Algorithmique

I - Généralités

Définition Un *algorithme* est une suite finie d'instructions permettant la résolution systématique d'un problème donné.

Un algorithme peut-être utilisé pour

- décrire par une suite d'instructions ou de procédures la marche complète à suivre pour résoudre un problème ;
- automatiser une tâche complexe ; on sait déjà dans ce cas résoudre le problème posé et on cherche à tirer parti de moyens informatiques pour effectuer automatiquement toutes les étapes et tous les calculs intermédiaires qui permettent d'aboutir au résultat ;
- chercher la solution d'un problème ; on ne sait pas a priori résoudre le problème posé mais on peut tirer parti d'un système informatisé pour explorer l'ensemble des possibilités, et ainsi tenter de trouver la solution, ou du moins une bonne approximation de celle-ci.

Un exemple courant : notice d'utilisation ou mode d'emploi

La notice d'utilisation d'une huile teck, imprimée au dos du flacon d'huile, est donnée ci-contre.

Mode d'application

1. Vérifier que la surface est bien du teck ou un bois exotique
2. Bien agiter avant emploi
3. Imprégner le bois généreusement
4. 20 minutes après, essuyer l'excédent à l'aide d'un chiffon
5. Laisser sécher 6 heures
6. Recommencer à partir de l'étape 2

7 Laisser couler quelques gouttes d'eau sur la surface traitée
Si les gouttes perlent à la surface,
Alors le bois est correctement huilé et imperméabilisé
Sinon, recommencer à l'étape 2.

En termes algorithmiques, on appelle **initialisation** l'étape 1, et **traitement** les étapes 2 à 7.
Cette notice comporte une **boucle** sur les étapes 2 à 7.
Enfin, en terme d'algorithme, si celle-ci n'est pas traîtée par un humain *pensant*, elle comporte une (grave ?) erreur (bug ?) !

Définition Un *langage de programmation* est un ensemble d'instructions et de règles syntaxiques compréhensibles par un système automatisé (calculatrice, ordinateur, puce électronique,...).

Un *programme* est la traduction d'un algorithme dans un langage de programmation particulier.

Il existe de très nombreux langages de programmation, par exemple Basic, Fortran, Python, C, C++, Matlab, assembleur..., ainsi que ceux implantés dans les calculatrices (alors dites "programmables"...).

II - Variables

Définition *On appelle* variable *tout emplacement de la mémoire dans lequel une information peut-être stockée. Une variable est constituée de :*

- *un nom qui permet à l'ordinateur de la localiser dans sa mémoire (une lettre :* `a`*,* `b`*, ..., x, ..., ou plusieurs :* `ma_variable`*, ...)*
- *une valeur : l'information (un nombre, un texte, ...) qu'elle contient.*

La valeur d'une variable peut bien sûr changer au cours de l'exécution de l'algorithme. Une **affectation** consiste à attribuer une `valeur` à une `variable`, ou à en modifier la valeur.

valeur→`variable`	ou	`variable`=*valeur*

III - Structures dans les algorithmes

1) Boucles itératives

Une boucle permet de répéter un ensemble d'instructions un nombre fixé de fois.

```
Pour variable de début à Fin
  instructions 1
  instructions 2
  ...
Fin Pour
```

Exercice 1

a) Ecrire un algorithme qui calcule et affiche la suite des carrés des nombres entiers de 1 à 10.

b) Modifier cet algorithme pour qu'il calcule et affiche la somme des carrés des entiers de 1 à 10.

2) Tests et instructions conditionnelles

Un test est une comparaison entre la valeur d'une variable et une valeur donnée, ou entre les valeurs de deux variables.

Un test a deux résultats possibles : 0 (faux), ou 1 (vrai).
Dans une structure conditionnelle, les instructions ne sont effectuées que *si* le test indiqué est vrai.

```
Si test
  instructions 1
  instructions 2
  ...
Fin Si
```

Exercice 2 Ecrire un algorithme qui demande un nombre à l'utilisateur et affiche en résultat si le nombre est positif ou négatif.

3) Boucles conditionnelles

Une boucle conditionnelle permet de répéter une série d'instructions sans connaître a priori le nombre d'itérations.
La boucle est répété *tant que* le test indiqué est vrai.

```
Tant que test
  instructions 1
  instructions 2
  ...
Fin Tant que
```

Exercice 3 Ecrire un algorithme qui demande un nombre entier à l'utilisateur et compte à rebours.

Exercice 4 Ecrire un algorithme qui demande successivement des nombres et qui s'arrête lorsque l'utilisateur entre le chiffre 5.

IV - Corrigés : Algorithmes et programmes des exercices

Exercice 1

a) Affichage des 10 premiers entiers :

Algorithme
```
Pour I de 1 à 10
   Afficher I*I
Fin Pour
```

Programme TI
```
For (I,1,10)
Disp I*I
End
```

Programme Casio
```
For 1→I To 10 ↵
I*I◢
Next
```

Programme Python
```
for i in range(1,11):
   print(i*i)
```

b) Calcul de la somme des carrés des 10 premiers entiers :

Algorithme
```
0→S
Pour I de 1 à 10
   S+I*I→S
Fin Pour
Afficher S
```

Programme TI
```
0→S
For (I,1,10)
S+I*I→S
End
Disp S
```

Programme Casio
```
0→S
For 1→I To 10 ↵
S+I*I→S ↵
Next
S◢
```

Programme Python
```
S=0
for i in range(1,11):
   S=S+i*i
print(S)
```

Exercice 2 Test : nombre positif ?

