Conception de Base de L'Algorithmique et Programmation en PASCAL

Cours Informatique 1er année S2.

LES BASES DE L'INFORMATIQUE (1ere année S1)

Dr. ABADA E-mail : abada_hec@yahoo.com

Plan de cours

Conception de Base de L'Algorithmique et Programmation en PASCAL

Chapitre 1.

Notions Générales pour l’algorithmique

1. Notions Générales

Le mot algorithme :

- Le mot **algorithme** vient du mot arabe الخوارزمي , nom du mathématicien persan du IX ^{e}(9) siècle Al-Khwârizmî.
- Al-Khwârizmî introduisit la numération décimale et les règles élémentaires des calculs.
- La notion d'algorithme est donc historiquement liée aux manipulations numériques, mais elle s'est progressivement développée pour porter sur des objets de plus en plus complexes, des textes, des images, des formules logiques, des objets physiques, etc.

1. Notions Générales

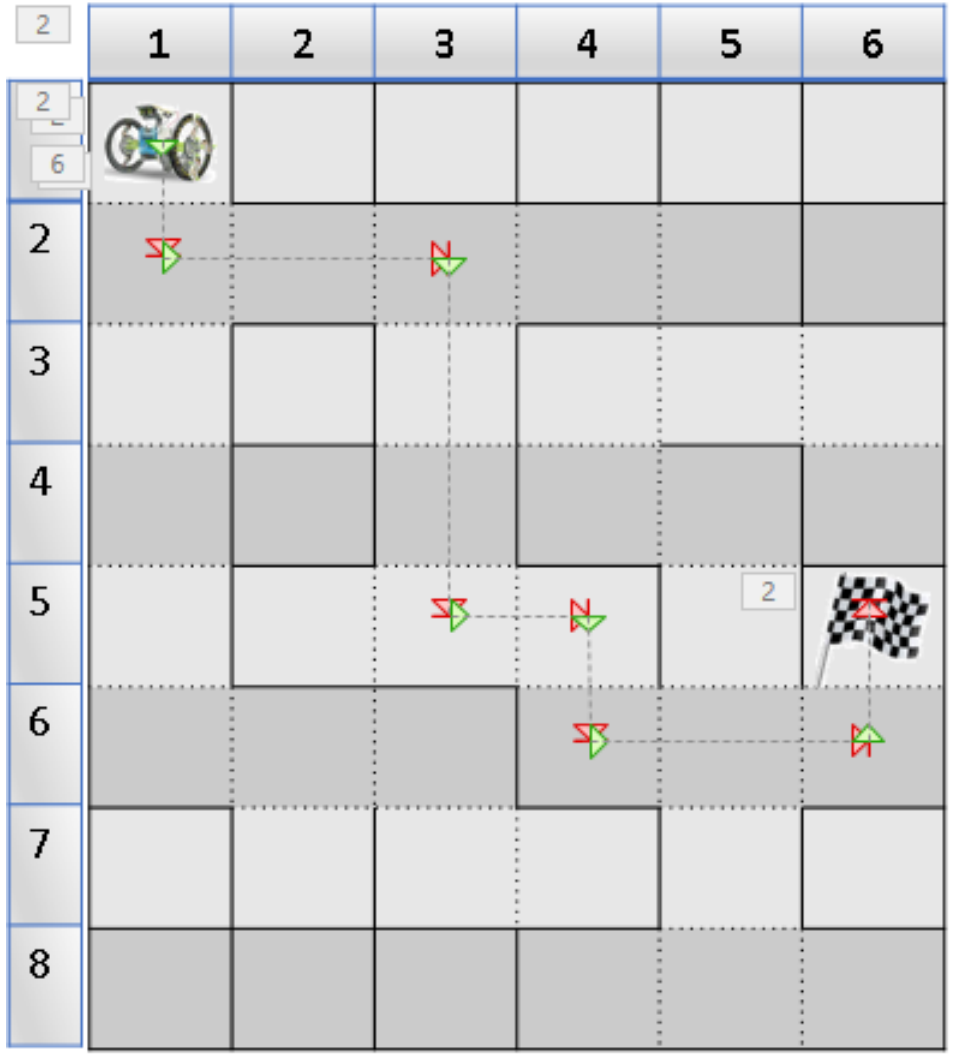
Qu'est-ce qu'un algorithme :

- Un **algorithme** est une méthode de résolution de problème;
- Énoncée sous la forme d'une série d'opérations ou d’instructions à effectuer;
- La mise en œuvre de l'algorithme consiste en l'écriture de ces opérations dans un langage de programmation;
- et constitue alors la brique de base d'un programme informatique.

1. Notions Générales

Exemples d'un algorithme :

1. Déplacement d'un robot dans un environnement:
2. Quelle sont les instructions pour déplacer le robot vers l'objectif ?
   a. **Objectif :** déplacer le robot de la case (1,1) à la case (5,6).
   b. **Les données :** le robot, la carte (la matrice), et le but
   c. **Début**
      i. Déplacer un pas vers le bas;
      ii. Déplacer deux pas vers la droit;
      iii. Déplacer trois pas vers le bas;
      iv. Déplacer un pas vers la droit;
      v. Déplacer un pas vers le bas;
      vi. Déplacer deux pas vers la droit;
      vii. Déplacer un pas vers le haut;
   d. **Fin.**



1. Notions Générales

Exemples d'un algorithme :

1. La somme de deux nombres :

2. Quelle sont les instructions pour faire la somme de deux nombres ?

   a. **Objectif :** calcul la somme de deux nombres.

   b. **Les données :** le premier nombre, le deuxième nombre, le résultat

   c. **Début**

      i. Saisir le premier nombre par le clavier.

      ii. Mettre le premier nombre dans un espace mémoire nommé x.

      iii. Saisir le deuxième nombre par le clavier.

      iv. Mettre le deuxième nombre dans un espace mémoire nommé y.

      v. Faire la somme des deux valeurs x et y et mettre le résultat dans un espace mémoire nommé z.

      vi. Afficher la valeur de z sur écran.

   d. **Fin.**

1. Notions Générales

Qu'est-ce qu'un algorithme :

- Un algorithme est une séquence bien définie d'opérations (calcul, manipulation de données, etc.)
- permettant d'accomplir une tache en un nombre fini de pas.
- En principe un algorithme est indépendant de toute implantation.
- Cependant dans la pratique de la programmation il s'avère indispensable de tenir compte des capacités du langage de programmation utilisé.

1. Notions Générales

Qu'est-ce qu'un algorithme :

La conception d’un algorithme passe par plusieurs étapes :

- **Analyse** : définition du problème en terme de séquences d’opérations de calcul de stockage de données, etc. ;
- **Conception** : définition précise des données, des traitements et de leur séquencement ;
- **Implantation** : traduction et réalisation de l’algorithme dans un langage précis ;
- **Test** : Vérification du bon fonctionnement de l’algorithme.

1. Notions Générales

Les caractéristiques d'un Algorithme :

Un algorithme est une marche à suivre :

- dont les opérations sont toutes définies et portent sur des objets appelés informations,
- dont l'ordre d'exécution des opérations est défini sans ambiguïté,
- qui est réputée résoudre de manière certaine un problème ou une classe de problèmes,
- s'exprime dans un langage indépendant des langages de programmation,

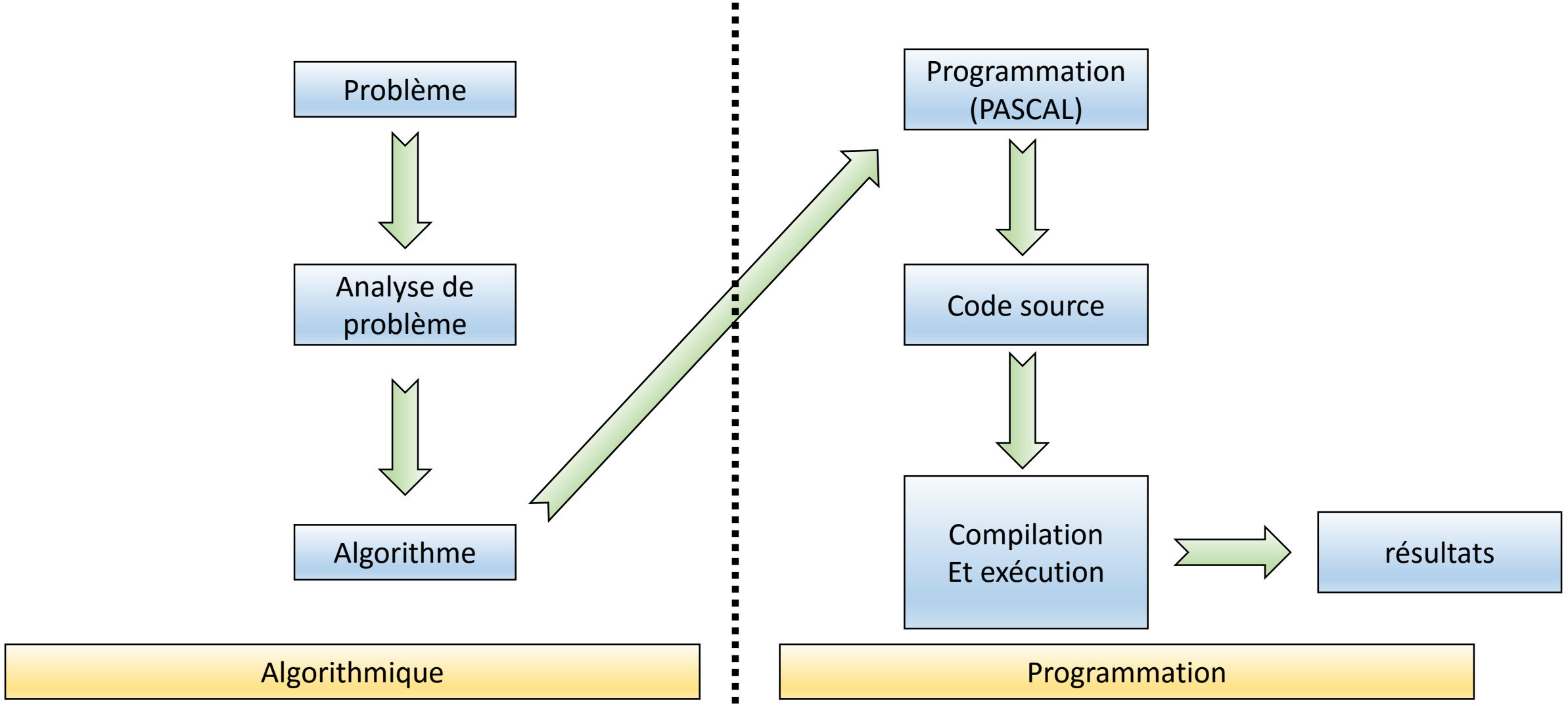
1. Notions Générales

Un algorithme et un langage de programmation :

- **Un programme est la traduction d'un algorithme dans un certain langage de programmation.**
- Il faut savoir qu'à chaque instruction d'un programme correspond une action du processeur.
- Le langage de programmation est utilisé pour traiter des données afin d'obtenir des résultats par ordinateur.
- Le langage de programmation est une Intermédiaire entre le langage machine (binaire) et le langage humain

1. Notions Générales

Un algorithme et un langage de programmation :



1. Notions Générales

Structure générale d'un algorithme :

❖ Un algorithme doit être lisible et compréhensible par plusieurs personnes.

