ENSIAS

ALGORITHMIQUE

A. ETTALBI

A.U. : 2018-2019

ettalbi1000@gmail.com

Module M1.1

Intitulé : Algorithmique et Programmation

Responsable : A. ETTALBI

Volume horaire : 54 Heures

Période : Semestre S1 (1er Semestre de la 1ère Année)

Composition de M1.1

2 Eléments de Module :

- M1.1.1 : **Algorithmique** (28H)
- M1.1.2 : **Programmation** (26H)

Caractéristiques des Eléments de Module du Module M1.1

	Nom	**Responsable**	**Volume horaire**	**Coeff**
M1.1.1	Algorithmique	ETTALBI	Cours : 14H TD : 14H	1
M1.1.2	Programmation	NASSAR	Cours : 12H TP : 14H	1

ENSIAS

ALGORITHMIQUE

A. ETTALBI

A.U. : 2018-2019

ettalbi1000@gmail.com

Plan Général

I- Définitions

II- Objectifs de la programmation

III- Les langages de programmation

IV- Algorithme et Organigramme

V- Structure d'un Algorithme

VI- Structures de données

VII- Fonctions (Sous-Programmes)

Exercices-Corrections

I- DEFINITIONS

- **Informatique :**

Définition 1 :

Traitement automatique de ***l'information***

Définition 2 :

Science de ***l'ordinateur***

I- DEFINITIONS

- **Information :**

  Toute donnée ***brute*** qui peut être ***quantifiée***, ***stockée*** pour être ***traitée*** afin de donner un ***résultat***.

I- DEFINITIONS

- **Ordinateur :**

Machine qui permet de ***mémoriser*** une grande quantité d'information et faire des ***calculs*** de base sur ces informations très ***rapidement***.

Composants de l'ordinateur

- **Composants internes :**
  - Unité Arithmétique et Logique
  - Mémoires (RAM, ROM, …)
  - Entrées/Sorties

Composants de l'ordinateur

- **Composants externes ou Périphériques :**

Eléments externes à l'ordinateur

qui communiquent avec lui

Composants de l'ordinateur

- **3 types de Périphériques :**

- d'entrée (clavier, souris, scanner)

- de sortie (écran, imprimante)

- de stockage (disques, CD-ROM)

II- OBJECTIF DE LA PROGRAMMATION

- Résolution automatique (par l'ordinateur) des problèmes.
- En profitant de la rapidité et de la capacité de stockage de l'ordinateur.

II- OBJECTIF DE LA PROGRAMMATION

- **<u>Etapes en général à suivre :</u>**
  - Position du Problème
  - Modèle de résolution
  - Algorithme de résolution
  - Programme informatique

EXEMPLE

Exemple :

Calcul de la consommation d'un véhicule

Etape 1 : *Position du Problème*

- Consommation = Nombre de litres consommés en 100 km
- Données :
  - K1 : Kilométrage au départ
  - K2 : Kilométrage à l'arrivée
  - L1 : Nombre de litres au départ
  - L2 : Nombre de litres à l'arrivée

Etape 2 : *Modèle de Résolution*

K2-K1 ⟶ L1-L2

100 ⟶ X ?

- Données d'entrée : K1, K2, L1, L2
- Résultat à calculer (à chercher) : X
- Donc :

X=100 * (L1-L2) / (K2-K1)

Etape 3 : *Algorithme de Résolution*

Début :

Afficher("Donner K1,K2,L1,L2 : ")

Lire(K1,K2,L1,L2)

Si (K1=K2) Alors Afficher("Erreur ")

Sinon Calculer X←100*(L1-L2)/(K2-K1)

Afficher("la consommation est : ",X, "%")

FinSi

Fin.

Etape 4 : *Programme en langage C*

```
#include<stdio.h>
 main() {
  int K1,K2,L1,L2; float X;
  printf("Donner K1,K2,L1,L2 :");
  scanf("%d%d%d%d",&K1,&K2,&L1,&L2);
  if(K1==K2) printf("Erreur");
  else { X = 100 * (L1-L2) / (K2-K1);
          printf("La consommation est %f %",X);
        }
}
```

III- LES LANGAGES DE PROGRAMMATION

- **Définition :**

– Moyen de communication entre l'homme et la machine

– Ensembles de conventions pour assurer un dialogue bidirectionnel entre le développeur et l'ordinateur.

III- LES LANGAGES DE PROGRAMMATION

- **3 Niveaux :**
  - Langage machine (binaire)
  - Langages d'assemblage
  - Langages évolués

LANGAGE MACHINE

- Lié à la structure **électronique** interne de l'ordinateur,
- Composé de **bits** (0 et 1),
- **Jamais** utilisé par les programmeurs,
- **Seul** langage compris par la machine,
- Tout programme informatique doit être **traduit** vers ce langage.

LANGAGES D'ASSEMBLAGE (Langages de bas niveau)

- Propres à chaque machine (càd à chaque **processeur**),
- **Proches** du langage machine,
- **Loin** du langage naturel,
- Nécessitent bien-sûr un **traducteur**.

LANGAGES EVOLUES
(Langages de haut niveau)

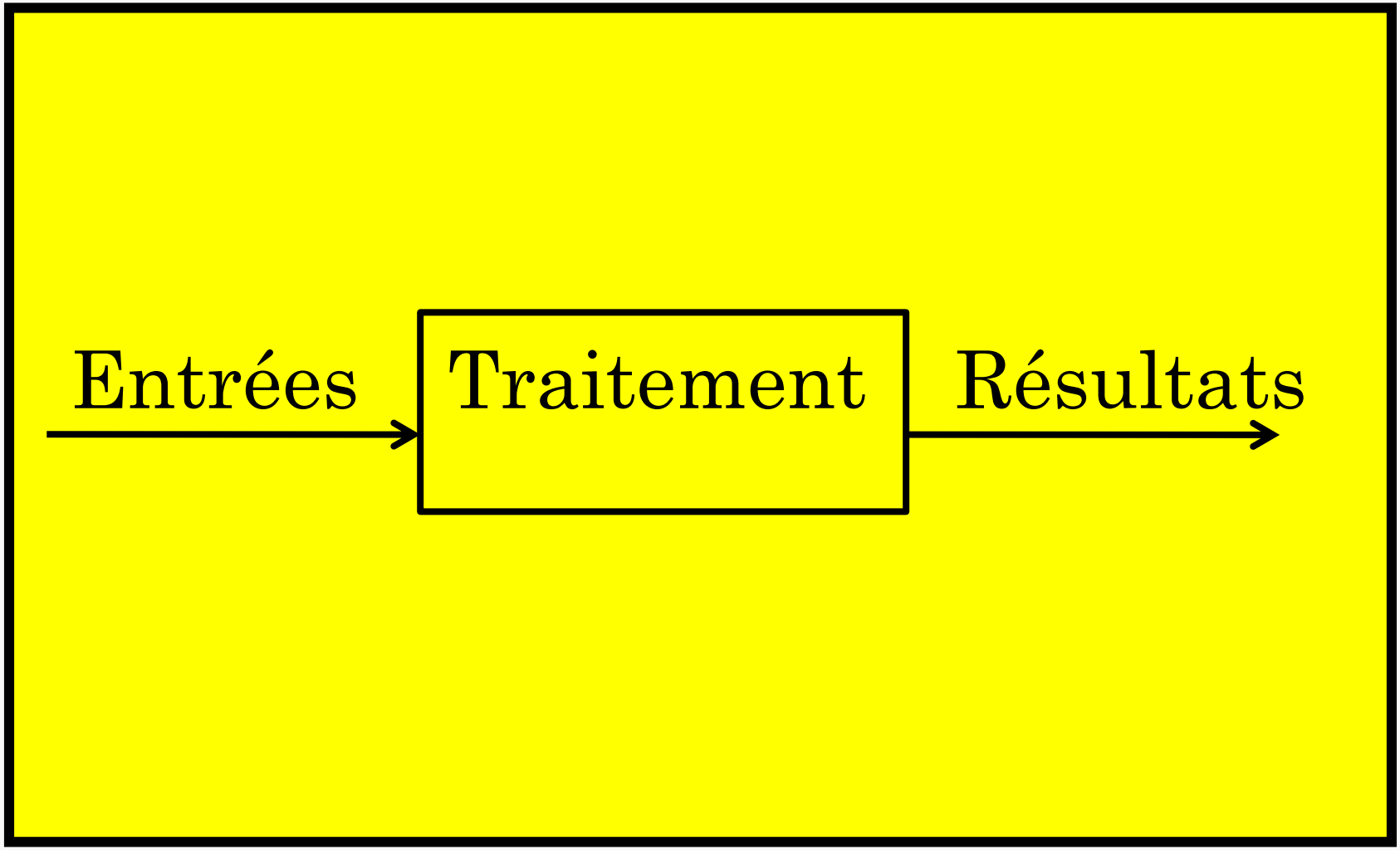
- **Proches** du langage naturel
- **Loin** du langage machine
- Nécessitent une traduction vers le langage machine par un ***compilateur*** ou un ***interpréteur***

