République Algérienne Démocratique et Populaire

Ministère de l'enseignement Supérieur et de la Recherche Scientifique

Université Ferhat Abbas SETIF-1

Faculté des Sciences Economiques, Commerciales et de Gestion

Algorithmique et Programmation VBA pour Excel

Polycopié de Cours

2ème Année Finance et Comptabilité

Par

Dr. Benaouda Nacéra

Année Universitaire 2021-2022

Préface

Excel est un logiciel tableur offert par Microsoft dans le kit Microsoft Office fourni en accompagnement du système d'exploitation Windows. Il offre des fonctions de calcul, d'établissement de graphiques, d'extraction et d'analyse de données tels que les tableaux croisés dynamiques, ainsi que des outils d'optimisation à partir de systèmes d'équations tels que le solveur.

Le VBA (Visual Basic for Application) permet à l'utilisateur d'étendre Excel en définissant lui même ses propres fonctions pour répondre à ses besoins spécifiques. Pour les économistes, Excel constitue un outil de grande utilité qui leurs permet de mettre en œuvre les techniques relatives à leur domaine que ce soit dans le cadre de la recherche ou bien dans la gestion l'entreprise.

Ce polycopié de Cours est destiné aux étudiants en $2^{ème}$ année, Sciences Economiques, filière Finances et Comptabilité. L'objectif est d'initier l'étudiant à la programmation, et à l'utilisation d'Excel à partir de VBA. Nous l'avons structuré en trois parties: Une Première partie qui concerne la notion d'algorithmique et la programmation en Visual Basic. La deuxième partie concerne la manipulation d'Excel à partir de VBA. La troisième partie comporte les corrigés des exercices des chapitres.

Pour permettre à l'étudiant de passer rapidement à la pratique, nous avons présenté au fur et à mesure, en parallèle les structures algorithmiques ainsi que les structures VBA qui leurs correspondent.

Comme pré-requis, ce cours nécessite la maîtrise de la manipulation d'Excel.

Bien que les exemples présentés, soient réalisés sur la version 2007 d'Excel. La programmation en VBA reste toujours valable pour les versions plus récentes.

Chapitre 1. Sommaire

Chapitre 2. Introduction

2.1 Etapes de l'informatisation

L'informatisation d'un problème consiste à écrire le programme que doit exécuter l'ordinateur pour fournir les solutions souhaitées au problème posé. La réalisation de cette opération se fait en deux phases : une première phase qui a lieu en dehors de l'ordinateur, et qui consiste à analyser le problème jusqu'à aboutir à l'écriture d'un algorithme ou d'un organigramme représentant les étapes par lesquelles doit passer l'ordinateur pour résoudre et fournir les solutions. Une deuxième phase consiste à traduire l'organigramme ou l'algorithme de la phase précédente dans un langage évolué (comme C, VBA,…), saisir le programme traduit, le faire traiter par ordinateur pour le transformer en programme exécutable dont l'exécution nous permettra d'obtenir les résultats souhaités.

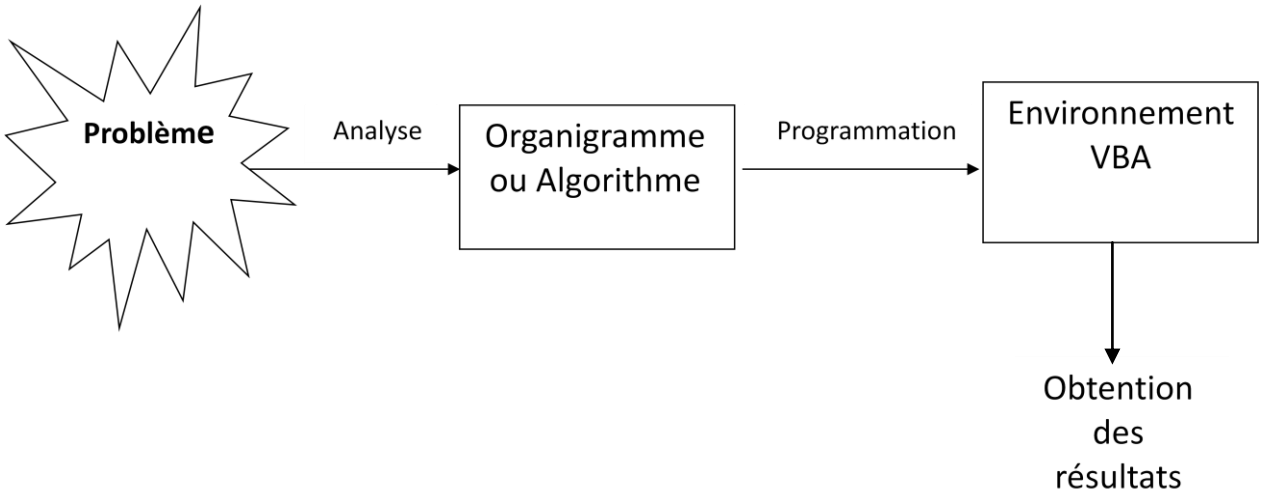


Figure 1: Les étapes de l'informatisation (Programmation en VBA)

2.2 La phase1 : Analyse du problème

L'analyse d'un problème consiste à étudier tous ses aspects et contraintes, et déterminer les étapes de sa résolution ainsi que les différents types et rôles des données sur lesquelles doivent être appliquées les différentes étapes. L'écriture formelle et unifiée de ces étapes peut être l'algorithme ou l'organigramme.

2.2.1 Exemple introductif

Problème => Résolution de l'équation $Ax2 + Bx + C = 0$

L'analyse :

Les données : A, B, C

Les résultats souhaités: les racines x1, x2

Les étapes que doit suivre l'odinateur:

1°) lire A, B, C

2°) Tester A

3°) Si A=0 tester B

4°) Si B=0

5°) Si C=0 Infinité de solutions, n'importe quelle valeur de x vérifie l'équation.

6°) S C≠0 Impossible. Il n' y a pas de solutions.

7°) Si B≠0 écrire x=-C/B

8°) Si A≠0 calculer Δ= B2 -4AC

9°) Tester Δ

10°) Si Δ < 0 pas de solutions dans l'ensemble des réèls

11°) Si Δ = 0 solution double écrire x = -B/2A

12°) Si Δ >0 deux racines : écrire x1= (-B- racine(Δ)) /2A et x2= (-B+racine(Δ)) /2A.

✓ Discussion :

1°) Les étapes telles qu'elles ont été décrites précédemment comportent : Des lectures / écritures, des affectations, des tests.

Tous ces éléments ont leurs équivalents dans le langage algorithmiques. Ce dernier fournit des syntaxes unifiées, pouvant être utilisées, et sont compréhensibles de la même manière par n'importe qui.

2°) Les données

Il existe principalement deux types de données :

- A, B, C que l'ordinateur doit lire en entrée, elles peuvent être des valeurs réelles ou entières selon ce qu'elles représentent dans la réalité (des densités, des tailles, des poids, etc …) .
- Les racines x1, x2 que l'ordinateur doit fournir et afficher à l'utilisateur, elles sont forcément réelles car elles sont calculées à l'aide de formules englobant l'opération de division.
- En mathématiques les variables et coefficients figurant dans les équations sont des variables abstraites, qui n'ont de sens qu'à travers leurs valeurs et le rôle qu'elles jouent. Pour un ordinateur toutes ces variables correspondent à des emplacements adressés en mémoire centrale.

Une fois les cases mémoire définies et les étapes de la solution bien déterminées; il s'agit d'écrire ces étapes dans le langage algorithmique pour obtenir un Algorithme qui sera par la suite traduit dans un langage évolué, qui est dans notre cas le VBA (Visual Basic for

Application). Ce programme doit être saisi sous l'éditeur VBE (Visual Basic Editor), ce dernier est un environnement de programmation offrant toutes les fonctionnalités nécessaires pour traiter et exécuter un programme écrit en VBA.

Le langage VBA est un langage de programmation fourni avec les applications de Microsoft pour bureautique à savoir, Excel, Word, Access, etc…. Il possède toutes les structures de base du langage Visual Basic, mais il a en plus une couche objet qui permet de manipuler les composants des applications bureautiques tels que les feuilles Excel et les cellules d'une feuille Excel, etc ….

Nous commençons par présenter, les concepts de l'algorithmique. Pour simplifier et faciliter l'assimilation d'un côté, et anticiper la pratique sur ordinateur d'un autre côté, nous présentons au fur et à mesure la traduction en VBA des structures algorithmiques présentées.

Chapitre 3. Les Algorithmes

3.1 Définitions : algorithme, algorithmique

Un **algorithme** est un énoncé écrit dans un langage bien défini d'une suite d'opérations permettant de donner la réponse à un problème. Il représente les étapes ordonnées que doit exécuter un ordinateur pour fournir les résultats à un problème. Le terme **algorithmique** désigne l'ensemble des activités logiques qui relèvent des algorithmes ; c'est-à-dire l'ensemble des règles et techniques qui sont impliquées dans la définition et la conception des algorithmes. L'organigramme est la représentation graphique de l'algorithme.

Pour être traité par ordinateur, l'algorithme doit être traduit en langage évolué, le VBA dans notre cas. Sous l'onglet développeur d'Excel, on peut accéder à l'environnement de programmation pour saisir un programme et l'exécuter.

3.2 Caractéristiques d'un algorithme

Comme l'algorithme est un moyen pour le programmeur de présenter son approche du problème à d'autres personnes. Il doit être donc :

- Lisible: l'algorithme doit être compréhensible même par un non-informaticien
- De haut niveau: l'algorithme doit pouvoir être traduit dans n'importe quel langage de programmation, il ne doit donc pas faire appel à des notions techniques relatives à un programme particulier ou bien à un système d'exploitation donné
- Précis: chaque élément de l'algorithme ne doit pas porter à confusion, il est donc important de lever toute ambiguïté
- Concis: un algorithme ne doit pas être très long (dépasser une page par exemple). Si c'est le cas, il faut décomposer le problème en plusieurs sous-problèmes
- Structuré: un algorithme doit être composé de différentes parties facilement identifiables

NB. Pour apprendre à programmer, il est nécessaire de commencer par l'algorithmique afin d'acquérir les notions fondamentales et générales de la programmation. Une fois ces notions maîtrisées, le passage au langage évolué devient tout simplement une traduction, tout en tenant compte des particularités du langage évolué cible, en question.

Nous présentons ci-après, la structure d'un algorithme suivie de toutes les syntaxes et structures utilisées avec des exemples. Nous donnerons à chaque fois les structures correspondantes en VBA.

3.3 Structure d'un algorithme

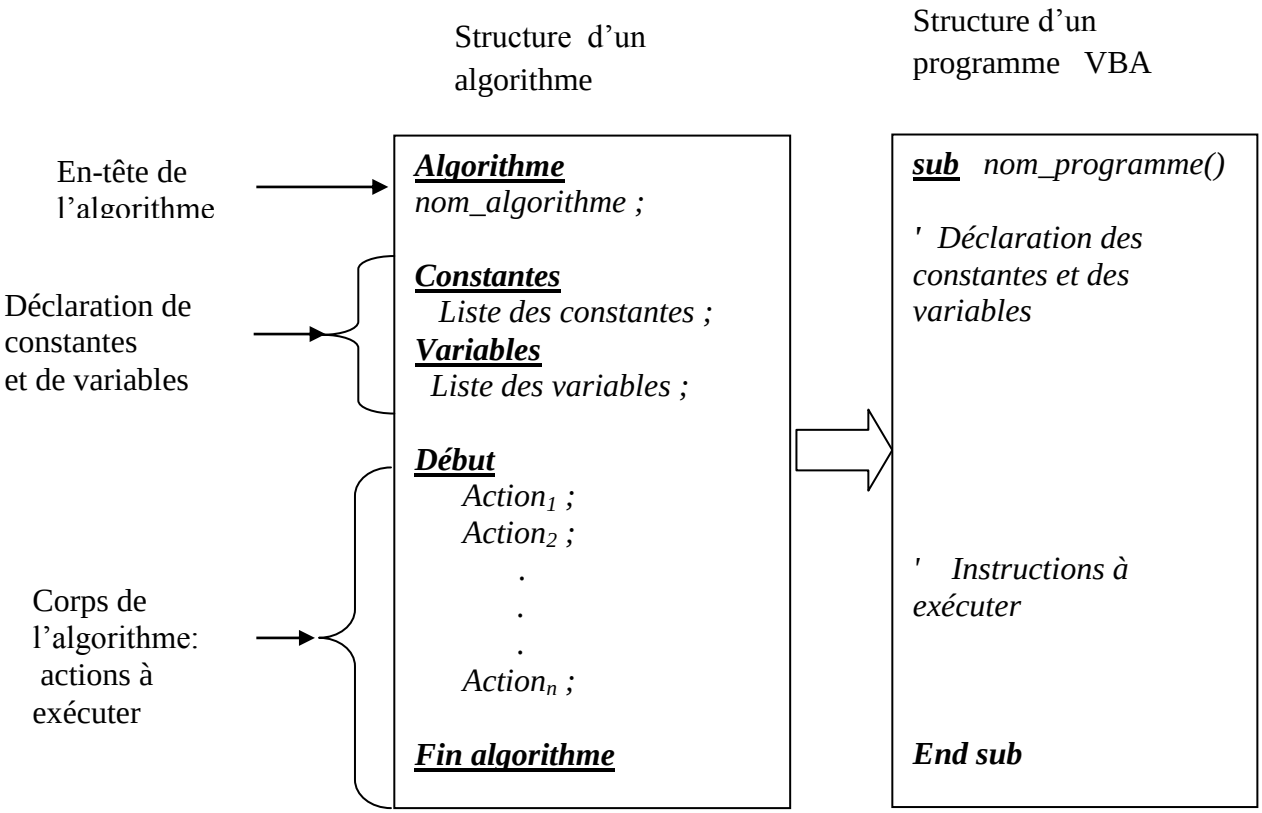


Figure 2: La structure générale d'un algorithme et la structure correspondante du programme VBA

L'algorithme est composé de 4 parties principales :

Entête de l'algorithme, déclarations de données (constantes et variables), déclaration de procédures et fonctions, corps de l'algorithme englobant les actions à exécuter.

3.3.1 L'en-tête

Permet d'identifier l'algorithme en lui attribuant un nom choisi par le programmeur.

	Algorithmique	VBA
Syntaxe	*Algorithme nom_algorithme*;	*Sub nom_programme()*
Exemple	*Algorithme calcul ;*	*Sub calcul()*

3.3.1.1 La déclaration des données

Dans le langage algorithmique, deux types de données sont considérés : Les constantes et les variables.

3.3.1.2 Les Constantes

Une constante représente un chiffre, un nombre, un caractère ou une chaîne de caractère dont la valeur ne peut pas changer au cours de l'exécution du programme. Les constantes sont déclarées dans la partie **Constantes** de l'algorithme.

	Algorithmique	VBA
Syntaxe	***Constantes*** *Nom_const1=valeur1 ;* *Nom_const2=valeur2 ;* *…*	*Const nom_const1 =valeur1 ,* *nom_const2=valeur2,…*
Exemple	***Constantes*** *Pi=3,14 ;* *Module = "Informatique" ;*	*const Pi=3,14, Module="Informatique"*

3.3.1.3 Les variables

Une variable représente un chiffre, un nombre, un caractère ou une chaîne de caractères dont la valeur peut changer au cours de l'exécution de l'algorithme. Une variable est caractérisée par son nom et son type.

	Algorithmique	VBA
Syntaxe	***Variables*** *Liste de variable : type1 ;* *Liste de variable: type2 ;* *…..*	*dim nom_variable1 as type1,* *nom_variable2 as type2,…*
Exemple	*Variables* *x, y : entier ;* *somme, moyenne : réel ;* *test : booléen ;* *nom, prenom : chaîne de caractères;* *c : caractère ;*	*dim x as integer, y as integer* *dim somme as single, moyenne as currency* *dim test as Boolean* *dim nom as string, prenom as string* *dim c as string*

Où liste de variables représente une liste de variables. C'est-à-dire que dans l'algorithme une liste de variables peut être associée à un même type, par contre en VBA on associe chaque variable à son type.

Au niveau algorithmique, Le type d'une variable peut être :

- En tier : qui couvre l'ensemble des valeurs entières : {…-4, -3, -2,-1, 0, 1, 2, 3, 4,…}

- réel : qui couvre l'intervalle des valeurs]-∞, +∞[
- caractère : qui couvre l'ensemble des caractères alphabétiques {A,…,Z, a,…,z}, numériques {0,…,9} ou caractères spéciaux {=,+,*,$, £,…} il est codé sur un octet.
- Chaîne de caractères : permet de déclarer des variables dont la valeur peut être une suite de caractères comme un nom, un prénom, une adresse,…. Une chaîne de caractères composée de N caractères est codée sur N octets.
- Booléen : qui peut prendre deux valeurs : VRAI ou FAUX.

Par contre en VBA, il existe plusieurs types entiers et plusieurs types réels: les types entiers en VBA sont : Byte, Integer et long; les types réels sont : Currency, Single et Double. Que ce soit dans l'entier ou dans le réel, les types VBA diffèrent selon l'intervalle de valeurs qu'ils couvrent. En pratique, une fois arrivé au VBA, c'est au programmeur de choisir le type entier (byte, integer ou long) ou réel (currency, single ou double) qui lui convient selon l'intervalle de valeurs qui peuvent être prises par ses variables. Le tableau en Annexe1 montre l'intervalle de valeurs couvert par chaque type en VBA

3.3.1.4 Les mots réservés

Un mot réservé dans le langage algorithmique est un mot qui appartient au langage, et qui ne peut pas être utilisé comme nom de variable, ou de constante ou de n'importe quelle structure utilisée dans l'algorithme. Les mots réservés du langages algorithmique sont :Algorithme, FinAlgorithme, variables, constantes, début, toutes les instructions (lire, écrire,etc…), etc…). La notion de mot réservé existe dans le langage algorithmique ainsi que dans tous les langages de programmation. En VBA par exemple, les mots réservés sont: (Toutes les instructions, sub, Dim, const, redim, etc…)

3.3.1.5 Le choix des identificateurs

Au niveau algorithmique, Le nom d'une variable, d'une constante ou de l'algorithme est choisi par l'utilisateur, et est appelé respectivement l'identificateur de la variable, de la constante ou de l'algorithme, et est représenté par une chaîne de caractères qui peut, généralement, être : formé de chiffres et de lettres, et commencer obligatoirement par une lettre. Cette règle est acceptée, par tous les langages de programmation, mais chaque langage peut apporter une spécificité dans le choix des identificateurs: le VBA, par exemple, accepte le caractère "_" qu'on appelle tiret bas (le tiret de la touche du chiffre 8) dans un identificateur.

Dans tous les cas, une règle en or est à respecter en programmation: Dans le choix du nom d'une variable ou constante, il est conseillé de choisir des noms qui correspondent au rôle de la variable ou de la constante, pour faciliter la lecture et la mise au point de l'algorithme ou du programme. Par exemple, il faut appeler 'nom' une variable qui contient le nom d'une personne comme 'Benali'.

3.3.1.6 Le Commentaire

Les commentaires sont des expressions (ou du texte) qui est inséré dans le programme ou l'algorithme. Leurs rôle est de documenter le programme pour le rendre plus facile à comprendre et à mettre au point. Ils n'appartiennent pas au langage, et ne sont pas traités par les compilateurs ou interpréteurs. Dans le langage algorithmique un commentaire commence et se termine par le caractère #. Pour les langages de programmation évolués, chaque langage offre une syntaxe propre. En VBA par exemple un commentaire commence par une simple côte " ' ", et est coloré en vert au niveau de l'éditeur.

3.3.2 Les instructions exécutables

3.3.2.1 Les entrées / sorties

3.3.2.1.1 Les entrées (la lecture)

On utilise pour la lecture dans le langage algorithmique l'instruction lire de syntaxe générale :

	Algorithmique	VBA
Syntaxe	*Lire (variable1, variable2,...);*	*Variable1=inputbox("message")* *Variable2=inputbox("message")* *...*
Exemple	*Lire (A, B, X) ;*	*A=inputbox("Donnez A")* *B=inputbox("Donnez B")* *X=inputbox("Donnez X")*

A l'exécution de cette instruction, l'utilisateur tape sur son clavier 3 valeurs : la première sera stockée dans la variable A, la deuxième dans B et la troisième dans X. Sachant que A, B et X sont des cases mémoires déclarées dans la partie **Variables** de l'algorithme. En VBA Inputbox permet de lire une variable à la fois dans une fenêtre avec affichage d'un message d'éclaircissement.