❖ Un algorithme doit suivre des règles précises, il est composé d'une **entête** et d'un **corps.**

1. l'entête, qui spécifie :
   - **Nom :** le nom de l'algorithme
   - **Rôle :** son utilité
   - **Entrée :** les données « en entrée », c'est-à-dire les éléments qui sont indispensables à son bon fonctionnement
   - **Sortie :** les données « en sortie », c'est-à-dire les éléments calculés, produits, par l'algorithme
   - **Déclaration :** les données locales à l'algorithmique qui lui sont indispensables
2. le corps, qui est composé :
   - **début :** un mot clef qui indique le début des instructions
   - **instructions :** une suite d'opérations
   - **fin :** un mot clef qui indique la fin des instructions

1. Notions Générales

Structure générale d'un algorithme :

❖ Un algorithr

❖ Un algorithr

1. l'entête, c
   - **Nom :** le
   - **Rôle :** so
   - **Entrée :**
   - **Sortie :** l
   - **Déclarat**
2. le corps, c
   - **début :** u
   - **instructions :** une suite d'opérations
   - **fin :** un mot clef qui indique la fin des instructions

ment

Algorithme : nom
Rôle : que fait cet algorithme
Entrée : les données nécessaires
Sortie : les résultats produits par l'algorithme
Variables : la déclaration des variables
} Facultatifs

Debut
Instruction 1
Instruction 2
… … … /* les commentaires explicatives des instructions */
Instruction k
Fin

1. Notions Générales

Exemples d'un algorithme :

a. **Nom:** calcul chemin

b. **Rôle :** déplacer le robot vers le but.

c. **Entrée :** position du robot (1,1), position du but (5,6).

d. **Sortie :** les déplacement (bas, droit, bas, droit, bas, droit, haut)

e. **Début**

i. Déplacer un pas vers le bas;

ii. Déplacer deux pas vers la droit;

iii. Déplacer trois pas vers le bas;

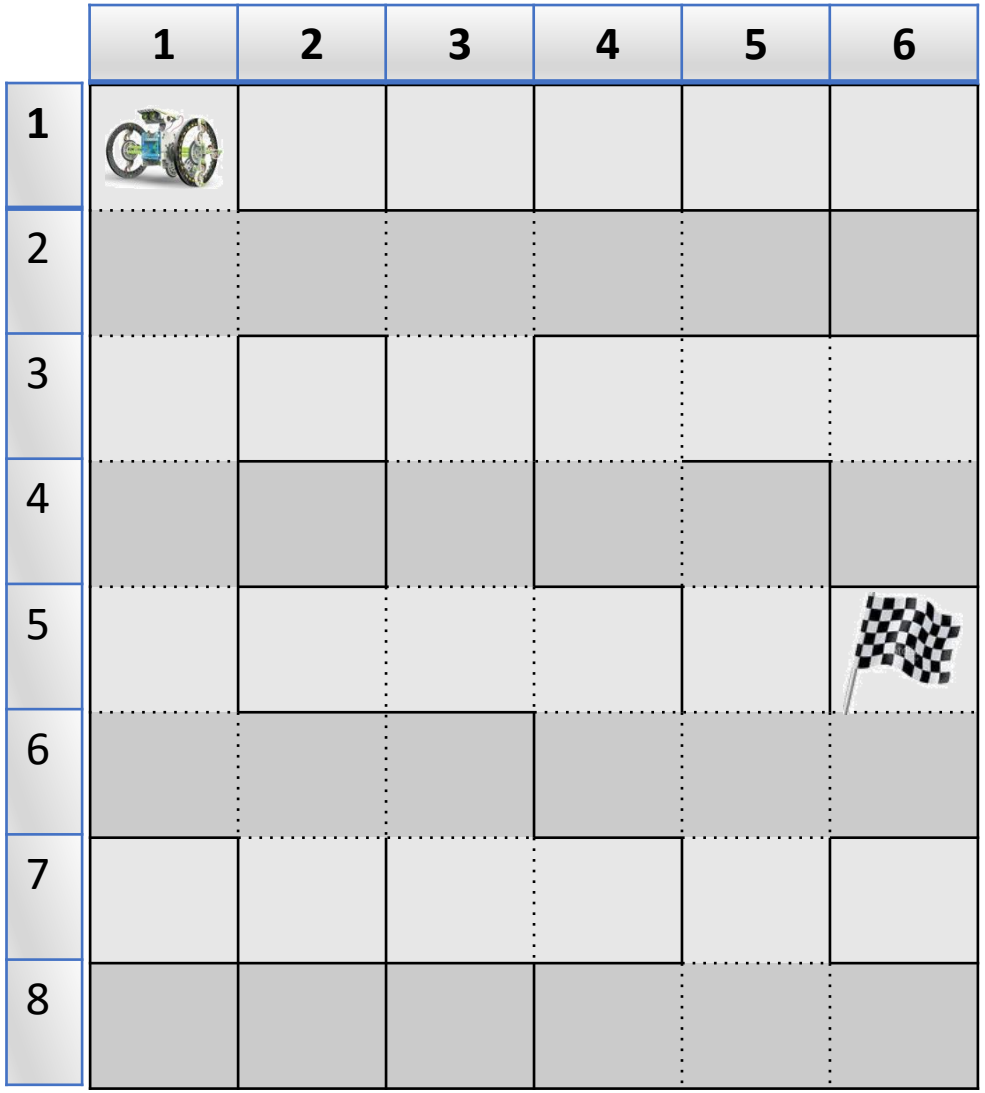
iv. Déplacer un pas vers la droit;

v. Déplacer un pas vers le bas;

vi. Déplacer deux pas vers la droit;

vii. Déplacer un pas vers le haut;

f. **Fin.**



1. Notions Générales

Exemples d'un algorithme :

1. La somme de deux nombres :
   a. **Nom :** calcul la somme
   b. **Rôle :** calcul la somme
   c. **Entrée :** le premier nombre x, le deuxième nombre y
   d. **Sortie :** résultat
   e. **Début**
      i. **Saisir le premier nombre par le clavier.**
      ii. Mettre le premier nombre **dans un espace mémoire nommé x**.
      iii. **Saisir le deuxième nombre par le clavier.**
      iv. Mettre le deuxième nombre **dans un espace mémoire nommé y**.
      v. Faire la somme des deux valeurs x et y et mettre le résultat dans un espace mémoire nommé z.
      vi. Afficher la valeur de z sur écran.
   f. **Fin.**

1. Notions Générales

Pseudo Langage :

- ✓ Un algorithme doit être lisible et compréhensible par plusieurs personnes.
- ✓ Il doit donc suivre des **règles précises** .
- ✓ Le plus important pour un algorithme sont **les déclarations** ainsi que **les instructions** qui constituent le corps de l'algorithme.

1. Notions Générales

Pseudo Langage :

- ✓ Dans un programme informatique, on va avoir en permanence besoin de stocker provisoirement des valeurs.
- ✓ Il peut s'agir de données issues du disque dur ou fournies par l'utilisateur (frappées au clavier)
- ✓ Ces données peuvent être de plusieurs types :
  - ✓ des nombres,
  - ✓ du texte,
  - ✓ Booléen (VRAI/FAUX)
  - ✓ … etc.

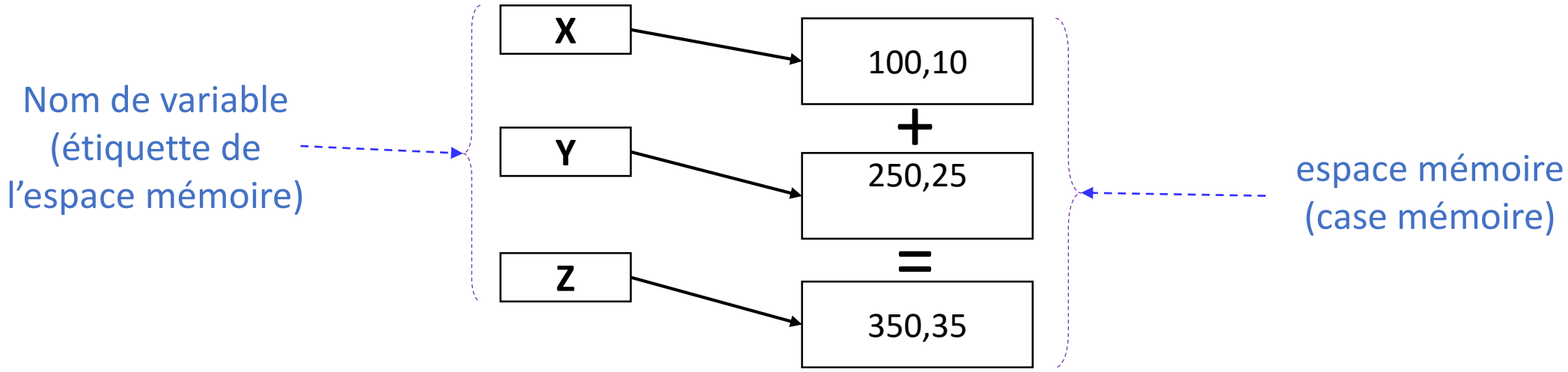
1. Notions Générales

Les variables :

Dès que l'on a besoin de stocker une information au cours d'un programme, on utilise une **variable.**

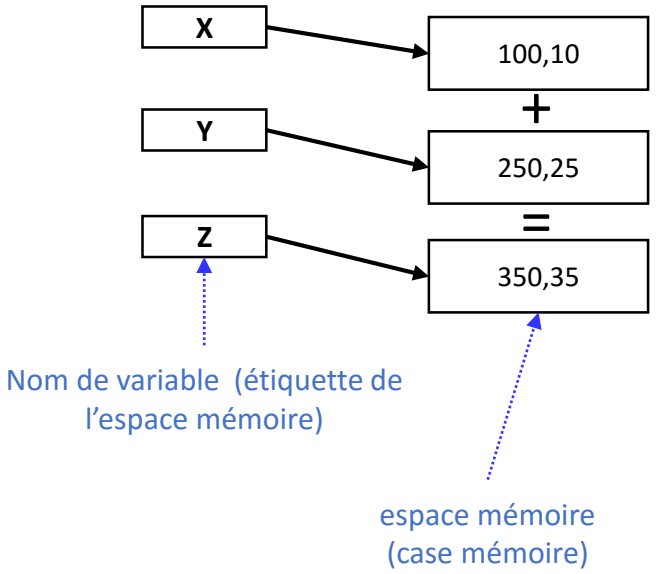
Déclaration des variables

- ✓ La première chose à faire avant de pouvoir utiliser une variable est de **créer la boîte et de lui coller une étiquette**.
- ✓ Ceci se fait tout au début de l'algorithme, avant même les instructions proprement dites.
- ✓ C'est ce qu'on appelle la **déclaration des variables**.
- ✓ Une variable ne peut être utilisée que s'elle est déclarée.
- ✓ La déclaration se fait par la donnée du **nom** de la variable et du **type** de la variable.



