Prof : **KONAN HYACINTHE**

Initiation à l'algorithmique
Chapitre 5 : Instruction repetitives
Licence 1 SRIT 2016-2017

Plan du Chapitre

Objectif du chapitre

A la fin de ce chapitre, vous devez être capable d'utiliser les instructions répétitives dans un algorithme.

1. Instruction TANTQUE … FAIRE … FINTANQUE

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 **FINTANTQUE** 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 (i.e. la valeur de **<condition>** est **FAUX**) on n'entre pas dans la structure et c'est l'instruction qui suit **FINTANTQUE** qui est exécutée.

Remarque :

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

Exemple

```
ALGORITHME SimulatioTanque
|       VARIABLE a, b : ENTIER
DEBUT
|1      a ← 1
|2      b ← a
|3      TANTQUE a < 100 FAIRE
|       |3.1    b ← b * 5
|       |3.2    ECRIRE (b)
|       FINTANTQUE
FIN
```

	a	b	Ecran
1	1		
2		1	
3			
3.1		5	
3.2			5
3			
3.1		25	
3.2			25
3	1	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 REPETER … JUSQUA ….

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 **< 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 **<expression booléenne>**.

Exemple :
Une expérience consiste à lancer N fois deux dés D1 et D2, et noter les résultats obtenus à chaque lancée. (exemple D1 = 3 et D2 = 5). Ecrire un algorithme nommé "JeuDeLance" permettant de lire les nombres affichés par D1 et D2 au fur et à mesure. On affichera le nombre N de lancée à l'issue duquel D1 + D2 = 12. Tradure l'algorithme en langage C.

ALGORITHME JeuDeLance
| **VARIABLE** nbLancee**,** d1**,** d2 : **ENTIER**
DEBUT
| nbLancee ← 0
| **REPETER**
| | **LIRE (**d1**)**
| | **LIRE (**d2**)**
| | nbLancee ← nbLancee + 1
| **JUSQUA** d1 + d2 = 12
| ECIRE("Le nombre de lancée est :", nbLancee)
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…DE…A…FAIRE …FINPOUR

Syntaxe :

(1)

POUR <variable> **DE** <expr_début> **A** <expr_fin> **FAIRE**

(2) <séquence>

FINPOUR

Mécanisme

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**

On a l'équivalence suivante :

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

Simulation d'un exemple

Simuler l'algorithme suivant puis le traduire en langage C.

ALGORITHME SimulatioPour
| **VARIABLE** n**, s** : **ENTIER**
DEBUT
|1 n ← 0
|2 s ← 0
|3 **POUR** i **DE** 1 **A** 3 **FAIRE**
| |3.1 n ← n + 5
| |3.2 s ← s + n
| **FINPOUR**
|4 **ECRIRE (**s**)**
FIN

	n	**s**	**i**	**Exp. bool**	**Ecran**
1	0				
2		0			
3			1	1 ≤ 3 : **VRAI**	
3.1	5				
3.2		5			
			2		
				2 ≤ 3 : **VRAI**	
	10				
		15			
			3		
				3 ≤ 3 : **VRAI**	
	15				
		30			
			4		
				4 ≤ 3 : **FAUX**	
					30

Remarque :

- 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