Exemples : Pascal, C, JAVA, SQL

PROGRAMME INFORMATIQUE

- Ensemble d'instructions agissant sur des données en entrées pour donner des résultats en sortie.
- Les informations manipulées sont codifiées sous forme de variables, constantes, …

PROGRAMME INFORMATIQUE



PROGRAMME INFORMATIQUE

- **Variable :**

Objet informatique identifié par un "***identificateur***", d'un ***"type"*** donné, possédant une ***case mémoire*** et une ***valeur*** qui peut ***changer*** au cours de l'exécution du programme.

PROGRAMME INFORMATIQUE

- **Constante :**

Objet informatique identifié par un ***"identificateur"***, possédant une ***case mémoire*** et une ***valeur qui ne peut pas changer*** au cours de l'exécution du programme.

PROGRAMME INFORMATIQUE

- **Identificateur :**

Chaîne de caractères ***alphanumériques*** qui ***commence*** par un caractère alphabétique, qui ***ne contient pas*** d'espaces et pas de ***caractères spéciaux*** (é, è, à, ê, î, ...)

PROGRAMME INFORMATIQUE

Exemple d'identificateurs *corrects*:

A, a, age, etudiant, UnEtudiant

Exemple d'identificateurs *incorrects*:

à, âge, étudiant, 1Etudiant

PROGRAMME INFORMATIQUE

- **Type (Domaine) :**

Représente l'ensemble des ***valeurs possibles*** pour une variable et spécifie l'ensemble des ***opérations possibles*** sur cette variable.

PROGRAMME INFORMATIQUE

- **Types prédéfinis :** Entiers, Réels, Chaînes de caractères, Dates, …
- **Types définis par l'utilisateur :** Etudiants, Vecteurs à 3 dimensions, Matrices carrées d'ordre N, …

ETAPES D'UN PROGRAMME INFORMATIQUE

- **Edition** : écriture du programme,
- **Compilation** : détection des erreurs et traduction vers le langage machine,
- **Exécution** : qui ***"doit"*** donner les résultats attendus du programme.

IV-ALGORITHME ET ORGANIGRAMME

Algorithme :

- Description des ***étapes*** de résolution d'un problème.
- Constitué d'un ensemble d'***opérations*** élémentaires (afficher, lire, calculer, …)

IV-ALGORITHME ET ORGANIGRAMME

Organigramme :

- Description des étapes de résolution d'un problème,
- Constitué d'un ensemble de ***figures géométriques*** permettant de schématiser les opérations.

EXEMPLE
Calcul du Périmètre d'un Cercle

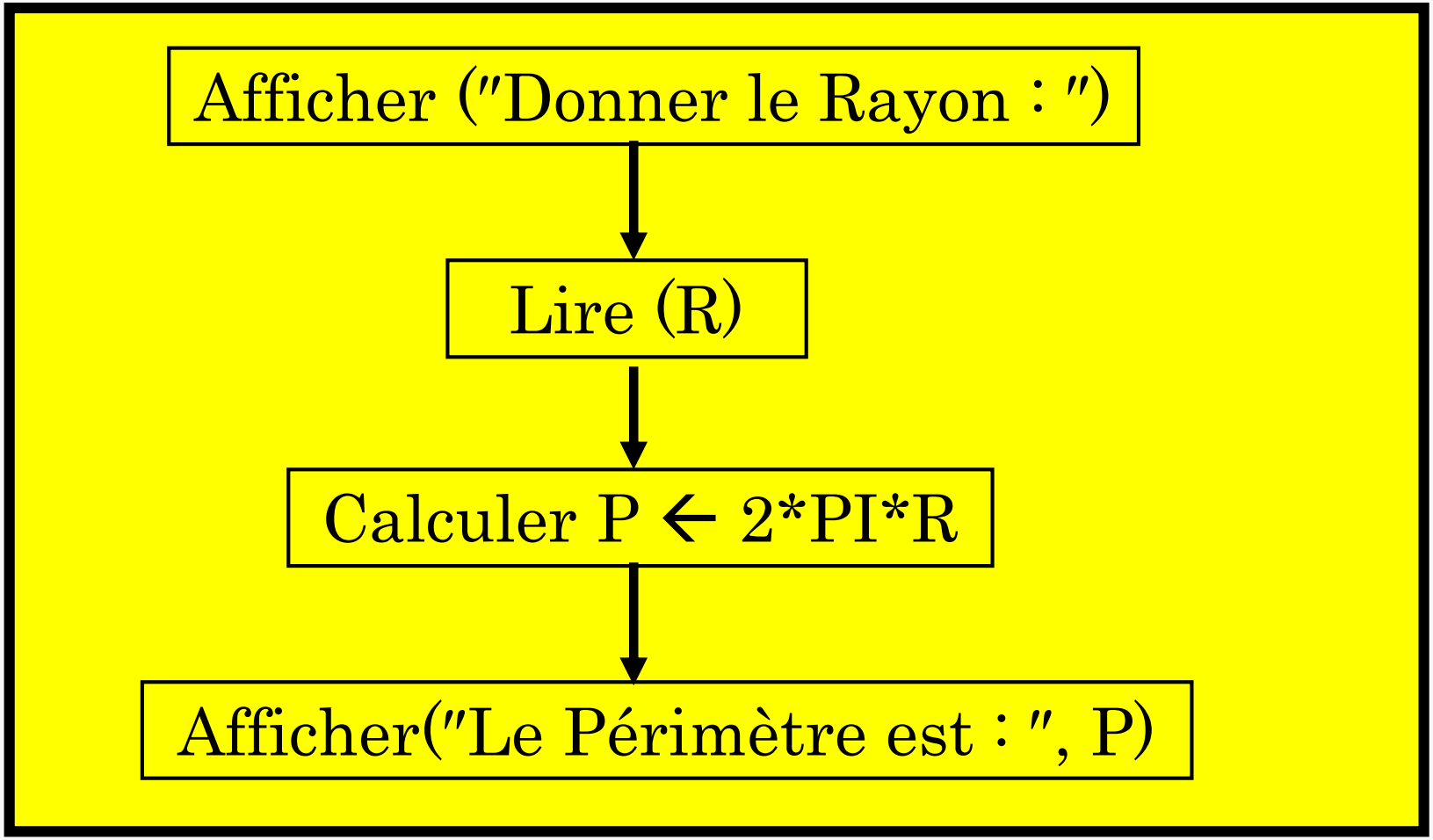
- **Données en entrées :**
  - R (Rayon)
  - PI (constante = 22/7)
- **Résultat à calculer :**
  - P (Périmètre)
- **Modèle de résolution :**

$$P = 2 * PI * R$$

EXEMPLE
Calcul du Périmètre d'un Cercle
Algorithme

```
Objets :
   PI : Constante=22/7
   P, R : variables réelles
Début :
    Afficher("Donner le Rayon : ")
    Lire(R)
    Calculer P ← 2*PI*R
    Afficher("Le Périmètre est : ", P)
Fin.
```

