Support de cours d'algorithme

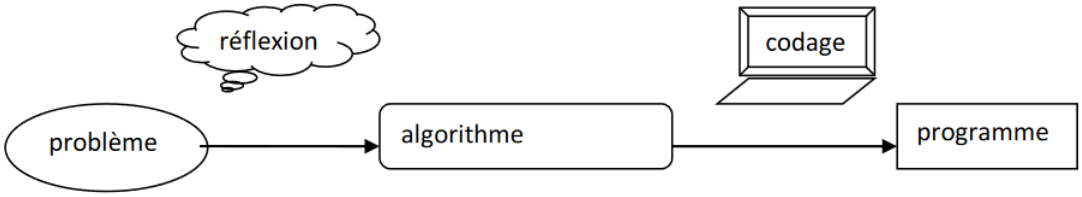
Année académique 2022-2023

Enseignant : Mamadou Moustapha DIAGNE

Chapitre 1 Généralités

L'algorithmique est un terme d'origine arabe, hommage à Al Khawarizmi (780-850) auteur d'un ouvrage décrivant des méthodes de calculs algébriques. Un algorithme est une méthode de résolution de problème énoncée sous la forme d'une série d'opérations à effectuer. La mise en œuvre de l'algorithme consiste en l'écriture de ses opérations dans un langage de programmation et constitue alors la brique de base d'un programme informatique.

1- Démarche d'analyse d'un problème et conception d'un algorithme et d'un programme: De manière schématique, on peut représenter l'activité de développement pour résoudre un problème donné sous la forme suivante :



La réalisation d'un programme exécutable par un ordinateur, nécessite le suivi d'une démarche constituée d'un ensemble d'étapes.

Première étape : Position du problème :

Le problème est souvent posé par un demandeur de solution informatique. C'est le cas du pharmacien, d'un élève, d'un banquier,...

Parfois, ces demandeurs ne savent plus exprimer leurs besoins avec précision. L'objectif de cette étape est bien formuler le problème pour pouvoir le résoudre correctement.

Deuxième étape : Spécification et analyse des problèmes : L'objectif de cette étape est bien comprendre l'énoncé du problème,

déterminer les formules de calculs, les règles de gestion,... L'analyse des problèmes s'intéresse aux éléments suivants :

- Les résultats souhaités (sorties) ;
- Les traitements (actions réalisés pour atteindre le résultat) ;
- Les données nécessaires aux traitements (entrées).

Troisième étape : Ecriture de l'algorithme.

Après avoir terminé l'analyse, il faut mettre les instructions dans leur ordre logique d'exécution. On obtient un algorithme.
Quatrième étape : Ecriture du programme.

Une fois l'algorithme du problème établit, on doit penser à son exécution par l'ordinateur. Il faut traduire l'algorithme à l'aide d'un langage de programmation.

Cinquième étape : Exécutions et test du programme.

Une fois compilé ou interprété, un programme doit être testé pour s'assurer de son fonctionnement et qu'il répond aux besoins exprimés par l'utilisateur.

Un programme est testé par un jeu de test (des valeurs différentes de données).

2-Concept d’un algorithme

Définition :

Un algorithme est une suite finie d’opérations élémentaires, à appliquer dans un ordre déterminé, à des données. Sa réalisation permet de résoudre un problème donné.

Exemples : suivre une recette de cuisine, suivre un plan, faire une division euclidienne à la main sont des exemples d’algorithme.
Remarques :

1.Un algorithme doit être lisible de tous. Son intérêt, c’est d’être codé dans un langage informatique afin qu’une machine (ordinateur, calculatrice, etc.) puisse l’exécuter rapidement et efficacement.

2.Les trois phases d’un algorithme sont, dans l’ordre :

- l’entrée des données ;
- le traitement des données ;
- la sortie des résultats.

