TP INFORMATIQUE 1

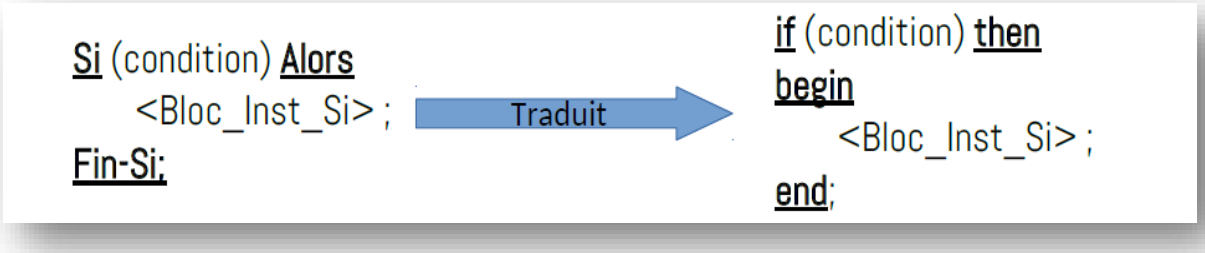
Corrigé SÉRIE DE TP N°04 (Tests : SI…FIN-SI SI…SINON…FIN-SI)

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 :

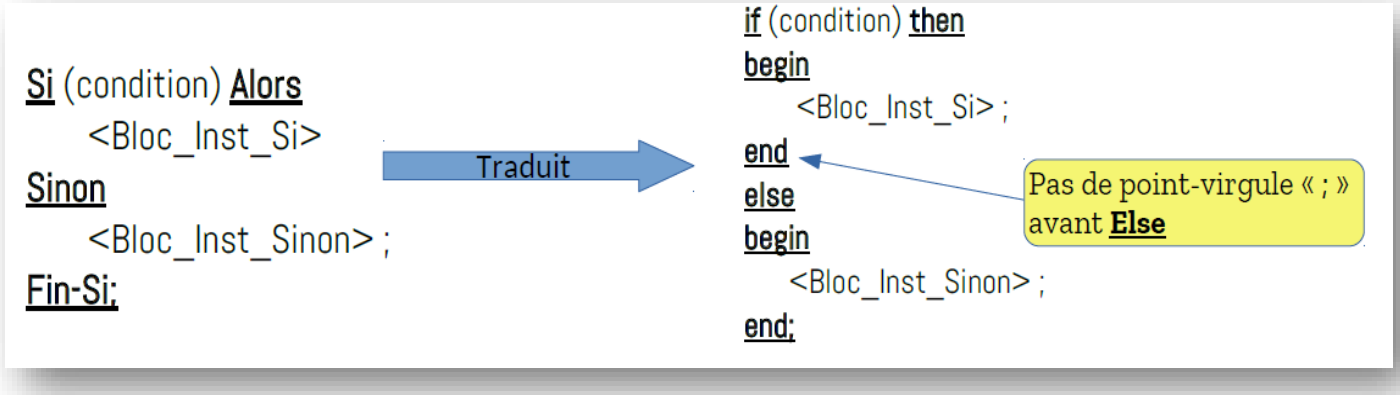
1. Structure conditionnelle simple :

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 :



