# INFORMATIQUE DE BASE

1er année ENCG

Prof. Mohamed BOUDCHICHE
Email : m.boudchiche@encgcasa.ma
Web : http://mohavic.com

Année universitaire 2023-2024

1

01 ALGORITHMIQUE
*Notions et instructions de base*

Prof. Mohamed BOUDCHICHE

2

## Pourquoi apprendre l'algorithmique pour apprendre à programmer ?

Un algorithme est une description complète et détaillée des actions à effectuer et de leur séquencement pour arriver à un résultat donné

- **Intérêt :** séparation analyse/codage (pas de préoccupation de syntaxe) l'algorithmique exprime les instructions résolvant un problème donné indépendamment des particularités de tel ou tel langage.
- **Qualités :** exact (fournit le résultat souhaité), efficace (temps d'exécution, mémoire occupée), clair (compréhensible), général (traite le plus grand nombre de cas possibles), …



3

## Représentation d'un algorithme

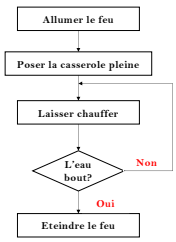
Historiquement, deux façons pour représenter un algorithme:

- **L'Organigramme :** représentation graphique avec des symboles (carrés, losanges, etc.)
  - offre une vue d'ensemble de l'algorithme
  - représentation quasiment abandonnée aujourd'hui
- **Le pseudo-code :** représentation textuelle avec une série de conventions ressemblant à un langage de programmation (sans les problèmes de syntaxe)
  - plus pratique pour écrire un algorithme
  - représentation largement utilisée



4

## Exemple d'organigramme






5

## Exemple de Pseudo code

Problème: addition
Entrée: deux chiffres A et B
Sortie: un chiffre C=A+B
Variables A, B, C : réel
Début
C ← A+B
Fin



6

## Les catégories d'ordres

Quatre familles d'instructions sont :

- Les variables et leurs **AFFECTATION**
- La **LECTURE / ÉCRITURE**
- Les **TESTS**
- Les **BOUCLES**



7

## Notions fondamentales

Karim possède 3 seaux : un seau en plastique d'une contenance de 10 litres, un seau en bois d'une contenance de 7 litres et un seau en fer d'une contenance de 9 litres.

- 10h00 : karim vide ses 3 seaux
- 10h05 : karim va rendre visite a Nabil, celui-ci met 6 litres dans le seau en bois de Karim
- 10h10 : karim transverse le contenu de son seau en bois dans le seau en fer
- 10h15 : karim revient vers nabil remplir à ras bord son seau en plastique
- 10h20 : karim déverse la moitié de son seau en plastique à l'égout
- 10h25 : karim transvase le contenu de son seau en plastique dans celui en bois
- 10h30 : karim transverse 2 litres de son seau en bois dans celui en fer
- 10h35 : karim informe Asmae du nombre de litres contenu dans ses seaux en plastique, en bois, en fer.

Quelles sont les quantités des trois seaux que Asmae a reçues?



8

## Notions fondamentales

- **Notion d'algorithme :** si les huis phrases sont bien exécutée par Karim, alors l'histoire est un algorithme
- **Notion d'instruction :** chacune de huis phrases est une instruction (un ordre)
- **Notion de valeur :** { 0, 3, 5, 6, 8, 10 }
- **Notion de mémoire :** elle est matérialisée par les seaux qui « mémorisent » les quantités de liquide



9

## Notions fondamentales

- **Notion de variable :** une variable est un emplacement mémoire, ici on a trois variables (le seau en plastique, le seau en bois et le seau en fer)
- **Notion d'environnement :** c'est l'ensemble des objet, informations, personnes qui on une existence hors de l'histoire mais qui interviennent dans son déroulement.
- **Notion des valeurs d'entrée et de sortie :** c'est les valeurs que le processeur reçoit de l'environnement et celles qu'il donne à l'environnement durant l'exécution. Valeurs en entrée :{6, 10} Valeurs en sortie = {0, 3, 8}