3-Structure d’un programme :

3-1-Structure d’un algorithme :

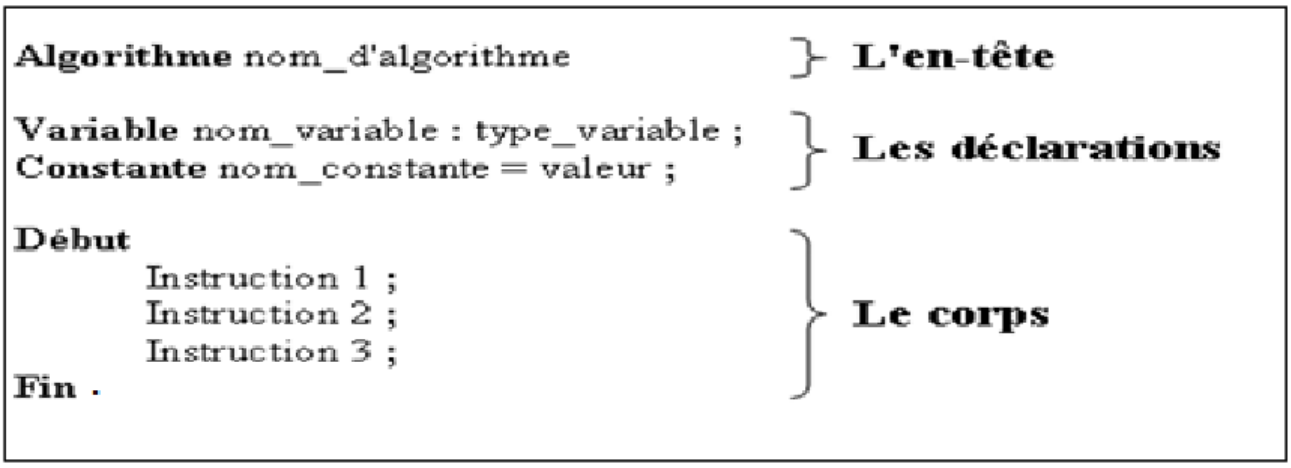
L'algorithme doit avoir une structure bien définie. Cette structure doit comporter :

-L'en-tête qui comprend le nom de l'algorithme pour identifier l'algorithme.

-Les déclarations des variables et des constantes.

-Le corps de l'algorithme qui contient les instructions.

Toutes les instructions doivent situer entre le mot Début et le mot Fin, et chaque instruction doit comporter un point-virgule à la fin.



4-strusture de données

Un algorithme ou un programme qui manipule des informations : les données. On distingue deux types de données (espèce de gestion des données) : les variables et les constantes. Comme leur nom l'indique, les "variables" sont des espaces qui stockes des données qui pourront varier (dont la valeur pourra changer) durant l'exécution du programme. A l'inverse, les "constantes" ont une valeur constante tout au long de l'exécution du programme.

4.1. Les variables et les constantes

Définition et caractéristiques

- Une VARIABLE est une donnée (emplacement) stockée dans la mémoire de la calculatrice ou de l'ordinateur. Elle est repérée par un identificateur (nom de la variable constitué de lettres et/ou de chiffres, sans espace) et contient une valeur dont le type (nature de la variable) peut être un entier, un réel, un booléen, un caractère, une chaîne de caractères...

– son type : une variable est utilisée pour représenter des données qui sont manipulées par le programme. Un type est utilisé pour caractériser l'ensemble des valeurs qu'une variable peut prendre.
Sa valeur : La variable contient une information qui peut varier au cours de l'exécution d'un programme. C'est cette information que l'on appelle valeur de la variable. La valeur d'une variable doit

correspondre au type de la variable. Ainsi, une variable quantité de type entier pourra prendre successivement les valeurs de 10, 25 et 3. La valeur d'une variable n'existe que lorsque le programme est exécuté. Les autres informations (nom, rôle et type) sont définies lors de la conception du programme, pendant la construction de l'algorithme. Le rôle, le nom et le type sont des informations statiques qui doivent être précisées lors de la déclaration de la variable. En revanche, la valeur est une information dynamique qui changera au cours de l'exécution du programme.
Remarque : Le nom d'une variable doit être significatif : il doit suggérer, si possible sans ambiguïté, la donnée représentée par cette variable

•Une CONSTANTE, comme une variable, peut représenter un chiffre, un nombre, un caractère, une chaîne de caractères, un booléen. Toutefois, contrairement à une variable dont la valeur peut être modifiée au cours de l'exécution de l'algorithme, la valeur d'une constante ne varie pas.

