Instructions Conditionnelles et Alternatives

Instructions élémentaires en algorithmique

Une **affectation** est exécutable si et seulement si la **partie droite** de l'affectation représente une **valeur bien définie.**

$$V \leftarrow A/B$$

- ***Cas 1 :***

  A = 12

  B = 4

  V ← 12/4

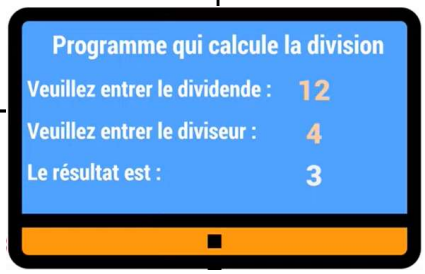
L'instruction d'***affectation*** **est exécutée** *et la variable* ***V*** *contiendra le résultat de la division réelle, donc* ***V = 3***

Question?

Ecrire un **algorithme** qui permet de **calculer et d'afficher** le **résultat** de la **division de deux nombres réels A et B**?

Instructions élémentaires en algorithmique

```
Algorithme Division;
  Var
        A, B, V: Réel;
  Début
        Ecrire ("Programme qui calcule la division");
        Ecrire ("Veuillez entrer le dividende :");
        Lire (A);
        Ecrire ("Veuillez entrer le diviseur :");
        Lire (B);
        V ← A/B;
        Ecrire ("Le résultat est : ", V);
  Fin
```



Instructions élémentaires en algorithmique

Une **affectation** est exécutable si et seulement si la **partie droite** de l'affectation représente une **valeur bien définie.**

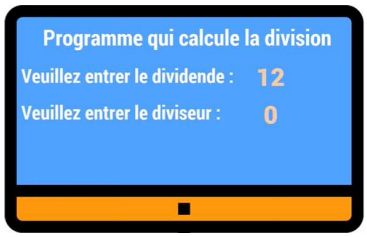
$V \leftarrow A/B$

- **Cas 2 :**

  $A = 12$

  $B = 0$

  $V \leftarrow 12/0$



ERREUR **: Division par zéro**
L'affectation n'**est pas exécutée**

Question?

Comment **améliorer l'algorithme** précédent, pour qu'il **prend en compte également** ce cas particulier (Division par zero) ?

Il faut utiliser les **STRUCTURES CONDITIONNELLES**

Structure séquentielle

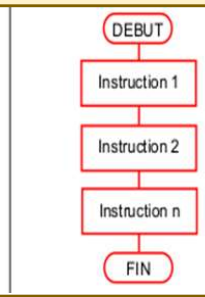
Un **algorithme** qui suit une **structure séquentielle** est un algorithme dont **toutes les instructions** sont **exécutées l'une après l'autre** de façon à ce que **l'ordre des instructions est respecté.**

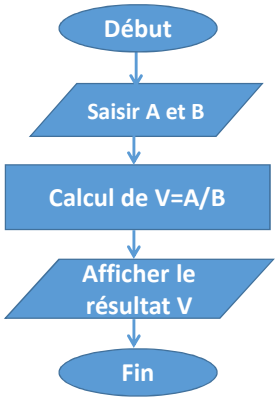
➔ Les **instructions sont toutes exécutées**, dans l'ordre ou elles **sont écrites.**

➔ Les **instructions sont écrites** l'**une après l'autre**, séparées par un **saut de ligne**

Structure LINEAIRE ou SEQUENTIELLE

- Instruction 1
- Instruction 2
- …
- Instruction n





Exemples :

Les algorithmes des diapos précédents.

4. Instructions conditionnelles et alternatives

4. 1. Notion de primitives de base structurées

Les **algorithmes** vus, jusqu'à présent, ont un **enchainement séquentiel et linéaire.**

Dans de nombreuses applications, on exige une **exécution par morceau**, des **sauts** ou des **répétitions d'un même bloc** d'instructions. *C'est le rôle des* ***primitives de base structurées***

On distingue **deux grandes familles d'instructions composées** :

1. **Les primitives de choix (conditionnelles et alternatives)**
2. **Les primitives d'itération (Boucles)**

58

4. Instructions conditionnelles et alternatives

4. 1. Notion de primitives de base structurées

1. Les **primitives de choix** qui permet de **choisir les instructions à exécuter** selon les valeurs courantes de certaines variables; elle sont de deux types:
   - ✓ **Les instructions conditionnelles**
   - ✓ **Les instructions alternatives.**
2. Les **primitives d'itération** qui sont utilisées lorsque l'on souhaite exécuter plusieurs fois le même traitement (*cf. chapitre suivant*)

Remarque: Les notions de choix et d'itération sont deux notions fondamentales de l'algorithmique.

