

# Introduction à l'informatique

## Chapitre 4 - Structures Itératives (boucles)

R. Groult, F. Levé

UFR des Sciences  
Université de Picardie Jules Verne, Amiens

mardi 27 septembre 2011

## La boucle « pour »

### But

Exécuter une même suite d'instructions plusieurs fois.

On connaît le nombre de fois à les exécuter : **boucle définie**.

### Syntaxe

```
pour ( variable allant de val_init à val_finale pas val_pas ) faire  
    instructions ;
```

**finpour**

### Fonctionnement

- *variable* prend successivement les valeurs comprises entre *val\_init* et *val\_finale* de *val\_pas* en *val\_pas* ;
- les instructions sont exécutées autant de fois.

## Exemple

```

pour (i allant de 2 à 4 pas 1) faire
  x ← 2*i ;
finpour
x ← x+3 ;

```

## Trace d'exécution

instruction	i	x
n←2;		
pour (...)	2	
x ← 2*i ;		4
pour (...)	3	
x ← 2*i ;		6
pour (...)	4	
x ← 2*i ;		8
x ← x+3 ;		11

## La boucle « pour » : remarques

- La variable et les valeurs initiale, finale et du pas doivent être du type nombre (entier ou réel)
- Les valeurs doivent être cohérentes par rapport au pas.
- La *variable* est souvent appelée « compteur ».
- Ce compteur ne DOIT PAS être modifié dans la boucle.
- Il vaut mieux ne pas utiliser ce compteur en dehors de la boucle.
- Les instructions à exécuter dans une boucle peuvent, bien sûr, contenir des saisies, des instructions conditionnelles, ou d'autres boucles.

## La boucle « pour » n'est pas toujours adaptée

### Question

Comment faire si l'on ne connaît pas le nombre d'itérations ? On ne peut plus utiliser la boucle « pour ».

### Exemple

- on souhaite afficher les valeurs successives des carrés de nombres entiers croissants à partir de 1, mais seulement tant que la valeur du carré ne dépasse pas 30.

*on initialise une variable  $n$  avec la valeur 1*

*on calcule le carré de  $n$*

***tant que*** la valeur du carré qu'on vient de calculer est inférieure à 30

*on affiche cette valeur*

*on calcule le carré du nombre entier suivant*

## Algorithme avec boucle « tant que »

on initialise une variable n avec la valeur 1

on calcule le carré de n

**tant que** la valeur du carré que l'on  
vient de calculer est  
inférieure à 30

on affiche cette valeur

on calcule le carré du nombre entier suivant

$n \leftarrow 1$  ;

carre  $\leftarrow n * n$  ;

**tant que** (carre  $\leq 30$ ) **faire**

écrire carre ;

$n \leftarrow n + 1$  ;

carre  $\leftarrow n * n$  ;

**fin tant que**

## Boucle « tant que »

### But

Exécuter une même suite d'instructions plusieurs fois, tant qu'une condition est vérifiée (on peut ne pas prévoir/déterminer le nombre d'itérations).

### Syntaxe

**tant que** ( condition ) **faire**

instructions à exécuter ;

**fintantque**

### Principe

Tant que la condition (de type booléen) est vérifiée, les instructions sont exécutées.

## Boucle « tant que » : fonctionnement

**tant que** ( condition ) **faire**

instructions à exécuter ;

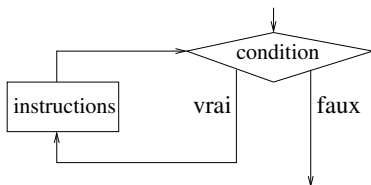
**fin tant que**

### Fonctionnement

La condition est testée

- si elle est vraie, les instructions sont exécutées, puis on recommence au test de la condition
- sinon la boucle s'arrête

### Diagramme



Remarque : le « corps de la boucle » n'est donc pas exécuté si la condition (« test de la boucle ») est fausse dès le départ.

## Exemple

## Exemple

```

01 entier n, p, i ;
02 début
03   n ← 6 ;
04   p ← 1 ;
05   i ← 0 ;
06   tant que (p ≤ n) faire
07     p ← p*2 ;
08     i ← i+1 ;
09   fin tant que
10   écrire p ;
11 fin

```

ligne	n	p	i	test	aff.
01	?	?	?		
03	6				
04		1			
05			0		
06				vrai	
07		2			
08			1		
06				vrai	
07		4			
08			2		
06				vrai	
07		8			
08			3		
06				faux	
10					8

## Éviter les boucles infinies

- Si la condition est TOUJOURS vraie, la boucle ne s'arrête jamais : on appelle cela une **boucle infinie**.
- Afin d'éviter les boucles infinies, il faut vérifier que la condition puisse prendre la valeur *faux*.