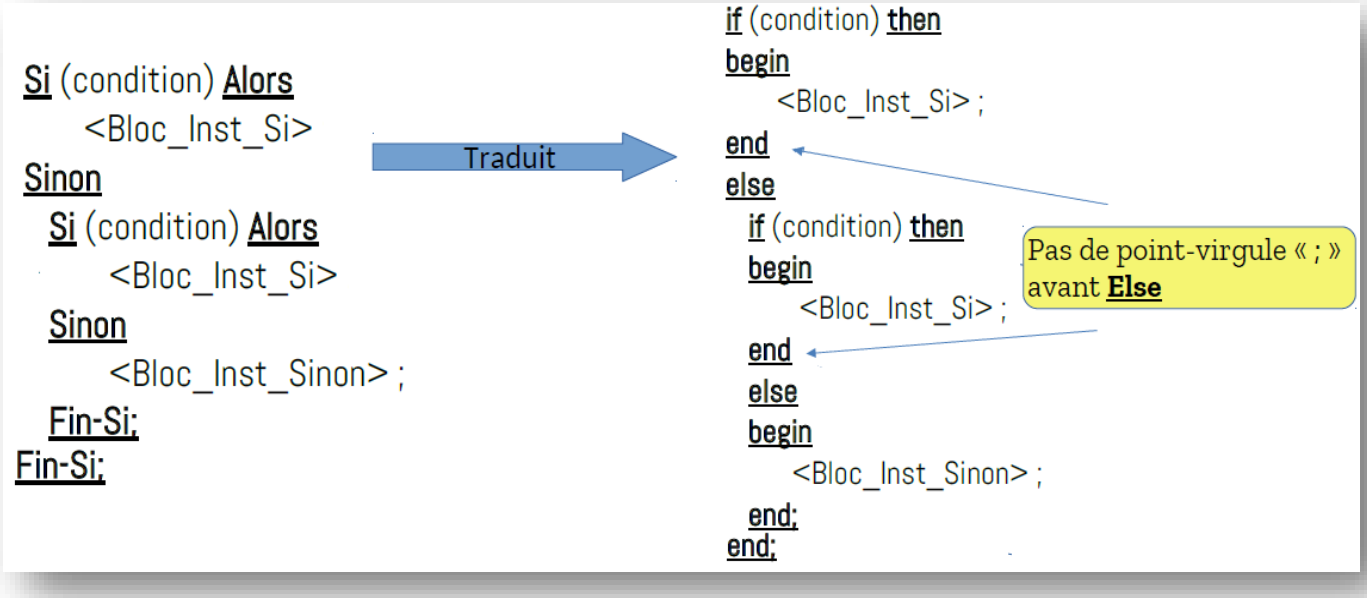
2. Structure conditionnelle alternée ou double :

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 :



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** »

Exercice N°01 : *(Algorithme ➔ Programme)*

1. ***Traduire l'algorithme en programme Pascal :***

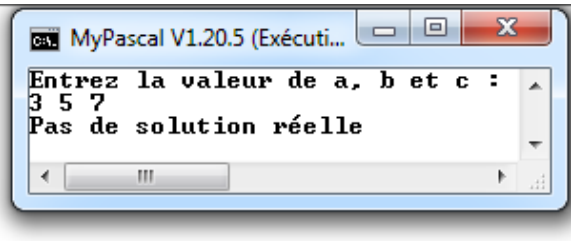
Algorithme	Programme Pascal

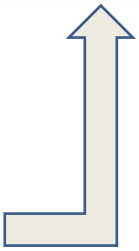
Algorithme Exercice1;
 	
Variables
a, b , c, d, x, x1, x2 : **réel**;
 	
Début
Écrire ('**Entrez la valeur de a, b et c:**');
{-*-*-*Entrées*-*-*-}
Lire(a, b, c);
{-*-*-*Traitement*-*-*-}
d ← Sqr(b) – 4* a *c ;
 	
Si (d > 0) **alors**
x1 ← (-b-Sqrt(d))/(2*a) ;
x2 ← (-b+Sqrt(d))/(2*a) ;
{-*-*-*Sorties*-*-*-}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

2. *Compiler et exécuter le programme pour les valeurs suivantes :*

- **a = 3 b = 5 c = 7**

```
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.
```

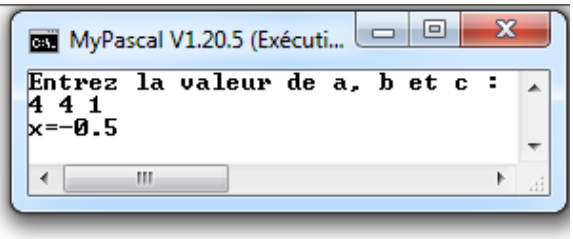


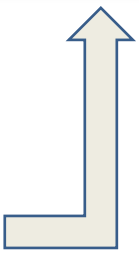


Après Exécution

- **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.
```

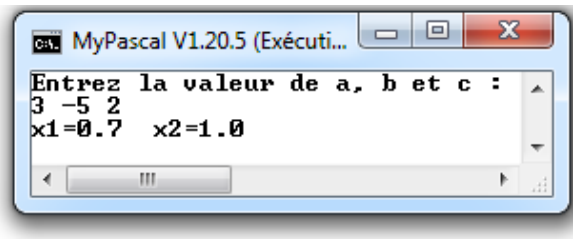




Après Exécution

- **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.
```



