Cours d'Informatique
Initiation à l'informatique, à l'algorithmique et à la programmation
Licence fondamentale et professionnelle

Pr. Rachid SEHAQUI
Université Hassan II Casablanca
Faculté des sciences Aïn Chck
r.sehaqui@fsac.ac.ma

ALGORITHMIQUE POUR L'INGENIEUR

Pr. Rachid SEHAQUI
Université Hassan II Casablanca
Faculté des sciences Aïn Chck
r.sehaqui@fsac.ac.ma

Objectif

- Familiariser les étudiants avec les techniques et outils permettant
  - de concevoir et de comprendre,
  - de réaliser puis traduire,
  - d'implémenter et enfin d'obtenir un programme produisant par son exécution sur un ordinateur, le résultat attendu.
- $\Rightarrow$ rigueur de la démarche scientifique

Chapitre 1 : Introduction à l'algorithmique

1.1 - Introduction

Matériel / Logiciel

- Domaine du **logiciel** (*software*)
  - instructions expliquant à l'ordinateur comment traiter un problème
  - algorithmes et représentation informatiques de ces instructions
  - **programme**

1.1 - Introduction

- Les programmeurs sont amenés à concevoir des logiciels de plus en plus complexes pour répondre aux besoins des utilisateurs (applications en réseau, base de données réparties, …)
- $\Rightarrow$maîtriser le développement des logiciels est très important

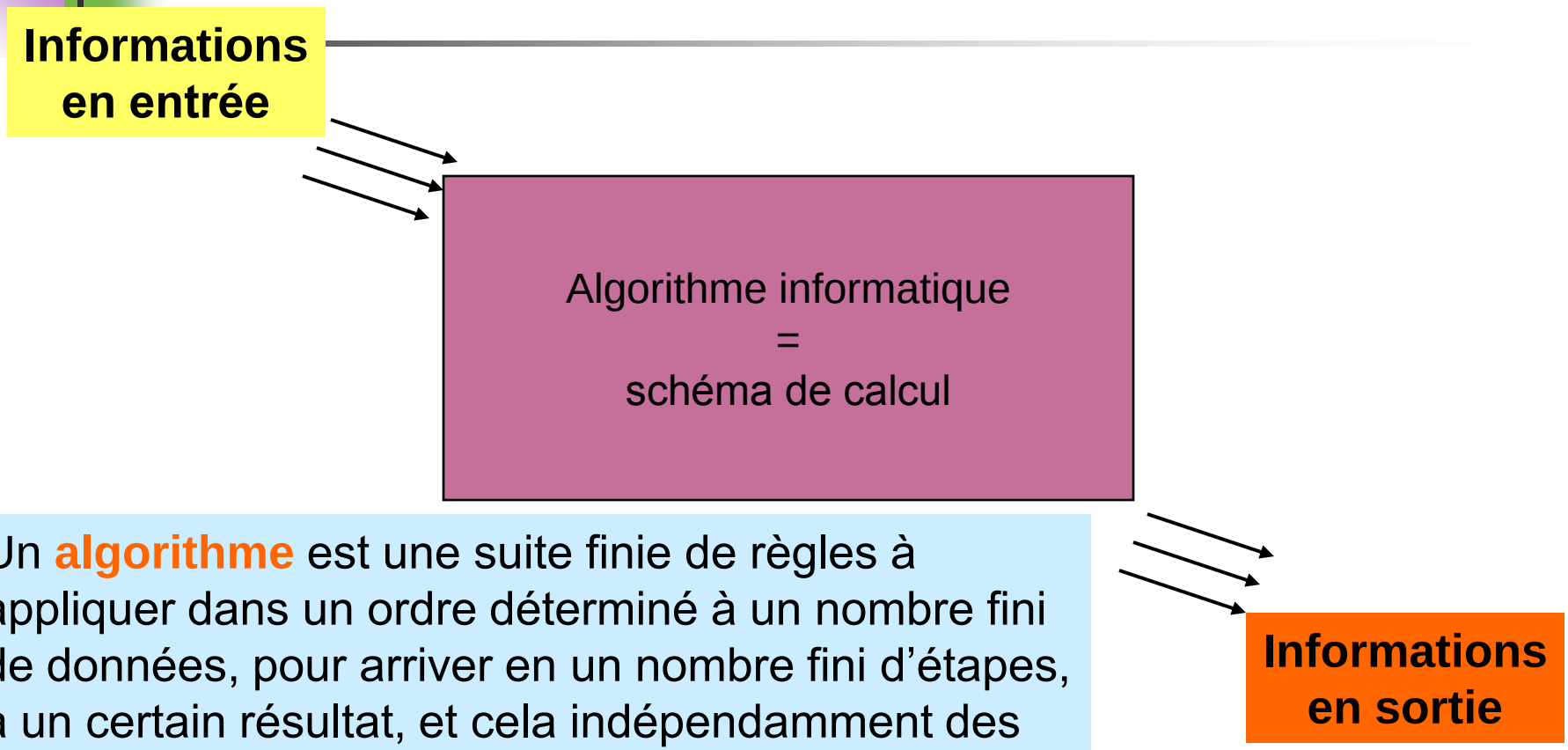
1.2 - Algorithmes et programmes

- Notion précise d'algorithme a été découverte en 825 par le mathématicien arabe Muhammmad ibn Musa al-Kharezmi
  - Moyen d'automatisation et d'économie de la pensée
- Petit Larousse : suite d'opérations élémentaires constituant un schéma de calcul ou de résolution de problème.

1.2 - Algorithmes et programmes

- Pour nous :
  - Un algorithme est une séquence précise et non ambiguë d'une suite d'étapes pouvant être exécutées de façon automatique par un ordinateur pour résoudre un problème
  - Spécification du schéma de calcul sous forme d'une suite finie d'opérations élémentaires obéissant à un enchaînement déterminé.

1.2 - Algorithmes et programmes



Un **algorithme** est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données, pour arriver en un nombre fini d'étapes, à un certain résultat, et cela indépendamment des données.

1.2 - Algorithmes et programmes

- Programme :
  - codage d'un algorithme afin que l'ordinateur puisse exécuter les actions décrites
  - doit être écrit dans un langage compréhensible par l'ordinateur
    - → langage de programmation
- Un programme est donc une suite ordonnée d'instructions élémentaires codifiées dans un langage de programmation