10

## Notion de variable

- Dans les langages de programmation une **variable** sert à stocker la valeur d'une donnée
- Une variable désigne en fait un emplacement mémoire dont le contenu peut changer au cours d'un programme (d'où le nom d'un d où variable)
- Règle: Les variables doivent être **déclarées** avant d'être utilisées, elle doivent être caractérisées par :
  - un nom (**Identificateur**)
  - un **type** (entier, réel, caractère, chaîne de caractères, …)



11

## Choix des identificateurs

Le choix des noms de variables est soumis à quelques règles qui varient selon le langage, mais en général:

- Un nom doit commencer par une lettre alphabétique
  exemple valide : A1 exemple invalide : 1A
- doit être constitué uniquement de lettres, de chiffres et du soulignement _ (Eviter les caractères de ponctuation et les espaces)
  valides: ENCG2012, ENCG_2012
  invalides: ENCG 2012, ENCG-2012, ENCG;2012
- doit être différent des mots réservés du langage (par exemple en Java: int, float, else, switch, case, default, for, main, return, …)
- La longueur du nom doit être inférieure à la taille maximale spécifiée par le langage utilisé



12

## Choix des identificateurs

Conseil: pour la lisibilité du code choisir des noms significatifs qui décrivent les données manipulées

exemples: TotalVentes2006, Prix_TTC, Prix_HT

Remarque: en pseudo-code algorithmique, on va respecter les règles citées.



13

## Types des variables

Le type d'une variable détermine l'ensemble des valeurs qu'elle peut prendre, les types offerts par la plus part des langages sont:

- Type numérique (entier ou réel)
  **Byte** (codé sur 1octet), **Entier court** , **Entier long**, **Réel simple précision**, **Réel double précision**
- Type logique ou booléen: deux valeurs **VRAI** ou **FAUX**
  **Si** bouton enfoncé
  **alors** lumière allumée = vrai
  **sinon** lumière allumée = faux
- Type caractère: lettres majuscules, minuscules, symboles, …
  exemples: 'A', 'a', '?', …

Type chaîne de caractère: toute suite de caractères,
exemples: " Nom, Prénom", "code postale: 1000", …



14

## Déclaration des variables

- Rappel: toute variable utilisée dans un programme doit avoir fait l'objet d'une déclaration préalable
- En pseudo-code, on va adopter la forme suivante pour la déclaration de variables

**Variables** liste d'identificateurs **: type d'identificateurs**

- Exemple:

```
Variables i, j, k : entier
          x, y : réel
          OK: booléen
          ch1, ch2 : chaîne de caractères
```

- Remarque: pour le type numérique on va se limiter aux entiers et réels sans considérer les sous types



15

## L'instruction d'affectation

**L'affectation**

- consiste à attribuer une valeur à une variable
- ça consiste à remplir où à modifier le contenu d'une zone mémoire

En pseudo-code, l'affectation se note avec le signe ←

Var ← e : attribue la valeur de **e** 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

- Ex valides: i ← 1 j ← i k ← i+j
  x ← 10.3 OK ← FAUX ch1 ← 'ENCG'
  ch2 ← ch1 x ← 4 x ← j
  (voir la déclaration des variables dans le transparent précédent)
- non valides: i ← 10.3 OK ← 'ENCG' j ← x



16

## L'instruction d'affectation

**L'affectation**

- consiste à attribuer une valeur à une variable
- ça consiste à remplir où à modifier le contenu d'une zone mémoire

En pseudo-code, l'affectation se note avec le signe ←

Var ← e : attribue la valeur de **e** 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

- Ex valides: i ← 1 j ← i k ← i+j
  x ← 10.3 OK ← FAUX ch1 ← 'ENCG'
  ch2 ← ch1 x ← 4 x ← j
  (voir la déclaration des variables dans le transparent précédent)
- non valides: i ← 10.3 OK ← 'ENCG' j ← x



17

## Quelques remarques

- L'affectation n'est pas commutative : A ← B est différente de B ← A
- L'affectation est différente d'une équation mathématique :
  - A=A+1 a un sens en langages de programmation
  - A+1=2 n'est pas possible en langages de programmation et n'est pas équivalente à A=1
- Certains langages donnent des valeurs par défaut aux variables déclarées. Pour éviter tout problème il est préférable d'initialiser les variables déclarées



18

## Exercices simples sur l'affectation (1)

Donnez les valeurs des variables A, B et C après exécution des instructions suivantes ?