Après Exécution

3. *Dérouler l'algorithme pour les valeurs suivantes :*

- **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 ← Sqr(b) – 4* a *c d ← Sqr(5) – 4* 3 *7 d ← 25 – 4* 3 *7 d ← 25 – 12 *7 d ← 25 – 84 d ← -59	3	-5	7	**-59**				
Si (d > 0) **Si** (**-59 > 0**) **Faux** ➔ On exécute le deuxième bloc (bloc Sinon) **Si** (d = 0) **Si** (**-59 = 0**) **Faux** ➔ On exécute le deuxième bloc (bloc Sinon) **Écrire** ('Pas de solution réelle')	3	-5	2	-59	/	/	/	**Pas de solution réelle**

➢ **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 ← Sqr(b) – 4* a *c d ← Sqr(-5) – 4* 3 *2 d ← Sqr(-5) – 4* 3 *2 d ← 25 – 4*3*2 d ← 25 – 12*2 d ← 25 – 24 d ← 1	3	-5	2	**1**				
Si (d > 0) **Si** (**1 > 0**) **Vrai** ➔ On exécute le premier bloc (bloc Si) x1 ← (-b-Sqrt(d))/(2*a) x1 ← (-(-5)-Sqrt(1))/(2*3) x1 ← (-(-5)-1)/(2*3) x1 ← (-(-5)-1)/(2*3) x1 ← (5-1)/(2*3) x1 ← 4/(2*3) x1 ← 4/6 **x1 ← 0.7** x2 ← (-b+Sqrt(d))/(2*a) x2 ← (-(-5)+Sqrt(1))/(2*3) x2 ← (-(-5)+1)/(2*3) x2 ← (-(-5)+1)/(2*3) x2 ← (5+1)/(2*3) x2 ← 6/(2*3) x2 ← 6/6 **x2 ← 1** **Écrire** ('x1=',x1:3:1, ' x2=',x2:3:1)	3	-5	2	1	/	/	/	**x1=0.7 x2=1.0**

4. Déduire ce que fait cet algorithme.

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 er x2:

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

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

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

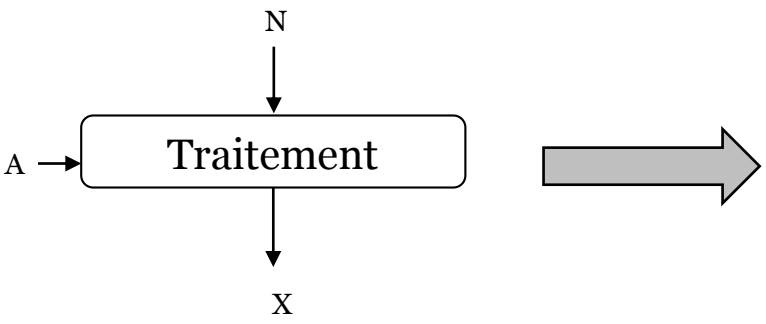
Si d < 0 ➔l'équation n'admet pas de solution réelle.

Exercice N°02 :

Algorithme	Programme Pascal

Algorithme Exercice2;
 	
Variables
N : **entier** ;
A, X : **réel** ;
 	
Début
Écrire ('**Entrez un nombre entier :** ') ;
{-*-*-*Entrée*-*-*-}
Lire(N) ;
{-*-*-*Traitement*-*-*-}
Si (N ≥ 0) **alors**
A ← (2+Sqr(N)+ 3*N)/2
Sinon
A ← (Sqr(N)+Abs(N))/2 ;
Fin-si ;
 	
X ← 1/2 * A ;
{-*-*-*Sortie*-*-*-}
Écrire ('**X =** ', X:5:1) ;
Fin.
	Program Exercice2;
	
	Var
	N : integer;
	A, X : real;
	
	Begin
	Write('Entrez un nombre entier :') ;
	{-*-*-*Entrée*-*-*-}
	Read(N) ;
	{-*-*-*Traitement*-*-*-}
	if (N >= 0) then
	A := (2+Sqr(N)+ 3*N)/2
	else
	A := (Sqr(N)+Abs(N))/2;
	
	X := 1/2 * A;
	{-*-*-*Sortie*-*-*-}
	Write('X = ', X:5:1) ;
	End.