3.3.2.1.2 Les sorties (l'écriture)

On utilise pour l'écriture dans le langage algorithmique l'instruction écrire de syntaxe générale :

	Algorithmique	VBA

Syntaxe	*écrire (expression1, expression2,…) ;*	*Msgbox expression1 & expression2 & ….*
Exemple	*Ecrire (x, a+b+1, "Informatique") ;*	*Msgbox x & a + b +1 & "Informatique"*

Sachant que expression peut être une constante, une variable, une expression arithmétique ou logique.

A l'exécution de cette instruction l'ordinateur affiche sur l'écran, la valeur de la variable x ; la valeur de l'expression a+b+1 et la chaîne de caractères "Informatique". En VBA on utilise Msgbox qui fait l'affichage dans une fenêtre.

NB. Dans un programme l'utilisateur manipule des identificateurs de variables. L'ordinateur manipule des adresses. C'est-à-dire qu'à chaque nom de variable est associée une adresse en mémoire centrale.

3.3.2.2 L'affectation

Cette instruction permet d'affecter la valeur d'une expression à une variable.

	Algorithmique	VBA
Syntaxe	*Nom_variable ← expression ;*	*Nom_variable = expression*
Exemple	*A ← 2* *X ← Y* *Z ← X + A* *I ← I +1*	*A=2* *X=Y* *Z=X+A* *I=I+1*

Où Nom_variable représente une variable déclarée dans la partie **variables** de l'algorithme.

Expression peut être :

Une valeur constante. (2 par exemple)

Un nom de variable (x par exemple)

Une combinaison de variables ou constantes séparées par des opérateurs arithmétiques ou logiques. (a+b+c par exemple)

Exemple :

A ← 2 mettre 2 dans la variable A.

X ← Y mettre le contenu de la variable Y dans la variable X.

Z ← X + A ajouter le contenu de A au contenu de X et mettre le résultat dans Z.

I ← I +1 Ajouter 1 au contenu de la variable I (incrémenter la variable I de 1)

ie. La nouvelle valeur de I est égale à l'ancienne valeur de I plus 1.

Exemple 1

Ecrire l'algorithme qui lit 3 nombres entiers a, b, c, calcule la somme et la moyenne et les affiche. Ecrire le programme VBA correspondant.

Algorithme	**Programme VBA**
Algorithme calcul ; *Variables* *A, b, c , somme : entier ;* *Moyenne : réel ;* *Début* *1←* *Lire (a, b, c) ;* *2←* *Somme ← a + b +c ;* *3←* *Moyenne ← somme / 3 ;* *4←* *Ecrire (somme, moyenne) ;* *5←* *fin algorithme*	*Sub calcul()* *Dim a As integer, b As integer, c As integer* *dim somme As integer, moyenne As Double* *a = InputBox("Donnez a")* *b = InputBox("Donnez b")* *c = InputBox("Donnez c")* *somme=a+b+c* *moyenne=somme/3* *MsgBox "somme=" & somme & " moyenne=" & moyenne* *End Sub*

Remarque

Les numéros entre les lignes de l'algorithme (1←, 2←, 3←, 4←, 5←) représentent des points d'observation dans l'algorithme, et sont utilisés pour le déroulement de l'algorithme. A chacun de ces points, c'est-à-dire après l'exécution de chaque instruction, nous consultons les valeurs des cases mémoire utilisées; ceci permettra de déduire l'action de l'instruction exécutée sur les variables. Par exemple au point 1, A, B, C sont toutes égales à zéro. Au point 2, A devient 16, B devient 14 et C devient 12, etc…

Déroulement de l'algorithme

Nous supposons que les valeurs lues au début sont :

16 dans A, 14 dans B, 12 dans C.

	A	B	C	Somme	moyenne
1	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~

2	16	14	12	Ø	Ø
3	16	14	12	42	Ø
4	16	14	12	42	14
5	16	14	12	42	14

Exemple 2

Ecrire un algorithme qui lit le nom, prénom, le nombre d'heures d'un employé vacataire, et calcule son salaire et l'affiche sachant que le salaire par heure est fixé à 300 DA, et que le salaire est calculé par la formule :

Salaire = salaire par heure X nombre d'heures

Les variables utilisées :

Nom : de type chaine de caractères pour le nom

Prenom : de type chaine de caractères pour le prénom

Nbheure : de type entier pour le nombre d'heures

Salaire : de type réel pour le salaire.

Les constantes SalaireH = 300

Algorithme	Programme VBA
Algorithme calculsalaire ; *Constantes* *salaireH= 300;* *Variables* *Nom, Prenom : chaîne;* *Nbheure : entier ;* *Salaire : réel ;* *Début* *1←* *Lire (nom, prenom, nbheure) ;* *2←* *Salaire ← nbheure * salaireh ;* *3←* *Ecrire(Nom, prenom, salaire) ;* *4←* *Fin algorithme*	*Sub calculsalaire()* *Const salaireH = 300* *Dim nom As String, prenom As String* *Dim nbheure As Byte, salaire As Double* *nom = InputBox("Donnez le nom")* *prenom = InputBox("Donnez le prenom")* *nbheure = InputBox("Donnez le nombre d'heures")* *salaire = nbheure * salaireH* *MsgBox nom & " " & prenom & " " & salaire* *End Sub*

Déroulement

Supposons qu'on lise :

'Benali' dans le nom et 'omar' dans le prénom, et 20 dans nbheures.

	Nom	Prenom	Nbheure	Salaire
1	' '	' '	⁄0	⁄0
2	'Benali'	'Benali'	20	⁄0
3	'Benali'	'Benali'	20	6000
4	'Benali'	'Benali'	20	6000

3.3.3 Comment accéder à VBA/Excel

Pour écrire un programme en VBA sous Excel, il faut suivre la démarche suivante:

1°) Ouvrir Excel

2°) Appuyer sur ALT-F11 ou cliquer sur l'onglet **Développeur.**

3°) Cliquer sur **insertion**, ensuite **module** pour créer un module.

4°) Cliquer sur **insertion**, ensuite **procédure** pour créer un programme ou **sub().**

Et dans tous cas suivre les boîtes de dialogues qui s'affichent à l'utilisateur.

Le mode d'utilisation de l'éditeur VBE est expliqué en détail dans [7]

3.3.4 Les expressions arithmétiques et logiques

Dans le cas général, une expression est une combinaison de variables et/ou constantes séparées par des opérateurs arithmétiques, logiques ou de comparaison. L'évaluation de cette expression est soumise à un ordre prédéterminé basé sur la priorité des opérateurs. Nous adoptons dans ce cours la priorité des opérateurs mise en œuvre en VBA.

3.3.4.1 Les expressions arithmétiques

Nous présentons dans le tableau ci-dessous, les opérateurs arithmétiques dans l'ordre décroissant de leur priorité.

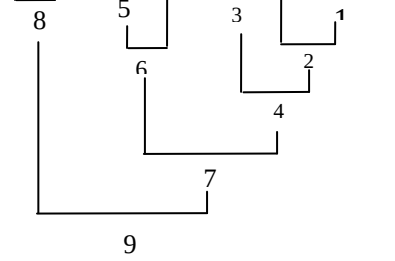
Opérateur arithmétique	**Algorithmique**	**VBA**	**Interprétation**
Elévation à la Puissance	** *ou* ^	^	*2^3=2*2*2=8*
Moins unaire ou Négation	-	-	*Le signe – précédant un opérande*
Multiplication et division	*, /	*, /	*3,5*3=10,5 5/2= 2,5*
Division entière	***Div***	\	*5 Div 2=2 c'est uniquement, la partie entière qui est donnée dans le résultat*

Arithmétique modulus (Mod)	***Mod***	***Mod***	*5 mod 2=1 signifie le reste de la division entière de 5 par 2 est 1*
Addition et soustraction (+, -)	**+, -**	**+, -**	*Addition et soustraction ordinaires des nombres*
Concaténation de chaînes (&)	**&**	**&**	*"J'aime " & "Excel" équivalent à "J'aime Excel"*

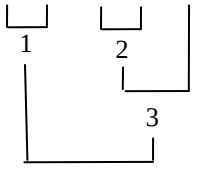
L'évaluation des expressions arithmétiques se fait de gauche à droite en respectant la règle de priorité précédente. L'utilisation des parenthèses casse cette règle de priorité en rendant l'expression entre parenthèses prioritaire au cours de l'évaluation, tout en respectant la règle de priorité à l'intérieur des parenthèses.

Exemples :

1°) A+12-14*4/13*(z+1-2*(n+5))



2°) B * b - 4 * a * c



3.3.4.2 Les expressions logiques

Les conditions figurant dans une instruction de test sont appelées aussi prédicat et peuvent avoir la valeur Vrai ou Faux. Elles sont représentées par des expressions logiques. On peut distinguer deux types d'expressions logiques : simples et complexes.

L'expression logique simple est composée de deux expressions arithmétiques séparées par un opérateur de relation (de comparaison).

Les opérateurs de comparaison

Comparaison	**Algorithmique**	**VBA**
Égalité	=	=
Inégalité	≠	<>
Inférieur à	<	<
Supérieur à	>	>
Inférieur ou égal à	≤	<=
Supérieur ou égal à	≥	>=

Syntaxe : expr1 OP expr2

Exemple : x+y > t+3

Z ≠ t

L'expression logique complexe est une combinaison d'expressions logiques simples séparées par des opérateurs logiques et éventuellement des parenthèses.

Notation

Supposons deux variables logiques booléennes A et B.

A=faux ⇔ non A

A=Vrai ⇔ A

L'évaluation des expressions logiques :

L'expression logique est évaluée de gauche à droite en tenant compte de l'ordre de priorité suivant :

Priorité des opérateurs logiques

Algorithmique	**VBA**	**Interprétation**
NON	NOT	Négation
ET	AND	Intersection
OU	OR	Union

Dans une expression logique complexe, les opérateurs arithmétiques sont évalués en premier, suivis des opérateurs de comparaison et des opérateurs logiques. Les opérateurs de comparaison ont tous une priorité égale; c'est-à-dire qu'ils sont évalués dans leur ordre d'apparition de gauche à droite. L'utilisation des parenthèses influe sur cet ordre prédéfini en rendant les expressions entre parenthèses plus prioritaires.

Tableaux d'évaluation de base pour les expressions logiques

Soient A et B deux conditions.

1) La négation

A	Non A
Vrai	Faux
Faux	Vrai

2) L'Union

A OU B	Vrai	Faux
Vrai	Vrai	Vrai
Faux	Vrai	Faux

3) L'Intersection

A et B	Vrai	Faux
Vrai	Vrai	Faux
Faux	Faux	Faux

3.4 Exercices

Exercice1

Soit la portion d'algorithme suivante. Donnez le type convenable pour chacune des variables: A, B, C, D, E, F, G. Quels sont les types possibles en VBA?

A ← -3 ;

B ←0.5 ;

C ←'1' ;

D ←C & '***' ;

E ←1 ;

F ← 15/2;

G ← F>B;

Exercice2

Soient les variables de type entier x, y, temp, ainsi que les trois séquences d'actions, agissant sur elles, suivantes :

Séquence 1	*Séquence 2*	*Séquence 3*
Lire(x, Y) ;	*Lire(X, Y) ;*	*Lire(X, Y) ;*
X← Y ;	*Temp ← X ;*	*Temp ← Y ;*
Y← X ;	*X← Y ;*	*Y← X ;*
Ecrire(x,y) ;	*Y←Temp ;*	*X←Temp ;*
	Ecrire(x,y) ;	*Ecrire(x,y) ;*

En supposant qu'on lise :

- 2 dans X et 5 dans Y dans la séquence1
- -20 dans X et 10 dans Y dans la séquence 2
- 30 dans X et 100 dans Y dans la séquence 3.

Donnez dans chaque cas les valeurs de x, y et temp après l'exécution de chaque action.

Ecrire l'algorithme complet qui lit deux variables A, B les permute, et les affiche.

Exercice3

Montrez l'ordre de priorité des opérateurs dans l'évaluation des expressions arithmétiques algorithmiques suivantes: (les variables utilisées sont des variables entières)

1°) a+b*c 2°) - a + b 3°) –a/-b+c 4°) –a/-(b+c) 5°) –b/2*a

6°) 4/3*∏*R**3 7°) –x**3/y 8°) a* y div 2 9°) y div 2 + a mod y

***Exercice* 4**

Soient A, B des variables logiques, x, y, z, t des variables numériques. Ecrire la négation de chacune des expressions logiques suivantes:

1°) A 2°) A ET B 3°) A OU B 4°) y>t

5°) x≤z 6°) (x<y) ET (z ≥ t) 5°) (y>t) OU (x<z)

Traduire en VBA

Chapitre 4. Les instructions conditionnelles

Le langage algorithmique offre deux façons de réaliser les tests: l'instruction SI, et l'instruction Selon Cas de branchement multiple. Le choix d'une syntaxe dépend du problème à résoudre.

4.1 L'instruction SI

Deux syntaxes sont possibles: Si…Alors…Sinon…Finsi et Si…Alors… Finsi

4.1.1 Si...Alors...Sinon...Finsi

	Algorithmique	VBA
Syntaxe	**Si** condition **Alors** Instructions1; **Sinon** Instructions2; **Finsi ;**	**if** condition **then** ensemble d'instructions à exécuter si la condition est vraie **Else** ensemble d'instructions à exécuter si la condition est fausse **endif**
Exemple	*Si totalMarchandise ≥ 100000* *Alors TauxRemise ← 0.02;* *Sinion TauxRemise ←0.01;* *Finsi;*	*If totalMarchandise>=100000 then* *TauxRemise =0.02* *Else* *TauxRemise =0.01* **endif**

Où condition représente une expression logique qui peut avoir la valeur VRAI ou FAUX et instructions1 et instructions2 sont deux ensembles d'actions à exécuter.

Lors de l'exécution, la condition est calculée. Si elle vaut VRAI alors exécution de **Instructions1**. Si la condition vaut FAUX, alors exécution de **Instructions2**. Ensuite, dans les deux cas, poursuite de l'exécution après **finsi**.

4.1.2 Si...Alors...Finsi

	Algorithmique	VBA
Syntaxe	**Si** condition **Alors** Instructions; **Finsi;**	**if** condition **then** Instructions; **endif**

Exemple	*Si age <18 alors* *# Mineur #* *Pension ←salaire/2;* *Finsi*	*If age<18 then* *' **Mineur*** *Pension =salaire/2* *Endif*

Dans ce cas, si la condition est VRAI alors exécution de l'ensemble d'actions **Instructions** et poursuite de l'exécution après **finsi**. Si la condition est fausse (FAUX), alors sortie et poursuite de l'exécution après **finsi.**

Exemple

Ecrire l'algorithme qui lit les trois nombres A, et B et C cherche le plus petit et l'affiche sur l'écran.

Principe de l'algorithme: Comme première étape, il s'agit de comparer deux nombres parmi les 3 entre eux; A et B par exemple, et stocker le plus petit dans la case min (on utilise la première forme du Si). Comme deuxième étape, on compare le troisième nombre (C) à min (on utilise la deuxième forme du Si).

Algorithme	Programme VBA
Algorithme minimum ; *Variables* *A, B, C, min : entier ;* *Début* *Lire(A, B, C) ;* *Si A < B* *Alors min ← A* *Sinon min ← B ;* *Finsi ;* *Si c < min* *Alors min ← c ;* *Finsi ;* *Ecrire(min) ;* *Fin algorithme.*	*Sub minimum()* *Dim A as integer, B as integer, C as integer , min as integer* *A= inputbox("donnez A :")* *B= inputbox("donnez B :")* *C= inputbox("donnez C :")* *If A<B then* *Min=A* *Else* *min=B* *Endif* *If C < min then* *min=C* *endif* *msgbox "min= " & min* *endsub*

4.2 L'instruction de branchement multiple (Selon…cas)

Deux formes existent:

4.2.1 1ère forme

	Algorithmique	VBA
Syntaxe	**Selon cas** *variable*1 **Cas** $valeur_1$ $Bloc\text{-}instructions_1$; … **Cas** $valeur_n$ $Bloc\text{-}instructions_n$; **Cas autre** $Bloc\text{-}Instruction_{n+1}$; **Finselon**;	**Select case** *variable*1 **Case** $valeur_1$ $Bloc\text{-}instructions_1$ … **Case** $valeur_n$ $Bloc\text{-}instructions_n$ **Case else** $Bloc\text{-}Instruction_{n+1}$ **End select**
Exemple: Ecriture du nom du mois en connaissant son numéro	*Selon cas mois* *Cas 1* *Ecrire("Janvier");* *Cas 2* *Ecrire ("Février");* *Cas 3* *Ecrire("Mars");* *Cas 4* *Ecrire("Avril");* *Cas 5* *Ecrire ("Mai");* *Cas 6* *Ecrire("Juin");* *Cas 7* *Ecrire("Juillet");* *Cas 8* *Ecrire ("Août");* *Cas 9* *Ecrire("Septembre");* *Cas 10* *Ecrire("Octobre");* *Cas 11* *Ecrire ("Novembre");* *Cas 12* *Ecrire("Décembre");* *Cas autre* *Ecrire ("Erreur");* ***Finselon;***	*Select case mois* *Case 1* *Msgbox "Janvier"* *Case 2* *Msgbox "Février"* *Case 3* *Msgbox "Mars"* *Case 4* *Msgbox "Avril"* *Case 5* *Msgbox "Mai"* *Case 6* *Msgbox "Juin"* *Case 7* *Msgbox "Juillet"* *Case 8* *Msgbox "Août"* *Case 9* *Msgbox "Septembre"* *Case 10* *Msgbox "Octobre"* *Case 11* *Msgbox "Novembre"* *Case 12* *Msgbox "Décembre"* *Case else* *Msgbox "Erreur"* ***End select***

Ou,

- *variable* est une variable de n'importe quel type (entier, réel, caractère, chaîne de caractères, booléen)
- *Valeur1,…, valeur*$_n$ sont des valeurs compatibles avec *variable.*

- *Bloc-Instruction*$_1$,…, *Bloc-Instruction*$_n$, *Bloc-Instruction*$_{n+1}$ sont des instructions qui peuvent inclure toutes les instructions exécutables.
- La partie **Cas autre** est facultative.

A l'exécution de cette instruction, la variable **variable1** est comparée à *valeur*$_1$,…., *valeur*$_n$.

Si **variable1** est égale à une *valeur*$_i$, alors exécution du bloc d'instructions orrespondant.

Si variable1 n'est égale à aucune des valeurs : *valeur*$_1$, …., *valeur*$_n$, alors exécution du bloc d'instructions après **cas autre.**

Et dans tous les cas, poursuite de l'exécution après **Fin selon.**

4.2.2 2ème forme

	Algorithmique	**VBA**
Syntaxe	**Selon cas** *variable*1 **Cas est** *op. comparaison valeur* *Bloc-instructions*1 **Cas** *valeur_début* **à** *valeur_fin* *Bloc-instructions*2 **Cas autre** *Bloc-Instructions*3 **Finselon**;	**Select case** *variable*1 **Case Is** *op. comparaison valeur* *Bloc-instructions1* **Case** *valeur_début* **to** *valeur_fin* *Bloc-instructions*2 **Case else** *Bloc-Instructions3* **End select**
Exemple	***Selon cas*** *performance* ***Cas*** *1* *Bonus =salaire * 0.1* *Cas 2, 3* *Bonus = salaire * 0.09* ***Cas*** *4* ***à*** *6* *salaire * 0.07* *Cas est > 8* *Bonus = 100* ***Cas autre*** *Bonus=0* ***Finselon****;*	*Select Case performance* *Case 1* *Bonus = salaire * 0.1* *Case 2, 3* *Bonus = salaire * 0.09* *Case 4 To 6* *Bonus = salaire * 0.07* *Case Is > 8* *Bonus = 100* *Case Else* *Bonus = 0* *End Select* *End Function*

OU

- *op. comparaison valeur* est une expression de comparaison de *variable1* avec *valeur*.
- *valeur_début à valeur_fin* représente un intervalle de valeurs.

Cette forme permet d'exécuter un bloc d'instructions si la variable est comprise dans une plage de valeurs.

A l'exécution de cette instruction, nous avons trois cas:

Cas1: Si le test ***variable1 op.comparaison valeur*** est vrai alors exécution du bloc d'instructions1.

Cas2: Si ***variable1*** est comprise dans l'intervalle ***valeur_début.. valeur_fin***, alors exécution du bloc d'instructions2.

Cas3: Si **Cas autre** existe, alors exécution du bloc d'instruction3.

Et dans tous les cas, poursuite de l'exécution après **Fin selon**.

