Algorithmique

Table des matières

I Qu'est-ce qu'un algorithme ?

Le mot algorithme vient du nom Al Khwarizmi (Abu Ja'far Mohammed Ben Mussa Al-Khwarismi), né vers 780 dans la région de Khwarezm (aujourd'hui Khiva), au sud de la Mer d'Aral (Ouzbékistan), et mort à Bagdad en 850.

Définition

Un algorithme est une suite d'opérations élémentaires, à appliquer dans un ordre déterminé à des données.
C'est donc une liste d'instructions élémentaires à suivre. Ces instructions fournissent en un nombre fini d'étapes des résultats.
Écrire un algorithme consiste à donner une méthode détaillée décrivant toutes les étapes d'une t,che à ccomplir.

Exemples : une notice de montage, une recette de cuisine, un chemin indiqué par un GPS, une division à la main, trier des cartes à jouer, calculer la somme des termes consécutifs d'une suite $\cdots$ sont des algorithmes.

On peut considérer un algorithme comme une machine fonctionnant en trois étapes :

1. les éléments dont on part : les entrées ;
2. les instructions à effectuer sur ces éléments : le traitement ;
3. les résultats obtenus : les sorties.

Exercice : On considère le programme de calcul suivant :

- Choisir un nombre.
- Lui ajouter 1.
- Multiplier le résultat par 2.
- Soustraire 3 au résultat.
- Afficher le résultat.

1. Appliquer cet algorithme à 3 ; -4, 0 et $\frac{1}{3}$.
2. Identifier les trois étapes de cet algorithme.

II Variables et affectation

Pour commencer un algorithme, il faut des éléments sur lesquels on souhaite travailler (dans l'exemple précédent, il nous faut un nombre). Ces éléments sont les données d'entrée qui seront utilisées lors des étapes du traitement.

Définition

Les données d'entrée sont stockées dans la mémoire de la calculatrice ou de l'ordinateur, à un emplacement appelé **variable** et repéré par un nom. On peut donc considérer une variable comme une boîte.
Le nom de la variable est son **étiquette**.
La variable peut contenir une valeur (un nombre, un mot, une liste de nombres $\cdots$)

Lorsque nous déclarons les variables, nous n'avons fait que réserver un espace dans la mémoire de l'ordinateur ou de la calculatrice.
C'est la même chose que lorsqu'un restaurateur marque « RÉSERVÉ » sur une table de restaurant.
Pour autant, la place n'est pas occupée.
Occuper cette place, c'est donner une valeur à l'espace mémoire réservé : c'est **l'affectation**.

Définition

Une **affectation** est l'attribution d'une valeur à la variable.

Si la variable s'appelle A, l'affectation peut s'écrire de différentes manières :

- Affecter à A la valeur 3 ;
- A prend la valeur 3 ;
- $A \leftarrow 3$;
- $3 \rightarrow A$.
- A=3

Exemples :

1. Considérons l'algorithme suivant :

Déclaration des variables	a ; b ; c
Début de l'algorithme	
Affectation	$a \leftarrow 3$
Affectation	$a \leftarrow a^2 - 2a + 1$
Affectation	$b \leftarrow 7$
Affectation	$b \leftarrow b^2 - 4b + 4$
Affectation	$c \leftarrow a + b - 9$
Instruction de sortie	Afficher c
Fin de l'algorithme	

1. Compléter le tableau suivant :

	Valeur de a	Valeur de b	Valeur de c
Étape 1			
Étape 2			
Étape 3			
Étape 4			
Étape 5			

2. Que permet de déterminer cet algorithme programmé avec le logiciel Algobox ?

```
VARIABLES
A EST_DU_TYPE NOMBRE
B EST_DU_TYPE NOMBRE
C EST_DU_TYPE NOMBRE
D EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE A
    LIRE B
    LIRE C
    D PREND_LA_VALEUR (A+B+C)/3
    AFFICHER D
FIN_ALGORITHME
```

3. Considérons l'algorithme suivant :

VARIABLES
A, B, B nombres réels
DEBUT ALGORITHME
Saisir un nombre strictement positif A
Saisir un nombre strictement positif B
$\sqrt{A^2+B^2} \rightarrow C$
Afficher C
FIN ALGORITHME