**Variables** A, B, C: Entier
**Début**
A ← 3
B ← 7
A ← B
B ← A+5
C ← A + B
C ← B – A
**Fin**



19

## Exercices simples sur l'affectation (2)

Donnez les valeurs des variables A et B après exécution des instructions suivantes ?

**Variables** A, B : Entier
**Début**
A ← 1
B ← 2
A ← B
B ← A
**Fin**



20

## Exercices simples

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

**Variables** A, B : Entier
**Début**

| | | |
|---|---|---|
| A ← 1 | A = 1 | B = ? |
| B ← A + 3 | A = 1 | B = 4 |
| A ← 3 | A = 3 | B = 4 |

**Fin**



21

## Exercices simples

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

**Variables** A, B , C : Entier
**Début**

| | | | |
|---|---|---|---|
| A ← 5 | A = 5 | B = ? | C = ? |
| B ← 3 | A = 5 | B = 3 | C = ? |
| C ← A + B | A = 5 | B = 3 | C = 8 |
| A ← 2 | A = 2 | B = 3 | C = 8 |
| C ← B – A | A = 2 | B = 3 | C = 1 |

**Fin**



22

## Exercices simples

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

**Variables** A, B , C : Entier
**Début**

| | | | |
|---|---|---|---|
| A ← 8 | A = 8 | B = ? | C = ? |
| B ← 11 | A = 8 | B = 11 | C = ? |
| C ← A + B | A = 8 | B = 11 | C = 19 |
| B ← A + B | A = 8 | B = 19 | C = 19 |
| A ← B | A = 19 | B = 19 | C = 19 |

**Fin**



23

## Exercices simples

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

**Variables** A, B : Entier
**Début**

| | | |
|---|---|---|
| A ← 5 | A = 5 | B = ? |
| B ← 2 | A = 5 | B = 2 |
| A ← B | A = 2 | B = 2 |
| B ← A | A = 2 | B = 2 |

**Fin**



24

## Exercices simples

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

**Variables** A, B, **C** : Entier

**Début**

A ← 5 — A = 5 B = ?
B ← 2 — A = 5 B = 2
C ← A — A = 5 B = 2 C = 5
A ← B — A = 2 B = 2
B ← C — A = 2 B = 5

**Fin** — A = 5 B = 2 / A = 2 B = 5



25

## Exercices simples (concatenation)

Que produit l'algorithme suivant ?

**Variables** A, B, **C** : Chaine de caractere

**Début**

A ← "423"
B ← "12"
C ← A & B — "42312".

**Fin**



26

## Expressions et opérateurs

Une expression peut être une valeur, une variable ou une opération constituée de variables reliées par des opérateurs

exemples: 1, b, a*2, a+ 3*b-c, …

L'évaluation de l'expression fournit une valeur unique qui est le résultat de l'opération

Les opérateurs dépendent du type de l'opération, ils peuvent être :

- des opérateurs arithmétiques : +, -, *, /, **%** (modulo), ^ (puissance)
- des opérateurs logiques : **NON**, **OU**, **ET**
- des opérateurs relationnels : =, ≠,<, >, <=, >=
- des opérateurs sur les chaînes : **&** (concaténation)

Une expression est évaluée de gauche à droite mais en tenant compte de priorités



27

## Priorité des opérateurs

Pour les opérateurs arithmétiques donnés ci-dessus, l'ordre de priorité est le suivant (du plus prioritaire au moins prioritaire) :

- ^ : (élévation à la puissance)
- *, / : (multiplication, division)
- % (modulo)
- + , - : (addition, soustraction)

exemple: 2 + 3 * 7 vaut 23

En cas de besoin (ou de doute), on utilise les parenthèses pour indiquer les opérations à effectuer en priorité

exemple: (2 + 3) * 7 vaut 35



28

## Les instructions d'entrées-sorties: lecture et écriture (1)

Les instructions de lecture et d'écriture permettent à la machine de communiquer avec l'utilisateur

