

Algorithmes : Structures itératives : Boucles.

A. Quand on connaît le nombre de répétitions

Exemple

L'algorithme ci-contre donne le résultat obtenu en augmentant de 4 % un montant S en euros (rappel : le résultat est $S \times (1 + 4/100) = S \times 1,04$). On dépose sur un livret d'épargne un montant S et il augmente chaque année de 4 %. On veut connaître le montant obtenu au bout de 10 ans. On répète pour cela 10 fois de suite l'augmentation de 4 %. On obtient l'algorithme 5 :

i prend la valeur 1, l'instruction s'effectue
i prend la valeur 2, l'instruction s'effectue
 etc ... *i* prend la valeur 10, l'instruction s'effectue.

C'est fini.

On parle d'une boucle et l'on dit que l'on passe « 10 fois dans la boucle ».

VARIABLE : S nombre
 ENTRÉE : Saisir S
 TRAITEMENT : S prend la valeur $S \times 1,04$
 SORTIE : Afficher S

Algorithme 5

VARIABLE : S nombre, i nombre
 ENTRÉE : Saisir S
 TRAITEMENT :

Pour i allant de 1 à 10 Faire
 | S prend la valeur $S \times 1,04$
FinPour

SORTIE : Afficher S

Bilan


La répétition d'une suite d'instructions un certain nombre de fois s'appelle une **boucle** ou une **structure itérative**.

Une répétition 100 fois d'une suite d'instructions s'écrit :

Pour i allant de 1 à 100 Faire
 | Suite d'instructions
FinPour

Dans l'algorithme ci-dessus, i est le **compteur** de la boucle. Il est **incrémenté** (augmenté) de 1 à chaque passage dans la boucle. On peut lui donner un autre nom que i . Il peut aller de 4 à 200, ou même de 1 à n où n est une variable contenant un nombre entier (au moins égal à 1).

Programmes écrits à partir de l'algorithme 5 dans différents langages

AlgoBox	Scratch	TI
<pre> VARIABLES - S EST_DU_TYPE NOMBRE - i EST_DU_TYPE NOMBRE DEBUT_ALGORITHME - LIRE S - POUR i ALLANT DE 1 A 10 - DEBUT_POUR - S PREND_LA_VALEUR S*1.04 - FIN_POUR - AFFICHER S FIN_ALGORITHME </pre>		<pre> :Input "S=",S :For(I,1,10) :S*1.04→S :End :Disp S </pre>
<pre> saisir(S); pour k de 1 jusque 10 faire S:=S*1.04; fpour; afficher(S); </pre> <p>Attention. Le nom i est réservé et ne peut être utilisé.</p>	<pre> 1 S=input("S="); 2 for i=1:10 3 S=S*1.04; 4 end 5 disp(S); </pre>	<pre> "S="?→S For 1→I To 10 S×1.04→S Next I S </pre>

Exercice

Quels affichages obtient-on avec les algorithmes ci-contre ? On dressera pour chacun un état des variables dans la boucle :

Boucle	$i: 1$	$n:$
	$i: 2$	$n:$

Algorithme 6

```
VARIABLES :  $n, i$  nombres
TRAITEMENT :
  Pour  $i$  allant de 1 à 9 Faire
  |  $n$  prend la valeur  $7 \times i$ 
  SORTIE : Afficher  $n$ 
FinPour
```

Algorithme 7

```
VARIABLES :  $n, i$  nombres
INITIALISATION :
   $n$  prend la valeur 1
TRAITEMENT :
  Pour  $i$  allant de 1 à 9 Faire
  |  $n$  prend la valeur  $2 \times n$ 
  FinPour
SORTIE : Afficher  $n$ 
```

B. Quand on connaît un test d'arrêt mais pas le nombre de répétitions

Exemple : On dépose sur un livret d'épargne une somme inférieure ou égale à 5 000 €. Elle augmente chaque année de 4 %. On veut savoir au bout de combien d'années elle aura dépassé 6 000 €.

