Les composants d'un algorithme

Chaque algorithme possède des **variables d'entrée**, des **variables de sorties**, des **constantes** et une partie du **traitement** :

1 Variables d'entrée

Les variables d'entrée sont les variables **lues** (l'instruction **Lire**) ;

```
Lire (R) ;
```

2 Variables de sortie

Les variables de sorties sont les variables **affichées** (l'instruction **Écrire**) ;

```
Écrire (V) ;
```

3 Données intermédiaires

Les données intermédiaires qui peuvent être des variables ou des constantes (**Pi**)
qui sont des données **non lues et non écrites** utilisées pendant le traitement ;

```
Pi=3.14 ;
```

4 Traitement

Traitement : contient les instructions d'affectation, tests et les boucles.

affectation | tests | boucles

Exercice N°01 : Algorithme → Programme en langage C

Soit l'algorithme suivant :

Algorithme

```
Algorithme Exo1;
Constantes
    Pi=3.14 ;
Variables
    R,H,V : Réel;
Début
    //Entrées
    Écrire("Donner le rayon R : ");
    Lire(R);
    Écrire("Donner la hauteur H : ");
    Lire(H);

    //Traitement
    V ← Pi*R*R*H ;

    //Sortie
    Écrire("Volume = ", V);
Fin.
```

Questions

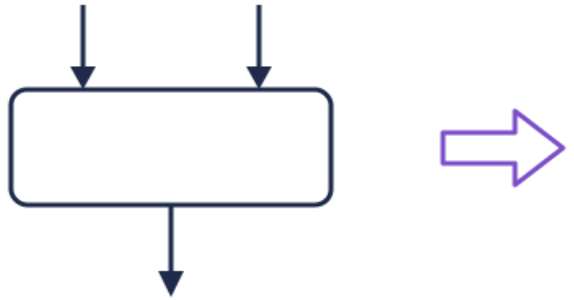
1- Traduire l'algorithme en **Programme C**.

2- Dérouler l'algorithme pour **R=2** et **H=6**

Instructions	Variables				Affichage
	V_1	V_2	...	V_n	
Instruction 1					
Instruction 2					
⋮					
Instruction N					

3- Déduire ce que fait l'algorithme/programme ?

4- Compléter le schéma suivant :



Entrées :

Traitement :

Sortie :

Exercice N°01 · Q1

Traduire l'algorithme en Programme C

Algorithme

```
Algorithme Exo1;
Constantes
    Pi=3.14 ;
Variables
    R,H,V : Réel;
Début
    //Entrées
    Écrire("Donner le rayon R : ");
    Lire(R);
    Écrire("Donner la hauteur H :");
    Lire(H);

    //Traitement
    V ← Pi*R*R*H ;

    //Sortie
    Écrire("Volume = ", V);
Fin.
```

⇄

Programme C

```
#include <stdio.h>
int main ()
{
    const float Pi=3.14;
    float R,H,V ;

    //Entrées
    printf("Donner le rayon R : \n") ;
    scanf("%f",&R) ;
    printf(" Donner la hauteur H : \n") ;
    scanf("%f",&H) ;

    //Traitement
    V = Pi*R*R*H ;

    //Sortie
    printf("Volume = %f ", V) ;
    return 0 ;
}
```

Définition

Le déroulement d'un algorithme (exécution manuelle)

Le **déroulement** (ou exécution manuelle) consiste à **exécuter l'algorithme à la main**, instruction par instruction, en suivant l'évolution des **variables** et de l'**affichage**.
Il sert à **comprendre**, **vérifier** et **corriger** un algorithme.

