L'ALGORITHMIQUE

I. Définir l'algorithme

Un algorithme[1] est le découpage d'un problème en une suite d'opérations élémentaires et qui va permettre de résoudre ce problème.

A. Caractériser l'algorithme

Un algorithme est caractérisé par cinq éléments :

- Un ensemble d'étapes à exécuter,
- Le contenu de chaque étape (opération(s) à exécuter),
- L'ordre impératif de succession des différentes étapes,
- L'existence éventuelle de conditions (ou expressions logiques) déterminant l'exécution ou la non-exécution de certaines étapes,
- Un début et une fin.

B. Notation algorithmique

Les actions ou opérations d'un algorithme sont décrites à l'aide de mots (verbes, locutions, ...) respectant des règles syntaxiques et de présentation qui en facilitent la compréhension. Toutes ces règles définissent la **notation algorithmique.**

Exemple : *algorithme relatif au changement d'une roue exprimé en langage naturel.*

Algo Changement de roue
Début
1. ouvrir le coffre
2. prendre la roue de secours
3. **Si** la roue de secours est crevée
4. **Alors** appeler un garagiste
Sinon
5. sortir le cric et la clé à boulons du coffre
6. dévisser légèrement les boulons avec la clé
7. **Répéter**
8. tourner la manivelle du cric
Jusqu'à ce que la roue crevée ne touche plus le sol
9. dévisser complètement les boulons
10. retirer complètement la roue endommagée
11. **Répéter**
12. tourner la manivelle du cric
Jusqu'à ce que la roue de secours puisse être placée
13. monter la roue de secours
14. visser les boulons
15 abaisser le véhicule avec le cric
16. visser fortement les boulons
17. Ranger le cric, la clé et la roue crevée
Fin si
18. fermer le coffre
Fin

[1] **Le Petit Robert : Algorithme** : Ensemble des règles opératoires propres à un calcul ou à un traitement informatique. - Calcul, enchaînement des actions nécessaires à l'accomplissement d'une tâche

Ce simple exemple permet de constater :

- ☞ que l'ordre des opérations est important. Il n'est guère possible d'intervertir les opérations 5 et 6 par exemple.
- ☞ qu'une ou plusieurs opérations peuvent être dépendantes de conditions (Si…Alors) ou bien répétées (Répéter…Jusqu'à).

Il faut compléter la définition d'un algorithme en précisant que les actions d'un algorithme s'articulent à l'aide de structures de base qui sont :

- ✓ la structure **séquentielle** (ensemble d'opérations à la suite les unes des autres) ;
- ✓ la structure **conditionnelle** (ensemble d'opérations soumises à une condition) ;
- ✓ la structure **répétitive** (ensemble d'opérations répétées un nombre fini de fois).

Ces trois structures constituent les briques de base de tout algorithme et permettent de résoudre n'importe quel problème, qu'il s'agisse de la simple addition de deux nombres ou du pilotage d'un module lunaire.

Un **programme** est la représentation informatique d'un algorithme dans un langage de programmation : les actions sont traduites en **instructions**.

II. Notion de constante et de variable

A. Objets

Un algorithme utilise des objets (objet constant ou variable).

Une **constante** est un objet qui ne peut pas être modifié dans l'algorithme.

Une **variable** est un objet appelé à subir des transformations au cours de l'algorithme.

Constante et variable se caractérisent par :

- Un **identificateur** ou un nom : appellation donnée à l'objet,
- Une **valeur** : c'est le contenu de l'objet. Cette valeur est susceptible d'évoluer si l'objet est une variable.
- Un **type** : domaine dans lequel l'objet puise sa valeur. Le type détermine les opérations que l'on peut appliquer à l'objet.

B. Types d'objet existants

Types d'objets	Exemples de valeurs possibles
BOOLEEN ou LOGIQUE	.VRAI./.FAUX.
DATE	4/4/2003 10 :10
NUMERIQUE : ENTIER ou REEL	 88 12,345
CHAINE DE CARACTERES	"Bonjour" "1774445555"

C. Convention syntaxique

Il faut nommer l'algorithme, le titre ou nom de l'algorithme résume l'objectif de l'algorithme.

Les mots clefs, constitutifs de la syntaxe de l'algorithme, sont indiqués en gras.

Tout objet doit être déclaré avant son utilisation à l'aide des mots clefs **Const** et **Var**.

Les lignes de l'algorithme sont **indentées** (décalées) afin d'en faciliter la lecture.

