dans les fonctions en remplacement de la référence à la cellule. La formule d'admissibilité de L2C5 est modifiée en remplaçant LC(-1) par Moyenne ; cette formule est propagée dans les lignes du dessous dans la colonne C5.

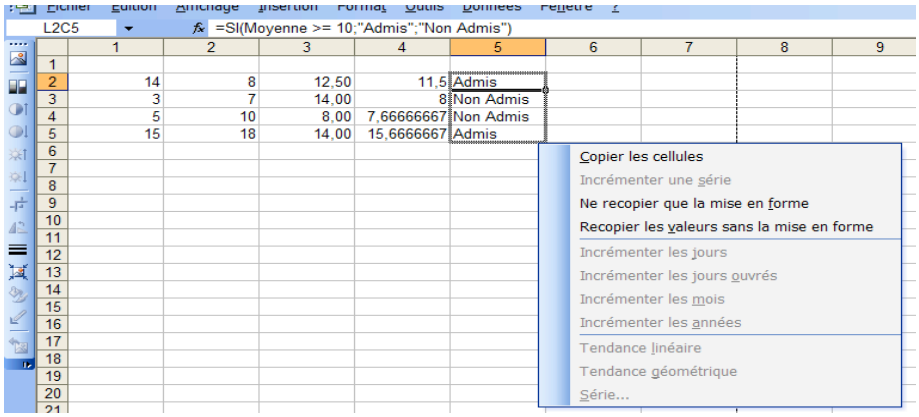


Figure 15

On peut de la même façon ne nommer qu'une cellule ou une ligne.

Notes : - pas de distinction entre majuscule et minuscule.
- utilisent par défaut des références absolues aux cellules.

Etiquettes pour les formules

Il est possible d'utiliser les titres, appelées *étiquettes*, figurant en tête des lignes ou des colonnes. Par défaut Excel ne reconnaît pas les étiquettes dans les formules. Pour cela, dans Outils/Options/Calcul activer **Accepter les étiquettes dans les formules**.
Dans notre exemple, supprimons le nom Moyenne et étiquetons la colonne C5.

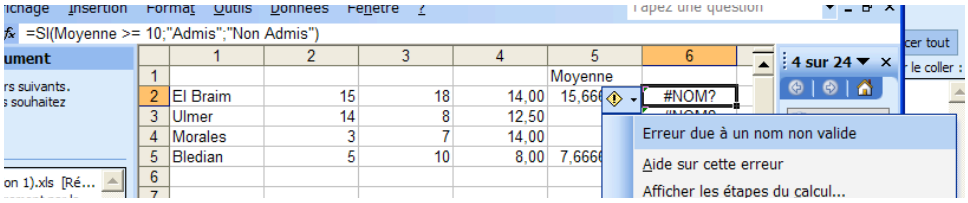


Figure 16

Excel considère que le « Moyenne » du **Si** est encore le nom. La **balise active** nous éclaire sur cette erreur. Il faut ressaisir Moyenne dans le **Si**.

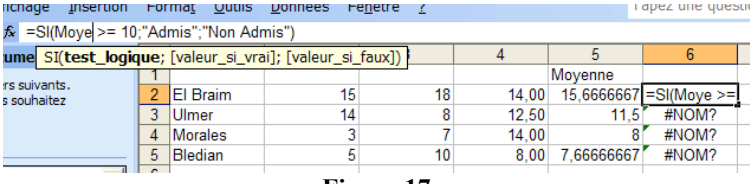


Figure 17

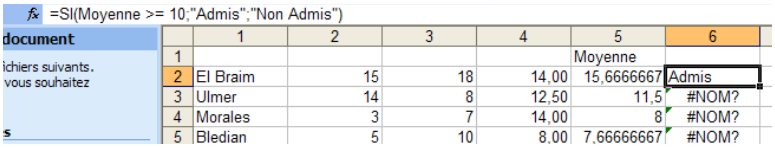


Figure 18

Une fois la formule propagée (voir au-dessus), nous obtenons le tableau suivant :

				Moyenne	
El Braim	15	18	14,00	15,6666667	Admis
Ulmer	14	8	12,50	11,5	Admis
Morales	3	7	14,00	8	Non Admis
Bledian	5	10	8,00	7,66666667	Non Admis

Notes : - voir aussi Insertion/Nom/Etiquette… pour ajouter une plage d'étiquettes.
- il est possible de rentrer un nombre en tant que texte ou étiquette.

Les tris prédéfinis

Nous avons inséré une colonne afin d'y rentrer les noms des élèves
(Insertion dans menu contextuel de la colonne 1) ; cette opération aurait pu se faire après le tri.
Nous désirons trier les élèves selon la colonne (du haut vers le bas) des moyennes (colonne 5). Nous sélectionnons auparavant la zone de tri puis Données/Trier. Le tri s'opère automatiquement. Si des notes sont modifiées, il s'appliquera avec la modification.

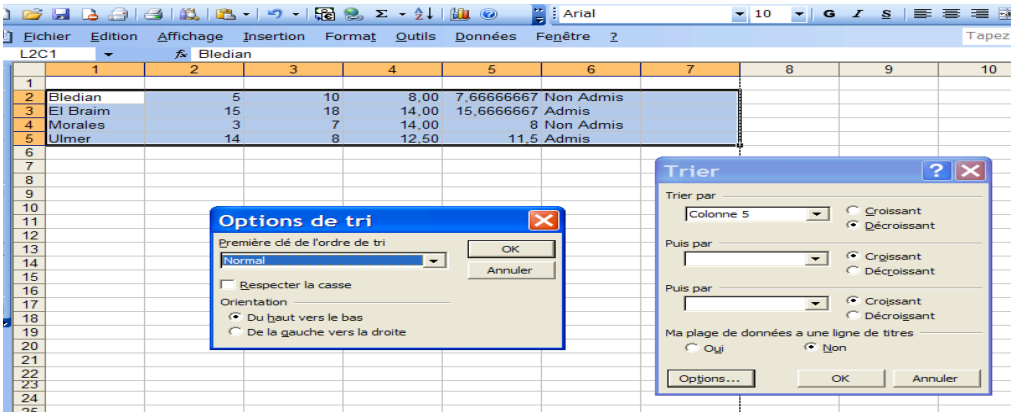


Figure 19

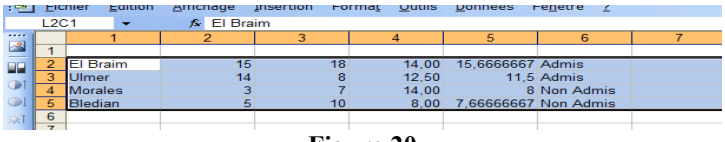


Figure 20

Exercice. Trier en fonction de l'ordre alphabétique des noms d'élèves.

Les graphiques

Réalisons un histogramme des élèves. Pour ce faire, nous utilisons les graphiques.

Pour qu'une feuille graphique se réalise automatiquement, il faut sélectionner la plage de cellules puis taper sur la touche F11. Pour obtenir des graphiques personnalisés nous allons procéder comme décrit par la suite.
Nous allons dans Affichage/Barre d'outils/Graphique :

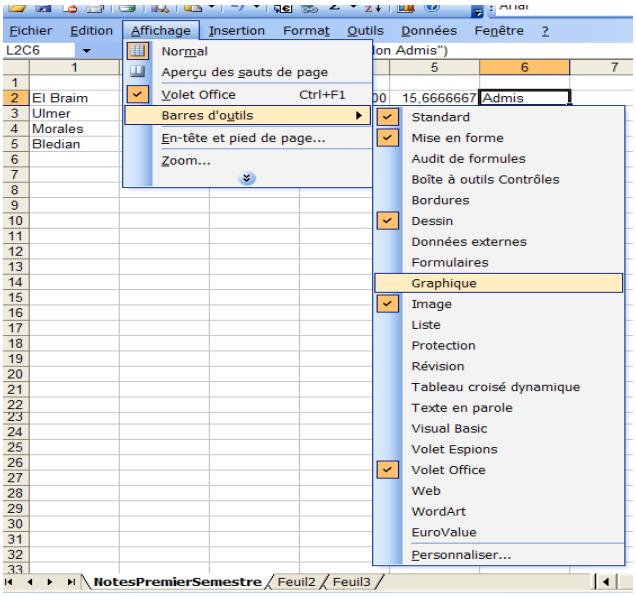


Figure 21

Créons une zone graphique en cliquant sur l'onglet « type graphique » et déplaçons cette zone dans la feuille à l'aide du Copier/Coller.

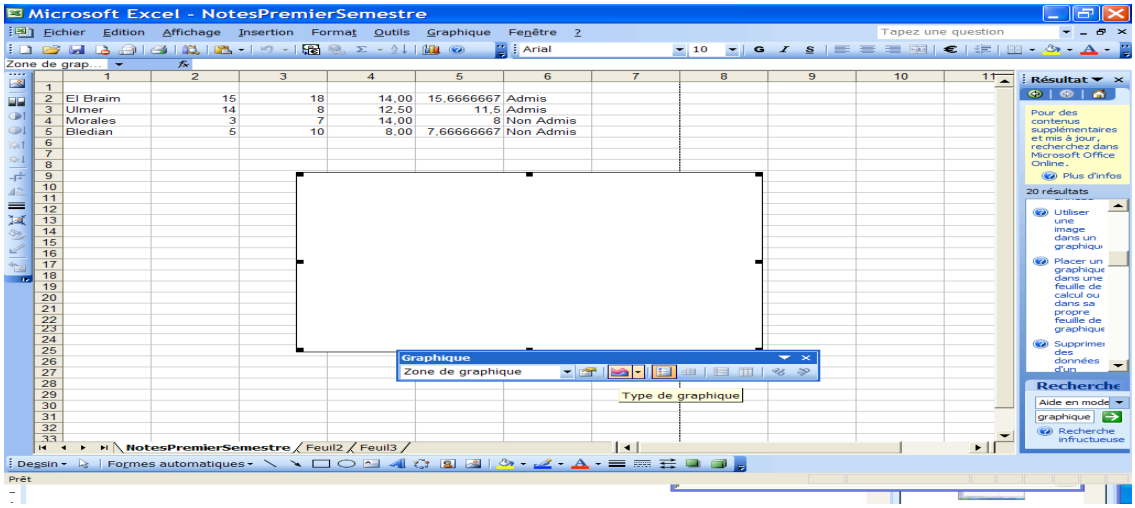


Figure 22

Nous rentrons dans le menu contextuel de cette zone :

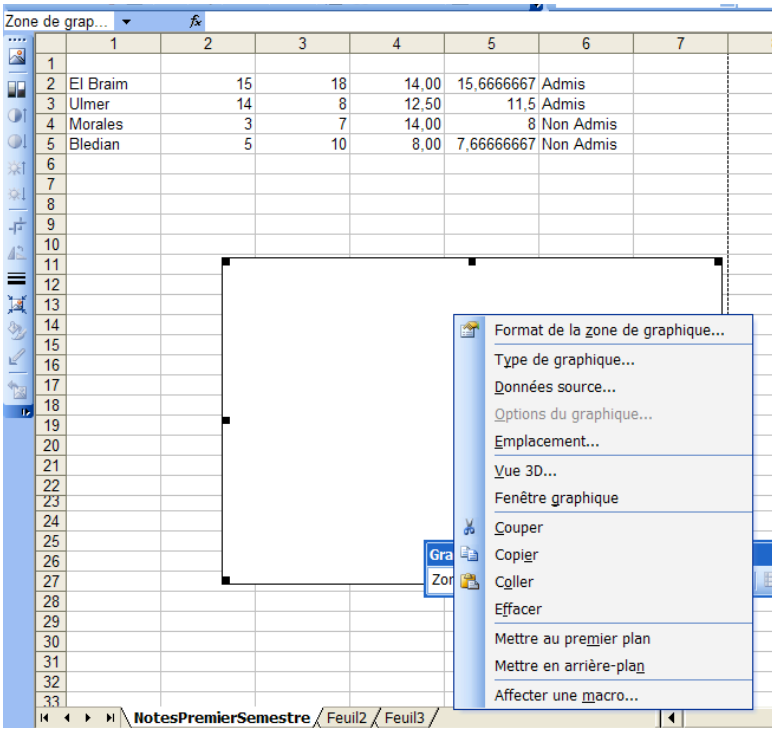


Figure 23

Nous choisissons notre type graphique :

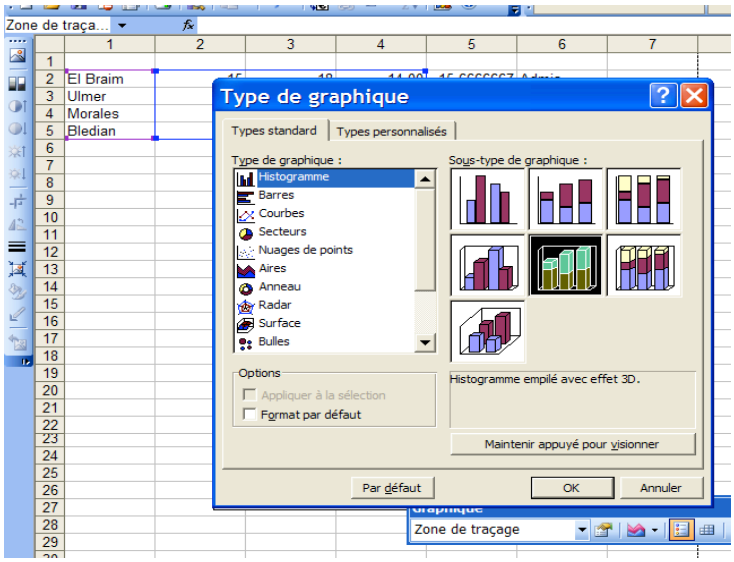


Figure 24

Nous sélectionnons notre plage de données (**Données Sources**) :

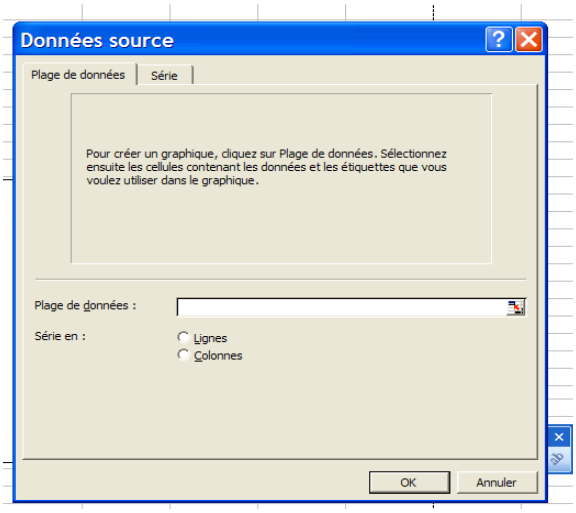


Figure 25

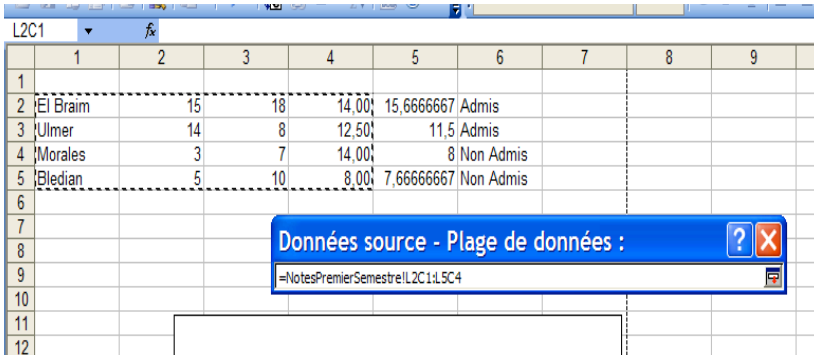


Figure 26

Lorsque la plage de données est sélectionnée (faire Entrée), une fenêtre représentant le graphique apparaît. Choisir série pour sélectionner, éventuellement, l'axe des abscisses. Si celui proposé par défaut convient alors faire OK sinon le sélectionner (toujours de la même manière : à la souris ou en rentrant « à la main »).

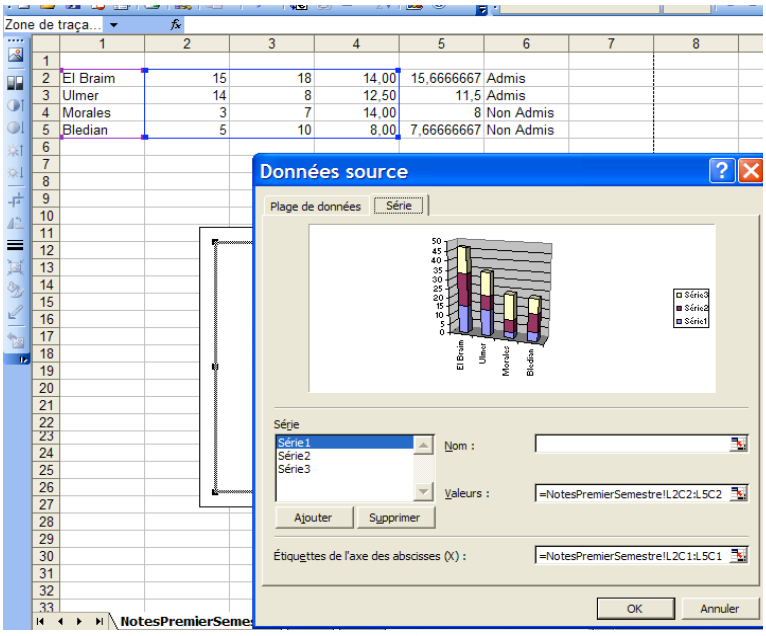


Figure 27

Terminer avec OK.

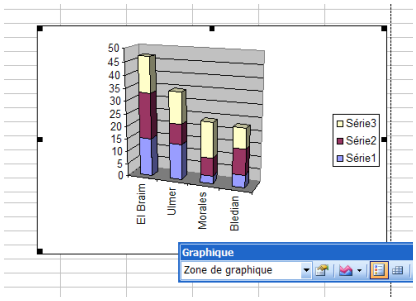


Figure 28

NOTES

1. Copier/Coller : Seules les formules sont copiées. Pour copier seulement une valeur, garder

Le clic droit enfoncé en vous mettant sur le bord de la cellule, déplacez-vous sur la cellule devant recevoir la valeur, lâchez le clic et choisissez ce que vous voulez copier. Pour désactiver la cellule que vous avez copiée, faites ENTREE.

2. Faire afficher les dépendances : OUTILS/AUDIT DES FORMULES
3. Faire afficher les formules à la place des résultats : dans OUTILS/AUDIT DES FORMULES choisir **Mode Audit des formules** (ou bien directement CTRL+ ").
4. Ajuster une colonne … : voir FORMAT
5. Se déplacer parmi des cellules sélectionnées : TAB ; MAJ + TAB ; ENTREE ; MAJ ENTREE

EXERCICES (à rendre impérativement au prochain cours)

1. Utilisez la propagation pour faire afficher tous les mois de l'année. Attention : avec la propagation, certaines données peuvent être effacées.
2. Réalisez un histogramme sur la moyenne générale des élèves.
3. Générez dans une colonne des nombres aléatoires ; puis dans la colonne suivante calculez -1/2*ln(cellule précédente). Dans une cellule, calculer le max des valeurs de la deuxième colonne. Puis recopier cette dernière cellule pour calculer le maximum de nouvelles valeurs (la fonction alea() est recalculée automatiquement). Que ce passe-t-il ? Que proposez-vous pour y remédier ?
4. Utilisez le cours pour créer les tables d'addition et de multiplication dans le corps Z/7Z. Vous utiliserez la propagation pour inscrire sur une ligne 0,…,6 et de même sur la colonne. Vous utilisez les fonctions (la fonction MOD qui retourne le reste de la division entière) pour calculer i+j modulo 7 (idem pour *). Vous propagerez ensuite la formule. Vous changerez aussi le Format des colonnes avec le mode audit des formules. Vous n'avez donc pas grand-chose à faire. Pour l'addition vous obtiendrez le tableau TAB 1.
5. Refaire l'exercice précédent en nommant la ligne des titres et la colonne des titres. La formule sera alors le produit de ces noms modulo 7.

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Chapitre 2 Excel et la statistique

Matrices

Reprenons notre exemple de notes d'élèves.
Nous recherchons la fréquence des notes avec les 7 intervalles :
[0,5],]5,7],]7,10],]10,12],]12,15],]15,18],]18,20].
Nous devons sélectionner l'espace de cellules où apparaîtra le résultat sous forme de vecteur colonne dans N^7. Nous tapons = dedans car la **forme matricielle** que nous nous apprêtons à calculer est le résultat d'une fonction, la fonction **FREQUENCE**. Le symbole = a pour effet de rendre accessibles des fonctions à travers la **zone nom**. La fonction FREQUENCE est une de celles proposées par défaut.

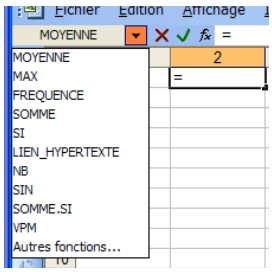


Figure 29

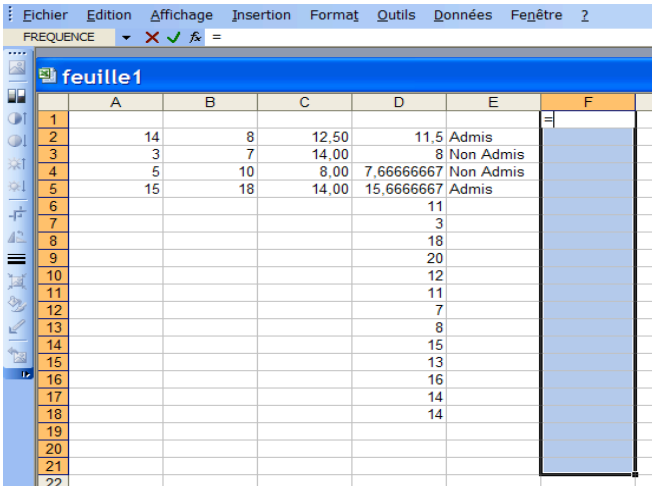


Figure 30

Dans la **zone nom**, sélectionnons la fonction **FREQUENCE**. Nous rentrons le tableau des données (i.e. **Tableau_données**) ; comme elles sont dans la feuille, nous les sélectionnons à la souris pour le déterminer. Nous rentrons la **Matrice_intervalles** « à la main » mais nous aurions pu utiliser la même méthode que pour le tableau des données si ces valeurs avaient été celles de cellules adjacentes.

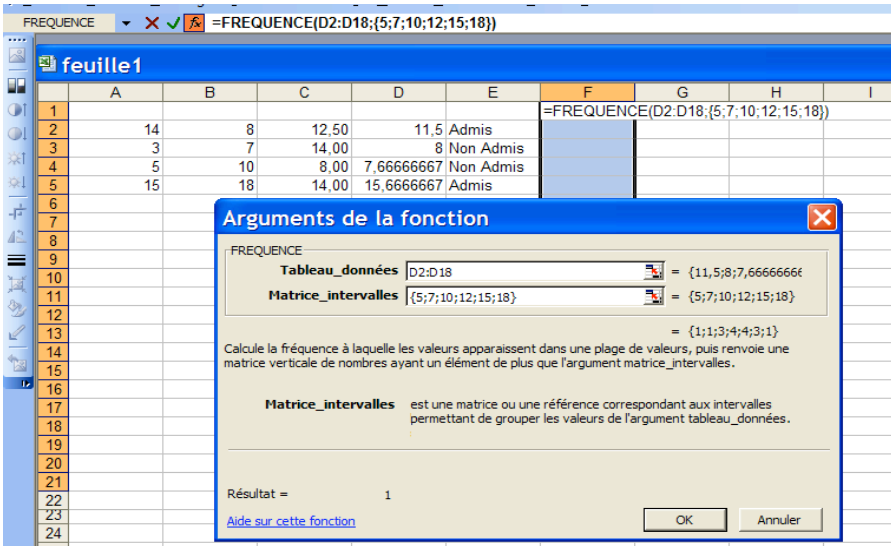


Figure 31

Pour que cette fonction soit enregistrée comme une forme matricielle, il faut valider avec CTRL + MAJ + ENTREE. Le contenu de la barre des formules est entouré d'accolades. Il y a 1 note inférieure à 5, 1 note dans]5,7], 3 dans]7,10],…

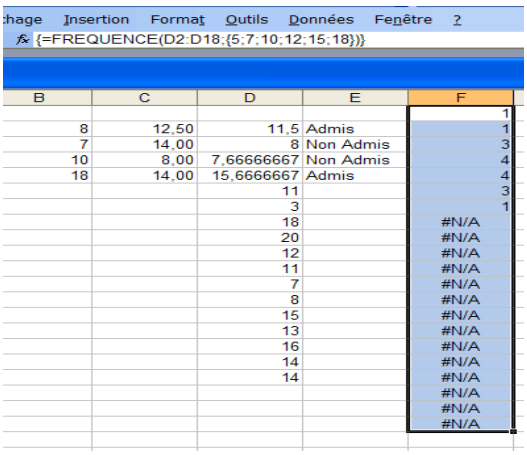


Figure 32

Notes : - les paramètres des fonctions sont séparés par des points virgules.
- nous avions sélectionné trop de lignes. Vous ne pourrez plus rien y faire. La dimension restera fixe. Si vous chercher à changer une cellule de la matrice et qu'Excel vous l'interdit, vous serez certainement amené à vous servir de la croix rouge de la barre des formules pour vous débloquer.

