

## Les structures algorithmiques de contrôle

### A. Les structures de contrôle conditionnelles

#### 1. La structure conditionnelle simple réduite

##### 1. Définition

Une structure de contrôle conditionnelle a une forme simple réduite si on se restreint (on se limite) à l'exécution d'un traitement quand une condition est vraie.

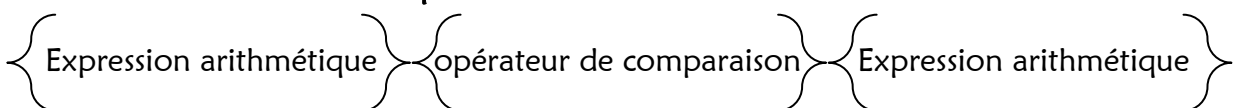
Elle a la forme générale suivante : **Si** condition **Alors** Traitement

Exemples : **Si**  $x \bmod 11 = 0$  **Alors** Ecrire (x, " est divisible par 11")

##### 2. Condition booléenne ?

Une condition booléenne est une proposition logique, c'est-à-dire une proposition dont on peut dire si elle est vraie ou fausse. Elle peut être **simple** ou **composée**.

Une condition booléenne **simple** est de la forme :



##### Remarques :

(1) Une expression arithmétique est une combinaison de constantes, de variables et des fonctions à l'aide d'opérateurs correspondant au type de données concernées.

(2) Les opérateurs de comparaison sont :  $<$ ,  $\leq$ ,  $>$ ,  $\geq$ ,  $=$ ,  $\neq$

Exemples : soient x et y deux variables de type réel :

$X = 3$  ;  $x \leq 0$  ;  $x < > 1$  ;  $2*x < y/2$  ;  $x \geq 2*y$  sont des conditions booléennes simples.

##### ⊖ Attention !

Il ne faut pas confondre  $x := 3$  et  $x = 3$

$X = 3$  est une condition, elle correspond à la question « x est-il égal à 3 ? »

$X := 3$  est une affectation, elle correspond à l'instruction « donner à x la valeur 3 »

Une condition booléenne **composée** est une conjonction de conditions booléennes simples via les connecteurs logiques NON, ET, OU.

##### Exemples :

(1) La condition booléenne  $(x > 0)$  ET  $(X \leq 1)$  signifie  $0 < x \leq 1$

(2) La condition  $x \in ]-\infty, 0[ \cup ]0, +\infty[$  s'écrit :

$(x < 0)$  OU  $(x > 0)$ , ou plus simplement  $x \neq 0$

(3) Enfin, pour traduire  $x \in ]-\infty, 0[ \cup [1, 2[$  on écrit :

$(x < 0)$  OU  $((x \geq 1)$  ET  $(x < 2))$



## 3. Syntaxe en algorithme et en pascal :

Syntaxe en algorithme	Syntaxe en pascal
<b>Si</b> Condition <b>Alors</b> Traitement <b>Fin Si</b>	<b>If</b> Condition <b>Then</b> Traitement ;

Remarques :

(1) Condition est une condition booléenne.

(2) Traitement : peut être une ou plusieurs instructions. (simples, composées, conditionnelles, répétitives).

(3) Si l'une des instructions du traitement est une structure conditionnelle alors on parle d'instructions conditionnelles **imbriquées**.

(4) Pour exécuter ces instructions, le compilateur teste la condition booléenne. Si celle-ci est vraie alors le traitement est exécuté sinon il ne se passe rien.

Exemple:

Program exemple :

Uses wincrt ;

Var x: integer;

Begin

Writeln ('entrer un entier:');

Readln (x) ;

If x mod 11 then

Writeln (x, ' est un divisible par 11');

End.

## II. La structure conditionnelle simple complète (alternative)

## 1. Définition

Une structure de contrôle conditionnelle a une forme simple complète si selon la valeur d'une condition, on exécute soit un traitement 1 ou un traitement 2.

