Université de Badji MoKhtar – Annaba (UBMA)
Faculté de Médecine
Département de Médecine

Module : Initiation à l'Informatique

Enseignants: S.BENTRAD & A.BOULEMDEN
LRI Laboratory / Badji Mokhtar-Annaba University
23000, BP 12, Annaba, Algeria
sassi_bentrad@hotmail.fr
a.boulemden@yahoo.com

Année Pédagogique: 1ème Année *Année Universitaire: 2012 / 2013*

INTRODUCTION À L'ALGORITHMIQUE

Les instructions conditionnelles et itératives

```
Algorithme CalculMoy
Variable
  N1,N2,N3 : réel (* les 3 notes *)
  C1,C2,C3 : réel (* les 3 coefficients *)
  MOY : réel (* la moyenne *)
Début
(* Traitement étudiant 1 *)
  Lire(N1, C1)
  Lire(N2, C2)
  Lire(N3, C3)
  MOY←(N1*C1+N2*C2+N3*C3)/(C1+C2+C3)
  Écrire(MOY)
(* Traitement étudiant 2 *)
  Lire(N1, C1)
  Lire(N2, C2)
  Lire(N3, C3)
  MOY←(N1*C1+N2*C2+N3*C3)/(C1+C2+C3)
  Écrire(MOY)
Fin
```

```
Algorithme CalculMoy
Variable
  N1,N2,N3 : réel (* les 3 notes *)
  C1,C2,C3 : réel (* les 3 coefficients *)
  MOY : réel (* la moyenne *)
Début
    Lire(N1, C1)
    Lire(N2, C2)
    Lire(N3, C3)
    MOY ← (N1*C1+N2*C2+N3*C3)/(C1+C2+C3)
    Écrire(MOY)
Fin .
```

Algorithme Inverse
Variables nombre: réel
Debut

Lire (nombre)
Ecrire (‘ L’inverse du nombre : ‘, nombre, ‘ est : ‘, 1/nombre)

Fin

INTRODUCTION

- L'exécution de ce programme va causer une fin anormale si le nombre tapé est nul (**zéro**).

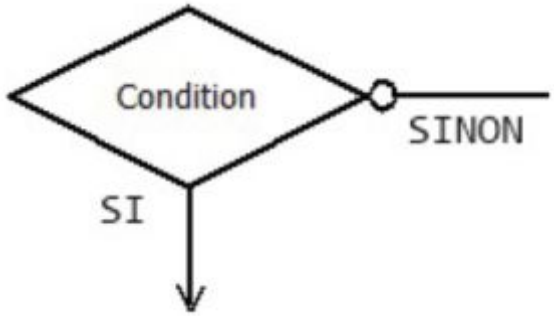
- Pour traiter ce genre de problèmes en programmation, on a besoin que l'ordinateur prenne une discision et choisisse un chemin d'exécution. **Ce qui veut dire ; ne pas exécuter forcement toutes les instructions d'un programme.**

- Dans cet exemple, l'ordinateur ne doit calculer l'inverse d'un nombre sauf **si ce nombre n'est pas nul**.
- Les instructions de base vues jusqu'à présent : **l'affectation**, la **lecture** et **l'écriture** ne peuvent pas répondre à ce besoin, ce sont des instructions impératifs.

- Donc, on a besoins de nouvelles instructions, se sont: **les instructions conditionnelles**.

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** est une expression logique elle peut être **vraie** ou **fausse**
- si la **condition** est vraie, se sont les **instructions1** qui seront exécutées
- si la **condition** est fausse, se sont les **instructions2** qui seront exécutées
- la **condition** peut être une **condition simple** ou une **condition composée** de plusieurs conditions

TESTS: INSTRUCTIONS CONDITIONNELLES (2)

- 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:

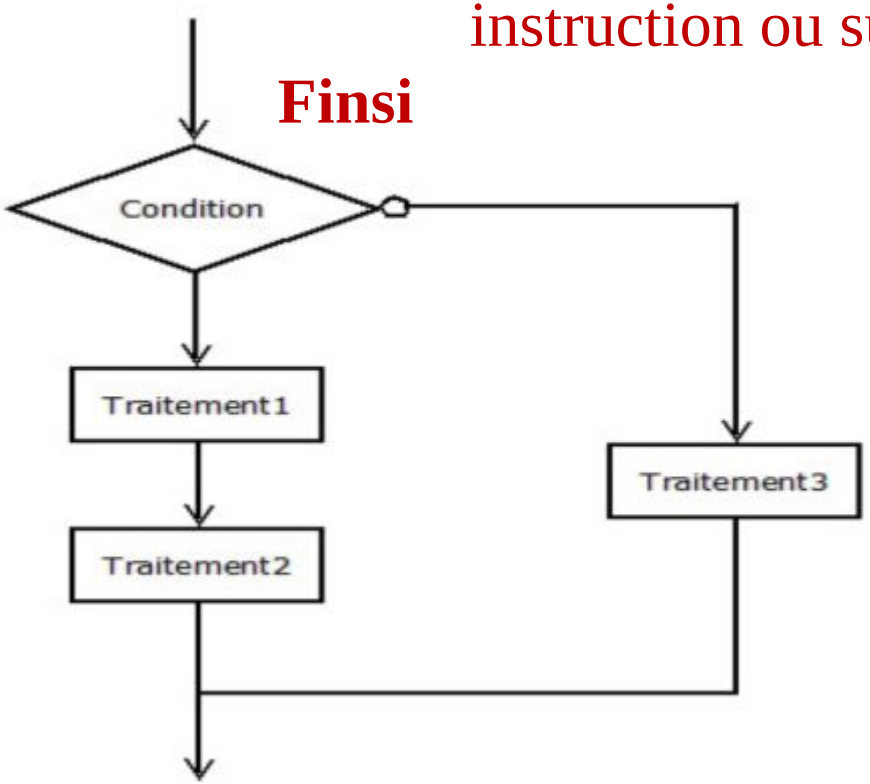
Si condition **alors**

instruction ou suite d'instructions1

Sinon

instruction ou suite d'instructions2

Finsi



TESTS: INSTRUCTIONS CONDITIONNELLES (3)

- 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 COND ALORS
  INSTRUCTION1
  INSTRUCTION2      } BLOC1
  ...............
  INSTRUCTION N
SINON
  INSTRUCTION1
  INSTRUCTION2      } BLOC2
  ...............
  INSTRUCTION N
FINSI
```

```
IF COND THEN
  BEGIN
    INSTRUCTION1 ;
    INSTRUCTION2 ;
    ...............
    INSTRUCTION N ;
  END (pas de ; )
ELSE
    BEGIN
    INSTRUCTION1 ;
    INSTRUCTION2 ;
    ...............
    INSTRUCTION N ;
    END ;
```

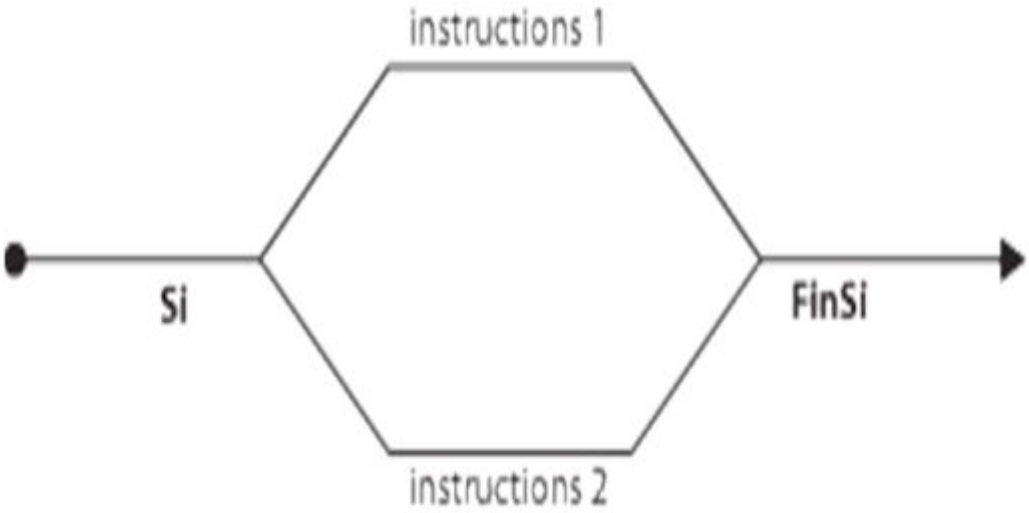
TESTS: INSTRUCTIONS CONDITIONNELLES (4)

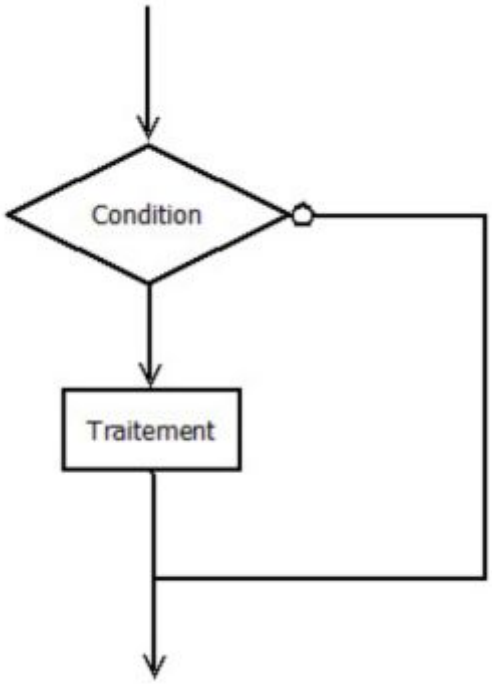
- La partie **Sinon** n'est pas obligatoire, quand elle n'existe pas et que la condition est fausse, **aucun traitement n'est réalisé**
- On utilisera dans ce cas la forme simplifiée suivante:

Si condition **alors**

instruction ou suite d'instructions1

Finsi





TESTS: INSTRUCTIONS CONDITIONNELLES (5)

- 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 COND ALORS
  INSTRUCTION1
  INSTRUCTION2
  ...............
  INSTRUCTION N
FINSI
```

```
IF COND THEN
  BEGIN
    INSTRUCTION1 ;
    INSTRUCTION2 ;
    ...............
    INSTRUCTION N ;
  END ;
```

- **Exemples (Si…Alors)**

```
Algorithme  Inverse1
Variables  nombre: réel
Debut

        Lire (nombre)
        Si ( nombre ≠ 0) alors
            Ecrire (‘ L’inverse du nombre : ‘, nombre, ‘ est : ‘, 1/nombre)
        FinSi

        Si (nombre =0) alors
            Ecrire (‘le nombre  tapé est nul ‘)
        FinSi
Fin
```

TESTS: INSTRUCTIONS CONDITIONNELLES (7)

- **Exemples (Si…Alors)**

Algorithme AffichageValeurAbsolue (version 1)

Var 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.