4.3 Exercices

***Exercice* 1**

```
Soit la portion d'algorithme suivante:
Lire (T);
Si  T>0  alors
            Y←1;
            X←3;
            Z←5;
           X←Y-2;
           Z←X+Y;
          Y←X+Z;
       sinon
        Si T=0  alors
                 Y←2;
                Z←5;
                X←Y;
                Y←Z;
                Z←X;
             sinon
                X←20;
                Y←2;
                Z←9;
                X←Y;
               Y←X;
       Finsi
Finsi
Ecrire(X,Y,Z);
```

Donnez les valeurs de X, Y et Z qui sont affichées sur l'écran en exécutant l'instruction Ecrire dans les cas suivant de la valeur de T lue dans l'instruction Lire.

1) T=20
2) T=0
3) T=-1

Exercice2

1) Ecrire l'algorithme qui lit les variables entières a1, b1, c1, a2, b2, c2 et résout le système d'équations:

$$\begin{cases} A1\,x + b1\,y = c1 \\ A2\,x + b2\,y = c2 \end{cases}$$

Et affiche les solutions *x* et *y* quand elles existent; sinon il nous informe qu'il n' y a pas de solutions.

2) Traduire cet algorithme en VBA.
3) Appliquez sur machine pour obtenir les résultats.

***Exercice* 3**

Quelles sont les valeurs de A et B affichées sur l'écran lors de l'exécution des portions de programmes VBA ci-dessous dans les 3 cas suivants:

1) A= 15 B= 2 2) A= 10 B=32 3) A=5 B=5

3

```
Sub exo33()
Dim A as integer, B as integer
A = InputBox("A=")
B = InputBox("B=")
If A - B >= 0 Then
   A = A - A * B
End If
B = A \ 2 + 56
MsgBox  "A=" & A & "B=" & B
End Sub
```

2

```
Sub exo32()
Dim A as integer, B as integer
A = InputBox("A=")
B = InputBox("B=")
If A - B > 0 Then
    A = A + B + 41
    B = A \ 2
 Else
    A = B Mod A + 20
 End If
B = 3 * A
MsgBox "A=" & A & "B=" & B
End Sub
```

1

```
Sub exo31()
Dim A as integer, B as integer
A = InputBox("A=")
 B = InputBox("B=")
 If A - B > 0 Then
      A = A + B
      B = A \ 2
 Else
      A = B Mod A
      B = 3 * A
 End If
 MsgBox "A=" & A & "B=" & B
End Sub
```

Exercice 4

Ecrire l'algorithme qui lit les variable entières a, b, c et qui résout l'équation: a x^2 + b x + c =0 (dans R), et affiche les résultats: les valeurs des racines *x1* et *x2* si elles existent, sinon il nous informe qu'il n'y a pas de solutions.

Exercice 5

Ecrire l'algorithme qui lit pour un individu (femme ou homme), le sexe, l'âge et la taille, et indique si cet individu est majeur (âge ≥ 18) ou mineur, et s'il est grand ou petit. Une femme est supposée grande si sa taille est supérieure à 1,70m et petite si elle est inférieure à 1,50m. Un homme est dit grand si sa taille est supérieure à 1,80m et petit si elle est inférieure à 1,60m.

Exercice 6

Ecrire l'algorithme qui lit pour un individu la taille TAILLE en mètres et le poids POIDS en kilogrammes, et calcule l'indice de masse corporelle IMC : IMC=$\frac{POIDS}{TAILLE^2}$ et indique l'état d'obésité de cet individu, sachant que :

IMC <17	➔	Gravement en sous-poids
Entre 17 et 18,49	➔	Sous-Poids
Entre 18,5 et 24,99	➔	Le poids idéal
Entre 25 et 29,99	➔	Surpoids

Entre 30 et 34,99	➔	Obese
Entre 35 et 39,99	➔	Gravement obese
Plus de 40	➔	Extremmement obese

(**Nb. Faire la traduction en VBA de tous les algorithmes des exercices précédents.**)

Chapitre 5. Les instructions répétitives

5.1 Exemple introductif

Ecrire un algorithme qui lit N nombres entiers et qui calcule leurs somme.

Principe de la solution

On suppose initialement que la somme est égale à zéro. A chaque étape on lit un nombre et l'ajoute à la somme, jusqu'à avoir terminé tous les nombres. Somme contiendra à la fin la somme des nombre. En termes de programmation, cela permet de construire une boucle.

Les variables utilisées

Variable	*Rôle*	*Type*
N	*Contient le nombre de nombres à sommer. Initialement Somme vaut 0.*	*Entier*
Somme	*Contient la somme des nombres*	*Entier*
A	*La variable dans laquelle on lit le nombre à ajouter à chaque étape.*	*Entier*
I	*Compteur des nombres. Initialement I vaut 1*	*Entier*

L'algorithme

```
Algorithme somme_NB;

Variables
     N, A, I, Somme     :  entier;
Début
   Lire(N);
   Somme←0;
   I ← 1;
1: Lire (A);
   Somme ← Somme + A;
   I ←  I+1;
   Si  I ≤ N
      Alors  Aller à  1;
   Finsi
   Ecrire (Somme);
FinAlgorithme.
```

Commentaires

1- L'instruction "aller à" permet de mettre en œuvre une boucle qui s'arrêtera de s'exécuter lorsque la valeur de I devient supérieure à N.
2- La boucle s'exécute: à partir de I=1, et tant que i<=N.
3- La valeur de I atteint la valeur N grâce à l'instruction d'incrémentation (I ← I+1) exécutée à chaque étape de la boucle. Cette instruction met en évidence le pas de la boucle qui vaut 1 dans cet exemple.
4- Il est clair que la variable I, qui est sémantiquement le compteur des nombres à sommer, constitue sur le plan technique une variable de contrôle pour la boucle.
5- Pour écrire correctement une boucle, il faut extraire des variables utilisées pour la solution du problème, une ou plusieurs variables qui vont jouer le rôle de variables de contrôle.
6- Dans l'écriture de la boucle il faut définir pour la variable de contrôle les 3 éléments suivant:
   a) La valeur initiale
   b) Une condition d'arrêt d'exécution
   c) Une instruction de modification (incrémentation par exemple) de la variable qui doit s'exécuter à chaque étape de la boucle.

En s'assurant que a), b) et c) sont bien définies, on garantit la bonne exécution de la boucle.

Remarque

L'instruction aller à 1 est une rupture de séquence en programmation, et peut constituer une source d'erreur. Nous l'avons utilisée dans un objectif pédagogique, mais son utilisation a été déconseillée. Elle a été remplacée par les instructions: tant que, pour et répéter.

5.2 Les instructions traduisant les boucles

5.2.1 L'instruction tant que

	Algorithmique	**VBA**
Syntaxe	***Tant que*** *condition* ***faire*** *Bloc-instructions;* ***Fintantque****;*	***while*** *condition* *Bloc-instructions* ***wend***
Exemple	*Algorithme somme_NB;* *# l'algorithme qui lit N nombre et fait leur somme. #*	***Sub*** *Somme_NB ()* *Dim N As Integer, A As Integer* *Dim i As Integer, Somme As*

	Variables *N, A, I, Somme : entier;* *Début* *Lire(N);* *Somme←0;* *I ← 1;* *Tant que i<=N* *faire* *Lire (A);* *Somme ← Somme + A;* *I ← I+1;* *FinTantque* *Ecrire (Somme);* *FinAlgorithme.*	*Integer* *N = InputBox("Donnez N: ")* *Somme = 0* *i=1* ***While** I <=N* *A = InputBox("Donnez le nombre suivant : ")* *Somme = Somme + A* *i=i+1* ***Wend*** *MsgBox "Somme : " & Somme* ***End Sub***

OU

Condition est une expression logique (test)

5.2.2 L'instruction Pour

	Algorithmique	**VBA**
Syntaxe	***Pour** variable ← expr1 **à** expr2 **pas** expr3* ***Faire** Bloc-instructions;* ***FinPour;***	***For** variable = expr1 **to** expr2 **step** expr3* *Bloc-instructions;* ***Next variable***
Exemple	*Algorithme somme_NB;* *Variables* *N, A, I, Somme : entier;* *Début* *Lire(N);* *Somme←0;* *Pour I ← 1 à N* *faire* *Lire (A);* *Somme ← Somme + A;*	***Sub** Somme_NB ()* *Dim N As Integer, A As Integer* *Dim i As Integer, Somme As Integer* *N = InputBox("Donnez N: ")* *Somme = 0* ***for** i=1 **to** N **step** 1* *A = InputBox("Donnez le nombre suivant: ")* *Somme = Somme + A* ***Next** i* *MsgBox "Somme : " & Somme*

	FinPour *Ecrire (Somme);* *FinAlgorithme.*	***End Sub***

Ou

- *Variable est une variable de type ordinal (entier ou caractère par exemple)*
- *Expr1, expr2, expr3 sont des expressions compatibles avec variable.*
- *Expr1 doit être inférieure à expr2 pour que la boucle soit exécutée.*
- *Lorsque le pas de la boucle est égal à 1 "pas 1" est omis.*
- *Bloc-instructions un groupe d'instruction.*

5.2.3 L'instruction Répéter

	Algorithmique	**VBA**
Syntaxe	**Répéter** *Bloc-Instructions;* **Jusqu'à** *condition;*	**Do** *Bloc-Instructions* **Loop until** *condition*
Exemple	*Algorithme somme_NB;* *Variables* *N, A, I, Somme : entier;* *Début* *Lire(N);* *Somme←0;* *I ← 1;* *Répéter* *Lire (A);* *Somme ← Somme + A;* *I ← I+1;* *Jusqu'à I>N;* *Ecrire (Somme);* *FinAlgorithme.*	***Sub*** *Somme_NB ()* *Dim N As Integer, A As Integer* *Dim i As Integer, Somme As Integer* *N = InputBox("Donnez N: ")* *Somme = 0* *i=1* ***Do*** *A = InputBox("Donnez le nombre suivant : ")* *Somme = Somme + A* *i=i+1* ***Loop until I >N*** *MsgBox "Somme : " & Somme* ***End Sub***

5.3 Les Tableaux

Certains problèmes nécessitent de stocker en mémoire un grand nombre de données de même type et de les traiter de la même manière. Dans ce cas on utilise la notion de tableaux.

Il existe, principalement, les tableaux à une dimension qu'on appelle aussi **vecteurs**, et les tableaux à 2 dimensions qu'on appelle aussi **matrices**.

5.3.1 Les tableaux à une dimension ou vecteurs

Un vecteur est un ensemble de cases mémoires contigües de même type. Il est complètement défini par son nom, son type et sa dimension qui est le nombre de cases qui le

composent. La figure 3 schématise un tableau V de dimension N qu'on note V(N). V[i] fait référence au i^{ème} élément du vecteur.

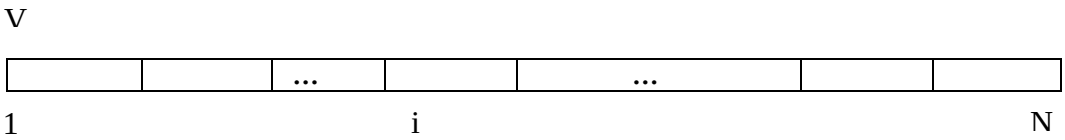


Figure 3: Tableau V de dimension N

5.3.2 Les tableaux à deux dimensions ou matrices

Un tableau à deux dimensions ou matrice est un ensemble de vecteurs lignes ou colonnes contigus. Il a par conséquent une dimension dans les lignes et une dimension dans les colonnes. Il est complètement défini par son nom, son type et son nombre de lignes et de colonnes. Notons la matrice Mat composée de N lignes et M colonnes Mat(NxM). MAT[i,j] fait référence à l'élément de l'intersection de la ligne i et de la colonne j.

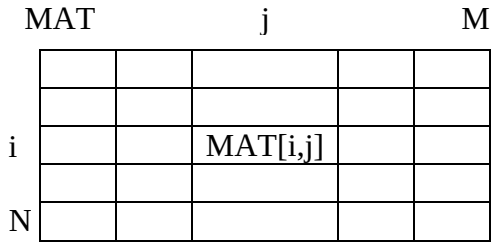


Figure 4: Matrice MAT(NxM)

5.4 La Déclaration des tableaux

5.5 Les vecteurs

	Algorithmique	**VBA**
Syntaxe	*nomduvecteur* **[***1***..***dimension***] : Tableau de** *Type*;	***Dim*** *nomduvecteur****(****1* ***to*** *dimension****)*** ***as*** *type*
Exemple *Tableau V de dimension N=10. de type entier.*	***Constantes*** N← 10; ***Variables*** *V[1..N] : Tableau de entier;*	***Const N=10*** ***Dim*** *nomduvecteur****(****1* ***to*** *N****)*** ***as*** *integer*

Ou

- dimension représente la dimension du vecteur
- type peut être l'un des types standard (entier, réel, etc…)

5.5.1 Les matrices

	Algorithmique	VBA
Syntaxe	*nomdematrice* **[***1..diml,1..dimc***] :** **Tableau de** *Type*;	***Dim*** *nomdematrice**(1* ***to*** *diml, 1* ***to*** *dimc**)*** ***as*** *type*
Exemple *La matrice Mat(NXM) (N=10 et M=20). Tous les éléments de type entier.*	*Constantes* *N ← 10;* *M ← 20;* *Variables* *Mat[1..N, 1..M] : Tableau de entier;*	***Const*** *N=10, M=20* ***Dim*** *Mat**(1*** ***to*** *N**, 1*** ***to*** *M**)*** ***as*** *integer*

Nb. Au moment de la déclaration des tableaux, leurs dimensions doivent être connus (ici dimension, diml et dimc). Il faut par exemple les définir dans la partie **constantes**, ou bien utiliser des valeurs constantes dans la déclaration. Cependant, VBA permet de déclarer des tableaux dynamiques de la manière suivante:

```
Dim V( ) as integer
```

Cette formule permet de déclarer un tableau V de dimension inconnue. Au moment de l'utilisation on peut fixer la dimension avec *Redim*.

Exemple

```
Sub test()
Dim v() as integer, i as byte
Dimension = inputbox("donnez la dimension")
Redim v(dimension)
   ...
End sub
```

Exemples

Algorithme	**Programme VBA**
Algorithme LectEcrVect; *# Lecture et écriture d'un vecteur #*	*Sub lectEcrVect()* *' Lecture et écriture d'un vecteur*

Constantes *N ←10;* *Variables* *V[1..N] : Tableau de entier;* *Début* *# La lecture #* *Pour i←1 à N* *Faire lire (V[i]);* *Finpour;* *# L'écriture #* *Pour i←1 à N* *Faire écrire("V[",i,"]=", V[i]);* *Finpour;* *FinAlgorithme.*	*Const n=10* *Dim i As Integer* *Dim v(1 To n) As Integer* *' Lecture du vecteur* *For i = 1 To n* *v(i) = InputBox("donnez V(" & i & ") --->")* *Next i* *' Ecriture du vecteur* *For i = 1 To n* *Msgbox " V(" & i & ") =" & v(i)* *Next i* *End Sub*

2°) L'algorithme qui lit et écrit la matrice réelle Mat composée de N lignes et M colonnes avec N=10 et M=20 (traitement Ligne par ligne).

Algorithme	Programme VBA
Algorithme LectEcrMat; *# Lecture et écriture d'une matrice #* *Constantes* *N ←5;* *M←4;* *Variables* *Mat[1..N,1..M] : Tableau de réel;* *i, j : entier;* *Début* *# La lecture #* *Pour i←1 à N faire* *Pour j←1 à M Faire* *lire (Mat[i,j]);* *Finpour;*	*Sub lectEcrmat()* *' Lecture et écriture d'une matrice* *Const n=5, m=4* *Dim mat(1 To n, 1 To m) As Integer* *Dim i as integer, j As Integer* *' lecture de la matrice* *For i = 1 To n* *For j = 1 To m* *mat(i, j) = InputBox("donnez mat(" & i & "," & j & ") =")* *Next j* *Next i*

Finpour;	*' Ecriture de la matrice*
# L'écriture #	
Pour i←1 à N faire	*For i = 1 To n*
Pour j←1 à M faire	*For j = 1 To m*
écrire("Mat[",i,",",j,"]=", Mat[i,j]);	*msgbox " mat(" & i & "," & j & ") =" & mat(i, j)*
Finpour;	*Next j*
Finpour;	*Next i*
FinAlgorithme.	*End Sub*

2°) Ecriture avec Tant que

Algorithme	Programme VBA
Algorithme LectEcrMat;	*Sub lectEcrmat()*
# Lecture et écriture d'une matrice #	*Const n=5, m=4*
Constantes	*Dim mat(1 To n, 1 To m) As Integer*
N ←5;	*Dim i, j As Integer*
M←4;	*' lecture de la matrice*
Variables	*I=1*
Mat[1..N,1..M] : Tableau de réel;	*While i <= n*
i, j : entier;	*J=1*
Début	*While j<= m*
# La lecture #	*mat(i, j) = InputBox("donnez mat(" & i & "," & j & ") =")*
i←1;	*j=j+1*
Tant que i≤ N faire	*Wend*
j←1;	*I=i+1*
Tant que j≤ M Faire	*Wend*
lire (Mat[i,j]);	*' Ecriture de la matrice*
j←j+1;	*I=1*
FinTant que;	*While i<= n*
i←i+1;	*J=1*
FinTantque;	*While j <= m*
# L'écriture #	*msgbox " mat(" & i & "," & j & ") =" & mat(i, j)*
i←1;	*j=j+1*
Tant que i≤ N faire	

j←1; *Tantque j≤ M Faire* *écrire("Mat[",i,",",j,"]=", Mat[i,j]);* *j←j+1;* *FinTant que;* *i←i+1;* *FinTantque;* *FinAlgorithme.*	*Wend* *I=i+1* *Wend* *End Sub*

3°) Ecriture avec Répéter

Algorithme	Programme VBA
Algorithme LectEcrMat; *# Lecture et écriture d'une matrice #* *Constantes* *N ←10;* *M←20;* *Variables* *Mat[1..N,1..M] : Tableau de réel;* *i, j : entier;* *Début* *# La lecture #* *i←1;* *Répéter* *j←1;* *Répéter* *lire (Mat[i,j]);* *j←j+1;* *Jusqu'à j>M;* *i←i+1;* *Jusqu'à i>N;* *# L'écriture #* *i←1;* *Répéter*	*Sub lectEcrmat()* *Const n = 2, m = 3* *Dim mat(1 To n, 1 To m) As Integer* *Dim i, j As Integer* *' lecture de la matrice* *i = 1* *Do* *j = 1* *Do* *mat(i, j) = InputBox("donnez mat(" & i & "," & j & ") =")* *j = j + 1* *Loop Until j > m* *i = i + 1* *Loop Until i > n* *' Ecriture de la matrice* *i = 1* *Do*

j←1; *Répéter* *écrire("Mat[",i,",",,j,"]=", Mat[i,j]);* *j←j+1;* *Jusqu'à j>M;* *i←i+1;* *Jusqu'à i>N;* *FinAlgorithme.*	*j = 1* *Do* *MsgBox " mat(" & i & "," & j & ") =" & mat(i, j)* *j = j + 1* *Loop Until j > m* *i = i + 1* *Loop Until i > n* *End Sub*

Remarque:

En ce qui concerne le traitement des matrices colonne par colonne, les algorithmes sont exactement pareil, sauf que la boucle extérieure est sur les colonnes (ou j) et la boucle intérieure se fait sur les lignes(ou i).

5.6 Syntaxes de boucles propres à VBA

Il existe en VBA d'autres structures répétitives équivalentes à celles données précédemment comme Do{while/until} loop, et for each next. Le tableau suivant récapitule les différentes syntaxes.

Exemples

	Syntaxe	*Exemple*	*Interprétation*
1	*Do while condition* *Instructions* *Loop*	*i=1* *Do while i<=10* *Msgbox "I like Excel"* *i=i+1* *loop*	*À partir de i =1 et tant que i<=10 il écrit l'expression "I like Excel".* *Equivalente à l'instruction while.*
2	*Do until condition* *Instructions* *Loop*	*i=1* *Do until I >10* *Msgbox "I like Excel"* *i=i+1* *loop*	*À partir de i =1 et jusqu'à ce que i devienne i>10 il écrit l'expression "I like Excel".* *Equivalente à l'instruction while.*
3	*Do* *Instructions* *Loop while condition*	*i=1* *Do* *Msgbox "I like Excel"* *i=i+1*	*Equivalente à la syntaxe 1 sauf que la condition d'exécution est testée à la fin de la boucle.*

		loop while I <=10	
4	*Do* *Instructions* *Loop until condition*	*i=1* *Do* *Msgbox "I like Excel"* *i=i+1* *loop until I >10*	*Equivalente à la syntaxe 1 sauf que la condition d'arrêt est testée à la fin de la boucle.*
5	*For each …* *Instructions* *Next*	*Dim tableau(2) as string* *Tableau(0)="A"* *Tableau(1)="B"* *Tableau(2)="C"* *For Each valeur In tableau* *Msgbox valeur* *Next*	*Syntaxe très pratique et très utilisée dans la couche objet spécifique à la manipulation d'Excel à partir de VBA.*

NB. La différence entre les syntaxes 1, 2 et les syntaxes 3, 4 dans le tableau précédent est que dans les syntaxes 1, 2 la condition (d'exécution ou d'arrêt) est testée au début de la boucle, et par conséquent la boucle peut ne pas s'exécuter si la valeur initiale de la variable de contrôle ne vérifie pas la condition d'exécution. Par contre, dans les syntaxes 3, 4 la condition d'arrêt ou d'exécution est testée à la fin de la boucle qui s'exécute au moins une fois même si la valeur initiale de la variable de contrôle ne vérifie pas la condition d'exécution.