1 Choisir les entrées
Donner des valeurs de départ aux variables d'entrée.

2 Exécuter pas à pas
Une instruction à la fois, dans l'ordre, comme l'ordinateur.

3 Noter l'évolution
Valeurs des variables et affichage, dans un tableau.

4 Vérifier le résultat
Comparer le résultat obtenu avec celui attendu.

Exemple : a = 3 et b = 5

Instructions	Variables			Affichage
	a	b	d	
Lire(a, b) ; ← on saisit 3 et 5	3	5	/	
d ← a + b ; ← 3 + 5 = 8	3	5	**8**	
Écrire(d) ;	3	5	8	**8**

Exercice N°01 · Q2

Déroulement de l'algorithme pour R=2 et H=6

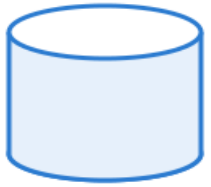
Instructions	Variables			Affichage
	R	H	V	
Écrire("Donner le rayon R :")	/	/	/	Donner le rayon R :
Lire (R)	2	/	/	
Écrire("Donner la hauteur H :")	2	/	/	Donner la hauteur H :
Lire (H)	2	6	/	
V ← Pi*R*R*H	2	6	75.360001	
Écrire ("Volume =", V) ;	2	6	75.360001	Volume = 75.360001

Exercice N°01 · Q3 et Q4

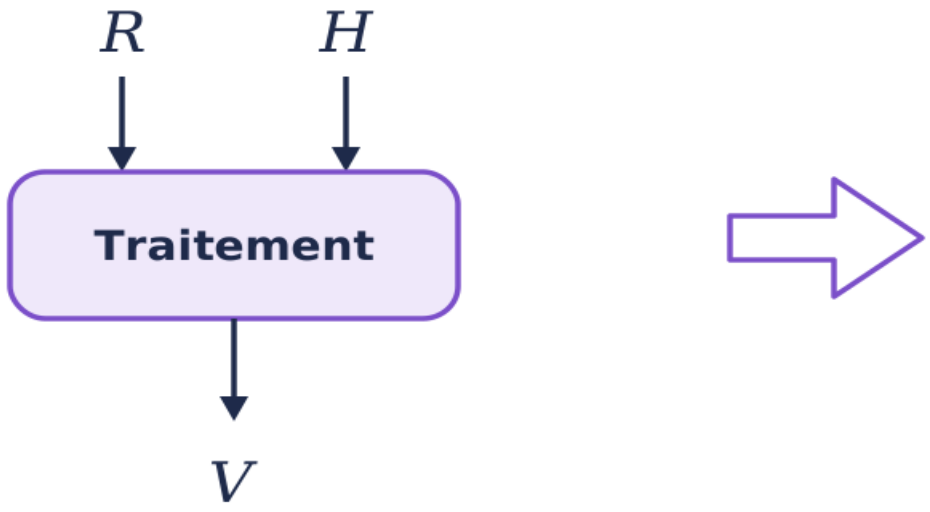
Que fait le programme ?

3) Déduire ce que fait le programme ?

Le programme calcule le **volume d'un cylindre**



4) Compléter le schéma suivant :



Entrées :

```
Lire (R) ;
Lire (H) ;
```

Traitements :

```
V ← Pi*R*R*H ;
```

Sortie :

```
Écrire (V) ;
```

Exercice N°02 : Algorithme → Programme

Soit l'algorithme suivant :

Algorithme

```
Algorithme exo1 ;
Variables
    a, b, c, d: entier ;
Début
    {-*-*- Entrées -*-*-}
    Écrire('Donner trois nombres entiers : ') ;
    Lire(a,b,c) ;
    {-*-*- Traitements -*-*-}
    Si (a<b) alors
        Si (a<c) alors
            d ← a
        Sinon
            d ← c ;
        Fin-Si
    Sinon
        Si (b<c) alors
            d ← b
        Sinon
            d ← c ;
        Fin-Si
    Fin-Si
    {-*-*- Sorties -*-*-}
    Écrire ('Le résultat = ', d) ;
Fin.
```

Questions

1. Traduire l'algorithme en un **programme** en **langage C**.
2. Que fait cet algorithme ?
3. **Dérouler** l'algorithme/Programme pour les cas suivants :
   - `a=1, b=2, c=4`
   - `a=4, b=2, c=0`
4. Donner l'**organigramme** de l'algorithme.