Les variables et les constantes sont définies dans la partie déclarative par deux caractéristiques essentielles :
• L'identificateur : c'est le nom de la variable ou de la constante, il est composé de lettres et de chiffres sans espaces.
• Le type : il définit la nature de la variable ou de la constante (entier, réel, caractère,...).

Remarques : Ne pas confondre la variable et son identificateur. En effet, la variable possède une valeur (son contenu) et une adresse (emplacement dans la mémoire où est stockée la valeur). L'identificateur n'est que le nom de la variable, c'est-à-dire un constituant de cette variable.

Le type d'une variable détermine l'ensemble des valeurs qu'elle peut prendre et les opérations réalisables qu'elle peut subir. L'utilisation d'une variable doit être précédée de sa déclaration.

La syntaxe pour déclarer une variable est la suivante :

Var identificateur de la variable : type de la variable

Exemple :
Variables
Prix : Réel

Nombre_etudiant : Entier

Nombre_groupe : Entier

Nom_section : Caractère

Nom_etudiant : Chaîne de caractères

La syntaxe pour déclarer une constante est la suivante :

const identificateur de la constante = valeur

Exemple :

Constante type Pi = 3.14, Ln2 = 2.93

Si la valeur de la variable peut changer au cours du déroulement de l'algorithme, en revanche son type est figé lors de déclaration.

CONVENTIONS DE NOMMAGE

Le nom d'un algorithme, d'une variable ou d'une constante doit respecter les règles suivantes

- commencer par une lettre ;
  Exemple1 : d1 ou D1 et non ~~1D~~ ;
- Doit être constitué uniquement de lettres, de chiffres et du soulignement (éviter les caractères de ponctuation et les espaces),
  Exemple2 : SM2013, USTHB et non ~~SM 2013~~, U S T H B.3.
- ne comporter ni caractère spécial (comme l'espace) ni ponctuation ;

- ne pas être un mot du langage algorithmique (comme « algorithme », « début », « fin», « variable », « non », « ou », « et », « si », « sinon », « pour »...)

4.2. Les types de données de bases

Les variables et les constantes peuvent avoir cinq types de base :

a) Type entier : un type numérique qui représente l'ensemble des entiers naturels et relatifs, tels que : 0, 45, -10,...

Mot clé : entier(integer)

b) Type réel : un autre type numérique qui représente les nombres réels, tels que : 0.5, -3.67, $1.5e^{+5}$,...

Mot clé : réel(real)

c) Type caractère : représente tous les caractères alphanumériques tels que :

′a′, ′B′, ′*′, ′9′, ′@′, ′ ′,...

Mot clé : car(char)

d) Type chaînes de caractères : concerne des chaines de caractères tels que des mots ou des phrases : ″informatique″, ″la section B″,...

Mot clé : chaîne(string)

e) Type booléen : ce type ne peut prendre que deux états : vrai ou faux

Mot clé : booléen (boolean)

. Les opérations sont Et, Ou et Non qui sont définies par la table de vérité suivante :

A	B	A Et B	A Ou B	Non A
VRAI	VRAI	VRAI	VRAI	FAUX
VRAI	FAUX	FAUX	VRAI	FAUX
FAUX	VRAI	FAUX	VRAI	VRAI
FAUX	FAUX	FAUX	FAUX	VRAI

Non (A Et B) = (Non A) Ou (Non B)

Non (A Ou B) = (Non A) Et (Non B)

Non (Non A) = A

5-Les opérateurs :
Un OPERATEUR est un outil qui permet d'agir sur une variable ou d'effectuer des calculs. Il existe plusieurs types d'opérateurs :

- L'opérateur d'affectation, représenté par le symbole «← » (:=), qui confère une valeur à une variable ou à une constante. (affectation de la valeur à la variable (ou à la constante)).

