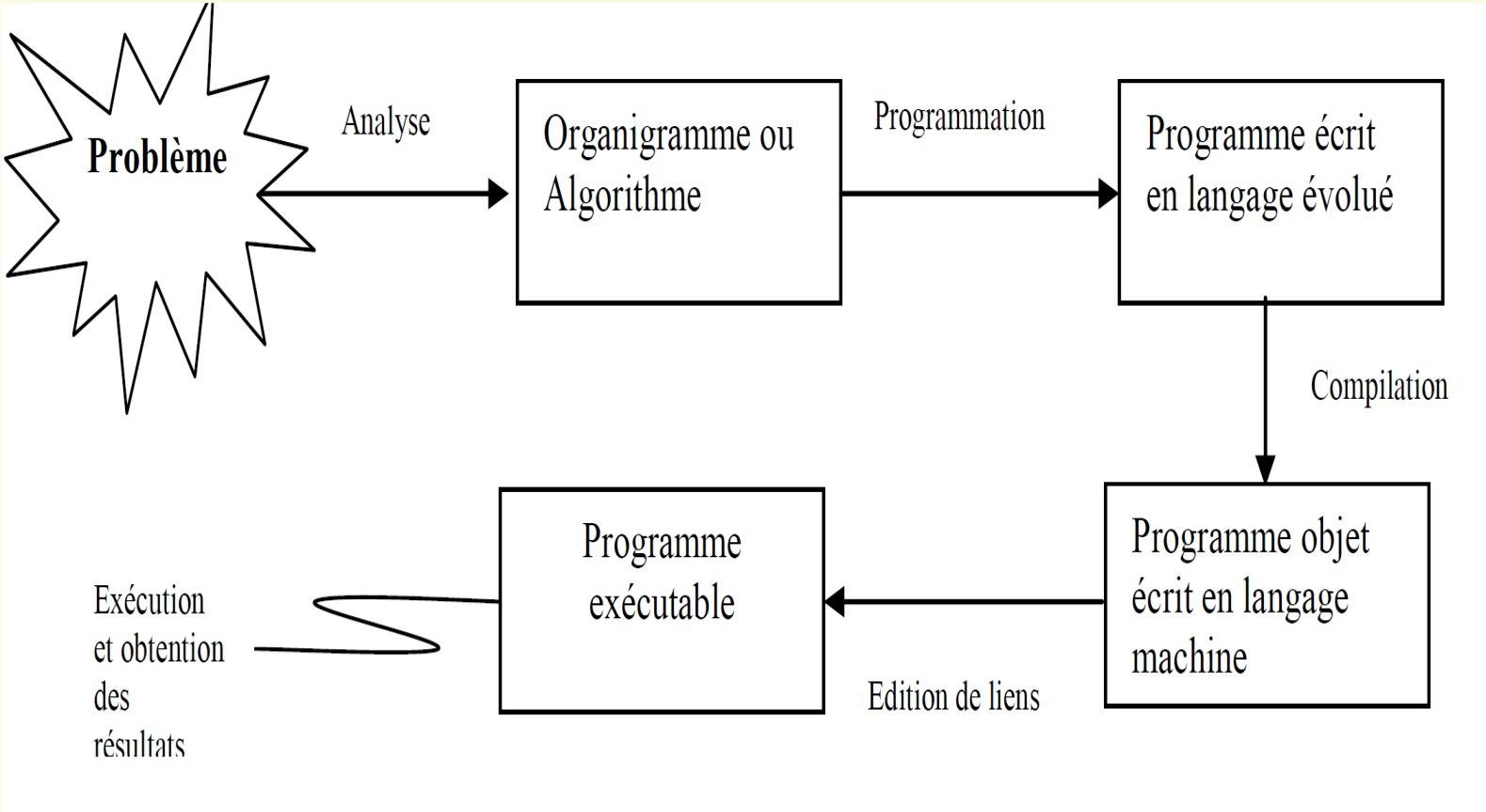
L’algorithmique

Mme M.LAKAB
Molakab@gmail.com

Notion d'Algorithmique

- L'informatisation d'un problème consiste à écrire le programme que doit exécuter l'ordinateur pour fournir les solutions souhaitées au problème posé.