EXEMPLE
Calcul du Périmètre d'un Cercle
Organigramme



EXEMPLE
Calcul du Périmètre d'un Cercle
Programme en C

```
#include<stdio.h>
#define PI 3.14  /* constante */
 main()
{ float P, R;
  printf("Donner le Rayon :");
  scanf("%f",&R);
  P=2*PI*R;
  printf("Le Perimetre est %f", P);
}
```

EXERCICE D'APPLICATION

Calcul de la moyenne de deux nombres

1) Donner les objets en entrée et en sortie.

2) Donner le modèle de résolution.

3) Dresser l'algorithme.

4) Ecrire le programme en langage C.

V- STRUCTURE D'UN ALGORITHME

Types d'instructions :

- Instructions **séquentielles.**
- Instructions **alternatives.**
- Instructions **itératives.**

INSTRUCTIONS SEQUENTIELLES

- S'exécutent ***toutes***.
- S'exécutent l'une après l'autre

dans un ordre ***séquentiel***.

Exemples :

Lire, Afficher, Calculer

EXEMPLE

Calcul du périmètre et de la surface d'un rectangle (Algorithme)

```
Objets :
  Long , Larg, P, S  : variables réelles,
Début :
   Afficher("Donner la longueur et la largeur : ")
   Lire(Long, Larg)
   Calculer P← 2*(Long + Larg)
   Calculer S← Long * Larg
   Afficher("Le Périmètre est : ", P)
   Afficher("La Surface est : ", S)
Fin.
```

EXEMPLE

Calcul du périmètre et de la surface d'un rectangle (Programme)

```
#include<stdio.h>
main()  { float Long, Larg, P, S;
  printf("Donner la longueur et la largeur : ");
  scanf("%f%f", &Long, &Larg);
  P = 2*(Long + Larg);
  S = Long * Larg;
  printf("Le Périmètre est : %f", P);
  printf("\nLa Surface est : %f", S);
}
```

INSTRUCTIONS ALTERNATIVES

2 types :

- A deux alternatives :

  1 Choix (Chemin) parmi 2.

- A plusieurs alternatives :

  1 Choix parmi plusieurs.

INSTRUCTIONS A 2 ALTERNATIVES

- Se basent sur une ***condition***
- Contiennent 2 blocs d'instructions.
- Si la condition est vraie, on exécute le 1er bloc, si elle est fausse, on exécute le 2ème bloc.
- Le 2ème bloc est facultatif

INSTRUCTIONS A 2 ALTERNATIVES

- **Syntaxe générale**

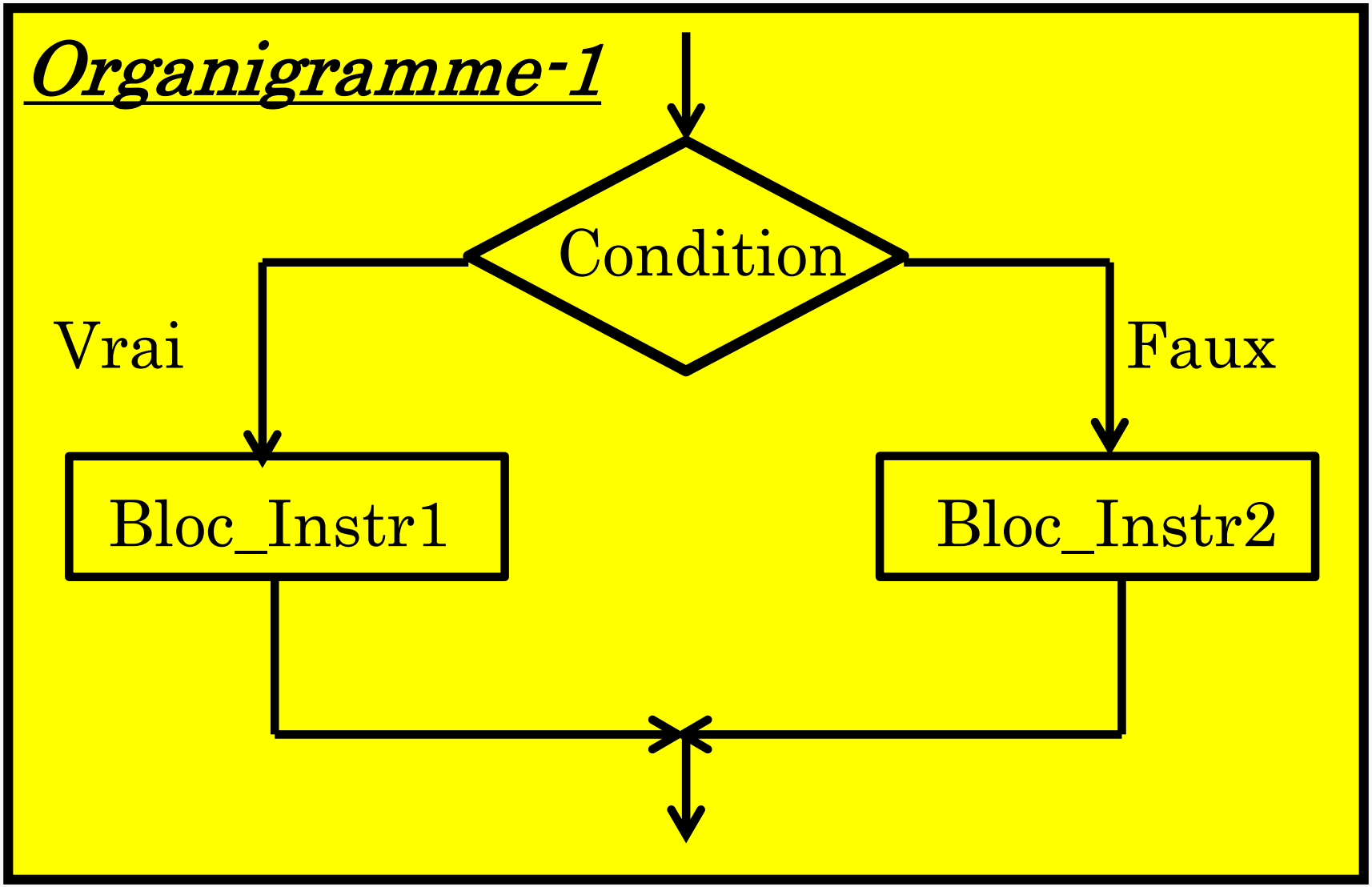
  Si Condition **alors** Bloc_Instructions1
  [Sinon Bloc_Instructions2**]**
  FinSi

Avec :

