Chapitre 2 :

Notion d'Algorithme et de Programme

Université A/Mira de Bejaia- 2020/2021
Département de Technologie
1ere année
Module : Informatique1
Enseignante : Mme MAMMERI

1. Exemple introductif

Si on veut résoudre une équation de 2eme degré $ax^2 + bx + c = 0$; a, b et c sont des réels, a≠0 par un ordinateur, comment expliquer à l'ordinateur ce qu'il doit faire pour résoudre ce problème et réaliser cette tâche?

Les étapes à suivre pour réaliser cette tâche par un ordinateur sont :

1. Entrer par le clavier les valeurs des variables a,b et c.
2. Définir l'expression de Delta.
3. Tester la valeur de delta.
   - Si delta>o alors on calcule deux solutions x1 et x2 puis on les affiche à l'utilisateur.
   - Si delta =0 alors on a une seule solution x1 à calculer puis à afficher.
   - Si delta<0 alors on affiche le message 'pas de solution réelle'.

- Chaque étape s'appelle instruction ou action.
- L'ensemble d'étapes ou d'instructions s'appelle Algorithme.

2. Quelques notions de base

Algorithme

Un algorithme est une suite d'instructions permettant la résolution d'un problème donné.

Instruction

Une instruction ou une action est une étape de l'algorithme qui indique à la machine ce qu'elle doit faire (entrer des données, faire des calculs, afficher les résultats, ….).

Programme

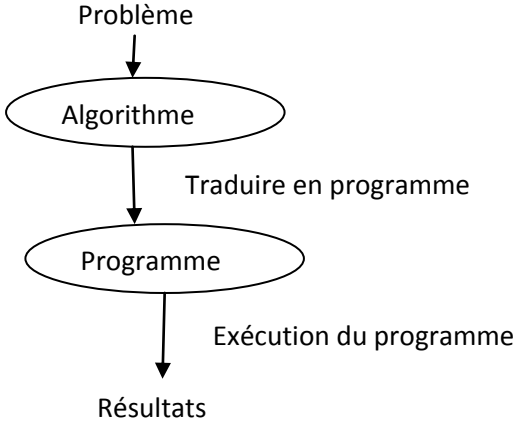
Un programme est la description d'un algorithme dans un langage de programmation. Un programme est une suite d'instructions exécutées par un ordinateur.

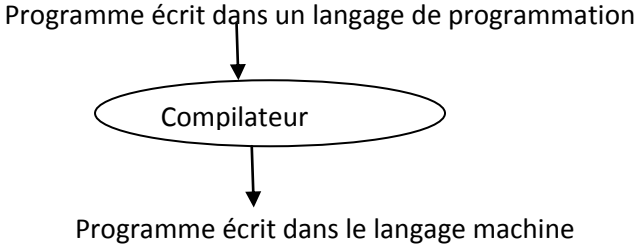
Langage de Programmation (langage évolué)

Un langage de programmation est un ensemble de règles syntaxiques à respecter utilisés pour écrire un programme, Il existe plusieurs langages de programmation, comme C/C++, Pascal, Java, Matlab, *etc.*

Compilateur

Un compilateur est un programme qui traduit d'autres programmes écrits dans un langage évolué en langage machine.





3. Structure d'un algorithme et d'un programme Pascal

Algorithme nom_Algorithme
Déclaration des constantes
Déclaration des variables
Debut

- <instruction 1>
- <instruction 2>
- <instruction 3>
- …….
- <instruction N>;

Fin.

Program nom_program ;
Uses wincrt ;
Déclaration des constantes ;
Déclaration des variables ;
Begin
<Instructions> ;
…..
end.

. Mots clés (Mots réservés)

Les mots clés sont des mots définis dans le langage de programmation, comme **program, for,begin, end, if , var , then, else** …. En Pascal.

. Identificateur

Un identificateur est utilisé pour désigné un objet (nom_objet), il est composés d'une suite de caractère alphanumériques (alphabétiques de [a-z] et [A-Z] et numérique [0-9]) et tiré 8 '_' (trait souligné), il doit commencer par un une lettre et il ne doit pas être un mot clé du langage.

