Chapitre A

LES BASES DE L'ALGORITHMIQUE

1-NOTION D'ALGORITHME

1-1 Vers une définition

Voici la définition du mot algorithme dans le dictionnaire Larousse illico :
« Ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations. Un algorithme peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur. »

Nous appellerons *algorithme*, un ensemble fini d'instructions d'un calcul ou de la résolution d'un problème. Un *programme* sera la traduction d'un algorithme dans un langage de programmation exécutable par un ordinateur ou une calculatrice programmable.

Le mot « Algorithme » vient du nom du mathématicien Al-Khuwarizmi (latinisé au Moyen Âge en Algoritmi) né vers 783 en Ouzbékistan et mort vers 850. C'est un des pionniers de l'algèbre. Dans son traité *Kitab al jabr w'al muqabalah*, il étudie les équations du second degré.

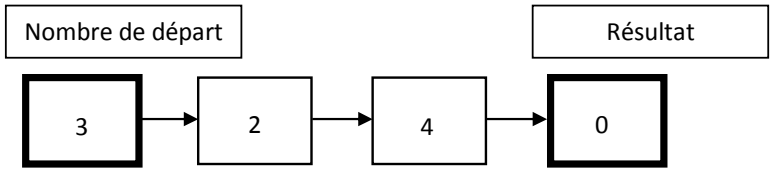
1-2 Exemples

- **Exemple n°1 :** un algorithme de calcul classique

On donne l'algorithme suivant écrit en *langage courant* :

- *Choisir un nombre.*
- *Lui soustraire 1.*
- *Elever au carré la somme obtenue.*
- *Ajouter l'opposé de 4.*
- *Ecrire le résultat.*

Appliquons cet algorithme au nombre 3 :



On souhaite automatiser ce calcul pour l'appliquer à n'importe quel nombre x de départ.
L'algorithme suivant écrit en *langage symbolique* donne toutes les étapes pour y parvenir :

Variables
Réels : x, a, b, R
Entrée
Saisir x
Traitement
$a \leftarrow x - 1$
$b \leftarrow a^2$
$R \leftarrow b - 4$
Sortie
Afficher R

Remarques :
- Le symbole ← est celui de l'*affectation*.
- Dans certains algorithmes apparaît en plus la phase d'*initialisation*.
- L'*initialisation* correspond à l'affectation d'une variable lors de la définition de cette variable.
- Toute modification de la valeur de la variable après sa définition n'est plus une *initialisation* mais une *affectation*.

Nous utiliserons aussi cette deuxième rédaction d'algorithme plus simple d'emploi car il est parfois difficile de bien séparer l'entrée, l'initialisation, le traitement et la sortie de l'algorithme.

Variables
Réels : x, a, b, R
Début
Saisir x
$a \leftarrow x - 1$
$b \leftarrow a^2$
$R \leftarrow b - 4$
Afficher R
Fin

➢ **Exemple n°2 :** permuter les valeurs de 2 variables

L'exemple suivant illustre une démarche très importante en algorithmique, celle de prendre une variable auxiliaire ou temporaire $Temp$ pour échanger le contenu de deux variables A et B. On évite ainsi d' « écraser » l'une des deux valeurs de A ou B.

Variables
Réels : A, B, $Temp$
Début
Saisir A, B
$Temp \leftarrow A$
$A \leftarrow B$
$B \leftarrow Temp$
Afficher A, B
Fin

Un exemple d'exécution de l'algorithme :

A	B	$Temp$	Instructions
2	3		Saisir A, B
2	3	2	$Temp \leftarrow A$
3	3	2	$A \leftarrow B$
3	2	2	$B \leftarrow Temp$
3	2		Afficher A, B

➢ **Exemple n°3 :** permuter les valeurs de 3 variables

Le principe est toujours le même : prendre une variable tampon $Tamp$.