L'information d'un problème!



La phase1 : Analyse du problème et écriture de l'algorithme

- L'analyse d'un problème consiste à étudier tous ses aspects et contraintes, et déterminer les étapes de sa résolution ainsi que les différents types et rôles des données sur lesquelles doivent être appliquées les différentes étapes. L'écriture formelle et unifiée de ces étapes peut être l'algorithme ou l'organigramme.

Exemple

- Problème => Résolution de l’équation

$$Ax2 + Bx + C = 0$$

L’analyse :

- Les données : A, B, C
- Les résultats souhaités: les racines x1, x2
- Les étapes que doit suivre l’odinateur:
- 1°) lire A, B, C
- 2°) Tester A
- 3°) Si A=0 tester B
- 4°) Si B=0 ambigüité car les coefficients de x dans l’équation sont nuls : pas de solutions.

- 5°) Si B≠0 écrire x=-C/B
- 6°) Si A≠0 calculer Δ= B2 -4AC
- 7°) Tester Δ
- 8°) Si Δ < 0 pas de solutions dans l'ensemble des réèls
- 9°) Si Δ = 0 solution double écrire x = -B/2A
- 10°) Si Δ >0 deux racines : écrire x1= (-B-racine(Δ)) /2A et x2= (-B+racine(Δ)) /2A.

Définitions : algorithme, algorithmique et organigramme

- **Un algorithme** est un énoncé dans un langage bien défini d'une suite d'opérations permettant de donner la réponse à un problème. Il représente les étapes ordonnées que doit exécuter un ordinateur pour fournir les résultats à un problème.

Définitions : algorithme, algorithmique et organigramme

- Le terme **algorithmique** désigne l'ensemble des activités logiques qui relèvent des algorithmes ; c'est-à-dire l'ensemble des règles et techniques qui sont impliquées dans la définition et la conception des algorithmes.
- **L'organigramme** est la représentation graphique de l'algorithme.