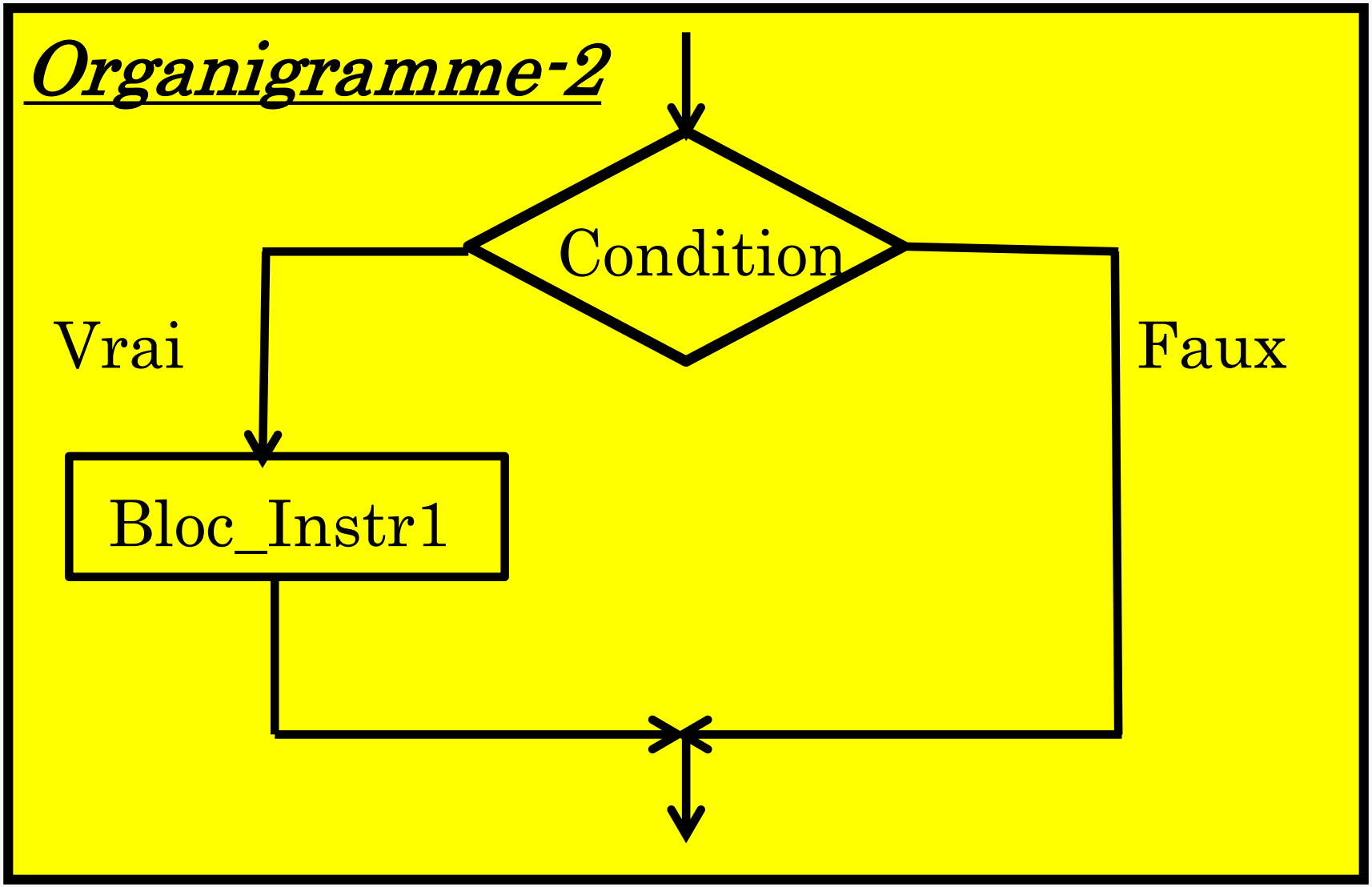
-**Condition** : expression booléenne (**V** ou **F**)

-**Bloc_Instructions1** et **Bloc_Instructions2** : peuvent être une instruction simple, des instructions séquentielles et/ou alternatives et/ou itératives

INSTRUCTIONS A 2 ALTERNATIVES



INSTRUCTIONS A 2 ALTERNATIVES



INSTRUCTIONS A 2 ALTERNATIVES

Exemple 1 :

Valeur absolue d'un entier a :

```
Si a>0 alors Abs←a
Sinon Abs←-a
FinSi
```

INSTRUCTIONS A 2 ALTERNATIVES

Exemple 2 :

Min, Max de 2 entiers a et b :

```
Si a>b alors Max←a
             Min←b
Sinon  Max←b
       Min←a
FinSi
```

Instructions alternatives imbriquées

Exemple 1 :

Lire 2 entiers a et b et afficher un message indiquant si a est supérieur strictement à b, b est supérieur strictement à a ou s'ils sont égaux.

Instructions alternatives imbriquées

Algorithme :

```
Objets : a, b : variables entières
Début
Afficher ("Donner 2 entiers : ")
Lire(a, b)
Si a>b alors Afficher(a, " > " , b)
Sinon  Si b>a  alors Afficher(a, " < " , b)
            Sinon Afficher(a, " = " , b)
            FinSi
FinSi
Fin.
```

Instructions alternatives imbriquées

Exemple 2 :

Lire 3 entiers a, b et c et afficher le maximum et le minimum entre eux en supposant qu'ils sont disjoints deux à deux.

Instructions alternatives imbriquées

Algorithme :

Objets : a, b, c : variables entières

Début :

```
Afficher("Donner 3 entiers disjoints : ")
Lire(a,b,c)
Si a>b alors  Si a>c alors Afficher("Max = " , a)
                Sinon Afficher("Max = " , c)
                FinSi
              Si b>c alors Afficher("Min = " , c)
              Sinon Afficher("Min = " , b)
              FinSi
```

Instructions alternatives imbriquées

Algorithme (Suite) :

```
Sinon  Si b>c alors Afficher("Max = " , b)
        Sinon Afficher("Max = " , c)
        FinSi
       Si a>c alors Afficher("Min = " , c)
       Sinon Afficher("Min = " , a)
       FinSi
FinSi
Fin.
```

EXERCICE D'APPLICATION
Affichage de la mention en fonction de la note

Ecrire un algorithme qui lit une note et affiche le message "admis avec mention" si cette note est supérieure à 14, "admis" si cette note est entre 12 et 14, "ajourné" si cette note est entre 10 et 12 et "exclus" si la note est inférieure à 10.

INSTRUCTIONS A PLUSIEURS ALTERNATIVES

- Se basent sur N conditions.
- Contiennent N Blocs d'instructions.
- Si la $i^{ème}$ condition est vraie, on exécute le $i^{ème}$ bloc d'instructions.