TESTS: INSTRUCTIONS CONDITIONNELLES (8)

- **Exemples (Si…Alors…Sinon)**

```
Algorithme  Inverse1
Variables  nombre: réel
Debut
        Lire (nombre)
        Si ( nombre ≠ 0) alors
           Ecrire (‘ L’inverse du nombre : ‘, nombre, ‘ est : ‘, 1/nombre)
        Sinon
           Ecrire (‘le nombre  tapé est nul ‘) ;
        FinSi
Fin
```

- **Exemples (Si…Alors…Sinon)**

```
Algorithme AffichageValeurAbsolue (version 2)
Var  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.
```

- **Exercice (tests)**

Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui teste et affiche s'il est divisible par 3

```
Algorithme Divsible_par3
Var  n : entier;
Début
    Ecrire (" Entrez un entier : “);
    Lire (n);
    Si (n mod 3 = 0) alors
        Ecrire (n," est divisible par 3")
    Sinon
        Ecrire (n," n'est pas divisible par 3");
    Finsi;
Fin.
```

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**

TESTS: INSTRUCTIONS CONDITIONNELLES (12)

- **Conditions composées: Tables de vérité**

C1	C2	C1 ET C2
VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	FAUX

C1	C2	C1 OU C2
VRAI	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

C1	C2	C1 XOR C2
VRAI	VRAI	FAUX
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