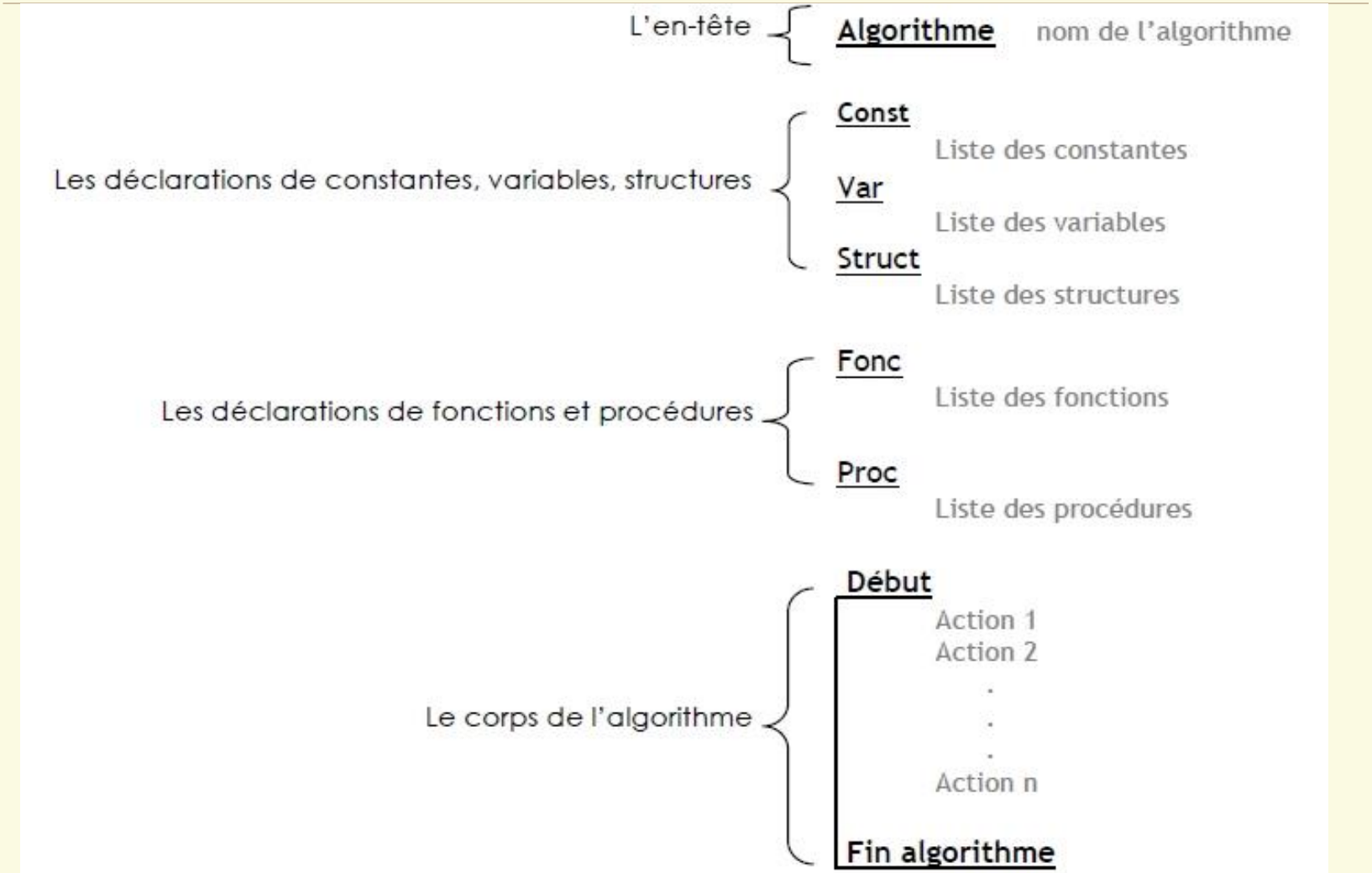
Caractéristiques d’un algorithme

- **lisible**: l'algorithme doit être compréhensible même par un non-informaticien
- **de haut niveau**: l'algorithme doit pouvoir être traduit en n'importe quel langage de programmation.

Caractéristiques d'un algorithme

- **précis**: chaque élément de l'algorithme ne doit pas porter à confusion, il est donc important de lever toute ambiguïté
- **concis**: un algorithme ne doit pas être très long .
- **structuré**: un algorithme doit être composé de différentes parties facilement identifiables

Structure d'un algorithme



Structure d'un algorithme

- L'en-tête : Il permet tout simplement d'identifier un algorithme.
- Les déclarations : C'est une liste exhaustive des objets utilisés et manipulés dans le corps de l'algorithme, cette liste est placée en début d'algorithme.
- Le corps : Dans cette partie de l'algorithme, sont placées les tâches (instructions, opérations…) à exécuter.

La déclaration des données

1. Les Constantes

- Une constante représente un chiffre, un nombre, un caractère ou une chaîne de caractère dont la valeur ne peut pas changer au cours de l'exécution du programme. Les constantes sont déclarées dans la partie constantes de l'algorithme.

La déclaration des données

2. Les variables

Une variable représente un chiffre, un nombre, un caractère ou une chaîne de caractère dont la valeur peut changer au cours de l'exécution de l'algorithme. Une variable est caractérisée par son nom et son type.

Les types d'une variable

- **entier** : qui couvre l'ensemble des valeurs entières : {…-4, -3, -2,-1, 0, 1, 2, 3, 4,…}
- **réel** : qui couvre l'intervalle des valeurs]-∞, +∞[
- **caractère** : qui couvre l'ensemble des caractères alphabétiques {A,…,Z, a,…,z}, numériques {0,…,9} ou caractères spéciaux {=,+,*,$, £,…} il est codé sur un octet.

