1 Algorithmique – Cours

1 Objectifs

1a Lire un algorithme

Un algorithme est donné et il faut savoir indiquer :

- les variables d'un algorithme ;
- les entrées et les sorties d'un algorithme ;
- les instructions d'affectation ;
- les instructions conditionnelles ;
- les structures itératives.

1b Exécuter un algorithme

Un algorithme est donné et il faut savoir l'exécuter, c'est à dire détailler ce qu'il fait. On peut utiliser un tableau donnant les valeurs successives des variables.

1c Comprendre un algorithme

Un énoncé précise un ou des objectifs pour un algorithme et un algorithme est proposé.

Il faut savoir déterminer si cet algorithme atteindra l'objectif fixé et non, et savoir justifier sa réponse.

1d Concevoir un algorithme

Un énoncé donne les ojectifs d'un algorithme et il faut le détailler.

Il peut être demandé de programmer, mais pas forcément.

2 Entrée, traitement, sortie

Un algorithme comporte souvent trois parties.

- **Les entrées :** l'algorithme lit des données saisies par l'utilisateur.
- **Le traitement :** cette partie peut contenir plusieurs instructions, des calculs, etc.
- **La sortie :** c'est le résultat affiché par l'algorithme.

Exemple : voici un algorithme permettant de calculer le périmètre d'un rectangle à partir de la donnée de sa largeur a et de la sa longueur b.

	Algorithme
Entrées	Lire a et b
Traitement	Stocker $2 \times (a + b)$ dans p
Sortie	Afficher p

3 Entrées, lecture des variables

Exemple : un élève écrit l'algorithme ci-dessous pour calculer le volume d'un cylindre à l'aide de la formule : $V = \pi r^2 h$.

	Algorithme
Entrées	Lire r
	Lire h
	Lire V
Traitement	$V \leftarrow \pi r^2 h$
Sortie	Afficher V

Remarque : cet algorithme est correct, mais contient une instruction inutile et même aberrante qui est « Lire V ».

À retenir

Cela ne sert à rien de lire une variable qui va être calculée par l'algorithme.
Si on le fait, cela veut dire qu'on donne un résultat avant que l'algorithme le calcule, or c'est l'algorithme qui doit effectuer le calcul et pas l'utilisateur.

4 Variables et affectation

4a Exemple, vocabulaire, propriété

Exemple : voici un algorithme dans la première colonne du tableau ci-dessous et dans la deuxième colonne, les valeurs successives de a.

Algorithme	a
Stocker 5 dans a	5
Stocker 8 dans a	8
Afficher a	8

Remarque

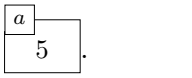
Si on veut stocker 5 dans a, on peut l'écrire de différentes manières :

- avec la calculatrice : `5→A`
- avec AlgoBox : `a PREND_LA_VALEUR 5`
- en Python : `a=5`
- dans un algorithme : $a \leftarrow 5$

Vocabulaire à retenir

On dit que a est une **variable**.
Le fait de stocker 5 dans a ou que a prenne la valeur 5 s'appelle une **affectation**.

Précision sur variable et affectation

- Dans l'algorithme ci-dessus, quand on exécute l'instruction « Stocker 5 dans a », le contenu de la variable a est 5 et on peut la représenter ainsi : 

.
- Quand on exécute ensuite l'instruction « Stocker 8 dans a », on efface le contenu de la variable a et on le remplace par 8, ce qui donne : 

.

À retenir :

- Quand on affecte une valeur à une variable, l'ancienne valeur est **remplacée** par la nouvelle valeur.
- Tant que la valeur d'une variable n'est pas modifiée par une affectation, cette variable garde sa valeur précédente.

4b Définition d'une variable

Selon les langages une variable peut être définie de différentes façons.

- Sur une calculatrice, une variable, comme `A` est toujours définie. Si on n'a jamais rien affecté à cette variable, par exemple quand la calculatrice est neuve, son contenu est zéro.
- Dans AlgoBox, comme en C ou en Java, avant de saisir un programme on doit « déclarer » toutes les variables qui vont être utilisées dans ce programme.
- Dans le langage Python, une variable est définie par sa première affectation. Si on utilise une variable dans un calcul et qu'on ne lui a pas encore affecté une valeur, on obtient un message d'erreur.

4c On ne peut pas affecter n'importe quoi à n'importe quoi

Un premier exemple

Avec une calculatrice, si l'on saisit `A→5` et qu'on appuie sur entrer, on obtient un message d'erreur, alors que la commande `5→A` est correcte.

De même en Python, si l'on exécute à la console `5=a`, qui veut dire $5 \leftarrow a$ on obtient un message d'erreur alors que la commande `a=5` est correcte.

Deuxième exemple

Avec une calculatrice on obtient un message d'erreur avec la commande `7→A+B`.

De même en Python, si l'on exécute à la console `a+b=7`, qui veut dire $a + b \leftarrow 7$, on obtient un message d'erreur.

À retenir

On ne peut stocker un nombre ou une expression que dans une variable, ce qui revient à dire que on ne peut pas affecter quelque chose à un nombre fixé ou à une expression, seule une variable peut prendre la valeur d'un nombre ou d'une expression.

5 Instruction conditionnelle

Exemple

	Algorithme
1	a prend la valeur d'un nombre aléatoire entre 0 et 200
2	Si $a < 100$
3	alors afficher « Inférieur à 100 »
4	sinon afficher « Supérieur ou égal à 100 »

Vocabulaire : dans l'algorithme ci-dessus,

c'est l'ensemble des lignes de commande 2, 3, 4 qui constituent une instruction conditionnelle.

6 Calcul itératif

Exemple

Algorithme 1	**Algorithme 2**
k prend la valeur 0	
Tant que $k \leqslant 4$	Pour $k = 0$ jusqu'à $k = 4$ de 2 en 2
Afficher « Coucou »	Afficher « Coucou »
$k \leftarrow k + 2$	
Fin du Tant que	Fin du Pour

Les deux algorithmes ci-dessus sont équivalents :

- la variable k prend successivement les valeurs 0, puis 2, puis 4 ;
- ainsi le message « Coucou » s'affiche trois fois.

Vocabulaire : il s'agit dans les deux cas d'une **structure itérative**.

Trois paramètres sont importants.

- **Initialisation :** c'est la première valeur de k,
  - donnée dans l'algorithme 1 par *k prend la valeur 0*
  - donnée dans l'algorithme 2 par *Pour $k = 0$ ...*
- **Arrêt de l'algorithme :** il est imposé par la dernière valeur de k,
  - donnée dans l'algorithme 1 par *Tant que $k \leqslant 4$*, ce qui provoque l'arrêt dès que $k > 4$,
  - donnée dans l'algorithme 2 par *jusqu'à $k = 4$*.
- **Le pas :** la variable k augmente de 2 en 2 à chaque étape, ce nombre 2 est le « pas »,
  - donné dans l'algorithme 1 par $k \leftarrow k + 2$,
  - donné dans l'algorithme 2 par *de 2 en 2*.