TP Structure des Ordinateurs et Applications

Corrigé de la Série de TP N°5 – *Les instructions itératives Pour, Tant-que et Répéter*

Rappel :

Structures de contrôle répétitives : nous permettent de répéter un traitement un nombre fini de fois.

Nous avons trois types de structures itératives (boucles) :

1. Boucle Pour (For)

La structure de contrôle répétitive ***Pour*** (***For*** en langage ***PASCAL***) utilise un indice entier qui varie (avec un ***incrément = 1***) d'une valeur initiale jusqu'à une valeur finale. À la fin de chaque itération, l'indice est incrémenté de 1 d'une manière automatique (implicite).

La syntaxe de la ***boucle pour*** est comme suit :

```
pour <indice>←<vi> à <vf> faire
    <instruction(s)>
finPour;
```

```
for <indice>:=<vi> to <vf> do
 begin
     <instruction(s)>;
 end;
```

<indice> : variable entière

<vi> : valeur initiale <vf> : valeur finale

2. Boucle Tant-que (While)

La structure de contrôle répétitive ***Tant-que*** (***While*** en langage ***PASCAL***) utilise une expression ***logique*** ou ***booléenne*** comme condition d'accès à la boucle : ***si*** la condition est vérifiée (elle donne un résultat ***vrai*** : ***TRUE***) donc on entre à la boucle, ***sinon*** on la quitte.

La syntaxe de la ***boucle tant-que*** est comme suit :

```
tant-que <condition> faire
    <instruction(s)>
finTant-que;
```

```
while <condition> do
 begin
     <instruction(s)>;
 end;
```

<condition> : expression logique qui peut être vraie ou fausse.

On exécute le bloc d'instructions ***tant-que*** la condition est ***vraie***. Une fois la condition est **fausse**, on arrête la boucle, et on continue l'exécution de l'instruction qui vient après fin ***Tant-que*** (après end).

3. Boucle Répéter (Repeat)

La structure de contrôle répétitive ***Répéter*** (***Repeat*** en langage ***PASCAL***) utilise une expression ***logique*** ou ***booléenne*** comme condition de sortie de la boucle : ***si*** la condition est vérifiée (elle donne un résultat ***vrai*** : ***TRUE***) on sort de la boucle, ***sinon*** on y accède (on répète l'exécution du bloc).

La syntaxe de la boucle répéter est comme suit :

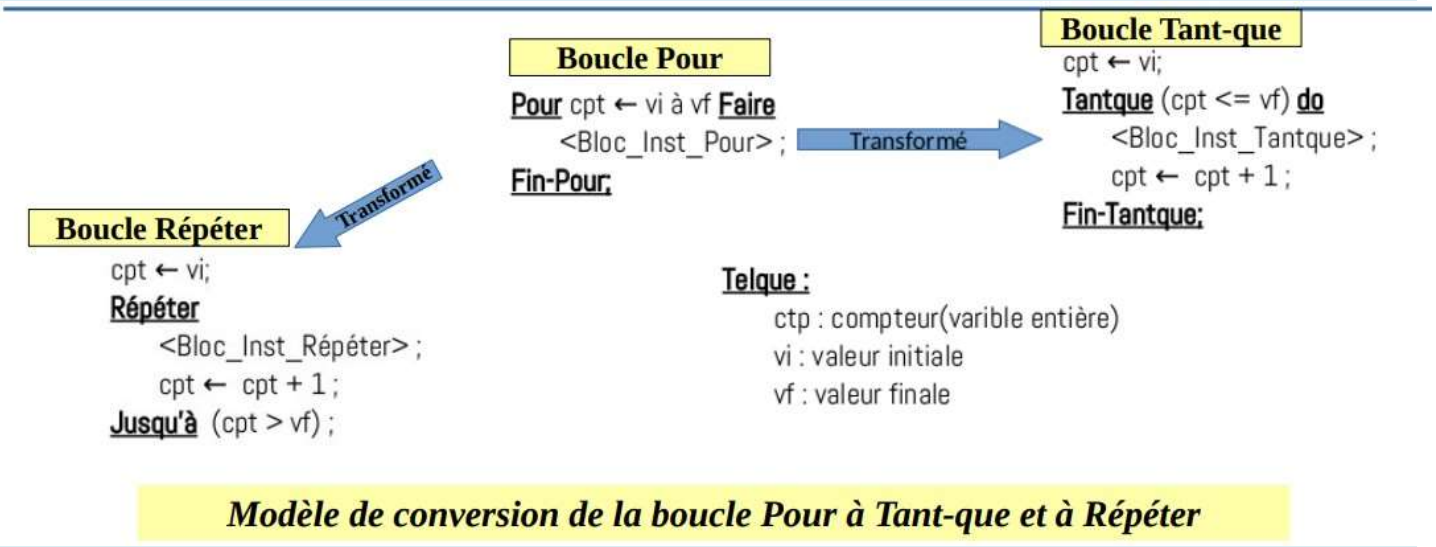
```
repéter
    <instruction(s)>
jusqu'à <condition>;
```

```
repeat
    <instruction(s)>;
until <condition>;
```