Variables
Réels : A, B, C, $Tamp$
Entrée
Saisir A, B, C
Traitement
$Tamp \leftarrow A$
$A \leftarrow B$
$B \leftarrow C$
$C \leftarrow Tamp$
Sortie
Afficher A, B, C

Un exemple d'exécution de l'algorithme :

A	B	C	$Tamp$	Instructions
2	3	4		Saisir A, B, C
2	3	4	2	$Tamp \leftarrow$ A
3	3	4	2	$A \leftarrow B$
3	4	4	2	$B \leftarrow C$
3	4	2	2	$C \leftarrow Tamp$
3	4	2		Afficher A, B, C

1-3 Les entrées/sorties, les mémoires ROM/RAM et le traitement des données

Un ordinateur communique avec l'utilisateur (l'extérieur) par les entrées (instruction *Saisir*) et les sorties (instruction *Afficher*). L'instruction *Saisir* transmet une information à l'ordinateur par le clavier. Cette information est stockée dans la *mémoire centrale* pour être utilisée immédiatement ou plus tard selon les besoins du programme.
Les résultats du traitement de l'information par l'ordinateur sont communiqués à l'utilisateur par l'instruction A*fficher* en général sur l'écran vidéo.
Lorsqu'une information est stockée de façon définitive sans qu'on puisse la modifier, on parle de *mémoire morte* ou ROM (Read Only Memory). Cette mémoire contient tous les programmes utilitaires installés à l'usine de fabrication dont l'ordinateur a besoin pour fonctionner. Elle est conservée quand on éteint l'ordinateur.
La *mémoire vive* ou RAM (Random Access Memory) est la zone mémoire dans laquelle un ordinateur peut lire les données, les écrire ou les modifier lors de leur traitement. Cette mémoire est caractérisée par sa rapidité d'accès et par le fait que les données sont perdues lorsqu'on éteint l'ordinateur.
Le programmeur stockera ses programmes dans la *mémoire de masse* en général le disque dur de l'ordinateur.
La *phase de traitement* des données (calculs arithmétiques et logiques) utilise l'unité de traitement de l'ordinateur.

1-4 Exercices

➢ **Exercice n°1**

1) Déterminer la fonction f définie par l'algorithme suivant :

Variables
Réels : x, a, b, c
Début
Saisir x
$a \leftarrow x + 1$
$b \leftarrow a^2$
$c \leftarrow b - a + 1$
Afficher c
Fin

2) Ecrire en langage symbolique l'algorithme de calcul suivant donné lui en langage naturel :

Choisir un nombre
Elever ce nombre au carré
Soustraire 5
Prendre l'inverse de ce nombre
Ajouter 3
Afficher le résultat

➢ **Exercice n°2**

Soit M un point du plan de coordonnées (x, y) dans un repère (O ; I, J).

1) Ecrire un algorithme qui retourne les coordonnées du point M' symétrique de M par la symétrie centrale de centre O.
2) Soit A un point de coordonnées $(x_A; y_A)$. Modifier l'algorithme précédent pour qu'il retourne les coordonnées du point M' symétrique de M par la symétrie centrale de centre A.

➢ **Exercice n°3**

Comment faire pour permuter les valeurs de 4 variables x, y, z et t ? Pouvez-vous en proposer un algorithme ?

➢ **Exercice n°4**

1) Compléter le tableau et dire ce que fait la séquence d'instructions suivante :

$C \leftarrow 2{,}5$
$D \leftarrow 5$
$C \leftarrow C + D$
$D \leftarrow C - D$
$C \leftarrow C - D$
Afficher C, D

C	D	Instructions

2) Pouvez-vous proposer une séquence d'instructions plus « performante » ?

2-LES INSTRUCTIONS CONDITIONNELLES

2-1 Deux structures possibles

Dans un algorithme selon qu'une *condition* est vraie ou fausse, on exécute un bloc d'*instructions* ou l'autre. On traduit ce *test* par l'instruction conditionnelle suivante **Si…alors…Sinon…Fsi** :

Si *condition* alors
    Instruction 1
    Instruction 2
    …
Sinon
    Instruction A
    Instruction B
    …
Fsi

Ou parfois plus simplement par **Si…alors…Fsi** :

Si *condition* alors
    Instruction 1
    Instruction 2
    …
Fsi

2-2 Exemples

➢ **Exemple n°1 :** prise en compte de la valeur interdite lors d'un calcul d'images

Variables
Réels : x, Y
Entrée
Saisir x
Traitement
Si $x = 10$ alors
    Afficher « Valeur interdite»
Sinon
    $Y \leftarrow (2x + 1)/(x - 10)$
Fsi
Sortie
Afficher Y

- **Exemple n°2 :** étudier la parité d'un entier n

La fonction *partie entière* sera notée Ent.
Ent(x) est égal au plus grand entier inférieur ou égal à x. Exemple : Ent(2,1) = 2. Nous allons utiliser cette fonction mathématique pour trouver la parité d'un nombre.
Exemple : comme $2 \times \text{Ent}\left(\frac{3}{2}\right) = 2 \times \text{Ent}(1{,}5) = 2 \times 1 = 2 \neq 3$ l'entier 3 n'est pas pair.