Exemple : x1, Delta, y, min sont des identificateurs correctes ou valides
1k, min-1, begin sont invalides

. Types simples

type	En algorithmique	En pascal	Exemple de valeur
entier	entier	integer	2020, 12, 56, 1, -5 ,-10
reèl	reel	real	99.99
caractère	caractere	char	'a' , '1', '+'
Chaine de caractères	chaine	string	'11', 'bonjour'
logique	booleen	boolean	Vrai, faux en algorithmique True, false en Pascal

3.1. Partie Déclaration

3.1.1 Déclaration de variables

Une variable correspond à un emplacement mémoire désigné par un identificateur dont une valeur est stockée, cette valeur peut être modifiée durant l'exécution du programme. Une variable est déclarée en respectant la syntaxe suivante :

En algorithmique: variable id_variable : type_variable

En Pascal: **Var** id_variable : type_variable ;

Exemple

En Algo : variable x : entier
y:reel

En pascal: **Var** x : integer;
y: real;

3.1.2 Déclaration de constantes

Une constante correspond à un emplacement mémoire désigné par un identificateur dont la valeur ne change pas durant l'exécution du programme. Une constante est déclarée en respectant la syntaxe suivante :

En algorithmique: constante id_constante=valeur_constante

En Pascal: **const** id_constante=valeur_constante;

Exemple

En Algo : constante pi=3.14

En pascal: **const** pi=3.14;

3.2. Partie Instruction

3.2.1 Les opérateurs

Opérateurs arithmétiques	Opérateurs logiques	Opérateurs relationnels
+, -, *	Et ---> and (en pascal)	= ----> =(en pascal)
div(division entière),	Ou ---> or	< ---> <
/ (division réelle),	Non ---> not	> ---> >
Mod (reste de la division entière) ,		≤ ---> <=
-unaire		≥ ---> >=
		≠ ---> <>

Priorité des opérateurs

1 les parenthèses () (()), on commence par les plus internes
2 les fonctions
3 - unaire , non
4 *, /, div, mod, et
5 +, -, ou
6 <, >,=, ≠, ≥, ≤

Quelques fonctions prédéfinies

$|x| \rightarrow abs(x)$
$x^2 \rightarrow sqr(x)$
$\sqrt{x} \rightarrow sqrt(x)$
$e^x \rightarrow \exp(x)$

3.2.2 Les expressions

- Une expression arithmétique est constituée d'opérandes numériques reliés par des opérateurs arithmétiques.

Exemple : 7*5+(8+2-4-4/2)

- Une expression booléenne (ou expression logique) est une expression dont le résultat est de type booléen. Elle peut comporter des opérateurs arithmétiques, des opérateurs de relation et des opérateurs booléens

Exemple : (5+2>8-6)et(7<9)

3.2.3 Instruction d'affectation

Une affectation consiste à mettre une valeur (immédiate, constante, variable ou calculée à travers une expression) dans une variable (espace mémoire).

Syntaxe :

en algo Id_variable <---- valeur
En pascal id_variable:=valeur;

Remarque : Id_variable et valeur doivent avoir le même type

Exemple 1

a ←3;
b ← 4;
c ←1;
Delata ←b*b-4*a*c
i← i+1 (instruction d'incrémentation)
j← j-1 (instruction de décrémentation)

Exemple 2

Soit l'algorithme suivant :

```
Algorithme affectation
Constante a=5
Variables x,y : entier
Debut
X←5
Y←x
X←x+y
Y←y+2*a
fin.
```

```
Program affectation ;
Uses wincrt;
Const a=5 ;
Var x,y: integer;
begin
X :=5;
Y := x;
X := x+y;
Y := y+2*a;
end.
```

Déroulement :

Instructions	x	y
x←5	5	/
y←x	~~5~~	5
X←x+y	10	~~5~~
Y←y+2*a	10	15

3.2.4 Instruction d'entrée sortie (lecture/Affichage)

3.2.4.1 Instruction de lecture : permet d'affecter une valeur tapée à une variable.

Syntaxe :

En algo lire(id-var) , lire(id_var1, id_var2)
En pascal read(id_var); read(id_var1, id_var2);

