Algorithmique

2.1 Introduction

L'algorithmique est la logique qui utilise des algorithmes.

En pratique, pour résoudre un problème à l'aide d'un ordinateur, on doit suivre certaines étapes. Ces dernières sont ordonnées comme suit :

- Bien cerner le problème à traiter; cela consiste à définir toutes les données avec leurs natures ainsi que les résultats auxquels on s'attende...
- Formulation de la résolution du problème sous forme d'une succession logique d'instructions pour passer des données aux résultats. (l'algorithme).
- Pour bien schématiser le problème, on dresse un graphique (organigramme) équivalant à l'algorithme.
- Ensuite on réécrit l'algorithme dans un langage évolué (programmation).
- On compile le programme en question (compilation).
- Une fois le programme est compilé, on l'exécute, puis on vérifie si les résultats correspondent aux solutions du problème.

2.2 Algorithmes

2.2.1 Définition

Le mot algorithme est dérivé du nom du mathématicien arabe "al-Khuwarizmi"
Les algorithmes sont constitués par un ensemble de règles précises et compréhensibles par tous. Ils s'appliquent à des données et élaborent les résultats en fonction des données initiales

L'algorithme consiste à bien analyser le problème avant de commencer n'importe quel programme.

2.2.2 Caractéristiques d'un algorithme :

Un algorithme informatique doit répondre à des règles opérationnelles :

- il doit se terminer après un nombre fini d'opérations;
- Chaque instruction doit être définie sans ambiguïté;
- Le fonctionnement de l'algorithme est déterministe (il donne les mêmes résultats pour les mêmes données).
- Il n'existe pas de méthode pour découvrir un algorithme. Sa réalisation est un acte créatif.

- De plus, un même problème peut être résolu au moyen de plusieurs algorithmes.

2.2.3 Structure générale d'un algorithme :

La structure d'un algorithme est composée de trois parties :

1. l'entête;
2. les déclarations
3. les actions.

Exemple : Revalorisation des salaires des retraités ayant un salaire inférieur à 8000DA/mois

```
ALGORITHME  Revalorisation de retraite     (1)
VARIABLES   somme_initiale , taux, majoration,  valeur_acquise  en réel      (2)
DEBUT
    ECRIRE " Introduisez le salaire initial (en Dinars): "
    LIRE somme_initiale
    SI somme_initiale >=8000DA  ALORS  ECRIRE "Majoration refusée"
      SINON
          ECRIRE  " Introduisez le taux de majoration (ex: 2.5 pour 2.5%): "
          LIRE  taux
          majoration <-- somme_initiale * taux / 100
          valeur_acquise <-- somme_initiale + majoration
          ECRIRE " La majoration fournie est de " majoration , "dinars "
          ECRIRE " Le nouveau salaire est " valeur_acquise , "dinars "
FIN
```

(3)

Le langage utilisé dans l'algorithme ci-dessus est appelé: *langage de description d'algorithme (LDA)* ou encore: *pseudocode*. Dans un langage de description, les actions sont généralement décrites par un symbole ou un verbe à l'infinitif bien choisi pour éviter les confusions.

L'entête :

L'entête contient le titre qu'on souhaite donner à l'algorithme. Dans l'exemple précédent elle est représentée par la partie (1).

2.2.4 Les déclarations :

Dans la partie déclaration, on doit déclarer tous les objets qui seront utilisés par la suite dans l'algorithme. (La partie (2) de l'exemple précédent)

L'objet : c'est un symbole ou un identificateur qu'on déclare avant sa manipulation.

2.2.4.1 La constante :

C'est un symbole ou un identificateur dont la valeur est introduit à l'avance dans le programme. Sa valeur reste donc constante pendant l'exécution du programme.

Exemple : Pi= 3.14, seuil=10.00…etc

2.2.4.2 Le type :

Un type est un ensemble de valeurs que peut prendre une donnée. On distingue :

- Types standards : entier, réel, booléen, caractère.

Exemple: X: caractère;

- Types non-standards :

1. Énumération :

Syntaxe : type identificateur = (id1, id2,…,idn)
Exemple : réel couleur = (jaune, vert, rouge,…)

2. Intervalle :

Syntaxe : type identificateur = inf….sup
Exemple : entier décimal = 0 . . 9

2.2.4.3 La variable :

C'est un symbole ou un identificateur dont la valeur est susceptible de changer au cours de l'exécution du programme. Il y a plusieurs types de variables: entier, réel, logique, chaîne de caractères, tableau…etc

2.2.5 Les actions ou instructions :

La partie instruction est le corps du programme (elle correspond à la partie (3) de l'exemple précédent) où sont écrites les instructions nécessaires pour arriver au résultat. Toute manipulation sur les objets est appelée action.

2.2.5.1 Lecture :

La lecture dans un programme reviens à introduite une donnée par l'utilisateur pendant l'exécution du programme. La donnée sera stockée immédiatement dans une variable choisi à l'avance par l'utilisateur.

Syntaxe (en français) : LIRE (a,b) est équivalent à: LIRE *a*, LIRE *b*, cela veut dire que l'utilisateur fait entrer, par clavier, la valeur de *a* puis celle de *b*.