59

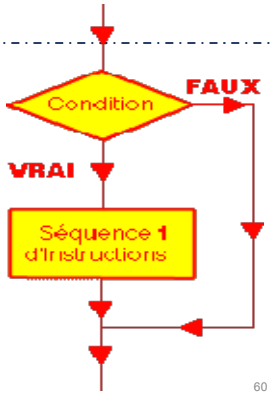
4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle simple (un seul choix)

Syntaxe:

Si (Condition) alors
Séquence 1 d'instructions;
Finsi

Organigramme:



1. **Si** la **condition** vaut **Vraie** alors la **séquence 1 d'instructions sera exécutée**,
2. **Sinon**, (condition est **Fausse**) la **séquence 1 d'instructions sera ignorée**.

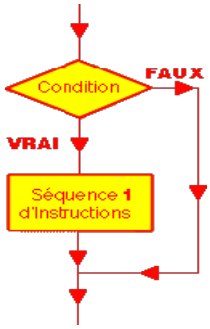
60

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle simple (Si … Alors … Finsi)

1. Si ***Condition*** est **VRAIE** alors la ***<Séquence 1 d'instructions>*** est exécutée
2. Si ***Condition*** est **FAUSSE** alors la ***<Séquence 1 d'instructions>*** n'est pas exécutée.

➔ On passe **directement** à l'instruction qui se trouve après ***Finsi***



Exemples de condition:

- **(A=1)** *: condition est **vérifiée** si la valeur contenu dans **A** égale à **1.***
- **(A=B)** *: est **vérifiée** si les valeurs contenu dans **A et B sont les mêmes.***
- **(B <> 5)** *: est **vérifiée** si **B** contient une valeur différente de **5.***
- **(1>5)** *: est toujours **FAUSSE*** et **ne dépend** pas des valeurs des variables.

61

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle simple (un seul choix)

Question? : Comment **améliorer l'algorithme précédent**, pour qu'il prend **également** en compte le cas particulier **(Division par zero)** ?

Utiliser les **STRUCTURES CONDITIONNELLES**
Quelle CONDITION à utiliser ?

```
Algorithme Division;
  Var
        A, B, V : Réel;
  Début
        Ecrire ("Programme qui calculi la division");
        Ecrire ("Veuillez entrer le dividende");
        Lire (A);
        Ecrire ("Veuillez entrer le diviseur");
        Lire (B);
        Si (B<>0) Alors
              V ← A/B;
              Ecrire ("Le résultat est: ", V);
        FinSi
  Fin
```

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle simple (un seul choix)

Exercice 4.1 :

La **valeur absolue** d'un nombre réel :

|x|=x si **x>0**

|x|=-x si **x<0**

Analyse du problème :

- ***Donnée en entrée :***

  x : nombre réel

- ***Valeur de sortie :***

  |x| : valeur absolue du nombre réel **x**

- ***Traitement (Comment faire ?) :***

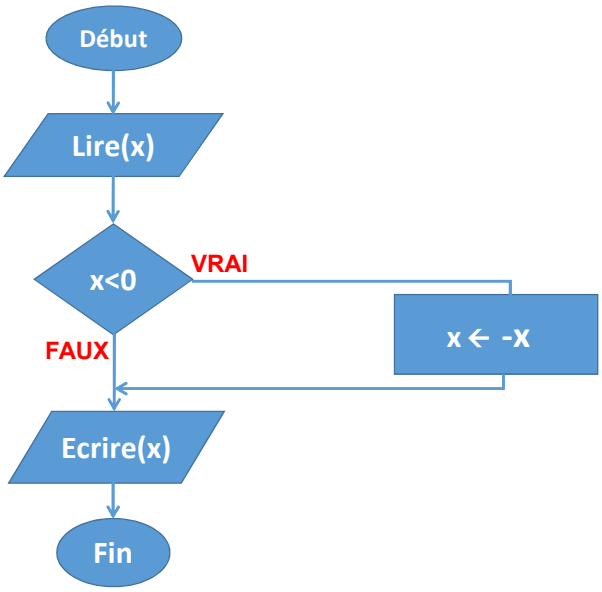
  On change le signe de **x** s'il est inférieur à **zéro**,
  Sinon il est affiché tel qu'il est.

63

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle simple (un seul choix)

Organigramme



4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle simple (un seul choix)

La **structure conditionnelle** est une structure dont les **instructions sont exécutées** selon les **réponses des conditions**.