Nous pouvons alors construire l'histogramme. Nous introduisons les valeurs 5,7,10,12,15,18 dans G1 :G6. Nous sélectionnons la plage de données à la souris (i.e. G1 :F7). Avec Insertion/Graphique (ou bien F11), nous choisissons le premier type d'histogramme. Dans l'assistant, nous choisissons des titres.

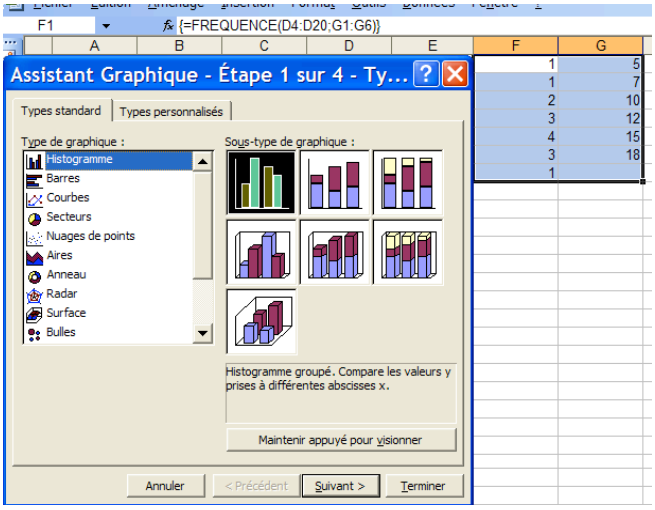


Figure 33

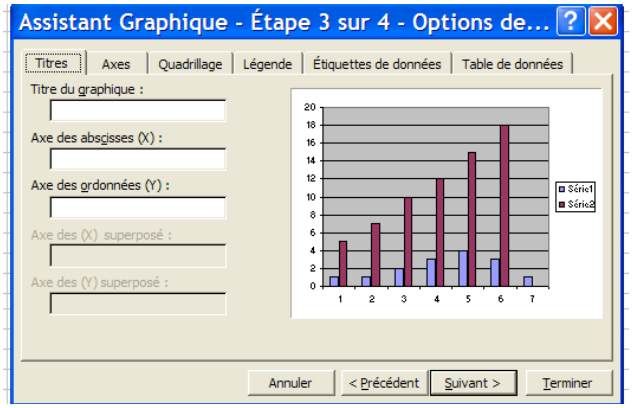


Figure 34

Nous sélectionnons la série 1. Nous indexons les intervalles (i.e. en ne prenant que le vecteur fréquence F1 :F7 comme donnée) ; puis nous tenons compte des intervalles (en sélectionnant G1 :G6 pour les abscisses). Nous supprimons la série 2.

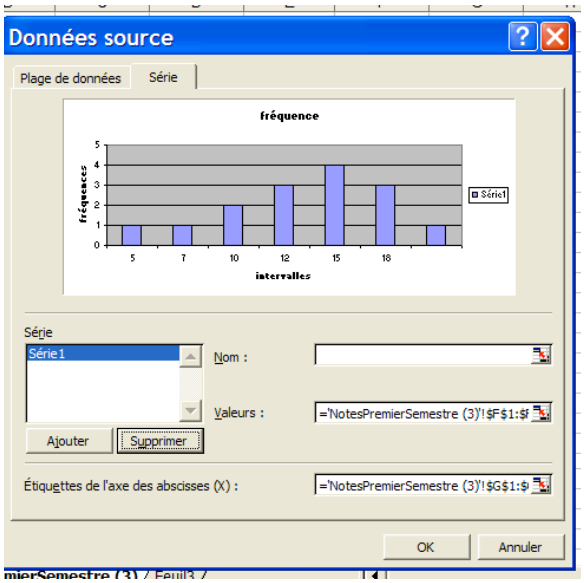


Figure 35

Les différents formats de la représentation graphique :
Nous désirons modifier la largeur des intervalles sur le graphique.
Dans la barre graphique, nous sélection notre série : **série 1**. Puis nous cliquons sur l'onglet Format de la série de données pour faire apparaître son format.

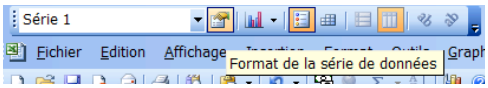


Figure 36

Nous choisissons 0 dans **Options/Largeur d'intervalle**.

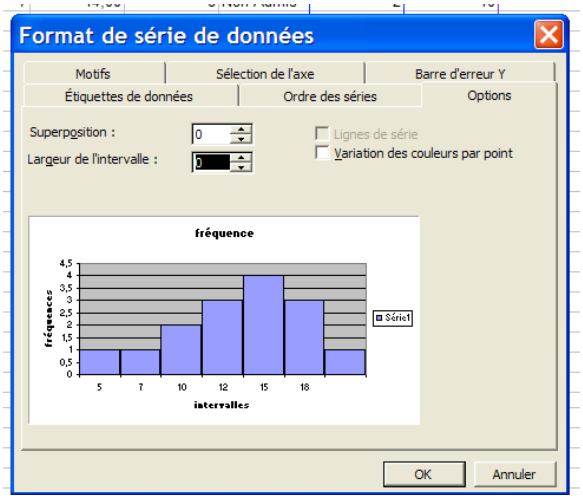


Figure 37

Le Solveur

Excel dispose d'un solveur permettant de résoudre des systèmes linéaires d'équations et des programmes d'optimisation. La feuille de calcul doit contenir :

- Des cellules qui contiendront les valeurs des variables en fin de résolution (les **cellules variables**)
- Des cellules fonctions contenant les fonctions des variables ; ce seront soit des fonctions objectif, soit des contraintes.

Voici donc ce que devra contenir votre feuille de calcul pour résoudre le système :
X+3Y= 12 et –X+4Y=-1.

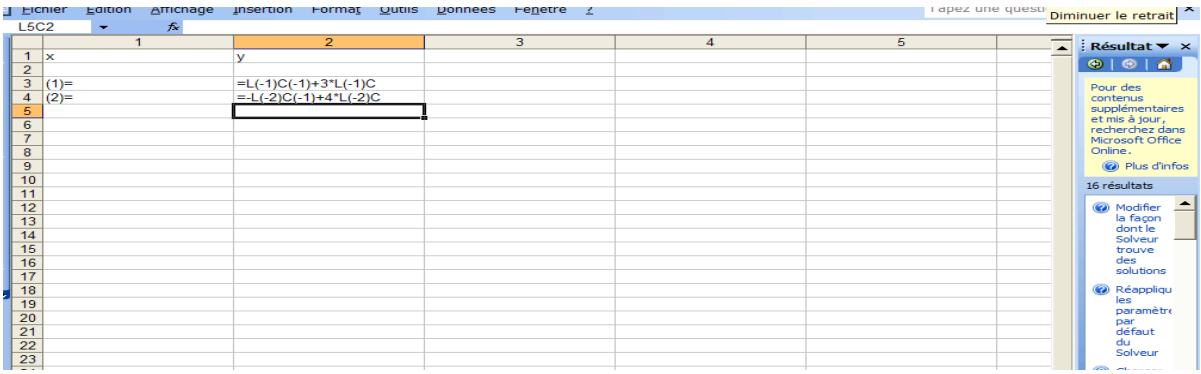


Figure 38

Ensuite dans **Outils/Solveur** définissez la cellule cible, les cellules variables et les contraintes.

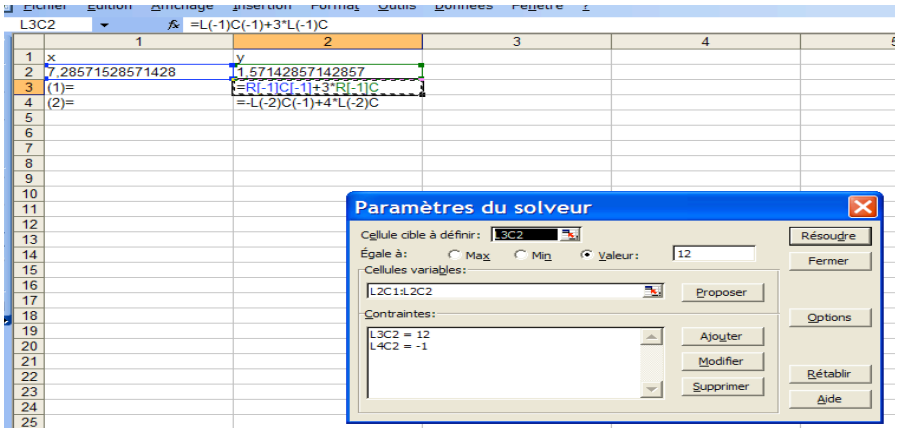


Figure 39

Note : Si **Solveur** n'est pas dans **Outils**, l'installer avec **Outils/Macros Complémentaires**. Tapez Solveur dans **Recherche** pour en savoir plus.

Graphe de dépendances

Un graphe orienté est composé de *sommets* et d'*arcs orientés*. Si *a* et *b* sont des sommets, tels que *a* →*b* soit un arc, *a* est appelé un *prédécesseur* (i.e. un **antécédent**) de *b* et *b* est appelé un *successeur* (i.e. un **dépendant**) de *a*. Un *chemin* dans un graphe orienté est une suite de sommets tels qu'il existe toujours un arc d'un élément de la suite vers le suivant.
Un *cycle* est un chemin dont le premier et le dernier élément sont identiques.

Nous avons vu que des cellules dépendent d'autres cellules. Cela peut se traduire par un *graphe de dépendances*. Les graphes de dépendances sont orientés et heureusement acycliques.

Pour faire apparaître les cellules dont une cellule dépend et celles qui en dépendent, nous irons dans **Outils/Audit des Formules**. Pour simplifier les manipulations, sélectionnons **Afficher la barre d'outils Audit des formules**.

Pour faire afficher toutes les dépendances : entrons = dans une cellule vide puis sélectionnons la zone de cellules pour laquelle doivent s'afficher les dépendances et terminons en cliquant autant de fois que nécessaire (pour faire apparaître les dépendances des différents niveaux) sur **Repérer les Antécédents**.

Note. Quand on modifie une formule, il faut mettre à jour le graphe de dépendance.

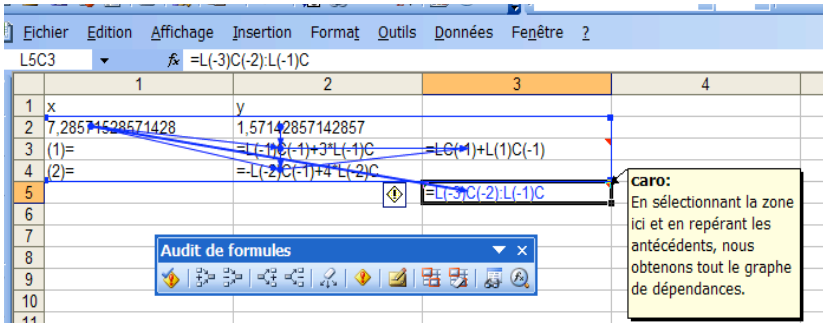


Figure 40

Note. Pour afficher un **commentaire**, dans la barre **Audit de formules** choisir **Nouveau Commentaire** ou bien dans le menu contextuel ou bien avec la touche ALT+F2. Pour afficher ou masquer des commentaires et leurs indicateurs, aller dans Outils/Options/Affichage. Pour effacer un commentaire, sélectionner la cellule puis aller dans **Edition/Effacer/commentaire**.

Les fonctions statistiques d'Excel

Un certain nombre de fonctions ont déjà été étudiées.
Certaines, comme la somme, sont directement applicables depuis la barre d'outil standard (symbole ∑), d'autres depuis la zone nom (ex. SI, SOMME.SI, FREQUENCE, MOYENNE, SOMME, **Autres fonctions…**), d'autres depuis **Outils/Macro** (voir plus loin) ou bien **Outils/Macros Complémentaires** (voir **Complément Solveur**). Si la syntaxe de la fonction est connue, il est toujours possible de la saisir dans la barre des formules.

Voici comment accéder aux fonctions prédéfinies :
Dans la **zone nom** sélectionner **Autres fonctions…** et dans **Insérer une fonction** sélectionner La catégorie **Statistiques** ou **Finances** :

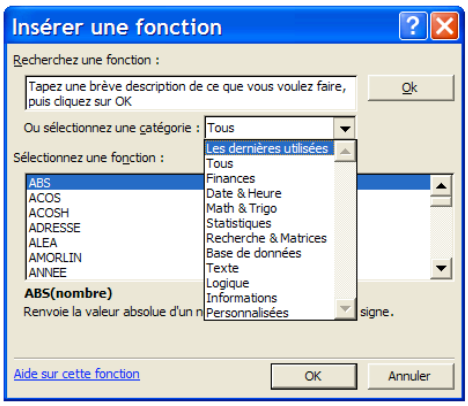
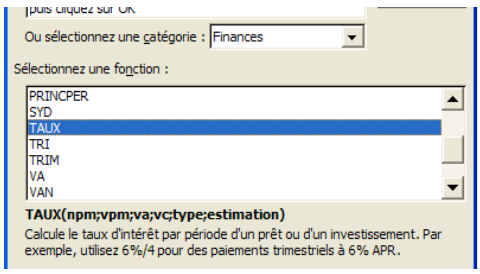


Figure 41



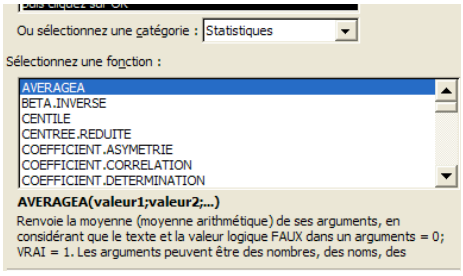
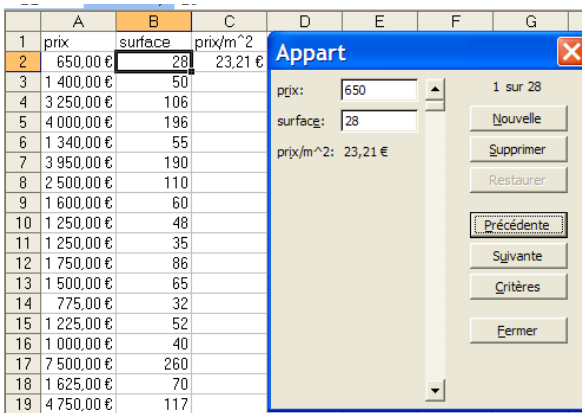


Figure 42

Les Bases de données

Utilisons les données d'appart.xls. Nous avons la colonne étiquetée prix, la colonne étiquetée surface et nous rajoutons une colonne (à calculer avec = A2 /B2) : prix/m^2. Allons dans **Données/Formulaire** :



	prix	surface	prix/m^2
2	650,00 €	28	23,21 €
3	1 400,00 €	50	
4	3 250,00 €	106	
5	4 000,00 €	196	
6	1 340,00 €	55	
7	3 950,00 €	190	
8	2 500,00 €	110	
9	1 600,00 €	60	
10	1 250,00 €	48	
11	1 250,00 €	35	
12	1 750,00 €	86	
13	1 500,00 €	65	
14	775,00 €	32	
15	1 225,00 €	52	
16	1 000,00 €	40	
17	7 500,00 €	260	
18	1 625,00 €	70	
19	4 750,00 €	117	

Figure 43

Nous passons d'une fiche à la suivante ; nous pouvons modifier des données sauf celles déjà calculées ; balayer les fiches selon des critères :

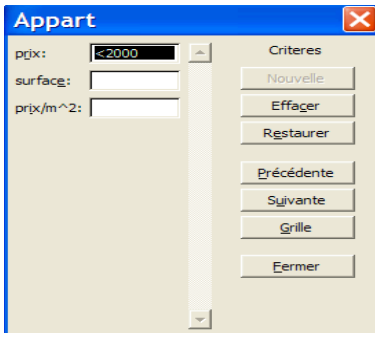
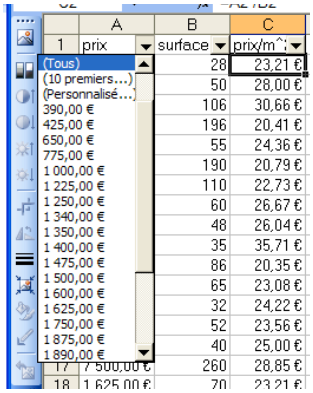


Figure 44

EN sélectionnant **Données/Filtrer/Filtre automatique**, Excel place des flèches de liste déroulante sous les étiquettes de la base :



	prix	surface	prix/m^2
2	650,00 €	28	23,21 €
3	1 400,00 €	50	28,00 €
4	3 250,00 €	106	30,66 €
5	4 000,00 €	196	20,41 €
6	1 340,00 €	55	24,36 €
7	3 950,00 €	190	20,79 €
8	2 500,00 €	110	22,73 €
9	1 600,00 €	60	26,67 €
10	1 250,00 €	48	26,04 €
11	1 250,00 €	35	35,71 €
12	1 750,00 €	86	20,35 €
13	1 500,00 €	65	23,08 €
14	775,00 €	32	24,22 €
15	1 225,00 €	52	23,56 €
16	1 000,00 €	40	25,00 €
17	7 500,00 €	260	28,85 €
18	1 625,00 €	70	23,21 €

Figure 45

Choisissons avec un filtre personnalisé les appartements à moins de 21 euros de mètre carré :

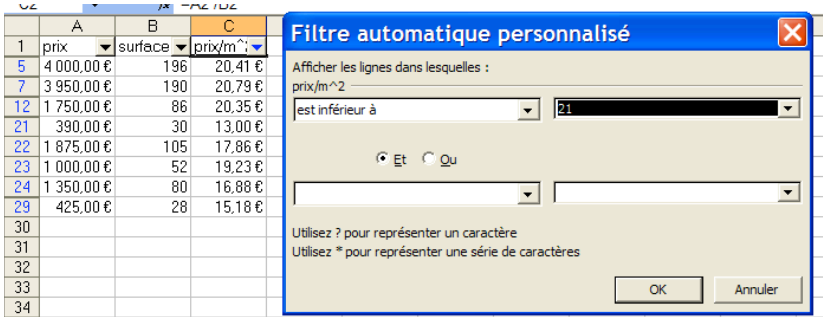


Figure 46

Pour annuler tous les critère à la fois : **Données/Filtrer/Afficher tout**.

Pour élaborer des filtres plus complexes : recopier la ligne (resp. colonne) des étiquettes (laisser au moins une ligne d'espace) ; définir les critères en dessous ; cliquer sur une des cellules de la plages (ici $A$1:$C$29) puis sélectionner **Données/Filtrer/Filtre élaboré** :

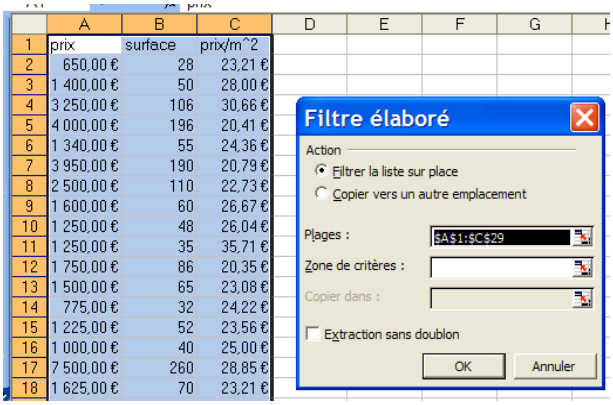


Figure 47

Figure 48

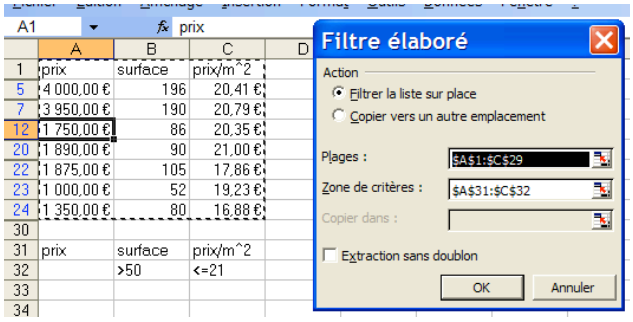


Figure 49

Attention : les numéros de lignes n'ont pas changés. Essayons de calculer la moyenne des valeurs filtrées dans la colonne C en les sélectionnant à la souris : nous trouvons une moyenne 23 euros 68 qui est supérieure à 21. C'est la moyenne des cellules C5 à C24, même celles non apparentes, qui est faite.

Les tableaux croisés

Ils constituent une méthode dynamique d'analyse et d'exploitation des données saisies. L'Assistant utilise des données d'une feuille de calcul ou d'une liste ou d'une liste de données existante ; on peut exploiter des données provenant de sources externes (Access, dBase, …). Les domaines d'application les plus courants sont : les rapports de suivi de ventes, les analyses de stock, les statistiques du personnel, l'exploitation de relevés statistiques,…

Vous pourrez retrouver une partie de ce qui suit dans le livre Excel 2003 de Jack Steiner éditer chez Eyrolles.