2.2.5.2 Ecriture :

Le terme ECRIRE veut dire affichage d'un message texte sur l'écran. En fait, l'instruction d'écriture est couramment utilisée dans deux situations :

- Soit avant la lecture des données.
- Soit pour afficher les résultats après l'exécution,

Syntaxe : ECRIRE (" Veuillez Introduire le seuil:"); // avant la lecture
ECRIRE ("La moyenne trouvée est :", moy) // après l'exécution

2.2.5.3 Affectation :

C'est une instruction qui consiste à attribuer une valeur à une variable lors de l'exécution du programme. La variable change de valeur à chaque affectation dans un même programme. *Exemples :*

1) A ← 6, B ← A+1, A ← 5, B ← A+4, après l'exécution A=5 et B=9
2) L'incrémentation : I ←I+1 (I est augmenté de 1)

2.2.5.4 Structures conditionnelles :

Différents opérateurs de comparaison utilisés:

- *Expressions et opérateurs* :

- Opérateurs numériques : +, - , *, / , mod. L'ordre des opérations est important, la priorité revient aux opérateurs / , *, et mod.[1]

Exemple : 4*3+7/2=(4*3)+(7/2) et non à 4*(3+7/2) ou (4*3+7)/2 .

- Opérateurs logiques : ET, OU, NON, et XOR.
- Opérateur de comparaison :

  Egal : =, Différent : < >, Egal ou supérieur : = >, Egal ou inferieur =<, supérieur : >, inférieur : <

- *Sélection simple :*

  Si (condition) Alors (instruction1)

[1] La priorité peut changer d'un langage à un autre il est fortement conseillé d'utiliser des parenthèse pour définir l'ordre d'application des opérateurs.

Sinon (instruction2)
Fin si

- *Tests imbriqués :*

Si (condition1) Alors (instruction1)
Sinon Si (condition2) Alors (instruction2)
Sinon Si (condition3) Alors (instruction3)
Fin Si
Fin Si
Fin Si

Exemple :

Ecrire un algorithme qui affiche les états de l'eau en fonction de sa température T.

ALGORITHME Etat_température
Variable T en réel
Début
Ecrire ('donnez la température T :')
Lire T
Si T=< 0 Alors Ecrire ("l'eau est dans un état solide")
Sinon Si T< 100 Alors Ecrire ("l'eau est dans un état liquide")
Sinon Ecrire ("l'eau est dans un état gazeux")
Fin Si
Fin Si

FIN

- Structures répétitives : Il existe trois type de structures répétitives :

1) *<u>La Boucle **Pour**</u> :*

Elle est souvent utilisée quand on connaît le nombre de boucles à réaliser.

Syntaxe :

Pour i ← (valeur initiale) à (valeur finale) **faire** (ensemble d'instructions)

Fin Pour

Exemple :

Effectuer la somme des n premiers entiers naturels.

ALGORITHME Somme

Variables s, i, n en entier

DÉBUT

Ecrire ("donnez un nombre n :")

Lire n

S← 0

Pour i←1 à n faire s←s+i

Fin Pour

Ecrire ('la somme est :', s)

FIN

2) La Boucle **Tant que** :

Syntaxe :

Tant que (condition) faire (ensemble d'instructions)

Fin Tant que

Remarque :

- Pas d'augmentation automatique d'une variable.
- Faire le bloc d'instructions tant que la condition est vérifiée.
- Attention, si la condition est toujours vraie l'exécution des instructions ne s'arrêtera jamais

ALGORITHME Somme

Variables s, i, n en entier

Début

Ecrire ('donnez un nombre n :')

Lire n

s← 0
i←1
Tant que i<=n faire

s←s+i

i←i+1
Fin Tant que

Écrire ("la somme est :", s)

Fin

3) *La Boucle **Répéter**:*

Syntaxe :

Répéter

Ensemble d'instructions

Jusqu'à (condition)

Fin Répéter

Remarques :

- Le bloc d'instructions est à faire si la condition n'est pas réalisée.
- Le bloc d'instructions est parcourue au moins une fois.

Exemple :

ALGORITHME Somme

Variables s, i, n en entier

DÉBUT

Ecrire ('donnez un nombre n :')

Lire n

s← 0
i←1
Répéter

s←s+i

i←i+1

jusqu'à i>n

Fin Répéter

Écrire ('la somme est :', s)

FIN

2.3 Les tableaux

Supposons en premier lieu qu'on veut calculer par exemple la moyenne d'un groupe de 12 étudiants, dans ce cas il faut déclarer 12 variables n1, n2,…, n12. Supposant maintenant qu'on a affaire à 60000 variables, dans un tel cas il est insensé de déclarer toutes ces variables. Pour surpasser ce problème on fait recours aux tableaux. Dans ce cas on déclare une seule variable tableau au sein de laquelle chaque variable sera indexée par un numéro renvoyant à la case qui la contient.

2.3.1 Les vecteurs (tableau à une dimension) :

Un vecteur est ensemble de valeurs ordonnées les unes après les autres. Il est défini par sa taille et le type de valeurs qu'il contient.

Exemple : Soit V(N) un vecteur qu'on représente comme suit :

12	15	2	6	14

N=5, V(1) =12, V(5)=14.

Déclaration d'un vecteur :

On doit préciser dans la variable, le plus grand indice et le type des valeurs que le vecteur contient.

Syntaxe :

Variable V : tableau (N) en entier

Remarque : L'avantage des tableaux c'est qu'on va pouvoir traiter des données à laide des boucles.

Exemple : La moyenne d'une section de 170 étudiants

```
ALGORITHME moyenne
VARIABLES s en réel
                i en entier
          N : tableau (170) en réel
Début
Pour i←1 à 170 faire
              Lire  N(i)
Fin Pour
```

s←0

Pour i← 1 à 170 faire

    s←s+N(i)

Fin Pour

Ecrire (‘la moyenne est:’, s/170)

FIN