Variable
Entier : n
Entrée
Saisir n
Traitement et sorties
Si $2 * \text{Ent}(n/2) = n$ alors
    Afficher « La valeur entrée est paire »
Sinon
    Afficher « La valeur entrée est impaire »
Fsi

Remarque : nous utiliserons aussi la fonction partie entière pour démontrer qu'un nombre est entier ou pour générer des entiers aléatoires.

- **Exemple n°3 :** déterminer le plus grand de deux nombres donnés

Variables
Entier : n
Réels : a, b
Entrée
Saisir a, b
Traitement et sorties
Si $a > b$ alors afficher « La plus grande valeur est », a
Si $b > a$ alors afficher « La plus grande valeur est », b
Si $a = b$ alors afficher « les deux valeurs sont égales »

L'algorithme précédent effectue dans tous les cas 3 comparaisons même si la première suffit.

L'algorithme qui suit est plus performant car il effectue moins de comparaisons.

Variables
Entier : n
Réels : a, b
Entrée
Saisir a, b
Traitement et sorties
Si $a > b$ alors
    Afficher « La plus grande valeur est », a
Sinon
    Si $b > a$ alors
        Afficher « La plus grande valeur est », b
    Sinon
        Afficher « les deux valeurs sont égales »
    Fsi
Fsi

Lorsque a est le plus grand des deux nombres, ce dernier algorithme n'effectue pas les autres comparaisons.

- **Exemple n°4 :** instructions conditionnelles imbriquées

En algorithmique, nous avons souvent à traduire la situation suivante :

Instruction = $\begin{cases} Instruction\ 1\ si\ condition\ A \\ Instruction\ 2\ si\ condition\ B \end{cases}$.

Comme dans l'exemple précédent, au lieu de :

Si *condition A* alors $Instruction = Instruction\ 1$
Si *condition B* alors $Instruction = Instruction\ 2$,
il est préférable de choisir la traduction suivante pour effectuer le moins de comparaisons possible :

Si *condition A* alors
    $Instruction = Instruction\ 1$
Sinon
    Si *condition B* alors
        $Instruction = Instruction\ 2$
    Fsi
Fsi

- **Exemple n°5 :** modélisation d'un tirage de boule

On considère un sac qui contient 2 boules blanches, 3 boules noires et 5 boules rouges indiscernables au toucher. On tire au hasard une boule de l'urne et on note sa couleur.

Phase de modélisation

L'instruction *hasard* retourne un nombre à virgule de manière aléatoire compris entre 0 et 1. Plus précisément 0 ≤*hasard*<1.
L'instruction Ent(*hasard*×10)+1 (avec Ent la fonction partie entière) renvoie un entier de manière aléatoire compris entre 1 et 10. Plus précisément :
Ent(*hasard*×10)+1 ∈ $\{1; 2; \dots; 9; 10\}$.

Nous pouvons modéliser ainsi le tirage de boule :

Si Ent(*hasard*×10)+1 ∈ $\{1; 2\}$, on a tiré une boule blanche.
Si Ent(*hasard*×10)+1 ∈ $\{3; 4; 5\}$, on a tiré une boule noire.
Si Ent(*hasard*×10)+1 ∈ $\{6; 7; 8; 9; 10\}$, on a tiré une boule rouge.

D'où l'algorithme suivant :

```
Variable
Entier : T
Début
T ← Ent(hasard * 10) + 1
Si T ≤ 2 alors
        Afficher « On a une boule blanche »
Sinon
        Si T ≤ 5 alors
                Afficher « On a une boule noire »
        Sinon
                Afficher « On a une boule rouge »
        Fsi
Fsi
Fin
```

3-LES TYPES DE DONNEES ET OPERATIONS LOGIQUES

3-1 Les types de données

Jusqu'ici nous n'avons utilisé que des variables de type nombre (entier ou réel).
Il existe aussi des variables de type :

- *booléen* qui prennent deux valeurs soit VRAI, soit FAUX ou encore les valeurs 1 ou 0.
- *Caractère ou chaîne de caractères*.

« Coucou papa » est une *chaîne de caractères* de longueur 11. L'espace entre les deux mots a aussi été comptabilisé.
Un *caractère* est soit une lettre minuscule ou majuscule, soit un chiffre ou un symbole (" a ", "B ", "1", "$",…). C'est une chaîne de caractères de longueur 1.