Syntaxe
identificateur_1← identificateur_2 ;(Affecte à la variable 1 le contenu de la variable 2)
identificateur_1← expression ; (Affecte à la variable 1 le résultat de l'expression)
Exemples où A, B et C sont de type real , i, j de type integer et S de type string (chaîne de caractères)
A← B ; { assigne à A la valeur de B }
A ←i ; { assigne à A la valeur de i }
i← A ; { donne une erreur de compilation : on ne peut assigner un real à un integer }
A← i/j ; { assigne à A le quotient (de type real) de i par j }
i← i+1 ; { i est incrémenté }
i← j div 10 { assigne à i le quotient (de type integer) de j par 10 (division entiere) }
S← S+'.' { ajoute le caractère '. ' à la fin de la chaîne S }

Remarques :
-Ne pas confondre ' = ' (affectation) avec ' == ' (comparaison des variables dans un test)
- Ne pas inverser les identificateurs ! ' A = B ' et ' B = A ' donnent des résultats différents

- Les opérateurs arithmétiques qui permettent d'effectuer des opérations arithmétiques entre opérandes numériques :

+	addition
-	soustraction
*	multiplication
/	division
mod	modulo
^	Puissance(en Pascal il n y a pas de puissance)
div	Division entiére

10 Mod 3 (le reste de la division entière de 10 par 3)

10Div3 (le quotient de la division entière de 10 par 3)

1Div2 (le quotient de la division entière de 1 par 2

- Les Opérateurs relationnels :

>	supérieur
<	inférieur
>=	supérieur ou égal
=<	inférieur ou égal
=	égal
≠ (< >)	différent

• Les opérateurs logiques qui combinent des opérandes booléennes pour former des expressions logiques plus complexes :
o Opérateur unaire : «non » (négation) (not)
o Opérateurs binaires : « et» (conjonction) (and), «ou » (disjonction) (or)
• L'opérateur de concaténation qui permet de créer une chaîne de caractères à partir de deux chaînes de caractère en les mettant bout à bout. « + »
Remarque : Les opérateurs dépendent du type de la constante ou de la variable :
• Opérateurs sur les entiers et les réels : addition, soustraction, multiplication, division, division entière, puissance, comparaisons, modulo (reste d'une division entière)
• Opérateurs sur les booléens : comparaisons, négation, conjonction, disjonction
• Opérateurs sur les caractères : comparaisons
• Opérateurs sur les chaînes de caractères : comparaisons, concaténation

• Priorité des opérateurs : A chaque opérateur est associée une priorité. Lors de l'évaluation d'une expression, la priorité de chaque opérateur permet de définir l'ordre d'exécution des différentes opérations. Aussi, pour lever toute ambiguïté ou pour modifier l'ordre d'exécution, on peut utiliser des parenthèses.
• Ordre de priorité décroissante des opérateurs arithmétiques et de concaténation :
▪ «^» (élévation à la puissance)
▪ «* », « /» et « div»
▪ « modulo»
▪ «+ » et «-»
▪ «+» (concaténation)
• Ordre de priorité décroissante des opérateurs logiques :
▪ «not » (non)
▪ « and» (et)
▪ « or» (ou)

Les commentaires

// Commentaire sur une ligne

// Commentaire

/* Commentaire */

/* Commentaire

sur
plusieurs
lignes */

Chapitre 2 Les instructions de base

Un algorithme se compose d'un certain nombre d'instructions. Ces dernières sont classées par catégories. Nous étudierons tout d'abord les instructions standard traitant l'information : l'affectation, la lecture et l'écriture. Ensuite, celles définissant l'ordre d'exécution d'un programme : la séquence, le choix et la boucle.