INSTRUCTIONS A PLUSIEURS ALTERNATIVES

- Syntaxe générale

```
Selon (variable)
    Si Val_1 : Bloc_Instructions1, Sortir
    Si Val_2 : Bloc_Instructions2, Sortir
    .........
    Si Val_N : Bloc_InstructionsN, Sortir
    [Si Autres : Bloc_Instructions]
FinSelon
```

INSTRUCTIONS A PLUSIEURS ALTERNATIVES

Exemple :

Tarifs d'un Zoo :

Code Visiteur	Type Visiteur	Tarif en DH
0	Enfant	10
1	Etudiant	12
2	Agé	15
3	Autres	25

INSTRUCTIONS A PLUSIEURS ALTERNATIVES

```
Algorithme :
Objets : Code, Tarif : entiers
Début :
Afficher ("Donner le code visiteur 0, 1, 2 ou 3 : ")
Lire(Code)
Selon (Code)
    Si 0 : Tarif←10, Sortir
    Si 1 : Tarif←12, Sortir
    Si 2 : Tarif←15, Sortir
    Si 3 : Tarif←25, Sortir
FinSelon
Afficher("Vous devez payer : ", Tarif, " DH")
Fin.
```

INSTRUCTIONS A PLUSIEURS ALTERNATIVES

Programme en C :

```
#include<stdio.h>
main()
{ int Code, Tarif;
  printf("Donner le type 0, 1, 2 ou 3 : ");
  scanf("%d",&Code);
  switch(Code)
   { case 0 : Tarif=10; break;
     case 1 : Tarif=12; break;
     case 2 : Tarif=15; break;
     case 3 : Tarif=25; break;
   }
printf("Vous devez payer %d DH", Tarif); }
```

Correction de l'exercice
Calcul de la moyenne de deux nombres

1) Donner les objets en entrée et en sortie.

2) Donner le modèle de résolution.

3) Dresser l'algorithme.

4) Ecrire le programme en langage C.

Correction de l'exercice
Calcul de la moyenne de deux nombres

1) Les objets en entrée :

a, b : variables entières

Les objets en sortie :

M : variable réelle.

2) Modèle de résolution :

M=(a+b)/2.0

Correction de l'exercice
Calcul de la moyenne de deux nombres

3) Algorithme:

Objets : a, b : variables entières

M : variable réelle

Début:

Afficher("Donner 2 entiers : ")

Lire(a, b)

Calculer M ← (a+b)/2.0

Afficher("Leur moyenne est : ", M)

Fin.

Correction de l'exercice
Calcul de la moyenne de deux nombres

4) Programme en langage C:

```
#include<stdio.h>
main()
{ int a, b; float M;
  printf("Donner 2 entiers :");
  scanf("%d%d", &a, &b);
  M=(a+b)/2.0;
  printf("Leur moyenne est %f", M);
}
```

EXERCICE D'APPLICATION
Affichage de la mention en fonction de la note

Ecrire un algorithme qui lit une note et affiche le message "admis avec mention" si cette note est supérieure à 14, "admis" si cette note est entre 12 et 14, "ajourné" si cette note est entre 10 et 12 et "exclus" si la note est inférieure à 10.

Correction de l'exercice *Affichage de la mention en Fonction de la note*

Algorithme:

Objets : N : variable réelle
Début :
Afficher("Donner la note :")
Lire(N)
Si N>=14 alors Afficher("Admis avec mention")
Sinon Si N>=12 alors Afficher("Admis")
Sinon Si N>=10 alors Afficher("Ajourné")
Sinon Afficher("exclus")
FinSi
FinSi
FinSi
Fin.

Correction de l'exercice *Affichage de la mention en Fonction de la note*

Programme en C :

```
#include<stdio.h>
main()
{ float N;
  printf("Donner la note :");
  scanf("%f", &N);
  if (N>=14) printf("Admis avec mention");
  else if (N>=12) printf("Admis");
        else if (N>=10) printf("Ajourné");
              else printf("exclus");
}
```

INSTRUCTIONS ITERATIVES (Boucles)

- Ecrites une seule fois.
- S'exécutent plusieurs fois.
- Sont caractérisées par :
  - La **condition de sortie** qui doit être valide (sinon boucle infinie),
  - Le **corps** de la boucle qui contient les instructions qui doivent se répéter.

INSTRUCTIONS ITERATIVES (Boucles)

- On distingue deux types :

– se basant sur le **nombre d'itérations** (On connait au départ le nombre de répétitions du corps de la boucle),

– se basant sur une **condition** (On ne connait pas le nombre de répétitions).

INSTRUCTIONS ITERATIVES
(se basant sur le nombre d'itérations)

- Utilisent une variable appelée compteur de boucle.
- Le compteur de boucle est une variable entière parcourant un intervalle [Min, Max] avec un pas donné.

INSTRUCTIONS ITERATIVES
(se basant sur le nombre d'itérations)

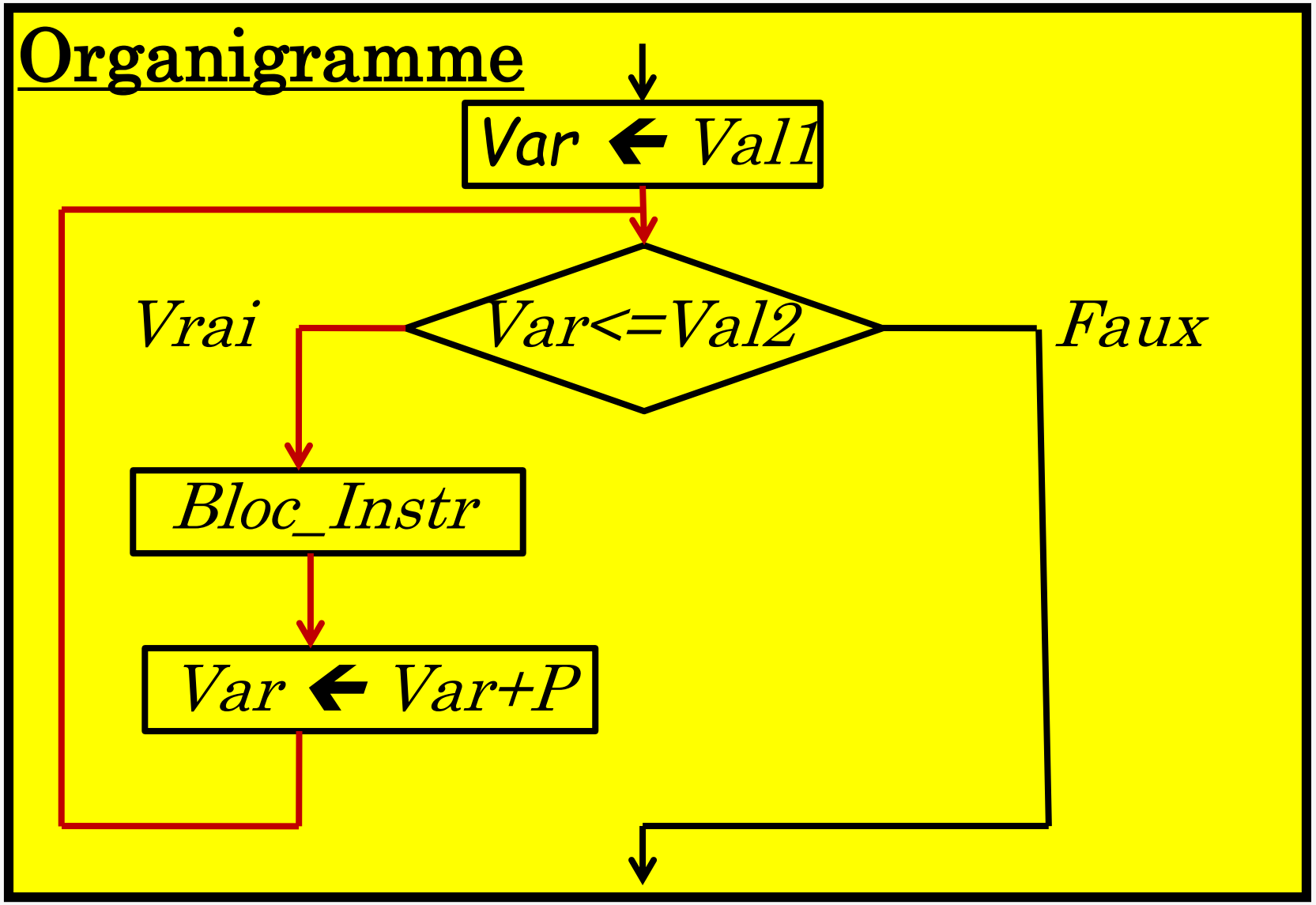
Syntaxe générale

Pour *Var ← Val1 jusqu'à Val2 [Pas=P] faire*
Bloc_Instructions
Fin-Pour

Avec :

- ***Val1***, ***Val2*** et ***P*** : des valeurs entières
- P : a par défaut la valeur 1
- ***Bloc_Instructions*** : peut être une instruction simple, des instructions séquentielles et/ou alternatives et/ou itératives.