Convention syntaxique

```
Algo Calcul_prix_de_vente_HT
Const Coefficient_de_marge=2,5
Var Prix_achat_HT, Prix_de_vente_HT : réel
Début
        {ei[2] : Coefficient_de_marge=2,5}
            Afficher "Quel est le prix d'achat HT ?"
            Saisir Prix_achat_HT
            Prix_de_vente_HT := Prix_achat_HT * Coefficient_de_marge
            Afficher "Le prix de vente HT est :", Prix_de_vente_HT
        {ef[3] : Prix_achat_HT, Prix_de_vente_HT affiché}
Fin
```

III. Expressions et opérateurs

Les variables et constantes peuvent être combinées entre elles au travers d'opérateurs pour former des expressions. On distingue les expressions arithmétiques et les expressions logiques.

- MontantHt x TauxTva est une expression arithmétique permettant de calculer un montant T.T.C. à l'aide d'un opérateur arithmétique.
- Longueur > Largeur est une expression logique utilisant un opérateur relationnel permettant de déterminer si la valeur de la variable Longueur est supérieure à la valeur de la variable Largeur.

A. Expressions et opérateurs arithmétiques

Soit 20 la valeur de a et 2 la valeur de b :

Opérations	Opérateurs	Exemples	Résultats
Addition	+	a + b	22
Soustraction	-	a - b	18
Multiplication	*	a*b	40
Division	/	a/b	10
Elévation à la puissance	^	a ^ b	400

B. Expressions logiques

1. *Opérateurs relationnels*

Ils s'appliquent essentiellement aux objets de type entier, réel, caractère et chaîne de caractères. Ainsi, il est possible de comparer des données de même type entre elles pour savoir si elles sont égales, plus grandes ou plus petites. La comparaison de deux chaînes de caractères est également possible et s'effectue caractère par caractère, de gauche à droite. Le résultat est soit vrai (l'expression logique est vérifiée) soit faux (l'expression logique n'est pas vérifiée).

Opérations	Significations	Exemples	Résultats
=	égal	20=10*2	vrai
<>	différent de	"A" <> "G"	vrai
<	inférieur	11<8	vrai
<=	inférieur ou égal	20<=10*2	vrai
>	supérieur	8>11	faux
>=	supérieur ou égal	"au revoir" >= "bonjour"	faux

[2] état initial
[3] état final

2. *Opérateurs logiques*

Les expressions logiques peuvent être reliées à l'aide d'opérateurs logiques. Ci-dessous, deux principaux opérateurs logiques :

Opérateur ET logique

Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

Opérateur OU logique

Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

Exemples :

Expressions logiques	Résultats intermédiaires	Résultats
Vrai OU (Vrai ET Faux)	Vrai OU (Faux)	Vrai
(11 > 10) OU (20 > 18)	Vrai OU Vrai	Vrai
(8 >= 11) ET (88 > 68)	Faux ET Vrai	Faux
(11 = 11) ET ("Au revoir" < "Bonjour")	Vrai ET Vrai	Vrai

C. Expressions chaînes de caractères

L'opérateur & permet la concaténation de deux chaînes de caractères.

Exemples :

Expressions chaînes de caractères	Résultats
"Henri " & "IV"	Henri IV
"Nous sommes " & "le 20 Janvier 1998"	Nous sommes le 20 janvier 1998

IV. Les instructions

A. Instruction d'affectation

Affecter, c'est donner une valeur à une variable.

L'instruction d'affectation permet de valoriser une variable à partir d'une variable, d'une constante, d'un littéral, d'une expression arithmétique ou d'une expression logique.

Le symbole utilisé est := ou ←, l'affectation remplace la valeur de l'objet indiqué à gauche du caractère := ou ←

Exemple d'une suite d'instructions d'affectation :

Total := 1000	*ou*	Total ← 1000
PrixHT :=Total		PrixHT ←Total
TVA := Total x 0,196		TVA ← Total x 0,196
PrixTTC := PrixHT + TVA		PrixTTC ← PrixHT + TVA

Après exécution de l'ensemble de ces instructions d'affectation,

Les variables :	Contiennent les valeurs :
Total	1 000
PrixHT	1 000
TVA	196
PrixTTC	1 196

B. Instruction d'entrée-sortie

Le rôle des instructions d'entrée-sortie est d'assurer la prise en compte et la restitution d'informations à partir des organes d'entrée-sortie de l'ordinateur (clavier, écran, imprimante…).