Soit le tableau suivant :

Réf	Client	Date	Montant HT	TVA	Total TTC	Réglé
1	Al	15/09/2005	1 000,00 €	5,5%	1 055,00 €	Oui
2	Bm	26/09/2005	200,00 €	19,6%	239,20 €	Non
3	Cp	26/09/2003	100,00 €	19,6%	119,60 €	Oui
4	Al	28/09/2003	350,00 €	19,6%	418,60 €	Oui
5	Wn	29/09/2003	150,00 €	19,6%	179,40 €	Non
6	Fg	29/09/2003	550,00 €	19,6%	657,80 €	Oui
7	Rs	29/09/2003	600,00 €	19,6%	717,60 €	Oui
8	Cp	30/09/2003	700,00 €	19,6%	837,20 €	Oui
9	Mg	30/09/2003	1 500,00 €	5,5%	1 582,50 €	Non
10	Am	01/10/2003	820,00 €	5,5%	865,10 €	Non
11	Rcp	01/10/2003	120,00 €	5,5%	126,60 €	Oui
12	Wn	01/10/2003	650,00 €	19,6%	777,40 €	Non
13	Uf	02/10/2005	28,00 €	5,5%	29,54 €	Non

Avec **Données/Rapport de tableau croisé dynamique**, s'ouvre la boite de dialogue

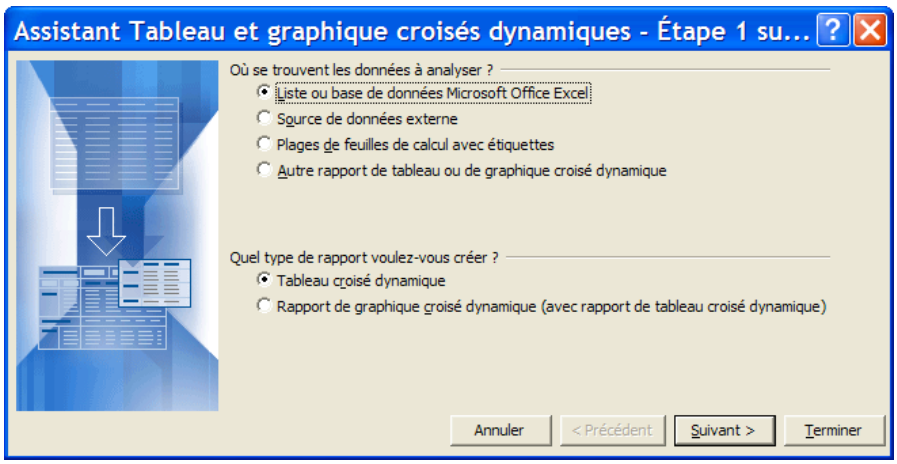


Figure 50

Nous sélection **Liste ou base de données Microsoft Office Excel** et **Tableau croisé dynamique** ; puis nous cliquons sur **Suivant**.
Dans la boite de dialogue suivante, une plage de cellule est proposée par défaut. C'est notre tableau qui est proposé. Nous choisissons ensuite si le tableau doit être réalisé dans le même feuille ou dans une autre. Nous arrivons alors à cette étape :

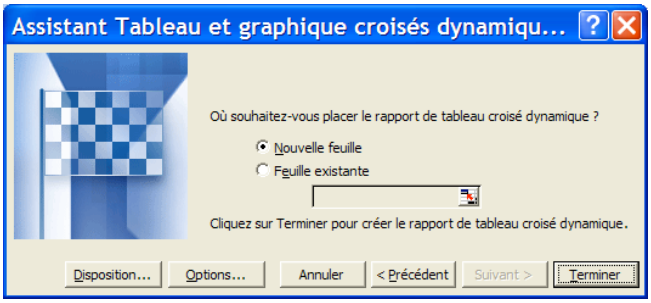


Figure 51

Si nous cliquons sur **Disposition…**, s'ouvre alors une boite de dialogue représentant le modèle du tableau dynamique. Nous faisons glisser les onglets Réglé, Client, Date et Montant HT, comme ci-dessous :

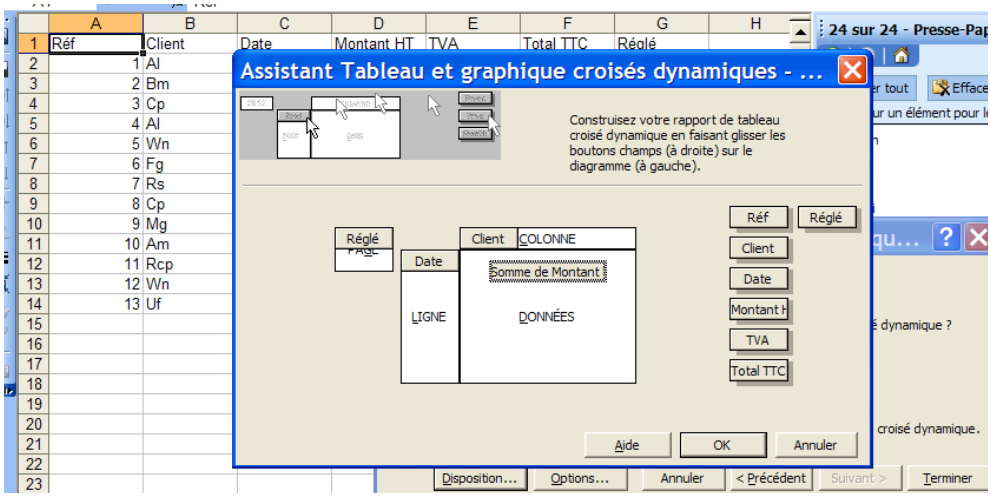


Figure 52

Note : Quand on sélectionne l'onglet Montant HT et qu'on le place dans la zone Données, nous obtenons automatiquement Somme de Montant.

Tapons OK ; puis Terminer.

Comme nous l'avons spécifié, nous obtenons dans une nouvelle feuille le tableau suivant :

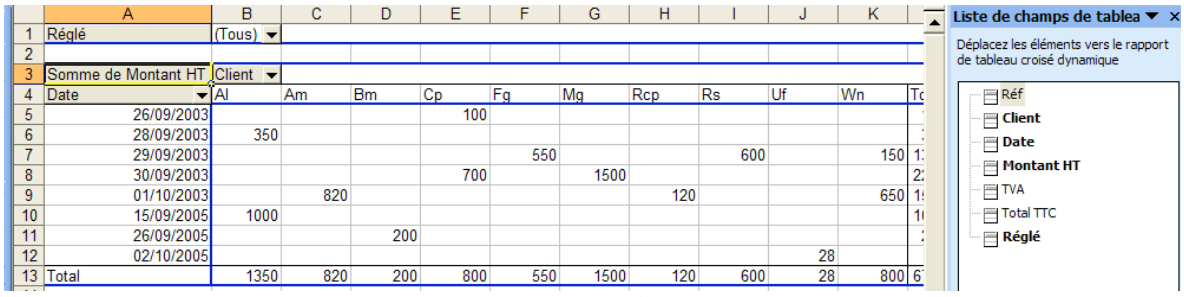


Figure 53

Apparaît également la barre d'outils des tableaux croisés dynamiques.

Figure 54

Comme nous le constatons lors du cours, il est impossible de modifier les valeurs dans ce tableau. Seul le tableau source est modifiable. Cependant, si nous modifions une valeur du

tableau source, ce tableau n'est pas actualisé. Il faut utiliser l'onglet **Actualiser les données** de la boite de dialogue ou par le menu contextuel du tableau dynamique :

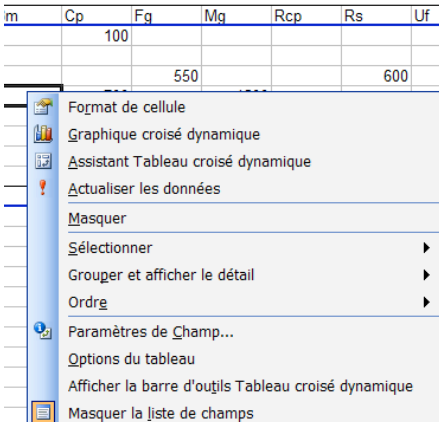


Figure 55

On peut faire glisser les champs et trier ainsi autrement (voir démo en cours) :

Somme de Montant HT		Client										
Réglé	Date	Al	Am	Bm	Cp	Fg	Mg	Rcp	Rs	Uf	Wn	Total
Non	29/09/2003										150	150
	30/09/2003						1500					1500
	01/10/2003		820								650	1470
	26/09/2005			200								200
	02/10/2005									28		28
Total Non			820	200			1500			28	800	3348
Oui	26/09/2003				100							100
	28/09/2003	350										350
	29/09/2003					550			600			1150
	30/09/2003				700							700
	01/10/2003							120				120
	15/09/2005	1000										1000
Total Oui		1350			800	550		120	600			3420
Total		1350	820	200	800	550	1500	120	600	28	800	6768

Note : nous verrons en cours que nous pouvons aussi choisir la disposition après en avoir terminer avec l'assistant.

Les Macros

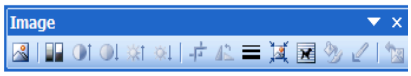
Excel vous offre la possibilité d'utiliser des *macros.*

En programmation, une macro diffère d'une fonction, en ce sens que c'est le texte de son corps qui est d'abord évalué. Une fonction prend des paramètres réels et selon son processus de liaison avec les paramètres formels, elle les associe, puis évalue son corps dans l'environnement de cette association. Pour les macros, c'est différent. Une pré-évaluation du corps est effectuée avant la première exécution ; ce qui empêche toute association des paramètres formels aux valeurs des paramètres réels qui à ce stade sont inconnus. Selon les langages, les macros prennent différentes formes et respectent différentes règles. Nous les étudierons dans la prochaine partie de ce cours.

EXERCICES (à rendre impérativement au prochain cours)

1. Calculer la moyenne, l'écart-type avec les modalités {0,1,2,…,20} et faire afficher l'histogramme de la fréquence pour des valeurs prises entre 0 et 20.
2. Faire le tour de ces fonctions et en appliquer quelques-unes. En particulier, utiliser **DROITEREG()**, **CROISSANCE**() et rendre des exemples personnels traités (travail individuel ; donc aucun exemple en commun).
3. Utilisez le solveur pour résoudre le problème : Max((x-y)(z+y)) sous les contraintes x+2y<3 ; x,y,z >0 ; x< 50 ;

Afin de faciliter votre CR d'exercices : utiliser **Edition/Presse-papier Office** permettant de récupérer des copies d'écran dans Word ou des copies d'Excel à Word. Pour rogner les images : **Affichage/Barres d'outils/Image** ; cliquer sur l'image puis utiliser l'onglet **rogner** de la barre Image en tirant sur les poignées situées aux quatre coins de l'image et au centre des cotés (poignées d'homothétie de l'image si on ne rogne pas).



Barre d'outils Image

Chapitre 3 Enregistrer et Modifier des macros

Note. A partir de ce chapitre, certaines parties seront inspirées du livre de Mikaël Bidault (EXEL & VBA 2003 2000/XP ; CampusPress ; www.pearsoneducation.fr).

Qu'est-ce qu'une macro en programmation ?

En programmation, une macro diffère d'une fonction, en ce sens que c'est le texte de son corps qui est d'abord évalué. Une fonction prend des paramètres réels et selon son processus de liaison avec les paramètres formels, elle les associe, puis évalue son corps dans l'environnement de cette association. Pour les macros, c'est différent. Une pré-évaluation du corps est effectuée avant la première exécution ; ce qui empêche toute association des paramètres formels aux valeurs des paramètres réels qui à ce stade sont inconnus. Selon les langages, les macros prennent différentes formes et respectent différentes règles.

Le *code* est le texte écrit dans le langage de programmation. Le *codage* est la génération du code.

Excel et les macros

Excel vous offre la possibilité d'utiliser des *macros.* Les commandes Excel (l'application hôte), les barres d'outils, les raccourcis clavier, les déplacements dans un classeur, la modification du classeur peuvent être enregistrées pour **coder** une macro.

Le codage d'une macro Excel est réalisable de deux façons différentes :

- Par enregistrement de macros ;
- avec la fenêtre code de Visual Basic Editor.

Ce chapitre s'appliquera à vous expliquer l'utilisation de l'enregistreur de macros.
Nous aborderons VBA (Visual Basic pour Application) dans les chapitres suivants après vous avoir initié à la philosophie des langages objets.

Enregistrement de macros

La méthode que nous allons suivre est :

A. Déclenchement de l'**Enregistreur de macros**.
B. Enregistrer le code de la macro.

A. *Déclenchement de **l'Enregistreur de macros***

1) Sélectionner une cellule (de référence) à laquelle sera attribuée la suite de commandes (i.e. le codage de la macro).
2) Avec **Outil/Macro/Nouvelle Macro** (ou bien Alt + F8) ouvrir la boite de dialogue **Enregistrer une macro** :

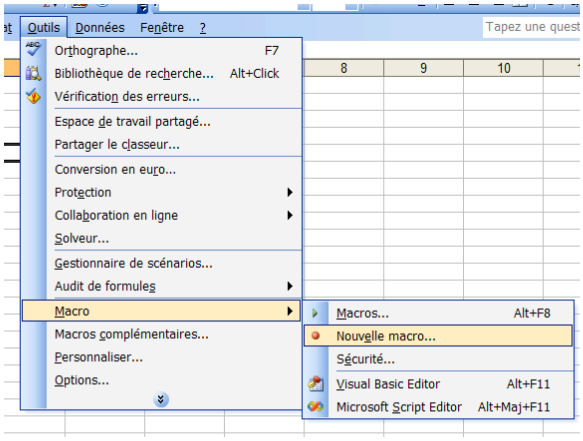


Figure 56

3) Choisir un nom (qui ait un sens).
4) Ecrire la description.
5) Saisir une lettre pour la **Touche de raccourci**. Si la lettre est b (resp. B), CTRL b (resp. B) déclenchera l'exécution de la macro. Attention de ne pas écraser une touche de raccourci déjà existante ! (on pourra la modifier ultérieurement en sélectionnant le nom de la macro dans Outils/Macro puis en cliquant sur Options…)

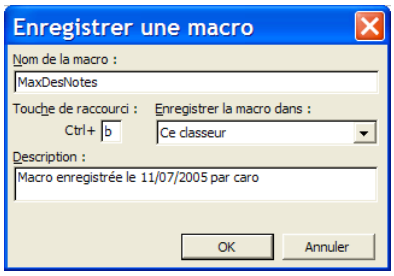


Figure 57

6) Cliquer sur OK. La barre d'outils **Arrêt de l'enregistrement** s'affiche à l'écran indiquant que l'enregistrement de la macro est commencé :

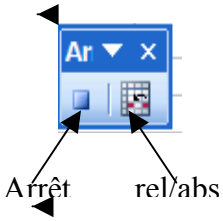


Figure 58

B. Enregistrer le code de la macro

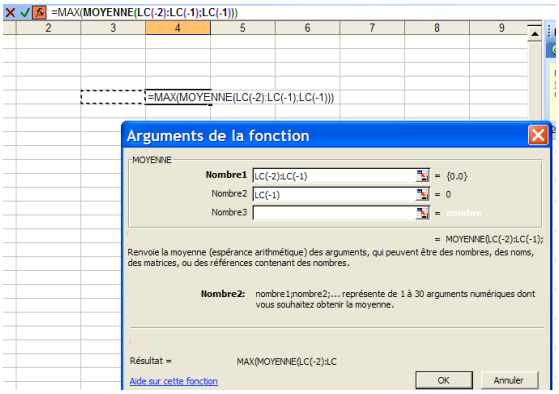


Figure 59

C. Exécuter la macro

Sélectionner la cellule à laquelle doit s'appliquer la macro et exécuter la macro dans Outil/Macro/Macros (ou bien, utiliser le raccourci CRTL b).

	1	2	3	4
1	Candidats	partiel	examen	
2	Alain	15	16	15,6666667
3	Xao	9	18	15
4	Juston	17	5	9
5	Ben	14	12	12,6666667
6	José	9	13	11,6666667
7		6	7	6,66666667

Figure 60

Notes

1. Sécurité et macros : Outils/Macro/Sécurité

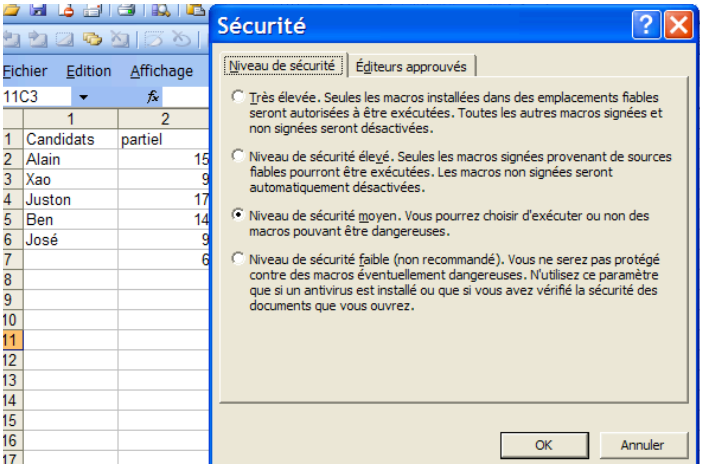


Figure 61

Voir le code de la macro en VBA

- Outils/Macro/
- Sélectionnez la macro
- Cliquez sur Modifier

Nous ouvrons ainsi Visual Basic et nous avons accès au code de la macro.

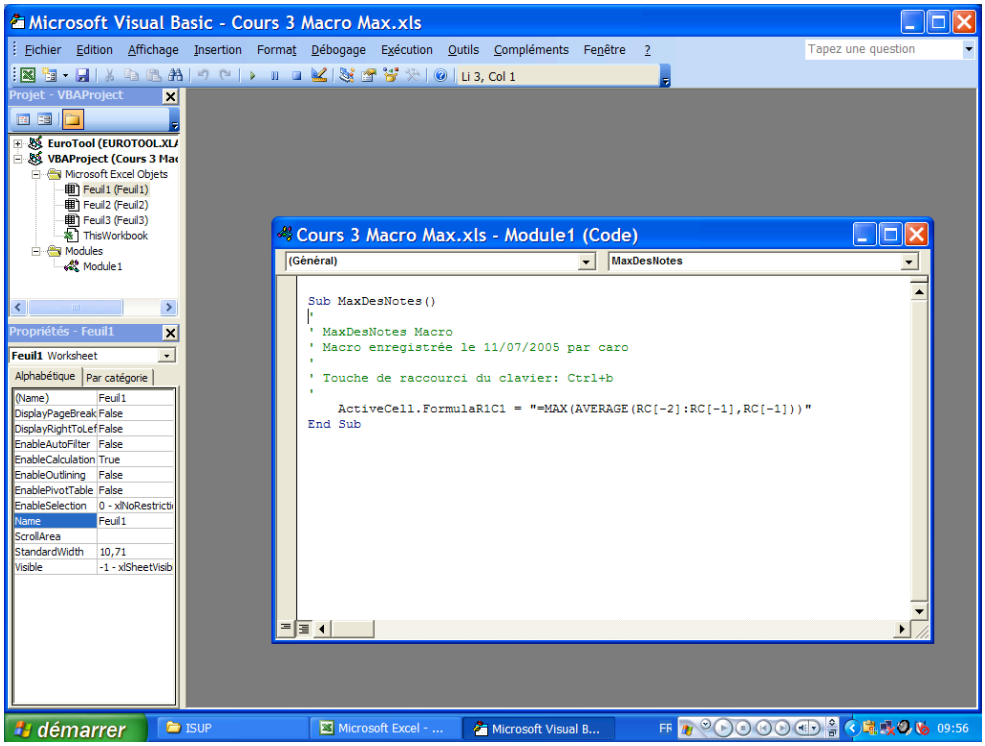


Figure 62

Nous constatons que

- le mode LiCi (RiCi, dans VBA) est repéré avec FormulaR1C1
- AVERAGE (pour Moyenne) est mal appliqué
- Que seule la cellule active (i.e. ActiveCell) peut accepter la macro ce qui nous empêche de l'exécuter à des plages de cellules sélectionnées (i.e. Selection).

Nous modifions simplement le code (voir démo en cours) pour obtenir le code suivant :

```
Sub MaxDesNotes()
'
' MaxDesNotes Macro
' Macro enregistrée le 11/07/2005 par caro
'
' Touche de raccourci du clavier: Ctrl+b
'
   Selection.FormulaR1C1 = "=MAX(AVERAGE(RC[-2]:RC[-1]),RC[-1])"
End Sub
```

qui s'applique à une plage de cellules (voir démo en cours). Vous serez souvent amenés à modifier des macros car elles ne réalisent pas exactement ce que vous attendiez.

Un autre exemple

Nous définissons la macro GrasItalique qui par la Touche CTRL+e transforme en gras et italique le contenu de la cellule sélectionnée. Nous procédons comme pour la macro précédente en cliquant sur le **G** et le *I* situés sur la barre d'outils standard. Il est possible d'appliquer cette macro à toute une plage de cellules sélectionnées.

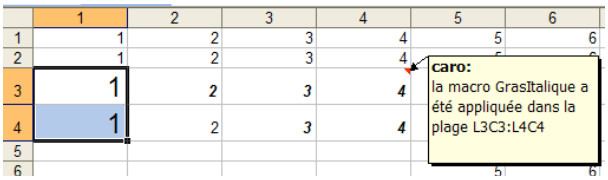


Figure 63

Si nous appliquons cette macro aux cellules L3C1 et L4C1 d'un autre format que celui de la cellule L3C2 de référence pour la création de la macro, leur format n'est pas modifié. S'il l'était, il faudrait modifier des lignes du corps de la macro sous Visual Basic. Voici le corps de cette macro

```
Sub GrasItalique()
' GrasItalique Macro
' Macro enregistrée le 24/08/2005 par caro
' Touche de raccourci du clavier: Ctrl+e
Selection.Font.Bold = True
Selection.Font.Italic = True
End Sub
```

Enregistrons désormais une macro appelée GrasItalique2 en choisissant **Gras Italique** dans la boite de dialogue Format/cellule :

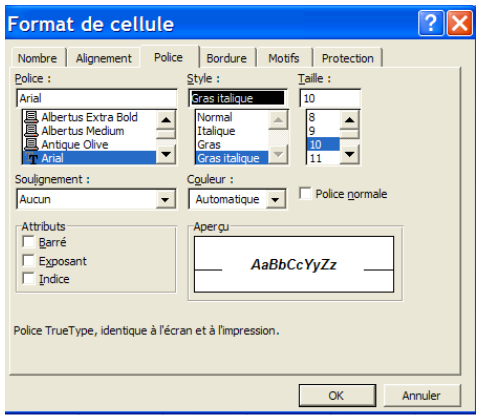
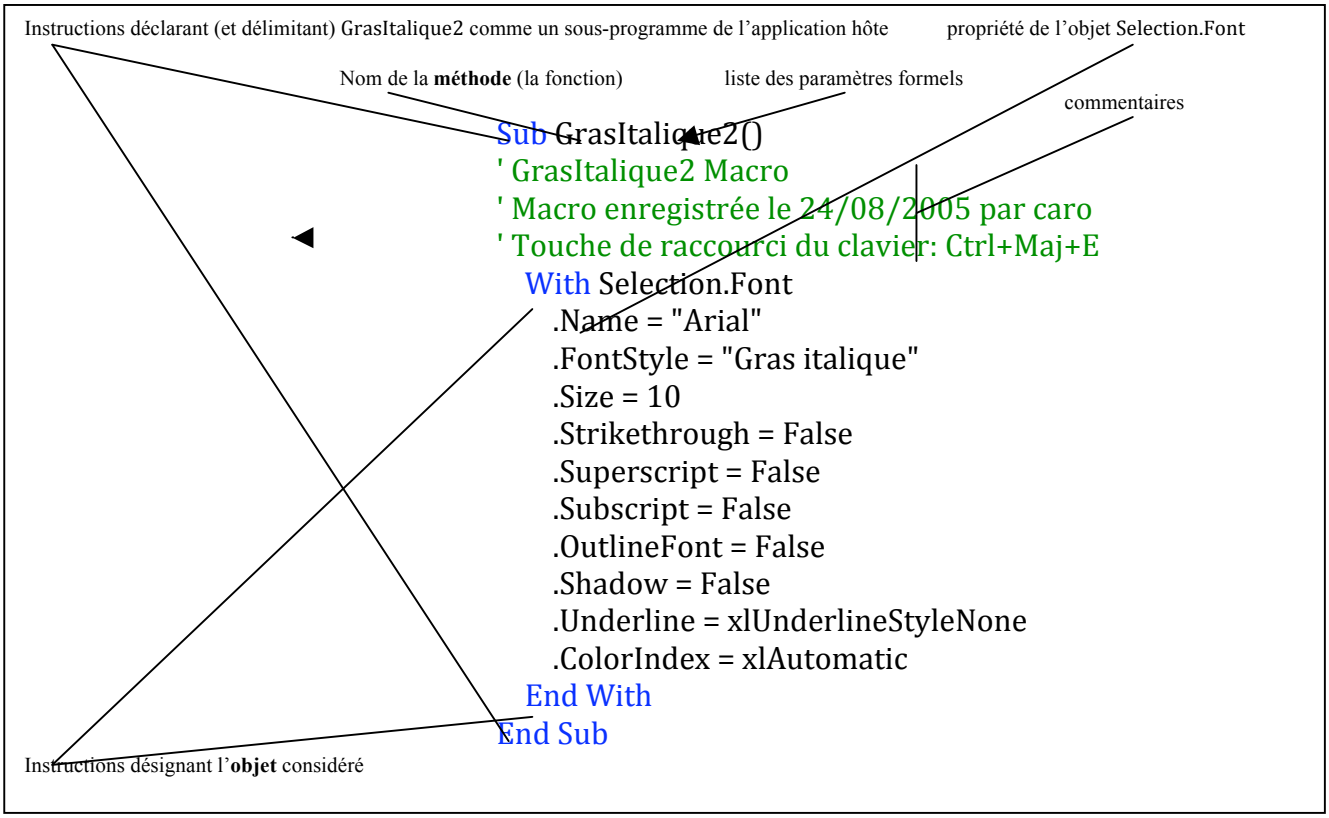


Figure 64

Si nous appliquons cette nouvelle macro à la cellule L3C1 dont la taille est de 18 points, sa taille est réduite à 10 points, celle la cellule de référence de définition de la macro GrasItalique2. Regardons le corps de cette macro :



Si nous ne gardons que .FontStyle = "Gras italique" comme *propriété* de l'*objet* Selection.Font les tailles ne seront plus modifiées.