Exemple :

Lire(x); read(x);
Lire(y,z); read (y,z);

3.2.4.2 Instruction d'affichage: permet d'afficher un message, une valeur d'une variable ou une valeur d'une expression.

Syntaxe :

En algo ecrire(id-var) , ecrire(expression), ecrire('message')
En pascal write (id_var); write(expression); write('message');

Exemple :

ecrire(x); ecrire(x,y,z); ecrire('x=',x); ecrire(x+y); ecrire (' bonjour');

Exercice:

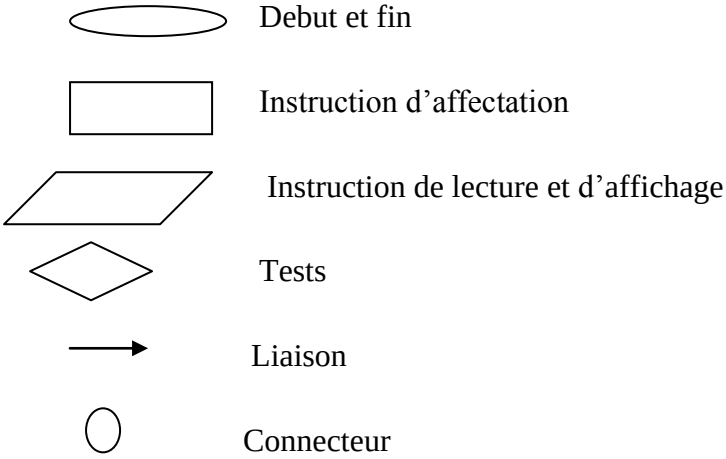
Ecrire un algorithme puis un programme Pascal qui calcule et affiche la somme et la moyenne de deux nombres entiers x et y.

```
Algorithme exercice
Variables x,y , s : entier
          M:reel
Debut
Lire(x,y)
S←x+y
M←s/2
Ecrire(S,M)
fin.
```

```
program exercice ;
Uses wincrt;
Var  x,y , S : integer;
          M:real;
begin
Read (x,y);
S := x+y ;
M:=S/2;
write( S, M);
end.
```

Organigramme : Un organigramme est la représentation graphique des actions d'un algorithme.

- Les symboles utilisés pour construire un organigramme sont:



Debut et fin

Instruction d'affectation

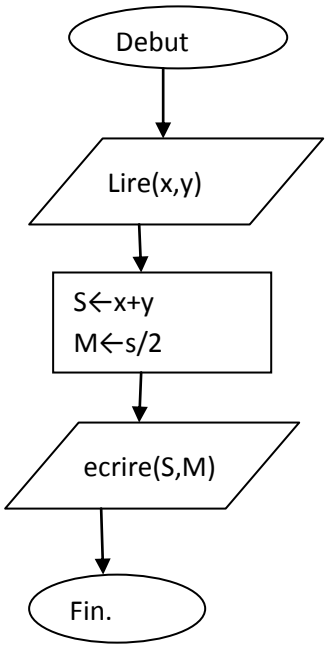
Instruction de lecture et d'affichage

Tests

Liaison

Connecteur

Organigramme exercice



4. Structures de contrôle

4.1 Structures de contrôle conditionnelles

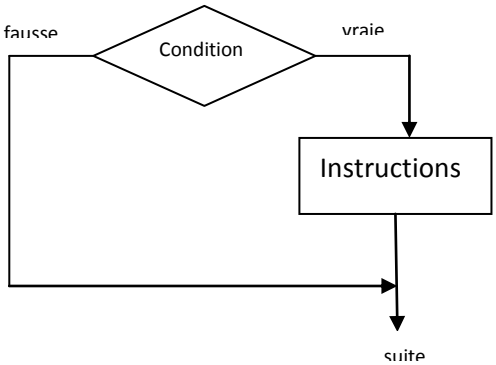
Les structures conditionnelles sont utilisées pour décider de l'exécution d'un bloc d'instructions (est ce que ce bloc sera exécuté ou non), ou pour choisir entre l'exécution de deux blocs différents. Il existe deux types de structures conditionnelles :