C1	NON C1
VRAI	FAUX
FAUX	VRAI

- **Tests imbriqués**

Les tests peuvent avoir un degré quelconque d'imbrications

Si condition1 **alors**

Si condition2 **alors**

instructionsA

Sinon

instructionsB

Finsi

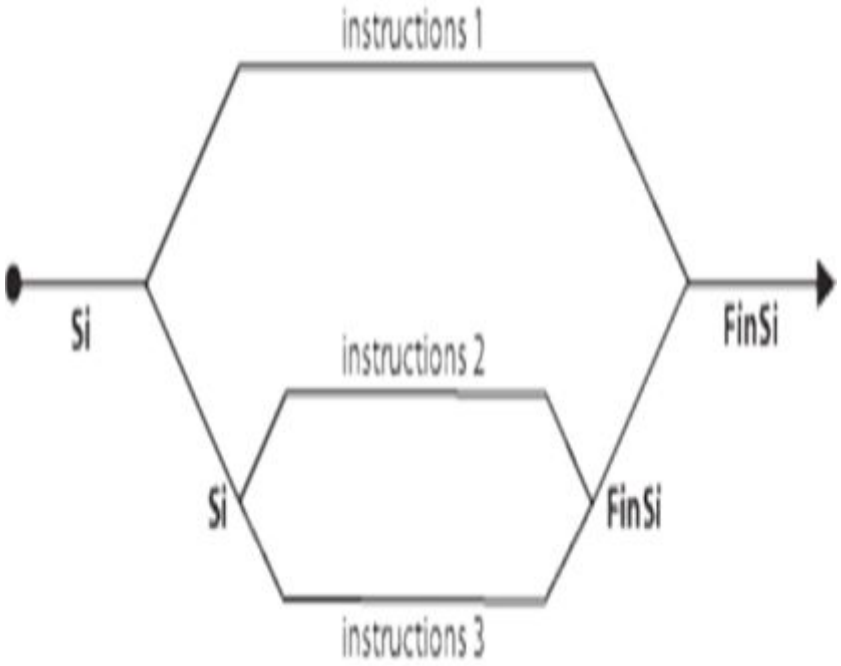
Sinon

Si condition3 **alors**

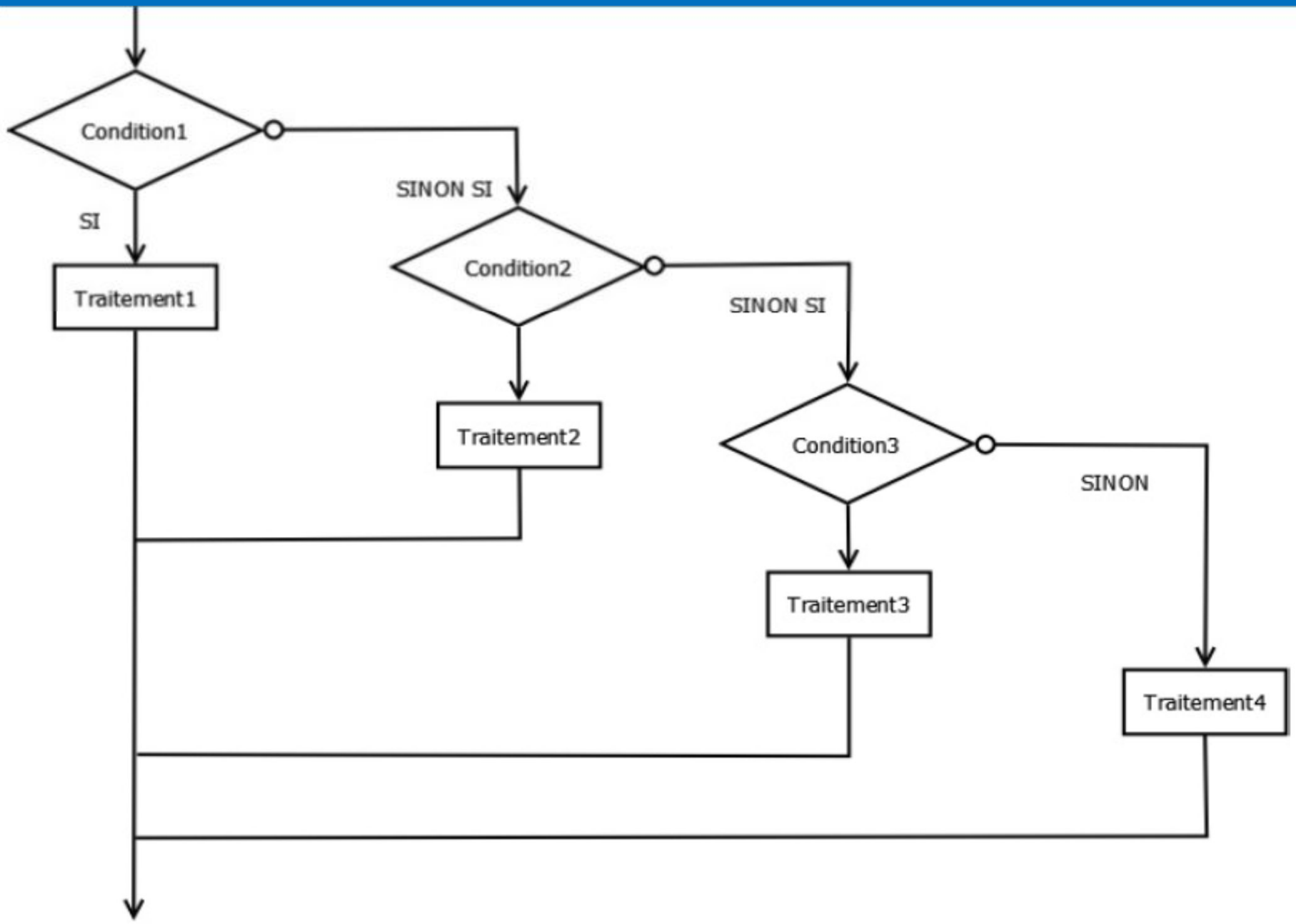
instructionsC

Finsi

Finsi



TESTS: INSTRUCTIONS CONDITIONNELLES (14)



- **Tests imbriqués**

```
SI COND 1 ALORS
    BLOC1
SINON
    SI COND 2 ALORS
        BLOC2
    SINON
        SI COND 3 ALORS
            BLOC3
        SINON
            BLOC4
        FINSI
    FINSI
FINSI
```

```
IF COND1 THEN
  BEGIN
    BLOC1
  END (pas de ; )
ELSE
    IF COND2 THEN
      BEGIN
        BLOC2
      END (pas de ; )
    ELSE
        IF COND3 THEN
          BEGIN
            BLOC3
          END (pas de ; )
        ELSE
          BEGIN
            BLOC4
          END ;
```

- **Tests imbriqués: exemple (version 1)**

```
Var 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

- **Tests imbriqués: exemple (version 2)**

```
Var 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.
```

- **Tests imbriqués: exercice**

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

- **Tests imbriqués: corrigé de l'exercice**

```
Variables copies : entier
          prix : réel
Début
    Ecrire ("Nombre de photocopies : ")
    Lire (copies)
    Si (copies < 10) Alors
        prix ← copies*0.5
    Sinon Si (copies) < 20 ) alors
            prix ← copies*0.4
        Sinon
            prix ← copies*0.3
        Finsi
    Finsi
    Ecrire ("Le prix à payer est : ", prix)
Fin
```

TESTS: INSTRUCTIONS CONDITIONNELLES (20)

❑ Tests imbriqués

```
Algo température;
Var Temp: entier
Début
Ecrire ("Entrez la température de l'eau :« );
Lire (Temp);

Si (Temp =< 0) Alors
  Ecrire ("C'est de la glace« );
FinSi

Si (Temp > 0 Et Temp < 100) Alors
  Ecrire ("C'est du liquide");
Finsi

Si (Temp > 100) Alors
  Ecrire ("C'est de la vapeur« );
Finsi

Fin
```

```
Algo température;
Var Temp: entier

Début

Ecrire ("Entrez la température de l'eau :« );
Lire (Temp);

Si (Temp =< 0) Alors
  Ecrire "C'est de la glace"
Sinon

  Si (Temp < 100) Alors
    Ecrire ("C'est du liquide« )
  Sinon
    Ecrire ("C'est de la vapeur");
  Finsi