1. Notions Générales

Les variables :



✓La syntaxe d'une déclaration de variable en pseudo-langage :

Variable <nom variable> : <type>

Variables <nom var1, nom var2,nom var3…> : <type>.

✓Exemple:

Variable x : entier

Variables x , y , z : entier.

1. Notions Générales

Noms de variables :

Le nom de la variable obéit à des règles qui changent selon le langage utiliser.

Les principales règles à respecter sont :

- ✓Le nom de variable peut comporter des lettres et des chiffres,
  On exclut la plupart des signes de ponctuation, en particulier les espaces.
- ✓Un nom de variable doit commencer par une lettre.
- ✓Le nombre maximal de caractères qui composent le nom d'une variable dépend du langage utilisé.
- ✓Ne pas utiliser les mots clés du langage de programmation.
- ✓**Exemples :** x , y , z , résultat , résultat1 , nb , nb_2 … Etc.

1. Notions Générales

Types de variables :

- ✓ Lorsqu'on déclare une variable, il ne suffit pas de réserver un emplacement mémoire; il faut préciser ce que l'on voudra mettre dedans, car de cela dépendent la **taille** de l'emplacement mémoire et le **type de codage** utilisé.
- ✓ Commençons par le cas très fréquent, celui d'une variable destinée à recevoir **des nombres**
- ✓ Si l'on réserve un octet pour coder un nombre, on ne pourra coder que $2^8 = 256$ valeurs différentes. Cela peut signifier par exemple les nombres entiers de 1 à 256, ou de 0 à 255, ou de –127 à +128.
- ✓ Si l'on réserve deux octets, on a droit à 2^{16} =65 536 valeurs ; avec trois octets, 2^{24} =16 777 216, etc

1. Notions Générales

Types de variables :

Type Numérique	Plage
Octet	(2^8) de 0 à 255
Entier simple	(2^{16}) de -32768 à 32767
Entier double	(2^{32}) de -2147483648 à 2147483647
Réel simple	de $-3.40x10^{38}$ à $-1.40x10^{-45}$ pour les négatives de $1,40x10^{-45}$ à $3.40x10^{38}$ pour les positives
Réel double	de $-1.79x10^{308}$ à $-4.94x10^{-324}$ les négatives de $4.94x10^{-324}$ à $1.79x10^{308}$ les positives

Alphanumérique, caractère	on stocke des caractères: lettres, signes de ponctuation, d'espaces, ou chiffres. Le nombre maximal de caractères stockés dépend du langage utilisé.
Chaine de caractère	Un groupe de caractères est appelé chaîne de caractères. En pseudo-code (Algo), une chaîne de caractères est toujours notée entre guillemets " ". Par exemple : "chaine 123"
Type booléen	on y stocke uniquement les valeurs logiques VRAI et FAUX

1. Notions Générales

Types de déclarations :

- ✓Variable : x:Entier;

  Y:Réel;

  Message: String;

- ✓Variables : x , y , z : Entier;

1. Notions Générales

Les constantes :

✓Une constante est désignée par un identificateur et une valeur, qui sont fixés en début de programme, par le mots clés **constante** ou **cons**.

✓La valeur ne peut pas être modifiée, et ne peut pas être une expression.

Par exemple:

✓constante : **pi** : réel = 3,14;

a : Entier = 100;

1. Notions Générales

Exemples de variables :

1. La somme de deux nombres :
   - **a.** **Nom :** calcul la somme
   - **b.** **Entrée :** x, y
   - **c.** **Sortie :** z
   - **d.** **Variables** x , y , z : entier
   - **e.** **Constante** pi : réel = 3,14
   - **f.** **Début**
     - **i.** **Saisir le premier nombre par le clavier.**
     - ii. Mettre le premier nombre **dans un espace mémoire nommé x**.
     - **iii.** **Saisir le deuxième nombre par le clavier.**
     - iv. Mettre le deuxième nombre **dans un espace mémoire nommé y**.
     - v. Faire la somme des deux valeurs x et y et mettre le résultat dans un espace mémoire nommé z.
     - vi. Afficher la valeur de z sur écran.
   - **g.** **Fin.**

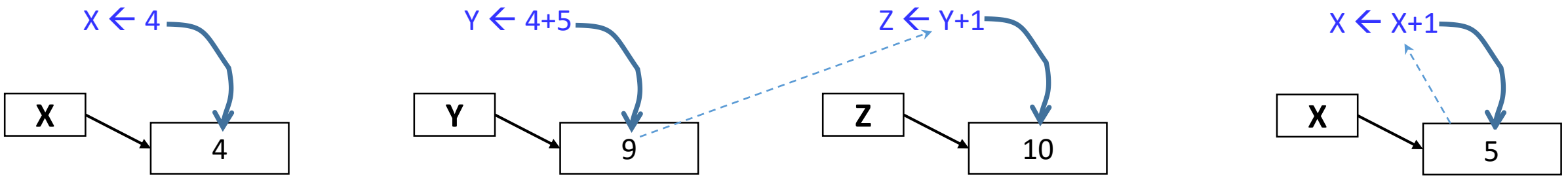
Chapitre 2.

Instructions Elémentaires

2. Instructions Elémentaires

Affectation :

- ✓ 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 ← (:= en Pascal)
- ✓ Elle signifie " prendre la valeur se trouvant du côté droit et la copier du côté gauche "
- ✓ Le coté droit représente toute constante, variable ou expression capable de produire une valeur
- ✓ Exemples:



2. Instructions Elémentaires

Affectation :

- ✓ Le coté droit représente toute constante, variable ou expression capable de produire une valeur
- ✓ Le coté gauche doit être une variable distincte et nommée (autrement dit, il existe un emplacement physique pour ranger le résultat).
- ✓ Par exemple:

X ← 4 correct

4←X Incorrect

X ← x+y+4 correct

4+x←X Incorrect

2. Instructions Elémentaires

Affectation :

Exercice 1: Donnez le résultat de l'affectation de chaque instruction :

Algorithme Affectation

Variables X , Y , Z : entier;

N : caractère;

Constates A1 : entier = 50;

Début

X ← 4

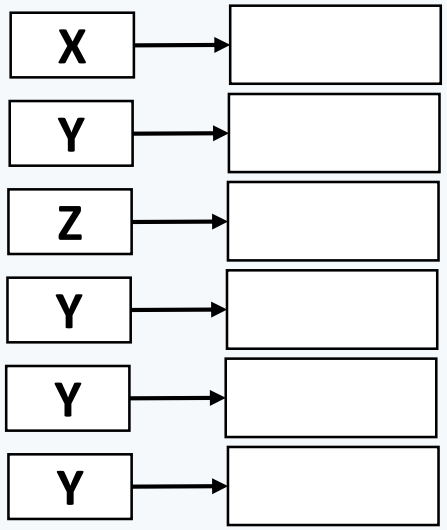
Y ← A

Z ← A + X

Y ← Y + 2

Y ← A + Z

N ← "EHEC"



Fin

Exercice 2 : Donnez les affectations nécessaire pour échanger deux valeur

Exemple : soit deux variables x = 10 et y = 20. on veut mettre la valeur de x dans y et la valeur de y dans x (x=20 et y = 10).

Algorithme Permutation

Variables X , Y : entier;

Début

1- .....

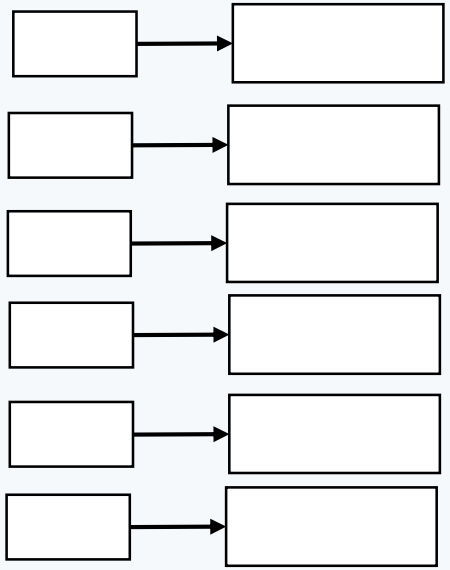
2- .....

3- .....

4- .....

5- .....

6- .....



Fin

2. Instructions Elémentaires

Affectation :

Exercice 1(Correction):

Algorithme Affectation

Variables X , Y , Z : entier;

N : caractère;

Constates A1 : entier = 50;

Début

X ← 4

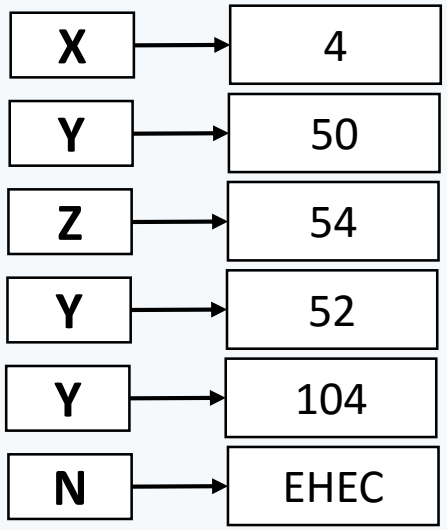
Y ← A1

Z ← A1 + X

Y ← Y + 2

Y ← A1 + Z

N ← "EHEC"



Fin

Exercice 2 (correction) : Donnez les affectations nécessaire pour échanger deux valeur

Algorithme Permutation

Variables X , Y, temp : entier;

Début

1- X ← 20

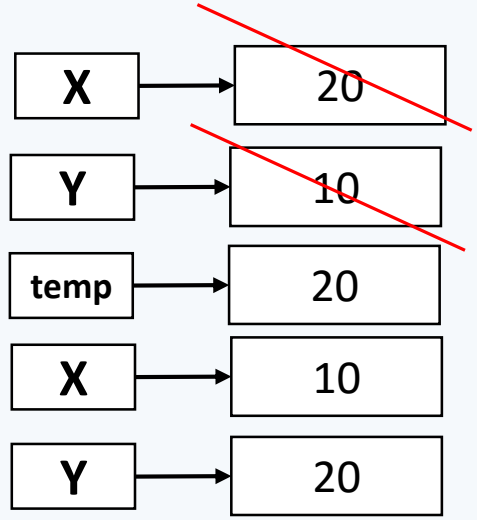
2- Y ← 10

3- temp ← X

4- X ← Y

5- Y ← temp

6- .....



Fin

2. Instructions Elémentaires

Affectation (Expression et opérateurs)

Expression arithmétique

Dans une instruction d'affectation, on trouve :

- ✓ à gauche de la flèche, un nom de variable,
- ✓ à droite de la flèche, ce qu'on appelle une expression : **un ensemble de valeurs, reliées par des opérateurs, et équivalent à une seule valeur**
- ✓ L'expression située à droite de la flèche doit être du même type que la variable située à gauche.

Si l'un des trois points énumérés ci-dessus n'est pas respecté, la machine sera incapable d'exécuter l'affectation, et déclenchera une erreur

2. Instructions Elémentaires

Affectation (Expression et opérateurs)

Opérateur

Un opérateur est un signe qui relie deux valeurs, pour produire un résultat.