Les types d'une variable

- **Chaîne de caractères** : permet de déclarer des variables dont la valeur peut être une suite de caractères comme un nom, un prénom, une adresse,…. Une chaîne de caractères composée de N caractères est codée sur N octets.
- **Booléen** : qui peut prendre deux valeurs : VRAI ou FAUX.

Les instructions de base sur des variables

- la saisie : on demande à l'utilisateur de l'algorithme de donner une valeur à la variable
- l'affectation : le concepteur de l'algorithme donne une valeur à la variable. Cette valeur peut-être le résultat d'un calcul ;
- l'affichage : on affiche la valeur de la variable.

Lire et écrire

Soit le programme suivant :

```
Variable A : entière
Début
    A ← 12^2
Fin
```

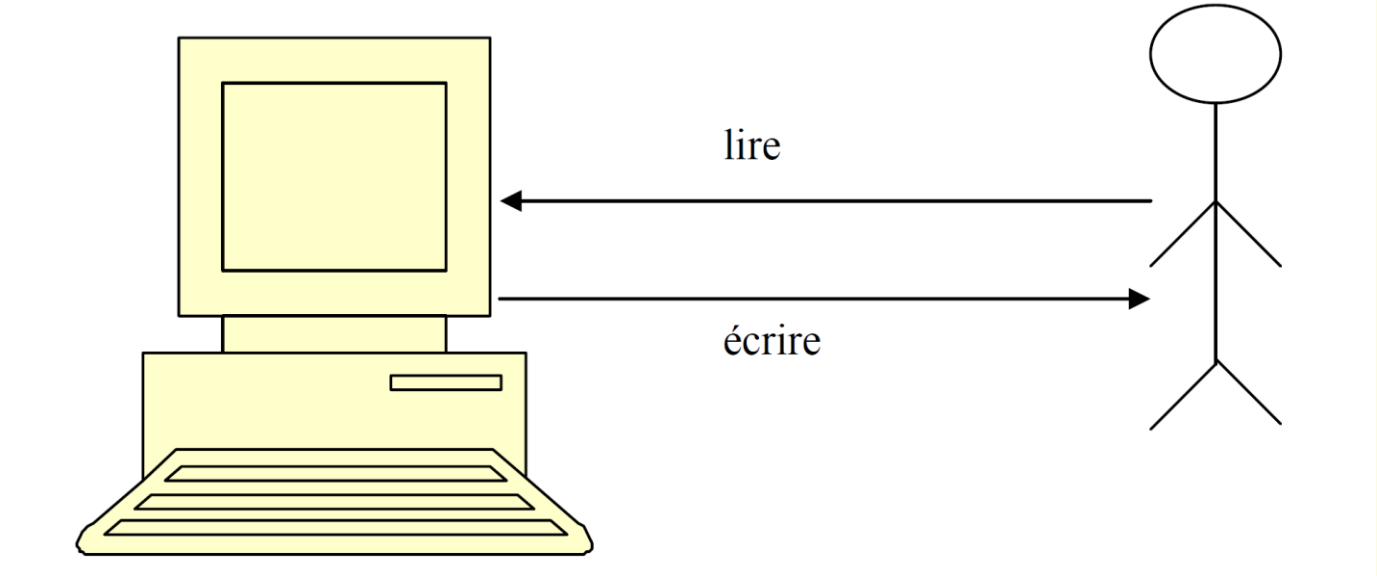
On remarque que :

- Si l’on veut le carré d’un autre nombre que 12, il faut réécrire le programme.
- Le résultat est calculé par la machine elle le garde pour elle, et l’utilisateur qui exécute ce Programme, ne saura jamais quel est le carré de 12.
- C’est pourquoi, il faut utiliser des instructions qui permettent à l’utilisateur de dialoguer avec la machine.

Lire et écrire

```
Variable A,X : entière
Début
 lire (A)
 X ← A^2
 écrire(X)
Fin
```

Lire et écrire



Affectation: Définition et notation

- **L’affectation** est l’action élémentaire dont l’effet est de donner une valeur à une variable (ranger une valeur à une place).
- L'affectation est réalisée au moyen de l'opérateur ← (ou = en C et := en Pascal). Elle signifie " prendre la valeur se trouvant du côté droit (souvent appelée rvalue) et la copier du côté gauche (souvent appelée lvalue) ".