5.7 Exercices

Exercice1

Ecrire l'algorithme qui lit les âges des élèves d'une classe composée de N élèves; et calcule leurs somme et moyenne et les affiche. Ecrire la boucle de trois manières différentes:

1°) Tant que 2°) Pour 3°) Répéter

Dérouler la boucle pour le nombre d'élèves N=5, et les âges: 10, 11, 14, 13, 12.

Exercice2

1) Quel est le résultat d'exécution des portions d'algorithmes suivantes **:**

3

```
lire(a,b);
 tant que  a-b > 0
    faire
        Ecrire ('bon
courage');
        lire(a,b);
Fintantque;
```

2

```
i ← -1 ;
Tant que  i < 10
     faire
         Ecrire('bon
courage' );
Fintantque;
 i←i+2;
```

1

```
i ← -1 ;
Tant que  i <  10
       faire
          ecrire('bon
courage');
           i ←i+1;
Fintantque;
```

2) Quelle est la/les variables de contrôle de chaque boucle.

3) Ecrivez chaque boucle en utilisant Pour, répéter.

Exercice3

Ecrire l'algorithme qui lit le nombre entier N (N≥0) et qui calcule son factoriel fact sachant que fact est défini comme suit:

$$Fact = \begin{cases} 1*2*\ldots*N & \text{si } N>0 \\ 1 & \text{si } N=0 \end{cases}$$

Exercice4

Ecrire l'algorithme qui lit deux matrices: A(3X4) et B(3X4) et qui fait leurs somme. Faire le traitement ligne par ligne. Traduire en VBA.

Chapitre 6. Les Sous-Programmes

6.1 Introduction

Certains problèmes conduisent à des programmes longs, difficiles à écrire et à comprendre. On les découpe en plusieurs parties appelées sous-programmes. Les sous programmes sont des modules (groupe d'instructions) indépendants désignés par un nom. Ils ont plusieurs intérêts :

- Permettent de "factoriser" les programmes en mettant en commun les parties qui se répètent.
- Permettent une structuration et une meilleure lisibilité des programmes.
- Facilitent la maintenance du code (il suffit de modifier une seule fois).
-Ces procédures et fonctions peuvent éventuellement être réutilisées dans d'autres programmes.

Il existe deux types de sous programmes : les procédures et les fonctions. Tous les langages de programmation permettent de définir ces deux types.

6.2 Les Sous-programmes Fonction et les Sous- programmes Procédures

6.2.1 Les fonctions

Le rôle d'une fonction en programmation est similaire à celui d'une fonction en mathématique : elle retourne un résultat à partir des valeurs des paramètres d'entrée, ce résultat est mis dans le nom de la fonction.

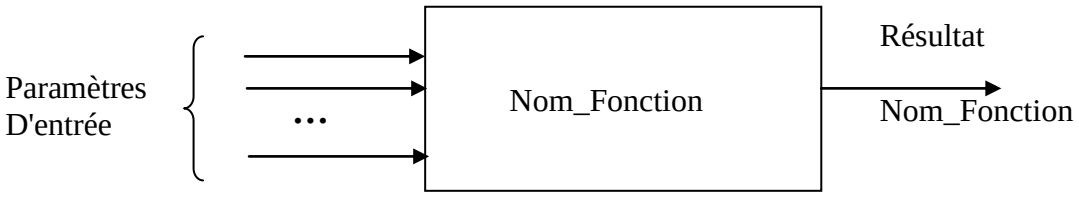


Figure 5 : Schématisation d'une fonction

L'écriture d'une fonction obéit à la syntaxe suivante.

	Algorithmique	**VBA**
Syntaxe	*Fonction nomfonction (paramètres et leurs types) : type_fonction ;* *Début*	*Function nomfonction(paramètres et leurs types):type_fonction* *' Instructions constituant le corps de la fonction*

	Instructions constituant le corps de la fonction *retourne ...* *FinFonction*	*nomfonction=….* ***End function***
Exemple	*Fonction SommeCarre (x : réel, y: réel) : réel* *Variables* *z : réel;* *Début* *z ←x^2+y^2;* *retourne (z); # ou bien* *SommeCarre← z #* *FinFonction*	*Function SommeCarre (x as dooble, y as double) : double* *Dim z as double* *z ←x^2+y^2;* *SommeCarre=z* *End Function*

- Pour le choix d'un nom de fonction il faut respecter les mêmes règles que celles pour les noms de variables
- type_fonction est le type du résultat retourné
- L'instruction retourne sert à retourner la valeur du résultat

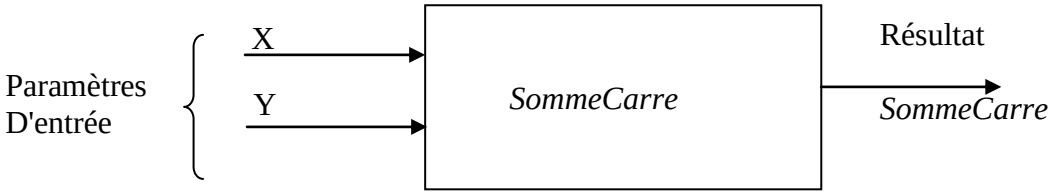


Figure 6: *Schéma de la fonction SommeCarre*

6.2.2 Les Procédures

Une procédure est un sous-programme auquel on donne un ou plusieurs paramètres en entrée et qui fait des calculs et donne un ou plusieurs résultats en sortie.

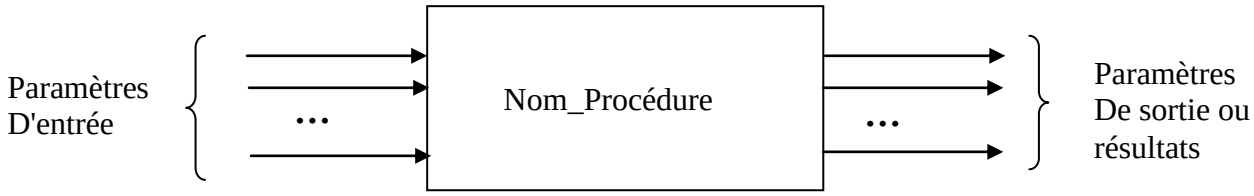


Figure 7: Schématisation d'une procédure

L'écriture des procédures doit obéir à la syntaxe suivante.

	Algorithmique	**VBA**
Syntaxe	*Procedure nom_Procédure (paramètres et leurs types) ;* *Début*	*Sub nom_Procédure (paramètres et leurs types)* *' Instructions constituant le corps de la*

	# Instructions constituant le corps de la procédure# *FinProcedure*	*procédure* *End Sub*
Exemple	*Procédure SommeCarre (x : réel, y: réel, z:réel);* *Début* *z ←x^2+y^2;* *FinProcedure*	*Sub SommeCarre (x as double, y as double, z as double)* *z ←x^2+y^2* *End Sub*

Figure 8 : Schématisation de la procédure *SommeCarre*

6.2.3 Exemple: Calcul de C_n^p en utilisant un sous programme Factoriel qui calcule X!

Ecrire un programme qui lit les nombres naturels (entiers positifs) N et P (N>P), et calcule la valeur C_n^p donnée par la formule:

$$C_n^p = \frac{n!}{p!\,(n-p)!}$$

1- En utilisant une fonction 'factoriel' qui calcule X! (X entier positif).

2- Transformer le programme en utilisant une procédure factoriel au lieu d'une fonction.

Cas1: Utilisation d'une fonction

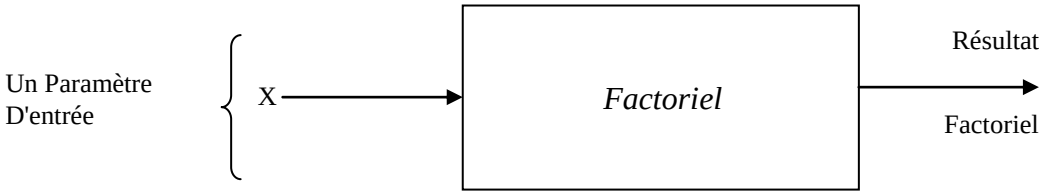


Figure 9: *Schématisation de la fonction Factoriel*

Algorithmique	VBA
	Function factoriel(x As Byte) As Long

Algorithme calcnp; *Variables* *n, p , factn, factp, factnp, I : entier;* *# Déclaration de la fonction factoriel #* ***Fonction** factoriel(x : entier) : entier;* *Variables* *I , y : entier;* *Début* *y←1;* *Pour i←1 à x* *Faire y←y*i;* *Finpour;* *Retourne (y)# Ou bien factoriel ← y #* ***finFonction*** *# Début du programme principal #* *Début* *Lire(n,p); # On suppose que n>p>0 #* *factn←factoriel(n);* *factp←factoriel(p);* *factnp←factoriel(n-p);* *cnp=factn/(factp*factnp);* *écrire("cnp=", cnp);* *FinAlgorithme*	*Dim i As Byte* *y = 1* *For i = 1 To x* *y = y * i* *Next i* *factoriel = y* *End Function* *Sub cnp()* *Dim n As Byte, p As Byte, c As Byte* *Dim factn As Long, factp As Long, factnp As Long* *n = InputBox("donnez n")* *p = InputBox("donnez p")* *factn = factoriel(n)* *factp = factoriel(p)* *factnp = factoriel(n - p)* *c = factn / (factp * factnp)* *MsgBox "Cnp= " & c* *End Sub*

Cas2: Utilisation d'une fonction

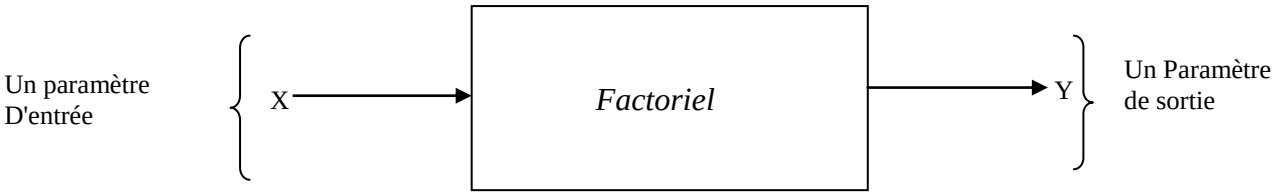


Figure 10 : Schématisation de la procédure *Factoriel*

Algorithmique	VBA
Algorithme calcnp; *Variables* *n, p , factn, factp, factnp, I : entier;* # *Déclaration de la procédure factoriel* # ***Procédure*** *factoriel(x : entier, y : entier);* *Variables* *I : entier;* *Début* *y←1;* *Pour i←1 à x* *Faire y←y*i;* *Finpour;* ***finProcédure*** ***Début*** *Lire(n,p); # On suppose que n>p>0 #* *factoriel(n, factn);* *factoriel(p, factp);* *factoriel(n-p, factnp);* *cnp=factn/(factp*factnp);* *écrire("cnp=", cnp);* ***FinAlgorithme***.	*Sub factoriel(x As Byte, y As Long)* *Dim i As byte* *y = 1* *For i = 1 To x* *y = y * i* *Next i* *End Sub* *Sub cnp()* *Dim n As Byte, p As Byte, c As Byte* *Dim factn As Long, factp As Long, factnp As Long* *n = InputBox("donnez n")* *p = InputBox("donnez B")* *Call factoriel(n, factn)* *Call factoriel(p, factp)* *Call factoriel(n - p, factnp)* *c = factn / (factp * factnp)* *MsgBox "Cnp= " & c* *End Sub*

Remarques et Commentaires

- Dans le langage algorithmique les procédures et fonctions sont placées les unes après les autres, après la déclaration des variables.
- En VBA les procédures et/ou les fonctions peuvent être placées avant ou après les macros qui les appellent.
- Dans le cas de la fonction l'appel peut se faire aussi de la manière suivante:

```
Sub cnp()
Dim n As Byte, p As Byte
n = InputBox("donnez n")
p = InputBox("donnez p")
MsgBox "Cnp= " & factoriel(n)/(factoriel(p)*factoriel(n - p))
End Sub
```

- Les noms des paramètres figurant entre parenthèses après le nom de la procédure ou de la fonction dans la partie déclaration, sont appelées des paramètres formels et ne sont pas des variables concrètes mais elles représentent des spécifications uniquement qui permettent d'illustrer le fonctionnement du sous programme.
- Les paramètres se trouvant entre parenthèses après le nom de la procédure ou de la fonction dans l'appel sont appelées des paramètres effectifs ou paramètres d'appel, c'est des variables concrètes qui doivent être déclarées ou connues par la macro appelante.
- Il existe une bijection entre les paramètres formels et les paramètres effectifs sur le plan: nombre, type et ordre d'apparition dans l'entête du sous programme.
- Les variables déclarées dans le sous-programme, sont des variables locales au sous-programme et ne peuvent pas être utilisées à l'extérieur de ce sous-programme.
- Tableau récapitulatif comparant les procédures et les fonctions en VBA

	Procédure	Fonction
Déclaration	*Sub nomProc(paramètres formels)* *…* *EndSub*	*Function nomFonc(paramètres formel) as type* *…* *nomFonc=résultat du calcul* *EndFonc*
Appel	*Call nomProc(paramètres d'appel)*	*Variable=nomFonc(paramètres d'appel)* *ou bien* *Msgbox nomfonc(paramètres d'appel)*

6.3 Les fonctions standards

Ce sont des fonctions qui sont définies dans le langage de programmation. Le programmeur peut appeler une fonction standard dans ses programmes sans qu'il soit obligé de la déclarer lui-même, en connaissant ses paramètres ainsi que le type du résultat qu'elle

retourne. Le VBA offre un grand nombre de fonctions standards, nous présentons dans le tableau ci-après quelques unes de ces fonctions.

Fonction	**Interprétation**
Sqr	*Renvoie une valeur de type Double indiquant la racine carrée d'un nombre*
Abs	*Renvoie une valeur de même type que celle transmise, indiquant la valeur absolue d'un nombre.*
Int	*Renvoie la partie entière d'un nombre.*
Round	*Renvoie un nombre arrondi à un nombre spécifié de positions décimales.*
Val	*Renvoie le nombre contenu dans une chaîne de caractère sous la forme d'une valeur numérique d'un type approprié.*

6.4 Exercices

Exercice1

On veut faire quelques mesures statistiques sur les âges d'un échantillon de N individus ayant participé à un sondage. On lit les âges dans un vecteur Age.

1°) Ecrire les sous programmes suivant, pour les éléments d'un vecteur V de dimension size:

a) Une procédure qui lit le vecteur.
b) Une procédure qui écrit le vecteur.
c) Une fonction qui calcule la moyenne.
d) Une procédure qui calcule la variance et l'écart-type.
e) Une procédure qui calcule le minimum.
f) Une fonction qui calcule le maximum.

2°) Ecrire un algorithme qui calcule la moyenne, la variance, l'écart-type, le minimum et le maximum des âges en utilisant les sous-programmes précédents.

Chapitre 7. Excel et VBA

7.1 Les fonctions personnalisées

7.1.1 Introduction

Excel offre un grand nombre de fonctions qui servent l'utilisateur pour résoudre ses problèmes. Mais en pratique, il existe plusieurs traitements qui sont à la fois très spécifiques et très utiles, et par conséquent, il n'existe pas de fonctions relatives à ces traitements. Excel offre à l'utilisateur la possibilité de créer, lui-même des fonctions personnalisées qui répondent à ses besoins bien spécifiques. Avant de présenter comment créer une fonction personnalisée à l'aide d'un exemple, nous commençons par montrer comment échanger les données entre la feuille Excel et un programme VBA.

7.2 Echange des données entre VBA et la feuille Excel

Dans tous les exemples que nous avons écrits jusqu'à présent, l'utilisateur échangeait les données avec le programme en utilisant les instructions inputbox() et msgbox() qui utilisent la notion de boîtes de dialogue. Parfois, l'utilisateur a besoin de traiter des données qui sont sur les cellules d'une feuille Excel, ou bien de récupérer ses résultats et les afficher sur une feuille Excel. VBA pour Excel offre les syntaxes appropriées pour lire/écrire depuis/sur une feuille Excel.

7.2.1 Lecture à partir de la feuille Excel

La lecture des données à partir de la feuille Excel nécessite que le programmeur place ses données dans les cellules appropriées avant l'exécution du programme sous VBE. Par exemple, si on veut lire trois valeurs A, B et X à partir de la feuille Excel, et qu'on stocke la valeur de A dans la cellule A1, et la valeur de B dans la cellule B1 et la valeur de X dans la cellule C1. Après avoir déclaré les variables A, B et X dans le programme, l'instruction A=[A1] permet de mettre le contenu de la cellule A1 dans la variable A, de même pour B=[B1] et X= [C1]. En définitive, la lecture de A, B et X à partir de la feuille Excel revient à écrire les 3 instructions:

A=[A1]

B=[B1]

X=[C1]

7.2.2 Ecriture des résultats d'un programme sur une feuille Excel

Dans le cas ou on veut faire l'affichage dans des cellules spécifiques de la feuille Excel. On écrit en VBA,

[A1]=X ➔ La valeur de X s'affiche dans la cellule A1

[A2]=a+b+1 ➔ La valeur de l'expression a + b + 1 s'affiche dans la cellule A2

[A3]="Informatique" ➔ Le mot 'Informatique' s'affiche dans la cellule A3

Dans ce cas une mise en forme des cellules A1, A2 et A3 est nécessaire pour correspondre aux valeurs qui y seront affichées.

7.3 Création d'une fonction personnalisée

Nous montrons dans ce qui suit, les différentes étapes de création d'une fonction personnalisée, sur la base d'un exemple simple**.**

7.3.1 Exemple: Calcul d'une remise en fonction de la quantité commandée

Un menuisier utilise Excel pour gérer les commandes des ses clients. La table suivante représente les quantités d'armoires commandées par ses clients. Le menuisier fait une remise de 20% si la quantité dépasse 200 unités, et 10% si elle dépasse 100 unités, et 0% sinon. Sachant que le prix unitaire est à chaque fois donné.

	A	B	C	D
1	Prix_Unitaire	20500		
2	Code_Client	Quantité	remise	Prix_Total
3	128105	450		
4	133976	200		
5	148878	150		
6	166844	300		
7	168307	175		
8	170829	190		
9	201041	230		

Figure 11: Tableau Excel calcul de la remise en fonction de la quantité commandée

Pour calculer il est possible d'utiliser une fonction personnalisée qu'on peut appeler REMISE et l'utiliser ensuite comme n'importe quelle autre fonction Excel.

Les fonctions personnalisées sont des sous-programmes VBA de type function. Le code suivant représente le code de la fonction REMISE qu'il faut saisir, sous l'éditeur VBE.

```
Function REMISE(quantite, prix)
' Calcul de la remise offerte par un menuisier
  If quantite >= 200 Then  ' vérifier si la quantité est>=200 unités
       REMISE = quantite * prix * 0.2    ' calcul de la remise à 20%
    Else
     If quantite >= 100 Then   'vérifier si la quantité est >=100 unités
```

```
        REMISE = quantite * prix * 0.1  ' calcul de la remise à 10%
      Else
        REMISE = 0   ' Pas de remise
    End If
  End If
End Function
```

7.3.1.1 Utilisation des fonctions personnalisées

Une fois la fonction REMISE saisie dans l'éditeur VBE, il est possible de revenir à Excel et taper dans la cellule C3 la formule

=REMISE (B3, B1)

Excel calcule la remise du premier client qui est de 20 % sur 450 unités à 20500 DA par unité et renvoie 7380000 DA.

