Chapitre 1

INTRODUCTION À L'ALGORITHMIQUE

1.L'intérêt des Algorithmes :

La raison d'être d'un algorithme est de résoudre un problème. La plus grande attention doit être portée à la compréhension du problème, faute de quoi un algorithme n'a aucune chance d'être correcte.

Généralement, lors de l'informatisation d'une tache les programmeurs doivent suivre une démarche logique contenant plusieurs étapes, l'aboutissement de cette démarche conduit à l'élaboration d'un algorithme offrant une solution au problème posé, cet algorithme sera par la suite traduit en langage de programmation afin d'être compris et exécuter par une machine.

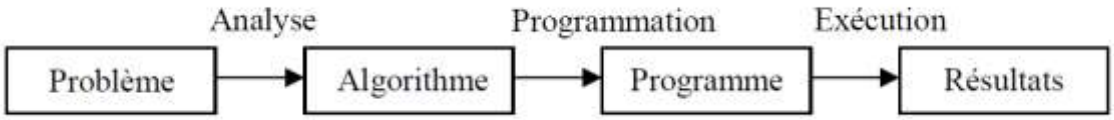

Figure1- étape de réalisation d'un programme

2. Définition d'un algorithme :

Un algorithme, c'est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné. Cet ensemble d'instructions suit un ordre chronologique bien déterminé.

Exemple :

Pour mieux comprendre la notion d'algorithme donnons deux exemples assez simples ; si en passant dans la rue un touriste vous demande de lui indiquer le chemin vers un endroit bien déterminer ou si vous lisez un manuel d'utilisation d'un appareil que vous venez d'acheter ; en fait vous êtes entrain de suivre un ensemble d'instructions étape par étape ; de ce fait on peut dire que l'algorithme est un ensemble d'instruction élémentaire qui se déroulent étape par étape dans un ordre bien déterminé.

3. Démarche de réalisation d'un Algorithme :

Pour concevoir un bon algorithme qui donnera les résultats attendues il faut suivre les étapes suivantes:

1- Bien comprendre l'énoncé du problème.

2- Décomposer le problème en sous-problèmes plus simple à résoudre.

3- Associer à chaque sous problème, une spécification :

- Les données nécessaires.
- Les données résultantes.
- La démarche à suivre pour arriver au résultat en partant d'un ensemble de données.

4. Elaboration d'un algorithme :

On peut représenter un algorithme à l'aide du schéma suivant:

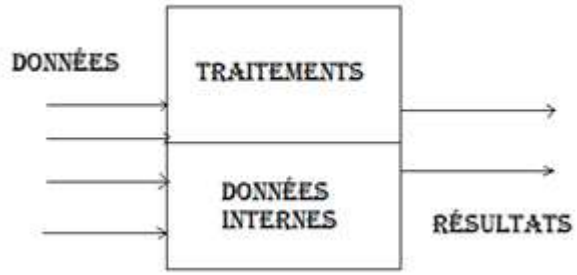

Figure2- représentation d'un Algorithme

5. Les variables :

Les données ainsi que les résultats des calculs intermédiaires ou finaux, sont rangés dans des ***cases-mémoires*** appelées ***variables***. Une variable est un emplacement mémoire nommé, de taille fixée prenant au cours du déroulement de l'algorithme un nombre indéfini de valeurs différentes.

Remarques :

- La lisibilité des algorithmes dépend étroitement des choix des noms de variables qui doivent être simples et significatifs, *Exemple* : « age » est un meilleur choix pour designer l'âge d'une personne que « y».

- Le nom ou l'identificateur d'une variable peut contenir des chiffres ainsi que des lettres mais doit obligatoirement commencer par une lettre ; il ne peut pas contenir de caractère accentuer ni d'espace.

6. Les constantes :

A l'opposer d'une variable une constante aura une valeur fixe dés le début de l'algorithme et qui ne pourra en aucun cas être changé par une autre valeur.

Une constante est déclarée comme suit :

```
Constante Nom_constante =valeur
```

Exemple :

Constante PI=3.14

Constante g=9.8

7. Types des variables :

Chaque variable dans un algorithme doit avoir un type précis afin de définir la plage des valeurs possibles qu'on peut attribuer à cette variable ainsi que les opérations autorisées sur cette dernière.

