ESATIC ECOLE SUPERIEURE AFRICAINE DES TIC	Année scolaire 2014 - 2015 PROF : KONAN HYACINTHE LICENCE 1

INTRODUCTION A L'ALGORITHMIQUE

CHAPITRE 5 :
INSTRUCTIONS REPETITIVES

1. Instruction TANT QUE

Syntaxe :
TANTQUE <condition> **FAIRE**
    < séquence d'instructions >
FINTANTQUE

Mécanisme

- la première fois : évaluation de <condition> Si la valeur de <condition> est vrai alors la séquence d'instructions est exécutée sinon (ie la valeur de <condition> est faux) on n'entre pas dans la structure et c'est l'instruction qui suit Fin tant que qui est exécutée

- les fois suivantes, on vient d'exécuter la séquence d'instructions. Il y a retour sur la condition pour la réévaluer. Si la valeur de <condition> est vrai alors la séquence d'instructions est exécutée sinon (ie la valeur de <condition> est faux) on n'entre pas dans la structure et c'est l'instruction qui suit **FINTANTQUE** qui est exécutée

Exemple : voir algorithme intérêts Chapître 1

Remarques :

- Il faut que la condition soit évaluable au moment où elle est évaluée
- La séquence peut ne jamais être exécutée si la condition est fausse
- Il faut que la condition soit modifiée par la séquence d'instructions, sinon soit on n'entre jamais, soit la boucle est éternelle

Exemple :

```
DEBUT
1       a ← 1
2       b ← a
3       TANTQUE a < 100 FAIRE
4               b ← b * 5
5               ECRIRE (b)
6       FINTANTQUE
FIN
```

	a	b	Ecran
1	1		
2		1	
3			
4		5	
5			5
3			
4		25	
5			25

Résumé
Instructions simples : agissent directement sur les variables.

Interactivité avec L'utilisateur	**Affectation**	Modifient certaines variables
	Ecriture	
	Lecture	

Instructions structurées : contrôlent l'exécution des instructions simples en fonction de l'état des variables à chaque instant du déroulement de l'algorithme.

SI <condition> **ALORS** <séquence_1> **SINON** <séquence_2> **FINSI**	Instruction conditionnelle
TANTQUE <condition> **FAIRE** <séquence> **FINTANQUE**	Instruction répétitives

D'autres instructions structurées seront présentées dans la suite.

2. Instruction Répéter

L'instruction **TANTQUE** était contrôlée par une expression booléenne d'entrée. On dispose d'une autre répétitive contrôlée par une expression booléenne de sortie

TANQUE somme < sommeFinale **FAIRE** < séquence > **FINTANQUE**	**REPETER** < séquence > **JUSQUA** somme >= sommeFinale

Syntaxe :

```
REPETER
        < séquence >
JUSQUA <expression booléenne>
```

Mécanisme

- éxécution de la séquence une première fois
- évaluation de <expression booléenne>. Si la valeur de <expression booléenne> est **VRAI** alors sortie et exécution de la suite de l'algorithme sinon (ie la valeur de <expression booléenne> est **FAUX**) retour pour exécuter à nouveau la séquence et tester l'expression booléenne.

exemple :

```
ALGORITHME Exemple
        /* Lexique */
        VARIABLE  nbCoups, d1, d2 : ENTIER  …
DEBUT
        …
        nbCoups ← 0
        REPETER
                LIRE (d1)
                LIRE (d2)
                nbCoups ← nbCoups + 1
        JUSQUA d1 + d2 = 12
        …
FIN
```

Remarques :

- la séquence doit modifier l'expression booléenne ou condition de sortie, sinon la boucle est éternelle
- la séquence est exécutée au moins une fois

3. Instruction Pour

Syntaxe :

```
 (1)
POUR <variable> DE <expr_début> A <expr_fin> FAIRE
      <séquence>
      (2)
FIN
```

Mécanisme

POUR i **DE** debut **A** fin **FAIRE** <séquence> **FINPOUR**	↔	i ← debut **TANTQUE** i <= fin **FAIRE** <séquence> i ← i + 1 **FINTANTQUE**

Au repère (1)

- la variable de boucle prend la valeur de <expr_début>
- l'expression booléenne <variable> ≤ <expr_fin> est évaluée si sa valeur est **VRAI** la séquence est exécutée sinon (ie si sa valeur est **FAUX**) exécution de l'instruction qui suit le **FINPOUR**

Au repère (2)