<condition> : expression logique qui peut être vraie ou fausse.

On exécute le bloc d'instructions ***jusqu'à*** avoir la condition ***correcte***. Une fois la condition est ***vérifiée***, on ***arrête la boucle***, et on continue l'exécution de l'instruction qui vient après ***jusqu'à (après until)***.

Dans la ***boucle repeat*** on n'utilise pas ***begin*** et ***end*** pour délimiter le bloc d'instructions (le bloc est déjà délimité par ***repeat et until).***



Modèle de conversion de la boucle Pour à Tant-que et à Répéter

Solution de l'exercice N°01 :

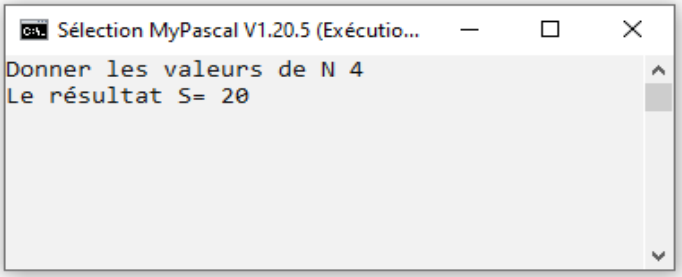
1- Traduction de l'algorithme en programme PASCAL :

Algorithme	Pascal

Algorithme TP5_Exo_01 ;
Variables
i, N,S : entier ;
 	
Début
{-*-*-*- Entrées -*-*-*-}
Écrire ('Donner la valeur de N ') ;
Lire (N) ;
{-*-*-*- Traitements -*-*-*}
S←0 ;
Pour i←1 **à** N **faire**
S ← S + 2*i ;
Fin-Pour;
{-*-*-*- Sorties -*-*-*}
Écrire ('Le résultat S= ', **S**) ;
Fin.
	program TP5_Exo_01;
	var
	i,N,S : integer;
	
	begin
	{-*-*-*- Entrées -*-*-*-}
	write ('Donner la valeur de N ') ;
	read (N) ;
	{-*-*-*- Traitements -*-*-*}
	S:=0 ;
	for i:=1 to N do
	begin
	S := S + 2*i ;
	end;
	{-*-*-*- Sorties -*-*-*}
	write ('Le résultat S= ', S) ;
	end.

☞ ***Compilation et exécution du programme pour N = 4***

```
1  program TP5_Exo_01;
2  var
3  i,N,S : integer;
4
5  begin
6     {-*-*-*- Entrées -*-*-*-}
7  write ('Donner les valeurs de N ') ;
8  read (N) ;
9   {-*-*-*- Traitements -*-*-*}
10   S:=0 ;
11     for i:=1 to N do
12     begin
13       S := S + 2*i ;
14     end;
15   {-*-*-*- Sorties -*-*-*}
16     write ('Le résultat S= ', S) ;
17 end.
```



2- Déroulement de l'algorithme pour N=4 :

Instructions	Variables			Affichages
	i	N	S	
Écrire ('Donner la valeur de N ') ;	/	/	/	**Donner la valeur de N**
Lire (N) ;	/	**4**	/	/
S←0 ;	/	**4**	**0**	/
Pour i←1 **S ← S + 2*i ; S ← 0 + 2*1; S ← 2 ;**	**1**	**4**	**2**	/
Pour i←2 **S ← S + 2*i ; S ← 2 + 2*2 ; S ← 2 + 4 ; S ← 6 ;**	**2**	**4**	**6**	/
Pour i←3 **S ← S + 2*i ; S ← 6 + 2*3 ; S ← 6 + 6 ; S ← 12 ;**	**3**	**4**	**12**	/
Pour i←4 **S ← S + 2*i ; S ← 12 + 2*4 ; S ← 12 + 8 ; S ← 20 ;**	**4**		**20**	/
Écrire ('Le résultat S= ', S) ;	**4**	**4**	**20**	**Le résultat S= 20**

