

## TP Informatique 1

### Corrigé de la Série de TP N°4 – (Tests : SI...FIN-SI SI...SINON...FIN-SI)

#### Structures de contrôle conditionnelles ou tests alternatifs

Ces structures sont utilisées pour décider de l'exécution d'un bloc d'instructions : est-ce qu'un bloc d'instruction sera exécuté ou non. Ou bien, pour choisir entre l'exécution de deux blocs différents.

Nous avons deux types de structures conditionnelles :

#### 1. Structure conditionnelle simple :

Un test simple contient un seul bloc d'instructions. Selon une condition (expression logique), on décide est ce que le bloc d'instructions sera exécuté ou non. Si la condition est vraie, on exécute le bloc, sinon on ne l'exécute pas. La syntaxe d'un test alternatif simple est donnée comme suit :

<u>Si</u> (condition) <u>Alors</u> <Bloc_Inst_Si> ; <u>Fin-Si</u> ;	➔ Traduit ➔	<u>if</u> (condition) <u>then</u> <u>begin</u> <Bloc_Inst_Si> ; <u>end</u> ;
---	-------------	---

#### 2. Structure conditionnelle alternée ou double :

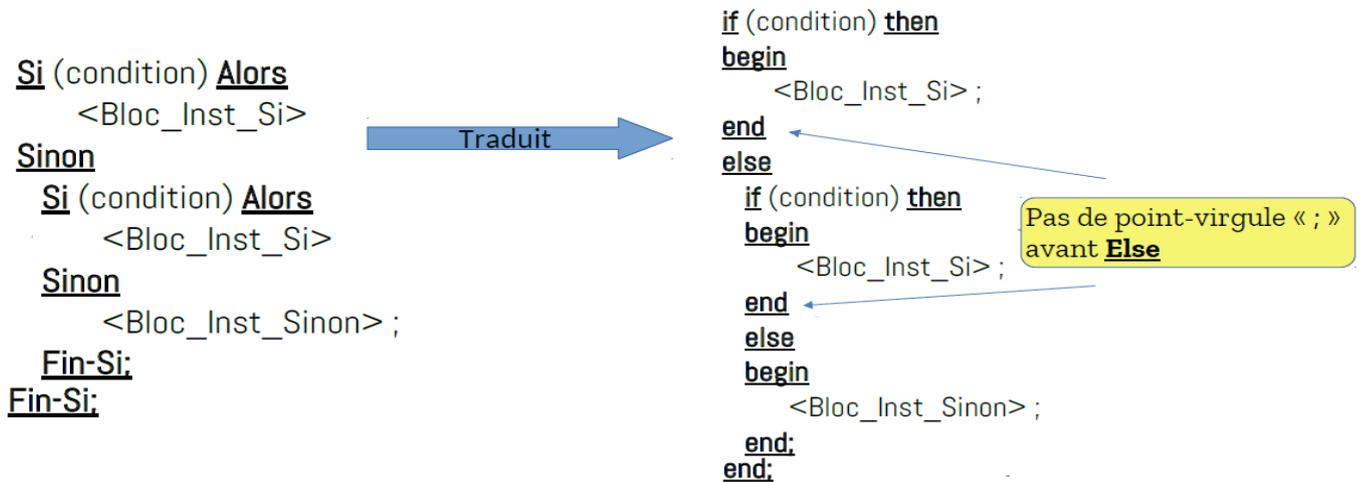
Un test double contient deux blocs d'instructions : on est amené à décider entre le premier bloc ou le second. Cette décision est réalisée selon une condition (expression logique ou booléenne) qui peut être vraie ou fausse. Si la condition est vraie on exécute le premier bloc, sinon on exécute le second. La syntaxe d'un test alternatif double est :

<u>Si</u> (condition) <u>Alors</u> <Bloc_Inst_Si> <u>Sinon</u> <Bloc_Inst_Sinon> ; <u>Fin-Si</u> ;	➔ Traduit ➔	<u>if</u> (condition) <u>then</u> <u>begin</u> <Bloc_Inst_Si> ; <u>end</u> <u>else</u> <u>begin</u> <Bloc_Inst_Sinon> ; <u>end</u> ;
--	-------------	---

Pas de point-virgule « ; » avant **Else**

Nous avons aussi, les **structures conditionnelles doubles et imbriquées** :

Un test double et imbriqué, tout comme un test double, contient deux blocs instructions avec au moins un des deux blocs (bloc Si et/ou bloc Sinon) est composé d'une instruction de condition simple ou double. Donc un test double et imbriqué contient au moins trois blocs d'instructions avec au moins deux conditions. La syntaxe d'un test alternatif double imbriqué avec trois blocs d'instructions est :



Dans les deux types de structure de contrôle conditionnelle, lorsque le bloc d'instructions est composé d'au moins deux instructions, les deux mots clés **begin** et **end** sont obligatoires dans le programme.