Algorithme :

```
Algorithme valeur_absolue;
Var
    x: réel;
Début
   Ecrire (" Entrer un nombre réel ");
   Lire(x);
   Si  (x<0)  alors
            x ⟵ -x;
   Finsi
   Ecrire(" La valeur absolue de ce nombre est ", x);
Fin
```


4. Instructions conditionnelles et alternatives

Exercice 2 :

Ecrire un algorithme qui **calcule le salaire d'un employé** à partir du **nombre d'heures** travaillées (**nh**), du **taux horaire** (**th**) et du **nombre d'années** de service.

Les employés ayant une ancienneté (**anc**) de **plus de 10 ans** bénéficient d'une **allocation supplémentaire de 150 Dh**.

Quelle appellation à utiliser?

Analyse du problème :

- ***Données d'entrées :***
  - **nbr_heures :** *nombre d'heures travaillées*
  - **tx_horaire :** *taux horaire (en Dirhams)*
  - **ancien :** *ancienneté (en Année)*
- ***Données de sortie (résultat attendu)***
  - ✓ **Salaire :** *salaire de l'employé (en Dirhams)*
- ***Traitement (Comment faire ?):***
  - **Calculer** le salaire de base : **Salaire ⟵ nbr_heures * tx_horaire**
  - **Recalculer** le salaire si l'employé a une ancienneté de plus de 10 ans :

```
Si (ancien > 10) Alors
        Salaire ⟵ Salaire + 150
FinSi
```

4. Instructions conditionnelles et alternatives

Exercie 2 :

Solution 1 :

```
Algorithme  calcul_salaire1;
Var
  nbr_heures, tx_horaire, ancien, salaire : Réel;
Début
    Ecrire(”Nombre d’Heures travaillées : ”) ;
    Lire(nbr_heures);
    Ecrire(”Taux horaire : ”);
    Lire(tx_horaire);
    Ecrire(”Ancienneté : ”);
    Lire(ancien);
    salaire ← nbr_heures * tx_horaire;
    Si (ancien > 10) Alors
          salaire ← salaire + 150;
    FinSi
    Ecrire (”Salaire de l employé = ”, salaire);
Fin.
```

Solution 2:

```
Algorithme  calcul_salaire2;
Const   ALL=150;
Var
nbr_heures, tx_horaire, salaire, ancien : Réel;
Début
   Ecrire(”Nbr Heures , Taux horaire,
Ancienneté:”);
    Lire(nbr_heures, tx_horaire, ancien );
    salaire ← nbr_heures * tx_horaire;
    Si (ancien > 10) Alors
        salaire ← salaire + ALL;
    FinSi
Ecrire (”Salaire de l employé = ”, salaire);
Fin.
```

67

4. Instructions conditionnelles

4. 1. Instruction : Si … Alors … Finsi

Syntaxe:

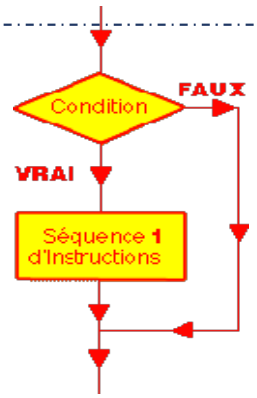
Si (Condition) alors

Séquence 1 d’instructions;

Finsi

Organigramme (un seul choix):

1. **Si** la **condition** vaut **Vraie** alors la **séquence 1 d’instructions sera exécutée**,
2. **Sinon**, (condition est **Fausse**) la **séquence 1 d’instructions sera ignorée**.



68

4. Instructions conditionnelles

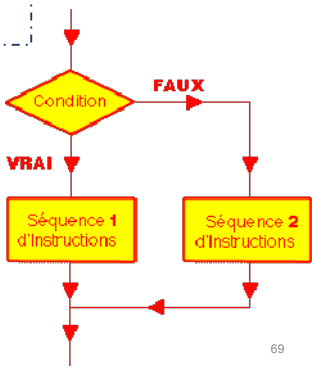
4. 2. Instruction : Si … Alors … Sinon … Finsi

Syntaxe:

```
Si (Condition) alors
        Séquence 1 d'instructions;
Sinon
        Séquence 2 d'instructions;
Finsi
```

Organigramme (deux choix):

Séquence 1 d'instructions est exécutée si la **condition est vérifiée, Sinon** c'est **la séquence d'instruction 2** qui sera exécutée.