Note : pas différence entre majuscules et minuscules : Font et font, c'est idem. VBA remplace automatiquement les initiales d'un mot en majuscule si ce mot est connu.

Accessibilité des macros

Macros personnelles (.xls)

Lors de l'enregistrement d'une macro, la portée est proposée : ce classeur (voir Figure 2). Ce classeur devra être ouvert pour toute action avec cette macro. Pour que la portée soit globale, il faut l'enregistrer dans un classeur de macros personnelles appelé PERSO.XLS. Ce classeur est créé lors de l'enregistrement de la première macro personnelle
ou plus exactement après avoir quitté Excel (démonstration durant le cours). Pour le faire afficher : **Afficher/Perso.xls**.
Consulter : C:\Documents and Settings\nom_login\Application Data\Microsoft\Excel\XLS-START\

Macros complémentaires (.xla)

Ce type de macros est adapté à la distribution. Les classeurs sont chargeables (pas ouverts) et donc accessibles au lancement de l'application.

Enregistrement de macros complémentaires

- dans VBA, ouvrir un module projet (ici MaxDesNotes) et choisir **Debogage/CompilerVBAProject**
- dans Excel, **Fichier/Propriétés/Résumé**

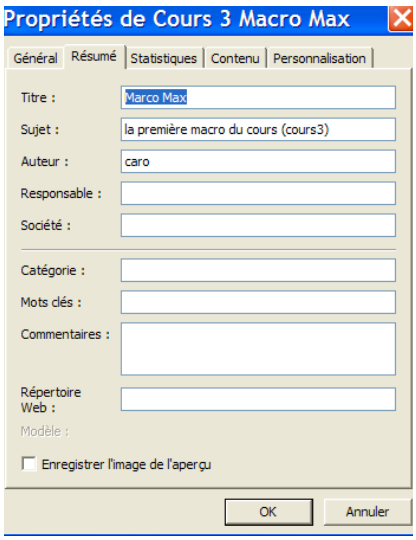


Figure 65

- choisir **Fichier/Enregistrer sous…** Dans la zone **Type de fichier**, sélectionner Macro complémentaire Microsoft Excel (*.xla). Le fichier Windows\Application Data\Microsoft\Macros complémentaires est activé par défaut.
- Choisir un nom et **Enregistrez**.

Activer/Désactiver une macro complémentaire

Dans Outils/Macros complémentaires…

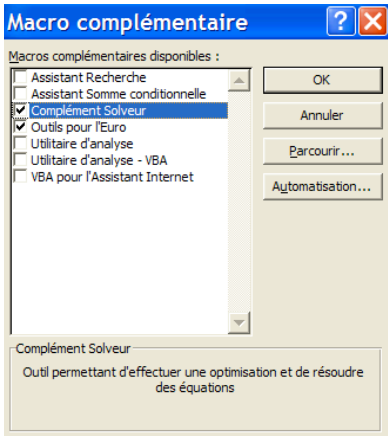


Figure 66

Note : si une macro complémentaire intégrée à Excel n'est pas disponible, il faut relancer l'installation d'Office et l'installer. (Windows : Démarrer/**Programmes/Ajout/Suppressions de programmes/Office/Ajouter/Supprimer des composants**).

EXERCICE 1 (à rendre)

Considérons le tableau suivant provenant du livre Méthodes Statistiques en Gestion de Michel Tenenhaus (Chapitre 4, DUNOD, 1994).

prix	surface
650	28
1400	50

3250	106
4000	196
1340	55
3950	190
2500	110
1600	60
1250	48
1250	35
1750	86
1500	65
775	32
1225	52
1000	40
7500	260
1625	70
4750	117
1890	90
390	30
1875	105
1000	52
1350	80
1475	60
4950	140
425	20
2475	100
425	28

En vous reportant au livre sus-cité, vous répondrez aux questions suivantes en détaillant ce que vous avez fait pour parvenir à vos résultats :

1. Calculez la droite des moindres carrés et comparez votre résultat à celui de la fonction prédéfinie DROITEREG.
2. Calculez (sans macros) les erreurs e_i et les leviers h_i (voir pages 60 et 63).
3. Sortir les graphiques de la figure 2, page 61.
4. Faire afficher le graphe des dépendances.
5. Refaites les questions 1 et 2 en définissant des macros et en faisant afficher leur code.
6. Enregistrez votre macro calculant la droite des moindres carrés comme macro complémentaire.

Vous afficherez les tableaux sous deux formes : celle sans les formules, celle avec les formules.

Note : une large utilisation des notions vues dans ce cours sera appréciée par le correcteur.

EXERCICE 2 (à rendre)

Revenons à l'exercice maximum des -1/2ln(alea()) du premier cours. Nous voulons automatiser cette action.

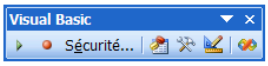
1. Enregistrez une macro CopieValeur qui réalise une copie valeur d'une cellule (contenant éventuellement une formule) dans une autre cellule.
2. Modifier le code de la macro afin de retirer tout ce qui est inutile et comprendre ainsi ce que chaque instruction signifie. Vous obtenez ainsi la macro CopieValeurMod.

3. Ecrire enfin la macro CopieValeurs qui recopie 20 fois la valeur de la cellule initiale (i.e. sans la formule) et l'appliquer à l'exercice sur alea() du cours 1. (Note : vous devrez chercher la syntaxe de la boucle For).

Chapitre 4 Etude des premiers programmes

Cette partie commence par un exposé introductif aux différents types de langages de programmation (compilé, interprété, typé ou non, interactif, objet,…) et tout particulièrement introduit les notion de classes, propriétés et méthodes des langages objets. Ensuite, nous abordons VBA à travers VB Editor.

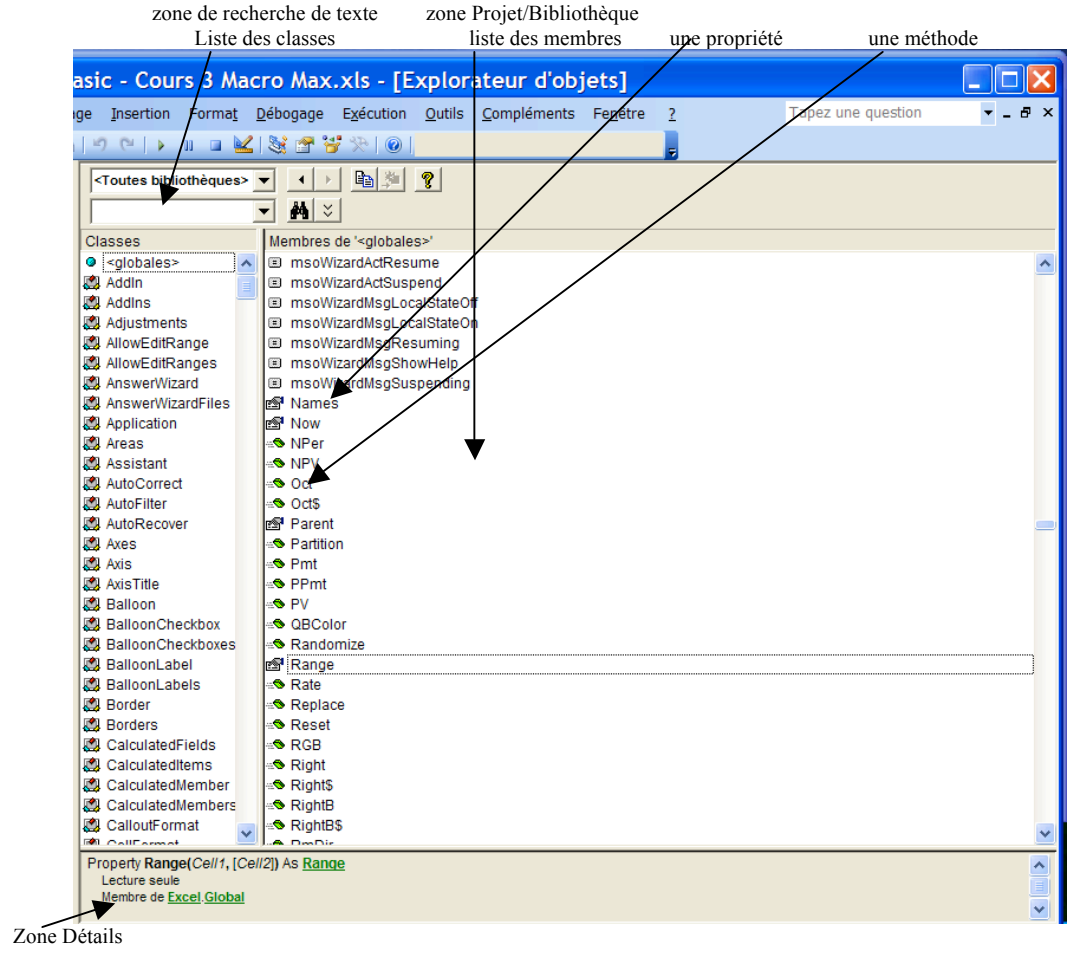
Sélectionner **Affichage/barres d'outils/Visual Basic** pour faire afficher la barre d'outils Visual Basic :



Etudions les différents boutons (démonstration en cours). Le bouton  (Visual Basic Editor) étant actionné, une fenêtre de l'éditeur s'ouvre (voir Figure 7, chapitre 3) (Faire Alt+F11 pour y accéder directement). Désormais, nous supposons être dans cette fenêtre. Allons dans l'explorateur d'objets (voir **Affichage/Explorateur d'objets** ou F2 ou bouton dans la barre Visual Basic).

L'explorateur d'Objets

Voici l'outil qui vous évitera d'apprendre par cœur une documentation complète : les **bibliothèques**, les **classes**, les **membres** et leur chemin (méthodes, propriétés, constantes, évènements associés à un objet).



Note : ce qui est créé par l'utilisateur apparaît en gras.

Sélectionnons le membre **Range** de <globales>:

D'après la zone Détails, c'est une propriété (Property), un membre, de la classe Excel.global retournant un objet Range.

Cliquons sur Range (la classe). Dans l'espace de description :

Class Range

Membre de Excel

Cliquons sur Excel :

Library Excel

C:\Program Files\Microsoft Office\OFFICE11\EXCEL.EXE

Microsoft Excel 11.0 Object Library

Cliquons sur Global :

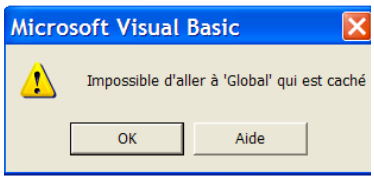


Figure 68

En cliquant sur **Aide** nous obtenons l'information suivante : *Pour les afficher, sélectionnez l'option **Afficher membres cachés** dans le menu contextuel de l'Explorateur d'objets*.

Ce menu contextuel est le suivant :

Figure 69

Désormais nous pouvons obtenir l'information :

Class Global

Membre de Excel

Si nous spécifions une bibliothèque particulière, comme Excel, de nombreuses classes et méthodes disparaissent.
Demandons, via le menu contextuel, de l'aide sur la propriété Range sélectionnée (ou bien avec F1):

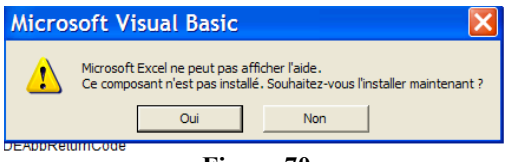


Figure 70

Installons ce composant. La fenêtre d'aide est la suivante (nous l'explorerons en cours) :

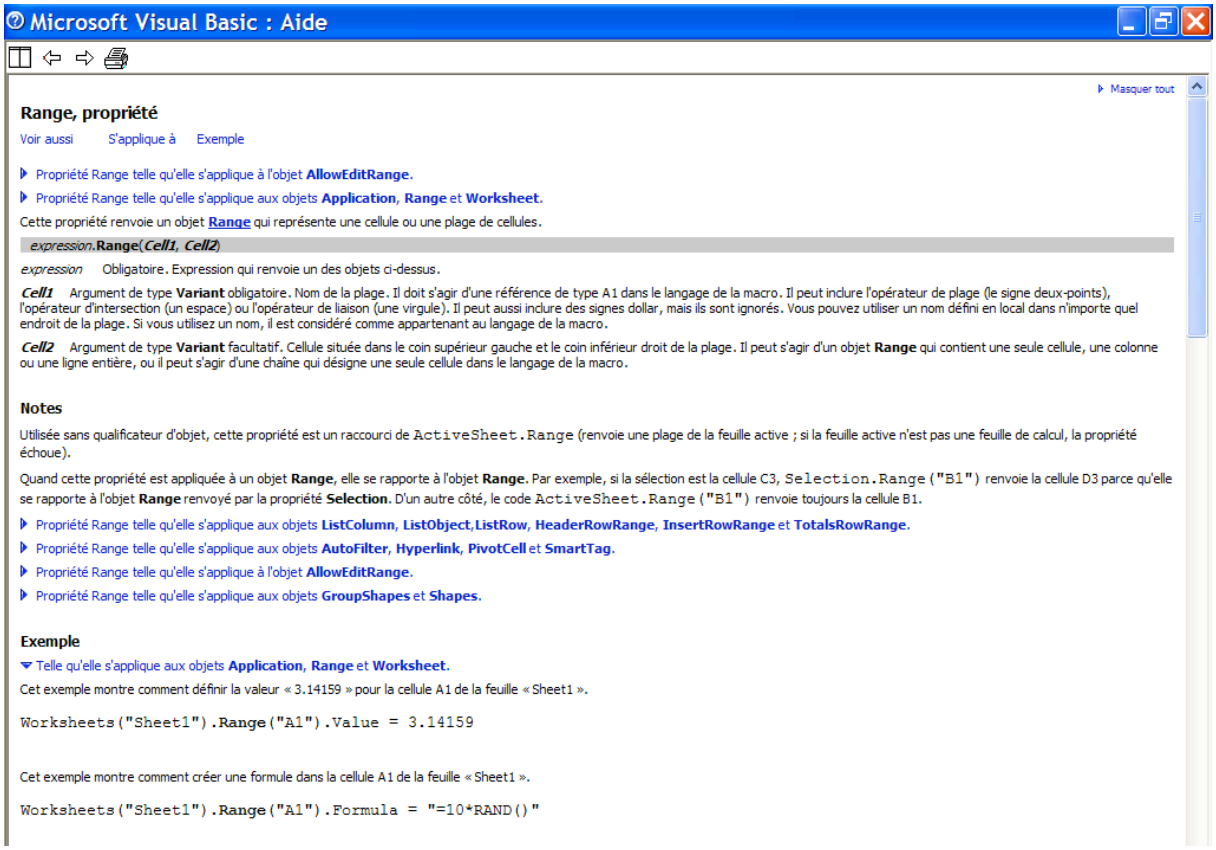


Figure 71

Note : l'aide est apportée en fonction de ce qui est sélectionné (membre, classe, rien). Si rien n'est sélectionné, l'aide indique le chemin d'accès à la bibliothèque ou au projet affiché.

Mettons en application le premier exemple proposé. Avec Insertion/Module, nous ouvrons un nouveau module (Module3) dans lequel, nous tapons notre texte de macro. Nous en savons désormais assez pour cela. Voici la fenêtre Code concernée :

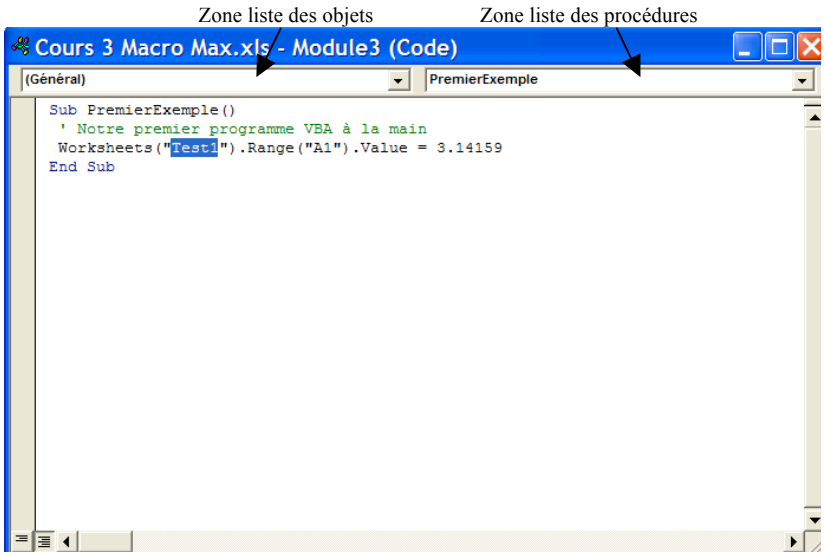


Figure 72

Nul besoin de valider, de compiler. Ouvrons un nouveau classeur Excel et renommons **Test1** la première feuille. Quelque que soit la feuille de ce classeur dans laquelle est exécutée cette macro, la cellule A1 de la feuille Test1 prend la valeur 3.141359. Si, en revanche, nous exécutons cette macro dans un classeur dont aucune feuille se nomme Test1, une erreur est déclanchée (c'est normal). Nous pouvons exécuter à partir de l'éditeur avec **Exécution\Ex Exécuter Sub**. (Ces étapes seront exécutées lors du cours).
Notons la manière donc l'éditeur VA complète et colore automatiquement lors de la saisie du code. Nous pouvons encore nous économiser avec **Insertion/Procédure** :

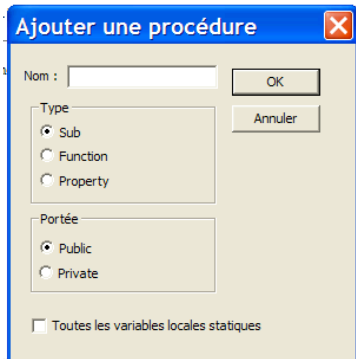


Figure 73

Sélectionnons la classe Corners d'Excel, puis Select dans la liste de ses membres. Demandons de l'aide :

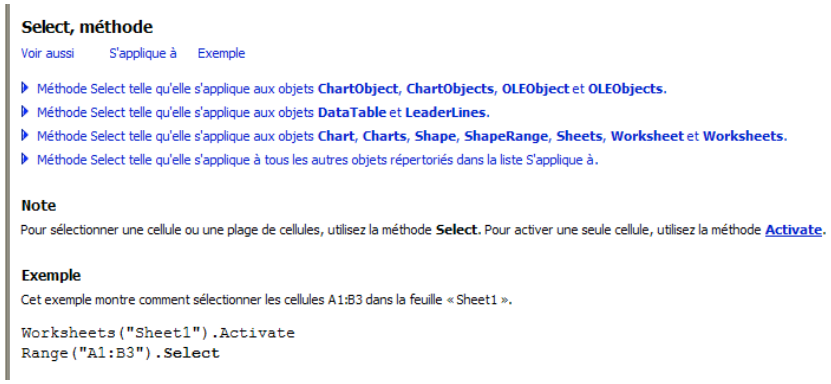


Figure 74

Pour pouvoir utiliser cette méthode, il faut l'appliquer à une expression retournant un des objets spécifiés (ChartObject, ChartObjects, ….).

La fenêtre UserForme

Elle permet d'ajouter des boites de dialogues dans les projets.

La fenêtre Code

Nous y avons déjà eu accès à plusieurs reprises. Une fenêtre code est attachée à l'un des modules (standard, module de classe, feuille, document rattaché au projet). Pour les faire afficher, il existe plusieurs moyens. Par exemple, sélectionner le module, cliquer avec le bouton droit et choisir **Code** dans le menu. Pour accéder directement à une procédure de la fenêtre, utiliser la **Zone de liste Procédures**.

EXERCICE 1

Définir une procédure pour créer une liste des macros complémentaires disponibles avec les propriétés spécifiées. Indication : regarder l'aide de la classe AddIn. Vous n'avez rien à connaître et vous obtiendrez un tableau ressemblant au suivant :

	A	Barre de formule	C	D
1	**Name**	**Full Name**	**Title**	**Installed**
2	lookup.xla	lookup.xla	Assistant Recherche	FAUX
3	sumif.xla	sumif.xla	Assistant Somme conditionnelle	FAUX
4	solver.xla	C:\Program Files\Microsoft Office\OFFICE11\Bibliothèque\SOLVER\solver.xla	Complément Solveur	VRAI
5	eurotool.xla	C:\Program Files\Microsoft Office\OFFICE11\Bibliothèque\eurotool.xla	Outils pour l'Euro	VRAI
6	analys32.xll	analys32.xll	Utilitaire d'analyse	FAUX
7	atpvbaen.xla	atpvbaen.xla	Utilitaire d'analyse - VBA	FAUX
8	html.xla	C:\Program Files\Microsoft Office\OFFICE11\Bibliothèque\html.xla	VBA pour l'Assistant Internet	FAUX
9				

EXERCICE 2 : Utilisation de la doc en ligne (suite)

Profitez de cette aide pour apprendre comment installer une macro complémentaire en VB et la structure de la boucle for. Chercher des exemples (avec la classe Characters, par exemple) où apparaît la structure de contrôle IF-THEN-ELSE.

EXERCICE 3 : manipulations de cellules

En utilisant la documentation en ligne, informez-vous sur Range, Cells, Row (et Rows), Column, ActiveCell, Selection, Offset, Select, Goto, Activate, Resize et Variant. Voir aussi les propriétés CurrentRegion, End de la classe Range et la propriété UsedRange.

Dans les exercices suivant, vous donnerez le code VBA puis le résultat sur une feuille de calcul Excel (si une *). Vous pouvez aussi utiliser la construction automatique des macros pour découvrir le code en VBA. Vous préciserez si vous travaillez sur des références relatives ou bien absolues.

1. Ecrire trois instructions permettant de sélectionner les cellules B1 : B4 ; C2 ; C6 puis d'activer la cellule B3 et enfin de lui affecter votre nom de famille (*).
2. Ecrire deux instructions permettant d'activer la feuille VotrePrénom du classeur VotreNom puis de sélectionner la cellule B3 de cette feuille.
3. Ecrire une instruction qui affecte votre prénom à la cellule B2 de la feuille Feuil1 du Classeur VotreNom (*).
4. Sélectionner la cellule située 2 lignes au dessus et 3 colonnes avant la cellule active.

Chapitre 5 Le Langage Visual Basic

Note : Dans un cours précédent, les principes généraux et une partie de la terminologie des langages objets ont été exposés au tableau. Désormais, les « écrans » seront encore moins détaillés . Les démonstrations du cours sont donc indispensables à sa compréhension.

Les feuilles

Un projet VBA est constitué de **feuilles**. A chaque **feuille** du projet est affecté un fichier dans le dossier **Feuilles** dans la fenêtre **Projet - VBAProject**. Ouvrir une boite de dialogue signifie ouvrir une feuille. Nous reviendrons sur les feuilles dans la partie 6 de ce cours.

Les modules

Ils forment un programme complet. Le code de l'interface est stocké dans un fichier **UserForm** accessible dans le dossier Feuilles.

Module standard : dossier **Module** ; code standard (voir Partie 4)
Module de classe : dossier **Module de classe** ; code décrivant les objets développés

Créer un module dans un projet actif (activé en cliquant dessus) : avec **Insertion** dans le menu contextuel du projet ou dans la barre d'outils standard de l'éditeur.