Finsi

Fin.
```

❑ Exercice récapitulatif

Ecrire un algorithme qui :

- Lit le nom, prénom,le groupe d'un étudiant.
- Lit la note de TP et la note de l'examen (entre 0.00 et 20.00)
- Lit le nombre d'absences de l'étudiant.
- Calcul la moyenne de l'étudiant selon la formule suivante :

Moyenne = (note TP + Note Examen*2) /3

- L'étudiant est évalué selon le barème suivant :
  1. Si Moyenne ≥ 10 et Nombre d'absence ≤ 3 : l'étudiant est Bon
  2. Si Moyenne ≥ 10 et Nombre d'absence > 3 : l'étudiant n'est pas sérieux
  3. Sinon l'étudiant est mauvais.
- L'algorithme affiche l'évaluation de l'étudiant selon la forme suivante :

Nom
Prénom
Groupe
Moyenne
Observation

Exercice récapitulatif

```
Algorithme  Evaluation ;
Var  nom, prénom, groupe, observation : chaîne de caractères
            Exam, TP , Moy : réel
            NbAbsence : entier
Début
      Lire(nom,prénom,groupe)
      Lire (Exam,TP)
      Lire(NbAbsence)
      Si  (Exam>=0)et(Exam<=20)et(TP>=0)et(TP<=20)  alors
            Moy = (Exam*2+ TP) /3 ;
            Si (Moy <=10) alors
                  Observation ← ‘ Mauvais’
            Sinon
                  Si  (NbAbsence <=3)  alors
                        Observation ← ‘ Bon’
                  Sinon
                        Observation ← ‘ Pas Sérieux’ ;
                  FinSi
            FinSi
      Finsi
      Ecrire(Nom) ;
      Ecrire(Prénom) ;
      Ecrire(groupe) ;
      Ecrire(Moy) ;
      Ecrire(Observation) ;
Fin.
```

BOUCLES: INSTRUCTIONS ITÉRATIVES (1)

- **Les boucles** servent à répéter l'exécution d'un groupe d'instructions un certain nombre de fois
- On distingue trois sortes de boucles en langages de programmation :

  - Les **boucles tant que** : on y répète des instructions tant qu'une certaine condition est réalisée
  - Les **boucles jusqu'à** : on y répète des instructions jusqu'à ce qu'une certaine condition soit réalisée
  - Les **boucles pour** ou avec compteur : on y répète des instructions en faisant évoluer un compteur (variable particulière) entre une valeur initiale et une valeur finale

- (Dans ce cours, on va s'intéresser essentiellement aux boucles ***Tant que*** et boucles ***Pour*** qui sont plus utilisées)

BOUCLES: INSTRUCTIONS ITÉRATIVES (2)

- **Les boucles Tant que**

Initialisation des variables de la condition d'exécution
Tant que (Condition_exécution_vérifiée)
Faire
    <Traitement>
    <mise_à_jour_condition_exécution>
Fin Tant que

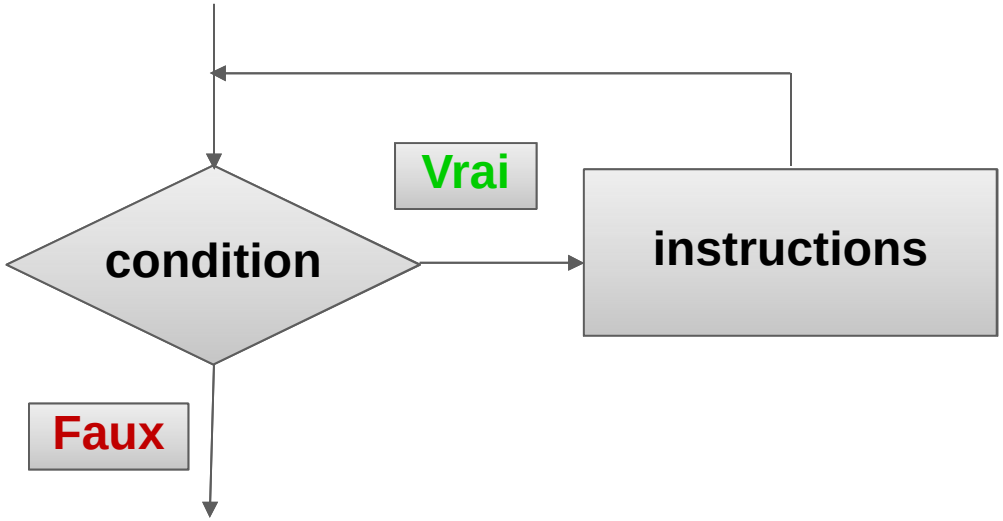
- la **condition** (dite condition de contrôle de la boucle) est évaluée avant chaque **itération**

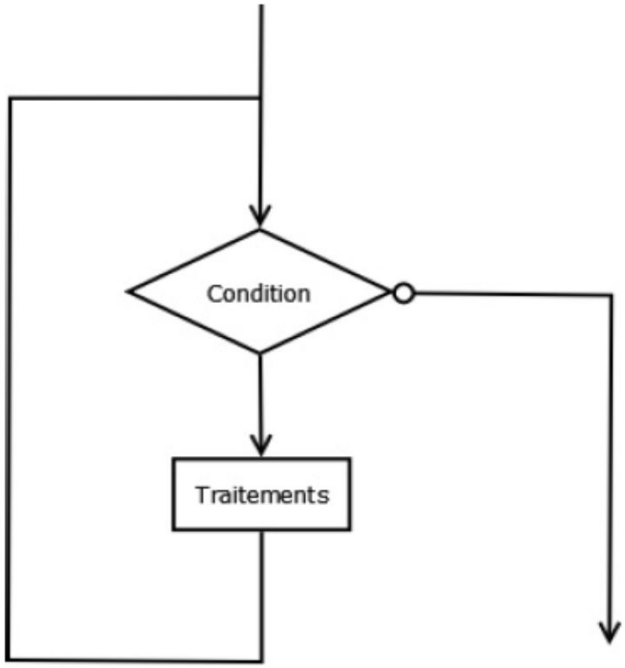
- **si la condition est vraie**, on exécute instructions (corps de la boucle), puis, on retourne tester la condition. **Si elle est encore vraie, on répète l'exécution**, …

- **si la condition est fausse**, on sort de la boucle et on exécute l'instruction qui est après **Fin Tant Que**

BOUCLES: INSTRUCTIONS ITÉRATIVES (3)

- **Les boucles Tant que**





- la **condition** (dite condition de contrôle de la boucle) est évaluée avant chaque **itération**

- **si la condition est vraie**, on exécute instructions (corps de la boucle), puis, on retourne tester la condition. **Si elle est encore vraie, on répète l'exécution**, …

- **si la condition est fausse**, on sort de la boucle et on exécute l'instruction qui est après **Fin Tant Que**

BOUCLES: INSTRUCTIONS ITÉRATIVES (4)

- **Les boucles Tant que**