69

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle simple (deux choix)

Exemple 1

```
Algorithme saisi;
Var
    a,b :  entier;
Début
  Ecrire(”Saisir deux nombres entiers a et b”);
  Lire(a, b);
  si  (a=b)  alors
     Ecrire(”vous avez saisi deux fois la même valeur, à savoir”, a);
  Sinon
     Ecrire(”vous avez saisi deux valeurs différentes”, a, ” et” , b,);
  Finsi
Fin
```

Instruction 1 est exécutée si la **condition est vérifiée, Sinon** c'est **l' instruction 2** qui sera exécutée.

70

4. Instructions conditionnelles et alternatives

4. 1. Instruction : Si … Alors … Sinon … Finsi

Exemple 2 : le nombre le plus grand entre x et y

Première forme

```
Algorithme maximum;
Var
    x,y: réel;
Début
   Ecrire(”donner deux nombres x et y”);
   Lire(x, y);
   si  (x>y)  alors
      Ecrire(”le plus grand est”, x);
   Sinon
      Ecrire(”le plus grand est”, y);
   Finsi
Fin
```

Instruction 1 est exécutée si la **condition est vérifiée, Sinon** c’est **l’instruction 2** qui sera exécutée.

71

4. Instructions conditionnelles et alternatives

4. 1. Instruction : Si … Alors … Sinon … Finsi

Exemple 2 : le nombre le plus grand entre x et y

Deuxième forme

```
Algorithme maximum;
Var
      x,y: réel;
      z: booléen;
Début
   Ecrire(”donner deux nombres x et y”);
   Lire(x,y);
   z ⟵ x>y;
   si (z) alors
      Ecrire(”le plus grand est”, x);
   Sinon
      Ecrire(”le plus grand est”, y);
   Finsi
Fin
```

72

4. Instructions conditionnelles

4. 1. Instruction : Si … Alors … Sinon … Finsi

Exemple 3: La valeur absolue de la différence de deux nombres réels:

|x-y|=x-y si **x>y**

|x-y|=y-x si **x<y**

Algorithmes: **Sans sinon**

```
Algorithme valeur_absolue2;
Var
    x, y, z: réels;
Début
  Ecrire ("Entrer deux nombres");
  Lire (x, y);
  z ⟵ x-y;
  Si (z<0) alors
        z ⟵ y-x;
  Finsi
  Ecrire ("Valeur absolue est ", z);
Fin
```

```
Algorithme valeur_absolue2;
Var
    x, y, z : réels;
Début
  Ecrire ("Entrer deux nombres réels");
  Lire (x, y);
  Si (x>y) alors
        z      x-y;
  Sinon
        z ⟵ y-x;
  Finsi
  Ecrire ("La valeur absolue est ", z);
Fin
```

4. Instructions conditionnelles et alternatives

Conditions composées

Une condition composée est une condition formée de plusieurs conditions simples reliées par des opérateurs logiques: **ET**, **OU**, **OU-Exclusif (XOR)**,**NON**

Exemples :

- Si **x** est comprise entre **2** et **5** :
  Si ((x>=2) ET (x<=5)) alors … Instructions …
- Si **y** est comprise strictement entre **3** et **6** :
  Si ((y>3) ET (y<6)) alors … Instructions …
- Si **m** est divisible par **3** ou par **2** :
  Si ((m MOD 3=0) OU (m MOD 2=0)) alors … Instructions …
- Parmi les valeurs de **x**, **y** et **z** deux valeurs au moins sont identiques:
  x = y OU x = z OU y = z
- Si deux valeurs et deux seulement sont identiques parmi **a, b et c :**
  ~~**(a=b) XOR (a=c) XOR (b=c)**~~ **: la condition est FAUSSE**
  (a = b ET a ≠ c) OU (a = c ET a ≠ b) OU (b = c ET a ≠ b)

L'évaluation d'une **condition composée** se fait selon les règles présentées dans ce qu'on appelle **Table de vérité**

74

4. Instructions conditionnelles et alternatives

Tables de vérités

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

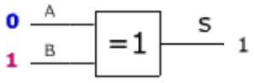
C1	C2	C1 XOR C2
VRAI	VRAI	FAUX
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

C1	NON (C1)
FAUX	VRAI
VRAI	FAUX

La fonction XOR (OU Exclusif)

$S = A \oplus B$

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0



La fonction "OU Exclusif" est en principe d'une fonction de deux variables :

S = A XOR B

La sortie est à 1 si une seule des deux entrées vaut 1.