3- Déduction de l'expression finale :

Selon le déroulement ci-dessus, nous avons :

$$pour\ i = 1 \rightarrow S = 0 + 2 = 2$$
$$pour\ i = 2 \rightarrow S = 2 + 4 = 6$$
$$pour\ i = 3 \rightarrow S = 6 + 6 = 12$$
$$pour\ i = 4 \rightarrow S = 12 + 8 = 20$$
$$\vdots$$
$$pour\ i = N \rightarrow S = 2 + 4 + 6 + 8 + \cdots + 2N$$

La formule générale est : $$S = \sum_{i=1}^{N} 2i$$

4) Réécriture de l'algorithme/PASCAL en remplaçant la boucle *Pour* par la boucle *Tant-que.*

Algorithme	Pascal

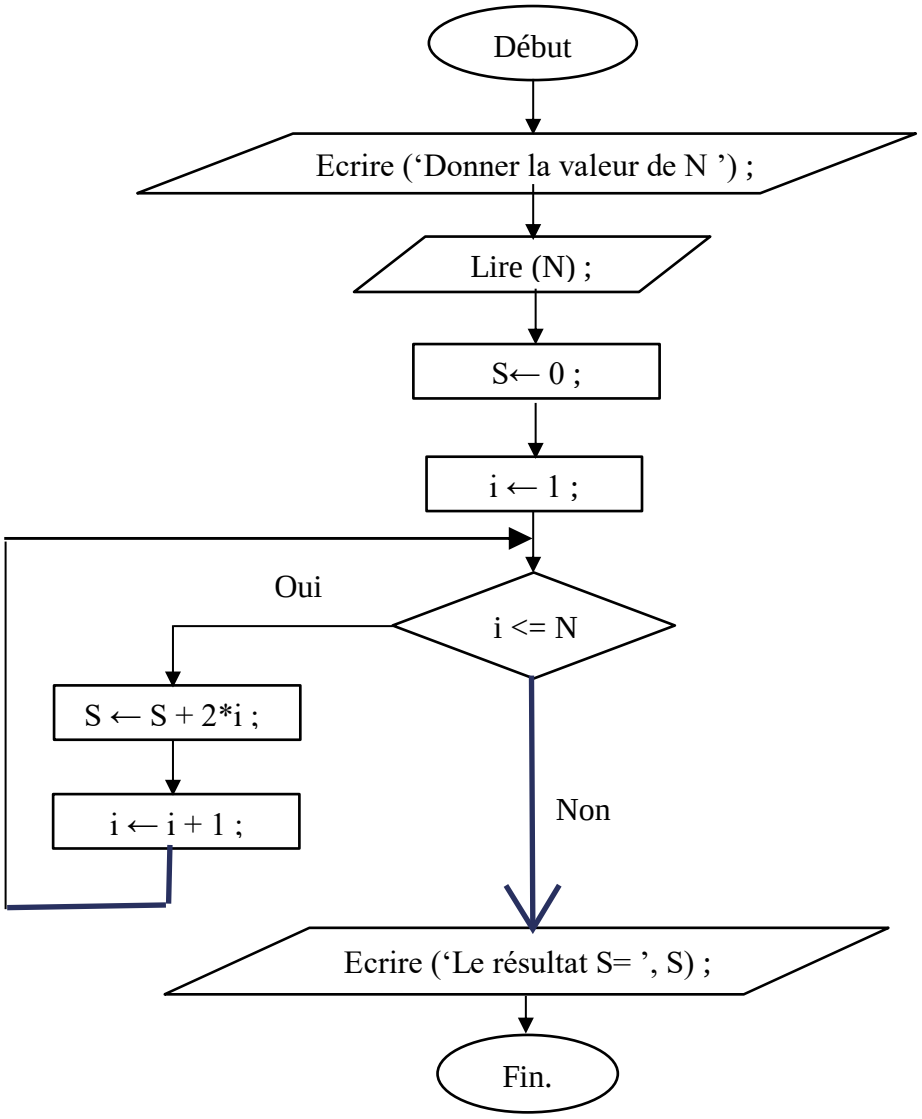
Algorithme TP5_Exo_01 ;
Variables
i, N,S : entier ;
 	