(a) Programmer cet algorithme sur Algobox puis sur calculatrice (voir le paragraphe sur les touches calculatrice à la fin du document).
On trouve :

```
VARIABLES
A EST_DU_TYPE NOMBRE
B EST_DU_TYPE NOMBRE
C EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE A
    LIRE B
    C PREND_LA_VALEUR sqrt(pow(A,2)+pow(B,2))
    AFFICHER "C="
    AFFICHER C
FIN_ALGORITHME
```

(b) Compléter le tableau ci-dessous :

Valeur de A	3	6
Valeur de B	4	8
Valeur de C		

(c) Que permet de faire cet algorithme ?

III Les instructions conditionnelles

On souhaite écrire un algorithme testant si un triangle est rectangle ou non en A.
Pour cela, il faut calculer BC^2 d'une part et $AB^2 + AC^2$ d'autre part, puis comparer les deux valeurs. Cela fait intervenir un test, aussi appelé instruction conditionnelle.

La résolution de certains problèmes nécessite la mise en place d'un test pour effectuer une t,che :

- si le test est positif, on effectue la t,che ;
- sinon, c'est-à-dire si le test est négatif, on effectue une autre t,che.

On parle d'**instruction conditionnelle** (ou test ou structure alternative).
En algorithmique, cette instruction conditionnelle se rédige ainsi :

Si *condition*
  alors *traitement 1*
  sinon *traitement 2*
FinSi

Exemples :

1) On considère l'algorithme suivant :

Variables
  x du type nombre
  y du type nombre

Entrée   demander un nombre x

Traitement
  Si $x < 0$
  Alors affecter à y le nombre $\frac{1}{x}$
  Sinon affecter à y le nombre $\sqrt{x}$
  FinSi

Afficher y

a) Tester cet algorithme pour $x = 1$; -2 ; 4 ; -1 ; 0.

b) Écrire la fonction obtenue sur $\mathbb{R}$.

2) **Suite de Syracuse** :

On définit la suite (u_n) ainsi : on choisit u_0 et $u_{n+1} = \frac{1}{2}u_n$ si u_n est pair, $u_{n+1} = 3u_n + 1$ si u_n est impair.
Écrire l'algorithme correspondant.

Avec Algobox, on a :

```
1: VARIABLES
2: a EST_DU_TYPE NOMBRE
3: b EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   LIRE a
6:   SI (a%2==0) ALORS
7:     DEBUT_SI
8:     b PREND_LA_VALEUR a/2
```

```
        FIN_SI
    SINON
        DEBUT_SINON
        b PREND_LA_VALEUR 3*a+1
        FIN_SINON
    AFFICHER "b="
    AFFICHER b
FIN_ALGORITHME
```

Remarque : cet algorithme ne permet que de calculer un terme de la suite, en fonction du terme précédent.
On aimerait pouvoir calculer un nombre déteriné de termes, par exemple les cent premiers termes.
Pour cela, il faut répéter le programme, c'est-à-dire faire une boucle ; c'est ce qu'on va voir dans le paragraphe suivant.

IV La boucle itérative

Exemples :

1. On souhaite calculer le nombre 2^{10}. Pour cela, on considère le nombre 2 puis on le multiplie par 2. On multiplie ensuite le résultat obtenu par 2 et ainsi de suite. On répête donc 9 fois l'opération : multiplier par 2. Répéter plusieurs fois de duite une même t,che s'appelle en algorithme **exécuter une boucle (à compteur)**.

   La boucle à compteur consiste à répéter une t,che fixe n fois, n étant fixé à l'avance.
   On la rédige de la manière suivante :

   Pour i allant de 1 à n
   | Traitement
   Fin boucle pour

2. Calcul des cent premiers termes de la suite de Syracuse :

```
VARIABLES
a EST_DU_TYPE NOMBRE
i EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE a
    POUR i ALLANT_DE 1 A 100
        DEBUT_POUR
        SI (a%2==0) ALORS
            DEBUT_SI
            a PREND_LA_VALEUR a/2
            FIN_SI
        SINON
            DEBUT_SINON
            a PREND_LA_VALEUR 3*a+1
            FIN_SINON
        AFFICHER "a="
        AFFICHER a
        FIN_POUR
FIN_ALGORITHME
```

3. On considère l'algorithme suivant :

Variables N du type nombre S du type nombre i du type nombre
Entrée Saisir N $0 \leftarrow S$
Traitement Pour i allant de 1 jusqu'à N faire $S + \dfrac{1}{i(i+1)} \leftarrow S$ FinPour
Afficher S

(a) Compléter le tableau ci-dessous en prenant comme valeur initiale de N : $N = 5$.