```
TANT QUE  COND FAIRE
|
|        Bloc
FIN TANT QUE
```

```
WHILE  COND Do
    Begin
        Bloc
    End ;
```

- **Les boucles Tant que : remarques**

  - Le nombre d'itérations dans une **boucle TantQue** n'est pas connu au moment d'entrée dans la boucle. Il dépend de l'évolution de la valeur de condition

  - Une des instructions du corps de la boucle doit absolument changer la valeur de condition de vrai à faux (après un certain nombre d'itérations), sinon le programme tourne indéfiniment

Attention aux boucles infinies

  - Exemple de boucle infinie :

i ← 2

TantQue (i > 0)

i ← i+1 **(attention aux erreurs de frappe : + au lieu de -)**

FinTantQue

❑ Les boucles Tant que : exemple 1

Un algorithme qui détermine le premier nombre entier N tel que la somme de 1 à N dépasse strictement 100

<u>version 1</u>

```
Variables som, i : entier;
Debut
        i ← 0 ;
        som← 0 ;
        TantQue (som <=100)  faire
                i ←  i+1;
                som ←  som+i;
        FinTantQue

        Ecrire (" La valeur cherchée est N=  ", i) ;
Fin
```

❑ Les boucles Tant que : exemple 1 (Version 2)

Un algorithme qui détermine le premier nombre entier N tel que la somme de 1 à N dépasse strictement 100

<u>version 2:</u> attention à l'ordre des instructions et aux valeurs initiales

```
Variables som, i : entier;
Debut
        som ← 0;
        i ← 1 ;
        TantQue (som <=100) faire
                som ← som + i;
                i ←  i+1;
        FinTantQue
        Ecrire (" La valeur cherchée est N=  ", i-1) ;
Fin
```

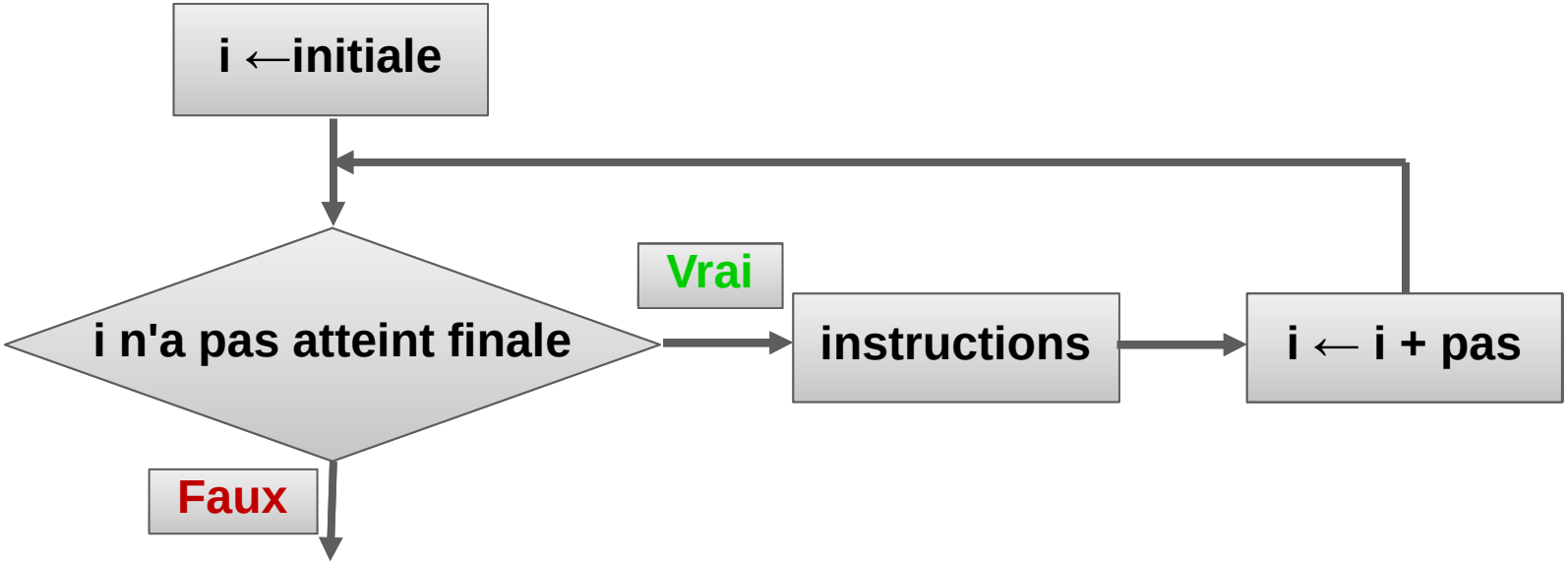
❑ Les boucles Pour

Pour compteur **allant de** initiale **à** finale par **pas** valeur du pas

instructions

FinPour

Pour I **allant** de Val_initiale **jusqu'à** Val_finale, (Pas = Val_pas) **Faire**
<Traitement>
Fin Pour



- **Les boucles Pour**

