

ALGORITHMIQUE

Le mot *algorithme* vient du nom de l'auteur persan **Al-Khuwarizmi** (né vers 780 - mort vers 850) qui a écrit en langue arabe le plus ancien traité d'algèbre dans lequel il décrivait des procédés de calcul à suivre étape par étape pour résoudre des problèmes ramenés à des équations.

Rédiger un algorithme consiste à décrire les différentes étapes à réaliser pour résoudre un problème : par exemple réaliser une recette de cuisine, convertir des kilomètres en mètres, des francs en euros, mise en marche d'une voiture, paiement avec une carte bancaire

On définit parfois les algorithmes de la manière suivante : *un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat et cela indépendamment des données.*

1. Les éléments de base d'un algorithme simple

a. Les trois étapes

La préparation du traitement

Il faut d'abord repérer les données nécessaires : sous forme numérique, de textes, de type logique (deux valeurs possibles, vrai ou faux), ou de type graphique (des points)...

Ces données pourront être rentrées après questionnement de l'utilisateur.

Il faut penser aux résultats intermédiaires indispensables au traitement.

Le traitement

Il s'agit de déterminer toutes les étapes des traitements à faire et donc les instructions à donner pour une exécution automatique.

Si ces instructions s'exécutent en séquence (l'une après l'autre), on parle d'algorithme séquentiel.

La sortie des résultats

Les résultats obtenus peuvent être affichés sur l'écran, ou imprimés sur papier, conservés dans un fichier...

b. Les instructions pour traiter les données.

Ce sont les constituants de base des algorithmes, qu'il faut agencer dans un ordre précis pour aboutir au résultat attendu.

Voici un exemple de programme réalisé avec le logiciel libre algobox téléchargeable à l'adresse <http://www.xmlmath.net/algobox/index.html>

```
VARIABLES
    cote EST_DU_TYPE NOMBRE
    aire EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    AFFICHER "donne le côté du carré "
    LIRE cote
    AFFICHER cote
    aire PREND_LA_VALEUR cote*cote
    AFFICHER "l'aire est "
    AFFICHER aire
FIN_ALGORITHME
```

Quel est cet algorithme ?

Il a fallu déclarer les variables *cote* (dont la valeur sera donnée par l'utilisateur) et *aire* (dont la valeur sera calculée par le logiciel).

Une variable est une "case de mémoire, un tiroir , avec une étiquette" où on range des données

Puis le programme commence :

il écrit "donne la longueur du côté"

il attend que l'utilisateur lui donne une valeur, et range cette valeur dans la variable *cote*

il écrit (affiche) cette valeur

il calcule $cote*cote$ et range le résultat dans la variable *aire*

enfin, il écrit un message et la valeur de la variable *aire*.

Avec une calculatrice TI

```

PROGRAM:AIRE
:Input "COTE ",C
:
:C*C→A
:Disp "AIRE ",A
:
:

```

Avec une calculatrice CASIO

```

"COTE = "?→C↵
C×C→A↵
"AIRE = "↵
A↵

```

l'algorithme peut se lire :

```

début
lire le côté
mettre côté *côté dans aire
afficher aire
fin

```

Sur Texas

Input s'obtient en tapant PRGRM I/O 1
 " s'obtient en tapant ALPHA "
 → s'obtient en tapant STO
 Disp s'obtient en tapant PRGRM I/O 3

Sur Casio ? s'obtient par SHIFT PRGM F4

= s'obtient par SHIFT PRGM F6 F3 F1

Remarque : sur Casio, si l'on veut faire afficher un résultat, attendre un appui sur la touche EXE pour continuer, il faut remplacer ↵ par ⏎ (Shift PRGM F5)

ex 1 : Modifier ces programmes pour qu'ils calculent en plus le périmètre du carré.

ex 2 : Modifier ces programmes pour qu'ils demandent la longueur et la largeur d'un rectangle, qu'ils calculent le périmètre et l'aire.

ex 3 : Faire calculer le périmètre d'un cercle et l'aire d'un disque de rayon donné par l'utilisateur.

ex 4 : Faire résoudre une équation du type $a + x = b$ d'inconnue x , du type $x - a = b$, du type $a \cdot x = b$, du type $a \cdot x + b = c$, du type $a \cdot x + b = c \cdot x + d$. où a et b sont des réels demandés à l'utilisateur.
 Pour les trois dernières, mettre en défaut votre algorithme.

ex 5 : Dans un triangle rectangle dont on connaît les côtés de l'angle droit, faire calculer l'hypoténuse.

ex 6 : Dans un triangle rectangle dont on connaît un côté de l'angle droit et l'hypoténuse, faire calculer le troisième côté. Mette en défaut votre algorithme.