On distingue principalement :

- **l'instruction d'acquisition** ou d'entrée autorisant la saisie de l'information à partir du clavier ;
- **les instructions de sortie** autorisant :
  - ✓ l'affichage des informations à l'écran,
  - ✓ l'impression des informations sur papier.

1. *La saisie de données*

L'instruction d'entrée Saisir autorise la saisie de données à partir du clavier. Un message permet de guider l'utilisateur. Cette instruction sert principalement à valoriser une ou plusieurs variables. Ainsi, l'utilisateur communique à l'ordinateur les valeurs des objets.

Exemple :

saisir "Veuillez indiquer la valeur du nombre : ", Nombre

saisir "Veuillez indiquer les paramètres a et b : ", A, B

Action : algorithme de base du fonctionnement de l'instruction Saisir

À la rencontre d'une instruction "saisir", le programme est interrompu.

- La chaîne de caractères est éventuellement affichée, pour guider l'utilisateur.
- Un curseur est affiché immédiatement après la chaîne (point clignotant).
- L'utilisateur entre au clavier la donnée.
- La saisie est terminée par l'appui sur la touche Envoi (Entrée).
- La(les) valeur(s) saisie(s) est(sont) affectée(s) à(aux) l'objet(s).
- Le déroulement du programme continue à l'instruction suivante.

2. *La restitution des résultats*

La restitution des résultats s'effectue à l'aide des instructions :

Afficher : affichage des données à l'écran

Imprimer : impression des données sur papier

Chacune des instructions peut indifféremment restituer le contenu d'une constante, variable, expression numérique ou expression logique. Des messages sous forme de chaînes de caractères permettent d'agrémenter la présentation des informations. Les différentes informations sont séparées par des virgules.

Exemples:

afficher *Total, Somme*

imprimer *"Le nom du client est : ", NomClient*

V. Les structures de base de l'algorithme

A. La structure séquentielle

Une séquence est constituée d'un ensemble d'actions à exécuter successivement sans exception, dans un ordre défini, du début jusqu'à la fin.

Début
"Action 1"
……
"Action 2"
Fin

B. La structure conditionnelle :

1. L'alternative

Lorsque le traitement à effectuer est fonction d'une ou plusieurs conditions, il est nécessaire d'utiliser une instruction conditionnelle.

La structure alternative partage une partie des actions à exécuter en deux ou plusieurs sous-ensembles dont l'exécution se fait de manière exclusive.

Syntaxe	Action
SI <expression logique> **ALORS** <instruction si vrai> **SINON** <instruction si faux> **FIN SI**	La sémantique de la structure alternative est la suivante : Quand <expression logique> est vraie, <instructions si vrai> est exécutée et <instructions si faux> est ignorée. Par contre, quand <expression logique> est fausse, on exécute <instructions si faux> et on ignore <instructions si vrai>.

2. Le choix

Syntaxe	Action
SELON **CAS 1 :** <Action1> **CAS 2 :** <Action2> …. **CAS N :** <ActionN> **AUTRES CAS** :<ActionN+1> **FIN SELON**	La sémantique de la structure de choix est la suivante : Quand <expression logique> du CAS 1 est vraie, <Action 1> est exécutée et les cas 2 à N+1 sont ignorés. Par contre, quand <expression logique> du CAS 1 est fausse, on passe au CAS 2. On teste l'expression logique comme précédemment et ainsi de suite jusqu'à ce qu'une expression soit vraie.

La structure de choix offre une **meilleure lisibilité** que l'expression par des SI…ALORS…SINON… imbriqués.

C. La structure répétitive ou itérative

La structure répétitive permet de répéter un ensemble d'instructions jusqu'à ce qu'une condition soit réalisée.

1. *Tant que…Fin tant que*

Syntaxe	Action
Tant que <expression logique> **faire** <instructions> **Fin tant que**	On teste d'abord si <expression logique> est vraie. Dans ce cas, on exécute <instructions> puis on boucle de nouveau sur le test de <expression logique>. Par contre si <expression logique> est fausse, la boucle se termine et le programme poursuit son exécution après "fin tant que".

La boucle peut ne jamais être exécutée si l'"expression logique" n'est jamais vérifiée.

2. *Répéter…Jusqu'à*

Syntaxe	Action
Répéter <instructions> **Jusqu'à** <expression logique>	1. Le bloc <instructions> est exécuté. 2. L'<expression logique> est testée. 3. Dans le cas où elle est égale à faux, on recommence au point 1. 4. Dans le cas où elle est égale à vrai, le programme poursuit son exécution après l'instruction "Jusqu'à".