Exercice 01

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

```
Variables A, B en Entier
Début
A ← 1
B ← A + 3
A ← 3
Fin
```

Exercice 02

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

```
Variables A, B, C en Entier
Début
A ← 5
B ← 3
C ← A + B
A ← 2
C ← B - A
Fin
```

Exercice 03

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

```
Variables A, B, C en Entier
Début
A ← 3
B ← 10
C ← A + B
B ← A + B
A ← C
Fin
```

Exercice 04

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

```
Variables A, B en Entier
Début
A ← 5
B ← 2
A ← B
B ← A
Fin
```

Exemple (Affectation):

- Ecrire l'algorithme qui lit 3 nombres entiers a, b, c, calcule la somme et la moyenne et les affiche.

```
Algorithme  calcul ;

Variables
    A, b, c                    : entier ;
    Somme, moyenne             : réel ;

Début
1←
    Lire (a, b, c) ;
2←
    Somme ←  a + b +c ;
3←
    Moyenne ←  somme / 3 ;
4←
    Ecrire (somme, moyenne) ;
5←
fin algorithme
```

Déroulement de l'algorithme

- Dérouler l'algorithme en supposant que les valeurs lues au début sont : 16 dans A, 14 dans B, 12 dans C.

	A	B	C	Somme	Moyenne
1	0	0	0	0	0
2	16	14	12	0	0
3	16	14	12	42	0
4	16	14	12	42	14
5	16	14	12	42	14

2ème exemple d'Algorithme

- Ecrire un algorithme qui lit le nom, prénom, le nombre d'heures d'un employé vacataire, et calcule son salaire et l'affiche sachant que le salaire par heure est fixé à 300 DA, et que le salaire est calculé par la formule :

Salaire = salaire par heure X nombre d'heures

2ème exemple d'Algorithme

Les variables utilisées :

- **Nom** : de type chaine de caractères pour le nom
- **Prenom** : de type chaine de caractères pour le prénom
- **Nbheure** : de type entier pour le nombre d'heures
- **Salaire** : de type réel pour le salaire.

Les constantes

SalaireH = 300

2ème exemple d'Algorithme

```
Algorithme    calculsalaire ;
 Constantes
   salaireH ← 300;
 Variables
   Nom, Prenom : chaîne;
   Nbheure        : entier ;
   Salaire        : réel ;
 Début
1←
   Lire (nom, prenom, nbheure) ;
2←
   Salaire ← nbheure *   salaireh ;
3←
   Ecrire(Nom, prenom, salaire) ;
4←
Fin algorithme
```

Déroulement (2ème exemple)

- supposons qu'on lise : 'Benali' dans le nom et 'omar' dans le prénom, et 20 dans nbheures.

	Nom	Prénom	Nbheure	Salaire
1	' '	' '	0	0
2	'Benali'	'Omar'	20	0
3	'Benali'	'Omar'	20	6000
4	'Benali'	'Omar'	20	6000

Les opérateurs arithmétiques

	En Algorithmique
Addition	+
Soustraction	-
Multiplication	*
Division	/
Résultat entier d'une division	div
Reste entière d'une division	mod

L’évaluation d’une expression arithmétique

- L’évaluation des expressions arithmétiques se fait de gauche à droite en respectant la règle de priorité . L’utilisation des parenthèses casse cette règle de priorité en rendant l’expression entre parenthèses prioritaire au cours de l’évaluation. La priorité est notée comme suit :

L'évaluation d'une expression arithmétique

- ✓ expressions entre parenthèses
- ✓ le moins unaire
- ✓ ** ou ↑, DIV, MOD
- ✓ *, /
- ✓ +, -

Exemples:

1°) a+b*c → a+(b*c)

2°) 2*n + p → (2*n) + p

3°) - a + b → (- a) + b

4°) -a/-b+c → ((-a)/(-b))+c

5°) - a / - (b + c) → (-a)/(-(b + c))