1.3 - Langages de programmation

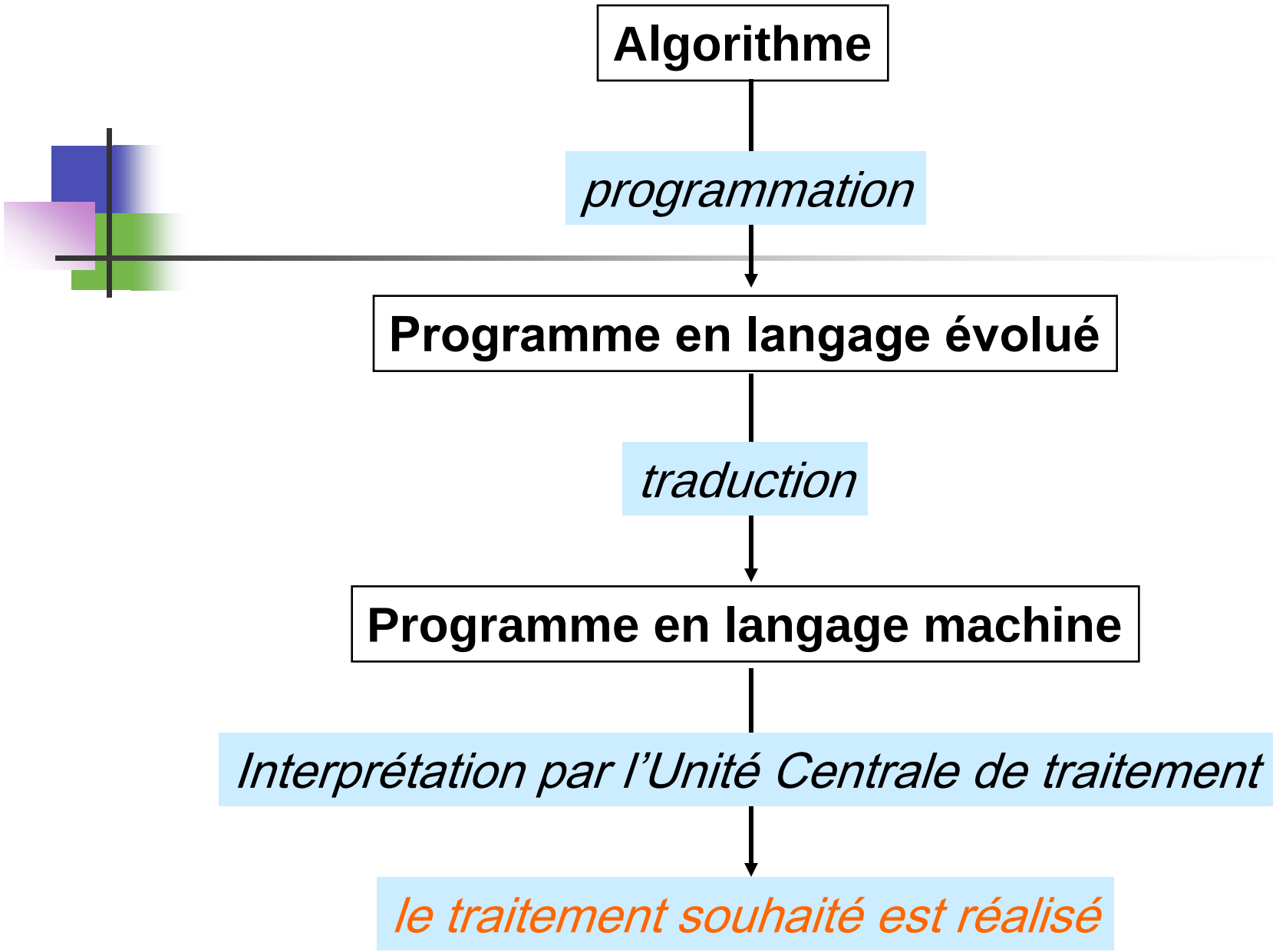
- RAPPEL: Langage machine
  - langage binaire
  - ses opérations sont directement compréhensibles par l'ordinateur
  - propre à chaque famille d'ordinateur

- Ecriture des premiers programme en langage machine

1.4 - Importance des algorithmes

Pour mener à bien un traitement sur un ordinateur il faut :

1. Concevoir un **algorithme** qui décrit comment le traitement doit être fait
2. Exprimer l'algorithme sous la forme d'un **programme** dans un langage de programmation adéquat
3. Faire en sorte que l'ordinateur exécute le programme : **compilation**



1.4 - Importance des algorithmes

Quels sont les aspects et propriétés des algorithmes qu'il est nécessaire d'étudier ?

1. La calculabilité des algorithmes (convergence de l'algorithme et existence de l'algorithme) la méthode existe t elle ?
2. La complexité des algorithmes (nombre d'opérations nécessaires)
3. L'efficacité des algorithmes (vitesse des algorithmes: raisonnable) TEMPS D'EXÉCUTION - MÉMOIRE OCCUPÉE

1.5 – Affinement des algorithmes

- Un algorithme doit décrire précisément le traitement qu'il doit exécuter et s'assurer que **tous les cas de figures possible** ont bien été prévus.
- Exemple : algorithme permettant de calculer la durée d'un voyage à partir du tableau

1. Consulter l'heure de départ
2. Consulter l'heure d'arrivé
3. Soustraire l'heure de départ de celle d'arrivée

1.5 – Affinement des algorithmes

- Problèmes :
  - fuseaux horaires différents
  - Si un point applique l'heure d'Été et pas l'autre
- Pour éviter de telles erreurs le concepteur doit suivre une démarche rigoureuse et méthodique :
  - Affinement progressive de l'algorithme
  - Démarche descendant, top down
  - Technique du « diviser pour mieux régner »

1.5 – Affinement des algorithmes

- Exemple : robot domestique avec un algorithme de préparation d'une tasse de café soluble

1. Faire bouillir l'eau
2. Mettre le café
3. Ajouter l'eau dans les tasses

1.5 – Affinement des algorithmes

- 1. faire bouillir l'eau

peut être affinée en

  - 1.1. remplir la bouilloire d'eau
  - 1.2. brancher la bouilloire sur le secteur
  - 1.3. attendre l'ébullition
  - 1.4. débrancher la bouilloire

1.5 – Affinement des algorithmes

- 2. mettre le café dans la tasse

pourrait être affiné en

  - 2.1. ouvrir le pot à café
  - 2.2. prendre une cuiller à café
  - 2.3. plonger la cuiller dans le pot
  - 2.4. verser le contenu de la cuiller dans la tasse
  - 2.5. fermer le pot à café

1.5 – Affinement des algorithmes

- 3. ajouter de l'eau dans la tasse

pourrait être affinée en

3.1. verser de l'eau dans la tasse jusqu'à ce que celle-ci soit pleine

1.5 – Affinement des algorithmes

- 1.1. remplir la bouilloire d'eau

- peut nécessiter les affinements suivants:

- 1.1.1. mettre la bouilloire sous le robinet
- 1.1.2. ouvrir le robinet
- 1.1.3. attendre que la bouilloire soit pleine
- 1.1.4. fermer le robinet

1.5 – Affinement des algorithmes

- Affinement ne se fait pas dans le vide
- Savoir où s'arrêter
- **Connaître les capacités du processeur**
- Exemple :
  - Brancher la bouilloire, activité interprétable
  - Remplir la bouilloire, activité non interprétable $\Rightarrow$ affinement

1.6 – Ecrire un algorithme

- Processeur informatique : ordinateur, capacités d'interprétation connues
- **Le concepteur d'un algorithme doit donc affiner ce dernier jusqu'à ce que les étapes puissent être écrites à l'aide d'un langage de programmation**

1.7 – Exemple

- Calcul de l'intérêt et de la valeur acquise par une somme placée pendant un an
- L'énoncé du problème indique
  - Les données fournies: deux nombres représentant les valeurs de la somme placée et du taux d'intérêt
  - les résultats désirés: deux nombres représentant l'intérêt fourni par la somme placée ainsi que la valeur obtenue après placement d'un an.

1.7 - Exemple

- Formalisation de l'algorithme: En français
  1. prendre connaissance de la somme initiale et du taux d'intérêt
  2. multiplier la somme par le taux; diviser ce produit par 100; le quotient obtenu est l'intérêt de la somme
  3. additionner ce montant et la somme initiale; cette somme est la valeur acquise
  4. afficher les valeurs de l'intérêt et de la valeur acquise.