INSTRUCTIONS ITERATIVES
(se basant sur le nombre d'itérations)



INSTRUCTIONS ITERATIVES
(se basant sur le nombre d'itérations)

Exemple 1:

Lire 10 notes et calculer leur moyenne.

Modèle :

Moyenne = la Somme des Notes divisée par le nombre de Notes

Exemple 1(Suite)

Variables à utiliser :

N : Note (réel),

MN : Moyenne des Notes (réel)

I : Compteur de la Boucle (entier)

Constante à utiliser :

NN=10 : Nombre de Notes

Exemple1 (Suite) : Algorithme

```
Objets : N, MN : variables réelles,
         I : variable entière,
         NN : constante entière=10,
Début :
MN←0
Pour I←1 jusqu'à NN faire
        Afficher("Donner une note :")
        Lire(N)
        MN←MN+N
Fin-Pour
MN←MN/NN
Afficher("La moyenne de ces notes est :", MN)
Fin.
```

Exemple 1(Suite) : Programme

```
#include<stdio.h>
#define NN 10  /* Constante */
main()
{ float N, MN; int I;
  MN=0;
  for (I=1 ; I<=NN ; I++)
        { printf("Donner une note :");
          scanf("%f",&N);  MN=MN+N;
        }
MN=MN/NN;
printf("La moyenne de ces notes est :%f", MN);
}
```

INSTRUCTIONS ITERATIVES
(se basant sur le nombre d'itérations)

Exemple 2:

Lire 2 entiers A et B et afficher les nombres compris strictement entre A et B ainsi que leurs carrés et leurs cubes.

Exemple 2(Suite)

Variables à utiliser :

A, B : Entiers à lire (**Entrées**)

I : Compteur de la boucle représentant aussi le $I^{ème}$ entier entre A et B

CA : Carré du $I^{ème}$ entier entre A et B

CU : Cube du $I^{ème}$ entier entre A et B

CA, CU, I : des **Sorties**

Exemple2 (Suite) : Algorithme

```
Objets : A, B, I, CA, CU : variables entières,
Début :
Afficher("Donner 2 entiers A et B avec A < B :")
Lire(A, B)
Pour I←A+1 jusqu'à B-1 faire
  Calculer CA ← I*I
  Calculer CU ← CA*I
  Afficher("L'entier est : ", I)
  Afficher("Son carré est : ", CA)
  Afficher("Son cube est : ", CU)
Fin-Pour
Fin.
```

Exemple 1(Suite) : Programme

```
#include<stdio.h>
main()
{ int A, B, I, CA, CU;
  printf("Donner 2 entiers A et B avec A < B :");
  scanf("%d%d", &A, &B);
  for (I=A+1 ; I<B ; I++)
        {CA = I*I; CU = CA*I;
         printf("L'entier est :%d", I)
         printf("\nSon carré est : %d", CA)
         printf("\nSon cube est : %d", CU)
        }
}
```

INSTRUCTIONS ITERATIVES
(se basant sur une condition)

- Se basent sur une condition (expression booléenne)
- N'utilisent pas de compteur mais l'utilisateur peut en définir en cas de besoin.

INSTRUCTIONS ITERATIVES
(se basant sur une condition)

2 types :

- On teste la condition **avant** d'entrer dans la boucle

  ⇒ Nombre d'itérations **>=** 0

- On exécute la boucle, **ensuite** on teste la condition

  ⇒ Nombre d'itérations > 0

INSTRUCTIONS ITERATIVES
se basant sur une condition

Syntaxe-1 :

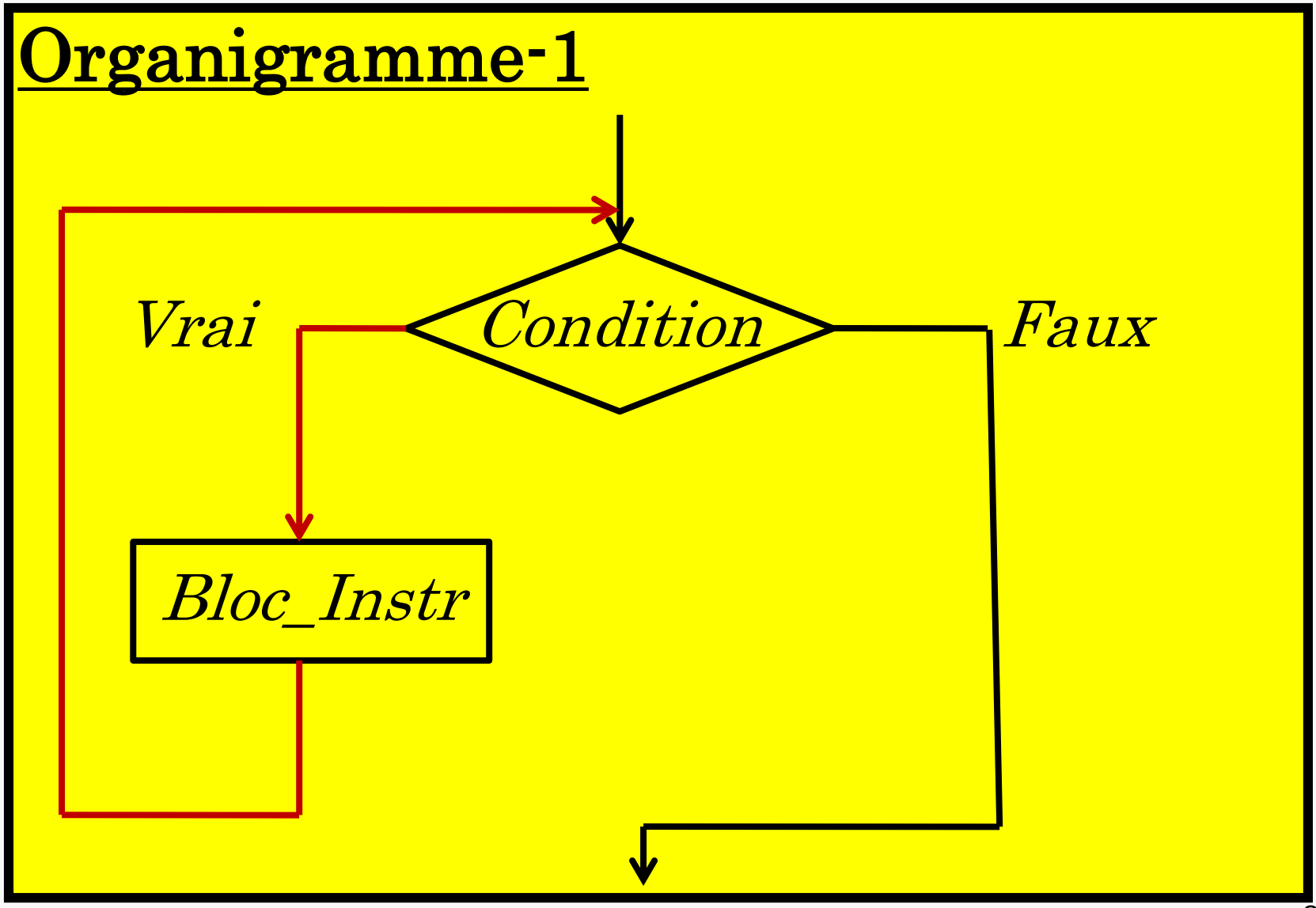
```
Tant que (Condition)
        Bloc_Instructions
Fin-Tant-que
```

Avec :

- **Condition** : expression booléenne
- **Bloc_Instructions** : peut être une instruction simple, des instructions séquentielles et/ou alternatives et/ou itératives.