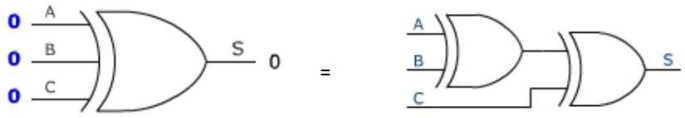
Porte XOR à plusieurs entrées

Pour calculer le résultat de **S = A XOR B XOR C** , il faut d'abord faire l'opération entre deux termes, puis refaire un ou exclusif entre le résultat obtenu et le troisième terme.

Ce qui se traduit par **S = (A XOR B) XOR C** ou par **S = A XOR (B XOR C)**

La fonction XOR (OU Exclusif)

On constate que l'appellation "Ou exclusif" n'est tout à fait exacte que pour deux variables. Avec trois variables, le résultat vaut 1 si une d'entre elles ou toutes les trois valent 1.



Le résultat est en fin de compte un bit de parité. Il vaut 1 si le nombre d'entrées à 1 est impair.

Table de vérité avec 3 entrées (C1, C2 et C3):

C1 (a=b)	C2(b=c)	C3(a=c)	C1 ⊕ C2 ⊕ C3
FAUX	FAUX	FAUX	FAUX
FAUX	FAUX	VRAI	VRAI
FAUX	VRAI	FAUX	VRAI
FAUX	VRAI	VRAI	FAUX
VRAI	FAUX	FAUX	VRAI
VRAI	FAUX	VRAI	FAUX
VRAI	VRAI	FAUX	FAUX
VRAI	VRAI	VRAI	VRAI

Exemple précédent:

- Deux valeurs et deux seulement sont identiques parmi **a, b et c**

((a = b) ET (a ≠ c)) OU ((a = c) ET (a ≠ b)) OU ((b = c) ET (a ≠ b))

- Parmi les valeurs de a, b et c deux valeurs au moins sont identiques :

(a=b) XOR (a=c) XOR (b=c)

➔ Si toute les valeurs de a, b et c sont identiques alors le résultat est VRAI

4. Instructions conditionnelles et alternatives

4. 1. Instruction : Si … Alors … Sinon … Finsi

Exemple 4 :

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).*

```
Algorithme signe_produit;
        Var
            n,m: réel;
        Début
          Ecrire("donner deux nombres n et m");
          Lire(n,m);
          Si ((n>0 ET m>0) OU (n<0 ET m<0)) Alors
                Ecrire(" le produit des deux nombres est positif ");
          Sinon
                Ecrire(" le produit des deux nombres est  négatif ");
          Finsi
        Fin
```

78

4. Instructions conditionnelles et alternatives

4. 1. Instruction : Si … Alors … Sinon … Finsi

Exemple 5

Ecrire un algorithme qui demande un nombre entier puis teste et affiche s'il est **divisible par 3 ou non.**

```
Algorithme divisible_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
```

79

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle imbriquée (Trois choix)

Syntaxe

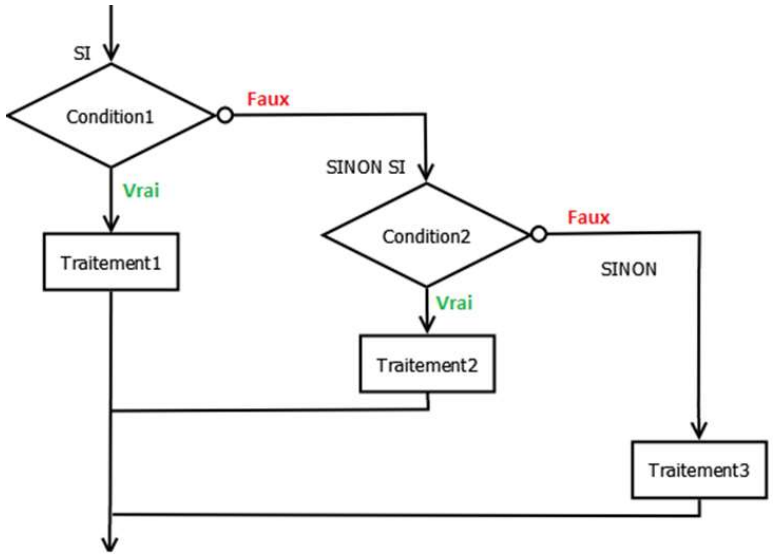
```
Si  (condition1)   alors
        Traitement 1;
  Sinon Si (condition2) alors
            Traitement 2;
        Sinon
                Traitement 3;
        Finsi
Finsi
```

