CHAPITRE 2

RECONNAITRE LES BASES

1-Traitement séquentiel (affectation, lecture et écriture)

2-Traitement alternatif(conditions)

3-Traitement itératif (boules)

Instruction d’affectation

L’affectation consiste à attribuer une valeur à une variable (c’est-à-dire remplir ou modifier le contenu d'une zone mémoire)

En pseudo-code, l'affectation est notée par le signe **=**

Var:= e : attribue la valeur de e à la variable Var

e peut être une valeur, une autre variable ou une expression

Var et e doivent être de même type ou de types compatibles

l’affectation ne modifie que ce qui est à gauche de la flèche

Instruction d’affectation

Exemples valides

i :=1 j ←i k := i+j x := 10.3 OK := FAUX ch1 := "SMI" ch2 := ch1 x := 4 x := j

(avec i, j, k : entier; x :réel; ok :booléen; ch1,ch2 :chaine de caractères)

Exemples non valides:

i := 10.3 OK := "SMI" j := x

Remarque

Les instructions d'un algorithme sont habituellement exécutées une à la suite de l'autre, en séquence (de haut en bas et de gauche à droite).

L'ordre est important

Instruction de lecture

Les instructions de lecture et d'écriture (Entrée/Sortie) permettent à la machine de communiquer avec l'utilisateur

La lecture permet d'entrer des donnés à partir du clavier.

En pseudo-code, on note :

lire (var)

La machine met la valeur entrée au clavier dans la zone mémoire nommée **var**

Le programme s'arrête lorsqu'il rencontre une instruction Lire et ne se poursuit qu'après la frappe d'une valeur au clavier et de la touche **Entrée**

Instruction d’écriture

L'écriture permet d'afficher des résultats à l'écran (ou de les écrire dans un fichier)

En pseudo-code, on note :

Ecrire (var)

La machine affiche le contenu de la zone mémoire **var**

Conseil

Avant de lire une variable, il est fortement conseillé d’écrire des messages à l’écran, afin de prévenir l’utilisateur de ce qu’il doit frapper.

Exemple : Ecrire(a, b+2, "Message")

CHAPITRE 2

RECONNAITRE LES BASES

1-Traitement séquentiel (affectation, lecture et écriture)

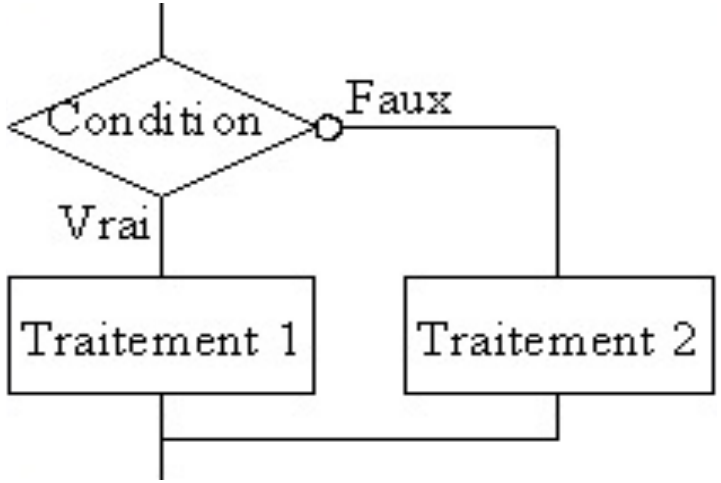
2-Traitement alternatif(conditions)

3-Traitement itératif (boules)

Traitement alternatif

Rappel :

Les instructions conditionnelles servent à n'exécuter une instruction ou une séquence d'instructions que si une condition est vérifiée.