3. *Pour…Fin pour*

Cette structure permet de répéter une action un nombre connu de fois.

Syntaxe	Action
Pour <variable> de <expression début> à <expression fin> pas de [<incrément>] <instructions> **Fin Pour**	1. <variable> est une variable assimilée à un compteur qui est automatiquement augmenté ou diminué en fonction de <incrément>. 2. Au début, <variable> prend la valeur de <expression début>. 3. Le bloc d'instructions est exécuté jusqu'à ce que <expression début> prenne la valeur de <expression fin>. 4. La valeur <variable> évolue de <expression début> jusqu'à <expression fin> en fonction de <incrément> qui par défaut est égale à 1.

Dans cet algorithme, le pas de progression de l'indice *k* est implicitement 1.

D. Les structures modulaires

L'algorithmique modulaire consiste à découper un programme en modules généraux qui sont développés "à part". On a donc un algorithme général qui va appeler les différents modules. Cet algorithme général va être allégé et ne présenter que la structure logique globale du programme sans être encombré par des instructions de gestion d'affichage, de contrôle des saisies…

Cela va permettre une **meilleure lisibilité, une facilité de maintenance** (il est plus facile de repérer les dysfonctionnements dans un module plutôt que dans un programme général) et une certaine **réutilisabilité** des programmes (d'où des gains de temps).

L'usage des fonctions et procédures représente la base de la programmation structurée.

1. *Un module*

C'est une procédure (ou une fonction) réutilisable. Un module peut-être appelé par autant de programmes que nécessaire. Cela évite de réécrire plusieurs fois le même programme.

2. *Une procédure*

Une procédure représente une séquence d'instructions.

Une procédure ne retourne jamais de résultat. Toutefois, elle peut admettre des arguments en "entrée" et éventuellement les modifier en "sortie".

C'est un sous programme à qui on fournit éventuellement des paramètres :

Exemple :

AFFICHER_MESSAGE ("insérer une disquette dans le lecteur").

3. *Une fonction*

Une fonction est un sous-programme qui donne un résultat.

Exemple :

Cela pourrait être par exemple une fonction définie sous Excel du type Calc_frais qui calculerait le montant de l'indemnité kilométrique en fonction du nombre de kilomètres.(Cf. macro sous Excel).

Lors de son appel (demande d'exécution), la fonction est évaluée à partir des arguments effectifs qui lui sont fournis et le résultat vient se substituer à son nom dans l'expression appelante.

Appel de la fonction dans l'action appelante	**Déclaration (description) de la fonction**
... Ecrire *NomF*(liste des arguments effectifs) ou ... NomVar ← *NomF*(liste des arguments effectifs)	**Fonction** *NomF*(liste des arguments formels) : *TypeFonction* **Données** *Action_1* ... *Action_N* **Retourner** *expression* **Fin fonction**

Le mot réservé **Retourner** indique comment obtenir la valeur qui est renvoyée à l'action appelante lorsque la fonction est exécutée.

VI. Les méthodes de production d'algorithmes

A. Validation et jeu d'essai

Une fois le programme écrit, il convient de s'assurer qu'il fonctionne correctement.

Pour cela, il est nécessaire de réaliser un jeu d'essai.

Dans ce jeu d'essai, on passe en revue tous les cas possibles et on indique les résultats attendus.

Exemple de jeu d'essai :

Pour l'algorithme : Remboursement frais 3:

Cas	**1**	**2**	**3**
Saisir le nom	Dupont	Durant	Martin
Le nb de km	4 500	5 200	10 900
Valeurs	1 350	1 554	3 039
Résultats attendus :	Les frais de déplacement de Dupont s'élèvent à : 1 350.	Les frais de déplacement de Durant s'élèvent à : 1 554.	Les frais de déplacement de Martin s'élèvent à : 3 039.
Test			

B. Contrôle et vraisemblance

Un utilisateur va contrôler que le programme fonctionne correctement. Muni du jeu d'essai, il va tester le programme.

L'utilisateur lance le programme et simule tous les cas prévus dans le jeu d'essai. Pour chaque cas, il vérifie que le résultat renvoyé par le programme est bien conforme à celui qui était inscrit dans le jeu d'essai. Dans chaque cas, l'utilisateur testeur indique sur le jeu d'essai le résultat du test.

Pour les programmes complexes, on peut prévoir plusieurs jeux d'essai, plusieurs utilisateurs testeurs (en particulier pour les programmes devant fonctionner simultanément pour plusieurs utilisateurs)…

Le jeu d'essai testé est remis au développeur qui effectuera éventuellement les corrections.