- la variable est incrémentée
- il y a retour pour évaluer l'expression booléenne <variable> ≤ <expr_fin> si sa valeur est **VRAI** la séquence est exécutée sinon (ie si sa valeur est **FAUX**) exécution de l'instruction qui suit le **FINPOUR**

Simulation d'un exemple

```
1     n ← 0
2     s ← 0
3     POUR i de 1 à 3 FAIRE
4           n ← n + 5
5           s ← s + n
6     FINPOUR
7     ECRIRE (s)
```

	n	n	i	Expr. bool.	Ecran
1	0				
2		0			
3			1	$1 \leq 3$: *vrai*	
4	5				
5		5			
6			2		
3				$2 \leq 3$: *vrai*	
4	10				
5		15			
6			3		
3				$3 \leq 3$: *vrai*	
4	15				
5		30			
6			4		
3				$4 \leq 3$: faux	
7					30

Remarques :

- La séquence ne doit modifier :
  - ni la variable de boucle
  - ni l'expression de fin
- la séquence peut ne jamais être exécutée si, avant d'aborder l'instruction pour, <expr_début> est déjà strictement supérieure à <expr_fin>
- Il existe une instruction pour par pas décroissant :

Simulation d'un exemple

```
1       n ← 0
2       s ← 0
3       POUR i de 1 à 3 FAIRE
4               n ← n + 5
5               s ← s + n
6       FINPOUR
7       ECRIRE(s)
```

	n	n	i	Expr. Bool.	Ecran
1	0				
2		0			
3			1	1 ≤ 3 : **VRAI**	
4	5				
5		5			
6			2		
3				2 ≤ 3 : **VRAI**	
4	10				
5		15			
6			3		
3				3 ≤ 3 : **VRAI**	
4	15				
5		30			
6			4		
3				4 ≤ 3 : **FAUX**	
7					30

Remarques :

- La séquence ne doit modifier :
  - ni la variable de boucle
  - ni l'expression de fin
- la séquence peut ne jamais être exécutée si, avant d'aborder l'instruction pour, <expr_début> est déjà strictement supérieure à <expr_fin>
- Il existe une instruction pour par pas décroissant :

POUR i DE debut A fin par PAS de −1 **FAIRE** <séquence> **FINPOUR**	↔	i ← debut **TANTQUE** i >= fin **FAIRE** <séquence> i ← i - 1 **FINTANTQUE**

4. Récapitulatif

- Instructions simples
  - affectation
  - lecture
  - écriture
- Instructions structurées
  - instructions de choix

    SI … ALORS … SINON … FINSI

    SELON … FINSELON

    SI … ALORS … SINONSI … ALORS … SINON … FINSI
  - instructions itératives

    TANTQUE - FINTANTQUE : contrôle à l'entrée

    REPETER - JUSQUA : contrôle à la sortie

    POUR - FINPOUR : contrôle sur le nombre de répétitions

Remarque 1 : Comment choisir la structure itérative adaptée ?

Nombre d'itérations connu avant l'exécution de la 1ère itération et non modifiable.	Nombre d'itérations non connu avant l'exécution de la 1ère itération.	
	0 itération possible	au moins 1 itération
POUR	**TANTQUE**	**REPETER**

Remarque 2 :
Voici 2 types d'algorithmes itératifs qui traitent des listes et qui se distinguent suivant que l'on doive traiter le dernier élément de la liste ou non. Les exemples choisis pour illustrer ces algorithmes abordent des circonstances très concrètes.

Exemple 1 :
A la caisse d'un supermarché, la caissière doit traiter tous les clients dans la file d'attente.

Exemple 2 :

A la caisse d'un supermarché, la caissière doit traiter (comptabiliser) tous les articles d'un client qui se trouvent sur le tapis roulant, sauf le dernier élément qui se trouve être la barre de séparation avec le client suivant. Pour peu que l'on dispose de fonctions permettant d'obtenir l'élément suivant et de tester si cet élément est le dernier de la liste, voici le principe de ces 2 algorithmes.

/* exemple 1 */ /* tous les éléments sont traités */ **REPETER** element = elementSuivant() traiter(element) **JUSQUA** dernier(element)	/* exemple 2 */ /* tous les éléments sont traités sauf le dernier */ element = elementSuivant() **TANTQUE** Non dernier(element) **FAIRE** traiter(element) element = elementSuivant() **FINTANQUE**

FIN DU CHAPITRE 5