Supprimer : avec le menu contextuel du module

Modifier son nom : sélectionner le module, actionner sa **Fenêtre Propriétés** (dans **Affichage** ou avec la touche clavier **F4**) et modifier son nom dans cette fenêtre.

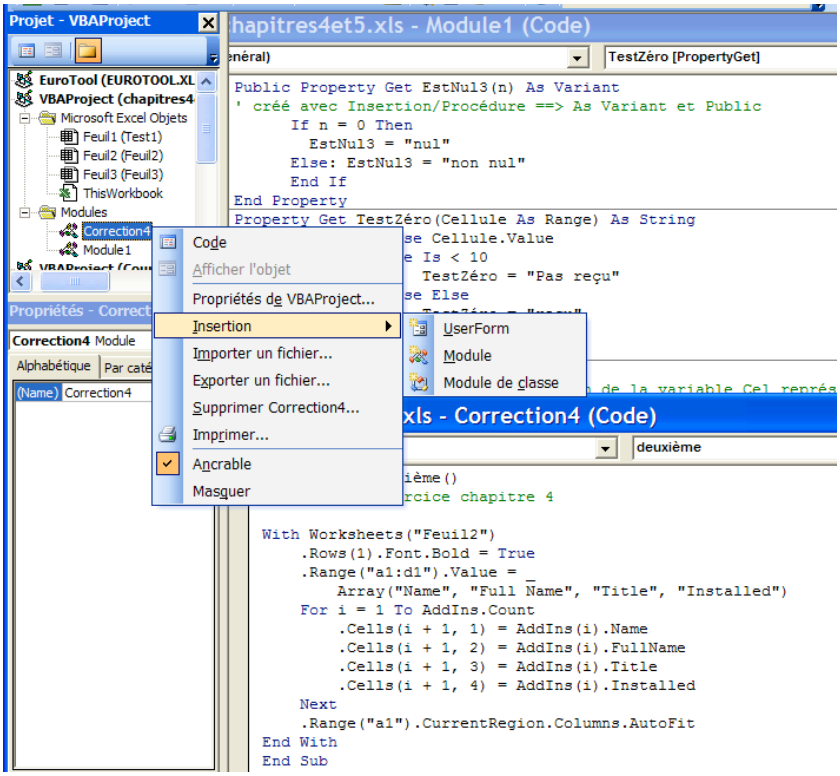


Figure 75

Chaque module est constitué de **procédures**.

Les Procédures

Un **programme** est l'ensemble des procédures d'un projet.

Procédure standard : exécutée par appel de la procédure par son nom.
Procédure évènementielle : déclanchée par un évènement (ex. clic de souris).

Dans un module **UserForm**, les procédures sont prédéterminées : une par évènement pouvant affecter un contrôle (voir Chapitre 6).

Le choix des procédures des modules standards doit être pensé avec **modularité** (i.e. ne pas tout mettre dans la même procédure).

Procédure Sub : ne renvoie pas de valeur

Définition :

```
[Private | Public] [Static] Sub NomProcédure([ParamètresFormels])
                    Corps
End Sub
```

Appel :

```
Call NomProcédure
```

Les éléments de ParamètresFormels sont séparés par une virgule (c'est une *série*).

Note : Pour associer un raccourci clavier à la macro, la sélectionner dans Excel : **Outils/Macros**, puis cliquer sur **Options**. Il est possible également d'associer une macro à un bouton.

Par défaut, une procédure est ***publique*** ; déclarée ***privée*** sa *portée* est restreinte au module.

Note : Une procédure évènementielle est, par définition, **privée** car ne peut-être appelée par aucune autre procédure (voir Chapitre 6).

Les valeurs des variables d'une procédure ***statique*** sont conservées durant l'exécution du programme (*mémo-variables*) même si elles ont une *visibilité* restreinte à la procédure.

Le code d'une procédure est constitué d'*instructions* évaluées séquentiellement.
Une ***instruction*** est formée de constantes, variables et mots clés (symboles reconnus par le langage). Il existe trois sortes d'instructions : de déclaration, d'affectation, exécutables. Nous les étudierons plus tard.

Procédure Function : Elle peut renvoyer une valeur.

Définition :

```
[Private | Public] [Static] Function NomFonction
([ParamètresFormels]) [As Type]
                    Corps
End Function
```

Pour retourner une valeur valeur (éventuellement résultat d'une *expression*), le corps de la fonction devra comporter l'instruction suivante :

```
NomFonction=valeur
```

Si As Type est donné, la valeur sera du type Type, sinon de type Variant. Pour économiser de la mémoire, il est préférable de préciser le type.
Si plusieurs instructions évaluées lors de l'exécution de la fonction sont de cette forme, c'est la dernière valeur affectée à NomFonction avant la fin de l'exécution de la fonction qui est retournée. Exécutons le programme suivant (à l'écran en cours) :

Exemple 1 :

```
Public Sub Test1()
' appel d'une fonction par une procédure

Dim n As Integer
  n = 10
  MsgBox "la valeur de " & n & " est " & Identité(n), vb0KOnly & vbInformation, " Test1 "
End Sub

Function Identité(n) As Integer
    Identité = n
End Function
```

Pour cette exécution, nous pouvons utiliser la barre d'outils débogage de l'éditeur.
Nous remarquons que :

- La fonction Identité n'est pas dans la liste des macros

- Elle est dans la liste des fonctions ; nous y accédons rapidement en choisissons la catégorie **personnalisées** dans la fenêtre **Insérer une fonction** d'Excel. Nous pouvons l'exécuter.

Nous avons pu aussi constater que nous n'avions pas à savoir si nous devions mettre As Int ou As Integer. L'intuition des premières lettres a suffit avec l'aide de l'éditeur, comme dans l'exemple ci-dessous :

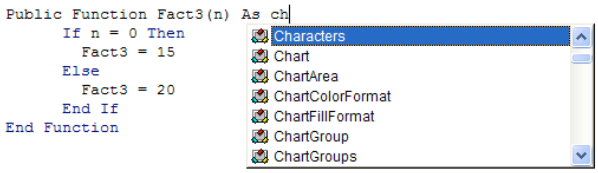


Figure 76

Exemple 2 :

```
Sub Test2()
  Dim n
  n = 10
  Worksheets("Test1").Range("A2").Value = n & " est " & EstNul1(n)
 End Sub

Function EstNul1(n) As Characters
  ' problème syntaxique
    If n=0 Then EstNul1 = "nul" Else EstNul1= "non nul" End If
End Sub
```

La fonction EstNul1 présente un problème syntaxique. Nous regardons en cours comment se passe l'exécution de la macro Test2. Voici la correction qui nous montre l'importance de la syntaxe :

```
Sub Test3()
  Dim n As Integer
  n = 10
  Worksheets("Test1").Range("A3").Value = n & " est " & EstNul2(n)
  MsgBox n & " est " & EstNul2(n), vb0KOnly & vbInformation, " Test3 "
End Sub

Public Function EstNul2(n) As String
    If n = 0 Then
     EstNul2 = "nul"
    Else
     EstNul2 = "non nul"
    End If
End Function
```

Exemple 3 :
A travers ce dernier exemple, nous constatons que les fonctions de VB supportent la *récursivité* :

```
Sub Test4()
  Dim n As Integer
  n = 4
  MsgBox n & " ! = " & Fact2(n), vb0KOnly & vbInformation, " Test4 "
End Sub

Public Function Fact2(n) As Integer
' Le calcul de 10! dépasse les capacités mais lesquelles ?
    If n = 0 Then
     Fact2 = 1
    Else
      Fact2 = n * Fact2(n - 1)  ‘ Fact2 s’appelle récursivement ; ici la récursion est
emboîtée par *
    End If
End Function
```

Pour éviter de dépasser les capacités d'espace mémoire, il faut *dérécursiver* la fonction. La première étape consiste à transformer la récursivité en récursivité terminale. Dérécursiver dans le cas simple d'un *emboîtement* : créer une variable solution et utiliser la boucle Tant Que (i.e. while). (Tout ceci sera étudié en cours).

Pour utiliser les fonctions de feuille de calcul Excel, il faut les considérer comme des méthodes de l'objet WorksheetFunction. Pour en savoir plus, consulter **Utilisation des fonctions de feuille de calcul Microsoft Excel disponibles dans Visual Basic.**

Exemple 4. Exécutons la procédure Test5 comme une macro ; cette procédure appelle la procédure paramétrée ProcPara qui n'est pas exécutable comme une macro (voir en cours) et qui appelle la fonction Excel Fact.

```
Sub Test5()
  Call ProcPara(10)
End Sub

Sub ProcPara(n)
  MsgBox n & " ! = " & WorksheetFunction.Fact(n), _
    vb0KOnly & vbInformation, " Test5 "
End
```

Nous en profiterons pour exécuter avec

- un point d'arrêt d'exécution (cliquer dans la colonne du module devant la ligne concernée ; un point rouge y apparaît et la ligne est colorée en rouge)
- et avec la fenêtre Variables Locales (i.e. **Affichage/Fenêtre Variables Locales)**.

Procédure Property Get : renvoie la valeur d'une propriété

```
[Private | Public] [Static] Property Get NomProcédure
([ParamètresFormels]) [As Type]
                Corps
            End Property
```

Se comporte comme les fonctions pour la valeur retournée :

NomProcédure=expression

et l'appel :

NomProcedure([ParamètresRéels])

En cours, nous ferons appel à l'aide VBA : Property Get.

Exemple :

```
Property Get TestZéro(Cellule As Range) As String
    Select Case Cellule.Value
        Case Is < 10
            TestZéro =  "Pas reçu"
        Case Else
            TestZéro =  "reçu"
End Property

Sub MonCom()
  Dim Cel As Range      'déclaration de la variable Cel représentant un objet de la classe Range
  For Each Cel In Selection
        Cel.AddComment (TestZéro(Cel))  ' méthode Add.comment de Range appliquée à l'objet Cel
  Next Cel
End Sub
```

Voici le résultat :

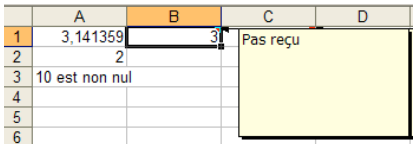


Figure 77

Procédure Property Let : définie la valeur d'une propriété

Définition :

[Private | Public] [Static] Property Let NomProcédure (VarStockage)

Corps

End Property

Appel :

NomProcédure = VarStockage

Procédure Property Set : établie une référence entre un objet et une propriété

Note : Sortir d'une procédure, d'une fonction : Exit Sub, Exit Property, Exit Function.

VARIABLES

A - Les Paramètres (les arguments)

Déclaration d'un paramètre formel ParamètreFormel :

[Optionnal] [ByVal] [ByRef] [ParamArray] ParamètreFormel [As Type] [= ValeurParDéfaut]

Optionnal : les arguments optionnels doivent être déclarés en dernier.
Omettre un paramètres réel est donc possible si le paramètre formel est **optionnel** mais le remplacer par rien ne signifie pas oublier les virgules (donc deux virgules consécutives).
Savoir si un paramètre formel ParamètreFormel a ou non une valeur (i.e. le paramètre réel a été ou non omis) :

IsMissing(ParamètreFormel)

Retourne True si aucune valeur et False sinon.

ByVal : le passage entre paramètres réels et paramètres formels est réalisé par **valeur**. Les paramètres formels prennent les valeurs des paramètres réels en respectant l'ordre d'apparition.

ByRef : le passage entre paramètres réels et paramètres formels est réalisé par **référence**. C'est l'adresse du paramètre réel qui est passée. Par défaut, le passage se fait par référence.

Exemples :

```
Dim v1, v2
Public Sub foo1()
  v1 = 2
  v2 = 2
  Call foo2(v1, v2)              'notons Call pour appeler une procédure, une fonction
  MsgBox " v1 = " & v1 & " et v2 = " & v2, vb0KOnly & vbInformation, " Test4 "
End Sub

Public Function foo2(v1, ByVal v2)          ' nous aurions pu mettre ByRef v1
 v1 = 3
 v2 = 3
End Function
```

Voici la boite de dialogue résultat :

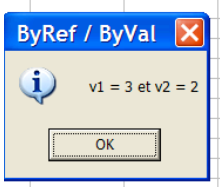


Figure 78

Note : Pour les connaisseurs, notez la similitude avec le langage Pascal (langage typé et compilé).

As Type : le type peut être Byte, Boolean, Integer, Long, Currency, Single, Double, Date, String, Object, Variant ou un type défini par l'utilisateur. Cela permet le contrôle de types.

= ValeurParDéfaut : utilisable uniquement avec un paramètre optionnel. Ce ne peut être qu'une constante ou une expression constante (3+2, par exemple).

Les arguments nommés
Lors de l'appel d'une procédure Sub, utiliser l'affection ParamètreFormel :=ParamètreRéel à la place de ParamètreRéel sert à éviter les erreurs en cas d'omission de paramètre réel (cas des paramètres optionnels) et à passer les paramètres formels dans un ordre quelconque. ParamètreRéel est dit un *argument nommé*. La plupart des fonction intégrées de VB intègrent des arguments nommés.

B- Déclarations des variables

Elles peuvent être *explicites* ou *implicites*. VB Editor permet de forcer les déclarations explicites avec l'instruction : Option Explicite dans la section **Déclarations** de la **fenêtre Code** (cliquer sur la première ligne de la fenêtre code et observer la **zone liste des procédures**) ou bien **Outils/Option/Editeur : Déclaration des variables obligatoire** pour systématiser.

Implicite : la variable sera de type Variant.

Explicite : évite les erreurs de frappe. Recommandé en cas de développement important.

Déclarer explicitement une variable : Dim NomVariable [As Type]. Encore une fois, comme selon le type l'espace mémoire réservé est plus ou moins important, la précision du type optimise l'occupation de l'espace mémoire.

Déclaration de plusieurs variables : Dim V1, V2,...,Vn [As Type],w1,...,wm ,[As Type],....

Les Types

La fonction **StrConv** : pour convertir un type de données de chaîne en un autre.

En demandant le **résumé des types de données** à l'Aide, nous obtenons le tableau suivant auquel sont rajoutées quelques précisions :

Type de données	Taille d'enregistrement	Plage
Byte (petit entier)	1 octet	0 à 255
Boolean	2 octets	**True** ou **False**
Integer	2 octets	-32 768 à 32 767
Long (entier long)	4 octets	-2 147 483 648 à 2 147 483 647
Single (à virgule flottante en simple précision)	4 octets	-3,402823E38 à -1,401298E-45 pour les valeurs négatives ; 1,401298E-45 à 3,402823E38 pour les valeurs positives

Double (à virgule flottante en double précision)	8 octets	-1,79769313486231E308 à -4,94065645841247E-324 pour les valeurs négatives ; 4,94065645841247E-324 à 1,79769313486232E308 pour les valeurs positives
Currency (entier à décalage)	8 octets	-922 337 203 685 477,5808 à 922 337 203 685 477,5807
Decimal	14 octets	+/- 79 228 162 514 264 337 593 543 950 335 sans séparateur décimal ; +/-7,9228162514264337593543950335 avec 28 chiffres à droite du séparateur décimal ; le plus petit nombre différent de zéro est +/- 0.0000000000000000000000000001.
Date	8 octets	1er janvier 100 au 31 décembre 9999
Object	4 octets	Toute référence à des données de type **Object**
String (longueur variable)	10 octets + longueur de la chaîne	0 à environ 2 milliards
String (longueur fixe)	Longueur de la chaîne	1 à environ 65 400
Variant (nombres)	16 octets	Toute valeur numérique, avec la même plage de valeurs qu'une donnée de type **Double**
Variant (caractères)	22 octets + longueur de la chaîne	Même plage de valeurs qu'une donnée de type **String** de longueur variable
Type défini par l'utilisateur (avec **Type**)	En fonction des éléments	La plage de valeurs de chaque élément correspond à celle de son type de données.

Tableaux (Array)

Les tableaux commencent à l'indice 0. Pour commencer à 1, écrire Option Base 1 dans la section **Déclaration** de la fenêtre Code ou bien délimiter avec … To … . En cours, nous ferons appel à l'aide avec **Utilisation des tableaux**.

Type nom simple *homogène* ou *hétérogène* (avec As Variant) :

```
Dim NomVariable(NombreEléments) As Type
Dim NomVariable(Début To Fin) As Type

Dim NomVariable(D1 To F1, D2 To F2,...,Dn To Fn) As Type
```

Les tableaux à deux dimensions sont utilisables pour stocker une feuille de calcul :

Vérification du type : IsArray(NomVariable)

Quel que soit le type de données, les tableaux nécessitent 20 octets de mémoire, auxquels viennent s'ajouter quatre octets pour chaque dimension et le nombre d'octets occupés par les données. L'espace occupé en mémoire par les données peut être calculé en multipliant le nombre d'éléments par la taille de chacun d'eux. Par exemple, les données stockées dans un tableau unidimensionnel constitué de quatre éléments de type **Integer** de deux octets chacun occupent huit octets. Ajoutés aux 24 octets d'espace mémoire de base, ces huit octets de données portent la mémoire totale nécessaire pour le tableau à 32 octets.

Une variable de type Variant contenant un tableau nécessite 12 octets de plus qu'un tableau seul.

Voir dans la docenligne : Ubound, LBound, Erase (libération de l'espace mémoire), Join.

Exemples :

```
Sub Dim1()
   Dim curExpense(364) As Currency
   Dim intI As Integer
   For intI = 0 To 364
      curExpense(intI) = 20
      Cells(intI + 1, 3).Value = curExpense(intI)
   Next
End Sub

Sub TableauMulti()
   Dim intI, IntJ As Integer
   Dim MultiDim(1 To 5, 1 To 10) As Single
   ' Remplit le tableau de valeurs.
   For intI = 1 To 5
      For IntJ = 1 To 10
         MultiDim(intI, IntJ) = intI * IntJ
         Cells(intI + 4, IntJ + 5).Value = MultiDim(intI, IntJ)
      Next IntJ
   Next intI
End Sub


Sub TestTableauxSub()
  Dim i As Integer
  Dim M2() As Variant
  Dim MonTableau As Variant ' Essayer As Integer
  MonTableau = Array(2, 5, 6)
  ' Pour cette boucle, éviter que la cellule active appartienne à une matrice
  For i = 0 To UBound(MonTableau)
     ActiveCell.Value = MonTableau(i)
     ActiveCell.Offset(0, 1).Activate
  Next i
```

```
  ActiveCell.Offset(0, -UBound(MonTableau) - 1).Activate
  ' Utilisation de FormulaArray pour recopier ; fonctionne aussi avec .Value
  Range("A2:C2").FormulaArray = Range(ActiveCell, _
         ActiveCell.Offset(0, UBound(MonTableau))).Value
  Range("A3:C3").FormulaArray = MonTableau
  Range("A4:C4").FormulaArray = Array(11, 12, 13)
  ' MsgBox MonTableau(1) & " " & MonTableau(2)
  M2 = MonTableau
  MsgBox IsArray(Range("A4:C4").FormulaArray), _
        vbInformation, "Est-ce un tableau ?"
 ' FormulaArray pour une matrice
  Range("D1:E1").FormulaArray = "=Sum(R1C1:R3C3)"
' Passer un tableau en paramètre
Dim MonTab(2) As Single
MonTab(0) = 5.1
MonTab(1) = 7.2
MsgBox MoyenneTableau(MonTab) , vbInformation,"moyenne"
End Sub

' MoyenneTableau calcule la moyenne des éléments du tableau MonTab
Public Function MoyenneTableau(MonTab() As Single) As Single
   Dim somme As Single
   somme = 0
   For i = LBound(MonTab()) To UBound(MonTab())
      somme = somme + MonTab(i)
   Next i
   MoyenneTableau = somme / CSng(UBound(MonTab()) - LBound(MonTab()) + 1)
End Function
```

Ci-dessous, nous retournons un tableau qui sera considéré comme une matrice sous Excel. Pour exécuter la fonction RetournerUneMatrice, Sélectionner la zone matricielle (i.e. deux cellules, ici) avant l'appel de la fonction puis l'exécuter avec CTRL+MAJ+ENTREE (voir Cours 2) ;
les valeurs 24 et 25 seront celles des cellules sélectionnées

```
Function RetournerUneMatrice()
   RetournerUneMatrice = Array(24, 25)
End Function
```

Avec un descriptif, ranger la fonction RetournerUneMatrice du module Tableaux2 dans la catégorie Fonctions Cours 5 (sinon elle sera par défaut dans la catégorie Personnalisée).

```
Sub AddFunctionavb2()
   Application.MacroOptions _
      Macro:="Tableaux2.RetournerUneMatrice", _
      Description:="Teste retourner un tableau en valeur", _
      Category:="Fonctions Cours 5"
End Sub
```

Nous exécuterons ces fonctions et procédures durant le cours.

Tableaux dynamiques :
Définition

```
Dim NomVariable()
```

Redimensionner :

```
ReDim NomVariable(Début To Fin)
```

Entraîne la perte de toutes les valeurs stockées. Sinon :

```
Preserve ReDim NomVariable(Début To Fin)
```

Avec comme conditions : la dimension du tableau ne peut être modifiée et la taille ne peut être qu'agrandie et seule la dernière dimension de la variable n'est redimensionnable.

Tableaux paramétrés :

Exemple :

```
Sub testTableauPara()
 TableauPara 1, 2, 5, 24
End Sub

 ' si plusieurs arguments, dernier argument de la procédure
Function TableauPara(ParamArray MonTableau() As Variant)
  Dim i As Integer
  For i = 0 To UBound(MonTableau())
    Cells(i + 1, 1).Value = MonTableau(i)
   Next
End Function
```

Types Utilisateur (personnalisé, comme le struct en C ou le Record en Pascal):

Définition :

```
Type MonType
 Ident1 As Type1
 Ident2 As Type2
      ...
 Identn As Typen
    End Type
```

Utilisation :

```
Dim Var1,Var2 As MonType
  Var1.Ident1 = valeur
   With Var2
      .Ident1 = valeur1
      .Ident2 = valeur2
    End With
```

Vérification des types : IsArray, IsDate, IsNumeric, IsObject, IsMissing, IsEmpty, IsNull, IsError (voir aussi VarType).
Déterminer le type d'une variable : VarType(NomVariable) (voir aussi TypeName).

Exemple :
```
Function EntrerDate()
  Do
    EntrerDate= InputBox("Entrer une date", "Vérification")
  Loop Until IsDate(EntrerDate) = True
End Function
```

Conversion de type : CBool(NomVariable), CByte, CCur, CDate,CDbl, Cdec,CInt,CSng,CVar,CStr

Les types doivent être compatibles. Il n'y a pas de troncation mais un déclanchement d'erreur. Si on veut ramener la partie entière d'un décimal Variable : Int(Variable).
Avec l'aide, nous étudierons les **conversions de type** : tapons **type** dans l'aide et sélectionnons **fonctions de conversion de type de données.**

Exemple de conversion :

```
Public Sub TestConv()
MsgBox n & CInt("245") + 3 & , vb0KOnly & vbInformation, " TestConv "
End Sub
```

Les constantes : Donner un nom Nom à une valeur Val durant toute l'exécution du programme :

```
Const Nom = Val
```

Les Objets

Une fois une variable objet (l'**instance** d'une classe) défini, il est possible de définir ou d'interroger ses propriétés, de lui appliquer les méthodes …

```
Dim NomVariable As NomClasse
```

Toute classe héritant de la classe Objet, il est toujours possible de prendre Objet pour NomClasse. L'objet est déclaré mais pas construit.

La construction de l'objet (d'une instance de la classe) se réalise en lui affectant avec Set un objet de la même classe déjà existant :

```
Set NomVariable = expression
```

Exemple :

```
Sub VariableObject()
  Dim Police As Font              'Définition
  Set Police = Workbooks("classeur1.xls").Sheets("Feuil1").Range("a1").Font     'Création
```

```
  Police.Bold = True
End Sub
```

On peut utiliser aussi **GetObject** dont les arguments sont **nommés** :

```
Set NomVariable = GetObject([CheminAccèsFichier] , classe)
```

Ou bien :

```
Set NomVariable = GetObject(CheminAccèsFichier)
```

permettant d'accéder à des tableaux Excel sans que ceux-ci soient ouverts.