Alternative Si-Alors-Sinon

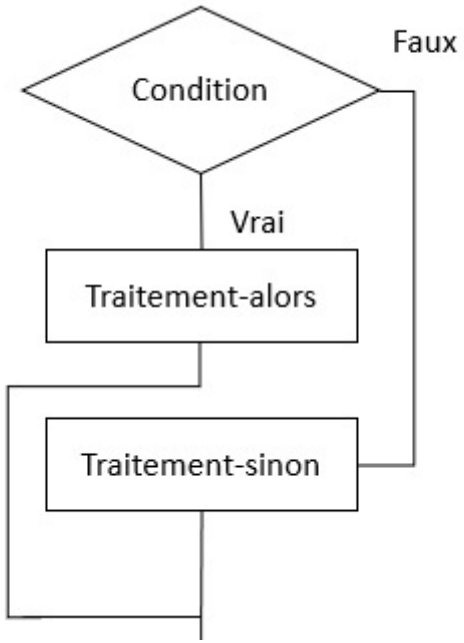
Schéma conditionnel généralisé ou multiple

Traitement alternatif

Alternative Si-Alors-Sinon

Elle permet d'effectuer tel ou tel traitement en fonction de la valeur d'une condition

Organigramme



En pseudo-code

```
Si condition  Alors
    <Actions_alors>
Sinon
   <Actions_sinon>
Fsi
```

Traitement alternatif

Exemple 1

```
si ( a≠0 )
alors
        a := 0
sinon
        a := b
        c := d
 fsi
```

Si la condition est vraie c'est à dire la variable **a** est différente de **0** alors on lui affecte la valeur **0**, sinon on exécute le bloc **sinon**.

Exemple 2

```
si ( a – b ≠ c )
alors
        a := c
sinon
        a := d
fsi
```

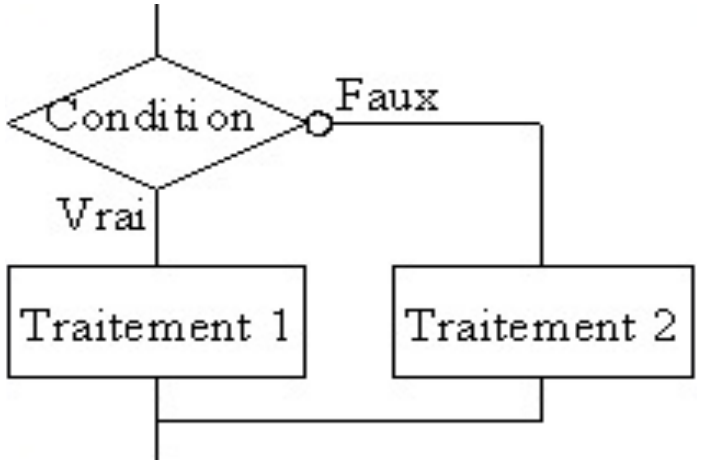
Si la condition est vraie, la seule instruction qui sera exécutée est l'instruction d'affectation **a : = c.**

Sinon la seule instruction qui sera exécutée est l'instruction d'affectation **a : = d.**

Traitement alternatif

Rappel

Les instructions conditionnelles servent à n'exécuter une instruction ou une séquence d'instructions que si une condition est vérifiée.



Alternative Si-Alors-Sinon

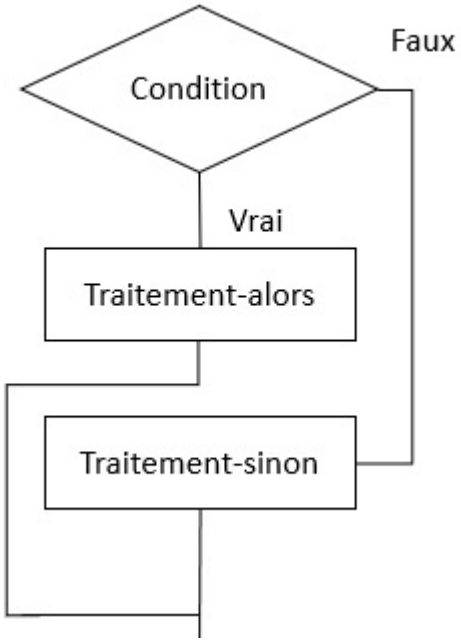
Schéma conditionnel généralisé ou multiple

Traitement alternatif

Alternative Si-Alors-Sinon

Elle permet d'effectuer tel ou tel traitement en fonction de la valeur d'une condition

Organigramme



En pseudo-code

```
Si condition  Alors
    <Actions_alors>
Sinon
   <Actions_sinon>
Fsi
```

Traitement alternatif

Exemple 1

```
si ( a≠0 )
alors
        a := 0
sinon
        a := b
        c := d
 fsi
```

Si la condition est vraie c'est à dire la variable a est différente de **0** alors on lui affecte la valeur **0**, sinon on exécute le bloc sinon.