Opérateurs numériques :

Ce sont les quatre opérations arithmétiques :

+ addition

- soustraction

***** multiplication

/ division

Mentionnons également le **^** qui signifie « puissance ».

La multiplication et la division sont prioritaires sur l'addition et la soustraction.

Opérateur alphanumérique & :

Cet opérateur permet de concaténer deux chaînes de caractères.

2. Instructions Elémentaires

Affectation (Expression et opérateurs)

Exemple :

- X*Y+Z ⇔ (X*Y)+Z
- X+Y*Z ⇔ X+(Y*Z)
- X*Y+Z/X ⇔ (X*Y)+(Z/X)
- 10+5*4 = 10+(5*4) = 10+20 = 30
- 10/2+5*4 = (10/2)+(5*4) = 5+20 = 25
- 4*2/4+10/2 = ((4*2)/4)+(10/2) = 8/4+5 = 2+5 = 7

Concaténation :

- "Ecole" & " Préparatoire" & " HEC" = "Ecole Préparatoire HEC";

2. Instructions Elémentaires

Instructions d'Entrée /Sortie

```
Algorithme addition
Variables   X , Y : entier;
Début
1- X = 10
2- Y = 20
3- Z = X + Y
Fin
```

Cet algorithme nous donne la somme de 10 et 20 soit 30.

On remarque que :

si l'on veut la somme des deux autres nombres que 10 et 20, il faut réécrire l'algorithme.

Le résultat est calculé par la machine elle le garde pour elle, et l'utilisateur qui exécute ce programme, ne saura jamais quel est la valeur de Z.

C'est pourquoi, il faut utiliser des instructions qui permettent à l'utilisateur de dialoguer avec la machine.

2. Instructions Elémentaires

Instructions d'Entrée /Sortie **(Lire)**

- Pour pouvoir effectuer un calcul sur une variable, la machine doit connaître la valeur de cette variable.
- Si cette valeur n'a pas été déterminée par des initiations ou des calculs précédents, il faut que l'utilisateur lui fournisse, c'est une **donnée**.
- Il s'agit alors d'introduire une valeur à partir de l'extérieur de la machine
- **l'algorithme doit contenir l'instruction qui permet de lire une donnée à partir de clavier.**

L'instruction est : **Lire(<variable>)** Exemple : **Lire(X);**

2. Instructions Elémentaires

Instructions d'Entrée /Sortie **(Lire)**

- Si on veut connaître le résultat d'un calcul ou le contenu d'une variable X, on doit l'afficher sur écran.

- **l'algorithme doit contenir l'instruction qui permet de lire une donnée à partir de clavier.**

L'instruction est : **Ecrire(<variable>)** Exemple : **Ecrire (X);**

Elle signifie : mettre sur l'écran (organe de sortie de la machine) le contenu de la case X.

2. Instructions Elémentaires

Instructions d'Entrée /Sortie

Utilisateur

Saisie une valeur avec clavier X

Saisie une valeur avec clavier Y

Afficher le résultat sur écran

Machine

Algorithme addition
Variables X , Y : entier;
Début
1- Lire(X);
2- Lire(Y);
3- Z ← X + Y;
4- Ecrire(Z);
Fin

Exercice : Réécrire l'algorithme de permutation avec les instructions de lecture et écriture

Algorithme permutation
Variables X , Y : entier;

Début

1- ……

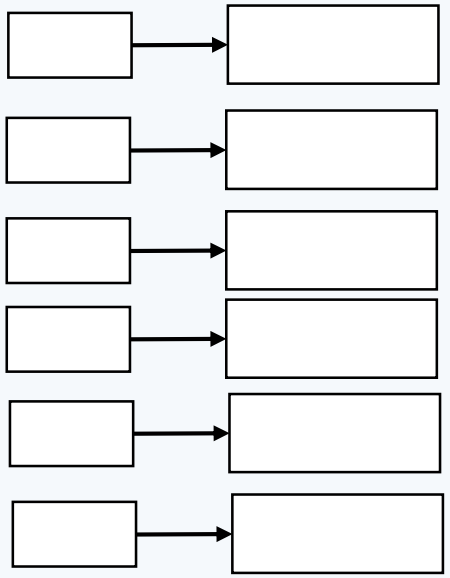
2- ……

3- ……

4- ……

5- ……

6- ……



Fin

Exercice : Solution 1 pour l'algorithme de permutation

Algorithme permutation
Variables X , Y **, temp** : entier;

Début

1- Lire (x);

2- Lire(y);

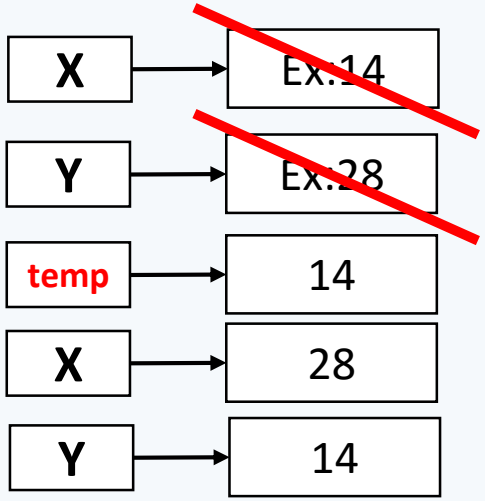
3- **temp** ← x;

4- X← y;

5- Y← **temp**;

6- écrire(x,y);

Fin



Exercice : Solution 2 pour l'algorithme de permutation

Algorithme permutation2
Variables X , Y: entier;

Début

1- Lire (x);

2- Lire(y);

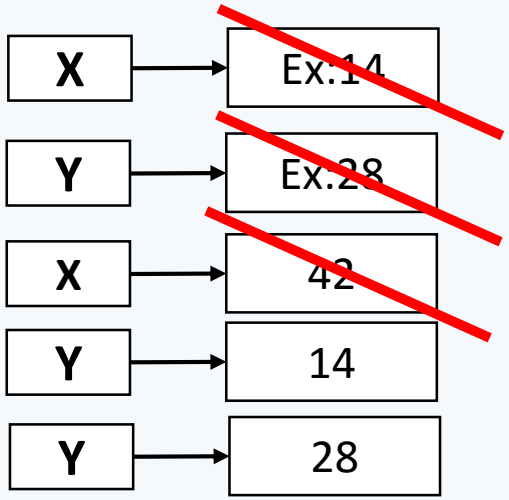
3- X← x + y;

4- Y← X-Y;

5- X←X-Y;

6- écrire(x,y);

Fin



2. Traduction en PASCAL

Editeur free PASCAL (ou my pascal)

Exercice 1 : écrire un programme pascal qui permet d'afficher un texte (chaine de caractères)

« je suis étudiant de EHEC ».

Solution :

Algorithme afficher_HEC;
Début

Ecrire('je suis étudiant de l'EHEC');

Fin.

```
PROGRAM afficher_EHEC;

BEGIN
  writeln('je suis étudiant de l''EHEC');
END.
```

```
PROGRAM afficher_EHEC;

BEGIN
  writeln('je suis étudiant de l''EHEC');
  readln
END.
```

2. Traduction en PASCAL

Editeur free PASCAL

Exercice 1 : écrire un programme pascal qui permet de lire et afficher un texte (chaine de caractères).

Solution :

```
Algorithme afficher_texte;
Variable
        t : caractère
Début

Lire (t);
Ecrire(t);

Fin.
```

```
PROGRAM afficher_texte;
VAR
   t : STRING;

BEGIN
   readln(t);
   writeln(t);

   readln
END.
```

2. Instructions Elémentaires

Traduction en PASCAL

Exercice 1 : écrire un programme pascal fait la somme de deux nombres.

Algorithme addition
Variables X , Y , Z : entier;

Début

1- Lire(X);

2- Lire(Y);

3- Z ← X + Y;

4- Ecrire(Z);

Fin

```
PROGRAM addition;
VAR
   x,y,z : INTEGER;

BEGIN
   readln(x);
   readln(y);

   z := x+y;


   writeln('la somme est :',z);

   readln
END.
```

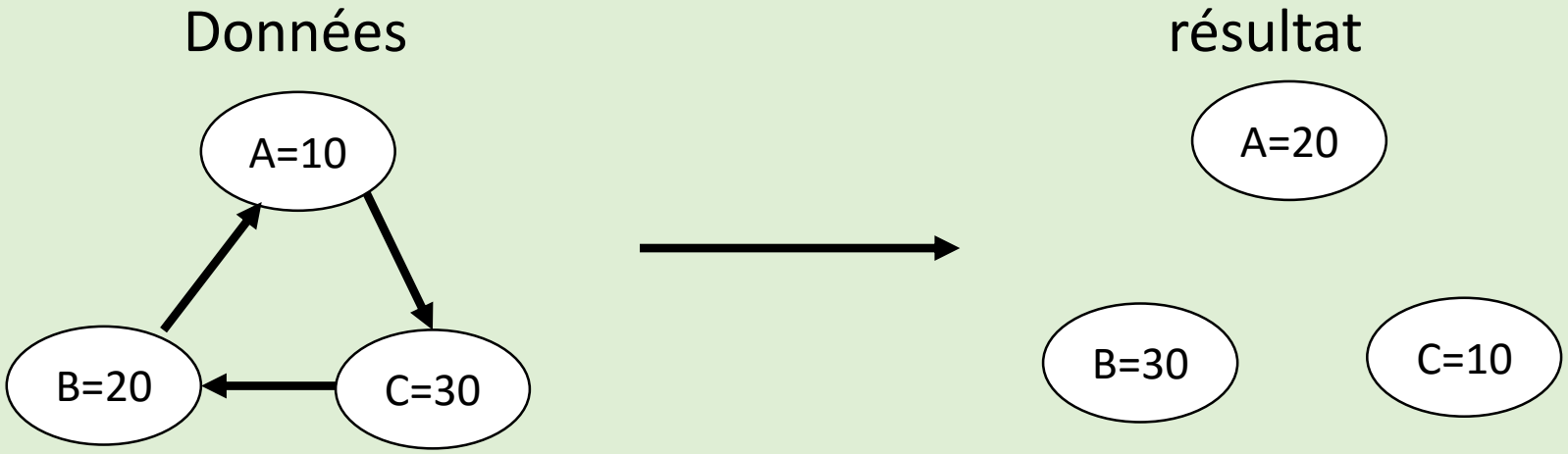
TD

Exercice N°1 : Soit trois nombres réels A, B, C. Ecrire un Algorithme qui fait une permutation circulaire de ces trois nombres. Traduire cet algorithme en code pascal et faire l'exécution.