- Si la ***condition1*** est **vraie**, alors le ***<Traitement 1>*** est **exécuté**.
- Sinon (la ***condition1*** est **fausse)**, alors on teste la ***condition2***,
- Si la **condition2** est **vraie**, alors on **exécute** le ***<Traitement 2>***,
- Sinon (la ***condition2*** est **fausse)**, on **exécute** le ***<Traitement 3>***

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle imbriquée (Trois choix)

Organigramme



4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle imbriquée (Multi-choix)

Dans de nombreuse applications, on peut avoir **plusieurs cas d'exécution selon différentes conditions**. Ainsi, il faut utiliser **plusieurs conditions** les uns à la suite des autres pour **englober** tous ces cas.

```
Si (condition1) alors
        Traitement 1;
  Sinon Si (condition2) alors
              Traitement 2;
        Sinon
              Traitement 3;
        Finsi
Finsi
```

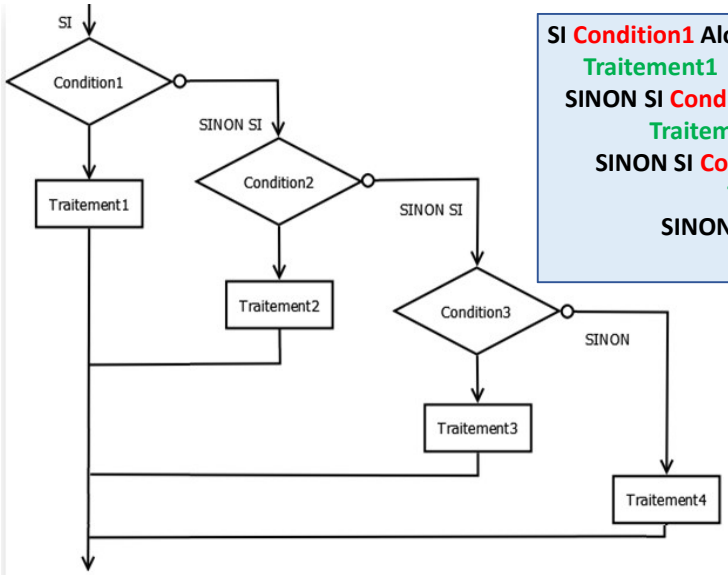
Lorsqu'on fait une condition dans une autre condition. Il faut respecter les règles suivantes:

- On peut avoir le **nombre d'imbrication qu'on veut.**
- L'**ordre des imbrications est important**

4. Instructions conditionnelles et alternatives

4. 1. Structure conditionnelle imbriquée (Multi choix)

Donner l'algorithme qui correspond à cet organigramme :




```
SI Condition1 Alors
    Traitement1
 SINON SI Condition2 Alors
        Traitement2
    SINON SI Condition3  Alors
            Traitement3
        SINON
            Traitement4
```

4. 1. Structure conditionnelle imbriquée (multiple choix)

Exemple 1: Ecrire un algorithme qui permet d'afficher le signe d'un nombre saisie par l'utilisateur (nombre **positif**, nombre **négatif** ou bien **nul**)

```
Algorithme exemple1;
Var
    n: réel;
Début
 Ecrire("Entrez un nombre réel:");
 Lire(n);
  Si  (n<0)  alors
     Ecrire(n, " est négatif");
  Sinon
      Si (n=0) alors
         Ecrire(n, " est nul ");
      Sinon
          Ecrire(n, " est positif");
      Finsi
   Finsi
Fin
```

```
Algorithme exemple1a;
Var
    n: réel;
Début
 Ecrire("Entrez un nombre réel:");
 Lire(n);
  Si  (n<0)  alors
     Ecrire(n, " est négatif");
  Finsi
  Si (n=0) alors
     Ecrire(n, " est nul ");
  Finsi
  Si (n>0) alors
     Ecrire(n, " est positif");
  Finsi
Fin
```

Dans **exemple1**, on fait **systématiquement trois tests** alors que dans **exemple1a,** si par exemple, le **nombre est négatif** on ne fait qu'**un seul test**.

84

4. 1. Structure conditionnelle imbriquée (multiple choix)

Exemple 2: Ecrire un algorithme qui permet d'afficher l'état de l'eau selon sa température. Trois réponses possibles (**Solide**, **liquide** ou **gazeuse**).

```
Algorithme exemple2;
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
```

```
Algorithme exemple2a;
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
```

Dans **exemple2a**, on fait **systématiquement trois tests** alors que dans **exemple2,** si par exemple, la température est inférieur à 0 on ne fait qu'**un seul test**.