4.1.1 Structure conditionnelle simple

Une structure conditionnelle simple contient un seul bloc d'instructions. Selon une condition (expression logique (booleenne)), on décide est ce que le bloc d'instructions sera exécuté ou non. Si la condition est vraie, on exécute le bloc d'instructions, sinon, le bloc d'instructions ne sera pas exécuté. La syntaxe et l'organigramme d'une structure conditionnelle simple sont comme suit :

En Algo **Si <Condition> alors**
<instruction(s)>
finsi;

En Pascal **if <condition> then**
begin
<instruction(s)>;
end;



Remarque : Lorsque le bloc d'instructions est composé d'une seule instruction, les deux mots clés **begin** et **end** sont facultatifs (ne sont pas nécessaires).

Exemple :

```
Algorithme exemple;
Variables  x,y  : entier;
Debut
Lire(x,y);
Si x>y alors
 x←x+10;
Finsi;
y←y-5;
Ecrire(x, ‘  ‘ , y);
Fin.
```

```
program exemple;
Uses wincrt;
Var  x,y : integer;
begin
Read (x,y);
If x>y then
begin
x := x+10 ;
End;
y:=y-5;
write( x, ‘  ‘ , y);
end.
```

Déroulement pour x=7 et y=3

Instructions	x	y	Affichage
Lire(x,y)	7	3	
Si x>y vrai x←x+10	17		
y←y-5		-2	
Ecrire(x, ' ' , y)			17 -2

Déroulement pour x=4 et y=6

Instructions	x	y	Affichage
Lire(x,y)	4	6	
Si x>y faux			
y←y-5		1	
Ecrire(x, ' ' , y)			4 1

Exercice 1: Ecrire un algorithme/programme pascal qui calcule la valeur absolue d'un nombre réel x donné par un utilisateur (sans utiliser la fonction prédéfinie abs).

```
Algorithme exercice1;
Variable  x  : reel;
Debut
Lire(x);
Si x<0 alors
 x← -x;
Finsi;
Ecrire(x);
fin.
```

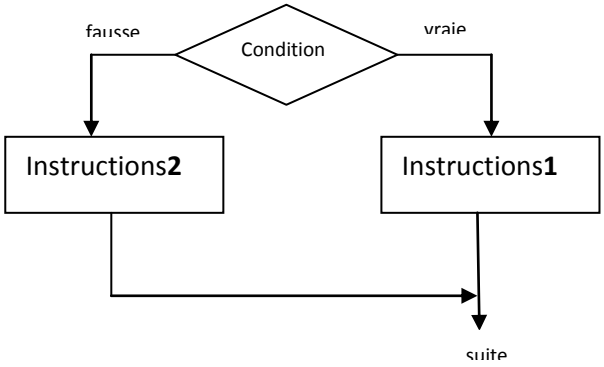
```
program exercice1;
Uses wincrt;
Var  x:real;
begin
Read (x);
If x<0 then
begin
x := -x ;
End;
write( x);
end.
```

4.1.2 Structure conditionnelle alternée (double)

Une structure conditionnelle alternée ou double contient deux blocs d'instructions et une condition. Selon la valeur de la condition, un seul bloc d'instructions sera exécuté.
Si la condition est vraie on exécute le premier bloc, sinon on exécute le second.
La syntaxe et l'organigramme d'une structure conditionnelle double sont comme suit :

En Algo **Si <Condition> alors**
<instruction(s)**1**>
sinon
<instruction(s**)2**>
finsi;

En Pascal **if <condition> then**
begin
<instruction(s)**1**>;
end
else
begin
<instruction(s)**2**>;
end;



Exercice 2 : Ecrire un algorithme /programme qui vérifie si un nombre entier x est pair ou impair**.**

```
Algorithme exercice2;
Variable  x  : entier;
Debut
Lire(x);
Si x mod 2 =0  alors
  ecrire ( x, ' est pair ')
Sinon
 ecrire (x, ' est impair ')
Finsi;
 fin.
```

```
program exercice2;
Uses wincrt ;
Var  x :integer ;
begin
Read (x);
If x mod 2=0 then
  Write (x, ' est pair ')
Else
 write( x, ' est impair ');
end.
```