Par contre, si le bloc instruction est composé d'une seule instruction, les deux mots clés **begin** et **end** sont facultatifs (optionnels).

Par ailleurs, l'instruction qui précède immédiatement le mot clé Sinon ou Else ne doit pas se terminer par un « **point-virgule** »

## Corrigé de l'exercice N°01 : (Algorithme → Programme)

### 1 / Traduction de l'algorithme en programme Pascal :

Algorithme	Programme Pascal
<p><b>Algorithme</b> Exercice1;</p> <p><b>Variables</b> a, b , c, d, x, x1, x2 : réel;</p> <p><b>Début</b> Écrire ('Entrez la valeur de a, b et c:'); {-.*-.*Entrées*.*-.*} Lire(a, b, c); {-.*-.*Traitement*.*-.*} d ← Sqr(b) - 4* a *c ;</p> <p>Si (d &gt; 0) alors x1 ← (-b-Sqrt(d))/(2*a) ; x2 ← (-b+Sqrt(d))/(2*a) ; {-.*-.*Sorties*.*-.*} Écrire ('x1=', x1:3:1, 'x2=', x2:3:1)</p> <p>Sinon Si (d = 0) alors x ← -b/(2*a) ; {-.*-.*Sortie*.*-.*} Écrire ('x=', x:3:1)</p> <p>Sinon {-.*-.*Sortie*.*-.*} Écrire ('Pas de solution réelle');</p> <p>Fin-si; Fin-si; Fin.</p>	<p><b>Program</b> Exercice1;</p> <p><b>Var</b> a, b , c, d, x, x1, x2 : real;</p> <p><b>Begin</b> Write ('Entrez la valeur de a, b et c :'); {-.*-.*Entrées*.*-.*} Read(a, b, c); {-.*-.*Traitement*.*-.*} d := Sqr(b) - 4* a *c ;</p> <p><b>If</b> (d &gt; 0) <b>then</b> <b>Begin</b> x1 := (-b-Sqrt(d))/(2*a) ; x2 := (-b+Sqrt(d))/(2*a) ; {-.*-.*Sorties*.*-.*} Write('x1=', x1:3:1, 'x2=', x2:3:1); <b>End</b></p> <p><b>Else</b> <b>If</b> (d = 0) <b>then</b> <b>Begin</b> x := -b/(2*a) ; {-.*-.*Sortie*.*-.*} Write('x=', x:3:1); <b>End</b></p> <p><b>Else</b> {-.*-.*Sortie*.*-.*} Write ('Pas de solution réelle');</p> <p><b>End.</b></p>

2/ *Compilation et exécution du programme pour les valeurs suivantes :*

☞  $a = 3 ; b = 5 ; c = 7$

The image shows a Pascal program editor on the left and a console window on the right. The editor contains the following code:

```

1 Program Exercice1;
2 Var
3   a, b , c, d, x, x1, x2 : real;
4 Begin
5   Write ('Entrez la valeur de a, b et c :');
6   {-.*-.*Entrées*.*-.*}
7   Read(a, b, c);
8   {-.*-.*Traitement*.*-.*}
9   d := Sqr(b) - 4* a *c ;
10
11  If (d > 0) then
12  Begin
13    x1 := (-b-Sqrt(d))/(2*a) ;
14    x2 := (-b+Sqrt(d))/(2*a) ;
15    {-.*-.*Sorties*.*-.*}
16    Write ('x1=', x1:3:1, ' x2=', x2:3:1);
17  End
18  Else
19  If (d = 0) then
20  Begin
21    x := -b/(2*a) ;
22    {-.*-.*Sorties*.*-.*}
23    Write ('x=', x:3:1);
24  End
25  Else
26  {-.*-.*Sorties*.*-.*}
27  Write ('Pas de solution réelle');
28 End.
```

The console window, titled 'MyPascal V1.20.5 (Exécuti...', shows the following output:

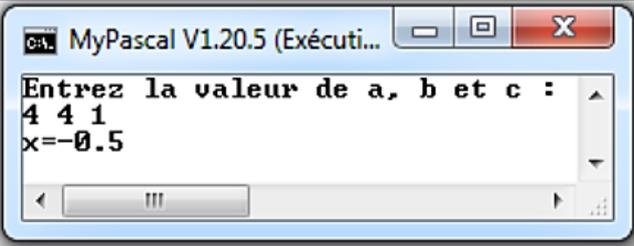
```

Entrez la valeur de a, b et c :
3 5 7
Pas de solution réelle
```

A large arrow points from the console window back to the code, and a box below it contains the text 'Après Exécution'.