Il s'agit encore d'une boucle mais on ne connaît pas le nombre de passages dans la boucle.

On va répéter l'augmentation jusqu'à ce que le montant devienne supérieur à 6 000 €.

Il faudra créer une variable qui compte le nombre de passages dans la boucle c'est-à-dire le nombre d'années. Cette variable s'appelle un compteur. On a deux structures possibles.

Algorithme 8 : Tant que ...

```
VARIABLES :  $S, n$  nombres
ENTRÉE : Saisir  $S$ 
INITIALISATION :  $n$  prend la valeur 0
TRAITEMENT :
  Tantque  $S \leq 6000$  Faire
  |  $S$  prend la valeur  $S \times 1,04$ 
  |  $n$  prend la valeur  $n + 1$ 
FinTantque
SORTIE : Afficher  $n$ 
```

Algorithme 9 : Répéter ... Jusqu'à

```
VARIABLE :  $S, n$  nombres
ENTRÉE : Saisir  $S$ 
INITIALISATION :  $n$  prend la valeur 0
TRAITEMENT :
  Répéter
  |  $S$  prend la valeur  $S \times 1,04$ 
  |  $n$  prend la valeur  $n + 1$ 
Jusqu'à  $S > 6000$ 
SORTIE : Afficher  $n$ 
```

Bilan

Pour écrire une boucle avec un test d'arrêt on a deux structures possibles :

Tantque condition Faire

| Suite d'instructions

FinTantque

On commence par tester si la condition est vraie.

Si elle est vraie, on exécute la suite d'instructions et on recommence.

Si elle est fautive, on s'arrête et on sort de la boucle.

Répéter

| Suite d'instructions

Jusqu'à condition


Après avoir exécuté la suite d'instructions, on teste si la condition est vraie. Si elle est vraie, on s'arrête et on sort de la boucle ; si elle est fautive, on recommence la suite d'instructions.

Avec une structure « répéter jusqu'à », la suite d'instructions est exécutée au moins une fois.

Avec la structure « Tant que », elle peut ne pas l'être puisque l'on peut ne pas entrer dans la boucle.

La boucle « Tant que » de l'algorithme 8 dans différents langages (programmes complets sur le site) @

La boucle « Répéter jusqu'à » de l'algorithme 9 dans différents langages (programmes complets sur le site)

<p>Algobox</p> <pre> TANT_QUE (S<=6000) FAIRE DEBUT_TANT_QUE S PREND_LA_VALEUR S*1.04 n PREND_LA_VALEUR n+1 FIN_TANT_QUE </pre>	<p>Calculatrice TI</p> <pre> :While S<=6000 :S*1.04→S :N+1→N :End </pre>	<p>Scratch</p> 
<p>Scilab</p> <pre> 2 while S<=6000 3 S=S*1.04; 4 n=n+1; 5 end </pre>	<p>Calculatrice Casio</p> <pre> While S<=6000⇐ S×1.04→S⇐ N+1→N⇐ WhileEnd⇐ </pre>	<p>Xcas</p> <pre> repete S:=S*1.04; n:=n+1; jusqua S>6000; </pre>
<p>Xcas</p> <pre> tantque S<=6000 faire S:=S*1.04; n:=n+1; ftantque; </pre>		<p>Calculatrice TI</p> <pre> :Repeat S>6000 :S*1.04→S :N+1→N :End </pre>

Exercices @

- 1 Vérifier en dressant l'état des variables pour une somme S de 4 800 € que les algorithmes 8 et 9 ci-dessus font bien afficher le même nombre d'années n .
- 2 Modifier les algorithmes 8 et 9 pour faire afficher le nombre d'années nécessaire pour doubler la somme S de départ.
- 3 S'inspirer des algorithmes ci-dessus pour demander un nombre entier naturel à l'utilisateur et renvoyer le plus petit entier naturel n tel que 2^n soit plus grand que ce nombre.

[Voir exercice résolu 4](#)