INSTRUCTIONS ITERATIVES

se basant sur une condition



INSTRUCTIONS ITERATIVES
se basant sur une condition

Syntaxe-2 :

Répéter

Bloc_Instructions

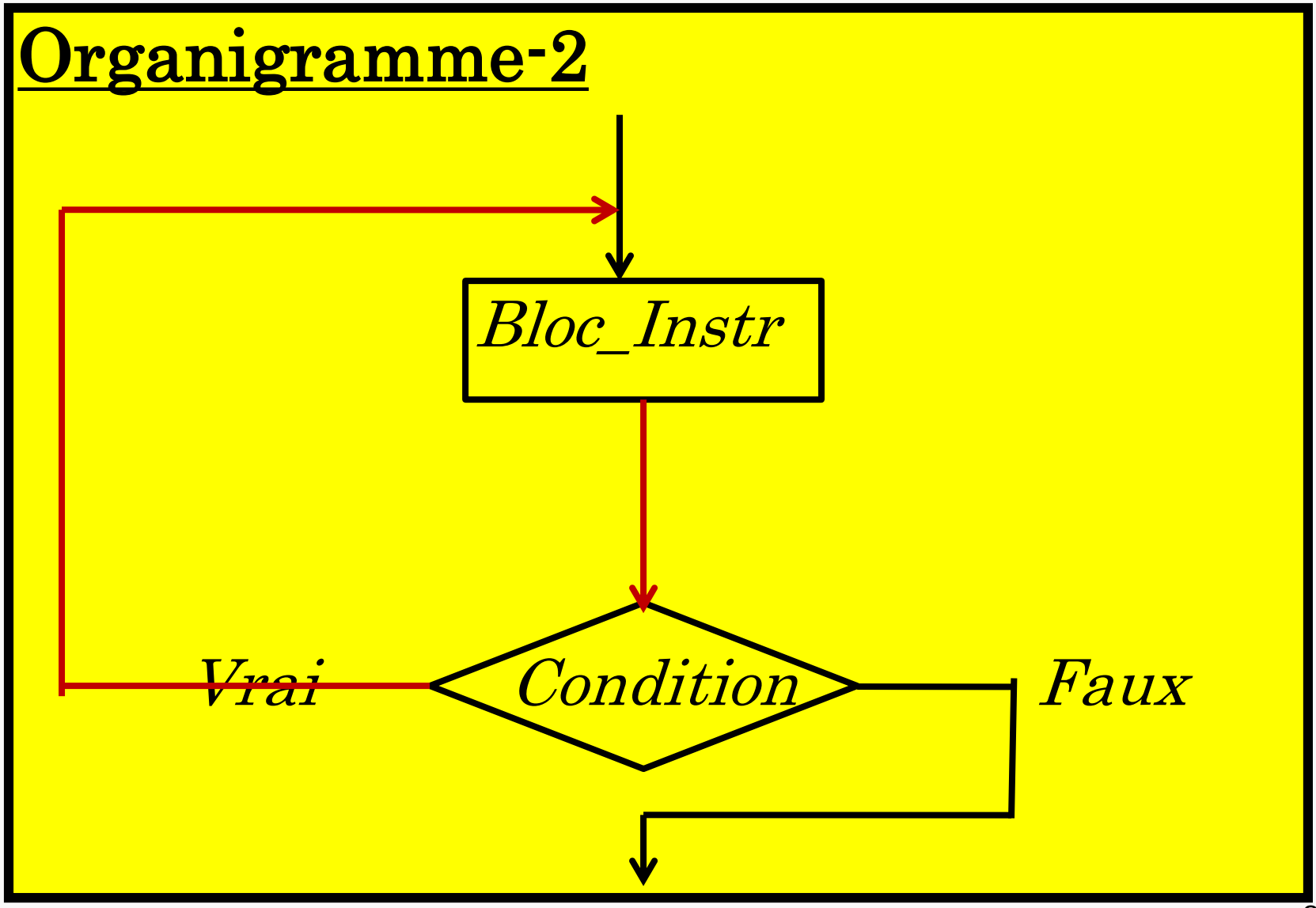
Tant que (Condition)

Avec :

- **Condition** : expression booléenne
- **Bloc_Instructions** : peut être une instruction simple, des instructions séquentielles et/ou alternatives et/ou itératives.

INSTRUCTIONS ITERATIVES

se basant sur une condition



INSTRUCTIONS ITERATIVES
se basant sur une condition

Exemple 1:

- Lire des entiers jusqu'à ce qu'on donne 0 puis afficher le nombre d'entiers lus ainsi que leur somme.
- Besoin d'un compteur : entier initialisé à 0 et incrémenté au besoin.

*Exemple1 (Suite) : Algorithme Avec **Tant-Que***

```
Objets : E, NE, SE : variables entières
Début :
        SE←0; NE←0; E←1
Tant que (E≠0) faire
   Afficher("Donner un entier et 0 pour sortir :")
   Lire(E)
   Calculer SE←SE+E
   Si E≠0 alors Calculer NE←NE+1
   FinSi
Fin-Tant-que
  Afficher("Le nombre d'entiers lus est :", NE)
  Afficher("Leur somme est :", SE)
Fin.
```

Exemple1 (Suite) : Programme en C

```
#include<stdio.h>
main()
{ int E, SE, NE;  SE=0; NE=0; E=1;
  while (E!=0)
   {printf("Donner un entier et 0 pour sortir :");
    scanf("%d", &E);
    SE=SE+E;
    if (E!=0) NE=NE+1;
    }
  printf("Le nombre d'entiers lus est : %d", NE);
  printf("\nLeur somme est :%d", SE);
}
```

*Exemple1 : Algorithme Avec **Répéter-Tant-Que***

```
Objets : E, NE, SE : variables entières,
Début :
SE←0; NE←0   /* Pas besoin de E←1; */
Répéter
   Afficher("Donner un entier et 0 pour sortir :")
   Lire(E)
   Calculer SE←SE+E
   Si E≠0 alors Calculer NE←NE+1 FinSi
Tant que (E≠0)
Afficher("Le nombre d'entiers lus est :", NE)
Afficher("Leur somme est :", SE)
Fin.
```

Exemple1 : Programme en C

```
#include<stdio.h>
main()
{ int E, SE, NE;  SE=0; NE=0;
  do {
        printf("Donner un entier et 0 pour sortir :");
        scanf("%d", &E);
        SE=SE+E;
        if (E!=0) NE=NE+1;
     }
  while (E!=0);
  printf("Le nombre d'entiers lus est : %d", NE);
  printf("\nLeur somme est :%d", SE);
}
```

INSTRUCTIONS ITERATIVES
se basant sur une condition

Exemple 2:

- Lire les Prix Unitaires et les Quantités de produits achetés et afficher le total à payer.
- On boucle tant que l'utilisateur le désire.