☞  $a = 4; b = 4; c = 1$

```
1 Program Exercice1;  
2 Var  
3   a, b, c, d, x, x1, x2 : real;  
4 Begin  
5   Write ('Entrez la valeur de a, b et c :');  
6   {*-.*-*Entrées*-.*-*}  
7   Read(a, b, c);  
8   {*-.*-*Traitement*-.*-*}  
9   d := Sqr(b) - 4*a*c;  
10  
11  If (d > 0) then  
12  Begin  
13    x1 := (-b-Sqrt(d))/(2*a);  
14    x2 := (-b+Sqrt(d))/(2*a);  
15    {*-.*-*Sorties*-.*-*}  
16    Write ('x1=',x1:3:1, ' x2=', x2:3:1);  
17  End  
18  Else  
19    If (d = 0) then  
20    Begin  
21      x := -b/(2*a);  
22      {*-.*-*Sorties*-.*-*}  
23      Write ('x=',x:3:1);  
24    End  
25    Else  
26      {*-.*-*Sorties*-.*-*}  
27      Write ('Pas de solution réelle');  
28  End.
```

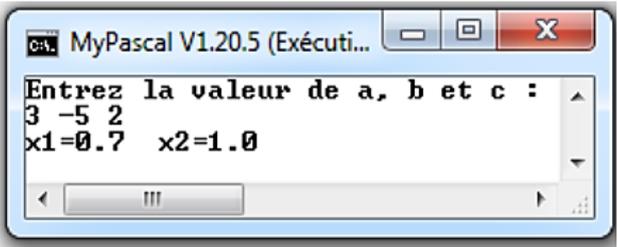


Entrez la valeur de a, b et c :  
4 4 1  
x=-0.5

Après Exécution

☞  $a = 3; b = -5; c = 2$

```
1 Program Exercice1;  
2 Var  
3   a, b, c, d, x, x1, x2 : real;  
4 Begin  
5   Write ('Entrez la valeur de a, b et c :');  
6   {*-.*-*Entrées*-.*-*}  
7   Read(a, b, c);  
8   {*-.*-*Traitement*-.*-*}  
9   d := Sqr(b) - 4*a*c;  
10  
11  If (d > 0) then  
12  Begin  
13    x1 := (-b-Sqrt(d))/(2*a);  
14    x2 := (-b+Sqrt(d))/(2*a);  
15    {*-.*-*Sorties*-.*-*}  
16    Write ('x1=',x1:3:1, ' x2=', x2:3:1);  
17  End  
18  Else  
19    If (d = 0) then  
20    Begin  
21      x := -b/(2*a);  
22      {*-.*-*Sorties*-.*-*}  
23      Write ('x=',x:3:1);  
24    End  
25    Else  
26      {*-.*-*Sorties*-.*-*}  
27      Write ('Pas de solution réelle');  
28  End.
```



Entrez la valeur de a, b et c :  
3 -5 2  
x1=0.7 x2=1.0

Après Exécution

### 3/ Déroulement de l'algorithme pour les valeurs suivantes :

$$a = 3 ; b = -5 ; c = 2$$

Instructions	Variables							Affichage
	a	b	c	d	x	x1	x2	
Écrire ('Entrez la valeur de a, b et c:')	/	/	/	/	/	/	/	Entrez la valeur de a, b et c :
Lire(a, b, c);	3	-5	2	/	/	/	/	
$d \leftarrow \text{Sqr}(b) - 4 * a * c$ $d \leftarrow \text{Sqr}(-5) - 4 * 3 * 2$ $d \leftarrow \text{Sqr}(-5) - 4 * 3 * 2$ $d \leftarrow 25 - 4 * 3 * 2$ $d \leftarrow 25 - 12 * 2$ $d \leftarrow 25 - 24$ $d \leftarrow 1$	3	-5	2	1	/	/	/	
Si ( $d > 0$ ) Si ( $1 > 0$ ) <b>Vrai</b> → On exécute le premier bloc (bloc Si) $x1 \leftarrow (-b - \text{Sqr}(d)) / (2 * a)$ $x1 \leftarrow (-(-5) - \text{Sqr}(1)) / (2 * 3)$ $x1 \leftarrow (-(-5) - 1) / (2 * 3)$ $x1 \leftarrow (-(-5) - 1) / (2 * 3)$ $x1 \leftarrow (5 - 1) / (2 * 3)$ $x1 \leftarrow 4 / (2 * 3)$ $x1 \leftarrow 4 / 6$ <b><math>x1 \leftarrow 0.7</math></b>  $x2 \leftarrow (-b + \text{Sqr}(d)) / (2 * a)$ $x2 \leftarrow (-(-5) + \text{Sqr}(1)) / (2 * 3)$ $x2 \leftarrow (-(-5) + 1) / (2 * 3)$ $x2 \leftarrow (-(-5) + 1) / (2 * 3)$ $x2 \leftarrow (5 + 1) / (2 * 3)$ $x2 \leftarrow 6 / (2 * 3)$ $x2 \leftarrow 6 / 6$ <b><math>x2 \leftarrow 1</math></b>  <b>Écrire ('x1=',x1:3:1, ' x2=',x2:3:1)</b>	3	-5	2	1	/	/	/	<b><math>x1=0.7</math> <math>x2=1.0</math></b>

