

Leçon 1 Structures de contrôle itératives complètes

Chapitre 4: Les structures de contrôle itératives

1-La structure Répéter ... Jusqu'à :

I-Définition itérative complète:

Un résultat a une définition itérative complète s'il est la répétition d'une suite d'instructions, un **nombre fini de fois connu à l'avance**.

→ **Parcours croissant:**

Vocabulaire et syntaxe:

Analyse & Algorithme	Pascal
R=[Init] Pour c de 1 à n faire Instruction 1 } Traitement Instruction 2 } à } Répéter Instruction p } FinPour ; {Init} FOR c:=1 TO n DO Begin Instruction_1; Instruction_2;; Instruction_p; End;

R : la répétition de p instructions (n fois), n nombre de répétition

Remarques:

- La partie Init contient les éventuelles initialisations des variables qui seront mis à jour au niveau de traitement répétitif.
- Le **compteur** doit être de type scalaire. (entier, caractère, booléen..)
- L'initialisation et l'avancement du compteur C est faite automatiquement. (incréméntation par défaut par un pas=1)
- Le traitement répétitif de la boucle POUR peut s'exécuter 0 ou n fois (n≥1)
- Lorsque le traitement répétitif est composé de plusieurs instructions, les expressions Begin et End sont nécessaires.

→ **Parcours décroissant:**

L'avancement du compteur se fait par un pas=-1

Analyse & Algorithme	Pascal
R=[inst1, inst2, ...instm] Pour i de n à 1 (pas=-1) faire Instruction 1 } Traitement Instruction 2 } à } Répéter Instruction p } FinPour ; {Init} FOR i:=n downTO 1 DO Begin Instruction_1; Instruction_2;; Instruction_p; End;

Décréméntation automatique du compteur (passage au prédécesseur de la valeur en cours).

Cas général:

Il y a des fois où le compteur entre dans le calcul fait par le module à répéter; en plus les opérations de calcul exigent des valeurs non entières et progressent avec un pas p non entier.

→ L'astuce consiste à chercher par division entière le nombre d'itération à accomplir et avec une expression généralement linéaire révenir au compteur dont on a besoin.

Cas général Analyse	Pascal
R=[iinit] Pour i de d à f (pas=p) faire Instruction 1 } Traitement Instruction 2 } à } Répéter Instruction m } FinPour ; {Init} n:=1+round((f-d)/p); FOR i:=1 TO n DO Begin c := i * p ; Instruction_1; Instruction_2;; Instruction_m; End;

Si p est positif, le parcours est ascendant et si p est négatif, le parcours est descendant.

Le nombre de répétition est $n=1+((f-d)/p)$ et dans ce cas le compteur effectif est $c=i*p$

Remarques: n est toujours positif, c'est le signe de p qui détermine le compteur c.

II-Les itérations complètes récurrentes:

Le résultat se forme au fur et à mesure et à une étape donnée, il dépend d'un certain nombre de résultats précédents. si relation lie deux éléments successifs (récurrence d'ordre 1) si elle lie trois éléments successifs (récurrence d'ordre 2) (voir exemple factoriel)

Leçon 2 Structures de contrôle itératives à conditions d'arrêt

I-Définition itérative à condition d'arrêt:

Un résultat a une définition itérative à condition d'arrêt s'il est la répétition d'une suite d'instruction et l'arrêt est géré par une condition.

Analyse	Pascal
..... [Init] Répéter Instruction 1 } Traitement Instruction 2 } à } Répéter Instruction N } Jusqu'à (condition d'arrêt) ; {Init} Repeat Instruction 1; Instruction 2; Instruction N; Until (condition d'arrêt);

Remarques:

- S'il y a un éventuel compteur, il faut l'initialiser avant la boucle; de même on doit assurer son avancement au sein de la boucle.
- Le traitement répétitif de la boucle répéter peut s'exécuter 1 ou n fois (n≥2).
- La condition à vérifier à chaque fois est considérée comme une **condition de sortie** car elle nous permet de **quitter la boucle**.
- Même si le traitement répétitif est composé de plusieurs instructions, on a jamais besoin des expressions Begin et End.
- La boucle répéter est utilisée entre autres dans le contrôle des données saisies.

*Les problèmes récurrents: voir exemple

2- La boucle Tant que:

Analyse	Pascal
..... [Init] Tant que (condition d'entrée) Faire Instruction 1 } Traitement Instruction 2 } à } Répéter Instruction N } Fin TantQue ; {Init} While (Condition) Do Begin Instruction 1; Instruction 2; Instruction N; End;

Remarques:

- Le traitement répétitif de la boucle Tant que peut s'exécuter 0 ou n fois. (0 fois dès le début si la condition n'est pas vérifiée).
- La condition à vérifier à chaque fois est considérée comme une **condition d'entrée** car elle nous permet **d'accéder au corps de la boucle**.
- Lorsque le traitement répétitif est composé de plusieurs instructions, les expressions Begin et End sont nécessaires.

Condition d'entrée = NON (Condition d'arrêt)