Début
{-*-*-*- Entrées -*-*-*-}
Écrire ('Donner la valeur de N ') ;
Lire (N) ;
{-*-*-*- Traitements -*-*-*}
S←0 ; i := 1 ;
Tant-que (i <= N) faire
S ← S + 2*i ;
i := i + 1 ;
Fin-Tant-que;
{-*-*-*- Sorties -*-*-*}
Écrire ('Le résultat S= ', **S**) ;
Fin.
	program TP5_Exo_01;
	var
	i,N,S : integer;
	
	begin
	{-*-*-*- Entrées -*-*-*-}
	write ('Donner la valeur de N ') ;
	read (N) ;
	{-*-*-*- Traitements -*-*-*}
	S:=0 ; i:=1;
	while (i<=N) do
	begin
	S := S + 2*i ;
	i := i + 1;
	end;
	{-*-*-*- Sorties -*-*-*}
	write ('Le résultat S= ', S) ;
	end.

5) Réécriture de l'algorithme/PASCAL en remplaçant la boucle *Pour* par la *boucle Répéter*.

Algorithme	Pascal

Algorithme TP5_Exo_01 ;
Variables
i, N,S : entier ;
 	
Début
{-*-*-*- Entrées -*-*-*-}
Écrire ('Donner la valeur de N ') ;
Lire (N) ;
{-*-*-*- Traitements -*-*-*}
S←0 ; i := 1 ;
Répéter
S ← S + 2*i ;
i := i + 1 ;
jusqu'à (i > N);
{-*-*-*- Sorties -*-*-*}
Écrire ('Le résultat S= ', **S**) ;
Fin.
	program TP5_Exo_01;
	var
	i,N,S : integer;
	
	begin
	{-*-*-*- Entrées -*-*-*-}
	write ('Donner la valeur de N ') ;
	read (N) ;
	{-*-*-*- Traitements -*-*-*}
	S:=0 ; i:=1;
	repeat
	S := S + 2*i ;
	i := i + 1;
	until (i > N);
	{-*-*-*- Sorties -*-*-*}
	write ('Le résultat S= ', S) ;
	end.

6- Donner l'organigramme de cet algorithme :



7-Modifier le programme afin qu'il additionne les nombres jusqu'à ce que la somme dépasse 100 et afficher le nombre d'itérations i effectuées pour obtenir cette somme :

Algorithme	Pascal

Algorithme TP5_Exo_01 ;
Variables
i, S : entier ;

Début
{-*-*-*- Traitements -*-*-*}
S←0 ; i := 1 ;
Tant-que (S <= 100) faire
S ← S + 2*i ;
i := i + 1 ;
Fin-Tant-que;
{-*-*-*- Sorties -*-*-*}
Écrire ('Le résultat S= ', **S**) ;
Écrire ('Le nombre d'itérations nécessaire pour obtenir une somme supérieure à 100 est : i = ', i) ;
Fin.

```
program TP5_Exo_01;
var
i,S : integer;

begin
{-*-*-*- Traitements -*-*-*}
  S:=0 ; i:=1;
     while (S <= 100) do
     begin
        S := S + 2*i ;
        i := i + 1;
     end;
  {-*-*-*- Sorties -*-*-*}
    writeln ('Le résultat s= ', S) ;
    writeln ('Le nombre d''itérations
nécessaire pour obtenir une somme
supérieure à 100 est : i = ', i) ;
end.
```

Compilation et exécution du programme :

```
program TP5_Exo_01;
var
i,S : integer;

begin
 {-*-*-*- Traitements -*-*-*}
 S:=0 ; i:=1;
   while (S <= 100) do
   begin
    S := S + 2*i ;
    i := i + 1;
   end;
 {-*-*-*- Sorties -*-*-*}
  writeln ('Le résultats S= ' , S);
   writeln ('Le nombre d''itérations nécessaire pour obtenir une somme supérieure à 100 est : i = ', i) ;
end.
```

```
Administrateur : MyPascal V1.20.5 (Exécution) D:\tp_pa...
Le résultats S= 110
Le nombre d'itérations nécessaire pour obtenir une somme
supérieure à 100 est : i = 11
```

Solution de l'exercice N°02 :

☞ ***Objectif de l'exercice :***

Cet exercice permet de comprendre l'utilisation de la boucle ***FOR*** en Pascal. On utilise cette boucle lorsqu'on connaît à l'avance le nombre de répétitions.