Ou bien **CreateObject** :

```
Set NomVariable = CreateObject(classe
[,AdresseMachineDistante])
```

Exemple 1:

```
Dim MonClasseur1,  MonClasseur2 As Workbook    'ce seront des instances de la classe
Workbook, i.e. des classeurs
Set MonClasseur1 = GetObject("C:\Documents and Settings\cvb\Bureau\classeur1.xls")
Set MonClasseur2 = CreateObject("Excel.Workbook ")  ' c'est-à-dire application.classe
```

Libérer l'espace mémoire occupé par une variable objet :

```
Set NomVariable = Nothing
```

Exemple 2: si dans **Outils/Références** de VB Editor, l'application Microsoft Word 11.0 Object Library est sélectionnée (voir Figure 5), nous avons accès à cette application non hôte. (Nous irons dans VBA sous l'application Word.)

Voici la fenêtre Références :

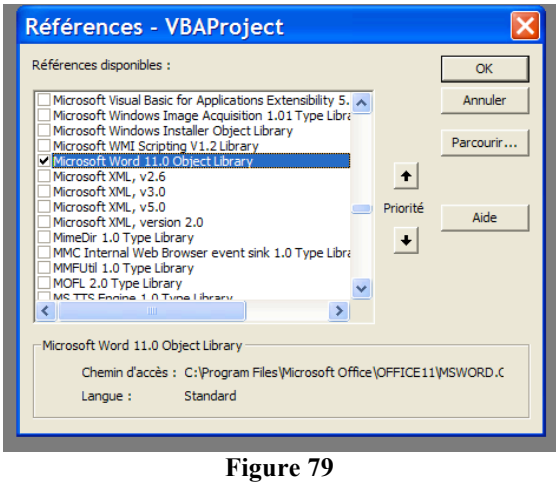


Figure 79

```
Const Chemin As String = "C:\Documents and Settings\cvb\Bureau\MonText.doc"
```

```
Public Sub ExcelEtWord()
'Classe Word.Document doit être accessible
Dim MonDoc As Word.Document

'Gestion d'erreur avec Resume Next
' si une erreur se produit lors d'une instruction, l'instruction suivante est évaluée
On Error Resume Next
Set MonDoc = GetObject(, "Word.Application")
'Gestion de l'erreur au cas où l'application Word ne soit pas accessible
If Err.Number <> 0 Then Err.Clear

'Création de l'instance MonDoc de la classe Word.Document
Set MonDoc = GetObject(Chemin)

' effaçons les anciennes données dans MonText.doc et dans la feuille Excel Active
If MonDoc.Tables.Count > 0 Then
  MonDoc.Tables(1).Delete
End If
Excel.Range("A:A").Delete

' création d'une table dans Mondoc
Dim Position As Word.Range
Set Position = MonDoc.Range(0, 0)
' creation d'un tableau dans le document MonText.doc à partir de l'application hôte Excel
MonDoc.Tables.Add Range:=Position, NumRows:=3, NumColumns:=4
' Ecrivons dans cette table
' en utilisant les outils de l'application Word
 Dim MaTable As Word.Table
 Set MaTable = MonDoc.Tables(1)
' Ecriture : Si l'application Word est fermée une erreur est déclanchée et rattrapée par Resume Next
 With MaTable
    ' écrit "c'est ici" devant le texte existant
    .Cell(1, 2).Range.InsertBefore "c'est ici"
    ' efface le contenu de la cellule et écrit "et là"
    .Cell(2, 3).Range.Text = "et là"
    ‘ copie le texte de MaTable contenu d'une cellule dans une autre
    .Cell(1, 3).Range.Text = .Cell(2, 3).Range.Text
   ‘ Lecture : Si Word est fermé il n'y pas d'erreur
   ' copions dans des cellules sous Excel, l'application hôte
    Excel.Range("A1").Value = .Cell(1, 2).Range.Text
    Range("A2").Value = .Cell(2, 3).Range.Text
    Dim mot As Variant
    For Each mot In .Cell(1, 2).Range.Words
      Excel.Range("A3").Value = Range("a3").Value & "" & mot.Text
      .Cell(3, 3).Range.Text = mot.Text
    Next mot
    ' pour se débarrasser du dernier mot parasite
    Dim i As Integer
    For i = 1 To .Cell(2, 3).Range.Words.Count - 1
     Excel.Range("a4").Value = Range("a4").Value & "" & .Cell(2, 3).Range.Words(i)
    Next i
End With
' sauvegarde du fichier MonText.doc
MonDoc.Save
End Sub
```

Nous exécuterons ce programme en faisant afficher Err.Number selon les cas de figure.

Nous irons sous VB Editor de l'application Word. Nous regarderons la classe Table, sa méthode Cell, la classe Cell, ses méthodes Formula, Id, Merge, Range … Nous regarderons ErrObj, OnError.
Voir dans la docenligne : Ecriture d'instructions de déclaration, Déclarations de variables

Portée et durée de vie des variables

Variable de niveau procédure : locale à une procédure
static : variable garde sa valeur d'un appel à l'autre (mémo-variable).

Variable au niveau module : déclarées dans la section Déclarations du module
public : visible par toutes les procédures (et fonctions) du projet
private : visible que par les procédures du module (par défaut)

Les instructions (structures de contrôle)

Une instruction est formée de constantes, variables et mots clés (symbole reconnu par le langage).
Il existe trois sortes d'instructions : de déclaration, d'affectation, exécutables.

- Do ... Loop et While ... Wend (boucler avec un test d'arrêt ; à utiliser pour dérécursiver)
- For ... Next (parcourir un ensemble par indiçage borné)
- For Each ... Next (parcourir un ensemble par appartenance)
- If ... Then ... Else
- Select Case ... Case ... Case Else ... End Select
- GoTo Etiquette

Attention aux boucles infinies : CTRL+Espace pour en sortir.

En cours, nous regardons les exemples de l'aide.

Opérateurs

Dans Excel, consulter l'aide : **a propos des opérateurs de calcul**
Dans Visual Basic Editor : l'aide Résumé des opérateurs **fournit le tableau suivant. (Si vous ne le trouvez pas, allez dans** and operator **puis cliquez sur** Voir aussi**.) (**Voir aussi : Is et Like.)

Opérateurs	Description
Opérateurs arithmétiques	Opérateurs permettant d'effectuer des calculs mathématiques.
Opérateurs de comparaison	Opérateurs permettant d'effectuer des comparaisons.
Opérateurs de concaténation	Opérateurs permettant de combiner des chaînes.
Opérateurs logiques	Opérateurs permettant d'effectuer des opérations logiques.

Boîtes de dialogue

En cours, nous regarderons l'aide.

Fonction InputBox :

Variable = InputBox(prompt, title[,default, left,top,helpFile, helpContextId,type])

La variable récupère le résultat tapé par l'utilisateur. Si l'utilisateur annule ou ferme la boîte de dialogue, une chaîne vide est retournée.

Méthode InputBox de l'objet Application d'Excel peut aussi être utilisée :

Set Variable = Application.InputBox(prompt, title [,default, left,top,helpFile, helpContextId,type])

La méthode permet de sélectionner une plage de cellules avant de cliquer sur OK.

Fonction MsgBox :

Variable = InputBox(prompt, buttons, title)

où Variable est de type Integer.prompt.

Boîtes de dialogues Excel

Ce sont des objets Dialog (collection Dialogs) ; nous leur appliquons les méthodes Show ou Display.

Application.Dialogs(Boîte).Show
Application.Dialogs(Boîte).Display

Où Boîte est une constante Excel indiquant la boîte de dialogue.
Voir aussi : **GetOpenFilename** et **GetSaveAsFilename**.

Se Renseigner sur … (aide VBA)

Ecriture de données dans des fichiers (grandes données).
Utilisation efficace des types de données.

Exercices :

Les programmes doivent être largement commentés (individuellement bien entendu …) et fournis avec un jeu d'exécutions personnel. Ne pas trop en faire non plus.

1. Ecrire une fonction Fact3 non récursive retournant le même résultat que Fact2 et comparer les capacités des deux fonctions en terme de dépassement. Vous ferez appel à l'**aide** pour trouver la syntaxe de la **boucle** while. Vous essaierez de nouveau 10 ! et expliquerez votre résultat en regardant la documentation du type Integer. Comparez avec la fonction Fact d'Excel. Qu'en concluez-vous ?
2. Ecrire une fonction Fact4 avec Do.
3. Donner un très petit exemple personnel pour la procédure **Property Let**.
4. Définir deux petites sub différentes et personnelles publiques et de même nom dans deux modules distincts d'un même projet. Essayez avec une privée et de même nom. Essayez dans deux projets différents (avec les classeurs simultanément ouverts). Faites de même avec deux fonctions. Qu'en concluez-vous ?
5. Réalisez des conversions de type avec des variables et observez ce qui est convertit : la variable ou bien la valeur de la variable ? Qu'en concluez-vous ?

6. Ecrire une macro qui stocke dans un tableau la table des ventes d'appartements vue précédemment et la recopie trois colonnes plus loin (correction la semaine prochaine dans le classeur appart.xls).
7. Avec la boucle For, écrire une fonction qui rajoute 1 à tous les éléments d'un tableau bidimensionnel d'entiers.
8. Ecrire votre fonction DroiteReg2 (qui retourne le même résultat que la fonction prédéfinie DroiteReg).
9. Ecrire une fonction RecFich(NomFichier) qui recopie les 10 premiers mots d'un fichier Word NomFich dans les cellules A1 à A10 de la feuille active du classeur actif d'Excel. Exécuter cette fonction à partir d'une procédure sans paramètres.
10. Ecrire une fonction récursive qui fait le produit d'une matrice d'entiers nxm par un vecteur de dimention n.
11. Soient v et w deux vecteurs de n entiers de coordonnées respectives vi et wi. En utilisant la gestion des erreurs, écrire une fonction qui prend v et w en arguments et qui retourne le vecteur m tel que mi=vi/wi, si wi est non nul et mi=vi, sinon.

Chapitre 6 Interfaces Utilisateur

Elles se réalisent par les feuilles (à ne pas confondre avec les feuilles d'un classeur Excel). Ce sont des zones sur lesquelles se placent les contrôles ActiveX (boutons, zones de texte, …).

Ces contrôles constituent une interface graphique interactive. Pour associer du code à un évènement : *procédures évènementielles* :

```
Private Sub Contrôle_Evènement()
        Corps
End Sub
```

Contrôles : Label, TextBox, ComboBox, ListBox, CheckBox, OptionBox, SpinButton.

Evènements : AfterUpdate, BeforeDragOver, BeforeDropOrPaste, BeforeUpdate, Change,Click, dblClick, DropButtonClick, Enter, Exit, Initialize, KeyDown, KeyPress, KeyUp, Mouse Down, MouseMove, MouseUp, SpinDown, SpinUp.

Une feuille UserForm

Pour ouvrir une feuille **User Form**, choisir **Insertion/UserForm** dans les menu contextuel de la fenêtre **Explorateur de Projets**.

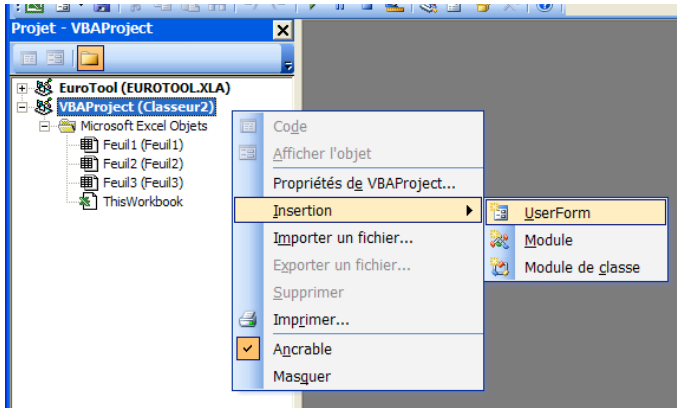


Figure 80

Est ainsi créé une feuille UserForm du Classeur (ici UserForm1 du Classeur2). C'est un objet de la classe UserForm (une instance) dont on peut modifier les propriétés via son menu contextuel.

En cours, nous irons dans l'explorateur d'objets où nous retrouverons notre objet UserForm1, nous commenterons la fenêtre Propriétés et la boîte à outils. En particulier, pour découvrir les contrôles, nous utiliserons l'aide avec la touche F1.

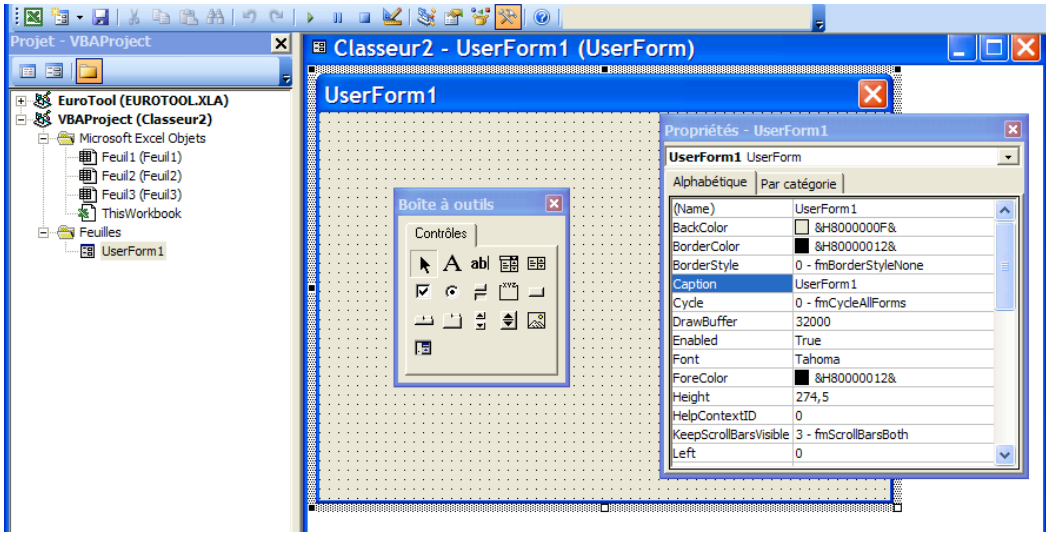


Figure 81

Le nom (i.e. Name) UserForme1 est celui de cette feuille (cet objet) dans le code des programmes.
A partir de l'explorateur d'objet, nous pouvons ainsi nous interroger sur la propriété KeepScrollBarsVisible (voir **Figure 2**).

Les Contrôles

Les propriétés de chaque objet contrôle sont également modifiables via son menu contextuel (en particulier, son nom). Chaque objet contrôle possède également ses méthodes et ses évènements. Dans l'exemple ci-dessous, nous mettrons à True la propriété Value de l'objet OptionButton1 et nous modifierons la propriété Caption des objets de la feuille afin qu'elles soient identiques (ce n'est pas obligé) aux noms respectifs de chaque objet (Name).

Arranger les contrôles

Nous passerons vite sur cette partie car la lecture d'une documentation est suffisante.

Outils/Options/Général/Aligner les contrôles sur la grille

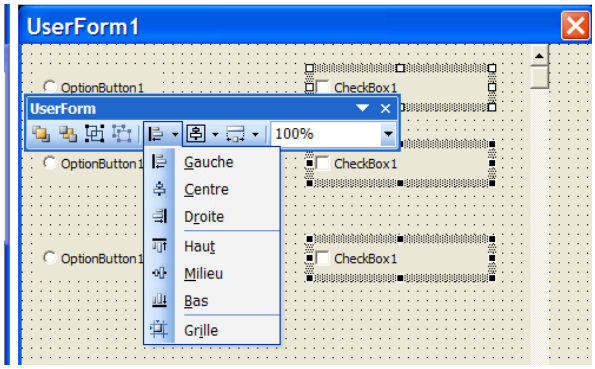


Figure 82

Contrôles supplémentaires

Voir aussi qu'il est possible de rajouter des pages via le menu contextuel de la boîte à outils.

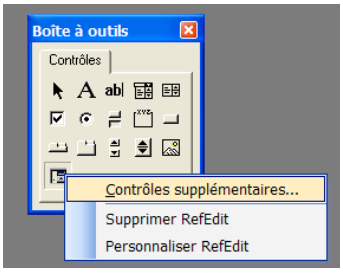


Figure 83

Afficher une Feuille

Dans la classe UserForm est définie la méthode Show qu'il suffit d'appliquer à une instance de cette classe (notre feuille UserForm1) pour que cet objet apparaisse : UserForm1.Show. Pour masquer une feuille, il y a la méthode Hide. Pour désigner la feuille (i.e. UserForm1) active, on peut utiliser la propriété Me. Si UserForm1 est cette feuille active, Me.Show affichera la feuille UserForm1. Dans les langages objets, ce genre d'astuce est souvent utile car le nom de l'objet n'est pas toujours connu (ici la feuille active) au moment ou le code est généré ; qui plus est, ici, ce code sera alors applicable pour chaque feuille active quelque soit son nom (penser, par exemple, au this de JAVA).

Ouvrons une fenêtre code (via le menu contextuel de l'objet UserForm1 ou celui du projetVBA) ; définissons la procédure suivante et exécutons-là ensuite :

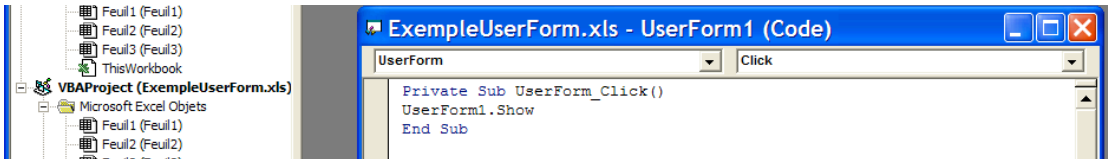


Figure 84

Notre feuille crée une boîte de dialogue dans la feuille (Sheet) Excel active.

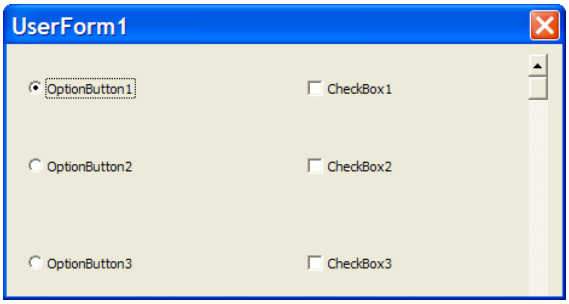


Figure 85

L'instruction UserForm1.Hide pourra être subtilement introduite dans la procédure évènementielle associée à un bouton de la boîte de dialogue à actionner afin de fermer la boîte.

Procédures évènementielles

Nous allons juste traiter un exemple expliquant le principe et échapper à une description fastidieuse et exhaustive. Supposons que la propriété Value de OptionButton1 soit à True (voir Figure 6).
Double cliquons sur OptionButton1 dans la fenêtre UserForm1 (voir Figure 3) ; nous accédons à la fenêtre code et la procédure associée au click nous est proposé ; les autres apparaissent choix sont facilement identifiables (voir Figure 7). Nous mettons l'instruction Range("A1").Value = 23 dans le corps de la procédure OptionButton1_Click() :

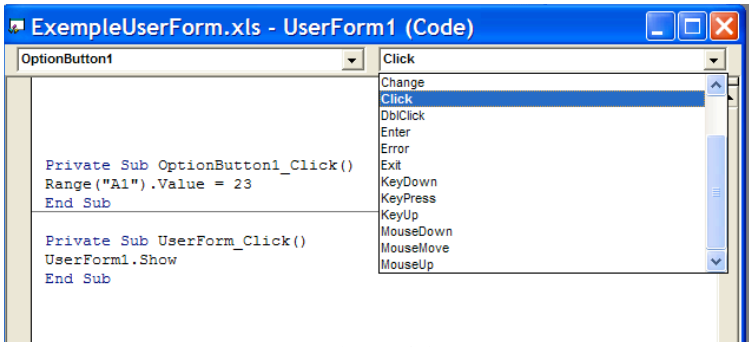


Figure 86

Lorsqu'on exécute, la cellule A1 de la fenêtre active a pour valeur 23.
Rajoutons deux procédures :

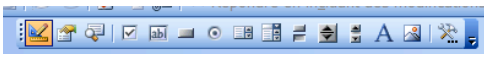
```
Private Sub OptionButton2_Click()
Range("A1").Value = 24
End Sub

Private Sub OptionButton3_Click()
Range("A1").Value = 25
End Sub
```

puis observons ce qui se passe sur la feuille de calcul Excel lorsque l'on clique dans les boutons de la boite de dialogue. Nous rajouterons un bouton que nous appellerons OK (avec sa propriété Caption) et nous écrirons sa méthode privée (procédure évènementielle) qui permet de fermer la boite de dialogue (avec UserForm1.Hide) si on le clique.

Dans la zone Liste des procédures de la fenêtre Code (voir Chapitre 4), nous trouvons les procédure évènementielles possibles. Cliquons sur l'un de ces choix.

Nous pouvons également rajouter des boutons sur la feuille de calcul Excel avec la Boîte à Outil Contrôles :



Cliquer sur le bouton d'activation de cette boîte (le premier), puis sur le contrôle choisi, puis sur l'endroit de la feuille où le bouton doit être placé. Pour créer une procédure évènementielle, cliquer sur la zone du bouton sur la feuille.

Nous profiterons de cette démonstration pour voir le code dans la barre des formules Excel.

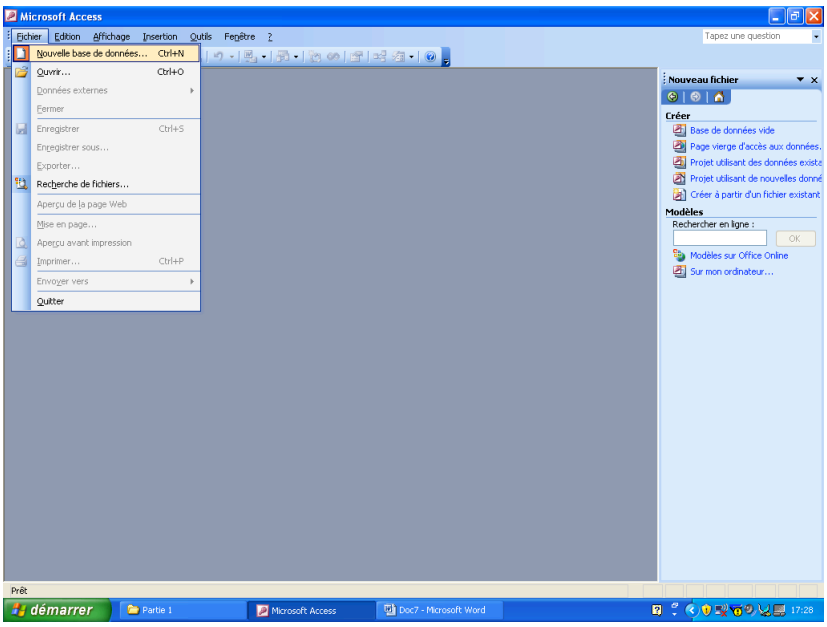
Chapitre 7 Bases de données : ACCESS

Ce chapitre applique à ACCESS les notions apprises dans la première partie du cours « Bases de données ». Comme pour Excel & VBA, cette partie du cours dédiée à ACCESS est interactive.