Dans la première ligne du code VBA, *function REMISE(quantité, prix)*, il est indiqué que la fonction REMISE nécessite deux arguments : *quantité* et *prix.* En appelant la fonction dans une cellule de feuille de calcul, nous devons inclure ces deux arguments. Dans la formule =REMISE(B3,B1), B3 est l'argument quantité et B1 est *l'argument* prix. On peut par la suite généraliser le calcul en copiant la formule REMISE dans la plage G4:G9 pour obtenir les résultats ci-dessous.

	A	B	C	D
1	Prix_Unitaire	20500		
2	Code_Client	Quantité	remise	Prix_Total
3	128105	450	1845000	7380000
4	133976	200	820000	8405000
5	148878	90	0	9225000
6	166844	300	1230000	7995000
7	168307	50	0	9225000
8	170829	190	389500	8835500
9	201041	70	0	9225000

Figure 12: Tableau Excel calcul de la remise en fonction de la quantité commandée avec calculs

En lançant la fonction REMISE(), Excel recherche le nom *REMISE* dans le workbook en cours et trouve qu'il s'agit d'une fonction personnalisée dans un module VBA. Les noms

d'arguments entre parenthèses, quantité et *prix,* sont des spécifications pour les valeurs sur lesquelles repose le calcul de la remise.

L'instruction Si du bloc de code REMISE examine l'argument quantité et détermine si le nombre d'articles vendus est supérieur ou égal à 200; si oui, la REMISE est de quantité *prix*02, si ce n'est pas le cas, un autre Si teste si la quantité est supérieure ou égale à 100 : si oui , la REMISE est de quantité *prix*01; sinon la REMISE est égale à 0.

Le résultat est stocké en tant que variable *Remise.* Comme *Remise* est aussi le nom de la fonction, la valeur stockée dans la variable *Remise* est renvoyée à la formule de feuille de calcul appelée fonction REMISE.

7.3.1.2 Particularités des fonctions personnalisées

Une fonction personnalisée obéit à toutes les règles d'écriture des fonctions Excel. Elle doit commencer par le mot clé *Function* et se terminer par *End Function,* et peut être avec ou sans arguments. Les instructions VBA utilisées dans les fonctions personnalisées doivent être des instructions qui prennent des décisions et font des calculs à l'aide des arguments transmis à la fonction et rendre le résultat à une formule dans une feuille de calcul, ou à une expression utilisée dans une autre macro ou fonction VBA. Par exemple, les fonctions personnalisées ne peuvent pas redimensionner des fenêtres, modifier une formule dans une cellule, ni modifier les options de police, de couleur ou de motif pour le texte dans une cellule. Si on inclue un code d'action de ce type dans une fonction personnalisée, la fonction renvoie le message d'erreur #VALUE!.

Par ailleurs, on peut utiliser l' instruction **InputBox** dans une fonction personnalisée afin d'obtenir des données de l'utilisateur exécutant la fonction, ou l'instruction **MsgBox** pour communiquer des informations à l'utilisateur.

7.3.1.3 Rendre une fonction personnalisée accessible sous Excel

Pour rendre la fonction REMISE accessible à partir de tous les classeurs Excel, il faut la créer initialement dans un module (module1 par exemple) du **VbaProject Personal.xlsb**.

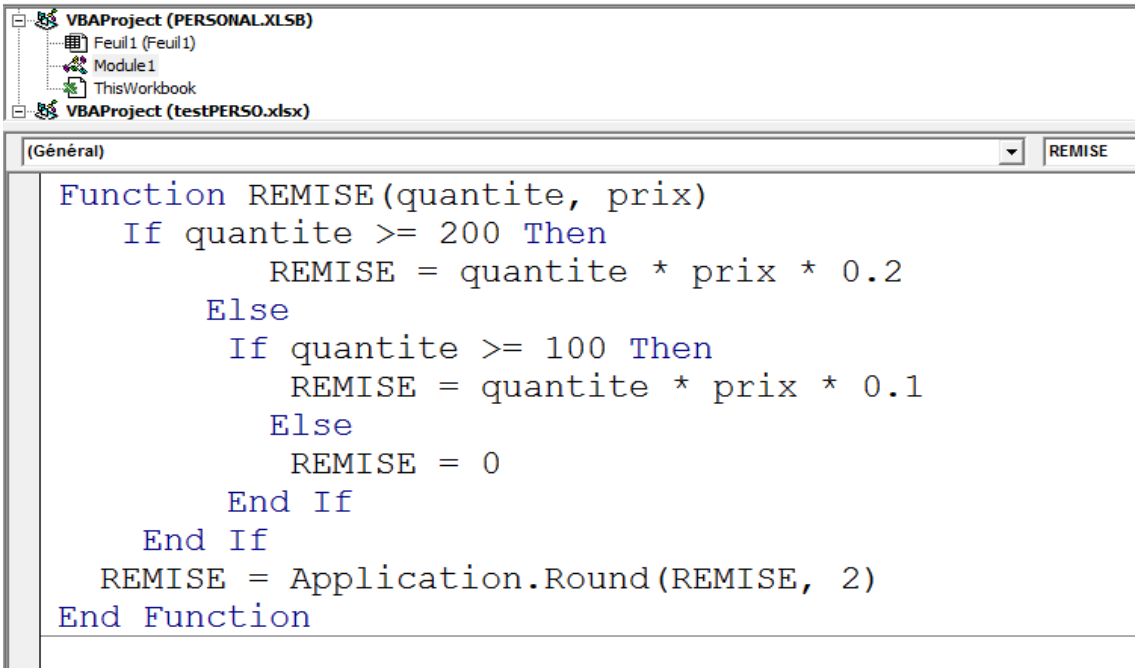


Figure 13: Code de la fonction Remise

Avec cela, la fonction REMISE est créée dans les fonctions personnalisées, mais elle est précédée de **personal.xlsb!.**

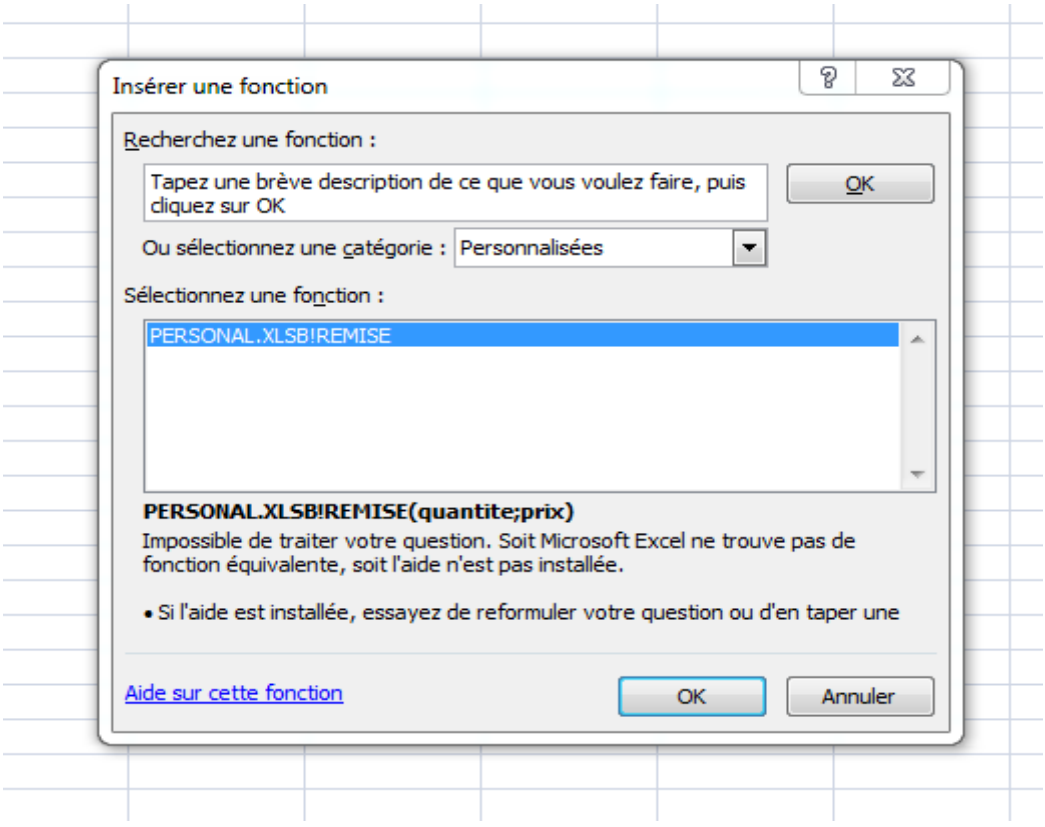


Figure 14: Le nom d'une fonction personnalisée avant de l'insérer dans les compléments Excel

Dans ce cas, à chaque fois qu'il faut utiliser la fonction sous Excel, il faut taper: =**personal.xlsb!REMISE()** et pas simplement **=REMISE().**

	A	B	C	D
1	Prix_Unitaire	20500		
2	Code_Client	Quantité	remise	Prix_Total
3	128105	=PERSONAL.XLSB!REMISE(B3;B1)		
4	133976	200		
5	148878	90		
6	166844	300		
7	168307	50		
8	170829	190		
9	201041	70		
10				

Figure 15: Il faut faire précéder Le nom de la fonction par personal.xlsb! lors de son appel sous Excel

Pour remédier à cela il faut enregistrer cette fonction en tant que complément Excel. Pour ce faire, il faut suivre les étapes suivantes:

a) Notre classeur Excel étant ouvert, on fait **Enregister sous**.

b) On donne un nouveau nom au fichier, par exemple **Mes_Fonctions_VBA**

c) On choisit le type **Complément Excel**

Cette opération doit être faite une seule fois, et le fichier Excel va être créé dans le répertoire Addins qui est proposé par windows.

d) Cliquer sur options Excel, ensuite Compléments et atteindre

e) Cocher Mes_Fonctions_Vba et OK.

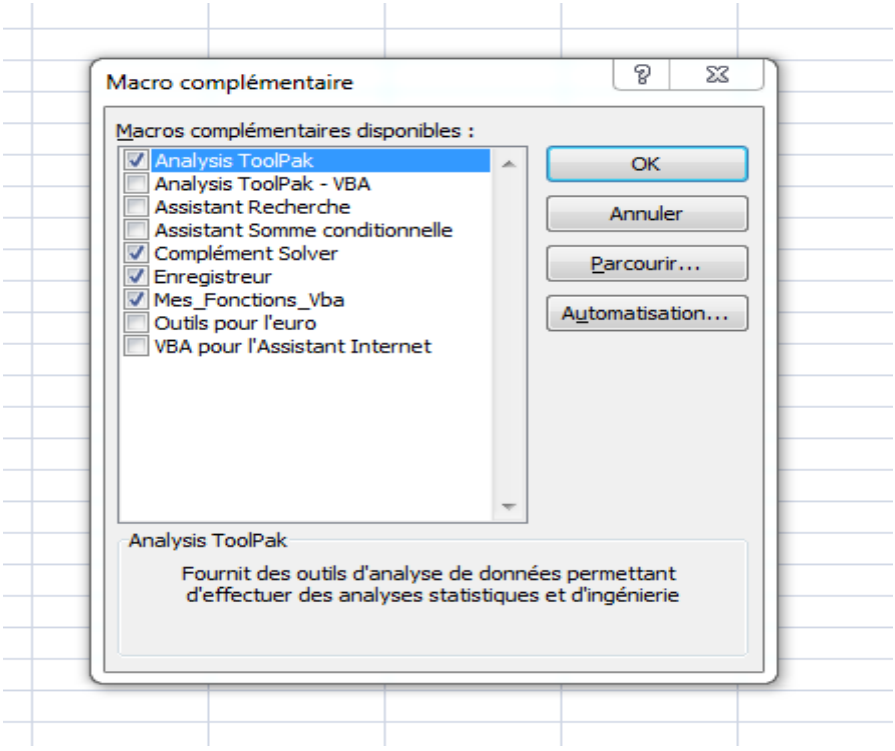


Figure 16: Fenêtre ajout de la fonction aux compléments Excel.

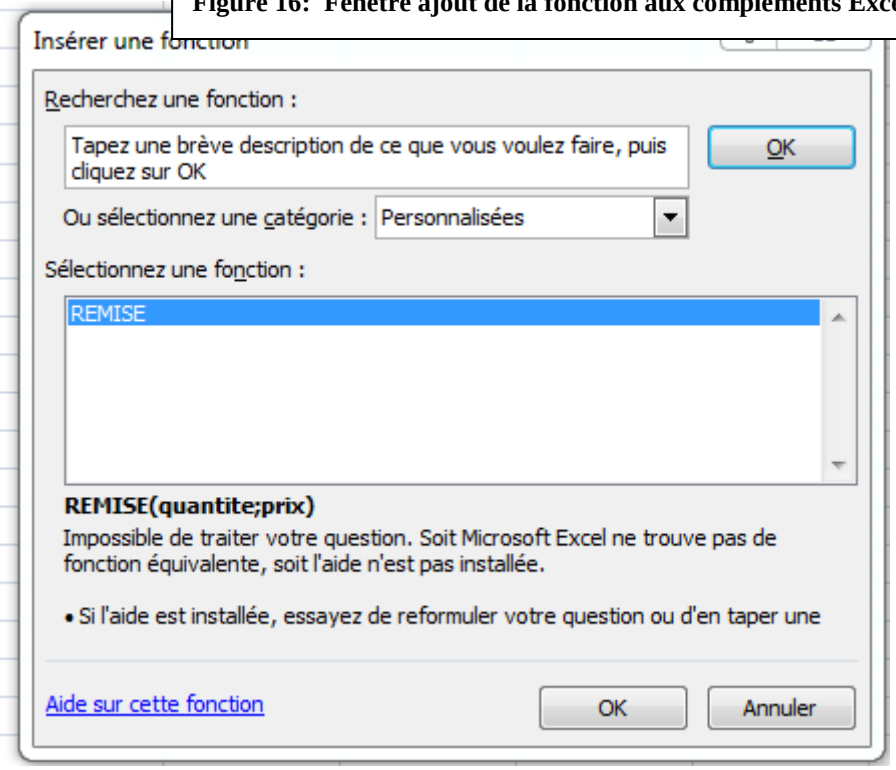


Figure 17: Une fois ajoutée aux compléments Excel, la fonction est appelée par son nom REMISE.

Après avoir suivi ces étapes, les fonctions personnalisées seront disponibles chaque fois qu'Excel est exécuté.

Le fait de double-cliquer sur ce module dans l'Explorateur Project entraîne l'affichage du code Visual Basic'éditeur de fonction. On peut créer autant de fonctions qu'on souhaite de cette manière, et elles seront toujours disponibles dans la catégorie définie par l'utilisateur dans la boîte de dialogue **Insérer** une fonction.

7.4 Création et programmation des boutons

7.4.1 Exemple : Calcul du bénéfice ou perte d'un vendeur d'œufs.

Un commerçant grossiste d'œufs veut calculer son bénéfice. Il achète l'œuf à 8DA, et le vend à 10DA. En cours du transport, en général les 5% des œufs cassent, quel est son bénéfice ou sa perte sachant qu'il a acheté 2500 plateaux contenant chacun 30 œufs.

On suppose :

Une fonction BENEFICE qui calcule le bénéfice.

N le nombre d'œufs=2500*30

PA le prix d'achat=8DA

PV le prix de vente=10DA

Le bénéfice =PV*(N-N*5%) - N*PA

=N*(PV*(1-5%)-PA)

Bénéfice <0 ➔ le commerçant est perdant

7.4.2 Les fonctions utilisées

7.4.2.1 La fonction BENEFICE

En VBA il s'agit de créer la macro fonction personnalisée BENEFICE de la manière suivante:

```
Function  BENEFICE (N as  integer, PA as currency, PV as currency)
as currency
 BENEFICE=N*(PV*(1-5%)-PA)
 End function
```

La fonction BENEFICE peut être exécutée comme n'importe quelle fonction Excel. Elle est accessible dans les fonctions personnalisées.

7.4.2.2 La fonction calcul Benefice

La deuxième étape consiste à écrire une autre fonction personnalisée qui appelle la fonction BENEFICE , appelons cette fonction calcul_Benefice.

```
Function Calcul_Benefice()
```

```
Dim N As Integer, PA As Currency, PV  As Currency
N = [B2]
PA = [C2]
PV = [D2]
[I2] = BENEFICE(N, PV, PA)
End Function
```

La fonction calcul_Benefice suppose que les paramètres d'appel de la fonction BENEFICE() sont respectivement dans les cellules B2 , C2 et D2 : B2 contient le nombre d'œufs, C2 contient le prix d'achat et D2 contient le prix de vente. Il faut préparer la feuille Excel de la manière suivante :

	A	B	C	D	E	F	G
1	Nombre de plateaux	Nombre d'œufs	Prix d'achat unitaire	Prix de vente unitaire			
2	150	4500	8	10			
3							
4							
5		Bénéfice=					
6							

Figure 18: Feuille Excel de l'exemple Calcul du bénéfice ou perte d'un vendeur d'œufs

Pour calculer le bénéfice on peut taper dans la cellule C5 la formule suivante:

=BENEFICE(B2,C2,D2) comme on la fait pour la fonction REMISE.

Mais notre objectif cette fois ci est de créer un bouton, sur lequel il suffit de cliquer pour que le bénéfice soit calculé et stocké dans la cellule C5. Pour cela, une des méthodes consiste à utiliser une autre fonction calcul_Benefice() et suivre la démarche suivante:

1°) Insertion d'une forme de notre choix à l'aide du menu Insertion dans le menu principal. On appelle le bouton calcul_Benefice en changeant le texte à l'intérieur de la forme.

	A	B	C	D	E	F	G
1	Nombre de plateaux	Nombre d'œufs	Prix d'achat unitaire	Prix de vente unitaire			
2	150	4500	8	10			
3						Calcul_Benefice	
4							
5		Bénéfice=					
6							

Figure 19: Feuille Excel de l'exemple Calcul du bénéfice ou perte d'un vendeur d'œufs avec le bouton programmé

2°) On clique à droite sur le bouton créé pour lui associer une macro. On choisit alors l'opération '**affecter**' une macro.

3°) Et on affecte la macro calcul_Benefice(). Cela entraîne que dès qu'on clique sur le bouton, la fonction calcul_ benefice() s'exécute. La fonction calcul_benefice appelle à son tour la fonction BENEFICE () pour calculer le BENEFICE.

4°) La programmation des boutons facilite énormément le travail sous Excel/VBA en exécutant les macros fonctions par de simples cliques.

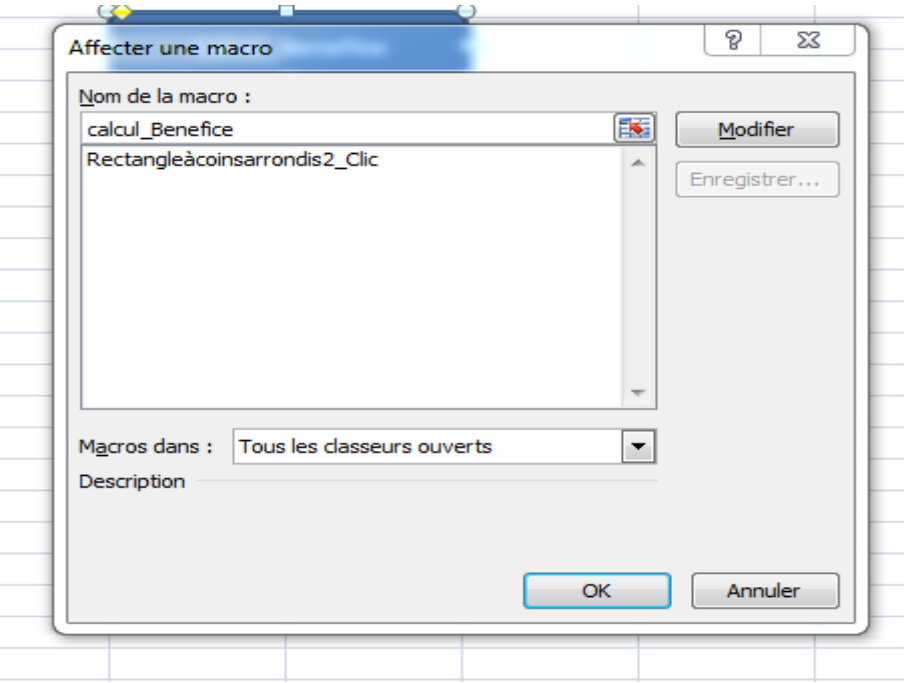


Figure 20: Fenêtre d'affectation de macro à un bouton

7.5 L'Enregistreur de macros

7.5.1 Introduction

Certain traitements sous Excel, comme les mises en forme spécifiques, utilisées de façon répétée, peuvent être réalisés pratiquement une seule fois et enregistrés en tant que macros. Le code de ces macros est automatiquement, généré par Excel sur la base de la manipulation de l'utilisateur. Ces macros peuvent être exécutées ensuite comme n'importe quelles autres macros. Nous présentons ci-après La démarche à suivre pour enregistrer une macro.