Exemple 2

```
si ( a – b ≠ c )
alors
        a := c
sinon
        a := d
fsi
```

Si la condition est vraie, la seule instruction qui sera exécutée est l'instruction d'affectation a := c. Sinon la seule instruction qui sera exécutée est l'instruction d'affectation a := d.

Traitement alternatif

Schéma conditionnel généralisé ou multiple

La structure cas permet d'effectuer tel ou tel traitement en fonction de la valeur des conditions 1ou 2 ou ..n

En pseudo-code

```
Cas <var> de:
        <valeur 1> : <action 1>
        < valeur 2> : <action 2>
        ...
        < valeur n> : <action n>
        Sinon : <action_sinon>
FinCas
```

la partie action-sinon est facultative

Traitement alternatif

Exemple 1

On dispose d'un ensemble de tâches que l'on souhaite exécuter en fonction de la valeur d'une variable choix de type entier, conformément au tableau suivant :

Valeur de choix	Tâche à exécuter
1	Commande
2	Livraison
3	Facturation
4	Règlement
5	Stock
Autre valeur	ERREUR

Traitement alternatif

Alternative Si-Alors-Sinon

```
si          choix = 1
alors       Commande
sinon       si    choix = 2
            alors       Livraison
            sinon       si          choix = 3
                        alors       Facturation
                        sinon       si
            choix = 4
                                    alors
            Règlement
                                    sinon
            si          choix = 5

            alors       Stock

            sinon        ecrire (''Erreur'')

            fsi
                                    fsi
                        fsi
            fsi
fsi
```

Structure à choix multiples

```
Cas choix de :
1:          Commande
2:          Livraison
3:          Facturation
4:      Règlement
sinon  ecrire (''Erreur'')

finCas
```

L'emboîtement de si en cascade est fastidieux à écrire

CHAPITRE 2

RECONNAITRE LES BASES

1-Traitement séquentiel (affectation, lecture et écriture)

2-Traitement alternatif(conditions)

3-Traitement itératif (boules)

Traitement itératif

Instruction itérative « pour »

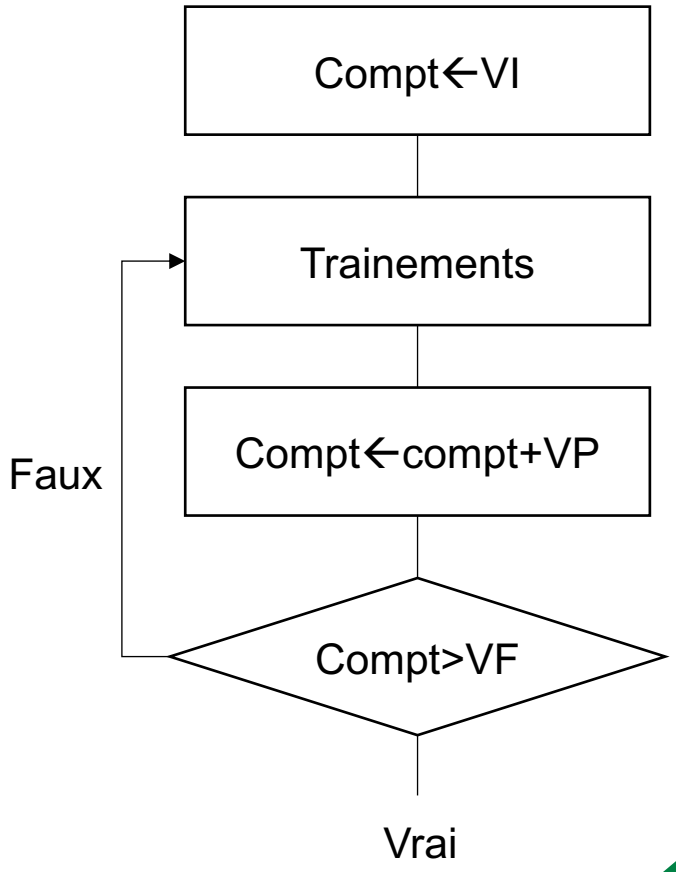
Répéter N fois une suite d'instructions à l'aide d'une variable entière servant de compteur