1.7 - Exemple

- SI=somme initiale
- T=taux d'intérêt (ex: 3 pour 3%)
- I=intérets=S*T/100
- SF=somme finale=S+I

1.8 – Environnement de l'algorithme

Un algorithme utilise un certain nombre d'objets pour fonctionner appelé : **environnement de l'algorithme,** Cet environnement est constitué :

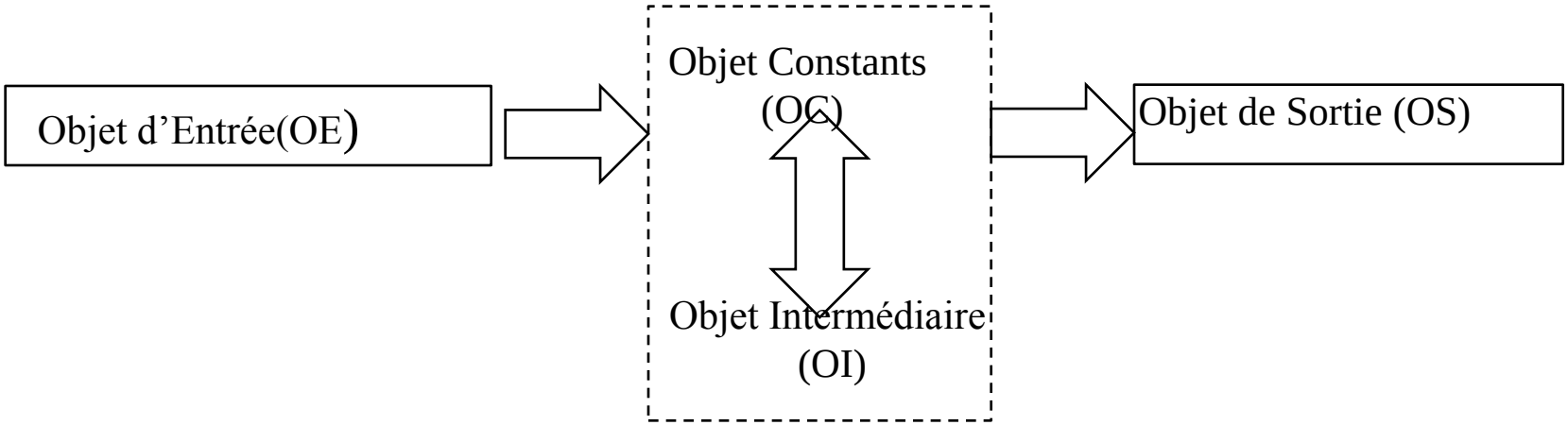
- **Des objets d'entrée (OE)** qui représentent l'ensemble de données que l'utilisateur doit introduire à l'algorithme.

- **- Des objets de sortie (OS)** qui représentent l'ensemble des résultats produits par l'algorithme.

- **- Des objets constants (OC)** qui représente l'ensemble des objets dont les valeurs ne changent pas au cours de toute l'exécution de l'algorithme.

1.8 – Environnement de l'algorithme

- **Des objets Intermédiaires (OI)** qui représentent l'ensemble des objets de traitement internes, ils ne sont ni entrés, ni sortis, ni constants mais ils sont soit des compteurs, soit calculés à partir des objets d'entrée et des objets constants et serviront pour produire les objets de sortie.



1. 9 Exemples

Exemples :

Exemple 1 Calcul de la quantité Q.

Exemple 2 Maximum de deux nombres;

Exemple 3 Calcul du périmètre d'un cercle;

Exemple 4 Solution d'une équation du second degré, $ax^2 + bx + c = 0$.

Exemple 5 Calcul de la moyenne d'un examen.

Dans une école un étudiant passe quatre matières à l'examen,

$1^{ère}$ matière écrite	: coefficient = 3
$2^{ème}$ matière écrite	: coefficient = 2
$1^{ère}$ matière orale	: coefficient = 4
$2^{ème}$ matière orale	: coefficient = 5

1. 10- Qu'est ce qu'un bon algorithme?

On peut noter qu'un bon algorithme est un schéma de résolution possédant les caractéristiques suivantes :

- **Correct** : il répond bien au problème posé;
- **Précis** : s'il fournit exactement les résultats attendus;
- **Efficace** : s'il utilise un temps d'exècution minimal indépendamment de la vitesse de la machine;
- **Clair et lisible** : s'il ne présente pas de difficulté de compréhension pour un autre programmeur désirant le maintenir ou le développer;
- **Résistant** : s'il est capable de détecter les cas de mauvaises utilisation.

Exemple : résistance d'un algorithme

Résolution d'une équation du premier degré $ax + b = 0$

Chapitre 2 : Outils de base de l'algorithmique

2. 1- Introduction

1. INTRODUCTION

- Vie quotidienne :
  - se préparer pour aller au travail le matin .
  - Conduire sa voiture correctement.
  - Préparer une recette de cuisine.
- Calculer le cout de production d'un produit donné.
- Calculer la moyenne d'examen pour une classe dans une école.
- Calcul scientifique : calcul d'une intégrale, solution d'une équation différentielle, résolution des systèmes d'équations.

2. 2- Notion d'objet

2. 2. NOTION D'OBJET

- Un algorithme est constitué d'un nombre fini d'objets appelé environnement et d'un ensemble fini d'actions permettant de traiter ces objets.

2 . 1 Identificateur d'objet

C'est un nom symbolique que nous attribuons à l'objet. Il est représenté par une suite quelconque de caractère et doit obéir aux exigences suivantes :

- Il doit commencer obligatoirement par une lettre ;
- La suite des caractère peut être composée soit de lettres non accentuées (a…z, A…Z) de chiffres (0…9), soit du caractère soulignement.

2. 2- Notion d'objet

Exemple : temps, calcul_vectoriel, mois1, a36b2 corrects.

1temps, calcul vectoriel, a36/b2 incorrects.

2. 2. 2 valeur d'un objet

- Constant : s'il ne peut pas être modifié par les instructions de l'algorithme tout au long de son exécution.
- Variable : dans le cas contraire.

2. 2. 3 Type d'un objet

Le type d'un objet constitue la nature de l'objet, c'est l'ensemble des valeurs qui peuvent être prises par cet objet,

2. 2- Notion d'objet

2. 2. 3. 1 Type Booléen

Type booléen correspond à un ensemble de deux états (vrai, faux),

Les opérations logiques : ET, OU, NON (table de vérité),

2. 2. 3. 2 Type Numérique

Le type numérique correspond à des intervalles de l'ensemble des entiers relatifs ($\mathbb{Z}$) et l'ensemble des nombres réels ($\mathbb{R}$).

L'ensemble des opérateurs les plus fréquents sur ces valeurs est composé des opérations arithmétique, elles sont résumé dans le tableau suivant:

2. 2- Notion d'objet

Opérateur	Signification	Exemple
+	Addition	R = a + b
-	Soustraction	R = a - b
*	Multiplication	R = a * b
/	Division	R = a / b si $b \neq 0$
^	Elévation à la puissance	y = x^n $\Leftrightarrow y = x^n$
DIV	Division entière	7 DIV 2 = 3
MOD	Reste d'une division	MOD(13,5) = 3
ENT	Partie entière	ENT(15,36) = 15

2. 2- Notion d'objet

2. 2. 3. 3 Type texte

Le type texte correspond à l'ensemble des chaines de caractères,

Une chaine de caractères est constituée des caractères de différents type qui peuvent être :

- Des des lettres de l'alphabet, majuscule ou minuscule (a,…z,A…Z),
- Des chiffres (0…9).
- Un espace.
- Des codes d'opérations (+, -, *, /…).
- Des caractères de ponctuation (. - , - ; - : - ! - …),
- Des caractères spéciaux (], $, @, ¥, €, {, ¥,…)

2. 2- Notion d'objet

2. 2. 3. 4 opérations de relations sur les objets

On peut appliquer entre deux objets de même type une opération de relation dont le résultat est une valeur booléenne : vrai ou faux. L'ensemble de ces opérations est décrit dans le tableau suivant :

Opération	**Symbole**
Strictement inférieur à	<
Strictement supérieur à	>
Inférieur ou égal à	≤ , <=
Supérieur ou égal à	≥ , ≥
Egal à	=
Différent de	<>

2. 2- Notion d'objet

Exemple

X et Y deux variables entières :

La valeur de x est 32, la valeur de Y est 25

- Les expressions suivantes donnent un résultat vrai:
  (Y < X), (X >= Y), (X <>Y).
- Les expressions suivantes donnent un résultat:
  (Y = X), (X < Y).