Exemple:

si A=10 , B=20 et C = 30

le résultat : doit être :A = 20, B=30 et C = 10



TD

Exercice N°1 : La solution:

Algorithme permutation_circulaire ;
Variables :
A,B,C,Temp : **entiers**;
Début

Lire (A);
Lire (B);
Lire (C);

Temp ← A;
A← B;
B←C;
C←Temp;

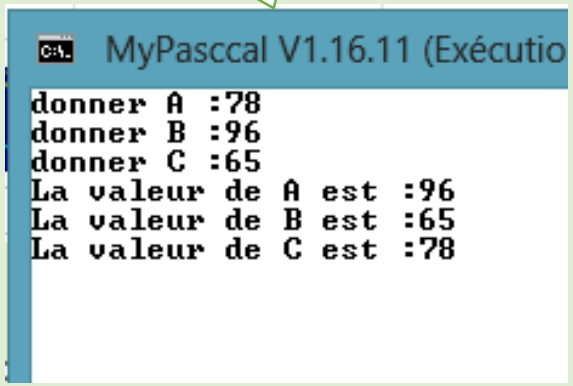
Ecrire (A);
Ecrire(B);
Ecrire(C);

Fin.

1- L'Algorithme

2- Le programme pascal saisie avec l'éditeur « My Pascal »

1- Apres l'exécution



permutation.pas

```
1   program permutation_circulaire ;
2   var
3   A,B,C,Temp : integer;
4   begin
5      //cette partie concerne les données
6      write('donner A :');
7      readln(A);
8      write('donner B :');
9      readln(B);
10     write('donner C :');
11     readln(C);
12
13     //cette partie concerne les traitements
14     Temp := A;
15     A:=B;
16     B:=C;
17     C:=Temp;
18
19     //cette partie concerne l'affichage
20     writeln('La valeur de A est :',A);
21     writeln('La valeur de B est :',B);
22     writeln('La valeur de C est :',C);
23  end.
```

TD

Exercice N°2 : Ecrivez un algorithme qui calcul et affiche la surface et la circonférence d'un cercle ($2\pi r$ et πr^2). L'algorithme demandera à l'utilisateur d'entrer la valeur du rayon.

TD

Exercice N°2 : La solution:

Algorithme permutation_circulaire ;
Variables :
Surf,Circ, Ray: **entiers**;
Constantes :
Pi:reel=3,14;
Début
Lire(R);
Surf ← Pi * Ray^2;
Circ ← 2*Pi*Ray;
Ecrire (Surf);
Ecrire(Circ);
Fin.

1- L'Algorithme

2- Le programme pascal saisie avec l'éditeur « My Pascal »

permutation.pas | exo2.pas

```
1   program calcul;
2   var
3      Surf,Circ,Ray : real;
4   //la constante Pi est déjà existé dans pascal
5   begin
6      //La partie donnée
7      write('Donner la valeur du Rayon :');
8      readln(Ray);
9
10     //La partie traitement
11     Surf:=Ray*Ray*Pi;
12     Circ:=2*Ray*Pi;
13
14     //La partie affichage des résultats
15     Writeln('la valeur de la Surface est :',Surf);
16     Writeln('la valeur de la circonférence est :',Circ);
17  end.
```

3- Après l'exécution

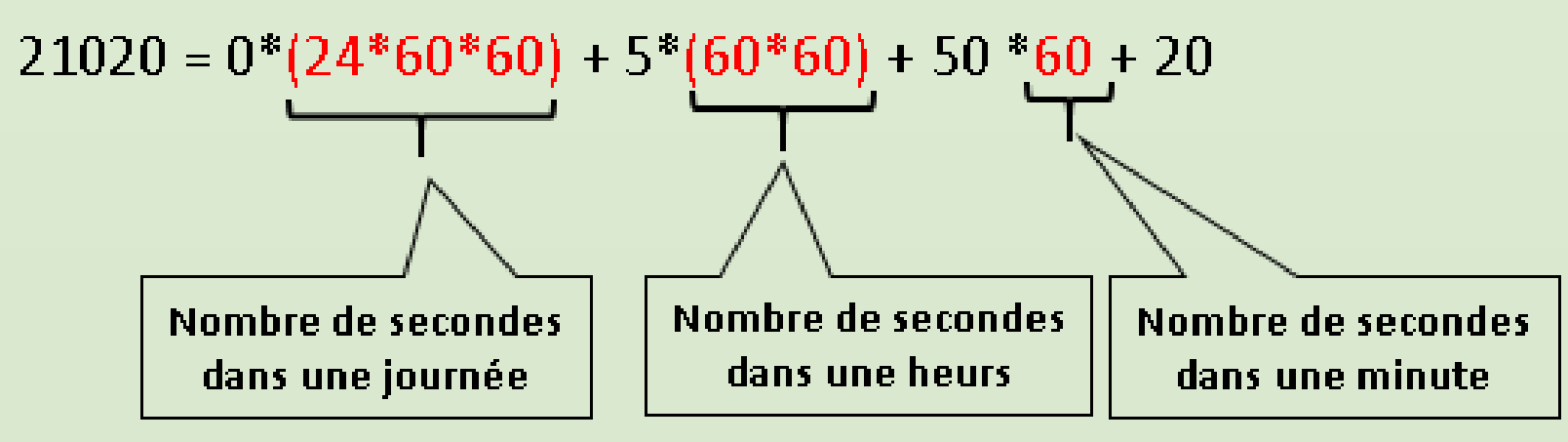
```
MyPasccal V1.16.11 (Exécution) C:\Users\ly\Desktop\ws_pasca\exo2.exe
Donner la valeur du Rayon :7.9
la valeur de la Surface est : 1.96066797510539O0E+002
la valeur de la circonférence est : 4.96371639267187 35E+001
```

TD (devoir maison)

Exercice N°3 : Écrire **l'algorithme et le program pascal** qui effectue la lecture du temps t en seconde, et il affiche le temps t en jours, heure, minutes, secondes.

Exemple :

si t=21020 secondes l'algorithme affichera 0 jours, 5 heures, 50 minutes et 20 secondes.



Chapitre 4

Les Instructions Conditionnelles

4.1. La structure conditionnelle.

4.2. La structure alternative.

4.3. Imbrication de "Si"

4. Les Instructions Conditionnelles

La structure conditionnelle.

➢Les instructions conditionnelles permettre à la machine de "choisir" les exécutions suivant les valeurs des données.

Si <condition> Alors

instructions;

FinSi

Exemple :

```
Algorithme positif ;
Variable :
        A: entier;
Début
        Lire(A);

        Si A > 0 Alors
            Ecrire(" A est positif ");
        FinSi
Fin.
```

4. Les Instructions Conditionnelles

La structure conditionnelle.

Si <condition> Alors

instructions;

Sinon

instructions;

FinSi

Exemple :

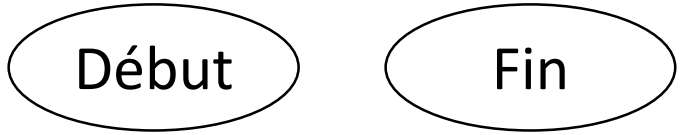
```
Algorithme positif ;
Variable :
        A: entier;
Début
        Lire(A);

        Si A > 0 Alors
            Ecrire(" A est positif ");
        Sinon
            Ecrire(" A est négatif ");
        FinSi
Fin.
```

4. Les Instructions Conditionnelles

L'Organigramme:

Un organigramme est une autre représentation des algorithmes, on utilise les formes géométriques suivantes:

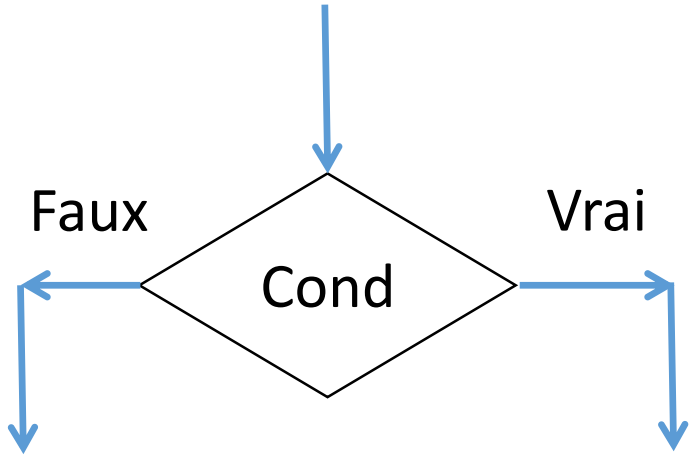
Le début et la fin d'un algorithme



Entrer des données par clavier

Afficher les Données sur Ecran

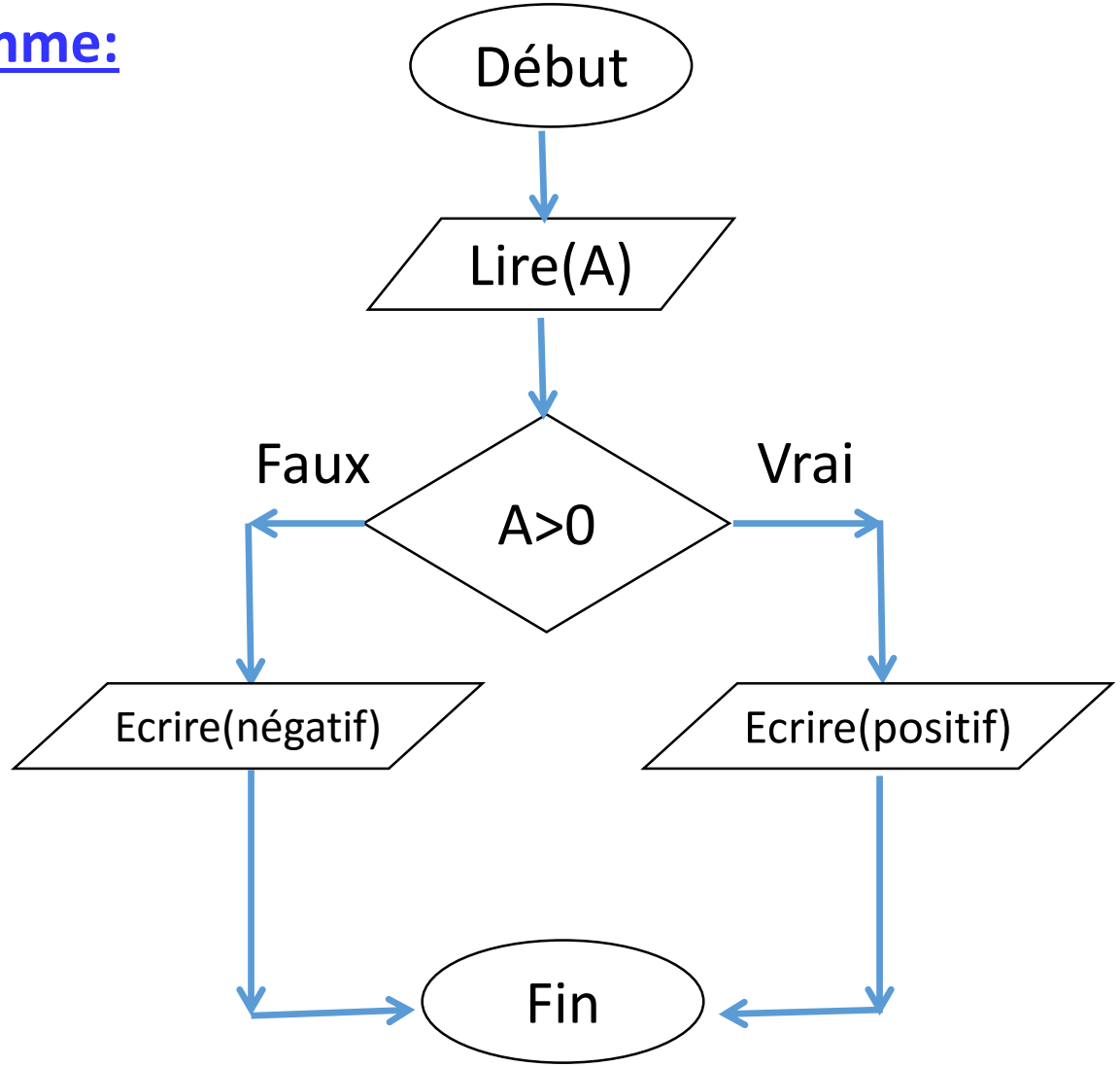
Faire une instruction ou action



Vérifier une condition

4. Les Instructions Conditionnelles

L'Organigramme:



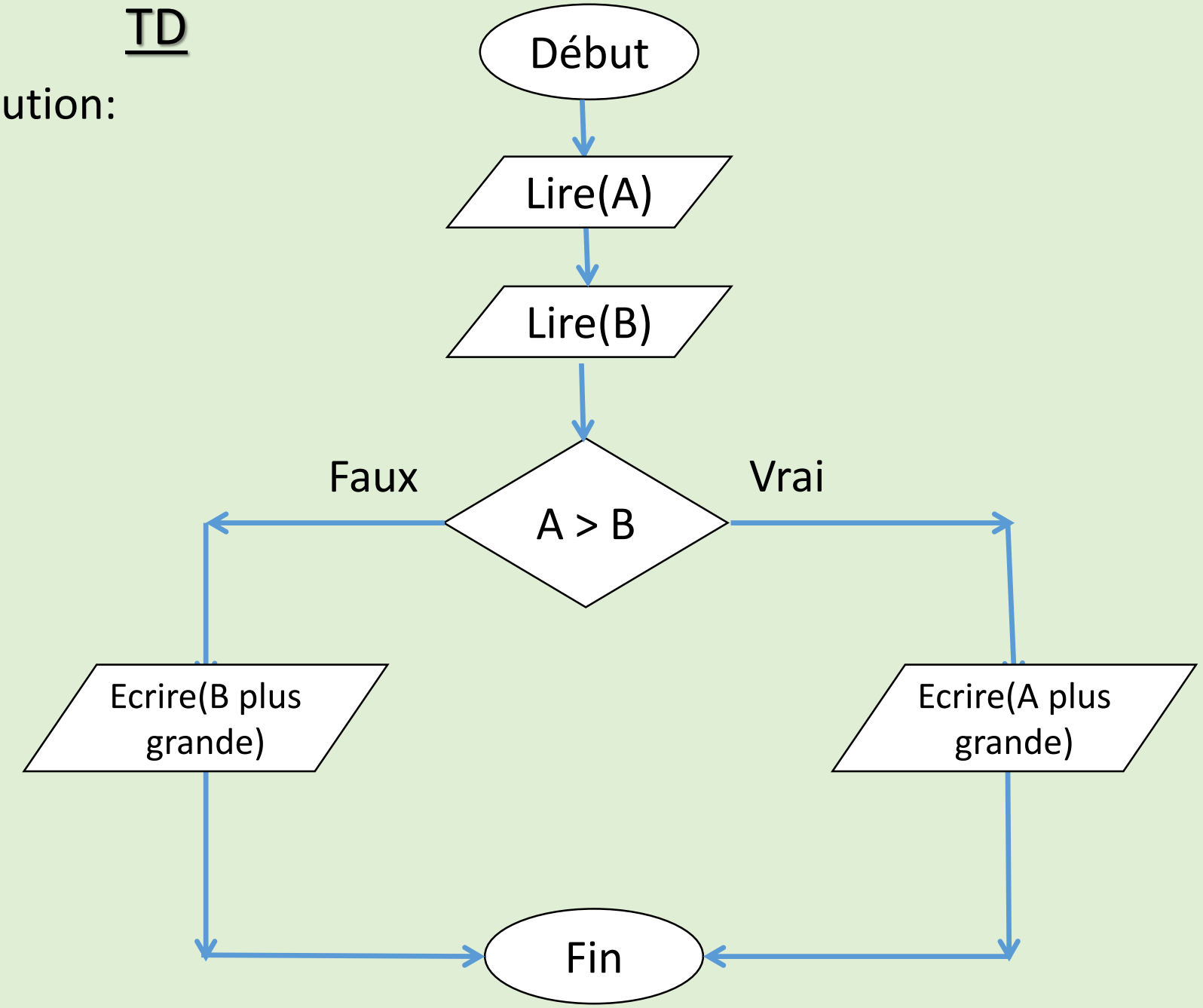

```
Algorithme signe;
Variable :
        A: entier;
Début
        Lire(A);

        Si A > 0 Alors
            Ecrire(" A est positif ");
        Sinon
            Ecrire(" A est négatif ");
        FinSi
Fin.
```

TD

Exercice N°4 : Ecrivez un algorithme qui demande à l'utilisateur d'entrer deux valeurs A et B et afficher la plus grande.

TD

Exercice N°4 : La solution:



TD

Exercice N°4 : La solution:

```
Algorithme plus_grande;
Variable :
        A,B: entier;
Début
    Lire(A);
    Lire(B);

    si A > B alors
        Ecrire (" A est plus grande");
     sinon
        Ecrire (" B est plus grande");
    finSi
Fin.
```

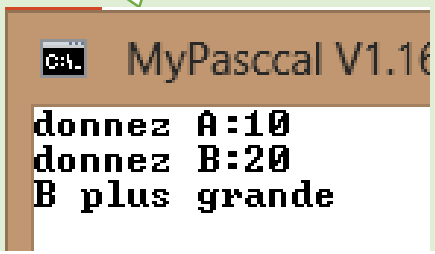
1- L'Algorithme

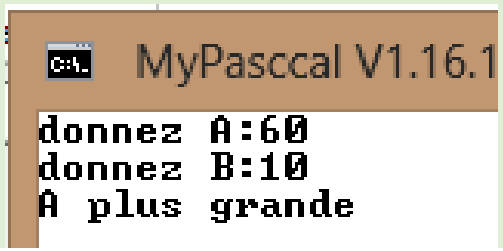
2- Le programme pascal saisie avec l'éditeur « My Pascal »

*plus_grande.pas

```
1   program grande;
2   var A,B:integer;
3   begin
4       write('donnez A:');
5       readln(A);
6       write('donnez B:');
7       readln(B);
8
9       if A > B then
10          writeln('A plus grande')
11      else
12          writeln('B plus grande')
13  end.
```

3- Après l'exécution





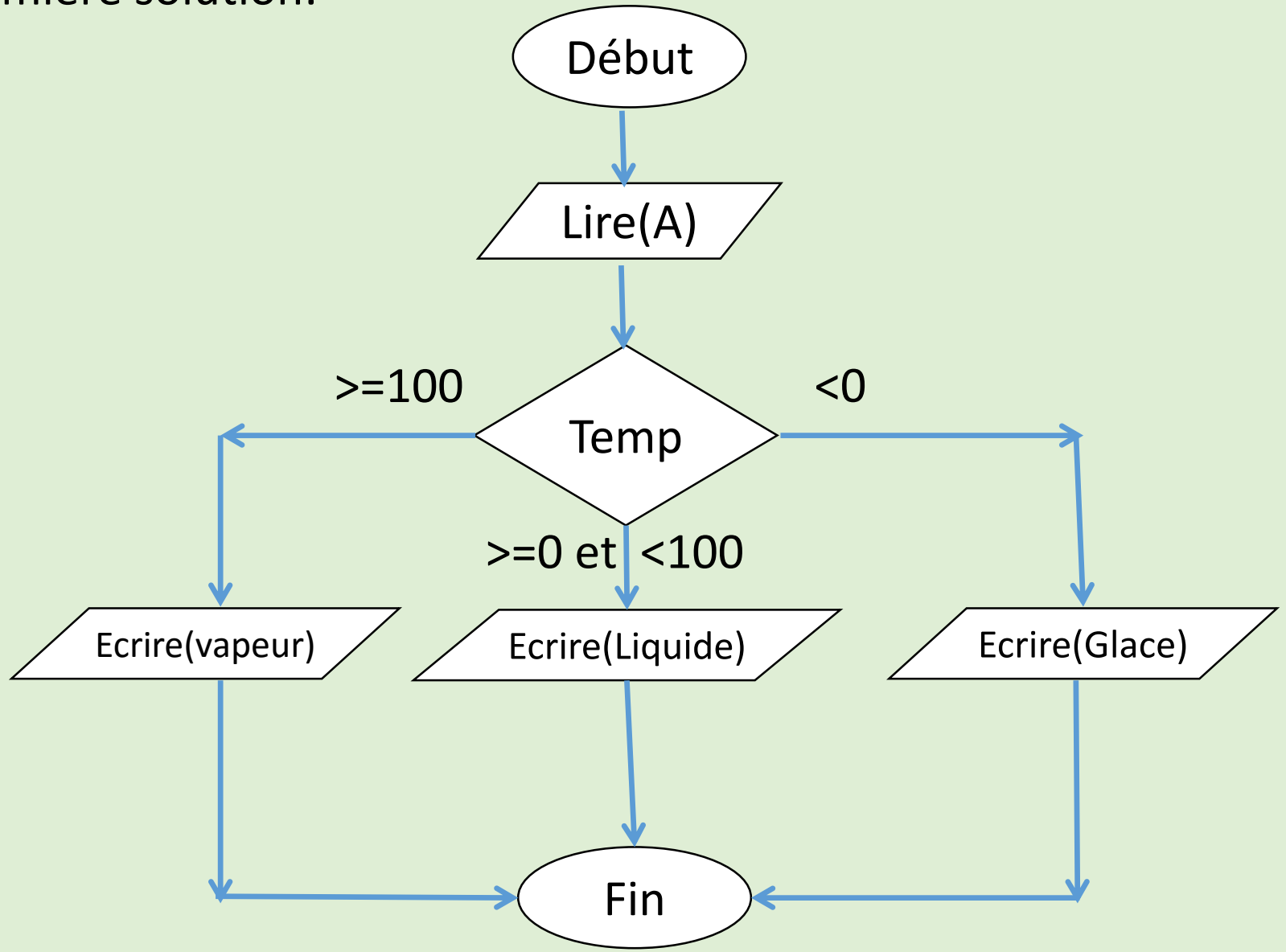
TD

Exercice N°5 : Ecrivez un algorithme qui demande à l'utilisateur d'entrer la valeur de la température (temp) de l'eau.

- Si temp < 0 → affiche « C'est de la glace »
- Si temp > 0 et < 100 → affiche « C'est du liquide »
- Si temp > 100 → affiche « C'est de la vapeur »

TD

Exercice N°5 : La première solution:



TD

Exercice N°5 : La première solution:

Algorithme température ;
Variable :
temp: **entier**;
Début
ecrire(" Donnez Temp :");
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.