**La lecture** permet d'entrer des données 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**

Remarque : 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



29

## Les instructions d'entrées-sorties: lecture et écriture (2)

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

En pseudo-code, on note: **écrire (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



30

## Exemple (lecture et écriture)

Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le double de ce nombre

```
Algorithme Calcul_double
    variables A, B : entier
    Début
        écrire("entrer la valeur de A ")
        lire(A)
        B ← 2*A
        écrire("le double de ", A, "est :", B)
    Fin
```



31

## Exemple (lecture et écriture)

Ecrire un algorithme qui vous demande de saisir votre nom puis votre prénom et qui affiche ensuite votre nom complet

```
Algorithme AffichageNomComplet
    variables Nom, Prenom, Nom_Complet : chaîne de caractères
    Début
        écrire("entrez votre nom")
        lire(Nom)
        écrire("entrez votre prénom")
        lire(Prenom)
        Nom_Complet ← Nom & Prenom
        écrire("Votre nom complet est : ", Nom_Complet)
    Fin
```



32

## Exemple (lecture et écriture)

Ecrire un algorithme en pseudo-code qui traduit les phrases suivantes:

1. Demander à l'utilisateur un nombre.
2. Lui ajouter 1.
3. Multiplier le résultat par 2.
4. Soustraire 3 au résultat.
5. Afficher le résultat.

```
Algorithme calcul_mathématique
    variables x : reel
    Début
        écrire(" introduire un nombre")
        lire(x)
        x ← x+1
        x ← x * 2
        x ← x -3
        écrire(x)
    Fin
```



33

## Exemple (lecture et écriture)

Ecrire un algorithme qui demande à l'utilisateur d'entrer la largeur et la longueur et afficher la surface d'un rectangle :.

```
Algorithme surface_rectangle
    Variables largeur, longueur, surface : réel
    Début
        Ecrire(" entrer la largeur : " )
        Lire(largeur)
        Ecrire(" entrer la longueur : " )
        Lire(longueur)
        Surface ← largeur*longueur
        Ecrire (" la surface d'un rectangle est : ",
surface)
    Fin
```



34

## Méthode de construction d'un algorithme simple (1/3)

**Exemple :**

Écrire un algorithme qui consiste calculer l'air d'un cercle selon la formule $S = Pi * R^2$.

Rappel : Pi = 3.14159 et R le rayon du cercle



35

## Méthode de construction d'un algorithme simple (2/3)

**Méthodologie a suivre :**

- constantes : Pi = 3.14159
- Variables : Rayon, Surface
- Types : Rayon, Surface : réel
- Expressions et affectation : Surface := Pi * (Rayon)²
- Opérations d'entrée-sortie : Lire (Rayon), Écrire (Surface)



36

## Méthode de construction d'un algorithme simple (3/3)

```
Algorithme Calcul_Aire
  Constantes Pi = 3,14159
  Variables Rayon, Surface : réels
    Début
      écrire (" Entrez le rayon : ")
      lire (Rayon)
      Surface ← Pi * (Rayon^2)
      écrire (Surface)
    Fin
```



37

## Exercice

Ecrire un programme qui lit le prix HT d'un article, la quantité d'articles et le taux de TVA, et qui fournit le prix total TTC correspondant.

```
Variables Qte, Prix_HT, TVA, Prix_TTC : reels
  Début
    Ecrire ("Entrez le prix hors taxes :")
    Lire (Prix_HT)
    Ecrire ("Entrez le nombre d'articles :")
    Lire (Qte)
    Ecrire ("Entrez le taux de TVA :" )
    Lire (TVA)
    Prix_TTC ← Qte * Prix_HT * (1 + TVA)
    Ecrire ("Le prix toutes taxes est : ", Prix_TTC)
  Fin
```



38

02

# ALGORITHMIQUE

*Les structures conditionnelles*



39

## Les structures conditionnelles

Les tests simples :