Corrigé

Exercice N°02 : Algorithme → Programme C

Algorithme

```
Algorithme exo1 ;
Variables
    a, b, c, d : entier ;
Début
    /* Entrées */
    Écrire('Donner trois nombres entiers : ') ;
    Lire(a,b,c) ;
    /* Traitements */
    Si (a<b) alors
        Si (a<c) alors
            d ← a
        Sinon
            d ← c ;
        Fin-Si
    Sinon
        Si (b<c) alors
            d ← b
        Sinon
            d ← c ;
        Fin-Si
    Fin-Si
    /* Sorties */
    Écrire('Le résultat = ', d) ;
Fin.
```

⇄

Programme C

```c
#include <stdio.h>
int main()
{
    int a, b, c, d;
    /* Entrées */
    printf("Donner trois nombres entiers : ");
    scanf("%d %d %d", &a, &b, &c);
    /* Traitements */
    if (a<b)
    {
        if (a<c) d = a;
        else d = c;
    }
    else
    {
        if (b<c) d = b;
        else d = c;
    }
    /* Sorties */
    printf("Le résultat = %d", d);
    return 0;
}
```

Que fait cet algorithme ?

2 Le programme donne **le plus petit** des trois nombres entiers.

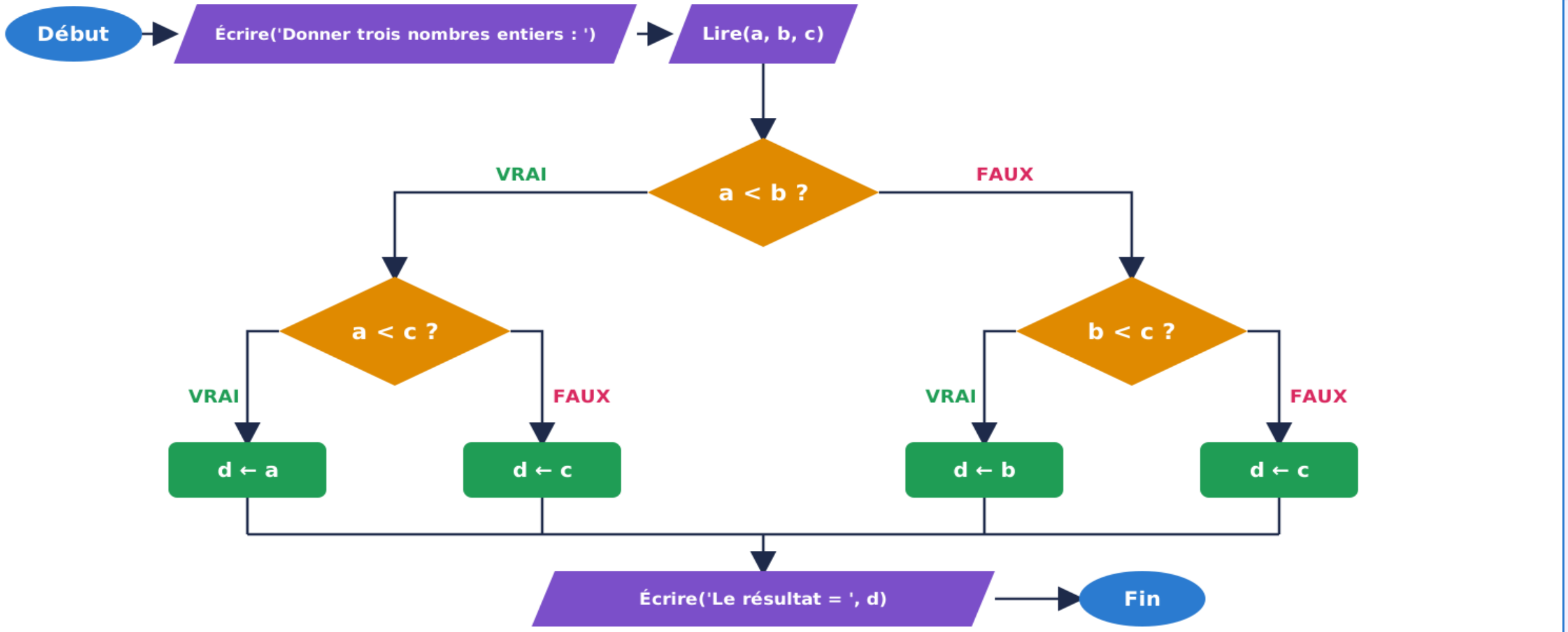
3-1) Déroulement pour a=1, b=2 et c=4

Instructions	Variables				Affichage
	a	b	c	d	
Écrire ('Donner trois nombres entiers : ') ;	/	/	/	/	Donner trois nombres entiers :
Lire (a,b,c) ;	1	2	4	/	
Si (a<b) **alors** **1 < 2** **True** ⇨ **On exécute le bloc Si** **Si** (a<c) **alors** **1 < 4** **True** ⇨ **On exécute le bloc Si** d ← a **d ← 1** **Fin-Si**	1	2	4	**1**	
Écrire ('Le résultat = ', d) ;	1	2	4	**1**	Le résultat = **1**

3-2) Déroulement pour a=4, b=2 et c=0

Instructions	Variables				Affichage
	a	b	c	d	
Écrire ('Donner trois nombres entiers : ') ;	/	/	/	/	Donner trois nombres entiers :
Lire (a,b,c) ;	4	2	0	/	