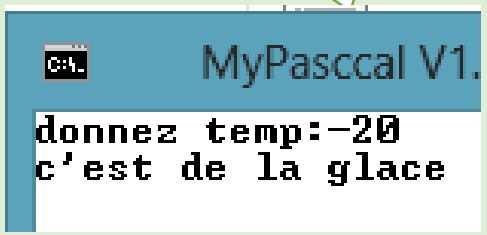
1- L'Algorithme

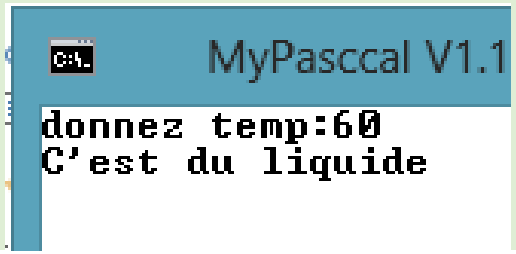
2- Le programme pascal saisie avec l'éditeur « My Pascal »

temp.pas

```
1   program temperature;
2   var temp:integer;
3   begin
4
5     write('donnez temp:');
6     readln(temp);
7
8     if temp < 0 then
9       write('c''est de la glace');
10
11    if (temp >= 0) and (temp<100) then
12      write('C'est du liquide ');
13
14    if temp >= 100 then
15      write('C'est de la vapeur ')
16
17  end.
```

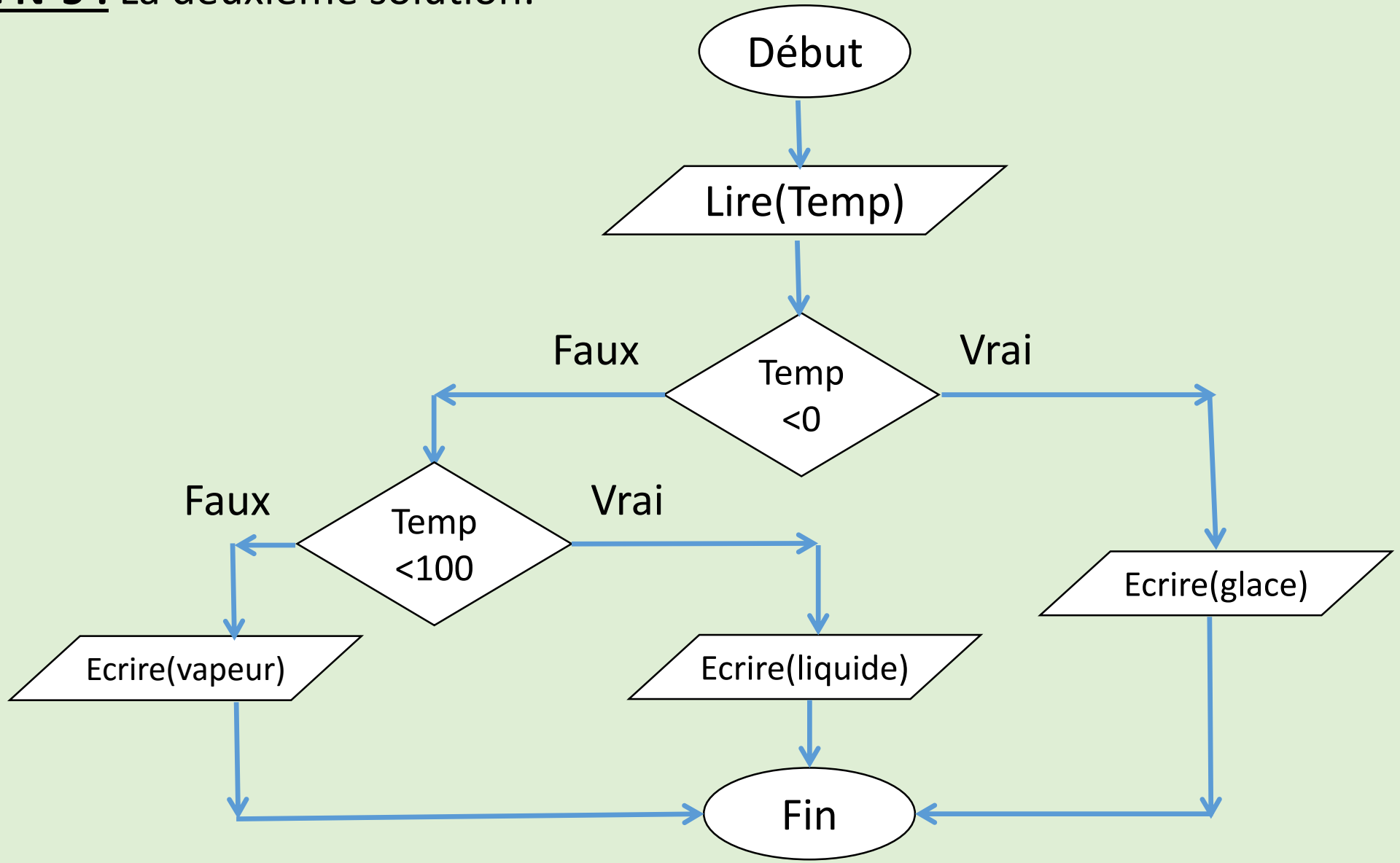
3- Après l'exécution





TD

Exercice N°5 : La deuxième solution:



TD

Exercice N°5 : La deuxième solution:

```
Algorithme température ;
Variable :
            temp: entier;
Début
      ecrire(" Donnez Temp :" );
       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.
```

1- L'Algorithme

2- Le programme pascal saisie avec l'éditeur « My Pascal »

temp2.pas

```
1   program temperature;
2   var temp:integer;
3   begin
4
5     write('donnez temp:');
6     readln(temp);
7
8     if temp < 0 then
9       writeln('c''est de la glace')
10    else if temp<100 then
11          writeln('C'est du liquide ')
12        else
13          writeln('C'est de la vapeur ')
14  end.
```

TD

Exercice N°6 : Ecrivez un algorithme qui donne le maximum de trois nombres saisis au clavier.

TD

Exercice N°7 : Ecrivez un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif, positif ou nul

attention : on ne doit pas calculer le produit des deux nombres.

TD

Exercice N°8 : Écrivez un algorithme qui permet de discerner une mention à un étudiant selon la moyenne de ses notes :

- **"Très bien"** pour une moyenne comprise entre **16 et 20**
- **"Bien"** pour une moyenne comprise entre **14 et 16**
- **"Assez bien"** pour une moyenne comprise entre **12 et 14**
- **"Passable"** pour une moyenne comprise entre **10 et 12**

TD

Exercice N°9 : Écrivez un algorithme qui permet de résoudre une équation du second degré

($ax^2 + bx + c = 0$ avec $a \neq 0$)

Chapitre 5

Les Structures Itératives

5.1. L'instruction "Tant que"

5.2. L'instruction "Répéter"

5.3. L'instruction "Pour"

4. Les Structures Itératives

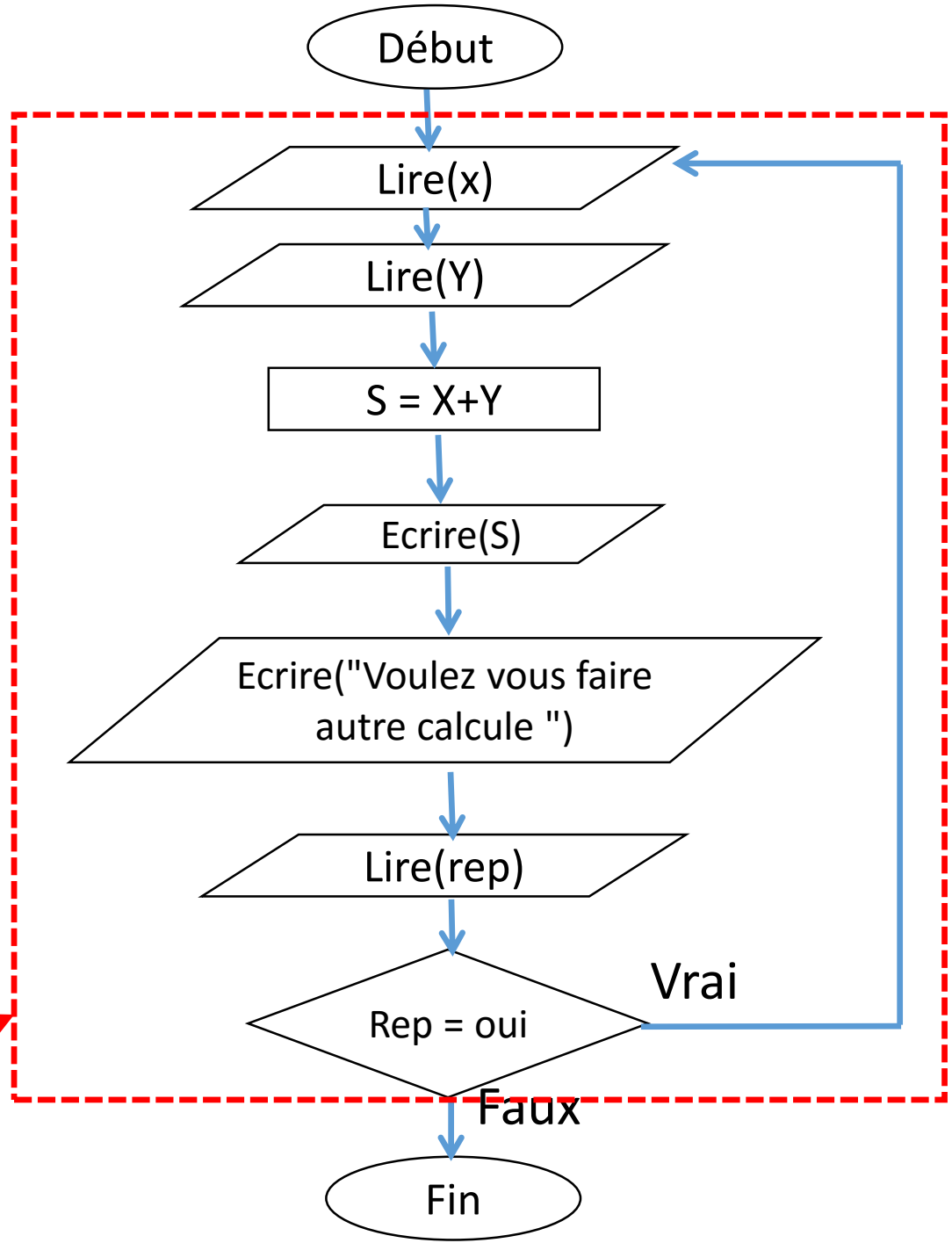
Exercice : Voici l'organigramme suivant :

- Donner le déroulement de cet organigramme pour :

rep = "oui" ensuite **"oui"** ensuite **"non"**

et **x** et **y** des valeurs **quelconques**.

- Que fait cet algorithme ?