**Si** condition **Alors** instructions …

Les instructions conditionnelles :

**Si** condition **Alors** … **Sinon** …



40

## Tests: instructions conditionnelles (1)

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

On utilisera la forme suivante :

```
Si condition alors
        instruction ou suite d'instructions1
Sinon
        instruction ou suite d'instructions2
Finsi
```

- la condition ne peut être que **vraie** ou **fausse**
- si la condition est **vraie**, se sont les instructions**1** qui seront exécutées
- si la condition est **fausse**, se sont les instructions**2** qui seront exécutées
- la condition peut être une condition simple ou une condition composée de plusieurs conditions



41

## Tests: instructions conditionnelles (2)

La partie **Sinon** n'est pas obligatoire, quand elle n'existe pas et que la condition est fausse, aucun traitement n'est réalisé

On utilisera dans ce cas la forme simplifiée suivante :

```
Si condition alors
        instruction ou suite d'instructions1
Finsi
```



42

## Exemple : (si … alors … sinon …)

```
Algorithme AffichageValeurAbsolue (version1)
    Variable x : réel
        Début
            Ecrire("Entrez un réel : ")
            Lire(x)
            Si x < 0 Alors
                Ecrire("la valeur absolue de ", x, "est:", -x)
            Sinon
                Ecrire("la valeur absolue de ", x, "est:", x)
            Finsi
        Fin
```



43

## Exemple : (si … alors…)

```
Algorithme AffichageValeurAbsolue (version2)
    Variable x, y : réel
        Début
            Ecrire("Entrez un réel : ")
            Lire(x)
            y ← x
            Si x < 0 Alors
                y ← -x
            Finsi
            Ecrire("la valeur absolue de ", x, "est:", y)
        Fin
```



44

## Conditions composées

Une condition composée est une condition formée de plusieurs conditions simples reliées par des opérateurs logiques :

ET, OU, OU exclusif (XOR) et NON

**Exemples :**

- x compris entre 2 et 6 : (x > 2) **ET** (x < 6)
- n divisible par 3 ou par 2 : (n%3=0) **OU** (n%2=0)
- deux valeurs et deux seulement sont identiques parmi a, b et c : (a=b) **XOR** (a=c) **XOR** (b=c)

L'évaluation d'une condition composée se fait selon des règles présentées généralement dans ce qu'on appelle tables de vérité



45

## Tables de vérité

| A | B | A **ET** B |
|---|---|---|
| VRAI | VRAI | **VRAI** |
| VRAI | FAUX | **FAUX** |
| FAUX | VRAI | **FAUX** |
| FAUX | FAUX | **FAUX** |

| A | B | A **OU** B |
|---|---|---|
| VRAI | VRAI | **VRAI** |
| VRAI | FAUX | **VRAI** |
| FAUX | VRAI | **VRAI** |
| FAUX | FAUX | **FAUX** |

| A | B | A **XOR** B |
|---|---|---|
| VRAI | VRAI | **FAUX** |
| VRAI | FAUX | **VRAI** |
| FAUX | VRAI | **VRAI** |
| FAUX | FAUX | **FAUX** |

| A | **NON** A |
|---|---|
| VRAI | **FAUX** |
| FAUX | **VRAI** |



46

## Tests imbriqués

Les tests peuvent avoir un degré quelconque d'imbrications

```
Si condition1 Alors
    Si condition2 Alors
        instructionsA
    Sinon
        instructionsB
    Finsi
Sinon
    Si condition3 Alors
        instructionsC
    Finsi
Finsi
```



47

## Tests imbriqués: exemple (version 1)

Ecrire un algorithme pour détecter si un entier est négatif, nul ou positif

```
Variable n : entier
    Début
        Ecrire("entrez un nombre : ")
        Lire(n)
        Si n < 0 Alors
            Ecrire("Ce nombre est négatif")
        Sinon
            Si n = 0 Alors
                Ecrire("Ce nombre est nul")
            Sinon
                Ecrire("Ce nombre est positif")
            Finsi
        Finsi
    Fin
```