Les opérateurs de comparisons

	En Algorithmique
Supérieur	>
Supérieur ou égal	≥
Inférieur	<
Inférieur ou égal	≤
Egal	=
Différent	≠

Les opérateurs logiques

	En Algorithmique
Fonction ET	Et
Fonction OU	Ou
Fonction NON	Non

Les tests

instruction conditionnelle

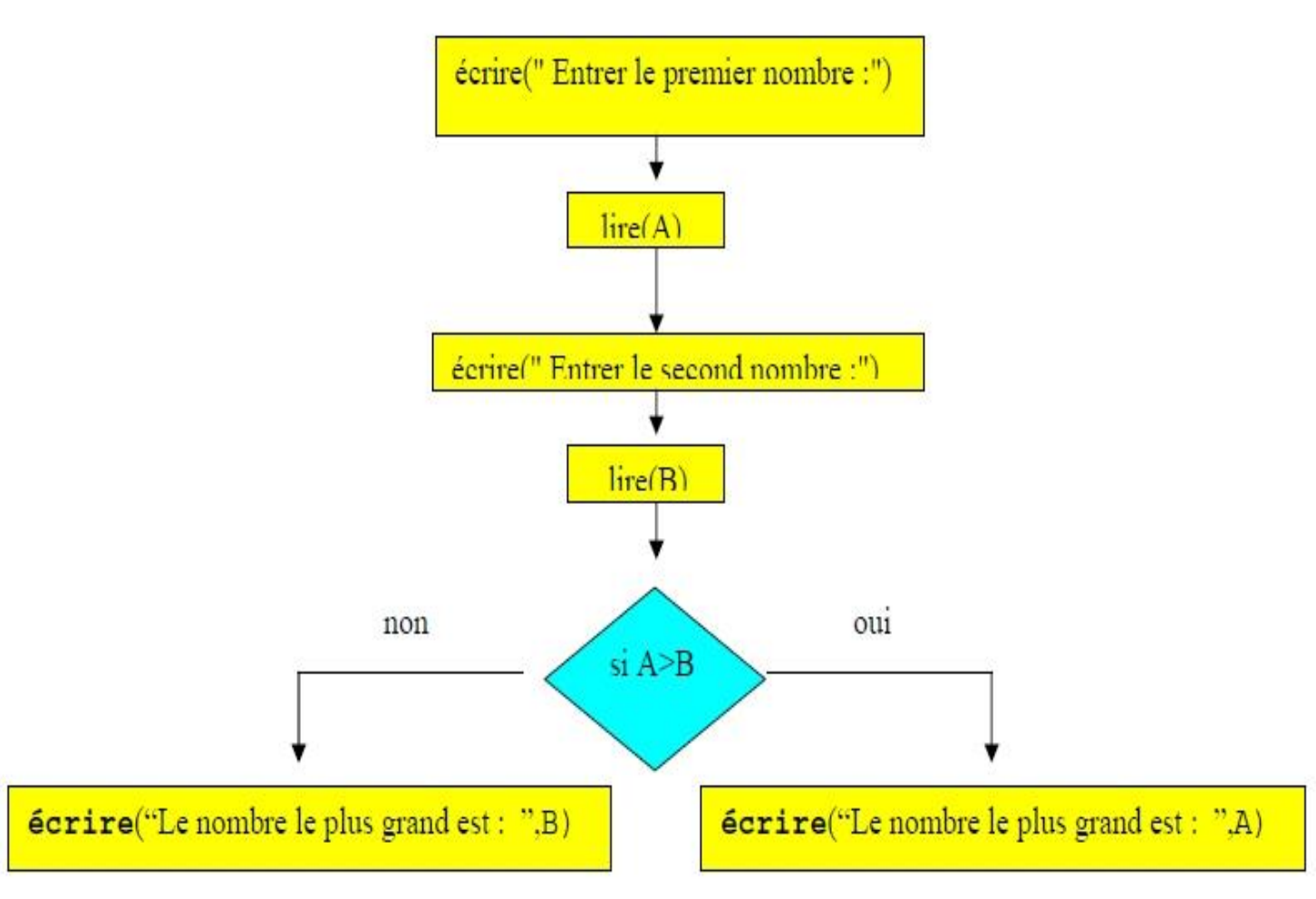
Si C Alors ...