Supposons qu'on veuille afficher l'expression suivante comme entête d'une feuille Excel, comme il est montré dans la figure ci-dessous.

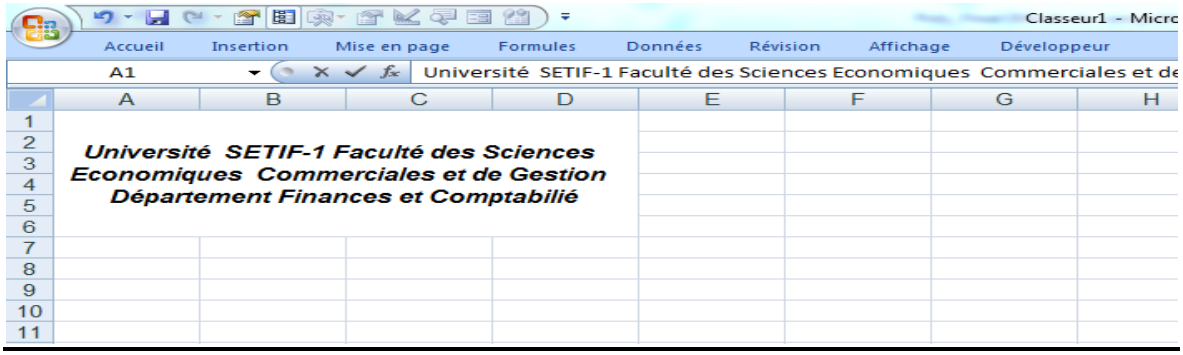


Figure 21: Entête créé avec l'enregistreur de macros

7.5.2 Démarche à suivre

La démarche à suivre est comme suit:

1°) Cliquer sur l'onglet Développeur, ensuite sur *enregistrer une macro*. La boîte de dialogue suivante apparaît. On choisit un nom pour la macro (Affichage_entête), le classeur dans lequel sera enregistrée la macro (classeur de macros personnelles), une touche de raccourci pour exécuter la macro (Ctrl + r), et enfin une description au choix de l'utilisateur (par exemple la date et l'heure de création de la macro) on clique sur ok et continue l'opération.

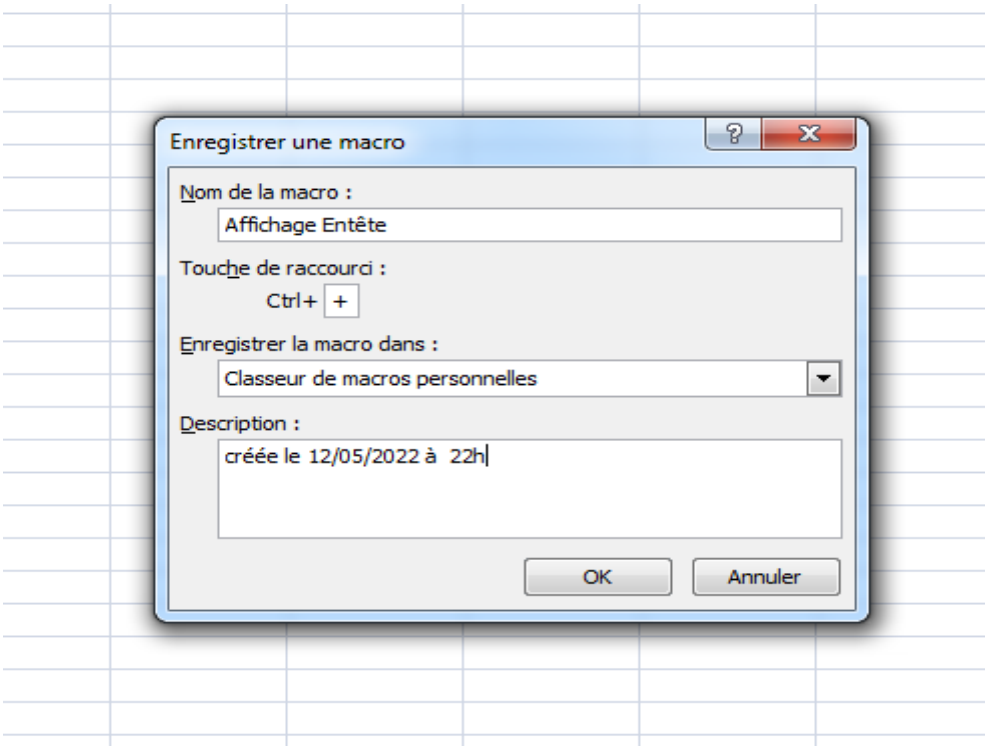


Figure 22: Boîte de dialogue permettant d'enregistrer une macro

2°) Sélectionner la plage de A1:D6

3°) Cliquez à droite au dessus de cette plage

4°) Cliquer sur format cellule et choisir:

- dans alignement : centrer verticalement et horizontalement, fusionner les cellules et passer à la ligne automatiquement
- dans police : Time New Roman, Gras italique et taille 12

5°) Cliquer ensuite sur arrêter l'enregistrement.

Dès que l'utilisateur clique sur *enregistrer macro,* le système se met à enregistrer toutes les actions de l'utilisateur, tout en générant au fur et à mesure le code VBA correspondant à la manipulation de l'utilisateur. Une fois l'enregistrement arrêté, on peut aller sous l'éditeur VBE pour observer le code généré.

7.5.3 Le code généré

```
Sub Display_Entete()
'
' Display_Entete Macro
' créée le 12/05/2022 à  22h
'
' Touche de raccourci du clavier: Ctrl+r
'
    Range("A1:D6").Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .WrapText = True
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = True
    End With
    With Selection.Font
        .Name = "Times New Roman"
        .FontStyle = "Gras italique"
        .Size = 12
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .ThemeFont = xlThemeFontMajor
```

```
    End With
    Range("A1:D6").Select
    ActiveCell.FormulaR1C1 = _
        "Université SETIF-1 Faculté des Sciences Economiques, Commerciales et de Gestion
Département Finances et Comptabilité"
    With ActiveCell.Characters(Start:=1, Length:=116).Font
        .Name = "Times New Roman"
        .FontStyle = "Gras italique"
        .Size = 12
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .ThemeFont = xlThemeFontMajor
    End With
    Range("A7").Select
End Sub
```

Le code généré n'est pas facilement, lisible ou compréhensible par l'utilisateur, mais l'avantage est que dorénavant, il suffit d'appuyer sur Ctrl+r pour afficher sur une feuille Excel cet entête.

7.6 Manipulation des feuilles Excel à partir de VBA

7.6.1 Définitions de base

La manipulation des classeurs Excel à partir d'un programme VBA nécessite l'utilisation du concept objet de VBA Excel. Nous commençons par présenter d'une façon simplifiée et détaillée des différents éléments du modèle objet utilisé en VBA Excel.

1°) **La Classe**

Une classe est une catégorie d'objets ou type d'objets. Un module est une classe, un classeur est une classe, une feuille Excel est une classe.

2°) **L'Objet**

Un objet est une instance d'une classe. Module1 par exemple est un objet de la classe Module, classeur1 est une instance de la classe classeur, Feuil1 est une instance de la classe feuille de calcul. En fait, un objet est une "entité" informatique qu'un programme informatique peut manipuler. Le cas le plus évident est une cellule de classeur. Pour la manipuler, VBA utilise une instance de la classe **Range**.

3°) **La propriété**

A chaque classe d'objets est associée une liste de propriétés, qui a pour rôle de décrire les objets. Les valeurs des propriétés diffèrent d'un objet à l'autre et, la liste des propriétés varie d'une classe à l'autre. Supposons la classe des feuilles Excel qui est décrite par la propriété **Name**, mais chaque feuille a son propre nom (feuil1, feuil2, etc…).

4°) **La méthode**

A chaque classe est associée une liste de méthodes. Une méthode est une action que peut exécuter un objet de la classe en question. La liste des méthodes n'est pas la même pour toutes les classes. Par exemple, la méthode **Copy** d'une cellule permet de copier le contenu de cette cellule dans le presse-papier ou dans une autre cellule.

5°) L'évènement

Un événement permet d'associer une procédure VBA à un objet. Par exemple, l'événement **Change** d'une feuille se produit lorsqu'une cellule de cette feuille est modifiée et permet d'associer une procédure nommée Worksheet_**Change** à cette feuille Excel**.**

6°) La collection

On définit la collection comme étant un ensemble d'objets d'une même classe qui peuvent être adressés avec le nom de la collection et un numéro d'item. Par exemple, un classeur Excel est une collection **Sheets** d'objets **WorkSheet** et/ou **Chart.**

7.6.2 Exemples d'utilisation des objets, propriétés et méthodes VBA Excel

Dans ce cas nous commençons par la définition des différents éléments sous le VBA. Les principaux objets sont workbooks pour faire référence à un classeur, sheets pour faire référence à une feuille, et cells pour faire référence à une cellule.

Exemples :

Workbooks("classeur1.xlsx") ➔ le classeur classeur1.xlsx

Workbooks("classeur1.xlsx").sheets("Feuil1") ➔ la feuille Feuil1 du classeur1.xlsx

Workbooks("classeur1.xlsx").sheets("Feuil1").cells(1,1) ➔ la cellule de coordonnées ligne 1, et colonne 1 de la feuille Feuil1 du classeur classeur1.xlsx.

Ouverture d'un classeur Excel

On utilise la méthode **Open()** de la manière suivante:

```
Workbooks.Open "D:\data\classeurs\Employes.xlsm"
```

Permet d' ouvrir le classeur **Employes.xlsm** qui se trouve dans le dossier **data\classeurs** du disque **D:**.

Fermeture d'un classeur

On utilise la méthode **Close()** de la manière suivante:

```
Workbooks("Employes.xlsm").Close
```

Activation d'une feuille Excel

On utilise la méthode Activate() de la manière suivante:

Worksheets("CLIENT").Activate permet d'activer la feuille CLIENT.

Soit qu'on ouvre la feuille CLIENT manuellement (on l'active), ensuite on bascule sous VBA en appuyant sur ALT+F11, et dans ce cas on n'est pas obligé d'activer la feuille avec l'instruction précédente.

Affecter une valeur à une cellule

Sheets("CLIENT").cells(3, 4) = 10 met le nombre 10 dans la cellule se trouvant à l'intersection de la ligne 3 et la colonne 4 (ou cellule D3), de la feuille CLIENT.

Détermination de la fin de la table stockée sur une feuille Excel

1°) soit le programme:

```
Sub test()
Dim Val1 as integer, Val2 as integer
Ligne=100
Val1= Range("A" & ligne)   ' Le contenu de A100
Val2 =Range("A" & ligne).row   ' Le numéro de la ligne 100
Msgbox "Val1=" & val1 & "   Val2=" & Val2
End Sub
```

Si on suppose que la cellule A100 contient la valeur 31, l'exécution de cette portion entraîne l'affichage:

Val1=31 Val2=100

4°) *Range("A" & Rows.Count).End(xlUp)* représente le contenu de la dernière cellule non vide de la colonne A

Range("A" & Rows.Count).End(xlUp).Row donne le numéro de ligne de la dernière cellule non vide dans la colonne A.

Pour une table de données incluant la colonne A, elle représente la fin de la table ou la fin du fichier.

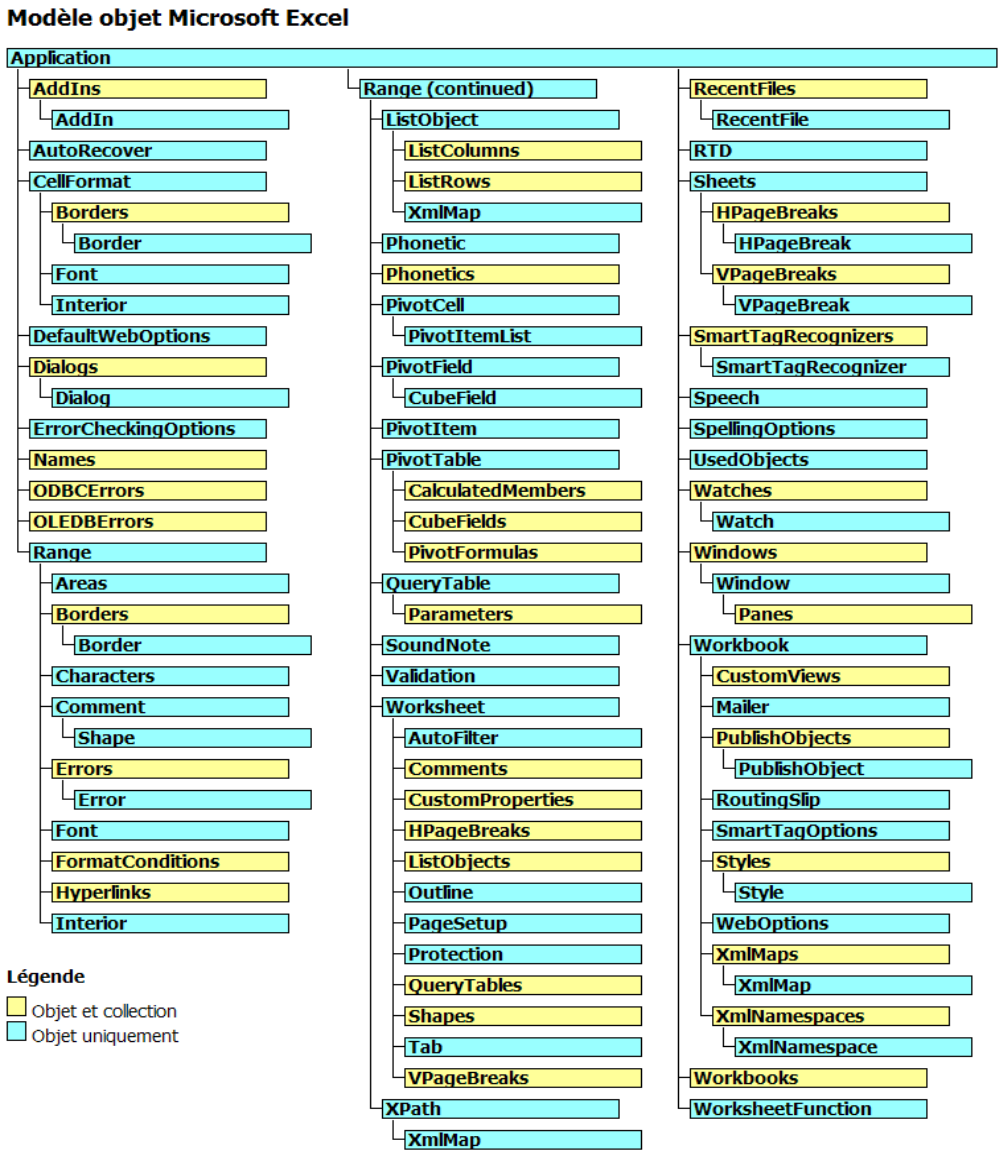


Figure 23 : modèle objet VBA Excel

7.6.3 Calcul sur feuille Excel à partir de VBA

7.6.3.1 Introduction

Le calcul sous Excel simplifie énormément le travail, et les possibilités offertes par VBA pour étendre Excel avantagent beaucoup l'utilisateur. Le calcul des cellules sous Excel peut être fait directement à l'aide de fonctions Excel, ou en utilisant des fonctions personnalisées. Calculer une cellule unique ne pose pas problème, mais copier ensuite coller cette formule dans toute une colonne pour généraliser le calcul, n'est pas très pratique surtout lorsque la table est relativement grande, avec des colonnes longues. L'une des solutions revient à faire le calcul sur la feuille Excel directement à partir de VBA, comme si on faisait un calcul

matriciel classique. Cette façon nécessite l'utilisation de la couche objet VBA/Excel. Nous allons essayer de mettre en œuvre ces notions à l'aide d'exemples.

7.6.3.2 Exemple1 : Remplir un tableau Excel à partir de VBA

Soit la feuille Excel suivante représentant la table d'un ensemble de clients d'un grossiste de légumes et fruits.

	A	B	C	D	E	F	G	H	I	J
1	Num Client	Nom	Prénom	Age	Adresse	Total marchandise	Remise	Total hors Taxes	montant TVA	Net_A_Payer
2	410303003	Abdelli	Mourad	30	Cité 500 Logts	34000,00				
3	410303004	Tounsi	Lamine	37	Cité 1000 Logts	38150,00				
4	410303005	Triaa	Zohir	27	Cité 300 Logts	24350,00				
5	410303006	Yahiaoui	Walid	27	Lapinède	16000,00				
6	410303007	Yousfi	Halim	50	Cité 1006 Logts	4000,00				
7	410303008	Zebboudj	Malik	46	El Hidhab	3500,00				
8	410303009	Zelmat	Samir	30	Cité 300 Logts	6600,00				
9	410303010	Mostefaoui	Fouad	35	Cité 400 Logts	8100,00				
10	410303011	Nabti	Mounir	36	Lapinède	3850,00				
11	410303012	Rebihe	Ali	25	Cité Hachemi	4550,00				
12	410303013	Saci	Houssem	27	Centre ville	4550,00				
13	410303014	Salem	Ali	32	Cité 200 Logts	11000,00				
14	410303015	Khellassi	Abdellah	25	Centre ville	2200,00				
15	410303016	Medjadji	Foudil	55	Bel air	3000,00				
16	410303017	Houari	Hani	54	Cité 1014 Logts	11700,00				
17	410303018	Kasmi	Mohamed	41	Cité 200 Logts	9800,00				
18	410303019	Khellassi	Abdellah	25	Centre ville	9900,00				
19	410303020	Medjadji	Fayçal	55	Bel air	10000,00				
20	410303021	Scandrani	Salim	45	Cité 1006 Logts	12000,00				
21	410303022	Zemirli	Omar	24	Cité 200 Logts	7150,00				
22	410303023	Dahmoune	Kamel	22	Gasrya	3150,00				
23	410303024	Merabet	Hamid	50	Cité 1000 Logts	4500,00				
24	410303025	Chitour	Réda	29	El Hidhab	6600,00				

Figure 24: Feuille Excel des clients du grossiste de légumes et fruits

Chaque client est décrit par les données suivantes:

a) Num_Client : représente le numéro du client.

b) Nom, prénom, Age : Le nom, le prénom et l'âge du client.

c) TotalMarchandise: Le prix total des légumes et fruits achetés par le client.

d) Remise : fixée à 20% si totalMarchandise dépasse la somme de 25000 Da, sinon si totalMarchandise dépasse 15000 Da la remise est de 10%, sinon aucune remise n'est accordée.

e) Total_hors_Taxe, montant_TVA, Net_A_Payer sachant que le taux de TVA est fixé à 17% .

Questions:

1°) Ecrire des fonctions VBA pour calculer, la remise, le total hors taxe, le montant de la TVA ainsi que le net à payer.

2°) Ecrire un programme VBA (SUB) qui appelle ces fonction pour remplir toute la table.

Solution

1-a) La remise est calculée en fonction de TotalMarchandise, le paramètre passé à la fonction est TotalM qui représente le TotalMarchandise.

```
Function remise(TotalM As Double) As Double
' Calcul de la remise:  TotalM  représente l'argument TotalMarchandise.
  If TotalM >= 25000 Then
       remise = TotalM * 0.2
    Else
      If TotalM >= 15000 Then
         remise = TotalM * 0.1
       Else
         remise = 0
       End If
   End If
End Function
```

1-b) TotalHorsTaxe

En général, Total Hors Taxe = Total Marchandise - Remise

Deux paramètres sont passés à la fonction TotalHorsTaxe: TotalM représentant le Total Marchandise et Remise représentant la remise.