Par convention, pour les valeurs de type booléen :

Faux < Vrai

2. 3- Structure simplifiée d'un algorithme

2. 3 Structure simplifiée d'un algorithme

2. 3. 1 Structure générale

Exemple

```
CONST
        Année_cours = 2008;
        Taille_min= 1.65;
        Reponse=''oui ''
VAR
        Nom,prenom      : chaine de caractère
        Age             : entier;
        Etat_civil      : booléen
```

2. 3- Structure simplifiée d'un algorithme

2. 3. 2 Présentation d'un algorithme

```
Algorithme nom_algorithme
        (*Déclarations*)
Const
        Nom_de _const=valeur_de_const;
Var
        Nom_de _var: nom_type_var;
Début
        (*Corps de l'algorithme*)
        (*Actions*)
Fin
```

2. 3- Structure simplifiée d'un algorithme

Exemple 2 : Calcul du périmètre d'un cercle

```
Algorithme perimètre_cercle

Const c = 2;
        Pi=3.14;
Var
        P,R : real;
Début
        lire(R);
        P ← c*Pi*R;
        Ecrire(P);
Fin
```

2. 3- Structure simplifiée d'un algorithme

Remarques

- **Mots réservés**

Exemple : algorithme, const, var, début, fin, lire, écrire, si, alors, si , tantque , faire , répéter , jusqu'à , pour , procédure , fonction , tableau…

- **Commentaire**

Commentaires ne sont pas pris en compte par la machine lors de la compilation du programme.

(* Ceci est un exemple *)

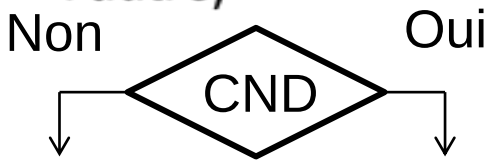
(*il n'est pas tenu en compte par la machine *)

- **Séquentialité**

2. 4- Organigramme

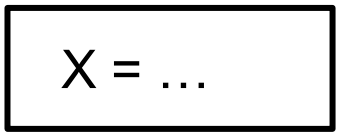
2. 4. ORGANIGRAMMES

↓ : **Séquence :** Marque le passage d'une action à l'autre;



: **Test de décision** : Marque un choix conditionnel d'une action à exécuter suivant

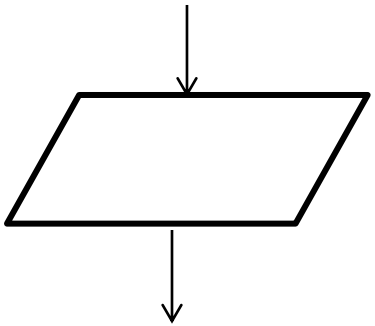
que la condition CND est vérifiée ou



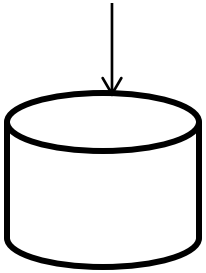
: **Opération : calcul**

Modifie une variable par l'affectation d'une nouvelle variable

2. 4- Organigramme



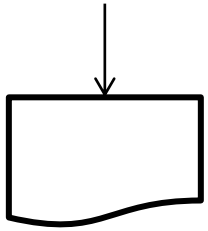
: **instruction d'entrée ou de sortie** : Entrée de données standard à partir du clavier ou sortie de données standard vers l'écran .



: **Opération : calcul**
Modifie une variable par l'affectation nouvelle variable.

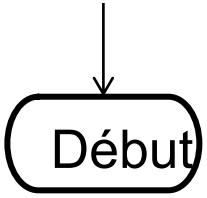
: **Sortie disque magnétique** sortie de données non standard dans un fichier, dans la disquette, ou le DD

2. 4- Organigramme



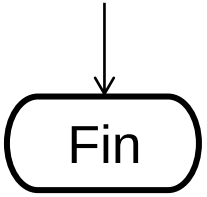
: **Sortie listing**

Sortie de données non standard: document en utilisant une imprimante sur le listing.



: **Début**

Démarrage d'un traitement.



: Fin

Arrêt d'un traitement.

Chapitre 3 : Les instructions élémentaires en algorithmique

Les instructions élémentaires en algorithmique

3. 1 INTRODUCTION

Les instructions élémentaires sont les instructions qui figurent le plus souvent dans tous les algorithmes, Elles sont au nombre de trois :

- L'affectation
- Les instructions d'entrée de données
- Les instructions d'entrée de données

3. 2 L'AFFECTATION

L'affectation permet d'assigner **une valeur à un objet**

3. 2. 1 Syntaxe

3. 2- L'affectation

3. 2. 1 Syntaxe

Identificateur_objet ⟵ Valeur

L'opération d'affectation présente certaines possibilités et impose certaines conditions.

3. 2. 2 Possibilités

- Une variable de même type que l' Identificateur_objet.

V ⟵ A

- Une constante de même type que l' Identificateur_objet.

V ⟵ 6

- Une expression dont le résultat final de même type que
- l' Identificateur_objet.

3. 2- L'affectation

V ⟵ 3*A + 2*B – 1

3. 2. 3 Conditions

8 ⟵ A; A + B ⟵ C :incorrectes

A ⟵ 0; V ⟵ 1/A

Var

A: entier;

C: Caractère;

Début

A 6

… ⟵

C ⟵ A

3. 3- LES INSTRUCTIONS D'ENTREE / SORTIE

- **Lire(V)** qui permet d'affecter à la variable (V), la valeur lue sur le périphérique d'entrée.
- **Ecrire(V1)** qui permet de transférer la valeur de (V1), vers le périphérique de sortie.

Remarques :

- L'identificateur (V) doit être une variable déclarée. Par conséquent les structures suivantes n'ont pas des sens.

Lire(6)

Lire(A+B)

- La valeur (V1) peut être :

3. 3- LES INSTRUCTIONS D'ENTREE / SORTIE

- Une variable déclarée : Ecrire(A);
- Une constante : Ecrire(7);
- Une expression : Ecrire(3*A + 2*B-1)

Exemple : Algorithme de calcul de la somme de eux nombres.

A éviter	Présentation conseillée
Algorithme Somme	Algorithme Somme
Var	Var
A,B,S: réels	A,B,S: réels
Début	Début

3. 3- LES INSTRUCTIONS D'ENTREE / SORTIE

A éviter	Présentation conseillée
Lire(A);	('Ecrire programme de calcul de la somme');
Lire(B);	('Entrée le premier nombre');
	Lire(A);
S ← A + B;	('Entrée le second nombre');
	Lire(B);
	S ← A + B;
Ecrire(S);	('Ecrire la somme des ces deux nombres est :' , S);
Fin	Fin

3. 4- EXERCICES D'APPLICATION

- 3. 4. 1. Exercice : Nombre formé

Ecrire un algorithme qui lit successivement 4 chiffres (allant de 0 à 9) au clavier et affiche à l'écran le nombre formé par ces 4 chiffres après l'avoir calculé et stocker en mémoire sous le nom Nb.