```
POUR i ALLANT DE valeur initiale à valeur finale FAIRE
    Bloc
FINPOUR
```

```
FOR i :=valeur initiale To valeur finale DO
    BEGIN
        Bloc ;
    END ;
```

BOUCLES: INSTRUCTIONS ITÉRATIVES (11)

- **Les boucles Pour**

- **Remarque:** le nombre d'itérations dans une **boucle Pour** est connu avant le début de la boucle

- **Compteur** est une variable de type entier (ou caractère). Elle doit être déclarée

- **Pas** est un entier qui peut être positif ou négatif. **Pas** peut ne pas être mentionné, car par défaut sa valeur est égal à 1. Dans ce cas, le nombre d'itérations est égal à finale - initiale+ 1

- **Initiale et finale** peuvent être des valeurs, des variables définies avant le début de la boucle ou des expressions de même type que compteur

❑ Déroulement des boucles Pour

La valeur initiale est affectée à la variable compteur

On compare la valeur du compteur et la valeur de finale :

- ❑ Si la valeur du compteur est > à la valeur finale dans le cas d'un pas positif (ou si compteur est < à finale pour un pas négatif), on sort de la boucle et on continue avec l'instruction qui suit FinPour
- ❑ Si compteur est <= à finale dans le cas d'un pas positif (ou si compteur est >= à finale pour un pas négatif), instructions seront exécutées
  - Ensuite, la valeur de compteur est incrémentée de la valeur du pas si pas est positif (ou décrémenté si pas est négatif)
  - On recommence l'étape 2 : La comparaison entre compteur et finale est de nouveau effectuée, et ainsi de suite …

❑ Boucle Pour : exemple 1

Calcul de **x à la puissance n** où x est un réel non nul et n un entier positif ou nul

```
Var  x, puiss : réel ;
                n, i : entier;
Debut
        Ecrire (" Entrez la valeur de x ");
        Lire (x);
        Ecrire (" Entrez la valeur de n ");
        Lire (n);

        puiss ← 1;
        Pour i allant de 1 à n  faire
                puiss← puiss*x ;
        FinPour

        Ecrire (x, " à la puissance  ", n, "  est égal à  ", puiss);
Fin
```

❑ Boucle Pour : exemple 1 (version 2)

Calcul de **x à la puissance n** où x est un réel non nul et n un entier positif ou nul (**version 2 avec un pas négatif**)

```
Var x, puiss : réel ;
                n, i : entier;
Début
    Ecrire (" Entrez respectivement les valeurs de x et n ");
    Lire (x, n);
    puiss ← 1 ;
    Pour i allant de n à 1 par pas -1  faire
            puiss← puiss*x ;
    FinPour

    Ecrire (x, " à la puissance  ", n, "  est égal à  ", puiss);
Fin
```

- **Boucle Pour : remarque**

  - Il faut éviter de modifier la valeur du **compteur** (et de finale) à l'intérieur de la boucle. En effet, une telle action :
    - perturbe le nombre d'itérations prévu par la boucle Pour
    - rend difficile la lecture de l'algorithme
    - présente le risque d'aboutir à une boucle infinie

Exemple : **Pour** i **allant de** 1 **à** 5 **faire**

i ← i -1;

écrire(" i = ", i) ;

Finpour

❑ Lien entre Pour et TantQue

La boucle Pour est un cas particulier de Tant Que (cas où le nombre d'itérations est connu et fixé) . Tout ce qu'on peut écrire avec Pour peut être remplacé avec TantQue (la réciproque est fausse)

Pour compteur **allant de** initiale **à** finale par **pas** valeur du pas **faire**

instructions ;

FinPour

peut être remplacé par : compteur ← initiale;

(cas d'un pas positif)

TantQue compteur <= finale **faire**

instructions ;

compteur ← compteur + pas;

FinTantQue

❑ Lien entre Pour et TantQue: exemple

Calcul de x à la puissance n où x est un réel non nul et n un entier positif ou nul (**version avec TantQue**)

```
Var x, puiss : réel
                n, i : entier
Debut
        Ecrire (" Entrez la valeur de x ");
        Lire (x) ;
        Ecrire (" Entrez la valeur de n ");
        Lire (n);

        puiss ← 1;
        i ← 1;
        TantQue (i<=n) faire
                puiss← puiss*x ;
                i ← i+1;
        FinTantQue