48

## Tests imbriqués: exemple (version 2)

```
Variable n : entier
Début
    Ecrire("entrez un nombre : ")
    Lire (n)
    Si n < 0 Alors Ecrire("Ce nombre est négatif")
    Finsi
    Si n = 0 Alors Ecrire("Ce nombre est nul")
    Finsi
    Si n > 0 Alors Ecrire("Ce nombre est positif")
    Finsi
Fin
```

- **Remarque :** dans la version 2 on fait trois tests systématiquement alors que dans la version 1, si le nombre est négatif on ne fait qu'un seul test
- **Conseil :** utiliser les tests imbriqués pour limiter le nombre de tests et placer d'abord les conditions les plus probables (minimiser la complexité)



49

## Exercices

**Exercice 1 :**

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul).

**Exercice 2 :**

Le prix de photocopies dans une reprographie varie selon le nombre demandé:
0,5DH la copie pour un nombre de copies inférieur à 10,
0,4DH pour un nombre compris entre 10 et 20,
0,3DH au-delà.

Ecrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées, qui calcule et affiche le prix à payer



50

## Exercices

Une entreprise de Textile a recruté un chef de projet. Le directeur des ressources humaines souhaite concevoir un bulletin de paie qui prenne en compte la réalisation des objectifs (quantitatifs et qualitatifs) qui sont attendus du chef de projet.

Le salaire mensuel de base du chef de projet est de 10000DH, l'objectif quantitatif porte sur le chiffre d'affaires mensuel réalisé, si ce chiffre d'affaires dépasse 1000000DH, le chef de projet perçoit une commission de 1% du chiffre d'affaires.

L'objectif qualitatif porte sur le nombre de clients non satisfaits (0 retours …): le chef perçoit une prime selon les modalités suivantes :

| Défauts, retours | Prime |
|---|---|
| 0 à 5 clients non satisfaits | 3000 DH |
| 6 à 10 clients non satisfaits | 2000 DH |
| Plus de 10 clients non satisfaits | 0 DH |

Le salaire brute totale = salaire de base + commission + prime

Ecrire un algorithme qui permet de saisir le salaire de base (salB), le chiffre d'affaires mensuel (CAM) réalisé et le nombre de clients non satisfaits (nbClient) et calculer et afficher la commission (com), la prime (prime) et le salaire brute total du chef de projet (salBTotal)



51

## 02 ALGORITHMIQUE

*Les boucles*



52

## Répétition d'un traitement boucle « pour »

Exemple: Cet algorithme fait la somme des nbVal données qu'il saisit

```
Algorithme FaitLeTotal
variables nbVal, cpt : entiers
          valeur, totalValeurs : réels
début
          ecrire("Combien de valeurs voulez-vous saisir ?")
          lire(nbVal)
          totalValeurs ← 0
          pour cpt ←1 à nbVal faire
                    ecrire("Donnez une valeur :")
                    lire(valeur)
                    totalValeurs ← totalValeurs + valeur
          finpour
          ecrire("Le total des ", nbVal, "valeurs est " , totalValeurs)
fin
```



53

## Boucle « pour »


```
pour  var ← valInit  à  valfin  faire
        traitement {suite d'instructions}
Finpour
```

- **Fonction :** répéter une suite d'instructions un certain nombre de fois
- Pour utilisée quand le nombre d'itération est connu



54

## Répétition d'un traitement à nombre itérations inconnu : « tant que … faire »

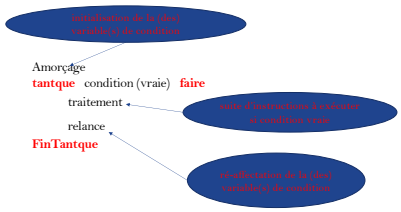
**Exemple :** Cet algorithme fait la somme des nbVal données qu'il saisit, arrêt à la lecture de -1

```
Algorithme FaitLeTotal
    constante STOP ← -1
    variables val, totalValeurs : entiers
    début
        totalValeurs←0
        ecrire("Donnez une valeur,", STOP, " pour finir.")
        lire(val)
        tantque val ≠ STOP faire
            totalValeurs←totalValeurs+ val {traitement}
            ecrire("Donnez une autre valeur,", STOP, " pour finir.")
            lire(val)
        fintantque
        ecrire("La somme des valeurs saisies est", totalValeurs)
    Fin
```