2.1 Instruction d'affectation

L'affectation est une instruction qui permet d'attribuer une valeur à une variable ou à un identificateur de fonction afin d'en renvoyer le résultat. Cette valeur retournée, doit être du même type que la variable. La syntaxe est la suivante :

<Variable> / <identificateur de fonction> = (Expression)
Ou
<variable> / <identificateur de fonction>:= (Expression)
Exemple : A = A+1

Delta = B*B –4 *A*C

2.2 Instructions d'Entrées

L'instruction d'entrée **Saisir** : elle offre à l'utilisateur la possibilité d'entrer plusieurs valeurs (données) dans des variables. La syntaxe générale est la suivante :

SAISIR (<Nom Var1>, <Nom Var2>,... <Nom VarN>

2.3 Instruction de Sorties

Nous avons deux types d'instructions de sortie : Afficher le texte pour annoncer à l'utilisateur de saisir une valeur.la syntaxe est la suivante :

afficher (``Texte``)
afficherligne (``Texte``)
L'instruction Afficher qui prend en charge une ou plusieurs variables et permet d'afficher un ou plusieurs résultats. La syntaxe est la suivante :

Afficher (``Texte``, Nom Var)
Afficher (``Texte``,Nom Var1, . . ., Nom Var2)
La fonction affichageligne permet l'affichage et le retour du curseur à la ligne.

Chapitre 3 Les instructions conditionnelles

Les instructions conditionnelles permettent de modifier l'ordre de la séquence d'un algorithme. Dans une séquence, les instructions sont exécutées les unes à la suite des autres sans interruption. Nous disposons de quatre instructions conditionnelles.

3.1 Instruction conditionnelle simple

La structure conditionnelle simple correspond à un choix unique.
Syntaxe générale :
SI<Condition> ALORS
Instruction
FINSI

3.2 Instruction conditionnelle alternative

La structure alternative correspond à un choix entre deux possibilités. Suivant la valeur issue de la condition spécifiée dans l'instruction, l'ordinateur exécute une suite d'instructions A ou une suite d'instructions B. En aucun cas, l'ordinateur n'exécute à la fois les instructions A et B.

SI<Condition> ALORS
Instruction A
FINSI
SINON
Instruction B

3.3 Instruction conditionnelle alternative Imbriquée

SI<Condition1> ALORS
Instruction A
FINSI
SINON SI <Condition2> ALORS
Instruction B
FINSI
SINON SI <Condition3> ALORS
Instruction C
FINSI
SINON SI <condition4> ALORS
Instruction D

FINSI
SINON
Instruction E
FINSI

3.4 Instruction conditionnelle choix Multiple

L'instruction se présente comme une alternative ; en effet, elle n'offre que deux choix possibles dépendant de la valeur d'une condition (valeur vraie ou fausse). En algorithme, nous disposons aussi d'une instruction permettant d'effectuer un choix entre plusieurs décisions.

L'instruction SELON ... DE

Syntaxe Générale

SELON <Expression> DE
Cas<Valeur 1> : Instruction A
Cas <Valeur 2> : Instruction B
Cas <Valeur 3> : Instruction C
Cas<Valeur 4> :
Cas <Valeur5 > : Instruction D
Defaut : InstructionE
FINSELON

Instruction peut être : soit une instruction, soit un bloc d'instructions. Les comparaisons sont effectuées au fur et à mesure en commençant par la première ; dès qu'une comparaison s'avère vraie, les instructions correspondantes sont exécutées et le programme se branche à l'instruction qui suit FINSELON (les autres comparaisons ne sont donc pas effectuées) ; si aucune comparaison ne s'avère vraie, les instructions correspondant à Défaut sont effectuées.

Chapitre 4 Instructions répétitives

Dans un programme plusieurs instructions peuvent se répéter ; il est alors plus intéressant d'écrire les instructions une seule fois et de les exécuter plusieurs fois. Cette action est ce que l'on appelle BOUCLE. En algorithmique, nous possédons trois Boucles.