4. 1. Structure conditionnelle imbriquée (multiple choix)

Exemple 3: Ecrire un algorithme qui permet d'afficher le signe de deux nombres (**positif** ou **négatif**)

Imbrication de Si

```
Si (a < 0) alors
        Si (b < 0) alors
            Ecrire( "a et b sont négatifs "  );
        Sinon
            Ecrire("a est négatif, b est positif ou bien nul ");
        Finsi
Sinon
        Si (b < 0) alors
            Ecrire( "b est négatif, a est positif  ou bien nul");
        Sinon
            Ecrire( "a est positif ou nul,  b est positif ou nul "  );
        Finsi
Finsi
```

Remarques: Si par exemple **a et b sont tous les deux positifs**, alors aucun des deux tests ne sera vraie, et c'est donc le **sinon** du **sinon** qui sera exécuté, et l'algorithme affichera le message " **a est positif ou nul, b est positif ou nul "**

Instructions conditionnelles et alternatives

Imbrication de Si

Dans de nombreuse applications, on veut avoir plusieurs cas d'exécution selon différentes conditions. Il faut exprimer plusieurs " Si " les uns à la suite des autres pour englober tous les cas.

Exercice 1:

Ecrire un algorithme permettant de résoudre une équation du premier degré **ax+b=0.**

Exercice 2:

Ecrire un algorithme permettant de résoudre une équation du seconde degré **ax ^{2}+bx+c=0 dans R (Voir TD N°2).**

87

4. Instructions conditionnelles et alternatives

```
Algorithme equationPremierDegre;
Var   a,b, x: réel;
Début
   Ecrire( ''Donner deux nombres a et b'');
   Lire (a, b);
      Si  (a=0)  alors
            Si (b=0) alors
                  Ecrire( ''Tous réel est solution de l'équation'');
            Sinon
                  Ecrire( ''Impossible'');
            Finsi
      Sinon
                  x ⟵ - b/a;
                  Ecrire (''La solution de l'équation est x='', x);
      Finsi
Fin
```

88

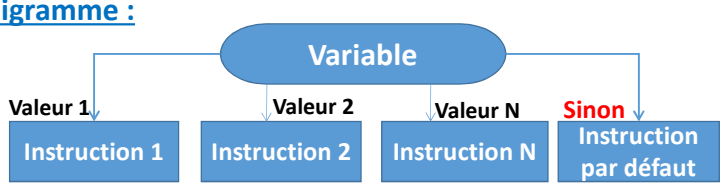
4. Instructions conditionnelles et alternatives

4. 2 Instructions alternative: la primitive Selon

Syntaxe :

```
Selon <variable> faire
    liste de valeur1: <séquence d'instruction1>;
    liste de valeur2: <séquence d'instruction2>;
    liste de valeur3: <séquence d'instruction3>;
    ……
   liste de valeurN: <séquence d'instructionN>;
Sinon <instruction_par_défaut>;
Finselon
```

Organigramme :



89

4. Instructions conditionnelles et alternatives

4. 2 Instructions alternative: la primitive Selon

Remarques importantes :

- **L'instruction** qui sera exécutée est celle qui correspondra à la valeur **courante** de la variable **testée**
- Lorsque cette **exécution est achevée**, on va **ignorer** toutes les séquences des **instructions suivantes**.
- Si **deux valeurs différentes** peuvent entrainer un même **traitement**. Nous pouvons alors les **regrouper séparées** par des **virgules** avant de décrire le traitement.
- Si la **variable** n'est égale à **aucune des valeurs testées**, c'est l'instruction **par défaut** qui sera exécutée.
- la **variable** de sélection doit être **déclarée**, et ne doit avoir qu'un **type discret** (**entier** ou **caractère** mais pas un **réel**).

90

4. Instructions conditionnelles et alternatives

4. 2 Instructions alternative: la primitive Selon

Exemple 1 :

- Ecrire un **algorithme** qui permet de **lire un numéro** compris entre **1 et 12** et d'afficher le **nom du mois** correspondant.
- Si le **numéro** entré est **en dehors** de cet intervalle, un **message d'erreur** doit être affiché.