Exemple :    **Si**  $n \bmod 2 = 0$  **Alors**  
                     Ecrire ("Nombre pair")  
                     **Sinon** Ecrire ("Nombre impair")  
                     **Fin Si**

## 2. Syntaxe en algorithme et en pascal :

Syntaxe en algorithme	Syntaxe en pascal
<b>Si</b> Condition <b>Alors</b> Traitement 1 <b>Sinon</b> Traitement 2 <b>Fin Si</b>	<b>If</b> Condition <b>Then</b> Traitement 1 <b>Else</b> Traitement 2 ;



Remarques :

- (1) Si Traitement 1 ou traitement 2 est constitué de plusieurs instructions, on doit les mettre entre **begin** et **end**.
- (2) Il n'y a jamais de point virgule à la fin de l'instruction qui précède **Else**.

Application :

On désire écrire un programme qui permet de transformer un caractère alphabétique en majuscule sans utiliser la fonction **majus**.

Questions :

- 1) Analyser le problème.
- 2) Donner un algorithme à ce problème.

**III. La structure conditionnelle généralisée****1. présentation**

Cette structure est utilisée lorsqu'on a plus qu'une condition. Sa syntaxe est :

Syntaxe en algorithmme	Syntaxe en pascal
<b>Si</b> Condition 1 <b>Alors</b> Traitement 1 <b>Sinon Si</b> Condition 2 <b>Alors</b> Traitement 2 <b>Sinon Si</b> Condition 3 <b>Alors</b> Traitement3 ... <b>Sinon Si</b> Condition N <b>Alors</b> Traitement N <b>Sinon</b> Traitement N+1 <b>Fin Si</b>	<b>If</b> Condition 1 <b>Then</b> Traitement 1 <b>Else if</b> Condition 2 <b>Then</b> Traitement 2 <b>Else if</b> Condition 3 <b>Then</b> Traitement 3 ... <b>Else if</b> Condition N <b>Then</b> Traitement N <b>Else</b> Traitement N+1; 

Exemple: Chercher la valeur absolue d'un entier X

```

Si X < 0 Alors
    Ecrire (" négatif")
    ABSx ← -X
Sinon Si X = 0 Alors
    Ecrire (" nul")
    ABSx ← 0
Sinon
    Ecrire (" positif")
    ABSx ← x
Fin Si
  
```

**Question :** Traduire cette structure en pascal



## IV. La structure conditionnelle à choix

### 1. Présentation

Cette structure conditionnelle est appelée aussi structure sélective ou à choix multiples. Car elle sélectionne entre plusieurs choix à la fois, et non entre deux choix alternatifs (le cas de la structure SI).

### 2. Syntaxe en algorithme et en pascal

Syntaxe en algorithme	Syntaxe en pascal
<b>Selon Sélecteur Faire</b> Valeur 1 : Traitement 1 Valeur 2 : Traitement 2 ... Valeur N : Traitement N <b>Sinon</b> Traitement M <b>Fin Selon</b>	<b>Case Sélecteur Of</b> Valeur 1 : Traitement 1 ; Valeur 2 : Traitement 2 ; ... Valeur N : Traitement N <b>Else</b> Traitement M ; <b>End ;</b>

#### Remarques :

- (1) Le sélecteur est une variable de type scalaire (entier, caractère, booléen).
- (2) La structure SELON évalue le "sélecteur", passe à comparer celui ci respectivement avec les valeurs dans la liste. En cas d'égalité avec une valeur, les actions correspondantes, qui sont devant cette valeur seront exécutées.
- (3) Valeur i ( $1 \leq i \leq N$ ) peut être formé d'une seule valeur, d'une liste de valeurs séparées par des virgules et/ou des intervalles de valeurs.

#### Application 1 :

Ecrire un programme qui permet de saisir l'extension d'un fichier sous forme de chaîne de caractères puis d'afficher, en fonction de cette extension (.pas, .c, .html, .doc, .xls), le type du fichier.