Algorithme
```
Lire A
Si A>0
   Afficher "A positif"
Sinon
   Afficher "A négatif"
Fin Si
```

Programme TI
```
Prompt A
If A≥0
Then
Disp "A positif"
Else
Disp "A négatif"
End
```

Programme Casio
```
"A="?→A ↵
If A>0 ↵
Then "A positif" ↵
Else "A négatif" ↵
IfEnd
```

Programme Python
```
A=input("A ?")
if A≥0:
   print("A
positif")
else:
   print("A
negatif")
```

Exercice 3 Compte à rebours :

Algorithme
```
Lire N
Tant que N>0
   N-1→N
   Afficher "N"
Fin Tant que
```

Programme TI
```
Prompt N
While N>0
N-1→N
Disp N
End
```

Programme Casio
```
"N="?→N
While N>1 ↵
N-1→N◢
WhileEnd
```

Programme Python
```
N=input("N ?")
While N>0:
   N=N-1
   print(N)
```

Exercice 4 Boucle arrêtée par l'utilisateur

Algorithme
```
Lire N
Tant que N≠5
   Lire N
Fin Tant que
```

Programme TI
```
Prompt N
While N≠5
Prompt N
End
```

Programme Casio
```
"N="?→N
While N≠5 ↵
"N="?→N
WhileEnd
```

Programme Python
```
N=input("N ?")
While not(N==5):
   N=input("N
?")
```

V - Jeu du nombre mystérieux

Ce jeu se joue à deux personnes de la manière suivante.

Un des deux joueurs choisit un nombre entier au hasard compris entre 1 et 100. Le but du deuxième joueur est de trouver ce nombre. Pour cela il propose un nombre au premier joueur qui lui fournit une des trois réponses :

— *Gagné*, si le nombre proposé est le bon ;
— *Trop grand*, si le nombre proposé est plus grand que le nombre mystérieux ;
— *Trop petit*, si le nombre proposé est plus petit que le nombre mystérieux ;

Si le nombre proposé n'est pas le bon, le deuxième joueur propose un autre nombre, et le jeu se poursuit jusqu'à ce qu'il trouve le nombre exact.

Le but du jeu est de trouver le nombre mystérieux avec le moins de tentatives possible.

1) L'ordinateur fait deviner. Le programme dans lequel l'ordinateur est le joueur choisissant un nombre au hasard compris entre 1 et 100, et l'utilisateur est le joueur qui doit trouver ce nombre.

Le programme doit aussi afficher le nombre de tentatives utilisées.

Lire attentivement l'algorithme suivant (ou un des programmes), bien suivre et comprendre la succession d'instruction, et indiquer le rôle de chacune des variables `M`, `C`, `N`.

Algorithme

```
M prend une valeur aléatoire entre 0 et 100
C prend la valeur 1
Afficher "Entrer un nombre"
Lire N
Tant que N≠M
   Si N<M alors
      Afficher "Trop petit"
   Sinon
      Afficher "Trop grand"
   Fin Si
   C prend la valeur C+1
   Afficher "Entrer un nombre"
   Lire N
Fin Tant que
Afficher "Gagne en",C," coups"
```

Programme TI

```
randInt(0,100)→M
1→C
Prompt N
while N≠M
if N<M
Then
Disp "Trop petit"
Else
Disp "Trop grand"
End
C+1→C
Prompt N
End
Disp "Gagne en",C," coups"
```

Programme Casio

```
Int(100×Ran#+1)→M↵
1→C ↵
"N="?→N ↵
While N≠M ↵
If N<M ↵
Then "Trop petit" ↵
Else "Trop grand" ↵
IfEnd ↵
C+1→C ↵
"N="?→N ↵
WhileEnd ↵
"Gagne" ↵
"Nombre de coups=" ↵
C◢
```

Programme Python

```
import random

M=random.randint(0,100)
N=input('Entrer un nombre: ')

C=1
while N!=M:
   N=input('Entrer un nombre: ')
   if N<M:
      print("Trop petit")
   else
      print("Trop grand")
   C=C+1
print("Gagne en ",C," coups")
```

2) L'ordinateur doit deviner. Ecrire un programme pour ce jeu avec les rôles inversés : vous pensez à un nombre entier compris entre 1 et 100, et l'ordinateur doit le trouver. Essayer de trouver une stratégie pour que l'ordinateur trouve ce nombre avec le moins de coups possible.

VI - Exercices

Exercice 5 Ecrire un algorithme qui demande à l'utilisateur un nombre entier n et calcule et affiche la somme $1+2+3+\cdots+n$.

Exercice 6 *Balayage de valeurs*
Ecrire un algorithme qui donne une valeur approchée à 10^{-2} près de la solution positive de l'équation $x^2=2$. L'algorithme procédera par balayage à partir de 0, c'est-à-dire en testant $0,\ 0,01,\ 0,02,\ldots$

Modifier cet algorithme pour obtenir une solution approchée à 10^{-3}, puis 10^{-4} près.

Exercice 7 *Distributeur de billets*
Ecrire un algorithme qui demande un montant N en euros et qui calcule et affiche le nombre minimal de billets de 20 euros, de 10 euros et de 5 euros à fournir pour faire le montant N.

Exercice 8 La population d'une ville augmente de 4 % par an. Ecrire un algorithme qui détermine le nombre d'années au bout duquel la population aura doublé.