Dans Algobox : Racine carrée d'une variable x : $\text{sqrt}(x)$.

c . Structure alternative.

dans l'exercice 6, nous avons vu que si on rentrait la longueur de l'hypoténuse inférieure à celle du côté de l'angle droit, il n'y avait pas de solution à notre problème.

Il va donc falloir tester la longueur de l'hypoténuse.

l'algorithme sera

début

lire le côté de l'angle droit a

lire la longueur de l'hypoténuse b

```

si       $b > a$       alors
|
|      calculer la longueur du 3ème côté

```

```

sinon
|
|      écrire "revoyez vos données"

```

fin

si la condition $b > a$ est réalisée, on va exécuter les instructions entre **alors** et **sinon**

Si elle n'est pas réalisée, on va exécuter les instructions écrites après **sinon**.

la condition peut être simple (une condition) ou complexe (exemple $(A < B)$ et $A > C$), $(A = B)$ ou $(A < C) \dots$)

Il se peut qu'il n'y aie rien à faire après **sinon**, alors dans ce cas, on n'écrit pas sinon.

Entre alors et sinon, il peut y avoir d'autres structures alternatives.

On peut par exemple avoir :

```

Si condition alors
  |
  | si condition alors
  | | traitement 1
  | |
  | | sinon
  | | | traitement 2
  | |
  | | sinon
  | | | traitement 3
  |
fin

```

Avec Algobox, on obtient le programme suivant (ne pas hésiter à utiliser l'aide pour découvrir les fonctionnalités).
 Pour **sinon** ne pas oublier de cocher ajouter sinon quand on écrit la condition

```

VARIABLES
  cote1 EST_DU_TYPE NOMBRE
  hypoténuse EST_DU_TYPE NOMBRE
  cote3 EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER "1er côté "
  LIRE cote1
  AFFICHER cote1
  AFFICHER "hypoténuse "
  LIRE hypoténuse
  AFFICHER hypoténuse
  SI (hypoténuse > cote1) ALORS
    DEBUT_SI
      cote3 PREND_LA_VALEUR sqrt(pow(hypoténuse,2)-pow(cote1,2))
      AFFICHER "le troisième côté = "
      AFFICHER cote3
    FIN_SI
  SINON
    DEBUT_SINON
      AFFICHER "revoyez vos données"
    FIN_SINON
FIN_ALGORITHME

```

Avec les TI :

```

PROGRAM:PYTHA
:Input "COTE1 ",
C
:Input "HYPO ",H
:If (H>C)
:Then
:√(H^2-C^2)→D
:Disp "COTE3 ",D
:Else
:Disp "REVOIR LE
S DONNEES
:End

```



Avec les CASIO

```

=====PYTHA =====
"COTE1 = "?→C
"HYPOTENUSE"?→H
If H>C
Then
√(H^2-C^2)→D
"COTE3= "
D
Else "REVOIR LES DONN
EES"
IfEnd





```





Sur les TI ,





If s'obtient avec   1
 Then s'obtient avec PRGM CTL 2
 Else s'obtient par PRGM CTL 3






> s'obtient avec   puis touche 3

Sur CASIO

If s'obtient avec    

Then    

Else    

>     

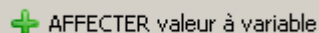
EXERCICE 7 Décrire l'algorithme qui calcule un entier n au hasard entre 1 et 6, demande un nombre entier a entre 1 et 6, et qui écrit gagné si n = a.

Sur algobox

random() donne un nombre au hasard entre 0 et 1, floor(x) donne la partie entière du nombre x.

n PREND_LA_VALEUR floor(6*random()+1)

pour affecter à n la valeur précédente il faut cliquer



sous algobox pour tester si a = b : if a == b.

Sur Texas $iPart(6*rand + 1) \rightarrow n$ (iPart se trouve dans MATH NUM3, rand se trouve dans MATH PRB1).