$$a = 3 ; b = 5 ; c = 7$$

Instructions	Variables							Affichage
	a	b	c	d	x	x1	x2	
Écrire ('Entrez la valeur de a, b et c:')	/	/	/	/	/	/	/	Entrez la valeur de a, b et c :
Lire(a, b, c);	3	5	7	/	/	/	/	
$d \leftarrow \text{Sqr}(b) - 4 * a * c$ $d \leftarrow \text{Sqr}(5) - 4 * 3 * 7$ $d \leftarrow 25 - 4 * 3 * 7$ $d \leftarrow 25 - 12 * 7$ $d \leftarrow 25 - 84$ $d \leftarrow -59$	3	-5	7	-59	/	/	/	
Si ( $d > 0$ ) Si ( $-59 > 0$ ) <b>Faux</b> → On exécute le deuxième bloc (bloc Sinon)  Si ( $d = 0$ ) Si ( $-59 = 0$ ) <b>Faux</b> → On exécute le deuxième bloc (bloc Sinon) <b>Écrire ('Pas de solution réelle')</b>	3	-5	7	-59	/	/	/	<b>Pas de solution réelle</b>

## 4/ Organigramme (Algorigramme)

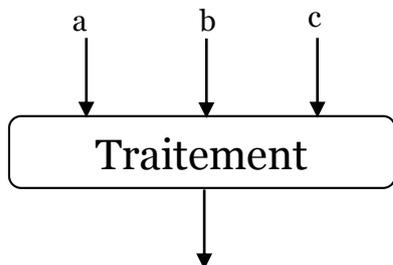
L'algorigramme, ou organigramme, est un diagramme qui représente les étapes d'un algorithme sous forme visuelle. Voici les éléments clés à prendre en compte :

### Symboles (Formes) courants (es) :

- **Rectangle** : Représente une action ou une opération (par exemple, une instruction ou une opération de calcul).
- ◇ **Losange** : Indique une décision (par exemple, une condition "oui" ou "non").
- **Ellipse** : Utilisé pour représenter le début ou la fin du processus.
- **Parallélogramme** : Est utilisé pour l'écriture et la lecture de données (entrées / sorties).
- **Flèches** : Montrent la direction du flux entre les étapes.

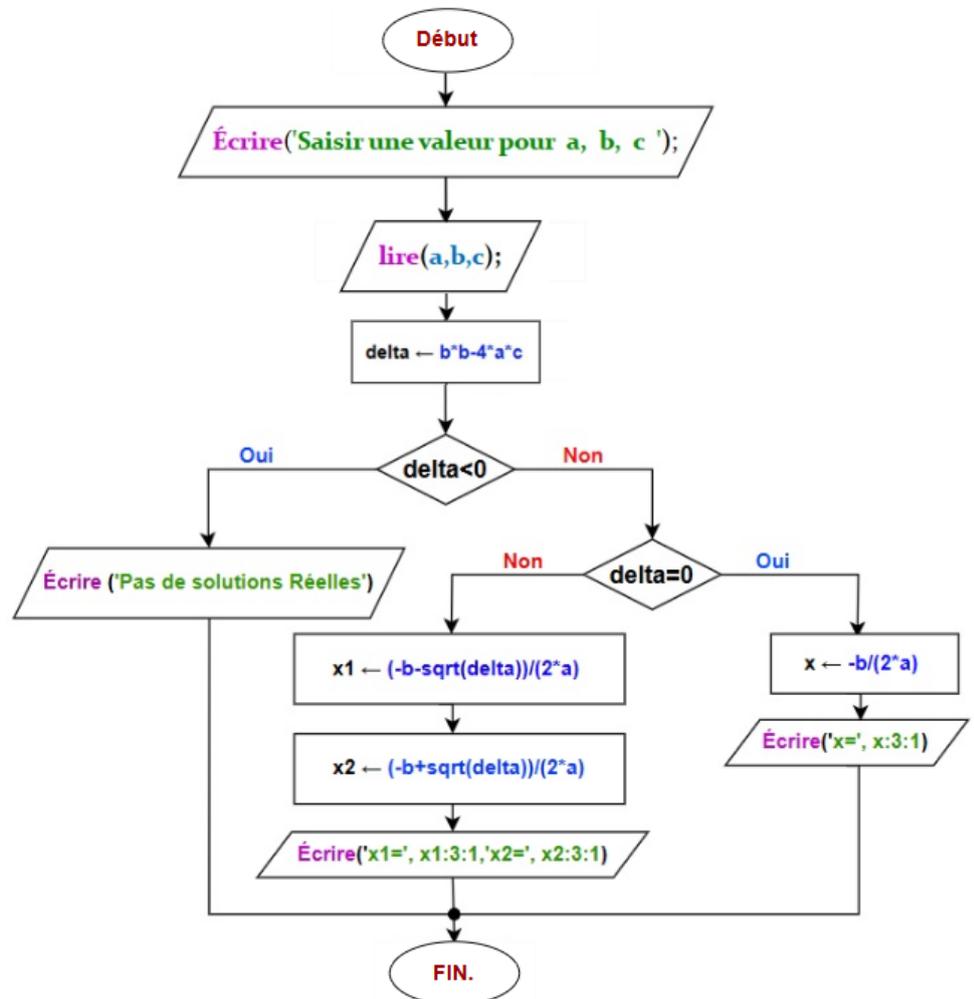
Les différentes formes d'organigramme(Algorigramme)			
Formes	Sémantique / Sense	Formes	Sémantique / Sense
	Représente le début et la Fin de l'organigramme		Tests et décision : on écrit le test à l'intérieur du losange
	Entrées / Sorties : Lecture des données et écriture des résultats.		Ordre d'exécution des opérations (Enchaînement)
	Calculs, Traitements		Connecteur
	<b>Sous-Programmes</b> Portion du programme considérée comme une simple opération		