91

4. Instructions conditionnelles et alternatives

4. 2 Instructions alternative: la primitive Selon

Exemple1 (version 1)

```
Algorithme mois;
Var
  n : Entier;
Début
   Ecrire(”Entrer le numéro du mois : ”);
   Lire(n);
   Selon (n) faire
     1 : Ecrire(”Janvier”);
     2 : Ecrire(”Février”);
     …
    12 : Ecrire(”Décembre”);
    Sinon
         Ecrire(”numéro de mois erroné … ”);
   FinSelon
Fin
```

4. Instructions conditionnelles et alternatives

4. 2 Instructions alternative: la primitive Selon

Version 2 *avec plusieurs «Si» imbriquées*

```
Algorithme mois;
  Var
    n : Entier;
  Début
     Ecrire(”Entrer le numéro du mois : ”);
     Lire(n);
     Si (n=1) alors
        Ecrire(”Janvier”);
     Sinon
          Si(n=2) alors
               Ecrire(”Février”);
          Sinon
               Si(n=3) alors
                  Ecrire(”Mars”);
                    …
Fin
```

Remarque :
L'exemple précédent peut être résolu en utilisant plusieurs instructions « Si » imbriquées, mais l'algorithme sera très lourd

93

Exemple2 (version 1)
Ecrire un algorithme demandant a l'utilisateur le jour de la semaine et affiche ensuite le jour du lendemain.

```
Algorithme  lendemain;
Var   Erreur: entier;
       jour, lendemain: chaine de caractère;
Début
       Ecrire(‘‘ saisir un jour de la semaine’’);
       Lire(jour);
       Erreur←0;
       Selon   (jour)  faire
            ‘‘ Lundi’’: lendemain← ‘‘ Mardi’’;
            ‘‘ Mardi’’: lendemain← ‘‘ Mercredi’’;
            ‘‘ Mercredi’’: lendemain← ‘‘ Jeudi’’;
            ‘‘ Jeudi’’: lendemain← ‘‘ Vendredi’’;
            ‘‘ Vendredi’’: lendemain← ‘‘ Samedi’’;
            ‘‘ Samedi’’: lendemain← ‘‘ Dimanche’’;
            ‘‘ dimanche’’: lendemain← ‘‘ Lundi’’;
            Sinon
                        Erreur←1;
       FinSelon
       Si (erreur=1)  alors
         Ecrire ("erreur de saisie");
       Sinon
        Ecrire("Le lendemain du ", jour, " est ", lendemain, ".")
       Finsi
Fin
```

Version 2 : On considère une variable booleen.

```
Algorithme lendemain;
Var   Erreur: booléen;
      jour, lendemain: chaine de caractère;
Début
      Ecrire(‘’ saisir un jour de la semaine’’);
      Lire(jour);
      Erreur←Faux;
      selon  jour  faire
          ‘’ Lundi’’: lendemain← ‘’ Mardi’’;
          ‘’ Mardi’’: lendemain← ‘’ Mercredi’’;
          ‘’ Mercredi’’: lendemain← ‘’ Jeudi’’;
          ‘’ Jeudi’’: lendemain← ‘’ Vendredi’’;
          ‘’ Vendredi’’: lendemain← ‘’ Samedi’’;
          ‘’ Samedi’’: lendemain← ‘’ Dimanche’’;
          ‘’ dimanche’’: lendemain← ‘’ lundi’’;
          Sinon
                    Erreur←Vrai;
     FinSelon
     Si (Erreur) Alors
          Ecrire ("erreur de saisie");
     Sinon
          Ecrire("Le lendemain du ", jour, " est ", lendemain, ".");
     Finsi
Fin
```

REMARQUE:
Il y a une **grande différence** entre **cet algorithme** et celui utilisant les **imbrications de Si**. Ce dernier semble **plus claire et lisible** et il n’y a pas de risque d’erreurs

Exemple2 (version 3)
Algorithme demandant a l'utilisateur le jour de la semaine et affiche ensuite le jour du lendemain.

Version 3 : On considère une seule variable de type chaine de caractère au lieu de trois variables.

```
Algorithme  lendemain;
Var
      jour: chaine de caractère;
Début
      Ecrire(‘’ saisir un jour de la semaine’’);
      Lire(jour);
      Selon  (jour)  faire
          ‘’ Lundi’’: Ecrire("Le lendemain du ", jour, "est Mardi.");
          ‘’ Mardi’’: Ecrire("Le lendemain du ", jour, "est Mercredi.");
          ‘’ Mercredi’’: Ecrire("Le lendemain du ", jour, "est Jeudi.");
          ‘’ Jeudi’’: Ecrire("Le lendemain du ", jour, "est Vendredi.");
          ‘’ Vendredi’’: Ecrire("Le lendemain du ", jour, "est Samedi.");
          ‘’ Samedi’’: Ecrire("Le lendemain du ", jour, "est Dimanche.");
          ‘’ Dimanche’’: Ecrire("Le lendemain du ", jour, "est Lundi.");
          Sinon
                    Ecrire("Erreur de saisie.");
      FinSelon
Fin
```