Sur Casio `Int (Ran# ×6+1)→N`

Int s'obtient par OPTN F6 NUM Int

Ran# s'obtient par OPTN F6 PROB F4

d) Structure répétitive 1.

Elle permet d'exécuter plusieurs fois de suite la même série d'instructions.

principe : on utilise Pour I de 1 jusqu'à n faire.

pour i de 1 à n **faire**
instructions à effectuer
fin

exemple : on veut faire sortir les résultats de 10 jets de dé.

début

Pour i de 1 à 10 faire

donner à n un entier au hasard entre 1 et 6

fin pour

fin

avec Algobox

```

1  VARIABLES
2  i EST_DU_TYPE NOMBRE
3  n EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5  POUR i ALLANT_DE 1 A 10
6  DEBUT_POUR
7  n PREND_LA_VALEUR floor(random()*6+1)
8  AFFICHER n
9  FIN_POUR
10 FIN_ALGORITHME
    
```

Avec TEXAS



```



PROGRAM:HASARD
:For(I,1,10)
:iPart(rand*6+1)
→N
:Disp N
:End
    
```



Avec CASIO



```





=====HASARD =====
For 1→I To 10
Int (Ran# ×6+1)→N
N
Next
    
```


For s'obtient avec   4


End   7




iPart   3




rand   1

For s'obtient avec     For

To  To

Next  Next

Int s'obtient par    Int

Ran# s'obtient par    Ran#

EXERCICE 8

Modifier le programme pour qu'il calcule la fréquence d'apparition du 6 dans une variable F.
sous algobox pour tester si $a = b$: `if a == b.`

EXERCICE 9

Ecrire un programme qui pour tout entier naturel n calcule $n! = 1 \times 2 \times 3 \times \dots \times n$.

EXERCICE 10

Ecrire un programme qui écrive la suite des entiers pairs.

sur texas `for(i, départ, fin,pas)`

sur Casio `for 1 → départ to fin step pas`

e) Structure répétitive 2 : tant que.

Elle permet d'exécuter plusieurs fois de suite la même série d'instructions tant qu'une condition est vérifiée.
principe : on utilise **tant que**.

Exemple :

On a une somme de 200 euros à la banque, on veut retirer chaque mois 30 €. Faire afficher chaque mois la somme restante

début

Initialisation

`somme = 200 mois = 0`

Traitement

Tant que `somme >= 30`

`somme = somme - 30`

`mois = mois + 1`

`Afficher mois somme`

fin Tant que

fin

Sous Algobox, on obtient

Sous Algobox :

```
1  VARIABLES
2  somme EST_DU_TYPE NOMBRE
3  n EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5  somme PREND_LA_VALEUR 200
6  n PREND_LA_VALEUR 0
7  TANT_QUE (somme>=30) FAIRE
8  DEBUT_TANT_QUE
9  somme PREND_LA_VALEUR somme-30
10 n PREND_LA_VALEUR n+1
11 AFFICHER "mois "
12 AFFICHER n
13 AFFICHER " il reste "
14 AFFICHER somme
15 AFFICHER " euros"
16 FIN_TANT_QUE
17
18 FIN_ALGORITHME
```

Exercice 1

En janvier 2010, je place 1000 €. Ce compte rapporte 2 d'intérêts par an.

Faire écrire chaque année le capital ainsi acquis, jusqu'à ce que le capital initial ait doublé.

Exercice 2

Ecrire un programme qui écrit les carrés d'entiers, ces carrés étant inférieurs à 1000.

Exercice 3 Jeu le compte est bon

Ecrire un algorithme du programme qui choisit d'abord un nombre entier au hasard à 3 chiffres.
Puis il demande au joueur de deviner ce nombre.

Si la réponse est exacte, il répond "Bravo" ainsi que le nombre d'essais.

Si elle est trop grande, il écrit "trop grand" ; si elle est trop petite, il écrit "trop petit" et il redemande une nouvelle réponse.

Exercice 4 Jeu le compte est bon "à l'envers".

L'utilisateur choisit un nombre entier à 3 chiffres.

Suivant le même principe, l'ordinateur doit trouver ce nombre.

Si sa réponse est trop petite, l'utilisateur répond 0 , si elle est juste, il répond 1 sinon il répond 2.