Le schéma global des variables d'entrées et de sorties est donné comme suit :



x1 et x2 ou x ou pas de solution

### Organigramme



L'algorithme permet de résoudre l'équation du second degré  $ax^2+bx+c=0$ , à partir de la valeur de a, b et c introduites par l'utilisateur, l'algorithme calcule la valeur de Delta « d » comme suit :

$$d = b^2 - 4*a*c$$

Si  $d > 0$  → l'équation admet deux solutions réelles notées x1 et x2:

$$x1 = \frac{-b-\sqrt{d}}{2a} \quad x2 = \frac{-b+\sqrt{d}}{2a}$$

Si  $d = 0$  → l'équation admet une solution réelle double notée x:

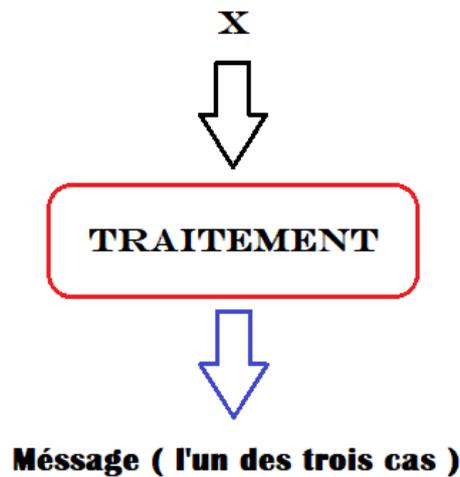
$$x = \frac{-b}{2a}$$

Si  $d < 0$  → l'équation n'admet pas de solution réelle.

## Corrigé de l'exercice N°02 : (Le signe d'un nombre)

**1/Algorithme et le programme PASCAL qui lit un nombre réel X, qui détermine et affiche son signe. Selon le cas : il affiche 'X est positif', 'X est négatif' ou 'X est nul'.**

Cet algorithme peut être schématisé comme suit :



On introduit un nombre réel quelconque, l'algorithme affiche un message indiquant si le nombre est positif, négatif et bien nul. Pour introduire ce nombre on a besoin d'une variable réelle X. Donc la première instruction est *lire(x)*. Pour afficher un message on utilise l'instruction *Ecrire* (exemple : *Écrire ('x est nul')*).

## Algorithme

## Programme

```

Algorithme Exercice_o2 ;
Variables
  x : entier;
Début
  {****Entrée****}
  Écrire('Saisir une valeurs pour x');
  lire(x);

  {****Traitement****}
  Si (x > 0) alors
    {****Sortie****}
    Écrire('x est positif')
  Sinon
    Si (x < 0) alors
      {****Sortie****}
      Écrire ('x est négatif')
    Sinon
      {****Sortie****}
      Écrire ('x est nul');
    FinSi
  FinSi
Fin.
  
```

```

Program Exercice_o2;
Var
  x : integer;

begin
  {**** Entrée ****}
  writeln ('Saisir une valeurs pour x');
  read(x);
  {**** Traitement ****}
  if(x > 0) then
    {**** Sortie****}
    write('x est positif')
  else
    if (x < 0) then
      {**** Sortie****}
      write ('x est négatif')
    else
      {**** Sortie****}
      write ('x est nul');
  end.
  