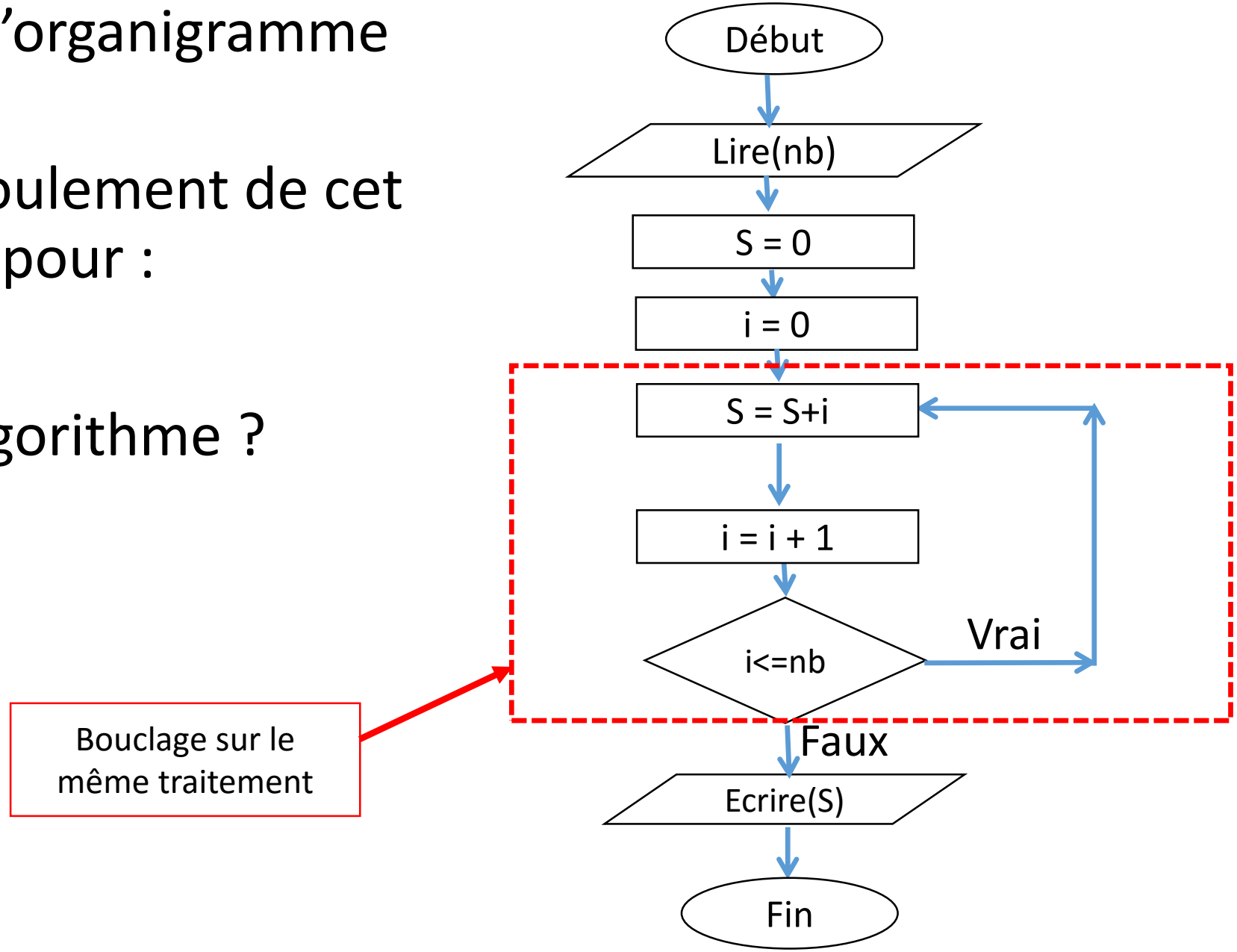
Bouclage sur le même traitement

Exercice : Voici l'organigramme suivant :

- Donner le déroulement de cet organigramme pour :

nb = 6 et nb = 8

- Que fait cet algorithme ?



4. Les Structures Itératives

La notion **d'itération (boucle)** est une notions fondamentales de l'algorithmique

On utilise les boucles quand on doit exécuter **le même traitement plusieurs fois**

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

1. **TantQue**
2. **Répéter … jusqu'à …**
3. **Pour … jusqu'à …**

4. Les Structures Itératives

La boucle « TantQue »

Le schéma de la boucle TantQue est :

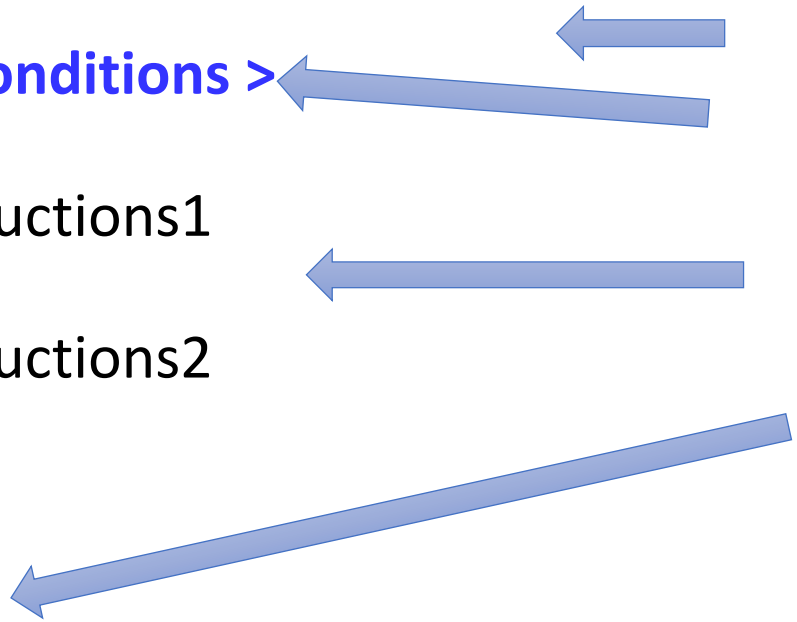
TantQue **<conditions >**

Instructions1

Instructions2

…

FinTantQue



Le principe est simple :

1. Le programme arrive sur la ligne du TantQue.
2. Il examine alors la valeur de la condition.
3. Si cette valeur est VRAI : le programme exécute les instructions qui suivent, jusqu'à ce qu'il rencontre la ligne FinTantQue.
4. Il retourne ensuite sur la ligne du TantQue, procède au même examen, et ainsi de suite.
5. On ne s'arrête que lorsque la condition prend la valeur FAUX.

4. Les Structures Itératives

La boucle « TantQue »

On veut écrire un algorithme qui affiche le message "Bonjour à tous" 10 fois utilisant la boucle TanQue.

```
Algorithme tantque;
Variable :
        i: entier;
Début
    i←1;
    TantQue(i<=100)
        Ecrire (" Bonjour à tous ");
        i←i+1;
    finTantQue
Fin.
```

while.pas

```
1  program boucle;
2  var i:integer;
3  begin
4     while i<=10 do
5     begin
6           writeln('while bonjour  :',i);
7           i:=i+1;
8  end;
9  end.
```

4. Les Structures Itératives

La boucle « Répéter … jusqu'à … »

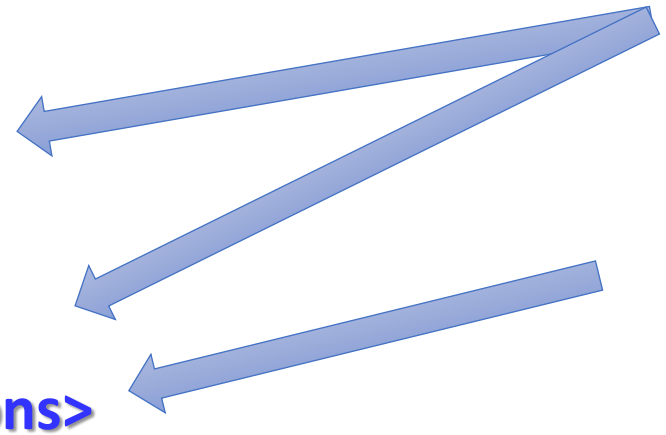
Répéter

Instruction1

Instruction2

…

jusqu'à <conditions>



1. Le programme exécute les instructions qui suivent **Répéter**, jusqu'à ce qu'il rencontre la ligne **jusqu'à**.
2. Il examine la valeur de la **condition**.
3. On s'arrête si la **condition** prend la valeur VRAI.
4. Sinon il retourne sur la ligne du **répéter** et ainsi de suite.

4. Les Structures Itératives

La boucle « Répéter … jusqu'à … »

On veut écrire un algorithme qui affiche le message "Bonjour à tous" 10 fois utilisant la boucle Répéter justqu'à.

```
Algorithme Répéter;
Variable :
        i: entier;
Début
    i=1;
    Répéter
        Ecrire (" C'est du liquide ");
        i:=i+1;
    Jusqu'à (i>100)
Fin.
```

repeat.pas

```
1  program boucle;
2  var i:integer;
3  begin
4     i:=0;
5     repeat
6           writeln('repeat bonjour  :',i);
7           i:=i+1;
8      until i>=10
9  end.
```

4. Les Structures Itératives

La boucle « Pour … jusqu'à … »

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

    Instruction1

    Instruction2

    …

FinPour

1. on initialise i par début
2. on test si on a pas dépassé fin
3. on exécute les instructions
4. on incrémente i (i ← i + 1)
5. on test si on a pas dépassé fin
6. etc.

4. Les Structures Itératives

La boucle « Pour … jusqu'à … »

Écrire un algorithme qui affiche le message "Bonjour à tous" 10 fois utilisant la boucle Pour.

Algorithme pour;
Variable :
    i: **entier**;
Début
    Pour i allant de 1 jusqu'à 10
        Ecrire (" C'est du liquide ");
    FinPour
Fin.

*for.pas

```
1  program boucle;
2  var i:integer;
3  begin
4     for i:=0 to 10 do
5     begin
6           writeln('for bonjour  :',i);
7     end
8  end.
```

```
1  program boucle;
2  var i:integer;
3  begin
4     while i<=10 do
5      begin
6           writeln('while bonjour  :',i);
7           i:=i+1;
8  end;
9  end.
```

*for.pas

```
1  progr
2  var i:integer;
3  begin
4     for i:=0 to 10 do
5     begin
6           writeln('for bonjour  :',i);
7     end
8  end.
```

pas

```
1  program boucle;
2  var i:integer;
3  begin
4     i:=0;
5     repeat
6           writeln('repeat bonjour  :',i);
7           i:=i+1;
8     until i>=10
9  end.
```

TD

Exercice N°9 : Donner les algorithmes qui permettent de calculer les sommes suivantes :

1- $S = 1 + 2 + 3 + 4 + \ldots\ldots\ldots\ldots\ldots + n$; $S = \sum_{i=1}^{n} (i).$

2- $S = 1 - 2 + 3 - 4 + \ldots\ldots\ldots\ldots\ldots \pm n$

3- $S = 2 + 4 + 6 \ldots\ldots\ldots\ldots\ldots + n;$

4- $S = 1 + 3 + 5 + \ldots\ldots\ldots\ldots\ldots + n;$

5- $S = 0! + 1! + 2! + 3! + 4! + \ldots\ldots\ldots\ldots\ldots + n!$

6- $e = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots\ldots\ldots\ldots + \frac{x^n}{n!}$