- 3. 4. 2. Exercice : Permutation circulaire de trois nombres

Ecrire un algorithme qui lit trois nombres dans trois variables, A, B, C, puis fait la permutation circulaire de ces trois nombres (sens trigonométrique) et affiche les nouveaux contenus des variables A, B et C.

- 3. 4. 3. Moyenne de l'examen

Dans une école un étudiant passe quatre matière à l'examen.

3. 4- EXERCICES D'APPLICATION

$1^{ère}$ matière écrite	: coefficient 3
$2^{ème}$ matière écrite	: coefficient 2
$3^{ème}$ matière orale	: coefficient 4
$4^{ème}$ matière orale	: coefficient 5

- 3. 4. 4. nombres complexes

Soient deux nombres complexes $(a_1 + ib_1)$ et $(a_2 + ib_2)$.

Ecrire un algorithme qui calcule leur produit et écrit leur rapport sous la forme algébrique.

3. 4. 5. nombres complexes

3. 4- EXERCICES D'APPLICATION

- 3. 4. 5. nombres complexes

Ecrire un algorithme qui calcule la surface d'un trapèze, ayant une grande base Gb et une petite base Pb :

$$S = \frac{(Gb + Pb) * h}{2}$$

- 3. 5. Exercices

Exercice 1

Ecrire un algorithme qui demande les coordonnées de deux points dans le plan, calcule et affiche à l'écran la distance entre ces deux points. On donne la fonction sqrt(x) qui renvoie la racine carrée.

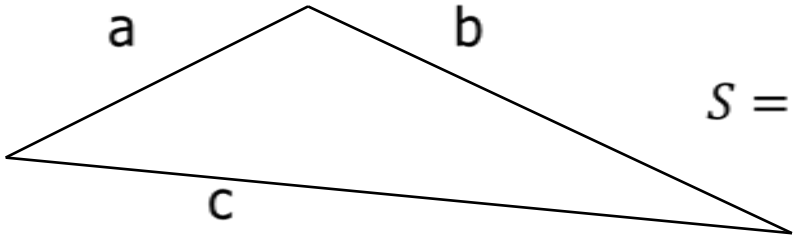
3. 5- EXERCICES

Exercice 2 : Calcul de la résistance équivalente.

Ecrire un algorithme permettant de demander les valeurs de trois résistances $R_1, R_2 et\ R_3$ et de calculer et d'afficher leur résistances équivalente dans les deux cas : série et parallèle.

Exercice 3 : Surface d'un triangle

Ecrire un algorithme qui permet de calculer la surface d'un triangle quelconque dont les côtés ont une longueur donnée a, b et c.



$$S = \sqrt{r(r-a)(r-b)(r-c},$$

$$r = \frac{a+b+c}{2}$$

3. 5- EXERCICES

Exercice 4 : Facture d'électricité

Ecrire un algorithme qui permet de calculer la facture d'électricité. On commence par calculer le puissance électrique consommée en Kwatt, puis on calcule les frais engendrés par cette puissance, sachant que pour 1 kwatt on paie 1,40DH.

Exemple de calcul :

Soit une installation contenant :

- Quatre lapes de 100 watts/h qu'on alimente pendant 10h chaque jour. Un téléviseur 200 watts/h qu'on alimente pendant 4h chaque jour. Un réfrigérateur de 800 watt/h qu'on alimente pendant 24h chaque jour. Un ordinateur de 400 watt/h qu'on alimente pendant 6h chaque jour.

Chapitre 4 : Les instructions élémentaires conditionnelles et alternatives

4. 1- Notions de primitives de base structurées

- 4. 1. Notions de primitives de base structurées

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 :

- Les primitives de choix qui permettent de choisir les instructions à exécuter selon les valeurs courantes de certaines variable; elles sont de deux types :
  - Les instructions conditionnelles
  - Les instructions alternatives
- Les primitives d'itération qui sont utilisées lorsqu'on souhaite exécuter plusieurs fois le même traitement.

4. 2- Instructions conditionnelles

- 4. 2 Instructions conditionnelles
  - 4. 2. 1. l'instruction Si…Alors…Finsi

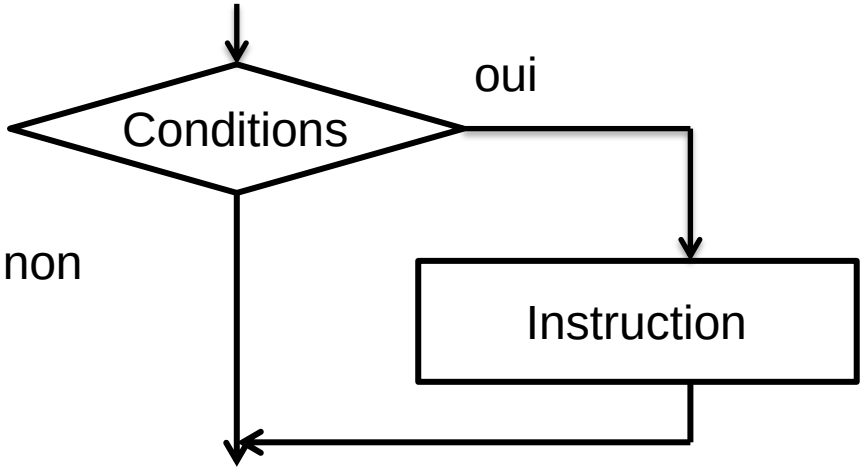
Syntaxe :

Si condition **alors**

Instruction (ou suite d'instructions)

Finsi

Organigramme:



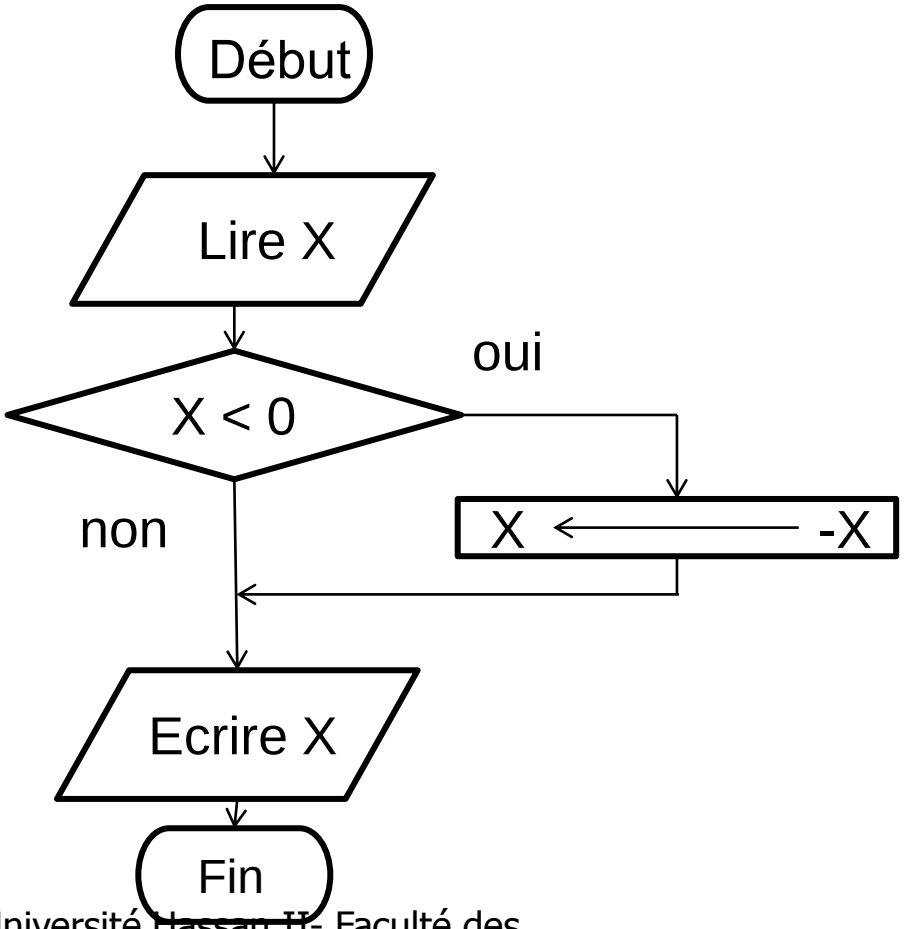
4. 2- Instructions conditionnelles

Exemple 1 : valeur absolue d'un nombre réel.

```
Algorithme valeur_absolue1;
Var
        X:réel;
Début
        Lire(X)
        Si
X<0  Alors
        X  ←————————  -X
        Finsi
        Ecrire ('la valeur absolue est :',X)
Fin
```

4. 2- Instructions conditionnelles

Organigramme



4. 2- Instructions conditionnelles

4. 2. 2. L'instruction Si…Alors…Finsi

Syntaxe

Si Condition **Alors**

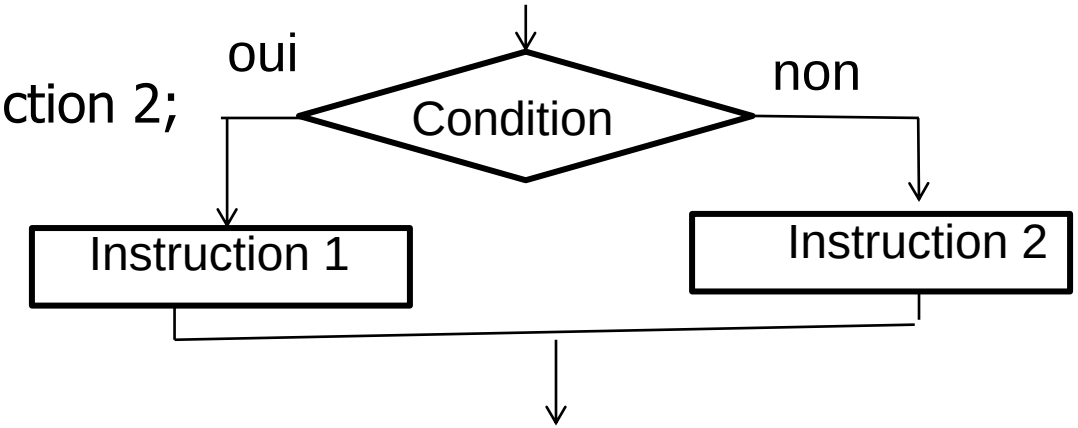
Instruction 1;

Sinon

instruction 2;

Finsi

Organigramme



4. 2- Instructions conditionnelles

Exemple2 :

Ecrire un algorithme qui permet d'afficher la valeur absolue de la différence de deux nombres réels saisis au clavier.

Analyse :

$$|X - Y| = X - Y \quad si\ X > Y$$
$$|X - Y| = -(X - Y) \quad si\ X < Y$$

```
Algorithme valeur_absolue2;
Var
        X,Y,Z:réel;
Début
        Lire(X);
        Lire(Y);
```

4. 2- Instructions conditionnelles

Si X > Y **Alors**

Z ⟵ X – Y

Sinon

Z ⟵ Y – X

Finsi

Ecrire ('la valeur absolue de la différence est :',Z)

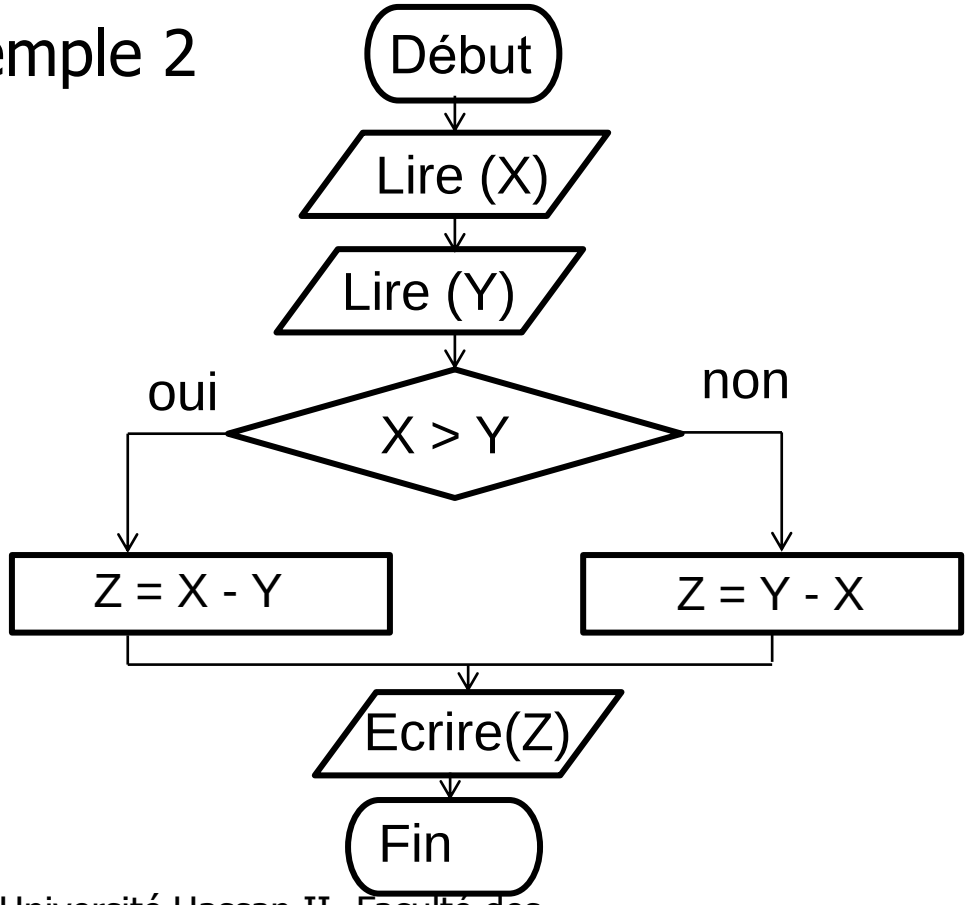
Fin

4. 2. 3. Imbrication de Si

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

4. 2- Instructions conditionnelles

Organigramme exemple 2



4. 2- Instructions conditionnelles

Exemple 3

- Un moniteur d'auto école veut apprendre à un candidat au permis de conduire ce qu'il faut faire dans un croisement avec feu, Il y'a trois possibilités :
- - si le feu est vert, le candidat peut passer;
- si le feu est orange, le candidat doit ralentir et se préparer pour s'arrêter.
- si le feu est rouge le candidat doit s'arrêter;

4. 3. La variante de l'alternative: la primitive ''cas"

```
Algorithme croisement_feu;
Var
        couleur_feu: chaine de caractère;
Début
        Ecrire('observer la couleur du feu:');
        Lire(couleur_feu);
        Si couleur_feu="rouge" Alors
                Ecrire('Arrêtez vous immédiatement');
        Sinon
                Si couleur_feu="orange" Alors
                        Ecrire('ralentissez et préparez vous pour
                        vous arrêter');
                Sinon
                Ecrire('passez');
                Finsi
        Finsi
Fin
```

4. 3- La variante de l'alternative: la primitive ''cas"

Exemple 4

On désire écrire un algorithme qui permet d'afficher le jour correspondant à un chiffre allant de 1 à 7, entré au clavier;

Avec la primitive **Cas,** nous allons résoudre complètement le problème.