```

### Exécution du Programme sur My PASCAL

The screenshot displays the My Pascal V1.20.5 environment. On the left, the code editor shows the Pascal program 'Exercice\_02.pas' with line numbers 1 to 20. The code is as follows:

```

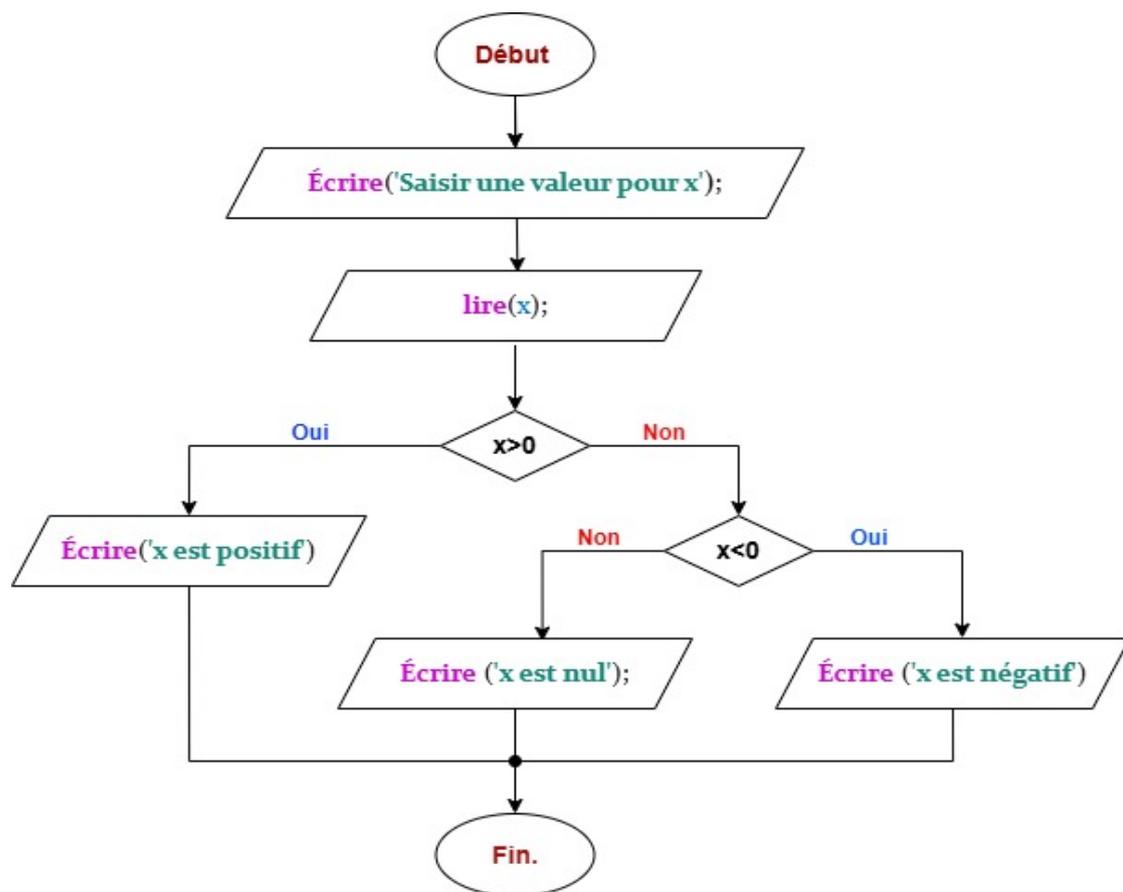
1 Program Exercice_02 ;
2 Var
3   x : integer;
4 begin
5   {**** Entrée ****}
6   writeln ('Saisir une valeur pour x');
7   read(x);
8   {**** Traitement ****}
9   if(x > 0) then
10    {**** Sortie****}
11    write('x est positif')
12  else
13    if (x < 0) then
14     {**** Sortie****}
15     write ('x est négatif')
16    else
17     {**** Sortie****}
18     write ('x est nul');
19  end.
20
  
```

On the right, three execution windows are shown, each titled 'MyPascal V1.20.5 (Exécution) C:\Use...'. Each window shows the program's output for a specific input value:

- Window 1: Input '2', output 'x est positif'.
- Window 2: Input '-77', output 'x est négatif'.
- Window 3: Input '0', output 'x est nul'.

An arrow points from the execution windows to a box labeled 'Après Exécution' (After Execution).