Valeur initiale

Valeur finale

```
Pour <compt> de <VI> à <VF> [pas <VP>] faire
   instructions
Fpour
```

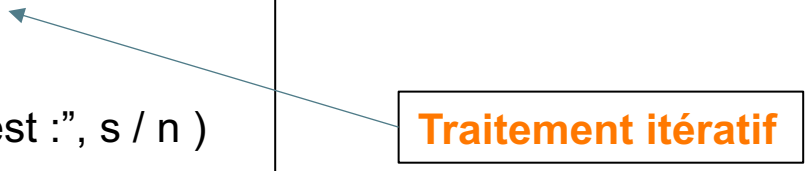
Valeur à ajouter à compt à chaque passage dans la boucle



Le traitement itératif

Exemple : un algorithme permettant de lire N entiers, de calculer et d'afficher leur moyenne

```
moyenne
var   n, i, x, s : entier
Début
        lire( n )
        s := 0
        pour  i  de  1  à  n  faire
                lire( x )
                s := s + x
        fpour
        ecrire( "la moyenne est :", s / n )
Fin
```

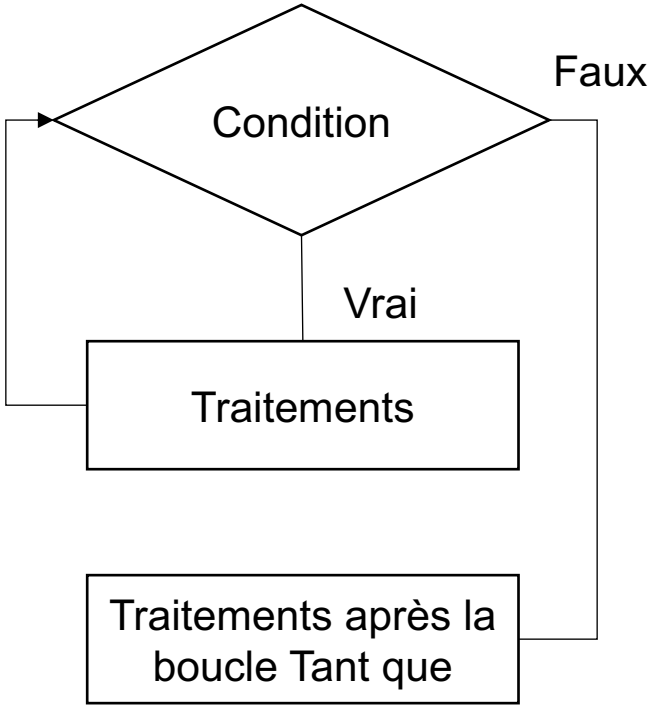


Le traitement itératif

Instruction itérative « Tant que »

Répéter une suite d'instructions tant que la condition est vraie

```
Tant que <condition> faire
    Instructions
 Fait
```



Le traitement itératif

Exemple : un algorithme permettant de lire une suite d'entiers positifs, de calculer et d'afficher leur moyenne.

```
moyenne
var    i, x, s : entier

Début

        lire( x )
        s := 0
        i := 0
        tant que  x > 0  faire
                i := i + 1
                s := s + x
                lire( x )
        fait
        si   i ≠ 0
        alors  écrire( "la moyenne est :", s /
        i )
        fsi
Fin
```

Condition obligatoire pour éviter de diviser par 0 si le premier entier lu est 0

Le traitement itératif

Exemple : un algorithme permettant de lire une suite d'entiers positifs, de calculer et d'afficher leur moyenne.

```
moyenne
var    i, x, s : entier

Début

        lire( x )
        s := 0
        i := 0
        tant que  x > 0  faire
                i := i + 1
                s := s + x
                lire( x )
        fait
        si   i ≠ 0
        alors  écrire( "la moyenne est :", s / i )
        fsi
Fin
```