C. Documentation et maintenance

Dès lors qu'un programme est complexe, il convient de l'accompagner d'une documentation.

On distingue trois catégories de documentation.

1. La documentation de spécification

C'est un document qui décrit le comportement attendu du programme. Il doit être lisible et compréhensible par un utilisateur. Il est élaboré avant la phase de développement par un informaticien (interviews) et approuvé par l'utilisateur demandeur.

L'objectif de cette documentation est double :

- Elle permet de définir ce que doit faire le programme avant de l'écrire.
- Elle permet de définir le jeu d'essai.

2. La documentation technique

C'est un document qui décrit l'architecture du programme (algorithmes, fonctions, procédures). Il est écrit par les informaticiens pour les informaticiens.

L'objectif de cette documentation est de pouvoir faciliter le transfert de compétence entre le créateur du programme et un autre informaticien (et donc d'assurer la maintenance).

3. Le manuel d'utilisation

C'est un document qui a pour but de permettre à un utilisateur qui ne connaît pas l'application d'apprendre à s'en servir. Il est établi par l'informaticien. Il peut être accompagné d'une aide en ligne, de supports de formation.

ECRIRE DES ALGORITHMES

La société Desplact désire calculer le montant mensuel du remboursement des frais de déplacement des représentants. Ce montant est calculé sur la base d'une indemnité kilométrique de 0,3€.

Les informations suivantes sont disponibles pour le mois de mars :

Nom du représentant	Total des km parcourus
Durant	5 200
Dupont	4 500
Martin	10 900
Henri	3 575
Quartz	11 700

I.Schéma séquentiel :

```
Algo Remboursement frais 1
  Const IndemKm=0,3
  Var NomRep chaîne
  Var TotalKm,frais : réel
  Début
    {ei : IndemKm=0,3}
    Saisir « Saisir le nom du représentant : »,NomRep
    Saisir « Combien de Km »&NomRep& « a-t-il parcouru ? »,TotalKm
    Frais ← TotalKm*IndemKm
    Afficher "Les frais de déplacement de  » &NomRep& « s'élèvent à : » &frais& « .
    {ef : Frais affichés }
  Fin
```

II.Schéma conditionnel :

A. L'alternative

Le montant du remboursement est en fait calculé en fonction du nombre de kilomètres parcourus selon le barème dégressif ci-dessous :

Jusqu'à 10 000 km, l'indemnité kilométrique de base est de 0,3€ ;

Au-delà de 10 000 km, l'indemnité kilométrique de base est minorée de 10%

```
Algo Remboursement frais 2
  Const IndemKm=0,3 : réel
  Const Tr1=10 000 : entier
  Var NomRep chaîne
  Var TotalKm,frais : réel
  Début
    {ei : IndemKm=0,3 ;Tr1=10000}
    Saisir « Saisir le nom du représentant : »,NomRep
    Saisir « Combien de Km »&NomRep& « a-t-il parcouru ? »,TotalKm
        Si TotalKm<=Tr1 Alors
            Frais ← TotalKm*IndemKm
            Sinon
            Frais ← Tr1* IndemKm + (TotalKm-Tr1)*IndemKm*0,9
        Finsi
    Afficher "Les frais de déplacement de  » &NomRep& « s'élèvent à : » &frais& « . »
    {ef : Frais affichés }
  Fin
```

B. Le choix

Le montant du remboursement est en fait calculé en fonction du nombre de kilomètres parcourus selon le barème dégressif ci-dessous :

Jusqu'à 5 000 km, l'indemnité kilométrique de base est de 0,3€ ;

De 5 000 à 10 000 km, l'indemnité kilométrique de base est minorée de 10% ;

Au-delà de 10 000 km, l'indemnité kilométrique de base est minorée de 30%.

```
Algo Remboursement frais 3
  Const IndemKm=0,3 : réel
  Const Tr1=5 000, Tr2=10 000 : entier
  Var NomRep chaîne
  Var TotalKm,frais : réel
  Début
   {ei : IndemKm=0,3 ;Tr1=5000 ; Tr2=10000}
   Saisir « Saisir le nom du représentant : »,NomRep
   Saisir « Combien de Km »&NomRep& « a-t-il parcouru ? »,TotalKm
          Selon
                TotalKm<=Tr1 : Frais ← TotalKm*IndemKm
                TotalKm<=Tr2 : Frais ← Tr1* IndemKm + (TotalKm-Tr1)*IndemKm*0,9
                Autres cas : Frais ← Tr1* IndemKm + (Tr2-Tr1)*IndemKm*0,9 + (TotalKm-
                Tr2)*IndemKm*0,7
          Finselon
   Afficher « Les frais de déplacement de  » &NomRep& « s'élèvent à : » &Frais& « . »
   {ef : Frais affichés}
  Fin
```

III.Le schéma itératif

Les frais à rembourser aux représentants sont calculés chaque mois et M. Desplact désire connaître leur montant total.