#### Questions :

- 1) Analyser le problème
- 2) Ecrire l'algorithme et le programme pascal correspondants



## B. Les structures de contrôle itératives

### I. La structure itérative complète

#### 1. Présentation

Cette structure exprime la répétition d'un traitement un nombre fini de fois connu à l'avance.

#### 2. Syntaxe en algorithme et en pascal

Syntaxe en algorithme	Syntaxe en pascal
<b>Pour</b> Compteur de Vi à Vf <b>Faire</b> Instruction 1 Instruction 2 ... Instruction n <b>Fin Pour</b>	<b>For</b> Compteur := Vi <b>to</b> Vf <b>do</b> <b>Begin</b> Instruction 1; Instruction 2; ... Instruction n; <b>End;</b>

#### Remarques :

(1) Compteur est une variable du type scalaire, qui compte le nombre de répétition du traitement.

(2) Vi : La valeur initiale du compteur

(3) Vf : La valeur finale du compteur

(4) La valeur du pas (l'avancement du compteur) est égale à 1 par défaut.

(5) En pascal, si  $Vi \leq Vf$ , **do** devient **downto**

(6) Comment ça marche ?

(a) Au début de la boucle on affecte à la variable Compteur la valeur de Vi

(b) Si Compteur  $\leq$  Vf alors

i. Exécution des instructions

ii. Incrémentation du compteur

iii. Retour à l'étape (b)

**Sinon** la boucle s'arrête et l'exécution se poursuit après **Fin Pour**

#### Application :

Ecrire un programme permettant de tester si un entier est premier ou non.

#### Questions :

1) Analyser le problème

2) Ecrire l'algorithme et le programme pascal correspondants.



## II. Les structures itératives à conditions d'arrêt

### 1. La structure Répéter...Jusqu'à

#### a) Présentation :

Cette structure permet de répéter un traitement une ou plusieurs fois et de s'arrêter sur une condition. En effet, lorsque la condition est vérifiée, la boucle s'arrête, si non elle ré exécute le traitement.

#### b) Syntaxe en algorithme et en pascal

Syntaxe en algorithme	Syntaxe en pascal
<pre>..... {initialisations} Répéter     Instruction 1     Instruction 2     ...     Instruction n Jusqu'à (condition d'arrêt)</pre>	<pre>..... {initialisations} Repeat     Instruction 1;     Instruction 2;     ...     Instruction n; Until (condition d'arrêt);</pre>

#### Remarques :

- (1) Cette structure est utilisée lorsque le nombre de répétitions à effectuer n'est pas connu à l'avance.
- (2) Dans cette boucle, le traitement (instruction1, instruction 2, ..., instruction n) est exécuté au moins une fois avant l'évaluation de la condition d'arrêt.
- (3) Il doit y avoir une action dans le traitement qui modifie la valeur de la condition d'arrêt.
- (4) Comment ça marche ?
  - (a) Exécution du traitement (instruction 1, instruction 2, ..., instruction n)
  - (b) Test de la valeur de la condition d'arrêt :
    - i. Si elle est vraie alors la boucle s'arrête et l'exécution se poursuit après la boucle
    - ii. Sinon retour à l'étape (a)

**Exemple :** Lecture d'un entier  $n > 0$

```
Répéter
    Ecrire ("entrer un entier : ")
    Lire (n)
Jusqu'à n > 0
```

#### Application:

Soit  $(U_n)_{n \in \mathbb{N}}$  la suite définie par son premier terme  $U_0 = a \in \mathbb{R}_+^*$ , et pour tout  $n$ , par la relation :  $U_{n+1} = (U_n)^2 - U_n + 1$

Ecrire un programme intitulé vitesse\_de\_convergence qui permet de saisir un réel  $a \in ]0,1[$  et d'afficher la première valeur de  $n$  pour laquelle  $|U_n - 1| < 10^{-6}$



Questions :

- 1) Analyser le problème
- 2) Ecrire l'algorithme et le programme pascal correspondants

**2. La structure Tant Que...Faire****a) Présentation**

Cette structure permet de répéter le traitement zéro ou plusieurs fois et de s'arrêter lorsque la condition d'exécution n'est plus vérifiée. En effet, lorsque la condition d'exécution est vérifiée, le traitement est exécuté, si non elle s'arrête.