Si (a<b) **alors** **4 < 2** **False** ⇨ **On n'exécute pas le bloc Si** (on passe au bloc **Sinon**) **Si** (b<c) **alors** **2 < 0** **False** ⇨ **On n'exécute pas le bloc Si** (on passe au bloc **Sinon**) d ← c ; **d ← 0** **Fin-Si**	4	2	0	**0**	
Écrire ('Le résultat = ', d) ;	4	2	0	**0**	Le résultat = **0**

Organigramme de l'algorithme



Exercice N°03

Objectif : appliquer les opérateurs logiques **ET** et **OU**.

Écrire un programme en **langage C** qui permet de vérifier si un triangle est **équilatéral**, **isocèle** ou **quelconque**.

Rappel :

Équilatéral

⟶ condition

```
a = b ET b = c
```

Isocèle

⟶ condition

```
a=b OU a=c OU b=c
```

Quelconque

⟶ condition

```
tous les autres cas
```

Compiler pour les valeurs suivantes :

- a = 5, b = 5, c = 5
- a = 4, b = 3, c = 4
- a = 3, b = 4, c = 5

Corrigé de l'exercice N°03

Triangle : équilatéral, isocèle, quelconque

Programme C

```c
#include <stdio.h>
int main()
{
    float a, b, c;
    printf("Donner la longueur du cote a : ");
    scanf("%f", &a);
    printf("Donner la longueur du cote b : ");
    scanf("%f", &b);
    printf("Donner la longueur du cote c : ");
    scanf("%f", &c);
    if (a == b && b == c)
    {
        printf("Le triangle est equilateral.\n");
    }
    else if (a == b || a == c || b == c)
    {
        printf("Le triangle est isocele.\n");
    }
    else
    {
        printf("Le triangle est quelconque.\n");
    }
    return 0;
}
```

Opérateurs logiques

`&&` = ET

`||` = OU

Résultats

a = 5, b = 5, c = 5

⇒ Le triangle est equilateral.

a = 4, b = 3, c = 4

⇒ Le triangle est isocele.

a = 3, b = 4, c = 5

⇒ Le triangle est quelconque.

Exercice N°04

Objectif : utilisation des conditions dans un exemple concret (TP : Structure de la matière).

Écrire un programme en **langage C** qui lit la **température de l'eau** en degrés Celsius et affiche son **état physique**.