Si C Alors ... Sinon ...

Si .. alors .. sinon ..

Algorithme Max
variables A, B : entiers
Début
écrire("Programme permettant de déterminer le plus grand de deux entiers positifs")
écrire("Entrer le premier nombre : ")
lire(A)
écrire ("Entrer le second nombre : ")
lire(B)
si (A>B) **alors**
écrire("Le nombre le plus grand est : ",A)
sinon
écrire("Le nombre le plus grand est : ", B)
finsi
Fin

Oraganigramme



Exemple 2

Ecrire l'algorithme qui lit les trois nombres A, et B et C cherche le plus petit et l'affiche sur l'écran.

```
Algorithme minimum ;
Variables
  A, B, C, min : entier ;
Début
  Lire(A, B, C) ;
  Si A < B
     Alors min ← A
     Sinon min ← B ;
  Finsi ;
  Si  c < min
     Alors min ← c ;
  Finsi ;
  Ecrire(min) ;
Fin algorithme.
```

Exemple (Tests imbriqués)

```
 Algorithme Imbriquer
Variable Temp en Entier
Début
   écrire("Entrez la température de l'eau :" )
   lire(Temp)
   si Temp =< 0 Alors
    écrire("C'est de la glace")
   sinon
     si Temp < 100 Alors
     écrire("C'est du liquide")
     sinon
     écrire("C'est de la vapeur")
     finsi
     finsi
Fin
```

Les structures répétitives

Exemple:

Ecrire un algorithme qui lit N nombre et fait leur somme

- Les variables utilisées:

Variable	Rôle	Type
N	Contient le nombre de nombres à sommer. Initialement Somme vaut 0.	entier
somme	Contient la somme des nombres	entier
A	La variable dans laquelle on lit le nombre à ajouter à chaque étape.	entier
I	Compteur des nombres . Initialement I vaut 1	entier

Les structures répétitives

Algorithme:

```
Algorithme sommeNB;

Variables
      N, A, I, Somme     :  entier;
Début
   Lire(N);
   Somme←0;
   I  ← 1;
1: Lire (A);
   Somme ← Somme + A;
   I  ←  I+1;
   Si  I ≤ N
      Alors  Aller à  1;
   Finsi
   Ecrire (Somme);
FinAlgorithme.
```

Les structures répétitives

- Dans l'écriture de la boucle il faut définir pour la variable de contrôle les 3 éléments suivant:

1. La valeur initiale
2. Une condition d'arrêt d'exécution
3. Une instruction de modification (incrémentation par exemple) de la variable qui doit s'exécuter à chaque étape de la boucle.

- En s'assurant que A), B) et C) sont bien définies garantit la bonne exécution de la boucle.

Les boucles

Il existe trois façons d'exprimer algorithmiquement l'itération :

- **TantQue**
- **Répéter … jusqu'à …**
- **Pour … jusqu'à …**

La boucle Tantque

- Le schéma de la boucle TantQue est :

TantQue conditions faire

...

Instructions

...

FinTantQue

La boucle Tantque

```
Algorithme somme_NB;
Variables
      N, A, I, Somme     :  entier;
Début
   Lire(N);
   Somme←0;
   I ← 1;
   Tant que i<=N   faire
            Lire (A);
            Somme ← Somme + A;
            I ←  I+1;
    FinTantque
  Ecrire (Somme);
FinAlgorithme.
```

Déroulement de l'algorithme

- Déroulement de l'algorithme pour N=5 et les nombres 10 , -4, 7, 6, 13

I	A	Somme
~~1~~	~~10~~	~~10~~
~~2~~	~~-4~~	~~6~~
~~3~~	~~7~~	~~13~~
~~4~~	~~6~~	~~19~~
~~5~~	13	32
6		