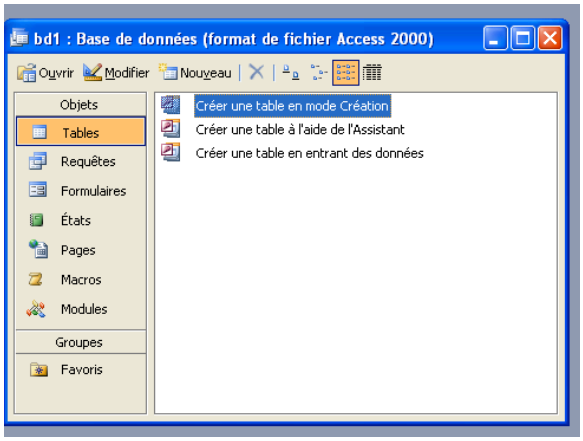
I. Création de tables

1- Appelons Access à partir de Windows
`démarrer/Programmes/Microsoft Office ACCESS`

2- Ouvrons une nouvelle base
`Fichier/Nouvelle base de données`

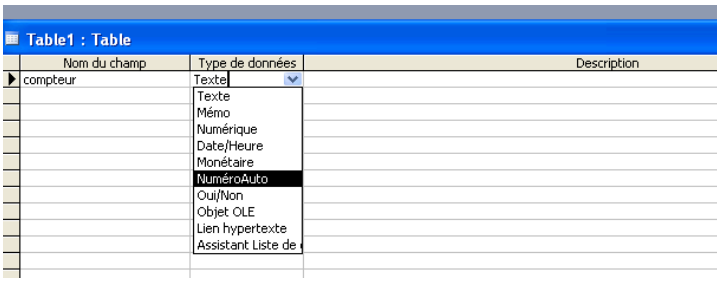


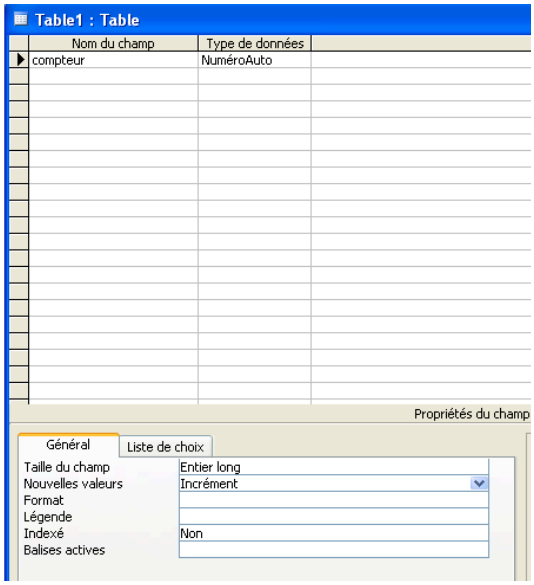
3- Rentrons en mode création



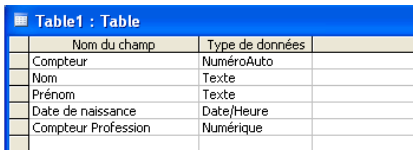
4- Créons une table comportant les champs suivants :

- `Compteur Nom`
  et choisir le type de compteur :
  `NuméroAuto -> Propriété : incrément`

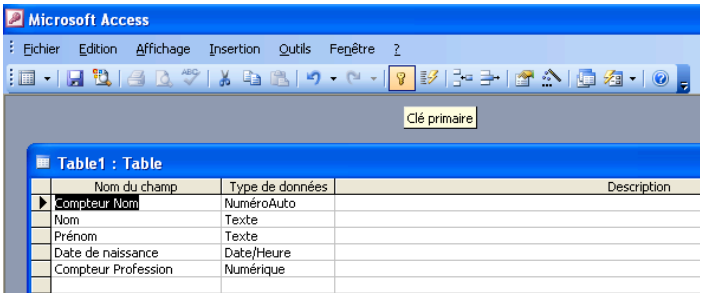


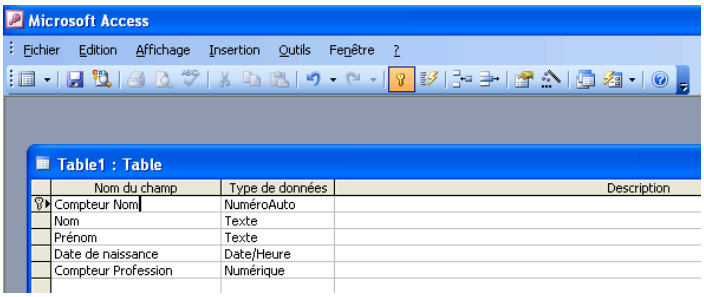


- `Nom` avec le type `Texte`
- `Prénom` avec le type `Texte`
- `Date de naissance` avec le type `Date`
- `Compteur Profession` avec le type `Numérique, entier long`.

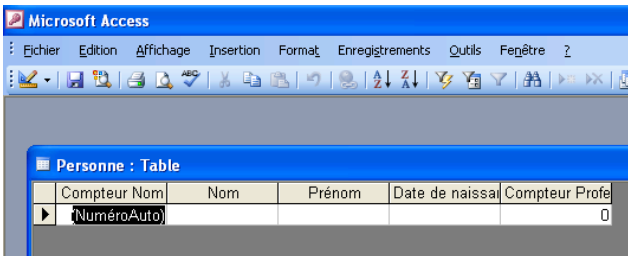


5- Définissons la clé primaire (pour accélérer les accès) :
Sélectionnons le champs `Compteur Nom` (qui est unique) puis cliquons sur la clé située sur la barre d'outil :





6- Rentrons en mode saisie en cliquant sur la première icône (à gauche) de la barre d'outils (représentant une table) ; ACCESS nous demande de sauvegarder ; nus sauvegardons avec le nom Personne. Remarquons que la première icône n'est plus une table.



7- Remplissons cette table :

Personne : Table

Compteur Nom	Nom	Prénom	Date de naissa	Compteur Profe
1	Dubois	Patrick	11/10/1975	28
2	Dupont	Ameline	20/03/1970	32
3	Azemar	Clothilde	12/12/1980	35
4	Balzac	Alain	07/06/1971	32
(NuméroAuto)				0

8- Créons une autre table sous le nom Profession :

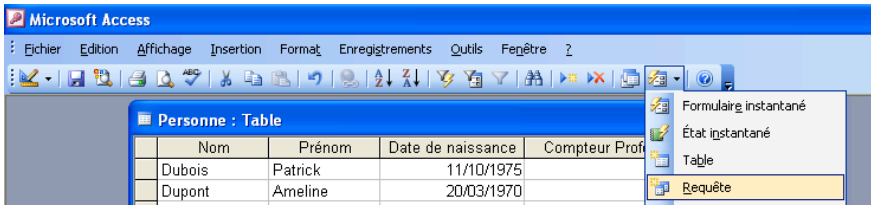
Profession

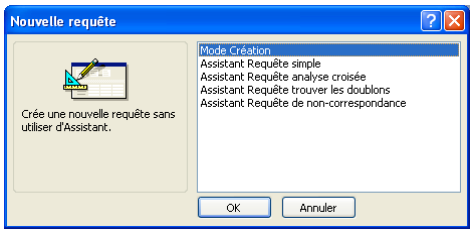
Compteur Profession	Profession	Indice
28	Agriculteur	25
32	Commerçant	550
35	Enseignant	121

II. Vues et Requêtes

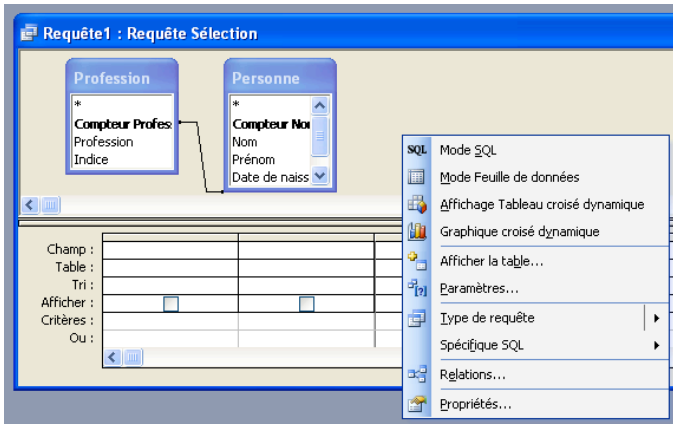
Une vue est une relation non matérialisée d'un shéma externe calculée à partir des relations de la base par une question. En ACCESS, nous parlerons de requêtes.

Soient nos deux tables : `Personne` et `Profession`. Nous allons créer une requête à partir de ces deux tables.
Dans la fenêtre Bases de données nous cliquons sur l'onglet `Requêtes` puis `Mode de création` :

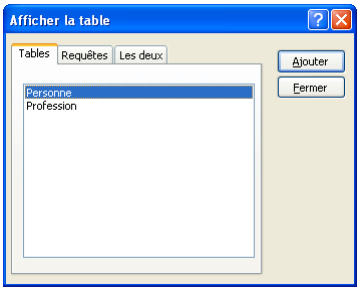




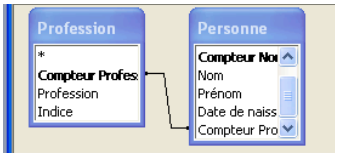
1- Dans le champs conceptuel de `Requête 1 : Requête sélection`, nous avons sélectionné `Afficher la table` …



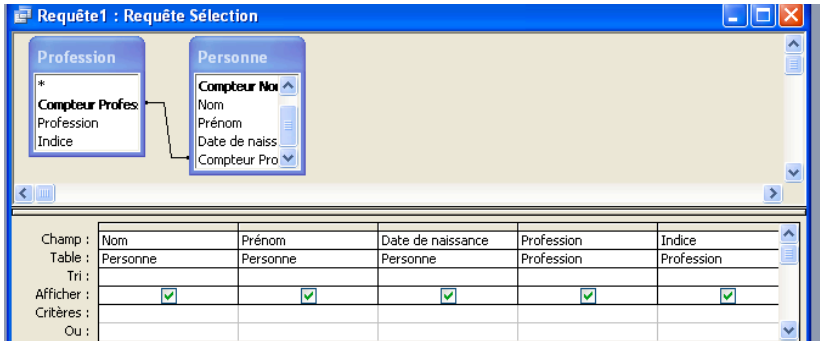
Puis les tables `Personne` et `Profession` :



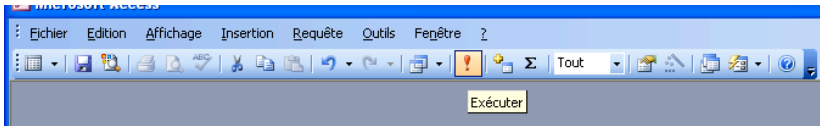
2- Réalisation des liens « à la main ». Ici le lien Compteur Profession est réalisé automatiquement :



Retirons cette jointure en cliquant dessus puis la remettre en cliquant sur le champs `Compteur Profession` de la table `Profession` et en amenant la souris sur celui de `Personne`. Ensuite cliquons sur les champs Nom, Prénom, Date de naissance de la table Personne et Profession, Indice de la table Profession (nous pouvons aussi rentrer ces données « à la main »). Nous avons sélectionné les attributs de notre requête :



Exécutons-la et savons-la (comme pour les tables) dans Requête1 (sinon ACCESS lui donnera ce nom par défaut) :

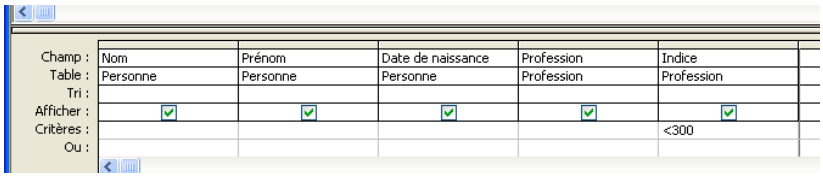


Requête1

Nom	Prénom	Date de naissance	Profession	Indice
Dubois	Patrick	11/10/1975	Agriculteur	25
Dupont	Ameline	20/03/1970	Commerçant	550
Azemar	Clothilde	12/12/1980	Enseignant	121
Balzac	Alain	07/06/1971	Commerçant	550

3- Requête avec contrainte

Nous voulons créer une nouvelle requête appelée `Requête2` qui comporte les **tuples** de `Requête1` tels que les indices soient inférieurs à 300 ; c'est une **restriction**. Dans le tableau de la `Requête2`, copie de `Requête1` pour le moment, rajouter < 300 à l'intersection de la ligne critère et de la colonne indice. Exécutons avec ! ou en cliquant sur le menu déroulant de la première case à gauche de la barre d'outils (choisir `Mode Feuille de données`).



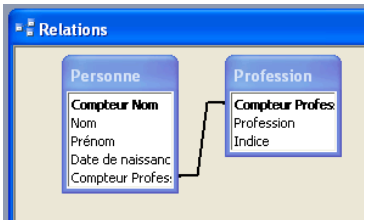
Requête2

Nom	Prénom	Date de naissance	Profession	Indice
Dubois	Patrick	11/10/1975	Agriculteur	25
Azemar	Clothilde	12/12/1980	Enseignant	121

4- Définition des relations entre les tables.

L'objectif est de pour avoir automatiquement des **jointures**. Allons dans Outils/Relations et ajouter les tables concernées. Faisons le lien entre les attributs concernés par la jointure (ici `Compteur Profession`) comme en 3-. La fenêtre suivante s'ouvre :

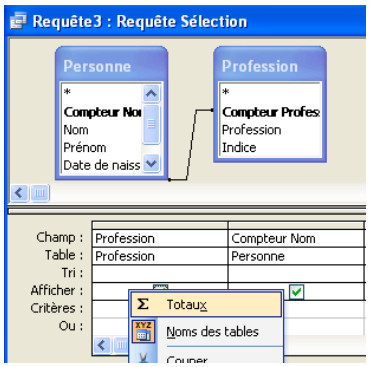
Cliquons sur `Appliquer l'intégrité référentielle` puis sur créer puis fermer relations.

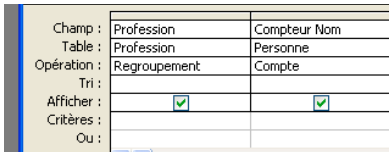


Désormais en ajoutant les tables, le lien sera automatiquement mis.

5- Les opérations sur les requêtes

Nous cherchons une requête qui nous donne le nombre de personnes par profession. Cliquer qui nous donne le nombre de personnes par profession. Créons une nouvelle requête. Ajoutons les tables `Personnes` et `Profession` et sélectionnons les champs `Professions` et `Compteur Nom`. Rajouter dans la requête la ligne `Opération` en sélectionnant `Totaux` dans `Afficher`. `Regroupement` est mis par défaut. Choisir `Regroupement` pour l'attribut `Profession` et `Compte` pour l'attribut `Compteur Nom`. Puis exécuter la requête.





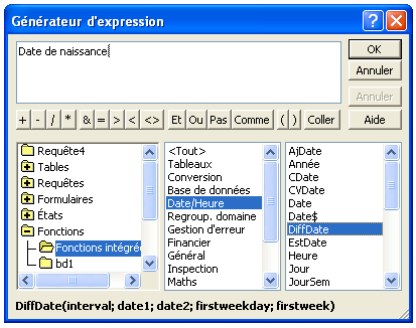
Exécutons notre requête :

Requête3	
Profession	**CompteDeCompteur Nom**
Agriculteur	1
Commerçant	2
Enseignant	1

III. Utilisation des fonctions dans les champs

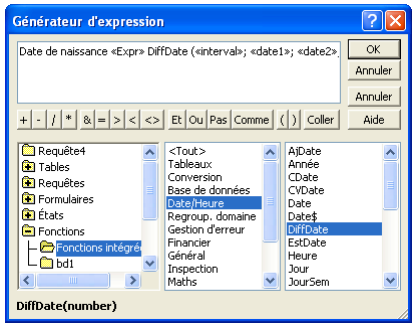
Nous cherchons à réaliser une nouvelle requête dans laquelle la date de naissance est remplacée par l'âge.

1- Créons une nouvelle requête avec l'attribut `Date de naissance` de la table `Personne` et l'attribut `Profession` de la table `Profession`.
2- Cliquons sur le champs Date de naissance puis sur `Créer`, la « baguette magique ».



Choisissons (double cliquer sur `DiffDate`)

`Fonction -> Fonctions intégrées -> Date/Heure -> DiffDate.`



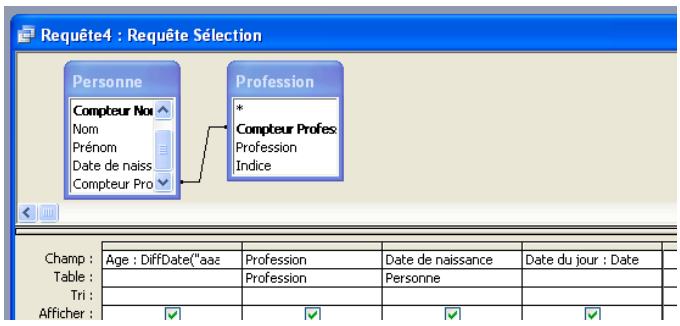
Dans la **zone Expression**, s'affiche un modèle de programme dans la fenêtre où nous pouvons taper le nôtre.

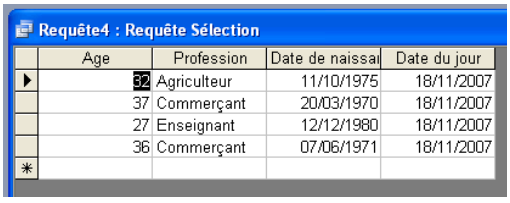
3- Tapons :

```
Age : DiffDate(''aaaa" ; Personne![Date de naissance] ;Date())
```

Dans la **zone Expression**.

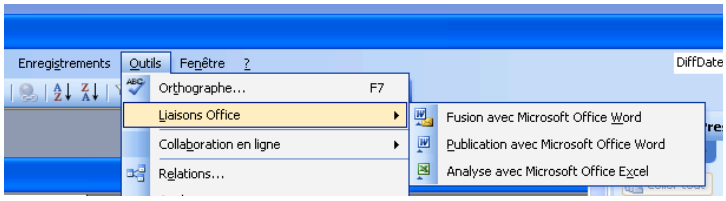
Pour vérifier le résultat, dans deux autres colonnes, nous demandons d'afficher la date de naissance et celle du jour.





Il ne reste plus qu'à donner un nom à la base de données.

IV. Liens avec Word et Excel

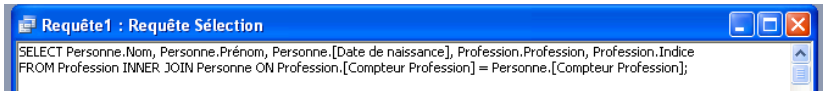


Nous essayerons en cours.

V. Passage à SQL

Choisissons `Affichage/Mode SQL` :

Pour Requête1, nous obtenons :



Pour Requête2 :

```
SELECT Personne.Nom, Personne.Prénom, Personne.[Date de naissance],
Profession.Profession, Profession.Indice
FROM Personne INNER JOIN Profession ON Personne.[Compteur Profession] =
Profession.[Compteur Profession]
WHERE (((Profession.Indice)<300));
```

Pour Requête3 :

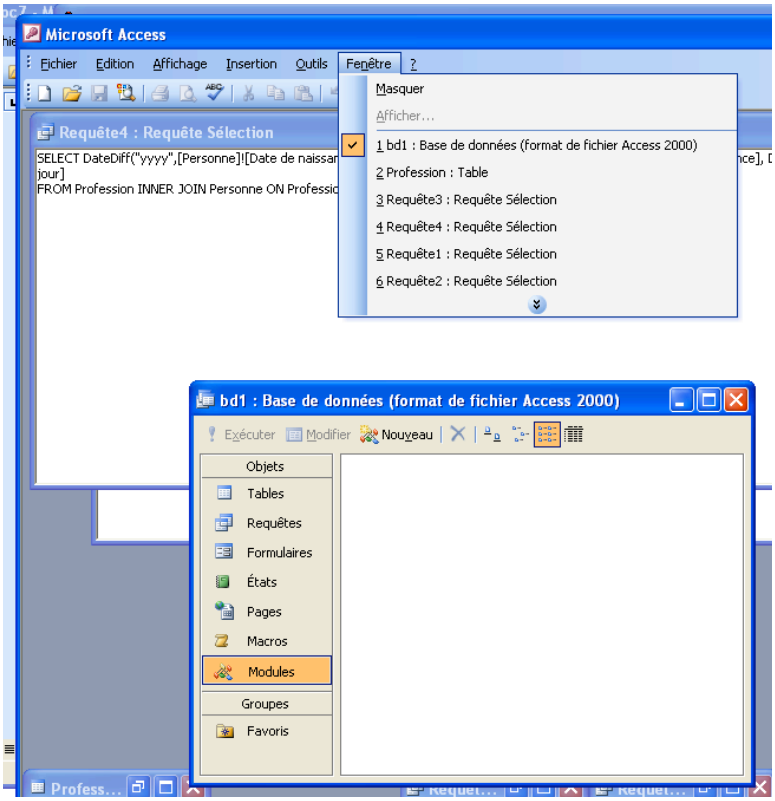
```
SELECT Profession.Profession, Count(Personne.[Compteur Nom]) AS
[CompteDeCompteur Nom]
FROM Profession INNER JOIN Personne ON Profession.[Compteur Profession] =
Personne.[Compteur Profession]
GROUP BY Profession.Profession;
```

Pour Requête4 :

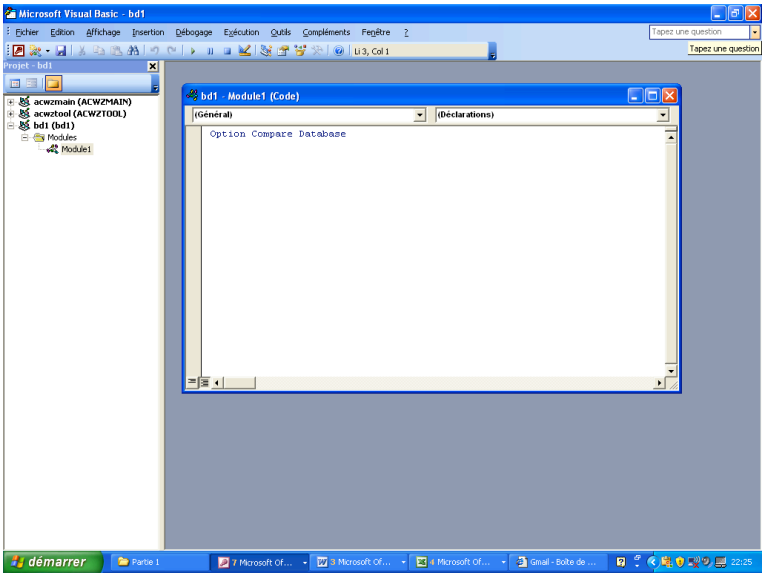
```
SELECT DateDiff("yyyy",[Personne]![Date de naissance],Date()) AS Age,
Profession.Profession, Personne.[Date de naissance], Date() AS [Date du
jour]
FROM Profession INNER JOIN Personne ON Profession.[Compteur Profession] =
Personne.[Compteur Profession];
```

VI. Et VBA ?

Sélectionnons **Modules** dans la base de données :



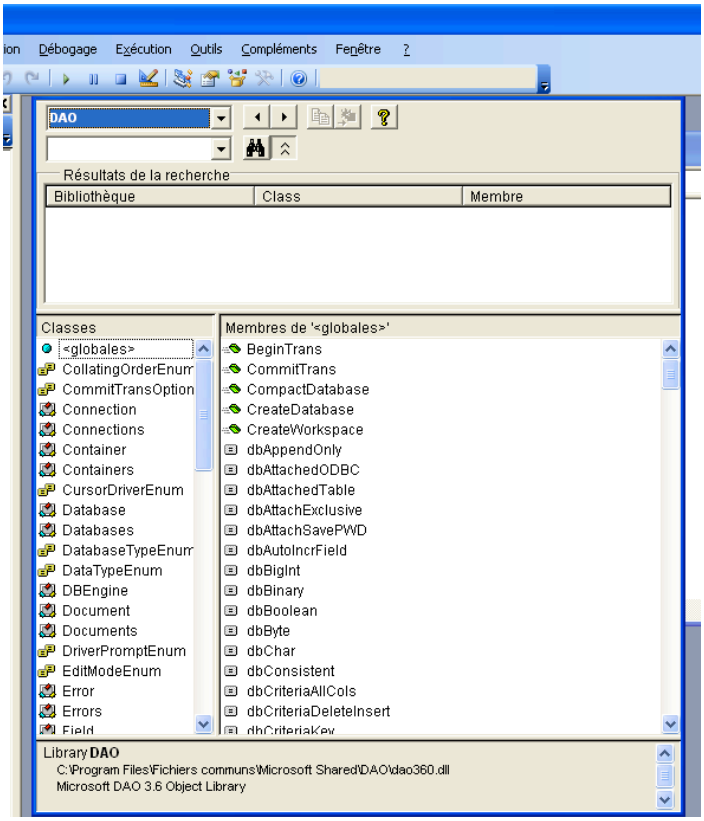
Nous nous retrouvons sous VBA (voir aussi `Affichage/Volet Office` ou bien `Affichage/Barres d'outils/Volet Office` ou `CTRL+F1`) :



Avec l'explorateur d'objet, nous consultons la documentation sur la classe `DataBase` , membre de la classe `DA0` (voir explorateur de projets), contenant en particulier :

Utilisez la méthode **CreateDatabase** pour créer un objet **Database** permanent automatiquement ajouté à la collection **Databases**, ce qui permet de l'enregistrer sur le disque dur.