*Exemple2 (Suite) : Algorithme Avec **Tant-Que***

```
Objets : PU, PT : variables réelles
         Qte, Choix : variables entières
Début :
PT←0; Choix←1
Tant que (Choix≠0) faire
  Afficher("Donner Le PU et la Qte achetée:")
  Lire(PU, Qte)
  Calculer PT←PT+(PU*Qte)
 Afficher("Taper 1 pour Continuer ou 0 pour sortir :")
  Lire(Choix)
Fin-Tant-que
Afficher("Le prix total à payer est :", PT, "DH")
Fin.
```

Exemple2 (Suite) : Programme en C

```
#include<stdio.h>
main()
{ float PU, PT; int Qte, Choix;
 PT=0; Choix=1;
 while (Choix!=0) {
  printf("Donner Le PU et la Qte achetée:");
  scanf("%f%d", &PU, &Qte);
  PT=PT+(PU*Qte);
 printf("Taper 1 pour Continuer ou 0 pour sortir :")
 scanf("%d", &Choix);
}
printf("Le prix total à payer est :%f DH", PT);
}
```

*Exemple2 : Algorithme Avec **Répéter-Tant-Que***

```
Objets : PU, PT : variables réelles;
         Qte, Choix : variables entières,
Début :
PT←0;
Répéter
 Afficher("Donner Le PU et la Qte achetée:")
  Lire(PU, Qte)
  Calculer PT←PT+(PU*Qte)
 Afficher("Taper 1 pour Continuer ou 0 pour sortir :")
  Lire(Choix)
Tant que (Choix≠0)
Afficher("Le prix total à payer est :", PT, "DH")
Fin.
```

Exemple2 : Programme en C

```
#include<stdio.h>
main()
{ float PU, PT; int Qte, Choix;
 PT=0;
 do {
  printf("Donner Le PU et la Qte achetée:");
  scanf("%f%d", &PU, &Qte);
  PT=PT+(PU*Qte);
 printf("Taper 1 pour Continuer ou 0 pour sortir :")
 scanf("%d", &Choix);
}
while (Choix!=0);
printf("Le prix total à payer est :%f DH", PT);
}
```

Exercice d'application N°1

Lister tous les nombres entiers pairs inférieurs ou égaux à un nombre N lu à partir du clavier.

- Utiliser les 3 boucles séparément
- Donner l'algorithme et le programme en C pour chaque cas.

Exercice d'application N°2

Lire un entier N et afficher sa table de multiplication.

- Utiliser les 3 boucles séparément.
- Donner l'algorithme et le programme en C pour chaque cas.

Attention aux erreurs logiques dans les boucles

- **Condition de sortie non valide** : on risque d'avoir une boucle infinie.
- **Corps de la boucle incorrect** : on risque de répéter des instructions qui ne doivent pas être répétées ou l'inverse.

Exemple 1 : Min et Max d'entiers

Algorithme

Objet : N, Min, Max : variables entières

Début:

Afficher("Donner un entier et 0 pour sortir :")

Lire(N)

Min←N; Max←N

Tant Que (N ≠ 0) faire ... Boucle Infinie Si !!...

Afficher("Donner un entier et 0 pour sortir :")

Si (Min > N) Min←N FinSi

Si (Max < N) Max←N FinSi

Fin-Tant-que

Afficher("Le Min est :", Min, " Le max est : ", Max)

Fin.

Exemple 2 : Norme d'un Vecteur

Algorithme

Objet : I, C : variable entière
X, N : variables réelles

Début:

Afficher("Donner la taille du Vecteur :")
Lire(C); N←0
Pour I←1 jusqu'à C faire
  Afficher("Donner la coordonnée :", I, " : ")
  Lire(X)
Fin-Pour
Calculer N←N+(X*X) ... N = ... !! ...
Afficher("La Norme est : ", RacineCarrée(N))
Fin.

Remarque

- Les 3 boucles sont équivalentes avec quelques modifications.
- On choisit la boucle au nombre d'itérations si on connait au départ combien de fois on va boucler.
- On utilise la boucle se basant sur une condition lorsque la sortie de la boucle est conditionnée par une condition et on ne connait pas au départ le nombre de répétitions.

Exemple : Somme des carrés des entiers inférieurs ou égaux à N

Algorithme avec la boucle Pour

Objet : N, S, I : variables entières
Début :
Afficher("Donner un entier :")
Lire(N); S←0
Pour I←1 jusqu'à N faire
    Calculer S←S+I*I
Fin-Pour
Afficher("La somme des carrés des entiers
            inférieurs ou égaux à ", N, " est : ", S)
Fin.

Exemple : Somme des carrés des entiers inférieurs ou égaux à N

Algorithme avec la boucle Tant-Que

Objet : N, S, I : variables entières
Début :
Afficher("Donner un entier :")
Lire(N); S←0; I←1
Tant Que (I<=N) faire
    Calculer S←S+I*I
    I←I+1
Fin-Tant-Que
Afficher("La somme des carrés des entiers
            inférieurs ou égaux à ", N, " est : ", S)
Fin.

Exemple : Somme des carrés des entiers inférieurs ou égaux à N

Algorithme avec la boucle Répéter

Objet : N, S, I : variables entières
Début :
Afficher("Donner un entier :")
Lire(N); S←0; I←1
Répéter
    Calculer S←S+I*I
    I←I+1
Tant Que (I<=N)
Afficher("La somme des carrés des entiers
                inférieurs ou égaux à ", N, " est : ", S)
Fin.

Les boucles imbriquées

- Le corps d'une boucle peut contenir une autre boucle
- On parle alors de boucles imbriquées
- Le système ne passe à l'itération suivante d'une boucle que lorsqu'il a terminé l'exécution de toute la boucle interne

Exemple 1: Affichage des nombres premiers inférieurs à 100

- Un entier N est premier si tous les entiers compris entre 2 et Racine(N) ne sont pas des diviseurs de N
- On a besoin d'une boucle pour parcourir l'intervalle [2, 100]
- Dans cette boucle, on aura besoin d'une autre boucle pour parcourir l'intervalle [2, Racine(N)]

Exemple 1: Affichage des nombres premiers inférieurs à 100

```
Objet : N, I, R : Variables entières
Début :
Pour N←2 jusqu'à 100 faire
        I←2;
        Calculer R←RacineCarrée(N)
        Tant que(N mod I ≠ 0 et I <= R) faire
                Calculer I←I+1
        Fin-Tant-que
        Si (I > R) alors
                Afficher(N, " est premier :")
        Fin-Si
Fin-Pour
Fin.
```

Exemple 2: Moyenne générale des étudiants

- On saisit les notes et les coefficients d'un étudiant et on calcule sa moyenne générale
- On répète cela tant que l'utilisateur le désire
- On aura donc besoin de 2 boucles : une pour parcourir les étudiants et l'autre pour saisir les notes et les coefficients.

Exemple 2: Moyenne générale des étudiants

```
Objet : N, I, R, C, SC : Variables entières
        No, SNC : variables réelles
Début :
Répéter
   Afficher("Donner le nombre d'examens :")
   Lire(N);    SC←0;    SNC←0
   Pour I←1 jusqu'à N faire
         Afficher("Donner la note et le coefficient:")
         Lire(No, C)
         Calculer SNC←SNC+(No*C)
         Calculer SC←SC+C
   Fin-Pour
   Afficher("La moyenne générale est :", (SNC/SC))
   Afficher("Taper 1 pour continuer:")
   Lire(R)
Tant que (R=1)
Fin.
```