1-c) Montant de la TVA

Montant TVA = Taux de TVA * Total Hors Taxe

Deux paramètres sont passés à la fonction *MontantTVA: taux* représentant le taux de la TVA et *totalhtaxe* représentant le Total Hors Taxe.

```
Function MontantTVA(taux As Double, totalhtaxe As Double) As Double
'Calcul du montant de la TVA
MontantTVA = totalhtaxe * taux
End Function
```

1-d) Net à Payer

Net à payer = Total Hors Taxe + TVA

Deux paramètres sont passés à la fonction *NetAPayer : totalhtaxe* représentant le total hors taxe et *MontantTVA représentant le montant de la TVA.*

```
Function NetAPayer(totalhtaxe As Double, MontantTVA As Double) As Double
' Calcul du Net à payer
NetAPayer = totalhtaxe + MontantTVA
End Function
```

2-) la procédure de type SUB *Remplir_Tableau() qui appelle les fonctions précédentes pour remplir le tableau de la feuille de calcul CLIENT.*

```
Sub Remplir_tableau()
' parcours du tableau Excel ligne par ligne et calcul des valeurs demandées dans les cellules appropriées
Dim temp1  As Double
Dim i As Integer
Dim derniere_ligne As Integer
Worksheets("CLIENT").Activate  ' Activation de la feuille CLIENT
' Fixation de la dernière ligne de la table
derniere_ligne = Range("A" & Rows.Count).End(xlUp).Row
i = 2
While i <= derniere_ligne
  temp1 = Cells(i, 6)
  Cells(i, 7) = remis(temp1)
  Cells(i, 8) = TotalHorsTaxe(Cells(i, 6), Cells(i, 7))
  Cells(i, 9) = MontantTVA(0.17, Cells(i, 8))
  Cells(i, 10) = NetAPayer(Cells(i, 9), Cells(i, 8))
  i = i + 1
Wend
End Sub
```

7.6.3.3 Exemple2: Programmer une fonction Recherche () en VBA

Soit la feuille Excel Feuil1 du fichier Employes.xlsx, représentant une liste des douaniers travaillant dans les différents aéroports d'Algérie. On souhaite compléter la colonne salaire de la liste des employés à partir de la table des salaires de base.

Liste Employés

Nom	Prénom	Sexe	Age	Lieu de Naissance	Qualifi-cation	Site(Aéroport d',,,)	Salaire
Zenati	Ibtissem	Femme	25	Blida	Cadre Supérieur	Alger	
zemirli	omar	Homme	30	Oran	Cadre Supérieur	Béjaia	
zelmat	samir	Homme	31	Alger	Cadre Supérieur	Oran	
Yousfi	Halim	Homme	29	Alger	Agent	Constantine	
Tebbani	merouane	Homme	40	Sétif	Cadre Supérieur	Chlef	
slimai	lila	Femme	50	Blida	Agent	Ghardaia	
selloum	Hanafi	Homme	24	Alger	Agent	Tlemcen	
sellami	merouane	Homme	25	Alger	Agent	In-Aménas	
Saouli	Nawel	Femme	28	Bougie	Maîtrise	Annaba	
Rabihi	hind	Femme	27	constantine	Maîtrise	Djanet	
rabhi	sofiane	Homme	35	Oran	Cadre Supérieur	Hassi Messaoud	
Nabti	mounir	Homme	30	Blida	Maîtrise	Tamanra-sset	
Mostefaoui	fouad	Homme	35	Blida	Cadre Supérieur	Alger	
Miloudi	loubna	Femme	25	el oued	Maîtrise	Béjaia	
mihoubi	houssem	Homme	24	ouargla	Agent	Oran	
Midouni	camélia	Femme	22	laghouat	Maîtrise	Constantine	
hedjress	Fayçal	Homme	26	chlef	Agent	Chlef	
haroun	yacine	Homme	32	msila	Cadre Supérieur	Ghardaia	
fillali	billel	Homme	34	saida	Cadre Supérieur	Tlemcen	
Chorfi	Samir	Homme	23	msila	Agent	In-Aménas	

Les salaires de base par qualification

Qualifi-cation	Salaire de base
Agent	50 000,00 DZD
Cadre Supérieur	100 000,00 DZD
Maîtrise	70 000,00 DZD

Figure 25: Feuille Excel d'une liste de douaniers des aéroports d'Algérie

Cela peut être fait de trois manières différentes :

1) Directement sous Excel à l'aide de la fonction recherche ()

   =RECHERCHE(F4;L$4:L$6;M$4:M$6)

2) Créer une fonction en VBA à l'aide de l'enregistreur de macros.

```
' Macro générée par l'enregistreur de Macros
Sub Macro1()
' Macro1 Macro
    ActiveCell.FormulaR1C1 = "=VLOOKUP(RC[-2],R4C[4]:R6C[5],2,FALSE)"
    Range("H4").Select
    Selection.AutoFill Destination:=Range("H4:H102"), Type:=xlFillDefault
    Range("H4:H102").Select
End Sub
```

3) Ecrire un programme VBA qui accède à la feuille Excel et qui fait le traitement.

```
' Macro écrite par le programmeur
' Il s'agit d'écrire soit même un programme qui fait la recherche et la généralisation
Sub automatisation()
```

```
Dim i As Integer, j As Integer
Dim val_cherche As String
Dim derniere_ligne As Integer
' Feuil1 activée
derniere_ligne = Range("A" & Rows.Count).End(xlUp).Row
For i = 4 To derniere_ligne
    val_cherche = Feuil1.Cells(i, 6).Value
    j = 4
    While j <= 6 And Feuil1.Cells(j, 12) <> val_cherche
       j = j + 1
  Wend
  Feuil1.Cells(i, 8).Value = Feuil1.Cells(j, 13).Value
 Next i
 Feuil1.Select
End Sub
```

7.7 Exercices

Exercice1

Ecrire un programme VBA qui fait la résolution d'une équation du deuxième degré $ax^2 + bx + c = 0$ de la manière suivante: Lit les valeurs de a, b, c à partir de la feuille Excel et qui affiche également , delta, les éventuelles solutions et messages sur la feuille Excel.

1) Utilise une fonction qui calcule delta et une fonction qui résoud une équation du premier degré.

2) Programmer un bouton pour exécuter l'équation.

Faire les mises en forme nécessaires sur la feuille Excel pour qu'il y est cohérence entre l'affichage VBA et la taille des cellules.

Exercice2

Soit la table Excel suivante:

	A	B	C	D	E	F	G	H	I	J
1		Module		Informatique	Matématique	physique				
2		Coefficient		5	4	3				
3	NºEtudiant	Nom et Prénom Etudiant		Note			Nbr Absence	Moyenne	Décision	Mention
4	1	Slimani Karim		10	8	15	6			
5	2	Ben Yahya Samira		10	12	16	3			
6	3	Guessmia Karim		9	14	10	8			
7	4	Bessou Amira		9	3	9	2			
8	5	Ibrahimi Salim		10	12	6	3			
9	6	Hadjeb Rachid		4	16	10	6			
10	7	Khaloufi Ahlem		10	5	15	4			
11	8	Bouzidi Farid		10	10	10	5			
12	9	Halitim Yacine		20	12	14	9			
13	10	Tali Fatima		17	10	12	0			

Ecrire le programme VBA qui complète cette table sachant que:

1. La moyenne de chaque étudiant est calculée par la formule:

$$\text{Moyenne} = \frac{\sum note\ \ du\ modulei * coefficient\ \ du\ modulei}{somme\ \ des\ coefficients}$$

2. La décision pour chaque étudiant est telle que :
   a. L'étudiant est « Exclus », si son nombre d'absences est supérieur ou égale à 05.
   b. L'étudiant est « Admis », si sa moyenne est supérieure ou égale à 10.
   c. L'étudiant est « Racheté », si sa moyenne est supérieure ou égale à 09.
   d. L'étudiant est « Ajourné », si sa moyenne est inférieure strictement à 09.
3. La colonne mention est définie de la manière suivante :
   a. mention= 'très bien' si moy > = 16
   b. mention = 'Bien' si 16 > moy >=14
   c. mention = 'Assez Bien' si 14 > moy >=12
   d. mention = 'Passable' si 12 > moy >=11
   e. mention = 'sans mention' si 11> moy >=10 ("sans mention" est affiché également pour les étudiants rachetés.)
   f. mention = '* ' si moy< 9

Corrigés des exercices des chapitres

1. Chapitre2

Exercice1

Variable	***Algorithme***	***VBA***
A	*Entier* ➔ *Réel* ➔	*Integer, long* *Currency, single, double*
B	*Réel*	*Currency single double*
C	*Caractère ou chaîne de caractère*	*String*
D	*Chaîne de caractères*	*String*
E	*entier* ➔ *réel* ➔	*Byte integer long* *Currency, single, double*
F	*Réel*	*Currency single double*
G	*Booléen*	*Boolean*

Exercice2

Séquence 1

1 →

Lire(x, Y) ;

2→

X← Y ;

3→

Y← X ;

4→

Ecrire(x,y) ;

5→

On lit 2 dans x et 5 dans Y

	X	Y
1	0	0
2	2	5
3	5	5
4	5	5
5	5	5

Séquence 2

1→
Lire(X, Y) ;
2→
Temp ← X ;
3→
X← Y ;
4→
Y←Temp ;
5→
Ecrire(x,y) ;
6→

On lit *-20 dans X et 10 dans Y dans la séquence 2*

	X	Y	Temp
1	0	0	0
2	-20	10	0
3	-20	10	-20
4	10	10	-20
5	10	-20	-20
6	10	-20	-20

Séquence 3

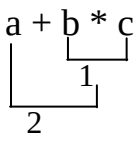
1→
Lire(X, Y) ;
2→
Temp ← Y ;
3→
Y← X ;
4→
X←Temp ;
5→
Ecrire(x,y) ;
6→

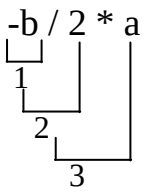
On lit 30 dans X et 100 dans Y dans la séquence 3.

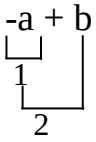
	X	y	Temp
1	0	0	0
2	30	100	0
3	30	100	100
4	30	30	100
5	100	30	100
6	100	30	100

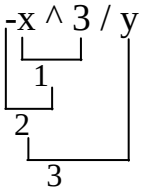
```
Algorithme  Permut;
Variables
   A,  B,  Temp  : entier;
Début
 Lire(A, B);
Temp ← A ;
A← B ;
B←Temp ;
Ecrire(A, B) ;
FinAlgorithme.
```

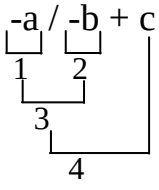
Exercice3

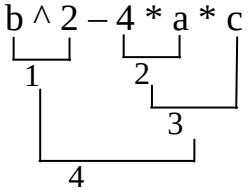


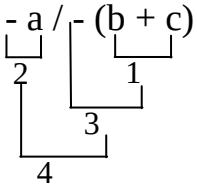


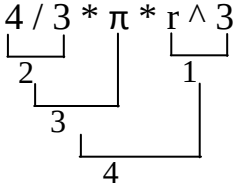


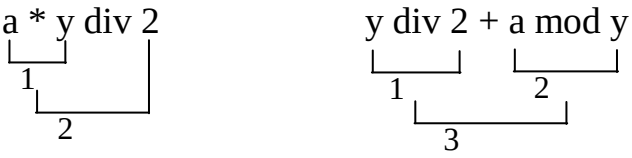












Exercice 4

Expression	*Négation*	*VBA*
A	*Non A*	*Not A*
A ET B	*NON A OU NON B*	*NOT A OR NOT B*
A OU B	*NON A ET NON B*	*NOT A AND NOT B*
y > t	*y ≤ t*	*y <= t*
x ≤ z	*x > z*	*x > z*
(x<y) ET (z ≥ t)	*(x≥y) OU (z< t)*	*(x>=y) OR (z< t)*
(y>t) OU (x<z)	*(y≤t) ET (x≥z)*	*(y<=) AND (x>=z)*

2. Chapitre3

Exercice1

T	*X*	*Y*	*Z*
20	*-1*	*-1*	*0*
0	*2*	*5*	*2*
-1	*2*	*2*	*9*

Exercice2

Algorithme	*VBA*
Algorithme *systemeEQ;* *# Résolution d'un système de deux 'équations à 2 inconnues #* ***Variables*** *a1, a2, b1, b2, c1, c2, delta : entier;* *x, y : réel;* ***Début*** *Lire(a1, a2, b1, b2, c1, c2);* *delta ← a1*b2 -a2*b1;* ***Si*** *delta = 0* ***alors*** *écrire("il n'y a pas de solutions");* ***sinon*** *x ← (c1*b2 - c2* b1)/ delta;* *y ← (a1*c2 – a2* c1)/ delta;* *Ecrire(x, y);* ***Finsi;*** ***FinAlgorithme.***	*Sub systemeEQ()* *' Résolution d'un système de deux équations à deux inconnues* *Dim a1 As Integer, a2 As Integer* *Dim b1 As Integer, b2 As Integer* *Dim c1 As Integer, c2 As Integer, delta As Integer* *Dim x As Double, y As Double* *' Lecture des données* *a1 = InputBox("donnez a1")* *b1 = InputBox("donnez b1")* *c1 = InputBox("donnez c1")* *a2 = InputBox("donnez a2")* *b2 = InputBox("donnez b2")* *c2 = InputBox("donnez c2")* *' Traitement* *delta = a1 * b2 - a2 * b1* *If delta = 0 Then* *MsgBox " Il n' y'a pas de solutions"* *Else* *x = (c1 * b2 - c2 * b1) / delta* *y = (a1 * c2 - a2 * c1) / delta* *MsgBox "x= " & x & " y=" & y* *End If* *End Sub*

Exercice3

Valeurs lues de A et B	*Portion 1*	*Portion 2*	*Portion 3*
A= 15 B= 2	*A=17 B=8*	*A=58 B=174*	*A= -15 B=49*
A=10 B=32	*A=2 B=6*	*A= 22 B=66*	*A= 10 B=61*
A=5 B=5	*A=0 B=0*	*A=20 B=60*	*A= -20 B=46*

Exercice4

Algorithme	***VBA***
Algorithme *EquaDegre2;* ***Variables*** *a, b, c, delta : entier ;* *X1, X2 : réel ;* ***Début*** *Lire(a, b, c) ;* ***Si*** *a = 0* ***alors*** ***Si*** *b = 0* ***alors*** # *Le cas où a, b et c = 0* # ***Si*** *c=0* ***alors*** *Ecrire("La solution est R") ;* # *Le cas où a, b =0 mais pas c* # ***Sinon*** *Ecrire("Pas de solution ") ;* ***FinSi*** #*Le cas où a =0 mais pas b, équation 1 degré* # ***Sinon*** *X1 ← -c/b ;* *Ecrire(X1) ;* ***FinSi*** # *Le cas où a ≠ 0* ***Sinon*** *delta ← b * b – 4 * a * c ;* ***Si*** *delta < 0* ***alors*** *Ecrire("Pas de solution") ;*	***Sub*** *EquaDegre2 ()* *Dim a As Integer, b As Integer, c As Integer* *Dim delta As Integer* *Dim X1 As Double, X2 As Double* *a = InputBox("Donner la valeur de a : ")* *b = InputBox("Donner la valeur de b : ")* *c = InputBox("Donner la valeur de c : ")* ***If*** *a = 0* ***Then*** ***If*** *b = 0* ***Then*** ***If*** *c = 0* ***Then*** *MsgBox "La solution est R"* ***Else*** *MsgBox "Pas de solution"* ***End If*** ***Else*** *X1 = -c / b* *MsgBox "La solution est X = " & X1* ***End If*** ***Else*** *delta = b * b - 4 * a * c* ***If*** *delta < 0* ***Then*** *MsgBox "Pas de solution"* ***Else*** *X1 = (-b - Sqr(delta)) / (2 * a)* *X2 = (-b + Sqr(delta)) / (2 * a)*

# Le cas où delta =0 est inclus implicitement # ***Sinon*** *X1 ← (-b - racine(delta)) / (2 * a) ;* *X2 ← (-b + racine(delta)) / (2 * a) ;* *Ecrire(X1, X2) ;* ***FinSi*** ***FinSi*** ***FinAlgorithme***	*MsgBox " X1 = " & X1 & " X2= " & X2* ***End If*** ***End If*** ***End Sub***

Exercice5

Algorithme
Algorithme *TraiteIndividu ;* ***Variables*** *sexe: chaine;* *age: entier;* *taille: réel;* ***Début*** *Lire(sexe, age, taille);* ***Si*** *age >= 18* ***alors*** ***Si*** *sexe = "f"* ***alors*** ***Si*** *taille >= 1.7* ***alors*** *Ecrire("femme majeure grande") ;* ***Sinon*** ***Si*** *taille < 1.5* ***alors*** *Ecrire("femme majeure petite") ;* ***Sinon*** *Ecrire("femme majeure avec taille moyenne") ;* ***FinSi*** ***FinSi*** ***Sinon*** # il s'agit d'un homme # ***Si*** *taille >= 1.8* ***alors***

```
            Ecrire("homme majeur grand") ;
         Sinon
            Si taille < 1.6 alors
               Ecrire("homme majeur petit") ;
            Sinon
               Ecrire("homme majeur avec taille moyenne") ;
            FinSi
         FinSi
   Sinon
      #  cas d'un individu mineur #
      Si sexe = "f" alors
         Si taille >= 1.7 alors
            Ecrire("femme mineure grande") ;
         Sinon
            Si taille < 1.5 alors
               Ecrire("femme mineure petite") ;
            Sinon
               Ecrire("femme mineure avec taille moyenne") ;
             FinSi
         FinSi
      Sinon   #  il s'agit d'un homme #
         Si taille >= 1.8 alors
            Ecrire("homme mineur grand") ;
         Sinon
            Si taille < 1.6 alors
               Ecrire("homme mineur petit") ;
            Sinon
               Ecrire("homme mineur avec taille moyenne") ;
            FinSi
      FinSi
   FinSi
FinAlgorithme
```

VBA

```
Sub TraiteIndividu()
    Dim sexe As String, age As Integer, taille As Double
    sexe = InputBox("sexe:")
    age = InputBox("age:")
    taille = InputBox("taille:")

    If age >= 18 Then
        If sexe = "f" Then
            If taille >= 1.7 Then
                MsgBox "femme majeure grande"
            Else
                If taille < 1.5 Then
                    MsgBox "femme majeure petite"
                Else
                    MsgBox "femme majeure avec taille moyenne"
                End If
            End If
        Else
            ' il s'agit d'un homme
            If taille >= 1.8 Then
                MsgBox "homme majeur grand"
            Else
                If taille < 1.6 Then
                    MsgBox "homme majeur petit"
                Else
                    MsgBox "homme majeur avec taille moyenne"
                End If
            End If
    Else
        ' cas d'un individu mineur
        If sexe = "f" Then
            If taille >= 1.7 Then
```

```
            MsgBox "femme mineure grande"
        Else
            If taille < 1.5 Then
                MsgBox "femme mineure petite"
            Else
                MsgBox "femme mineure avec taille moyenne"
            End If
        End If
    Else
        ' il s'agit d'un homme
        If taille >= 1.8 Then
            MsgBox "homme mineur grand"
        Else
            If taille < 1.6 Then
                MsgBox "homme mineur petit"
            Else
                MsgBox "homme mineur avec taille moyenne"
            End If
    End If
  End If
End Sub
```

Exercice6

Algorithme
Algorithme CalculIMC; *# Calcul de l'indice de masse corporelle #* ***Variables*** *Taille, poids, IMC : réèl;* ***Début*** *Lire (taille, poids);*

```
IMC ← poids/taille^2;
Selon  cas IMC
  Cas est <17
      Ecrire("IMC=", IMC, "Gravement en sous-poids");
  Cas 17 à 18,49
      Ecrire( "IMC= ",  IMC , "  En sous-poids")
  Cas 18.5 à  24.99
      Ecrire( "IMC= " , IMC , " Poids idéal")
  Cas 25 à  24.99
      Ecrire( "IMC= " , IMC , " Surpoids")
  Cas 30 à 34.99
      Ecrire( "IMC= " , IMC ," Obese")
  Cas 35 à 39.99
      Ecrire( "IMC= " , IMC , " Gravement Obese")
  Cas est > 40
     Ecrire( "IMC= " , IMC , " Extrêmement Obese")
Fin Selon
FinAlgorithme.
```

VBA

```
Sub calculIMC()
' Calcul de l'indice de masse corporelle
Dim taille As Double, poids As Double, IMC As Double
taille = InputBox("donnez la taille")
poids = InputBox("donnez le poids")
```

```
IMC = poids / taille ^ 2
Select Case IMC
  Case Is < 17
     MsgBox "IMC= " & IMC & "  Gravement en sous-poids"
  Case 17 To 18.49
     MsgBox "IMC= " & IMC & "  En sous-poids"
  Case 18.5 To 24.99
     MsgBox "IMC= " & IMC & " Poids idéal"
  Case 25 To 24.99
     MsgBox "IMC= " & IMC & " Surpoids"
  Case 30 To 34.99
     MsgBox "IMC= " & IMC & " Obese"
  Case 35 To 39.99
     MsgBox "IMC= " & IMC & " Gravement Obese"
  Case Is > 40
     MsgBox "IMC= " & IMC & " Extrêmement Obese"
End Select
End Sub
```