55

## Boucle «tant que … faire »




- **Fonction :** répéter une suite d'instructions un certain nombre de fois



56

## Comparaison boucles « pour » et « tant que » (1)

```
pour  cpt ←1  à nbVal faire
        ecrire("Donnez une valeur :")
        lire(valeur)
        totalValeurs ← totalValeurs + valeur
finpour

Est équivalent à

cpt ← 0
tantque cpt < nbVal faire
        ecrire("Donnez une valeur :")
        lire(valeur)
        totalValeurs ← totalValeurs + valeur
        cpt ←cpt + 1
fintantque
```



57

## Comparaison boucles « pour » et « tant que » (2)

- **Implicitement, l'instruction pour:**
  - initialise un compteur
  - incrémente le compteur à chaque pas
  - vérifie que le compteur ne dépasse pas la borne supérieure
- **Explicitement, l'instruction tantque doit**
  - initialiser un compteur {amorçage}
  - incrémenter le compteur à chaque pas {relance}
  - vérifier que le compteur ne dépasse pas la borne supérieure {test de boucle}



58

## Quand choisir « pour » ou « tant que » ?

- Nombre d'itération connu à l'avance : **POUR**
  - Parcours de tableaux
  - Test sur un nombre donné de valeurs
- Boucle s'arrête sur événement particulier : **TANTQUE**
  - Itération avec arrêt décidé par saisie utilisateur



59

## Boucle « répéter …tant que »

Exemple : Cet algorithme a besoin d'une valeur positive non nulle

```
Algorithme Essai
Variables valeur : entier
Début
    Répéter
        ecrire("Donnez une valeur positive non nulle : ")
        lire(valeur)
    tantque valeur ≤0
    ecrire("La valeur positive non nulle que vous avez saisie est ")
    ecrire( valeur )
fin
```



60

## Boucle « répéter …tant que »

```
Répéter
    (ré)affectation de la (des) variable(s) de condition
    traitement
tantque condition (vraie)
```

- **Fonction :** exécuter une suite d'instructions **au moins une fois** et la répéter tant qu'une condition est remplie
- **Remarque :** le traitement dans l'exemple précédent se limite à la réaffectation de la variable de condition ( **lire**(valeur) )



61

## Comparaison «répéter» et «tant que»

```
Répéter
        ecrire("Donnez une valeur positive paire :")
        lire(valeur)
tantque ( valeur < 0 OU (valeur % 2) ≠0 )
```

**Équivaut à**

```
ecrire("Donnez une valeur positive paire :")
lire(valeur)
tantque ( valeur < 0 OU (valeur % 2) ≠0 ) faire
        ecrire("Donnez une valeur positive paire :")
        lire(valeur)
fintantque
```



62

## Comparaison «répéter» et «tant que»

- boucle tant que
  - condition vérifiée avant chaque exécution du traitement
  - le traitement peut donc ne pas être exécuté
  - de plus : la condition porte surtout sur la saisie de nouvelles variables (relance)
- boucle répéter … tant que
  - condition vérifiée après chaque exécution du traitement =>le traitement est exécuté au moins une fois
  - de plus: la condition porte surtout sur le résultat du traitement

**Remarque :** la boucle répéter est typique pour les saisies avec vérification



63

## Exercice

Un responsable de la comptabilité d'un service de vente d'une boutique en ligne souhaite construire un algorithme simulant une caisse automatique, l'algorithme lit le nombre d'articles commandé (N), puis il demande la saisie du prix hors taxe (pht) de chaque article ainsi que sa quantité commandée (qte) et calcule et affiche le prix TTC (pttc) de l'article en question sachant que la TVA=20%, l'algorithme doit calculer aussi la somme totale de tous les prix TTc les articles commandés (totalFacture).

Ecrire un algorithme pour satisfaire la demande du responsable.



64