4.1 les instructions répétitives conditionnelles

4.1.1 la boucle TANTQUE

C'est la boucle la plus utilisée en informatique. Son fonctionnement est simple. Elle exécute toutes les instructions comprises entre les mots réservés TANTQUE et FINTANTQUE tant que la condition de départ reste vérifiée.

La syntaxe générale est la suivante :

Compteur=valeur_initiale

TANTQUE <Condition> FAIRE
Action(s)
Compteur=Compteur+1

FINTANTQUE
L'ordinateur commence par vérifier si la condition est vraie. Si c'est le cas, il exécute les instructions de la boucle. Si ce n'est pas le cas les instructions suivant le FINTANTQUE sont exécutées.

L'incrémentation de la boucle passe par la variable Compteur, cette variable doit être initialise avant le début de la boucle. Dans certains cas de figure la variable Condition peut prendre la place de celle du compteur.

NB : Pour éviter une boucle infinie, il faut modifier la variable du test à l'intérieur de la boucle.

4.1.2 la boucle REPETER …JUSQU'A

Cette boucle permet d'exécuter les instructions comprises entre REPETER et JUSQUA jusqu'à ce que la condition du JUSQUA soit vérifiée.

Syntaxe générale :

Compteur=valeur_initiale

REPETER
Instruction 1
Instruction 2
Instruction N
Compteur=Compteur+1
JUSQUA<Condition>
Les instructions allant de 1à N vont s'exécuter jusqu'à ce que <Condition> soit vérifiée. Mais il faut noter que cette boucle exécute au moins une fois ces instructions avant de tester la condition. L'incrémentation de la boucle passe par la variable Compteur, cette variable doit être initialise avant le début de la boucle. Dans certains cas de figure la variable Condition peut prendre la place de celle du compteur.

4.2 l'instruction répétitive inconditionnelle

4.2.1 La boucle POUR

Cette boucle permet d'exécuter un certain nombre de fois une suite d'instructions.

Syntaxe générale :

POUR<Nom_Var>=<Borne minimale>à<Borne maximale> FAIRE
<Action>
FINPOUR
L'ordinateur exécute l'instruction<Action>Bornemaximale–Borneminimale+1 fois.

Chapitre 5 les Tableaux

Un type tableau permet de définir des données composées d'un nombre fixe d'éléments ayant tous le même type.

5.1 Tableaux à une dimension

5.1.1 définition

Syntaxe Générale :

NomDuTableau: Tableau [Min .. Max] de Type avec (min >0)

Ou

Définition d'un tableau

Tab1 : Tableau [1...10] de Entier

Tab2: Tableau[1...8] de réel

Tab3 : Tableau[1…7] de caractère

Tab1

5	8	99	0	5	33	22	11	77	88

Tab2

5.4	3.2	8.7	9.0	10.7	11.2	19.0	18.2

Tab3

K	L	N	F	C	H	I

Tab1 est un vecteur de 1 à 10 éléments de type entier. Tab2 est un tableau de dimension 1 contenant des réels et Tab3 contient des caractères.

Pour se positionner à une cellule du tableau, il suffit d'indiquer le numéro de cellule. Ce numéro commence par l'indice 1.

5.1.2 l'instructions d'entrées appliquées dans un Tableaux à une dimension

La syntaxe est :

Saisir(nom_tableau[taille])

Nous prenons l'exemple de Tab1et l'appliqué à la boucle POUR

```
Pour i = 1 à 10 faire
Saisir (Tab1[i])
FINPOUR
```

5.1.3 l'instructions de sorties appliquées dans un Tableaux à une dimension

La syntaxe est :

afficher(``TEXTE``, nom_tableau[taille])
ou
afficher(nom_tableau[taille])

Nous prenons l'exemple de Tab1et l'applique à la boucle REPETER

```
 i = 1
REPETER
Afficher (``les valeurs du tableau Tab1 sont : ``, Tab1[i]))
i=i+1
JUSQUA(i=10)
FINPOUR
```