Le schéma (global et des instructions) des variables d'entrée, traitement et sorties est donné comme suit :



Entrée

Lire(N)

Traitement :

Si (N ≥ 0) **alors**
A ← (2+Sqr(N)+ 3*N)/2
Sinon
A ← (Sqr(N)+Abs(N))/2 ;
Fin-si ;
X ← 1/2 * A ;

Sortie :

Écrire(X)

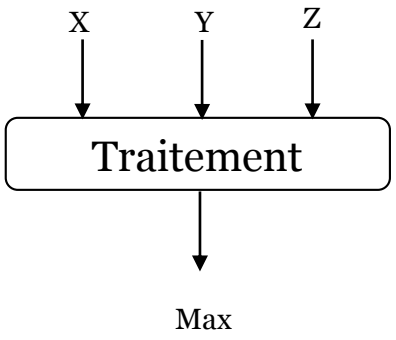
Exercice N°03 :

Algorithme	Programme Pascal

Algorithme Exercice3;
 	
Variables
X, Y, Z, Max : **entier** ;
 	
Début
Écrire ('**Entrez trois nombres entiers :** ') ;
{-*-*-*Entrée*-*-*-}
Lire(X, Y, Z) ;
 	
{-*-*-*Traitement et sorties*-*-*-}
Si (X > Y) et (X > Z) **alors**
Max ← X
Sinon
Si (Y > X) et (Y > Z) **alors**
Max ← Y
Sinon
Max ← Z;
Fin-si;
Fin-si;
 	
Écrire('**Max=**',Max) **;**
Fin.
	Program Exercice3;
	
	Var
	X,Y,Z, Max: integer;
	
	Begin
	Write('Entrez trois nombres entiers :') ;
	{-*-*-*Entrées*-*-*-}
	Read(X, Y, Z) ;
	
	{-*-*-*Traitement et sorties *-*-*-}
	If (X > Y) and (X > Z) then
	Max := X
	Else
	If (Y > X) and (Y > Z) then
	Max := Y
	Else
	Max := Z;
	
	Write('Max = ', Max);
	End.

Remarque : Il existe d'autres solutions algorithmiques pour le calcul du maximum de trois nombres.

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



Exercice N°04 :

Algorithme	Programme Pascal

Algorithme Exercice4;

Variables
nb, c, d , u : **entier**;

Début
Écrire ('**Entrez un nombre compris entre 1 à 999 :**');
{-*-*-*Entrées*-*-*-}
Lire(nb);
{-*-*-*Traitement*-*-*-}
Si (nb >= 100) **et** (nb < 1000) **alors**
c := nb **div** 100;
d := (nb **mod** 100) **div** 10 ;
u := nb **mod** 10 ;
{-*-*-*Sorties*-*-*-}
Écrire(c, ' **centaines** ', d, ' **dizaines** ', u, ' **unités**');
Sinon
Si (nb >= 10) **et** (nb < 100) **alors**
d := nb **div** 10;
u := nb **mod** 10 ;
{-*-*-*Sorties*-*-*-}
Écrire(d, ' **dizaines** ', u, ' **unités**');
Sinon
Si (nb < 10) **alors**
u := nb;
{-*-*-*Sortie*-*-*-}
Écrire(u, ' **unités**');
Sinon
{-*-*-*Sortie*-*-*-}
Écrire(' **Traitement impossible** ');
Fin-si;
Fin-si;
Fin-si;

Fin.

```
Program Exercice4;

Var
 nb, c, d , u : integer;

Begin
  Write ('Entrez un nombre compris entre 1 à 999 :');
  {-*-*-*Entrées*-*-*-}
  Read(nb);
    {-*-*-*Traitement*-*-*-}
  If (nb >= 100) and (nb < 1000) then
     Begin
      c := nb div 100;
      d := (nb mod 100) div 10 ;
      u := nb mod 10 ;
       {-*-*-*Sorties*-*-*-}
      Write (c, ' centaines ', d, ' dizaines ', u, ' unités');
     End
   Else
     If (nb >= 10) and (nb < 100) then
       Begin
         d := nb div 10;
         u := nb mod 10 ;
        {-*-*-*Sorties*-*-*-}
         Write (d, ' dizaines ', u, ' unités');
       End
      Else
        If (nb < 10) then
         Begin
         u:=nb;
         {-*-*-*Sorties*-*-*-}
         Write(u, ' unités');
          End
         Else
           {-*-*-*Sorties*-*-*-}
           Write (' Traitement impossible');
End.
```