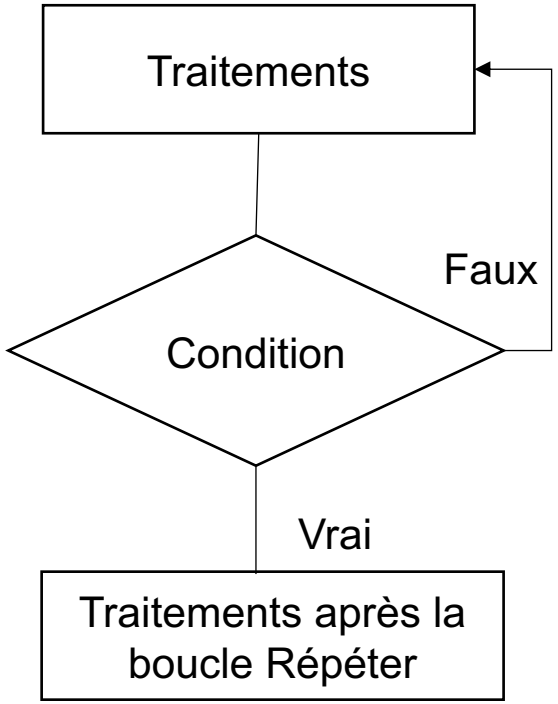
Condition obligatoire pour éviter de diviser par 0 si le premier entier lu est ≤ 0

Le traitement itératif

Instruction itérative « Répéter »

Répéter une suite d'instructions jusqu'à que la condition soit évaluée à Faux

```
Répéter
    instructions
Jusqu'à <condition>
```



Le traitement itératif

Exemple : un algorithme permettant de lire deux entiers, de calculer et d'afficher le résultat de la division du premier par le second (quotient)

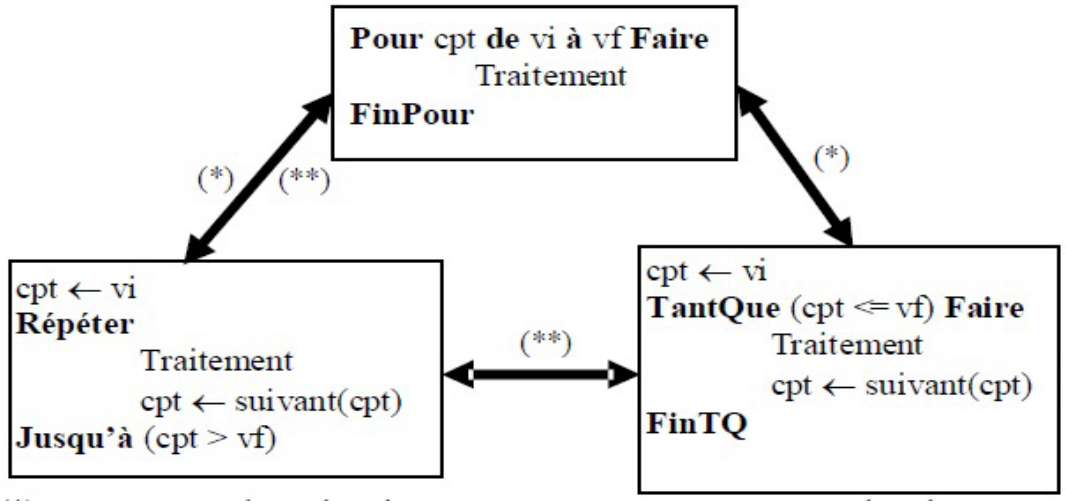
```
quotient
var   x, y : entier

Début

      lire( x )
      répéter
            lire( y )
      jusqu'à  y > 0
      écrire( x / y )
Fin
```

un contrôle obligatoire doit être effectué lors de la lecture de la deuxième valeur

Passage d'une structure itérative à une autre

(*) : Le passage d'une boucle « répéter » ou « tantque » à une boucle « pour » n'est possible que si le nombre de parcours est connu à l'avance

(**) : Lors du passage d'une boucle « pour » ou « tantque » à une boucle « répéter », faire attention aux cas particuliers (le traitement sera toujours exécuté au moins une fois)



Choix de la structure itérative