        Ecrire (x, " à la puissance  ", n, "  est égal à  ", puiss);
Fin
```

- **Boucles imbriquées**

  - Les instructions d'une boucle peuvent être des instructions itératives. Dans ce cas, on aboutit à des **boucles imbriquées**

  - **Exemple:**

```
Pour i allant de 1 à 5
        Pour j allant de 1 à i
                écrire("O")
                FinPour
        écrire("X")
FinPour
```

Exécution

```
OX
OOX
OOOX
OOOOX
OOOOOX
```

BOUCLES: INSTRUCTIONS ITÉRATIVES (19)

- **Les boucles Répéter … jusqu'à …**

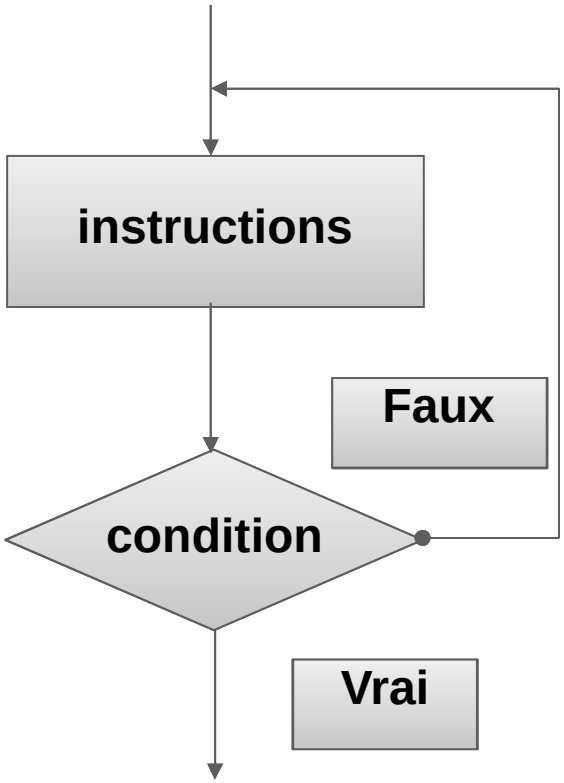
Répéter

instructions;

Jusqu'à condition

```
RÉPÉTER
    Bloc
JUSQU'À COND
```

```
REPEAT
    Bloc
UNTILL (Cond) ;
```



- Condition est évaluée après chaque itération

- les instructions entre ***Répéter*** et ***jusqu'à*** sont exécutées **au moins une fois** et leur exécution est répétée jusqu'à ce que condition soit vrai (tant qu'elle est fausse)

- **Boucle Répéter jusqu'à : exemple**

Un algorithme qui détermine le premier nombre entier N tel que la somme de 1 à N dépasse strictement 100 **(version avec répéter jusqu'à)**

```
Var som, i : entier;
Debut
    som ← 0;
    i ← 0;
    Répéter
        i ←  i+1;
        som ←  som+i;
    Jusqu'à ( som > 100)
    Ecrire (" La valeur cherchée est N=  ", i) ;
Fin
```

- **Choix d'un type de boucle**

- Si on peut déterminer le nombre d'itérations avant l'exécution de la boucle, il est plus naturel d'utiliser ***la boucle Pour***

- S'il n'est pas possible de connaître le nombre d'itérations avant l'exécution de la boucle, on fera appel à l'une des *boucles* ***TantQue*** *ou* ***répéter jusqu'à***

- Pour le choix entre ***TantQue*** et ***jusqu'à*** *:*

  - Si on doit tester la condition de contrôle avant de commencer les instructions de la boucle, on utilisera ***TantQue***

  - Si la valeur de la condition de contrôle dépend d'une première exécution des instructions de la boucle, on utilisera ***répéter jusqu'à***

❑ Comparer les trois schémas répétitifs

Critère	Répéter(1)	Tant que (2)	Pour (3)
Traitement	Au moins 1 fois	De 0 à N fois	De Val_initiale jusqu'à Val_finiale
Initialisation	Oui avant Répéter	Oui avant Tant que	Oui : Val_initiale
Mise à Jour Variable	Oui après traitement	Oui après traitement	Oui, Automatique en fonction du Pas
Sortie de la boucle	Condition_arrêt atteinte	Condition_exécution non vérifiée	Compteur dépassant les bornes
Type de la condition	Expression logique avec ET, OU, etc.	Expression logique avec ET, OU, etc.	Une seule variable de type entier
Passage d'un schéma à l'autre	(1) → (2) oui sans problème (1) → (3) n'est pas toujours possible	(2) → (1) oui sans problème (2) → (3) n'est pas toujours possible	(3) → (1) oui sans problème (3) → (2) oui sans problème

Un site web pour vous

Medical Recorder

Best way to record, save and share your notes

https://www.recordermedical.com/

https://itunes.apple.com/

Description:

Medical Recorder is a secure and convenient service for doctors and medical students to take voice notes, along with text and images. It offers unbeatable recording quality and a set of unique features.

Main features:

Record voice with unbeatable clarity

Skip silence with unique voice detection function

Add photos or text to your note

Save your notes in the cloud

or send them via e-mail

Merci de votre attention !

LRI Laboratory / Badji Mokhtar-Annaba University (UBMA)
23000, BP 12, Annaba, Algeria
sassi_bentrad@hotmail.fr & a.boulemden@yahoo.com