Numéro de passage dans la boucle	Valeur de S avant le passage dans la boucle	Valeur de S aprês le passage dans la boucle
1		
2		
3		
4		
5		

(b) À quoi sert cet algorithme ?

V La boucle conditionnelle

Les boucles vues précédemment sont três pratiques pour répéter des instructions à condition de savoir le nombre de répétitions nécessaires.
Or ce n'est pas toujours le cas : il est alors possible d'avoir recours à la structure TANT QUE...qui permet de répéter une série d'instructions (comprises entre DEBUT TANT QUE et FIN TANT QUE) tant qu'une certaine condition est vérifiée :

Exemple 1 :
Un individu a emprunté à un ami une somme de 2 500 euros (prêt sans intérêts).
Pour rembourser son ami, il prévoit de lui remettre 110 euros par mois. Mais comme cela ne correspond pas à un nombre pile de mois, il se demande quel sera le montant à rembourser le dernier mois.
En utilisant la variable montant pour représenter le montant qu'il reste à rembourser, le problême peut se résoudre avec l'algorithme suivant :

```
1: VARIABLES
2: montant EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
```

```
montant PREND_LA_VALEUR 2500
TANT_QUE (montant>=110) FAIRE
    DEBUT_TANT_QUE
    montant PREND_LA_VALEUR montant-110
    FIN_TANT_QUE
AFFICHER montant
FIN_ALGORITHME
```

Exemple 2

On cherche à connaître le plus petit entier N tel que 2^N soit supérieur ou égal à 10 000.
Pour résoudre ce problême de façon algorithmique, l'idée est de calculer les puissances consécutives de 2 jusqu'à ce qu'on atteigne 10 000.
Une structure TANT QUE est particuliêrement adaptée à ce genre de problême car on ne sait pas a priori combien de calculs seront nécessaires.
En utilisant la variable N, le problême peut se résoudre avec l'algorithme suivant (réalisé avec Algobox)
.

```
VARIABLES
N EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
N PREND_LA_VALEUR 1
TANT_QUE (pow(2,N)<10000) FAIRE
    DEBUT_TANT_QUE
    N PREND_LA_VALEUR N+1
    FIN_TANT_QUE
AFFICHER N
FIN_ALGORITHME
```

Exemple 3 :

On constate expérimentalement que la suite de Syracuse se termine toujours par le cycle 1 - 4 - 2 pour tous les nombres testés comme premier terme de la suite.
On appelle temps de vol le nombre d'étapes pour trouver un terme de la suite égal à 1, donc le rang du premier terme de la suite égal à 1.

Algorithme possible pour calculer ce temps de vol (page suivante) :

Algorithme

```
VARIABLES
n EST_DU_TYPE NOMBRE
u_n EST_DU_TYPE NOMBRE
u_n_temp EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    AFFICHER "Entrer un entier naturel : "
    LIRE u_n
    AFFICHER "u_n="
    AFFICHER u_n
    n PREND_LA_VALEUR 0
    TANT_QUE (u_n !=1) FAIRE
        DEBUT_TANT_QUE
        SI (u_n%2==0) ALORS
            DEBUT_SI
            u_n_temp PREND_LA_VALEUR u_n/2
            FIN_SI
        SI (u_n%2==1) ALORS
            DEBUT_SI
            u_n_temp PREND_LA_VALEUR 3*u_n+1
            FIN_SI
        u_n PREND_LA_VALEUR u_n_temp
        AFFICHER "u_n="
        AFFICHER u_n
        n PREND_LA_VALEUR n+1
        FIN_TANT_QUE
```

=0