États à afficher :

1	2	3	4	5
État solide (glace)	**Fusion**	**État liquide**	**Ébullition**	**État gazeux** (vapeur)
⟶ si	⟶ si	⟶ si	⟶ si	⟶ si
`T < 0 °C`	`T = 0 °C`	`T < 100 °C`	`T = 100 °C`	`T > 100 °C`

Corrigé de l'exercice N°04

Température de l'eau : état physique

Programme C

```c
#include <stdio.h>
int main()
{
    float T;
    printf("Entrer la temperature en degres Celsius : ");
    scanf("%f", &T);
    if (T < 0)
    {
        printf("Etat solide (glace)\n");
    }
    else if (T == 0)
    {
        printf("Fusion\n");
    }
    else if (T > 0 && T < 100)
    {
        printf("Etat liquide\n");
    }
    else if (T == 100)
    {
        printf("Ebullition\n");
    }
    else // T > 100
    {
        printf("Etat gazeux (vapeur)\n");
    }
    return 0;
}
```

Condition composée

`T > 0 && T < 100`

&& = **ET** : les deux conditions doivent être vraies

Exemples

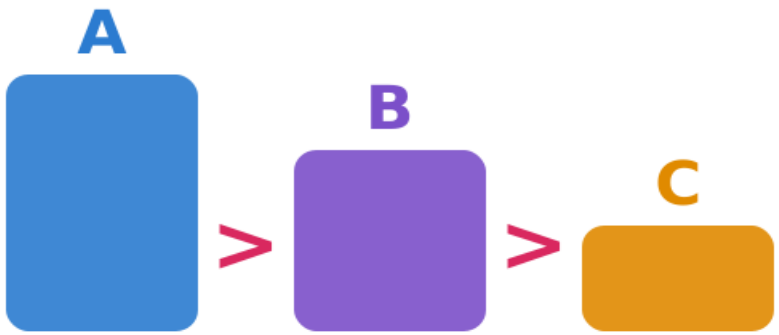
T = -5	⇒	Etat solide (glace)
T = 0	⇒	Fusion
T = 25	⇒	Etat liquide
T = 100	⇒	Ebullition
T = 130	⇒	Etat gazeux (vapeur)

Exercice N°05

Objectif : utilisation des conditions dans un exemple concret (Analyse 1).

Écrire un programme en **langage C** qui permet de vérifier si une suite composée de trois valeurs **A**, **B** et **C** est **monotone décroissante**.

Rappel : suite monotone décroissante



Condition à vérifier :

```
A > B ET B > C
```

Les **deux** inégalités doivent être vraies.

Sinon, la suite n'est pas monotone décroissante.

Corrigé de l'exercice N°05

Suite monotone décroissante

Programme C

```c
#include <stdio.h>

int main()
{
    float A, B, C;
    printf("Donner la valeur de A : ");
    scanf("%f", &A);
    printf("Donner la valeur de B : ");
    scanf("%f", &B);
    printf("Donner la valeur de C : ");
    scanf("%f", &C);

    if (A > B && B > C)
    {
        printf("La suite est monotone decroissante : A > B > C\n");
    }
    else
    {
        printf("La suite n'est pas monotone decroissante.\n");
    }

    return 0;
}
```

Condition composée

```
A > B && B > C
```

&& = ET : les deux conditions doivent être vraies

Exemples

A=5, B=3, C=1 — (5>3) ET (3>1) = vrai

⇒ La suite est monotone decroissante

A=1, B=2, C=3 — (1>2) = faux

⇒ La suite n'est pas monotone decroissante.

A=5, B=5, C=2 — (5>5) = faux

⇒ La suite n'est pas monotone decroissante.

8 points

Exercice 06

1 **a)** Classer ce qui suit dans les trois catégories :

1. **Systèmes d'exploitation**
2. **Langages de programmation**
3. **Logiciels spécialisés**

Pascal | Microsoft Word | Lecteur Media | C++ | Windows Vista | Google Chrome | Mac OS | Matlab | Linux

b) Quelle est la signification des acronymes suivants :

CPU **UAL** **RAM** **ROM**

2 Évaluer les expressions suivantes en respectant **l'ordre de priorité des opérateurs** :

```
E1 = (a/c) - ((d/2*a+4)/4 - c)/2b
avec a=6, b=1, c=1, d=4
```

```
E2 = (1>c) && !(7 % a*2 <= b) || (b >= 8) && (a=b)
avec a=6, b=3, c=1
```