A. Tant que

```
Algo Remboursement frais 4
  Const IndemKm=0,3 : réel
  Const Tr1=5 000, Tr2=10 000 : entier
  Var NomRep chaîne
  Var TotalKm,frais,Montant_Frais : réel
  Début
   {ei : IndemKm=0,3 ;Tr1=5000 ; Tr2=10000}
   MontantFrais ← 0
   Saisir "Saisir le nom du représentant (ou F si terminé) :" ,NomRep
          TANT QUE NomRep<> « F »
                Saisir « Combien de Km »&NomRep& « a-t-il parcouru ? »,TotalKm
                   Selon
                         TotalKm<=Tr1 : Frais ← TotalKm*IndemKm
                         TotalKm<=Tr2 : Frais ← Tr1* IndemKm + (TotalKm-Tr1)*IndemKm*0,9
                         Autres cas : Frais ← Tr1* IndemKm + (Tr2-Tr1)*IndemKm*0,9 + (TotalKm-
                         Tr2)*IndemKm*0,7
                   Finselon
                Montant_Frais ← Montant_Frais+Frais
                Afficher « Les frais de déplacement de  » &NomRep& « s'élèvent à : » &Frais& « . »
                Saisir "Saisir le nom du représentant (ou F si terminé) :" ,NomRep
```

```
            FIN TANT QUE
        Afficher « Le total des frais de déplacement du mois s'élèvent à : » & Montant_Frais & « . »
    Fin
```

B. Répéter

```
Algo Remboursement frais 5
    Const IndemKm=0,3 : réel
    Const Tr1=5 000, Tr2=10 000 : entier
    Var NomRep chaîne
    Var TotalKm,frais,Montant_Frais : réel
    Début
      {ei : IndemKm=0,3 ;Tr1=5000 ; Tr2=10000}
      MontantFrais ← 0
      Saisir "Saisir le nom du représentant :" ,NomRep
            Répéter
                  Saisir « Combien de Km »&NomRep& « a-t-il parcouru dans le mois ? »,TotalKm
                      Selon
                            TotalKm<=Tr1 : Frais ← TotalKm*IndemKm
                            TotalKm<=Tr2 : Frais ← Tr1* IndemKm + (TotalKm-Tr1)*IndemKm*0,9
                            Autres cas : Frais ← Tr1* IndemKm + (Tr2-Tr1)*IndemKm*0,9 + (TotalKm-
                            Tr2)*IndemKm*0,7
                      Finselon
                  Montant_Frais ← Montant_Frais+Frais
                  Afficher « Les frais de déplacement de  » &NomRep& « s'élèvent à : » &Frais& « . »
                  Afficher "Saisir le nom du représentant (ou F si terminé) :"
                  Saisir NomRep
            Jusqu'à Nomrep= « F »
      Afficher « Le total des frais de déplacement du mois s'élèvent à : » & Montant_Frais & « . »
    Fin
```

IV. Fonction

Cela pourrait être par exemple une fonction définie sous Excel du type Calc_frais qui calculerait le montant de l'indemnité kilométrique en fonction du nombre de kilomètres.(Cf. macro sous Excel).

A. Fonction

```
Fonction Calc_frais(TotalKm : réel) : réel
Const Tr1=5 000, Tr2=10 000 : entier
      Selon
            TotalKm<=Tr1 : Frais ← TotalKm*IndemKm
            TotalKm<=Tr2 : Frais ← Tr1* IndemKm + (TotalKm-Tr1)*IndemKm*0,9
            Autres cas : Frais ← Tr1* IndemKm + (Tr2-Tr1)*IndemKm*0,9 + (TotalKm-
            Tr2)*IndemKm*0,7
      Finselon
Fin Fonction
```

B. Appel de la fonction

```
Saisir « Combien de Km avez-vous parcourus dans le mois ? »,TotalKm
Afficher « Les frais de déplacement s'élèvent à » & Calc_frais(TotalKm)
```

TABLE DES MATIÈRES