Cas variable

variable_1 : instruction1;

……..;

variable_n : instruction n;

Sinon

Instruction_par_défaut

Fincas

```
Algorithme Affiche_jour
Var
        jour: entier;
Début
        Ecrire('entrer chiffre jour:');
        Lire(jour);
        Si jour=1 Alors
                Ecrire('Lundi');
        Sinon Si jour=2 Alors
                Ecrire('Mardi');
            Sinon Si jour=3 Alors
                Ecrire('Mercredi');
                Sinon Si jour=4 Alors
                Ecrire('Jeudi');
                    Sinon Si jour=5 Alors
                        Ecrire('Vendredi');
                        Sinon Si jour=6 Alors
                            Ecrire('Samedi');
                            Sinon Si jour=7 Alors
                                Ecrire('Dimanche');
                                Sinon
                                    Ecrire('ce n'est pas un jour de
                                     semaine')
                                Finsi
                            Finsi
                        Finsi
                    Finsi
                Finsi
            Finsi
        Finsi
Fin
```

4. 3- La variante de l'alternative: la primitive ''cas''

```
Algorithme Affiche_jour;
Var
        jour: entier;
Début
        Ecrire('entrer le chiffre jour:')
        Lire('jour')
        Cas jour
        1 : Ecrire('Lundi');
        2 : Ecrire('Mardi');
        3 : Ecrire(Mercredi');
        4 : Ecrire('Jeudi');
        5 : Ecrire('Vendredi');
        6 : Ecrire(Samedi');
        7 : Ecrire('Dimanche');
        Sinon
        Ecrire('ce n'est pas un jour de semaine:');
        Fincas
Fin
```

4. 4- Exercices d'application

Exercice 1 : Ordre de deux nombres

Ecrire un algorithme qui permet de saisir deux nombres entiers X, Y et les afficher à l'écran dans l'ordre croissant.

Exercice 2: Equation du second degré.

Ecrire un algorithme qui permet de résoudre une équation du second degré. Etudier tous les cas possibles.

$$ax^2 + bx + c = 0$$

Exercice 3 : Prix d'abonnement

Un libraire décide de faire des remises sur les prix d'abonnement à une revue scientifique selon le menu suivant:

4. 4- Exercices d'application

Ancien abonné	**: -15%**
Etudiant	**: -20%**
Nouvel abonné	**: -00%**
Etranger	**:+25%**

Le calcul du prix d'abonnement se fait en fonction du tarif normal d'abonnement (TN) de la qualité de l'abonné (Q). (Une seule qualité est accepté par abonné)

Ecrire un algorithme qui permettant de calculer le prix final à payer.

4. 4- Exercices d'application

Exercice 4 : Calcul sur les nombres

Ecrire un algorithme de résolution

On dispose de trois nombres réels saisis au clavier. Selon un choix effectué à partir d'un menu affiché à l'écran, on désire calculer la somme ou le produit ou la moyenne ou chercher le minimum ou chercher le maximum de ces trois nombres. Le menu doit se présenter à l'écran de la manière suivante :

……**MENU…**……

1…….Somme……..

2…….Produit……..

3……..Moyenne…..

4……..Minimum……

5…….MAXIMUM……

Chapitre 5 : Les instructions répétitives : les boucles

5. 1- Introduction

5. 1. introduction

Les primitives itératives constituent des boucles qui répètent l'exécution d'un même bloc d'instructions un certain nombre de fois. Ce nombre est soit précisé à l'avance soit il dépendra de l'évolution de l'action algorithmique effectuée par le bloc répétée,

Pour que la structure d'une boucle soit correcte, il faut qu'elle soit composée de quatre blocs:

- **Bloc d'initialisation de la boucle:** il sert comme point de départ des itérations.
- **Bloc de processus itératif ou corps de la boucle:** il contient toutes les instructions à répéter à chaque itération.

5. 2- Les instructions répétitives

- **Bloc de progression ou régression de l'indice** (compteur de la boucle): c'est un compteur qui fait évoluer la boucle.
- **Bloc test de continuation:** il effectue le contrôle pour décider de la continuation ou l'arrêt de la récurrence.

5. 2. Instructions répétitives

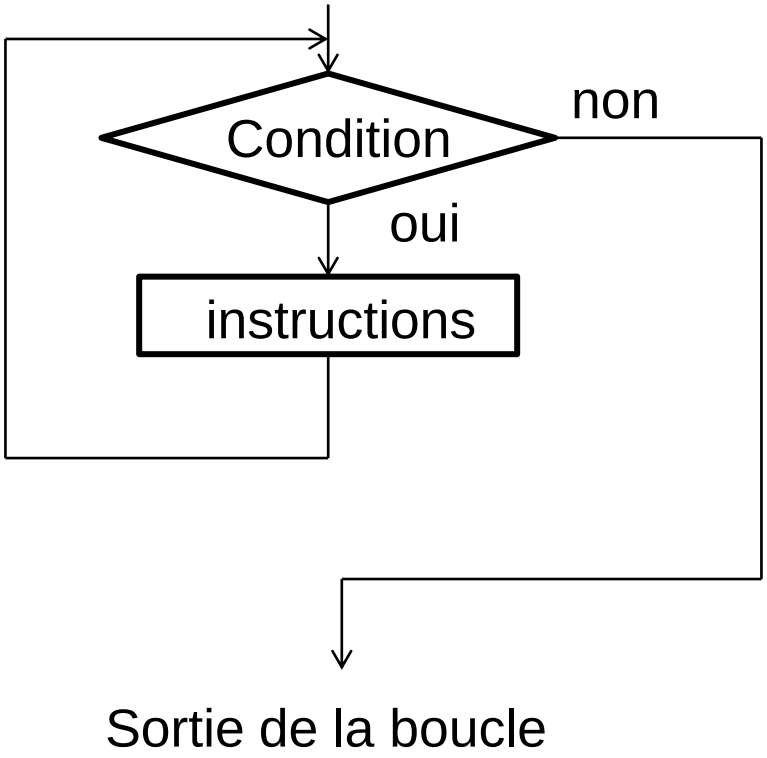
5. 2. 1. cas où le nombre d'itérations n'est pas connu à l'avance

- Le test de contrôle est effectué au début de la boucle,
- Le test de contrôle est effectué à la fin de la boucle.