**b) Syntaxe en algorithme et en pascal**

Syntaxe en algorithme	Syntaxe en pascal
<pre> ..... {initialisations} Tant Que (condition d'exécution) Faire     Instruction 1     Instruction 2     ...     Instruction n Fin Tant Que           </pre>	<pre> ..... {initialisations} While (condition d'exécution) Do Begin     Instruction 1;     Instruction 2;     ...     Instruction n; End;           </pre>

Remarques :

- (1) Cette structure est utilisée lorsque le nombre de répétitions à effectuer n'est pas connu à l'avance.
- (2) Dans cette boucle, le traitement peut ne pas être exécuté aucune fois, c'est lorsque la condition d'exécution est à faux dès le départ.
- (3) Les paramètres de la condition d'exécution doivent être initialisés par lecture ou par affectation avant la boucle.
- (4) Il doit y avoir une action dans le traitement qui modifie la valeur de la condition d'exécution.
- (5) Comment ça marche ?
  - (a) Tester la valeur de la condition d'exécution
  - (b) Si elle est vraie alors :
    - i. Exécution du traitement
    - ii. Retour à l'étape (a)
  - Sinon la boucle s'arrête et l'exécution se poursuit après Fin Tant Que
- (6) La condition d'arrêt et l'inverse de la condition d'exécution.



**Exemple** : Lecture d'un entier  $n > 0$

```

Ecrire ("entrer un entier : ")
Lire (n)
Tant Que  $n \leq 0$  Faire
    Ecrire ("entrer un entier : ")
    Lire (n)
Fin Tant Que
  
```

**Application** :

Soit la suite  $(U_n)_n \in \mathbb{N}$  définie par  $U_0 = a \in \mathbb{N}^*$  et pour tout entier naturel  $n$  par la relation :

$$U_{n+1} = \begin{cases} \frac{U_n}{2} & \text{Si } U_n \text{ est pair} \\ 3U_n + 1 & \text{Si } U_n \text{ est impair} \end{cases}$$

Quelque soit la valeur choisie pour  $U_0$  la suite converge vers le cycle **4 2 1** en un certain nombre fini d'étapes et le cycle **4 2 1** se répète indéfiniment.

On appelle :

- **Vol** : La liste des valeurs prises par la suite de  $U_0$  jusqu'au premier 1 (bornes comprises).
- **Longueur du vol** : Le nombre de termes de la liste formée par  $U_0, U_1, U_2, \dots$ , jusqu'à la rencontre du premier 1.

**Exemples** :

1) Pour  $n = 13$  et  $U_0 = 5$

Les 13 premiers termes de la suite  $U$  sont : **5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2**

Le vol de  $U$  est : **5, 16, 8, 4, 2, 1**

La longueur du vol de  $U$  est égale à 6

2) Pour  $n = 10$  et  $U_0 = 2$

Les 10 premiers termes de la suite  $U$  sont : **2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1**

Le vol de  $U$  est : **2, 1**

La longueur du vol de  $U$  est : 2

On veut écrire un programme qui permet de :

- Saisir deux entiers  $n > 0$  et  $U_0 > 0$ .
- Calculer et afficher les  $n$  premiers termes de la suite  $U$ .
- Afficher le **vol** de  $U$ .
- Afficher la **longueur du vol** de  $U$

**Questions** :

- 1) Analyser le problème.
- 2) En déduire l'algorithme correspondant.