3 Traduire les expressions suivantes en **langage Pascal**

$$E_1 = \frac{-\sqrt{a} + e^{3b} + \sqrt{a^2 + ab}}{2a + |b|} \qquad E_2 = \frac{a^2 + \sqrt{e^a + c}}{3\sqrt{b}}$$

4 Effectuer les conversions suivantes :

$(120)_{10} = (?)_2 = (?)_8$ $(10110101111)_2 = (?)_{16}$

4 points

Exercice 07

On demande d'écrire l'**algorithme d'une fiche de paie journalière** d'un ouvrier rémunéré à la tâche.
Pour cela, on donne :

Rémunération par pièce

La valeur de cette rémunération par pièces réalisées :

VP

Salaire brut (SB)

Calculé selon le nombre de pièces correctes réalisées pendant la journée (**NPC**) comme suit :

l'ouvrier touche :

```
Si NPC ≤ 100 → NPC*VP
Si NPC > 100 → 150*VP
```

Charges sociales (CS)

On enlève à la fin **10 %** du salaire pour les charges sociales (CS).

− 10 %

Calculer et afficher

SB : le salaire journalier brut

CS : les charges sociales

SN : le salaire journalier net

NB : Salaire brut = salaire totale ; Salaire net = salaire sans les charges sociales.

Université A. Mira de Bejaia
Faculté de Technologie
Département de Technologie

Tasdawit n Bgayet
Université de Béjaïa

Année universitaire 2026/2027
1ère Année Technologie

TP Structure des ordinateurs et applications

Série de TP N°3 – Les instructions de test : Si…Fin-Si & Si…Sinon…Fin-Si

But de TP :
Le but du TP est de permettre aux étudiants de comprendre et de maîtriser les structures de contrôle conditionnelles en programmation.

Exercice N°01 : Soit l'algorithme suivant :

```
Algorithme exo1 ;
Variables a, b, c : entier ;
          d, x, x1, x2 : réel ;
Début
   // Entrées
    Écrire ("Entrez les valeurs de a, b et c : ") ;
    Lire (a, b, c) ;
  // Traitements
    d ← b² – (4*a*c) ;
   Si (d > 0) alors
      x1 ← (-b + √d) / (2*a) ;
      x2 ← (-b - √d) / (2*a) ;
      //sorties
      Écrire ("x1= ", x1 , "x2= ", x2 ) ;
   Sinon
      Si (d = 0) alors
         x ←-b / (2*a) ;
            //sortie
         Écrire ("x= ", x ) ;
      Sinon
            //sortie
         Écrire ("Pas de solution réelle.") ;
      Fin-si ;
   Fin-si ;
Fin.
```

Questions :

1. Traduire l'algorithme en un programme en langage C.
2. Compiler et exécuter le programme pour les valeurs suivantes :
   - a = 1 , b = -2 , c = 1 ;
   - a = 2 , b = 2 , c = 3 ;
   - a = 1 , b = -3 , c = 2 ;
3. Déduire ce que fait cet algorithme ?
4. Donner l'algorigramme (Organigramme) de cet algorithme.

Série TP N°03

Exercice N°02 :

Écrire un algorithme, puis un programme en langage C, qui permet à l'utilisateur de saisir deux nombres entiers différents **A** et **B**, puis de déterminer et d'afficher **le plus grand des deux nombres.**

- Compiler pour les valeurs suivantes : A = 4 ; B = 11
  A = 20 ; B = 6

Exercice N°03 :

Écrire un programme en langage C, qui permet de lire les longueurs des trois côtés d'un triangle (**a, b** et **c**), de type réel, puis de déterminer et d'afficher s'il est **équilatéral, isocèle ou quelconque**.

Rappel :

- **Triangle équilatéral** : (a = b) ET (b = c) ;
- **Triangle isocèle** : (a=b) OU (a=c) OU (b=c) ;
- **Triangle quelconque** : les trois côtés ont des longueurs différentes.

- Compiler pour les valeurs suivantes : a = 5 ; b = 8 ; c = 5 | a = 7 ; b = 4 ; c = 9 | a = 6 ; b = 6 ; c = 6 .


Merci