5.2 Tableaux à deux dimensions

5.2.1 Définition

Syntaxe Générale :

NomDuTableau: Tableau [Min .. Max] [Min …Max] de Type avec (min >0)

Tab4 : Tableau [1...4][1…..4] de Entier

ou

Tab4: Tableau[1...4,1…..4] de Entier

Tab5 : Tableau[1…3][1…4] de reel

Tab6 : Tableau[1…2][1…3] de caractère

Tab4

3	18	55	6
7	99	33	9
0	3	99	2
5	44	11	1

Tab5

5.0	4.6	8.9	10.6
4.3	5.8	10.6	19.77
9.0	44.6	88.9	39.5

Tab6

K	L	M
D	C	P

5.2.2 l'instructions d'entrées appliquées dans un Tableaux à deux dimensions

La syntaxe est :

Saisir(nom_tableau[taille1] [taille2])

Nous prenons l'exemple de Tab4 et l'appliqué à la boucle POUR

Pour i = 1 à 4 faire
Pour j = 1 à 4 faire
Saisir (Tab4[i][j])
FINPOUR
FINPOUR

5.2.3 l'instructions de sorties appliquées dans un Tableaux à deux dimensions

La syntaxe est :

afficher (``TEXTE``, nom_tableau[taille1][taille2])
ou
afficher(nom_tableau[taille1][taille2])

Nous prenons l'exemple de Tab4 et l'appliqué à la boucle POUR

```
Pour i = 1 à 4 faire
Pour j = 1 à 4 faire
afficher (Tab4[i][j])
FINPOUR
FINPOUR
```

5.2.4 Chaines de caractères

Un type chaîne de caractères permet de définir des données correspondant à une suite de caractères (appelée chaîne de caractères) dont la longueur peut être spécifiée dans la définition. Si elle n'est pas spécifiée, le système lui attribue une longueur égale à 255 caractères.

Exemple:

Variable Expression = Chaîne [15]

Dans cet exemple, la variable Expression a été spécifiée comme étant un type chaîne pouvant contenir au maximum 15 caractères.

Chapitre 6 les Sous-programmes

6.1 Définition

Un sous-programme est rédigé de façon telle que son exécution puisse être commandée par un programme. Celui-ci est appelé programme appelant. Il fournit des données au sous-programme et récupère les résultats de ce dernier. On distingue deux types de sous programmes : les procédures et les fonctions. La différence est qu'une fonction renvoie une valeur alors qu'une procédure ne renvoie pas de valeur.

6.2 Les procédures

Les procédures sont composées d'un en-tête de procédure et d'un bloc d'instructions : c'est la partie déclarative et le corps de la procédure. La syntaxe générale est la suivante :

Procédure<identificateur> (<liste de paramètres>)
Début
Déclaration de constante
Déclaration de variable
Instruction(s)
FinProcedure

L'appel d'une procédure se fait au sein du programme principal avec une instruction composée de l'identificateur de la procédure suivi des paramètres effectifs. Les paramètres permettent à un programme appelant de transmettre à un sous-programme des données lors de l'appel de ce dernier. Un sous-programme est rédigé de façon à pouvoir recevoir des données du programme appelant ; cela est possible grâce aux paramètres. Ils sont appelés formels lors de la définition et effectifs lors de l'appel. Il existe deux types de paramètres : les paramètres transmis par valeur et les paramètres transmis par adresse.

6.2 Les fonctions

Les fonctions sont constituées d'un en-tête de fonction (partie déclaration) et d'un bloc d'instructions (corps de la fonction). Les fonctions effectuent certaines

opérations avant de renvoyer un résultat ; elles sont donc appelées dans une expression.

La syntaxe générale d'une fonction est la suivante :

Fonction <identificateur> (<liste de paramètres>): <Type résultat>

Début

Déclaration de constante

Déclaration de variable

Instruction(s)

<Identificateur> = Valeur retour

FinFonction