3. Chapitre4

Exercice1

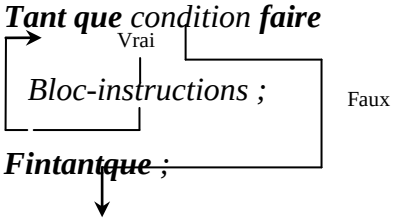
1. ***La boucle Tant Que:***

Syntaxe générale:

Tant que *condition* ***faire***

Bloc-instructions ;

Fintantque *;*



Algorithmique :	***VBA :***
Algorithme *SomMoyAges ;*	***Sub*** *SomMoyAges ()*
Variables	*Dim N As Integer, Age As Integer, Somme As*
N, Age, somme, i : entier ;	*Integer*
Moy : réel;	*Dim i As Integer, Moy as double*
Début	
Lire(N) ;	*N = InputBox("Donnez N: ")*
somme ← 0 ;	*Somme = 0*
i ← 1 ;	*i = 1*
Tant que *i ≤ N* ***faire***	***While*** *i <= N*
Lire(Age) ;	*Age = InputBox("Age : ")*
somme ← somme + Age ;	*Somme = Somme + Age*
i ← i + 1 ;	*i = i + 1*
Fintantque *;*	***Wend***
Moy ← somme/N;	*Moy=somme/N*
Ecrire(somme, Moy) ;	*MsgBox "Somme : " & Somme & "moy : " & Moy*
FinAlgorithme	***End Sub***

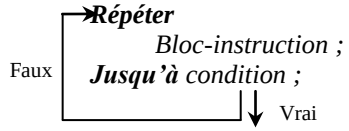
Le déroulement de la boucle pour N=5, et les nombres 10, 11, 14, 13, 12 :

N	***I***	***Age***	***Somme***
5	~~*1*~~	~~*10*~~	~~*10*~~

	~~2~~	~~11~~	~~21~~
	~~3~~	~~14~~	~~35~~
	~~4~~	~~13~~	~~48~~
	~~5~~	~~12~~	~~60~~
	6		60

2. La boucle répéter:

Syntaxe générale:



Algorithmique :	***VBA :***
Algorithme *SomMoyAges ;* ***Variables*** *N, Age, somme, i : entier ;* *Moy : réel;* ***Début*** *Lire(N) ;* *somme ← 0 ;* *i ← 1 ;* ***Répéter*** *Lire(Age) ;* *somme ← somme + Age ;* *i ← i + 1 ;* ***Jusqu'à*** *i > N ;* *Moy←somme/N;* *Ecrire(somme, Moy) ;* ***FinAlgorithme***	***Sub*** *SomMoyAges ()* *Dim N As Integer, Age As Integer, Somme As Integer* *Dim i As Integer, Moy as double* *N = InputBox("Donnez N: ")* *Somme = 0* *i = 1* ***do*** *Age = InputBox("Age : ")* *Somme = Somme + Age* *i = i + 1* ***Loop Until*** *i>N* *Moy=somme/N* *MsgBox "Somme : " & Somme & "moy : " & Moy* ***End Sub***

3. La boucle Pour:

Syntaxe générale:

Pour *compteur ←x* ***à*** *y* ***pas*** *z* ***faire***
Bloc-instruction ;
FinPour *;*

N.B. *Si l'argument « **z** » n'est pas spécifié, le compteur est incrémenté de 1.*

Algorithmique :	***VBA :***
Algorithme *SomMoyAges ;*	***Sub*** *SomMoyAges ()*
Variables	*Dim N As Integer, Age As Integer*
N, Age, somme, i : entier ;	*Dim Somme As Integer*
Moy : réel;	*Dim i As Integer, Moy as double*
	N = InputBox("Donnez N: ")
Début	*Somme = 0*
Lire(N) ;	***for*** *i=1* ***to*** *N* ***step*** *1*
somme ← 0 ;	*Age = InputBox("Age : ")*
Pour *i ←1* ***à*** *N* ***pas*** *1* ***faire***	*Somme = Somme + Age*
Lire(Age) ;	***Next*** *i*
Somme←Somme+Age ;	*Moy=somme/N*
FinPour *;*	*MsgBox "Somme : " & Somme & "moy : " & Moy*
Moy ←somme/N	***End Sub***
Ecrire(somme, Moy) ;	
FinAlgorithme	

Exercice2

Algorithme	***Variables de Contrôle***	***Résultat d'exécution***	***VBA***	***Pour***	***Répéter***

1	*I*	*écriture de: 11 fois "Bon courage"*	*i =-1* ***while*** *i < 10* *msgbox "bon courage"* *i=i+1* ***wend***	***pour*** *i ← -1* ***à*** *9* ***faire*** *écrire('bon courage');* ***Finpour;***	*i ← -1 ;* ***Répéter*** *ecrire('bon courage');* *i←i+1;* ***jusqu'à*** *i>=10;*
2	*I*	*écriture de: infinité de fois "Bon courage" car le pas de la boucle n'est pas spécifié.*	*i=-1,* ***while*** *i < 10* *msgbox "bon courage"* ***wend*** *i=i+2*	***pour*** *i ← -1 à 9* ***pas****=0* ***faire*** *écrire('bon courage');* ***Finpour****;* *i←i+2*	*i ← -1 ;* ***répéter*** *Ecrire('bon courage');* ***jusqu'à*** *i>=10;* *i←i+2;*
3	*a, b*	*L'écriture dépend des valeurs lues de a et b*	*a=inputbox("A=")* *b= inputbox("B=")* ***while*** *a-b > 0* *msgbox "bon courage"* *a=inputbox("A=")* *b= inputbox("B=")* ***wend***	*Ce n'est pas possible car le pas n'est pas connu. Et l'expression de contrôle n'est pas de type ordinal.*	*lire(a,b);* ***répéter***

N.B. *Pour la portion d'algorithme 3, La traduction avec répéter est possible, mais au niveau de l'exécution ce n'est pas pareil. Car la boucle répéter s'exécute au moins 1 fois quelque soient les valeurs lues de A et B au début. Par contre la boucle tant que peut ne jamais s'exécuter si les valeurs initiales de A et B ne vérifient pas la condition d'exécution. Par contre, pour la portion 1, les boucles tant que, pour et répéter sont tout à fait équivalentes.*

Exercice3

Algorithme	**VBA**
Algorithme *calcFactoriel;*	***Sub*** *calcFactoriel()*

# lecture de l'entier naturel N et calcul de N! # ***Variables*** *Fact, N, i : entier;* ***Début*** *Lire(N); # on suppose que N est > ou =0 #* *Fact←1;* ***Pour** i←1 à N **Faire*** *Fact ← Fact*i;* ***Finpour;*** *Ecrire (Fact);* ***FinAlgorithme.***	*' Lecture de N>=0 et calcul de N!* *Dim Fact as long , N as Byte, i as Byte* *N=inputbox("Donnez N")* *Fact=1* ***For** i=1 to N* *Fact=Fact*i* ***Next** i* *Msgbox N & "! =" & Fact* ***End Sub***

Exercice4

Algorithmique	VBA
Algorithme sommat; # calcul de la somme de deux matrices # Constantes N←3; M←4; Variables *A[1..N,1..M] : Tableau de réel;* *B[1..N,1..M] : Tableau de réel;* *C[1..N,1..M] : Tableau de réel;* *i, j : entier;* Début # La lecture # *i←1;* *Tant que i≤ N faire* *j←1;* *Tant que j≤ M Faire* *lire (A[i,j], B[i,j]);* *C[i,j] ← A[i,j] + B[i,j];*	*Sub sommat()* *' Calcul de la somme de deux matrices* *Const n = 3, m = 4* *Dim A(1 To n, 1 To m) As Integer* *Dim B(1 To n, 1 To m) As Integer* *Dim C(1 To n, 1 To m) As Integer* *Dim i, j As Integer* *' lecture des matrices A et B et calcul de la somme* *i = 1* *While i <= n* *j = 1* *While j <= m* *A(i, j) = InputBox("donnez A(" & i & "," & j & ") =")* *B(i, j) = InputBox("donnez B(" & i & "," & j & ") =")* *C(i, j) = A(i, j) + B(i, j)* *j = j + 1* *Wend* *i = i + 1* *Wend*

j←j+1; *FinTant que;* *i←i+1;* *FinTantque;* *# L'écriture de la matrice résultat #* *Pour i ← 1 To n faire* *Pour j ← 1 To m faire* *écrire("C[",i,",",j,"]=", C[i,j]);* *FinPour;* *FinPour;* *FinAlgorithme.*	*' Ecriture de la matrice* *For i = 1 To n* *For j = 1 To m* *MsgBox " C(" & i & "," & j & ") =" & C(i, j)* *Next j* *Next i* *End Sub*

Déroulement dans le cas des deux matrices :

$$A=\begin{pmatrix}12 & 21 & -12 & 25\\ -5 & 18 & 3 & 6\\ 10 & 0 & 7 & -1\end{pmatrix}$$

$$B=\begin{pmatrix}-4 & 3 & 15 & 18\\ 10 & 8 & 0 & 21\\ 3 & -4 & 16 & 31\end{pmatrix}$$

I	J	A[i,j]	B[i,j]	C[i,j]
1	1	12	-4	**8**
	2	21	3	**24**
	3	-12	15	**3**
	4	25	18	**43**
	5			
2	1	-5	10	**5**
	2	18	8	**26**
	3	3	0	**3**
	4	6	21	**27**
	5			
3	1	10	3	**13**
	2	0	-4	**-4**
	3	7	16	**23**
	4	-1	31	**30**
	5			
6				

4. Chapitre5

Algorithme

```
Algorithme Trait_Ages;
Variables
  K , minimum            :  entier;
  Variance, ecart_type  :  réel;
 V[1..20]                 : entier;

Procédure  lect_vect(N :entier, vect():entier)
Vari ables
   i : entier;
Début
  Pour  i ← 1 To N
       Lire(vect(i))
 Finpour
FinProcédure

Procédure  Ecr_vect(N :entier, vect():entier)
Vari ables
   i , somme: entier;
Début
  Pour  i ← 1 To N
       Ecrire(vect(i))
 Finpour
FinProcédure

Fonction moyenne(N :entier, vect() : entier) :réel);
Vari ables
   i , somme: entier;
Début
somme ← 0
Pour  i ← 1 To N
   somme ← somme + vect(i)
```

```
Finpour
moyenne ← somme / N
FinFonction

Procédure calc_vari_ectype(N :entier, vect():entier, moy :réel, vari  :réel, ectype  :réel)
Vari ables
   i , somme: entier;
Début
   somme ← 0
   Pour i ← 1 To N
      somme ← somme + (vect(i) - moy) ^ 2
   Finpour
  vari ← somme / N
  ectype ← Sqr(vari)
FinProcédure

Procédure calc_min(N : entier,  vect(): entier, min : entier)
Vari ables
   i : entier;
Début
min ← vect(1)
Pour  i ← 2 à N
   Si    vect(i) < min   Alors
      min ← vect(i)
   Finsi
FinPour
FinProcédure

Fonction max(N :entier, vect() :entier) :entier;
Variables
   I  :  entier;
Début
  max ← vect(1)
```

```
  Pour i ← 2 To N
     Si vect(i) > max Alors
        max ← vect(i)
     Finsi
  Finpour
FinFonction

#  Programme Principal  #
Début
   Lire(k);
  Lect_vect(k,v);
  Calc_vari_ectype(k,v,moyenne(k,v),  variance,ecart_type)
  Ecr_vect(k, v)
  calc_min(k, v, minimum)
  Ecrire(" moyenne= " ,  moyenne(k, v) ,  "  variance= " , variance , "  Ecart_type= " ,
ecart_type , "  min= " ,  minimum,  "  max=" , max(k, v))
FinAlgorithme.
```

VBA

```
Sub lect_vect(N As Byte, vect() As Integer)
Dim i As Byte
For i = 1 To N
vect(i) = InputBox("donnez v(" & i & ")<=20")
Next
End Sub

Sub Ecr_vect(N As Byte, vect() As Integer)
Dim i As Byte
For i = 1 To N
MsgBox "vect(" & i & ")= " & vect(i)
Next
End Sub
```

```
Function moyenne(N As Byte, vect() As Integer) As Double
Dim somme as Integer,  I as integer
somme = 0
For i = 1 To N
somme = somme + vect(i)
Next
moyenne = somme / N
End Function

Sub calc_vari_ectype(N As Byte, vect() As Integer, moy As Double, vari As Double,
ectype As Double)
Dim somme As Integer
somme = 0
For i = 1 To N
somme = somme + (vect(i) - moy) ^ 2
Next
vari = somme / N
ectype = Sqr(vari)
End Sub


Sub calc_min(N As Byte, vect() As Integer, min As Integer)
min = vect(1)
For i = 2 To N
If vect(i) < min Then
min = vect(i)
End If
Next
End Sub


Function max(N As Byte, vect() As Integer) As Integer
max = vect(1)
```

```
For i = 2 To N
If vect(i) > max Then
max = vect(i)
End If
Next
End Function

Sub Trait_Ages()
Dim k As Byte, v() As Integer, variance As Double
Dim ecart_type As Double, minimum As Integer
k = InputBox("donnez la dimension")
ReDim v(k) As Integer
Call lect_vect(k, v)
Call calc_vari_ectype(k, v, moyenne(k, v), variance, ecart_type)
Call Ecr_vect(k, v)
Call calc_min(k, v, minimum)
MsgBox "moyenne= " & moyenne(k, v) & " variance= " & variance & " Ecart_type= "
& ecart_type & " min= " & minimum & " max=" & max(k, v)
End Sub
```

5. Chapitre6

```
' Résolution d'une équation du deuxième degré en lisant a, b et c de la feuille Excel
' et en écrivant les solutions sur la feuille Excel avec utilisation d'une  fonction  pour
calculer delta et une fonction pour ' résoudre une équation du 1er degré


Sub q4_equa2()
Dim a, b, c As Integer
[A1] = "a="
[A2] = "b="
[A3] = "c="
a = [B1]
b = [B2]
c = [B3]
If a = 0 Then
If b = 0 Then
If c = 0 Then
[H15] = "la solution est R : n'importe quelle valeur de x vérifie l'équation "
Else
[H15] = "il n'y a pas de solution"
End If
Else
' equation du premier degré
[E1] = "Equation degré 1 x="
[F1] = equa1(b, c)
End If
Else
' equation du 2ème degré
' calcul de delta
[C2] = "delta="
[D2] = delta(a, b, c)
If delta(a, b, c) < 0 Then
[H15] = "il n'y a pas de solution dans R"
Else
```

```
If delta(a, b, c) = 0 Then
'solution double
[E1] = "Solution Double x="
[F1] = -b / (2 * a)
Else
'2 solutions
[E1] = "X1="
[E2] = "X2="
[F1] = (-b - Sqr(delta(a, b, c))) / (2 * a)
[F2] = (-b + Sqr(delta(a, b, c))) / (2 * a)
End If
End If
End If
End Sub

Function delta(x, y, z As Integer) As Integer
delta = y * y - 4 * x * z
End Function

Function equa1(x, y As Integer) As String
If x = 0 Then
equa1 = "erreur"
Else
equa1 = -y / x
End If
End Function
```

Exercice2

```
' La fonction moyenne reçoit en entrée les notes des modules  et leurs coefficients,
' et retourne  comme résultat la moyenne
Function moyenne(note1 As Double, coef1 As Byte, note2 As Double, coef2 As Byte, note3 As
```

```
Double, coef3 As Byte) As Double
moyenne = (note1 * coef1 + note2 * coef2 + note3 * coef3) / (coef1 + coef2 + coef3)
End Function
' La fonction  decision reçoit en entrée le nombre d'absences nb_abs et la moyenne  moy
 ' et retourne  la décision  decision    sous forme de chaîne de caractères.
Function decision(nb_abs As Byte, moy As Double) As String
 If nb_abs >= 5 Then
    decision = "Exclu"
  Else
    If moy >= 10 Then
       decision = "Admis"
      Else
       If moy >= 9 Then
          decision = "Racheté"
         Else
          decision = "Ajourné"
       End If
    End If
 End If
End Function

'  La fonction Mention reçoit en entrée le nombre d'absences nb_abs  et la moyenne moy
' et retourne la mention sous forme de chaîne de caractères.
Function mention(nb_abs As Byte, moy As Double) As String
If nb_abs < 5 Then
```

```
 Select Case moy
   Case Is >= 16
      mention = "Très bien"
   Case 14 To 15.99
      mention = "Bien"
   Case 12 To 13.99
      mention = "Assez Bien"
   Case 11 To 11.99
      mention = "Passable"
   Case 9 To 11
      mention = "Sans mention"
   Case Else
      mention = "*"
  End Select
 Else
   mention = "*"
 End If
End Function
' La procédure Remplir_tableau() n'a pas de paramètres.  Elle appelle les fonctions
' Moyenne, Decision et Mention pour remplir le tableau Excel.
Sub remplir_tableau()
Dim i As Integer
Dim derniere_ligne As Integer
derniere_ligne = Range("A" & Rows.Count).End(xlUp).Row
i = 4
```

```
While i <= derniere_ligne
  Cells(i, 8) = moyenne(Cells(i, 4), Cells(2, 4), Cells(i, 5), Cells(2, 5), Cells(i, 6), Cells(2, 6))
  Cells(i, 9) = decision(Cells(i, 7), Cells(i, 8))
  Cells(i, 10) = mention(Cells(i, 7), Cells(i, 8))
  i = i + 1
Wend
End Sub
```

Bibliographie

[1]Mikael Bidault, Le Programmeur, Excel et VBA, Développez des macros compatibles avec les versions d'Excel 1997-2010), ISBN : 978-2-7440-4158-7, Copyright © 2010 Pearson Education France.

[2]Chelali Herbaji, La Gestion sous EXCEL et VBA, Eyrolles, 2012

[3]Daniel Jean David, Excel 2013 Programmation VBA, Eyrolles, 2014

[5] *https://www.mediaforma.com/vba-Excel-modele-objet-dExcel*

[6] *https://mrproof.blogspot.com/2010/12/le-modele-objet-dExcel-et.html*

[7] *https://www.lecompagnon.info//vba-Excel/*

[8]*https://docs.microsoft.com/fr-fr/office/vba/language/reference/user-interface-help/operator-precedence*

Annexe: ***Tableau récapitulatif des instructions algorithmiques et VBA.***

Algorithmique	**Visual Basic**	**Description**
Algorithme nomalgo;	sub nomalgo()	Début du programme
Finalgorithme.	End sub	Fin du programme
Constantes B ←5	Const B=5	B est une constante = 5
Variables A : Réel;	Dim A as Double	A ∈ $[-1{,}797.10^{308}, +1{,}797.10^{308}]$
Entier	Byte Integer Long	[0, 255] [-32768, +32767] [-2147483648, +2147483647]
Réel	Currency Single Double	$[-922{,}33*10^{12}, +922{,}33*10^{12}]$ $[-3{,}402*10^{38}, +3{,}402*10^{38}]$ $[-1{,}797.10^{308}, +1{,}797.10^{308}]$
Lire(A)	A=inputbox(" ")	Lire A dans une fenêtre
Ecrire(A)	Msgbox A	Ecrire A dans une fenêtre
←	=	Opérateur d'affectation
+, -, *, /	+, -, *, /	Opérateurs arithmétiques
= , ≠ , < , ≤ , > , ≥	= , <> , < , <= , > , >=	Les opérateurs de comparaison
Et , Ou , Non	And , Or , Not	Opérateurs logiques
x div y	x \ y	Résultat de la division entière
x mod y	x MOD y	Le reste de la division entière
Si condition	If condition	La 1ère forme de l'instruction conditionnelle
alors	Then	
Sinon	Else	
Finsi	Endif	
Si condition	If condition	La 2ème forme de l'instruction conditionnelle
Alors	Then	
Finsi	Endif	
Selon cas *variable*1 **Cas** $valeur_1$ **…** **Cas** $valeur_n$ **Cas autre** **Finselon**;	**Select case** *variable*1 **Case** $valeur_1$ **…** **Case** $valeur_n$ **Case else** **End select**	La 1ère forme de l'instruction de branchement multiple
Selon cas *variable*1 **Cas est** *op. comparaison valeur* **Cas** *valeur_début* **à** *valeur_fin* **…** **Cas autre** **…**	**Select case** *variable*1 **Case Is** *op. comparaison valeur* **Case** *valeur_début* **to** *valeur_fin* **Case else** **…** **End select**	La 2ème forme de l'instruction de branchement multiple

Finselon;		
Pour i←a Jusqu'à b pas c	For i=a To b Step c	La Boucle Pour
FinPour	Next i	
TantQue (Condition) Faire	While Condition	La boucle Tant que
FinTantque	Wend	
Répéter	Do	La Boucle Répéter
Jusqu'à (condition)	Loop Until condition	

Table des figures