La boucle Répéter … jusqu'à …

- Le schéma de la boucle répéter est :

Répéter

…

Instructions

…

jusqu'à conditions

Ecriture de l'exemple précédent avec Répéter

```
Algorithme somme_NB;

Variables
      N, A, I, Somme     :  entier;
Début
   Lire(N);
   Somme←0;
   I ← 1;
   Répéter
            Lire (A);
            Somme ← Somme + A;
            I ←  I+1;
    Jusqu'à I>N;
  Ecrire (Somme);
FinAlgorithme.
```

La boucle Pour … jusqu'à …

- Le schéma de la boucle Pour est :

Pour i allant de début **jusqu'**à fin **faire**

…

Instructions

…

FinPour

Ecriture de l'exemple précédent avec Pour

```
Algorithme somme_NB;

Variables
      N, A, I, Somme     :  entier;
Début
   Lire(N);
   Somme←0;
   Pour I  ← 1 à N
      faire
         Lire (A);
         Somme ← Somme + A;
   FinPour
  Ecrire (Somme);
FinAlgorithme.
```

La boucle tantque

Algorithme TT
Variable T en entier
Debut
  Truc ⟵ 0
  TantQue (T < 15) faire
  T ⟵T + 1
  Ecrire (’’ Passage numéro : ", T)
  FinTantQue
 Fin

La boucle Répéter

Algorithme TT
Variable T en entier
Debut
T ⟵ 0
Répéter
T ⟵T + 1
Ecrire (’’ Passage numéro : ", T)
Jusqu’à (T >= 15)
Fin

La boucle Pour

Algorithme TT

Variable T en entier

Debut

Pour T⟵ 1 à 15) **faire**

Ecrire (’’ Passage numéro : ", T)

Finpour

Fin

Les boucles imbriquées

```
Algorithme  gg
Variables i, j entier
Debut
  Pour i allant de 1 à 10 faire
   écrire("Première boucle") ;
     Pour j allant de 1 à 6 faire
     écrire("Deuxième boucle") ;
     Finpour ;
   Finpour ;
Fin
```

Les boucles imbriquées

```
Algorithme  ggg
Variables i, j entier
Debut
  Pour i allant de 1 à 10 faire
   écrire("Première boucle") ;
  Finpour ;
  Pour j allant de 1 à 6 faire
     écrire("Deuxième boucle") ;
  Finpour ;
Fin
```

Méthodologie pour l'écriture d'une boucle :

- repérer une action répétitive, donc une boucle.
- choix entre boucle avec compteur ou sans

Question ? Peut-on prévoir/déterminer le nombre d'itérations ?

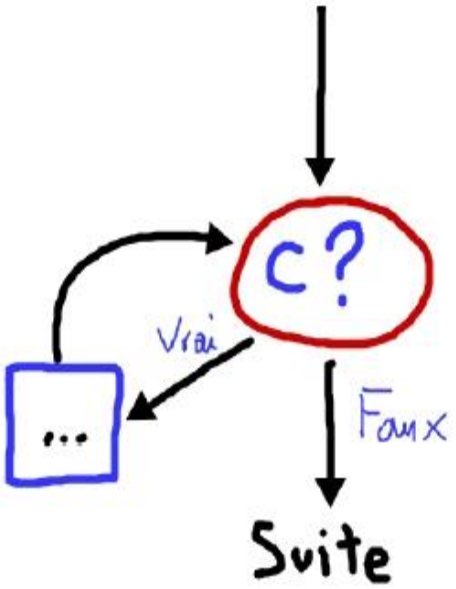
- ✓ si oui, boucle avec compteur : **la boucle pour …**
- ✓ si non, boucle sans compteur

Est ce que il faut commencer l'action avant de tester ou l'inverse ?

- ❖ si tester d'abord, alors boucle **TantQue**
- ❖ si action puis tester, alors **Répéter … jusqu'à**

Les boucles

Tant que C Faire: ...



Répéter ... jusqu'à C