- **Exemple 1**

```
x ← 1 ;
tant que (x != 10) faire
    x ← 2*x ;
fintantque
```

- **Exemple 2**

```
x ← 1 ;
tant que (x != 10) faire
    écrire x ;
fintantque
```

- **Exemple 3**

```
x ← 1 ;
n ← 10 ;
tant que (x != n) faire
    x ← n + 1 ;
fintantque
```

## Attention aux valeurs limite

Conseil : faire une trace d'exécution (au brouillon) afin de :

- vérifiez tous les cas qui pourraient causer un mauvais comportement de la boucle, et en particulier vérifier les *cas limite*
- vérifiez que votre boucle peut s'exécuter (votre test doit pouvoir être vrai).  
Par exemple ( $a > 0$  et  $0 > a$ ) est toujours faux.
- testez votre boucle sur quelques valeurs pour vérifier son fonctionnement

Lorsque votre condition est une expression booléenne composée, analysez les cas possibles de sortie de boucle.

Par exemple « tantque ( $a > 0$  et  $0 > b$ ) faire ... » : que dire de la valeur de  $a$  et de la valeur de  $b$  une fois sortie de la boucle ?

# Simuler une boucle « pour » avec une boucle « tant que »

## pour

```
pour (i allant de init à finale pas pas) faire  
    instructions à exécuter ;  
finpour
```

## tant que

```
i ← init ; // initialiser le compteur de boucle i à init  
tant que ( $i \leq \textit{finale}$ ) faire  
    instructions à exécuter ;  
    i ← i + pas ; // incrémenter le compteur de boucle i de pas  
fantantque
```

## Exemple 1

### Exemple

Écrire un algorithme qui, pour chacun des 100 entiers saisis par l'utilisateur, affiche s'il est pair ou impair.

### Algorithme

entier val, i;  
début

**pour (i allant de 1 à 100 pas 1) faire**

lire val ;

si ((val mod 2) == 0) alors

écrire "Le nombre " + val + " est pair" ;

sinon

écrire "Le nombre " + val + " est impair" ;

finsi

**finpour**

fin

## Exemple 2

Écrire un algorithme affichant le nombre d'entiers pairs parmi 100 entiers saisis par l'utilisateur.

```

01 entier val, nbr, i;
02 début
03   nbr ← 0;
04   pour (i allant de 1 à 100 pas 1) faire
05     lire val;
06     si ( (val mod 2) == 0 ) alors
07       nbr ← nbr + 1;
08   finsi
09 finpour
10   écrire nbr + " pair(s)";
11 fin
  
```

ligne	test	val	nbr	i	affichage
01		?	?	?	
03			0		
04				1	
05		4			
06	vrai				
07			1		
04				2	
05		7			
06	faux				
04				3	
..	...	...	...	...	
07			14		
..	...	...	...	...	
04				100	
05		2			
06	vrai				
07			15		
08					15 pair(s)

## Exemple 3

Écrire un algorithme affichant le nombre d'entiers pairs parmi les entiers saisis par l'utilisateur jusqu'à ce qu'il saisisse une valeur négative ou nulle.

### Algorithme

entier val, nbr ;

début

  nbr ← 0;

  lire val;

**tantque (val > 0) faire**

    si ( (val mod 2) == 0 ) alors

      nbr ← nbr + 1;

    finsi

**lire val;**

**fintantque**

  écrire nbr + " pair(s)";

fin

## Méthodologie pour l'écriture d'une boucle

- Faire soi-même le calcul sur un ou plusieurs exemples judicieusement choisis : cas généraux, cas particuliers
- Repérer une action répétitive, donc une boucle
- Choisir entre boucle « pour » et boucle « tant que »  
Question : peut-on prévoir/déterminer le nombre d'itérations ?
  - si oui, boucle « pour »
  - si non, boucle « tant que »
- Écrire l'action répétitive et l'instruction de boucle choisie
- Initialiser les variables utilisées (si nécessaire)
- Écrire les conditions d'arrêt
- Écrire l'incrémentación de la variable de contrôle s'il y en a
- Exécuter pour les cas extrêmes et au moins un cas « normal »
- (Essayer de prouver que cela fonctionne dans tous les cas)