Pour en savoir plus sur `CreateDatabase` consultons `DA0` :



Nous sélectionnons `CreateDataBase` :

Function CreateDatabase(Name As String, Locale As String, [Option]) As Database Membre de DAO.DBEngine

Nous regardons la méthode comme membre de Workspace :

Function CreateDatabase(Name As String, Connect As String, [Option]) As Database Membre de DAO.Workspace

Dans l'aide de `CreateDataBase`, nous trouvons :

CreateDatabase, méthode

Crée un objet Database, enregistre la base de données sur disque et renvoie un objet **Database** ouvert (espaces de travail Microsoft Jet uniquement).

Syntaxe

Set *database* = *workspace*.**CreateDatabase (***name***,** *locale***,** *options***)**

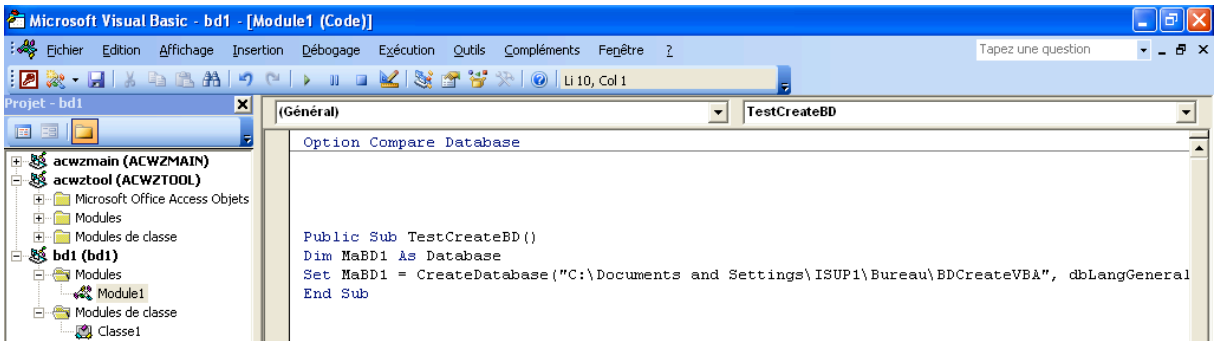
La syntaxe de la méthode **CreateDatabase** comprend les éléments suivants :

Élément	Description
database	Variable objet représentant l'objet **Database** à créer.
workspace	Variable objet représentant l'objet Workspace existant qui va contenir la base de données. Si vous omettez *workspace*, la méthode **CreateDatabase** utilise l'objet **Workspace** par défaut.
name	Donnée de type String d'une longueur maximale de 255 caractères, indiquant le nom du fichier de base de données à créer. Cet argument peut préciser le chemin et le nom de fichier complets, comme `"C:\db1.mdb"`. Si vous ne précisez aucune extension, .mdb est automatiquement ajouté. Vous pouvez également indiquer un chemin d'accès réseau, si votre réseau le permet, par exemple, `"\\server1\share1\dir1\db1"`. Cette méthode ne permet de créer que des fichiers .mdb.
locale	Expression de chaîne indiquant l'ordre de tri utilisé lors de la création d'une base de données, comme indiqué dans Valeurs. Si vous omettez cet argument, une erreur se produit.

Et encore :

La méthode **CreateDatabase** permet de créer et d'ouvrir une base de données vierge, puis renvoie l'objet **Database**. Il vous faut compléter sa structure et son contenu à l'aide d'objets DAO supplémentaires. Pour créer une copie complète ou partielle d'une base de données, vous pouvez utiliser la méthode CompactDatabase, qui permet de créer une copie personnalisée.

Essayons :



Sur le Bureau, cette BD nommée `BDCreateVBA` est effectivement crée et nous pouvons l'appeler à partir d'ACCESS.

Comme pour les classes sous Excel, nous constatons que l'explorateur d'objets est l'outil indispensable au développement des applications sous ACCESS via VBA.

A vous de continuer ! …

Chapitre 8 Langages Objets : quelques notions

Classe (Class) : Définition commune d'un ensemble
Sous-classe : un chien et un chat sont des animaux
Objet d'une classe : instance de cette classe
Collection : ensemble des objets d'une classe

Un objet possède l'ensemble des *méthodes* et *propriétés* définies dans la classe dont il est une instance.

Application hôte : Excel, Word , Access, ….
Nous travaillerons sous l'application Excel, mais nous pourrons faire appel aux objets des autres applications.

Sommet du modèle objet : Application
Sous l'objet Application : collection Workbooks englobant les objets Workbook (classeurs ouverts)
Sous l'objet Workbook : collection Worksheets des feuilles de calcul du classeur, les objets Worksheet (voir aussi Sheet).

Accéder à un Objet : les objets d'une collection sont repérables par leur indice dans cette collection ou par leur nom : NomCollections("NomObjet") où NomCollections est en fait une propriété de la classe NomCollection (i.e. de même nom sans s) retournant l'objet désigné en paramètre.

Appliquer une méthode, une propriété à un objet : Objet.méthode ou Objet.propriété.

Exemple : Application.Workbooks.("Classeur1").Sheets("Feuil1").Activate

Si Application.Workbooks.("Classeur1") est le classeur actif alors
Sheets("Feuil1").Activate suffit.

Propriétés (**attributs**) : modifications, interrogations. Types : chaîne de caractères, numérique, booléen, constante.

Méthodes : actions que l'objet peut exécuter.

d'instance : Objet.Méthode , Objet.Propriété
de classe : Classe. Méthode , Classe.Propriété

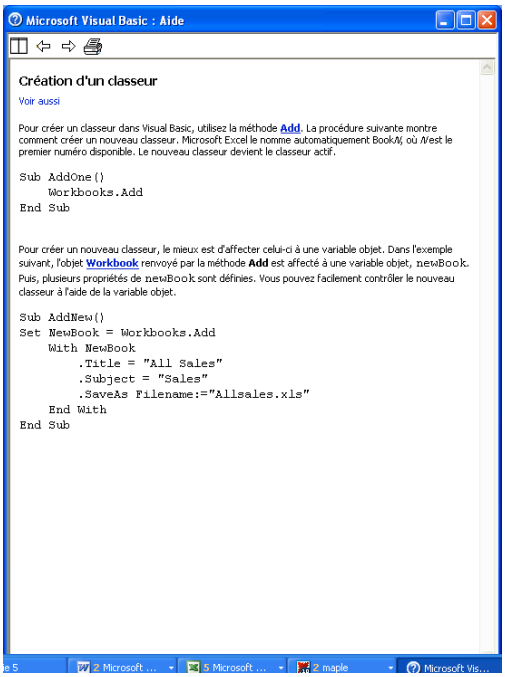
Exemple : fermer tous les classeurs : Workbooks.Close.
Créer un nouveau classeur : Workbooks.Add.

Evènements : actions reconnues par les objets. Ils déclanchent des programmes appelés *procédures évènementielles*.

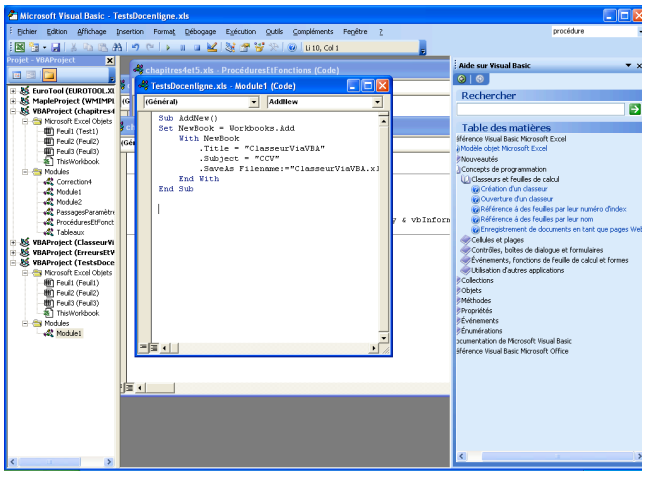
Fonctions : évaluation de leur corps dans un environnement dans lequel les paramètres formels sont associés aux paramètres réels et pouvant retourner un résultat.

Nous aborderons également l'héritage, le polymorphisme, …

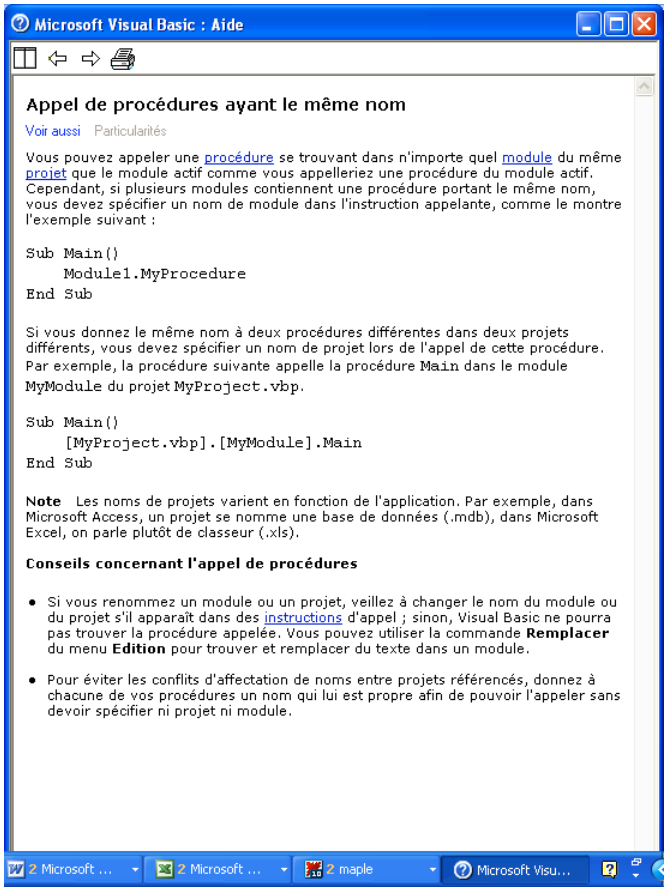
Création d'un classeur et Polymorphisme sous VBA



Je copie le code dans le module d'un projet (en l'occurrence le module 1 du projet TestDocenligne) et j'exécute. Le classeur ClasseurViaVBA est alors créé.



Nous poursuivrons cette docenligne en cours. En particulier, le cours 5 portera sur la partie « Visual Basic/Rubriques Conceptuelles ».


Microsoft Visual Basic : Aide
Appel de procédures ayant le même nom
Voir aussi Particularités
Vous pouvez appeler une procédure se trouvant dans n'importe quel module du même projet que le module actif comme vous appelleriez une procédure du module actif. Cependant, si plusieurs modules contiennent une procédure portant le même nom, vous devez spécifier un nom de module dans l'instruction appelante, comme le montre l'exemple suivant :
Sub Main()
Module1.MyProcedure
End Sub
Si vous donnez le même nom à deux procédures différentes dans deux projets différents, vous devez spécifier un nom de projet lors de l'appel de cette procédure. Par exemple, la procédure suivante appelle la procédure Main dans le module MyModule du projet MyProject.vbp.
Sub Main()
[MyProject.vbp].[MyModule].Main
End Sub
Note Les noms de projets varient en fonction de l'application. Par exemple, dans Microsoft Access, un projet se nomme une base de données (.mdb), dans Microsoft Excel, on parle plutôt de classeur (.xls).
Conseils concernant l'appel de procédures
Si vous renommez un module ou un projet, veillez à changer le nom du module ou du projet s'il apparaît dans des instructions d'appel ; sinon, Visual Basic ne pourra pas trouver la procédure appelée. Vous pouvez utiliser la commande Remplacer du menu Edition pour trouver et remplacer du texte dans un module.
Pour éviter les conflits d'affectation de noms entre projets référencés, donnez à chacune de vos procédures un nom qui lui est propre afin de pouvoir l'appeler sans devoir spécifier ni projet ni module.
2 Microsoft ...
2 Microsoft ...
2 maple
Microsoft Visu...

Chapitre 9 Cellules en VBA sous l'hôte Excel

Range : constructeur de la classe Range (Voir Chapitre 4)

Cells : propriété

Cet exemple montre comment affecter une taille de caractères de 14 points à la cellule C5 dans la feuille Sheet1.

```
Worksheets("Sheet1").Cells(5, 3).Font.Size = 14
```

Cet exemple montre comment supprimer la formule dans la cellule 1 de la feuille Sheet1.

```
Worksheets("Sheet1").Cells(1).ClearContents
```

Cet exemple montre comment affecter une police de caractères Arial et une taille de 8 points pour toutes les cellules dans la feuille Sheet1.

```
With Worksheets("Sheet1").Cells.Font
    .Name = "Arial"
    .Size = 8
End With
```

Cet exemple montre comment affecter le style italique aux cellules A1:C5 dans la feuille Sheet1.

```
Worksheets("Sheet1").Activate
Range(Cells(1, 1), Cells(5, 3)).Font.Italic = True
```

Cet exemple montre comment balayer une colonne de données intitulée « myRange ». Si une cellule contient la même valeur que celle située immédiatement au-dessus, l'adresse de la cellule contenant le doublon est affichée.

```
Set r = Range("myRange")
For n = 1 To r.Rows.Count
    If r.Cells(n, 1) = r.Cells(n + 1, 1) Then
        MsgBox "Duplicate data in " & r.Cells(n + 1, 1).Address
    End If
```

Cet exemple montre comment effectuer une boucle sur les cellules A1:J4 dans la feuille Sheet1. Si une cellule contient une valeur inférieure à 0,001, la valeur est remplacée par 0 (zéro).

```
For rwIndex = 1 to 4
    For colIndex = 1 to 10
        With Worksheets("Sheet1").Cells(rwIndex, colIndex)
            If .Value < .001 Then .Value = 0
        End With
    Next colIndex
Next rwIndex
```

Row : propriété

Cette propriété renvoie le numéro de la première ligne de la première zone de la plage. Type de données **Long** en lecture seule.

Cet exemple montre comment affecter une hauteur de 4 points à toutes les autres lignes de la feuille Sheet1.

```
For Each rw In Worksheets("Sheet1").Rows
    If rw.Row Mod 2 = 0 Then
        rw.RowHeight = 4
    End If
Next rw
```

Rows : propriétés qui renvoie un objet Range

Cet exemple montre comment supprimer la ligne 3 dans Sheet1.

```
Worksheets("Sheet1").Rows(3).Delete
```

Column et Column : idem Row

ActiveCell : propriété, construit un objet Range representant la cellule active de la feuille de calcul (voir Chapitre 3).

Selection : renvoie l'objet sélectionné dans la feuille de calcul (voir Chapitre 3).

Cette propriété renvoie l'objet sélectionné dans la fenêtre active pour un objet **Application**, et dans une fenêtre spécifiée pour un objet **Windows**.

Notes

Le type de l'objet renvoyé dépend de la sélection en cours (par exemple, si une cellule est sélectionnée, cette propriété renvoie un objet **Range**). La propriété **Selection** renvoie **Nothing** si rien n'est sélectionné.

Utiliser cette propriété sans identificateur d'objet équivaut à utiliser `Application.Selection`.

Exemples

Cet exemple montre comment effacer la sélection effectuée dans la feuille Sheet1 (en supposant que cette sélection concerne une plage de cellules).

```
Worksheets("Sheet1").Activate
Selection.Clear
```

Cet exemple montre comment afficher le type d'objet Visual Basic de la sélection.

```
Worksheets("Sheet1").Activate
MsgBox "The selection object type is " & TypeName(Selection)
```

Offset : renvoie un objet Range qui représente un objet décalé par rapport à la feuille de calcul (voir Exercice 2 Chapitre 3).

Cette propriété renvoie un objet **Range** qui représente une plage décalée de la plage spécifiée. En lecture seule.

expression.**Offset(*RowOffset*, *ColumnOffset*)**

expression Obligatoire. Expression qui renvoie un objet **Range**.

RowOffset Argument de type **Variant** facultatif. Nombre de lignes (valeur positive, négative ou égale à 0 (zéro)) de décalage à appliquer à la plage. Les valeurs positives correspondent à un décalage vers le bas et les valeurs négatives à un décalage vers le haut. La valeur par défaut est 0.

ColumnOffset Argument de type **Variant** facultatif. Nombre de colonnes (valeur positive, négative ou égale à 0 (zéro)) de décalage à appliquer à la plage. Les valeurs positives correspondent à un décalage vers la droite et les valeurs négatives à un décalage vers la gauche. La valeur par défaut est 0.

Exemples

Cet exemple montre comment appliquer un décalage de cellule de trois colonnes vers la droite et de trois lignes vers le bas à la cellule active de la feuille « Sheet1 ».

```
Worksheets("Sheet1").Activate
ActiveCell.Offset(rowOffset:=3, columnOffset:=3).Activate
```

Cet exemple suppose que la feuille « Sheet1 » contient un tableau doté d'une ligne d'en-tête. L'exemple montre comment sélectionner le tableau sans sélectionner la ligne d'en-tête. La cellule active doit se situer à un endroit quelconque du tableau avant l'exécution de l'exemple.

```
Set tbl = ActiveCell.CurrentRegion
tbl.Offset(1, 0).Resize(tbl.Rows.Count - 1, _
    tbl.Columns.Count).Select
```

Select : Sélectionner l'objet spécifié, méthode

Exemple

Cet exemple montre comment sélectionner les cellules A1:B3 dans la feuille « Sheet1 ».

```
Worksheets("Sheet1").Activate
Range("A1:B3").Select
```

Pour sélectionner une cellule ou une plage de cellules, utilisez la méthode **Select**. Pour activer une seule cellule, utilisez la méthode **Activate**.

Goto : Sélectionner l'objet spécifié, méthode

Cette méthode sélectionne une plage ou une procédure Visual Basic quelconque dans n'importe quel classeur et active celui-ci s'il ne l'est pas.

expression.**Goto(*Reference*, *Scroll*)**

expression Obligatoire. Expression qui renvoie un objet **Application**.

Reference Argument de type **Variant** facultatif. La destination. Il peut s'agir d'un objet **Range**, d'une chaîne de caractères comportant une référence de cellule en notation de type R1C1 ou d'une chaîne de caractères

contenant un nom de procédure Visual Basic. Si vous ne spécifiez pas cet argument, la destination est la dernière plage que vous avez sélectionnée à l'aide de la méthode **Goto**.

Scroll Argument de type **Variant** facultatif. Affectez-lui la valeur **True** pour faire défiler la fenêtre de telle sorte que le coin supérieur gauche de la plage apparaisse dans le coin supérieur gauche de la fenêtre. Affectez-lui la valeur **False** pour ne pas faire défiler la fenêtre. La valeur par défaut est **False**.

Notes

Cette méthode diffère de la méthode **Select** des façons suivantes :

- Si vous spécifiez une plage située dans une feuille qui ne se trouve pas au premier plan, Microsoft Excel bascule vers cette feuille avant de procéder à la sélection. (En revanche, si vous utilisez la méthode **Select** pour sélectionner une plage située dans une feuille qui ne se trouve pas au premier plan, la plage est sélectionnée mais la feuille n'est pas activée.)
- Cette méthode utilise un argument ***Scroll*** qui vous permet de faire défiler la fenêtre de destination.
- Lorsque vous utilisez la méthode **Goto**, la sélection précédente (celle qui précède l'exécution de la méthode **Goto**) est ajoutée dans le tableau des sélections précédentes (pour plus d'informations, consultez la rubrique relative à la propriété **PreviousSelections**). Vous pouvez utiliser cette caractéristique pour basculer rapidement entre quatre sélections au maximum.
- La méthode **Select** utilise un argument ***Replace***, ce qui n'est pas le cas de la méthode **Goto**.

Exemple

Cet exemple montre comment sélectionner la cellule A154 de la feuille Sheet1, puis faire défiler la feuille de calcul pour afficher la plage.

```
Application.Goto Reference:=Worksheets("Sheet1").Range("A154"), _
    scroll:=True
```

Resize : modifie l'ampleur d'une plage de cellules

Cette propriété redimensionne la plage spécifiée. Cette propriété renvoie un objet **Range** qui représente la plage redimensionnée.

expression.**Resize(*RowSize*, *ColumnSize*)**

expression Obligatoire. Expression qui renvoie un objet **Range** à redimensionner.

RowSize Argument de type **Variant** facultatif. Nombre de lignes de la nouvelle plage. Si vous n'avez pas spécifié cet argument, le nombre de lignes de la plage demeure inchangé.

ColumnSize Argument de type **Variant** facultatif. Nombre de colonnes de la nouvelle plage. Si vous n'avez pas spécifié cet argument, le nombre de colonnes de la plage demeure inchangé.

Exemples

Cet exemple montre comment étendre d'une ligne et d'une colonne la sélection effectuée dans la feuille « Sheet1 ».

```
Worksheets("Sheet1").Activate
numRows = Selection.Rows.Count
numColumns = Selection.Columns.Count
Selection.Resize(numRows + 1, numColumns + 1).Select
```

Cet exemple suppose que vous disposez d'un tableau avec ligne d'en-tête dans « Sheet1 ». L'exemple montre comment sélectionner le tableau sans sélectionner la ligne d'en-tête. La cellule active doit se trouver quelque part dans le tableau avant d'exécuter l'exemple.

```
Set tbl = ActiveCell.CurrentRegion
tbl.Offset(1, 0).Resize(tbl.Rows.Count - 1, _
    tbl.Columns.Count).Select
```

Chapitre 10 Calcul Formel sous Excel

Lors du cours 2, nous avons constaté notre insatisfaction face à l'utilisation du solveur sous Excel. Désormais, le système de calcul formel Maple est utilisable dans les feuilles de calcul Excel. C'est assez simple. Voyons ensemble de quelle manière.

Tout d'abord, voici comment aller chercher la macro complémentaire nécessaire à l'interfaçage.

1. Ouvrir la boite de dialogues Outils/Macros complémentaires.
2. Choisir Parcourir pour vous rendre dans le répertoire Maple 10. Par exemple **C:/Program Files/Maple 10/Excel/**.
3. Sélectionner le fichier WMIMPLEX pour l'insérer comme une macro complémentaire dans Excel. Observer les icônes Maple apparaître dans la feuille de calcul (voir Figure ci-après).

Regardons maintenant comment utiliser Maple.

1. Taper

```
=Maple("sin(x)")
```

Dans une cellule (ou dans la barre des formule).

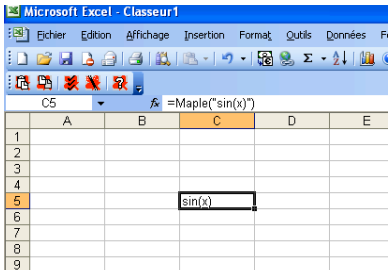


Figure 87

2. Essayer également

```
=Maple("plot(cos(x),x=1..10)")
```

Maintenant utiliser le mode création automatique de macros pour voir le code sous VBA.
Il est désormais possible de faire du Maple sous VBA avec Excel comme Application.

Pour en savoir plus, utiliser la documentation Maple en tapant : Excel.

Il est également possible d'utiliser des tableurs sous Maple (voir Documentation Maple).