## 2- Organigramme (Algorithme)

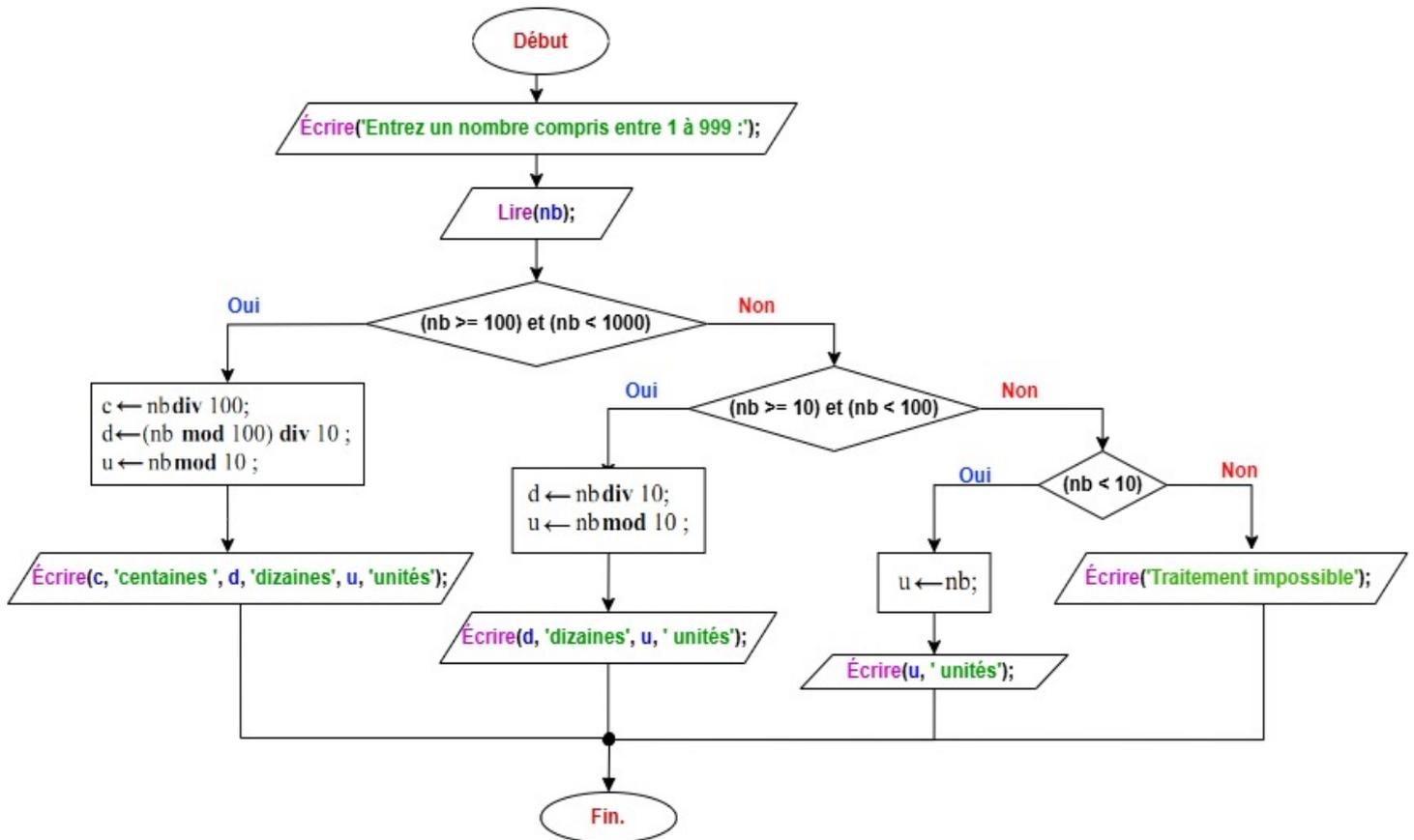


## Corrigé de l'exercice N°03 : (Algorithme → Programme)

1/ Algorithme et sa traduction en programme Pascal qui permet de calculer et afficher le nombre de **centaines**, **dizaines** et **unités** constituant un nombre entier « **nb** » ( $0 < \text{nb} < 1000$ ).