La boucle ***FOR*** avance automatiquement avec un ***pas de 1***, ce qui convient parfaitement dans cet exercice puisque l'utilisateur saisit un nombre N et que le programme doit afficher tous les nombres de 1 jusqu'à N.

☞ ***Etapes de résolution du problème***

1. Demander à l'utilisateur de saisir un entier positif ***N***.
2. Utiliser une boucle ***FOR*** pour parcourir tous les nombres de ***1*** à ***N***.
3. À chaque passage dans la boucle :
   - Tester si le nombre est pair ou impair en utilisant l'opérateur **modulo** (***i mod 2***).
   - Afficher le nombre ainsi que son type (pair ou impair).

☞ ***Algorithme/Programme en PASCAL***

Algorithme	**Programme Pascal**

Algorithme Exercice02;
 	
Variables
N, i : **Entiers**;
 	
Début
{--*-*- Entrées -*-*-*-}*
// Demande à l'utilisateur de saisir un entier positif
Ecrire ('Saisir un entier positif N : ');
Lire (N);
 	
{--*-*- Traitements -*-*-*-}*
// Boucle FOR : i varie de 1 jusqu'à N
Pour i := 1 **à** N **faire**
 	
// Vérifier si le nombre est pair ou impair
Si (i mod 2 = 0) **alors**
Ecrire (i, ' est un nombre pair')
Sinon
Ecrire (i, ' est un nombre impair');
Fin-Si;
 	
Fin.
	Program Exercice02;
	
	Var
	N, i : **Integer**;
	
	Begin
	{--*-*- Entrées -*-*-*-}*
	// Demande à l'utilisateur de saisir un entier positif
	Write ('Saisir un entier positif N : ');
	Read (N);
	
	{--*-*- Traitements -*-*-*-}*
	// Boucle FOR : i varie de 1 jusqu'à N
	For i := 1 **To** N **Do**
	Begin
	// Vérifier si le nombre est pair ou impair
	If (i mod 2 = 0) **Then**
	Writeln (i, ' est un nombre pair')
	Else
	Writeln (i, ' est un nombre impair');
	End;
	
	End.

☞ ***Compilation et exécution du programme pour N = 7 :***

```
1  Program Exercice02;
2
3  Var
4   N, i : Integer;
5
6  Begin
7   {-*-*-*- Entrées -*-*-*-}
8   // Demande à l'utilisateur de saisir un entier positif
9   Write('Saisir un entier positif N : ');
10  Read (N);
11
12 {-*-*-*- Traitements -*-*-*-}
13  // Boucle FOR : i varie de 1 jusqu'à N
14  For i := 1 To N Do
15  Begin
16   // Vérifier si le nombre est pair ou impair
17   If (i mod 2 = 0) Then
18    Writeln(i, ' est un nombre pair')
19   Else
20    Writeln(i, ' est un nombre impair');
21  End;
22
23 End.
```

```
Administrateur : MyPascal V1.20.5 (Exécution) D:\tp_pascal\tp5_exo2....
Saisir un entier positif N : 7
1 est un nombre impair
2 est un nombre pair
3 est un nombre impair
4 est un nombre pair
5 est un nombre impair
6 est un nombre pair
7 est un nombre impair
```

Solution de l'exercice 03 :

☞ *Objectif de l'exercice :*

Cet exercice permet de comprendre l'utilisation de la boucle ***WHILE*** en Pascal. Dans ce programme, la vitesse ne s'incrémente pas de 1 comme dans les exemples habituels, mais de 4 à chaque étape. Cela montre que, lorsque l'on doit contrôler nous-même l'évolution d'une variable, la boucle ***WHILE*** est plus adaptée. Elle permet de répéter le calcul tant que la vitesse ne dépasse pas la valeur maximale choisie par l'utilisateur.

NB : La boucle ***FOR*** est pratique lorsque l'incrément est simple et régulier ***(+1)***. Dès qu'on a besoin d'un pas différent de 1, les boucles ***WHILE*** ou ***REPEAT...UNTIL*** deviennent plus appropriées.

☞ *Etapes de résolution du problème*

1- Demander à l'utilisateur de saisir la valeur maximale de la vitesse ***N.*** Cette valeur doit être un entier supérieur ou égal à ***2***.

2- Initialiser la vitesse **v** à ***1 m/s.***

3- Calculer ***l'énergie cinétique Ec,*** avec la formule donnée, pour chaque vitesse **v** de ***1m/s jusqu'à N***, avec un pas de ***4m/s*** à chaque itération.