5. 2. 1. 1. L'instruction **Tant que…faire…Fin tant que**

5. 2- Les instructions répétitives

organigramme



5. 2- Les instructions répétitives

Exemple 1

Ecrire un algorithme permettant de lire une suite de nombres réels sur le clavier, Le dernier élément à lire est un zéro. L'algorithme doit afficher la plus petit élément de la suite ainsi que la somme des éléments lus.

```
Algorithme  sommz_min;
Var
        Elt,S,min:réels
Début
        S  ←    0
        Lire(Elt); ←——— Bloc d'initialisation
        min ←  Elt;
```

5. 2- Les instructions répétitives

Tantque elt<> 0 **Faire;** ← Bloc test de continuation

S ← S + Elt

Si Elt < min **Alors**

min ← Elt;

Finsi

Lire(Elt) ← Bloc de progression de boucle

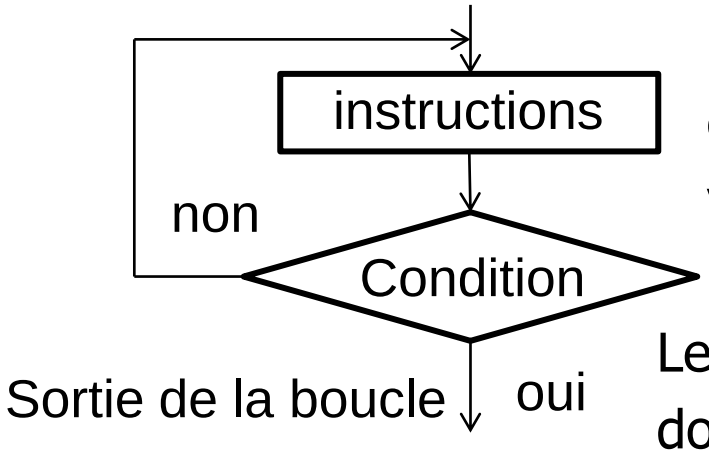
Bloc de processus itératif

Fintantque

Fin

5. 2- Les instructions répétitives

- 5. 2. 1. 2. l'instruction **Répéter…jusqu'à**



les instructions délimitées par répéter et jusqu'à ce que la condition soit vérifiée (valeur booléenne Vrai)

Le test est effectué à la fin de la boucle, donc les instructions sont exécutées au moins une fois même si la condition est est fausse dés l'entrée de la boucle.

Exemple 2

Ecrire un algorithme permettant de calculer la somme et la moyenne des N premiers nombres entiers

5. 2- Les instructions répétitives

```
Algorithme Somme_Moyenne;
Var
        N, S, i : entiers;
        Moy     : réel;
Début
        lire(N);
        S ⟵ 0;
        i ⟵ 0;
        Répéter
        i ⟵ i + 1;
        S ⟵ S + i
```

5. 2- Les instructions répétitives

Jusqu'à i = N

Moy ⟵ S/N;

Ecrire(S);

Ecrire(Moy);

Fin

5. 2. 2. Cas où le nombre d'itérations est connu à l'avance

Lorsque le nombre d'itérations est connu à l'avance, on utilise :

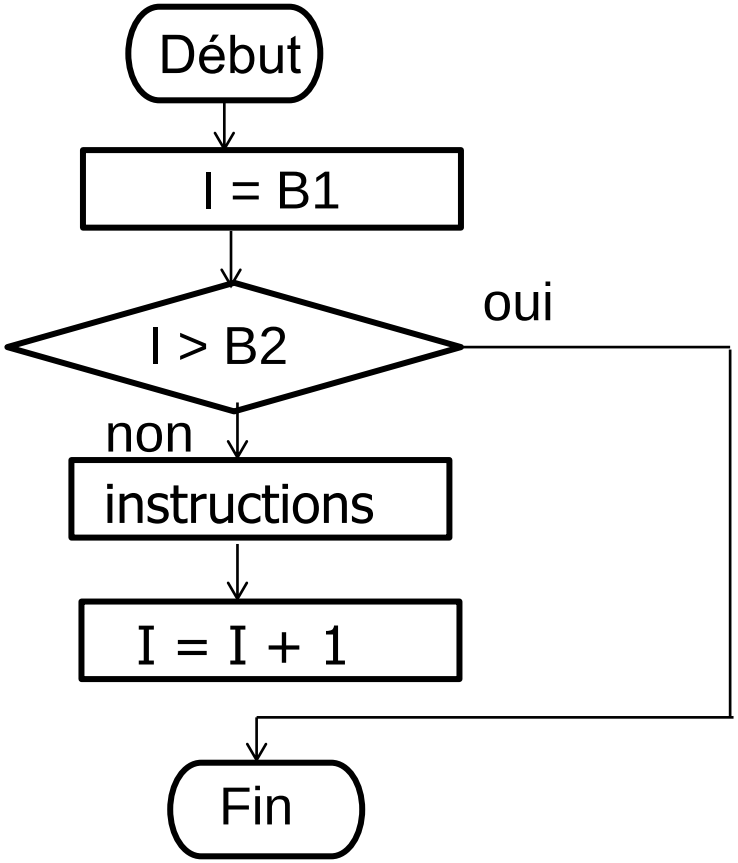
Pou…Faire…Finpour.

Pour variable de contrôle ⟵ borne 1 à borne 2 **Faire**

instructions;

FinPour

5. 2- Les instructions répétitives



5. 2- Les instructions répétitives

Exemple 4

Ecrire un algorithme permettant de calculer la factorielle d'un nombre entier N.

Algorithme

Var

N, i, Fact, : entiers;

Début

Lire(N);

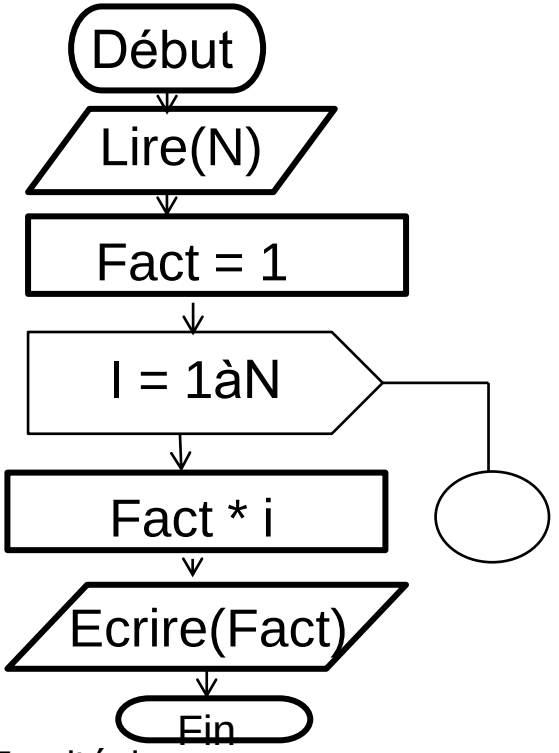
Fact ⟵1

<u>Pour</u> i⟵1 à N F**aire**

Fact ⟵ Fact * i

Finpour

Fin



5. 2- Les instructions répétitives

Exemple 5

Pour avoir une idée du niveau des élèves d'une classe , on a décidé de calculer la moyenne de la classe à partir des moyennes générales de tous les élèves qui sont au nombre de 30.

$$Moy = \frac{1}{30} \sum_{i=1}^{i=30} NT_i$$

Algorithme Moyenne;

Var

NT, SN , Moy : réels;

i : entier;

5. 2-. Les instructions répétitives

Début

SN ⟵ 0

Pour i ⟵ 1 à 30 **Faire**

Ecrire(‘entrer la moyenne de l’élève N°’ , i);

Lire(NT);

Sn ⟵ SN + NT;

Finpour i

Moy ⟵ SN/30;

Ecrire(‘ecrire la moyenne de la classe est:’,Moy);

Fin

5. 3- Exercices d'application

Exercice 1 : Maximum et minimum d'une liste

Ecrire un algorithme qui permet de retrouver le maximum, le minimum ainsi que la somme d'une liste de nombres positifs saisis par l'utilisateur. La fin de la liste est indiquée par un nombre négatif. La longueur de la liste n'est pas limitée.

Exemple : Si la liste des éléments est : 7 3 20 15 2 6 5 -1.

Le maximum est 20, le minimum est 2.

Exercice 2 : Ecrire un algorithme permettant de calculer:

$$\sum_{i=0}^{N} \frac{X^i}{i!}$$