La déclaration d'une variable se fait de la manière suivante :

```
Variables
        Nom_variable : Type
```

Les différents types utilisés :

- **Type Entier** : pour manipuler les nombres entiers positifs ou négatifs. Par exemple : 5, -15, etc.
- **Type Réel** : pour manipuler les nombres à virgule. Par exemple : 3.14, -15.5, etc.
- **Type Caractère** : pour manipuler des caractères alphabétiques et numériques. Par exemple : 'a', 'A', 'z', ' ?', '1', '2' etc.
- **Type Chaîne** : pour manipuler des chaînes de caractères permettant de représenter des mots ou des phrases. Par exemple : "bonjour, Monsieur"
- **Type Booléen** : pour les expressions logiques. Il n'y a que deux valeurs booléennes : vrai et faux.

Exemple :

Variables

n : entier

r : réel

a, b: booléens

nom_etudiant : chaîne

A un type donné, correspond un ensemble d'opérations définies pour ce Type.

7.1 Type Entier :

Le type Entier prend ses valeurs dans l'ensemble Z, les opérations permises sur les entiers sont :

Opération	***Symbole***
1- Addition	+
2- Soustraction	-
3- Multiplication	*
4- Division entière	Div
5- Le reste de la division entière	Mod
6- Exposant	^
7- Comparaison	<, <=, > ,>= , =, #

Exemples :

x, y, z : entier

x=4, y=3

z=x+y => z=7

z=x-y => z=1

z=x Div y =>z=1

z=x Mod y =>z=1

z=x^y =>z=64

7.2 Type Réel :

Le type réel prend ses valeurs dans l'ensemble R, les opérations permises sur les réels sont :

Opération	***Symbole***
1- Addition	+
2- Soustraction	-

3-	Multiplication	*
4-	Division entière	Div
5-	Exposant	^
6-	Comparaison	<, <=, > ,>= , =, #

Exemple :

x, y, z : réel

x=1, y=2

z=x/y => z=0.5

7.3 Type Caractère:

Un caractère peut prendre les valeurs suivantes :

De "0" à "9", "A" à "Z", "a" à "z", et les caractères spéciaux tels que : "*", "#", "%", "&", "$", etc.…

Un caractère est toujours encadré entre les deux guillemets.

Les opérations permises sur ce type sont :

Opération	***Symbole***
1- Inférieur	<
2- Inférieur ou égal	<=
3- Supérieur	>
4- Supérieur ou égal	>=
5- Egal	=
6- Différent	#

On peut se demander comment il existe une comparaison entre les caractères la réponse est simple car la comparaison entre les caractères se fait par leur code d'ASCII.

Exemple :

"A" < "a" car le code ASCII de "A" est 65 et le code ASCII de "a" est 97

"0" < "1 " car le code ASCII de "0" est 48 et le code ASCII de "1" est 49

7.4 Type Booléen :

Le type Booléen représente les variables logiques qui ne peuvent prendre que deux valeurs possibles vrai(1), faux(0).

Ce type de variable est utilisé pour évaluer une expression ou une condition.

Les principales opérations permises sur ce type sont :

Opération	***Symbole***
1- Négation	Non
2- Le et logique	Et

3- Le ou logique	Ou

Il existe d'autres opérateurs qui sont en fait la combinaison des trois opérations principales.

Ci-dessous les tables de vérités de chaque opération :

A	Non(A)
Vrai	Faux
Faux	Vrai

A	B	A Et B
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux
Vrai	Vrai	Vari

A	B	A Ou B
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux
Vrai	Vrai	Vari

7.5 Priorités des Opérations :

Une expression est une combinaison de variables et d'opérations.

Exemple :

a*b-c/(2*d)+h : est une expression arithmétique.

(a<b) Et (c>d) : est une expression logique.

Afin de bien évaluer une expression et pour éviter toute ambigüité il y'a un ordre ou une priorité selon laquelle se déroulent les opérations.

Opérations Arithmétiques :

Pour évaluer les opérations arithmétiques sans commettre des erreurs lors des opérations de calculs, on doit suivre les ordres de priorité des opérations définies ci-dessous :

Opération	***Priorité***
Signe négatif -	1
Parenthèse ()	2
Exposant ^	3
Multiplication *, Division /	4
Addition +, Soustraction -	5

Opérations Logiques :

Pareil que pour les opérations arithmétiques toute opération logique possède une priorité. Les tableaux ci dessous résument ces ordres de priorités :

Opération	***Priorité***
Non	1
Et	2
Ou	3

Opération	***Priorité***
>	1
>=	2
<	3
<=	4
=	5
#	6

Indication :

Comme il y'a des types prédéfinis en algorithme le programmeur a aussi la possibilité de définir de nouveaux types selon ses besoins.

La définition de nouveau type se fait comme suit :

```
Type
        Nom_type=valeur
```

Exemple :

```
Type
Jour = ("lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi", "Dimanche")
Variables
J : Jour
Type
Moyenne=0..20
Variables
M :Moyenne
```