Algorithmes – Le savoir minimum

1 Traitement séquentiel

Un algorithme est une suite finie d'opérations ou d'instructions permettant de résoudre un problème ; si elles sont traitées les unes à la suite des autres, on parle de `traitement séquentiel`.

Exemple 1

Problème : chercher l'image d'un nombre x par la fonction $f : x \longmapsto x^2 + 3x - 5$.

Variables	x, y : nombres
Initialisation	entrer la valeur de x
Début	y prend la valeur $x^2 + 3x - 5$
	afficher « l'image de x est »
	afficher y
Fin	

Exemple 2

Problème : chercher le terme de rang n de la suite (u_n) définie sur $\mathbf{N}$ par $u_n = 3n^2 + 2n - 1$.

Variables	n, u : nombres
Initialisation	entrer la valeur de n
Début	u prend la valeur $3n^2 + 2n - 1$
	afficher « le terme de rang n de la suite est »
	afficher u
Fin	

2 Traitement conditionnel

Le `si ... alors ... sinon ...` permet de faire une action si une condition est vérifée, et d'en faire une autre dans le cas contraire.
Le `sinon` est facultatif.

Exemple 1

Problème : on veut savoir si un nombre est divisible par 3.

Variables	n : nombre
Initialisation	entrer la valeur de n
Début	si $\dfrac{n}{3}$ est un nombre entier
	alors afficher « ce nombre est divisible par 3 »
	sinon afficher « ce nombre n'est pas divisible par 3 »
	fin si
Fin	

Pour voir si le nombre $\dfrac{n}{3}$ est entier, on utilise la fonction « partie entière », notée E, en codant ainsi : si $E\left(\dfrac{n}{3}\right) = \dfrac{n}{3}$.

Exemple 2

Problème : résoudre une équation du second degré à coefficients réels.

Variables a, b, c, d, r, s : nombres
Initialisation entrer les valeurs de a, de b et de c
Début d prend la valeur $b^2 - 4ac$
si $d < 0$
alors afficher « cette équation n'a pas de solution réelle »
sinon r prend la valeur $\dfrac{-b+\sqrt{d}}{2a}$
s prend la valeur $\dfrac{-b-\sqrt{d}}{2a}$
afficher « cette équation a pour solutions »
afficher r et s
fin si
Fin

On peut coder cet algorithme sur sa calculatrice.

Sur Texas Instruments

```
: Input "A :", A
: Input "B :", B
: Input "C :", C
: B*B - 4*A*C → D
: Disp "DELTA :", D
: If D < 0
: Then
: Disp "PAS DE RACINE"
: Else
: (-B − √(D))/(2A) → R
: (-B + √(D))/(2A) → S
: Disp "RACINES :"
: Disp R, S
: End
```

Sur CASIO

```
"A"? → A ↵
"B"? → B ↵
"C"? → C ↵
B*B - 4*A*C → D ↵
"DELTA" ↵
D ◢
If D < 0 ↵
Then "PAS DE RACINE" ↵
Else (-B − √(D))/(2A) → R ↵
(-B + √(D))/(2A) → S ↵
"RACINES" ↵
R ◢
S ↵
IfEnd
```

3 Traitement itératif - Boucle `pour`

Quand on veut répéter une ou plusieurs instructions, on utilise des boucles.
On utilise la boucle `pour ... variant de ... à ...` lorsque l'on connait le nombre de tours de boucle que l'on veut faire.

Exemple 1

Problème : chercher le terme de rang n d'une suite (u_n) définie par une relation de récurrence.

Soit (u_n) la suite définie par $u_0 = 2$ et, pour tout n, $u_{n+1} = 3u_n + 1$.
On veut calculer le terme de rang n.

Variables	n, i, u : nombres
Initialisation	entrer la valeur de n
	u prend la valeur 2
Début	pour i variant de 1 à n
	u prend la valeur $3u + 1$
	fin pour
	afficher u
Fin	

Dans cet algorithme, la variable de boucle i est gérée automatiquement par la structure `pour` : à chaque tour de boucle, la variable de boucle i est incrémentée (augmentée) de 1.

On peut rajouter une ligne à l'intérieur de la boucle pour faire afficher tous les termes de la suite de rangs 1 à n ; l'affichage du terme de rang n en dernière ligne est alors superflu.

Variables	n, i, u : nombres
Initialisation	entrer la valeur de n
	u prend la valeur 2
Début	pour i variant de 1 à n
	u prend la valeur $3u + 1$
	afficher u
	fin pour
Fin	

Exemple 2

Problème : on lance n fois deux dés bien équilibrés et on fait la somme des numéros situés sur les faces supérieures.

Variables	n, i, r : nombres
	ℓ : liste
Initialisation	entrer la valeur de n
	on met à 0 tous les termes de la liste entre 2 et 12
Début	pour i variant de 1 à n
	r prend la valeur aléa$(1,6)$ + aléa$(1,6)$
	on ajoute 1 au terme de rang r de la liste ℓ
	fin pour
	afficher la liste ℓ entre 2 et 12
Fin	

Dans cet algorithme, l'instruction aléa$(1,6)$ donne un nombre entier au hasard entre 1 et 6.
Ajouter 1 au terme de rang r de la liste ℓ se code souvent : $\ell[r]$ reçoit $\ell[r] + 1$.
L'affichage de la liste ℓ peut aussi nécessiter l'utilisation d'une boucle :

```
pour i variant de 2 à 12
    afficher ℓ[i]
fin pour
```

4 Traitement itératif - Boucle `tant que`

Quand on ne connait pas le nombre de tours de boucle qu'il faut faire, on utilise un autre type de boucle, la boucle `tant que`.
Dans ce cas, il faut impérativement modifier la condition d'entrée dans la boucle (ce qui est après le `tant que`) à l'intérieur de la boucle, sinon on entre dans une boucle infinie !

Exemple

Problème : déterminer le plus petit entier n_0 tel que, à partir de n_0, tous les termes d'une suite (u_n) sont plus grands qu'un nombre N.

Soit (u_n) la suite définie par $u_0 = 2$ et, pour tout n, $u_{n+1} = 3u_n + 1$.

Variables	N, i, u : nombres
Initialisation	entrer la valeur de N
	u prend la valeur 2
	i prend la valeur 0
Début	tant que $u \leqslant N$
	i prend la valeur $i + 1$
	u prend la valeur $3u + 1$
	fin tant que
	afficher i
Fin	

Il faut bien sûr des conditions à la suite (u_n) pour que le problème soit possible : dans cet exemple, la suite est croissante et a pour limite $+\infty$, donc la condition d'arrêt du `tant que` sera vérifiée à partir d'un rang n.

Il suffit de rajouter une ligne dans la boucle pour faire afficher, à chaque étape, le terme de rang i :

Variables	N, i, u : nombres
Initialisation	entrer la valeur de N
	u prend la valeur 2
	i prend la valeur 0
Début	tant que $u \leqslant N$
	i prend la valeur $i + 1$
	u prend la valeur $3u + 1$
	afficher u
	fin tant que
	afficher i
Fin	