Algorithme	Programme Pascal
<pre> <b>Algorithme</b> Exercice4;  <b>Variables</b> nb, c, d, u : entier;  <b>Début</b> Écrire ('Entrez un nombre compris entre 1 à 999 :'); {--*--*Entrées*--*--} Lire(nb); {--*--*Traitement*--*--} <b>Si</b> (nb &gt;= 100) <b>et</b> (nb &lt; 1000) <b>alors</b>     c ← nb div 100;     d ← (nb mod 100) div 10 ;     u ← nb mod 10 ;     {--*--*Sorties*--*--}     Écrire(c, ' centaines ', d, ' dizaines ', u, ' unités'); <b>Sinon</b>     <b>Si</b> (nb &gt;= 10) <b>et</b> (nb &lt; 100) <b>alors</b>         d ← nb div 10;         u ← nb mod 10 ;         {--*--*Sorties*--*--}         Écrire(d, ' dizaines ', u, ' unités');     <b>Sinon</b>         <b>Si</b> (nb &lt; 10) <b>alors</b>             u ← nb;             {--*--*Sortie*--*--}             Écrire(u, ' unités');         <b>Sinon</b>             {--*--*Sortie*--*--}             Écrire(' Traitement impossible ');         <b>Fin-si</b>;     <b>Fin-si</b>; <b>Fin-si</b>;  <b>Fin.</b>                 </pre>	<pre> Program Exercice4;  Var nb, c, d, u : integer;  Begin Write ('Entrez un nombre compris entre 1 à 999 :'); {--*--*Entrées*--*--} Read(nb); {--*--*Traitement*--*--} <b>If</b> (nb &gt;= 100) <b>and</b> (nb &lt; 1000) <b>then</b>     Begin         c := nb div 100;         d := (nb mod 100) div 10 ;         u := nb mod 10 ;         {--*--*Sorties*--*--}         Write (c, ' centaines ', d, ' dizaines ', u, ' unités');     End <b>Else</b>     <b>If</b> (nb &gt;= 10) <b>and</b> (nb &lt; 100) <b>then</b>         Begin             d := nb div 10;             u := nb mod 10 ;             {--*--*Sorties*--*--}             Write (d, ' dizaines ', u, ' unités');         End     <b>Else</b>         <b>If</b> (nb &lt; 10) <b>then</b>             Begin                 u:=nb;                 {--*--*Sorties*--*--}                 Write(u, ' unités');             End         <b>Else</b>             {--*--*Sorties*--*--}             Write (' Traitement impossible');         End End.                 </pre>

## 2/ Organigramme (Algorigramme)



## Corrigé de l'exercice N°04 : (Programme PASCAL)

### 1/ Programme PASCAL

```
*exo04_SERIE4.pas
1  program TrigonometricFunction;
2  uses
3  Math; { Pour accéder aux fonctions trigonométriques }
4  var
5  x, y: real;
6  begin
7  { Lire la valeur de x }
8  writeln('Entrez la valeur de x :');
9  readln(x);
10 { Convertir x en radians }
11 x := x * (Pi / 180);
12 { Vérifier la valeur de x et calculer y }
13 if x < 0 then
14   y := cos(x)
15 else if (x >= 0) and (x < Pi) then
16   y := sin(x)
17 else
18   y := cos(x) + sin(x);
19 { Afficher le résultat }
20 writeln('La valeur de y est : ', y:3:6); { Affiche y avec 3 décimales }
21 end.
22
```

```
MyPascal V1.20.5 (Exécution) C:\Users\...
Entrez la valeur de x :
-45
La valeur de y est : 0.707107

MyPascal V1.20.5 (Exécution) C:\Users\Maison\Docum...
Entrez la valeur de x :
10
La valeur de y est : 0.173648

MyPascal V1.20.5 (Exécution) C:\Users\...
Entrez la valeur de x :
195
La valeur de y est : -1.224745
```

## Vérifications de résultats

☞ Pour  $x=-45$  degrés, cela se traduit en radians et calculer le résultat en fonction des conditions établies.

### Étapes

1. Conversion de  $-45$  degrés en radians :

$$\text{Radians} = -45 \times (\pi / 180) \approx -0.7854$$

2. Évaluation la fonction :

- Comme  $-0.7854 < 0$ , on utilise la condition pour les angles négatifs :

$$y = \cos(-0.7854)$$

- Utilisant la propriété  $\cos(-x) = \cos(x)$ , nous avons :

$$y = \cos(0.7854) \approx 0.7071$$

☞ Pour  $x=10$  degrés, on procède pour le convertir en radians et calculer le résultat.

### Étapes

1. Conversion de  $10$  degrés en radians :

$$\text{Radians} = 10 \times (\pi / 180) \approx 0.1745$$

2. Évaluation de la fonction :

Comme  $0.1745$ , est compris entre  $0$  et  $\pi$ , nous utilisons la fonction sinus :

$$y = \sin(0.1745) \approx 0.1736$$

☞ Pour  $x=195$  degrés, on procède pour le convertir en radians et calculer le résultat.

### Étapes

3. Conversion de  $195$  degrés en radians :

$$\text{Radians} = 195 \times (\pi / 180) \approx 3,4034$$

4. Évaluation de la fonction :

Comme  $3.4034$  est supérieur à  $\pi$ , nous utilisons la troisième condition :

$$y = \cos(3.4034) + \sin(3.4034) \approx (-0.9980) + (-0.0523) \approx -1.0503$$

## 2 / Organigramme (Algorigramme)