4- Pour chaque vitesse v, afficher l'énergie cinétique correspondante, avec deux chiffres après la virgule.

☞ *Programme en PASCAL*

```
1   Program Exercice03;
2
3   const
4   m = 3;  { masse constante de l'objet }
5
6   Var
7    v, N : Integer;     { v : vitesse, N : limite maximale }
8    Ec : Real;          { énergie cinétique }
9
10  Begin
11   {-*-*-*- Entrées -*-*-*-}
12   Write('Donner la valeur maximale de la vitesse N : ');
13   Read (N);
14
15  {-*-*-*- Traitements -*-*-*-}
16   v := 1;  { première vitesse }
17   While (v <= N) do
18     Begin
19       Ec := 1/2 * m * sqr (v)  ;      { calcul de l'énergie cinétique }
20       Writeln('Pour v = ', v, ' m/s --> Ec = ', Ec:0:2, ' J');  { afficher le résultat }
21       v := v + 4;  { incrémentation par pas de 4 }
22     End;
23
24  End.
```

☞ ***Compilation et exécution du programme pour N=18 :***

```
1  Program Exercice03;
2
3  const
4  m = 3;  { masse constante de l'objet }
5
6  Var
7   v, N : Integer;     { v : vitesse, N : limite maximale }
8   Ec : Real;          { énergie cinétique }
9
10 Begin
11  {-*-*-*- Entrées -*-*-*-}
12  Write('Donner la valeur maximale de la vitesse N : ');
13  Read (N);
14
15 {-*-*-*- Traitements -*-*-*-}
16  v := 1;  { première vitesse }
17  While (v <= N) do
18   Begin
19    Ec := 1/2 * m * sqr (v)  ;     { calcul de l'énergie cinétique }
20    Writeln('Pour v = ', v, ' m/s --> Ec = ', Ec:0:2, ' J');  { afficher le résultat }
21    v := v + 4;  { incrémentation par pas de 4 }
22   End;
23
24 End.
```

```
Administrateur : MyPascal V1.20.5 (Exécution) D:\tp_pascal\tp...
Donner la valeur maximale de la vitesse N : 18
Pour v = 1 m/s --> Ec = 1.50 J
Pour v = 5 m/s --> Ec = 37.50 J
Pour v = 9 m/s --> Ec = 121.50 J
Pour v = 13 m/s --> Ec = 253.50 J
Pour v = 17 m/s --> Ec = 433.50 J
```

Solution de l'exercice N°04 :

☞ ***Objectif de l'exercice :***

Cet exercice permet de comprendre l'utilisation de la boucle ***REPEAT…UNTIL*** en Pascal. Cette boucle est utilisée lorsque l'on ne connaît pas à l'avance combien de fois une action doit être répétée.

Dans ce programme, on ne sait pas combien de tentatives l'utilisateur fera avant de taper le bon mot de passe. La boucle ***REPEAT*** permet donc de répéter la saisie tant que la condition n'est pas remplie.

☞ ***Etapes de résolution du problème***

1. Demander à l'utilisateur de saisir un ***mot de passe***.
2. Lire la valeur entrée.
3. Vérifier si le mot de passe est correct (***4951***) en utilisant une condition **SI**.
4. Tant que le mot de passe est incorrect :
   • afficher « ***Mot de passe incorrect, veuillez réessayer.*** »
   • redemander une nouvelle saisie.
5. Quand le mot de passe correct est saisi, afficher « ***Accès autorisé !*** ».

☞ ***Programme en PASCAL***

```
1   Program Exercice04;
2
3   Var
4    MP : Integer;
5
6   Begin
7
8    Repeat
9     Write('Saisir le mot de passe composé de 4 chiffres : '); { demander la saisie du mot de passe }
10    Readln(MP); { lire la valeur saisie }
11
12    If (MP <> 4951) Then { vérifier si le mot de passe est incorrect }
13     Writeln('Mot de passe incorrect, veuillez réessayer.'); { afficher un message d'erreur }
14
15   Until (MP = 4951); { répéter tant que le mot de passe n'est pas correct }
16
17   Writeln('Accès autorisé ! '); { afficher un message lorsque le mot de passe est correct }
18
19  End.
```

☞ ***Compilation et exécution du programme pour : MP = 1234 ; MP = 4586 ; MP = 4951***