Exercice 3 : Ecrire un algorithme qui affiche le maximum de deux nombres entiers x et y.

```
Algorithme exercice3;
Variable  x ,y : entier;
Debut
Lire(x,y);
Si x >y alors
  ecrire ( x, ' est le maximum  ')
Sinon
 ecrire (y, ' est le maximum ')
Finsi;
 fin.
```

- **Structure conditionnelle alternée imbriquée**

Syntaxe

En Algo

```
Si <Condition1> alors
    <instruction(s)1>
sinon
    Si <Condition2> alors
        <instruction(s)2>
    sinon
        <instruction(s)3>
    finsi;
finsi;
```

En Pascal

```
if <condition1> then
begin
    <instruction(s)1>;
end
else
begin
    if <condition2> then
    begin
    <instruction(s)2>;
    end
    else
    begin
    <instruction(s)3>;
    end;
end;
```

Exercice 4: écrire un algorithme/programme pascal qui affiche le minimum de trois nombres entiers x,y et z.

```
Algorithme exercice4;
Variable  x ,y ,z: entier;
Debut
Lire(x,y,z);
Si (x <y) et (x<z) alors
  ecrire ( x, ' est le minimum  ')
Sinon
   Si (y <x) et (y<z) alors
     ecrire(y, ' est le minimum')
   sinon
     ecrire(z, 'est le minimum')
  finsi;
Finsi;
Fin.
```

```
Program exercice4;
Uses wincrt;
Var  x ,y ,z: integer;
begin
read(x,y,z);
if (x <y) and (x<z) then
  write ( x, ' est le minimum  ')
else
   if(y <x) and (y<z) then
     write(y, ' est le minimum')
   else
     write(z, 'est le minimum') ;
End.
```

Exercice 5 : Ecrire un algorithme/programme Pascal qui permet de résoudre une équation de 2eme degré $ax^2 + bx + c = 0$ a,b et c sont des réels, $a \neq 0$.

```
Algorithme exo5
Variables a,b,c,delta,x1,x2:reel
Debut
Lire(a,b,c);
Delta← b*b-4*a*c
Si delta>0 alors
 x1 ← (-b-sqrt(delta))/(2*a)
 x2 ← (-b+sqrt(delta))/(2*a)
  ecrire (x1, x2)
Sinon
  Si delta=0 alors
    x1 ← -b/(2*a)
    ecrire (x1)
   sinon
    ecrire('  pas de solution réelle ')
   finsi ;
Finsi;
Fin.
```

```
Program exo5;
Uses wincrt;
Var a,b,c,delta,x1,x2:real;
Begin
Read(a,b,c);
Delta:=b*b-4*a*c;
if delta>0 then
 begin
   x1:=(-b-sqrt(delta))/(2*a);
   x2 :=(-b+sqrt(delta))/(2*a);
   write (x1, x2);
 end
 else
       if delta=0 then
        begin
          x1:= -b/(2*a);
          write (x1);
         end
       else
        write('  pas de solution réelle ');
End.
```

4.2 Structures de contrôle répétitives (itératives)

Les structures itératives permettent d'exécuter plusieurs fois une instruction ou un bloc d'instructions. Ce type de structure est appelé « boucles ». Dans une boucle, le nombre de répétitions (itérations) peut être connu (fixe à l'avance), comme il peut dépendre d'une condition (expression booléenne).

4.2.1 Boucle pour (for)

Dans cette boucle, le nombre d'itérations et connu.

Syntaxe :
En algo

pour <compteur> ← <valeur_initiale> à <valeur_finale> faire
    <instruction(s)>
finPour;

En Pascal

for <compteur>:=<valeur_initiale> **to** <valeur_finale> **do**
begin
  <instruction(s)>;
end;

Fonctionnement

- Pour chaque valeur de la variable <compteur > qui varie de la valeur initiale à la valeur finale avec un pas d'incrémentation égal à 1, le bloc <instruction(s)> sera exécuté.
- Quand la valeur initiale égale à la valeur finale, <instruction(s)> est exécuté et la boucle sera terminée.

Remarques

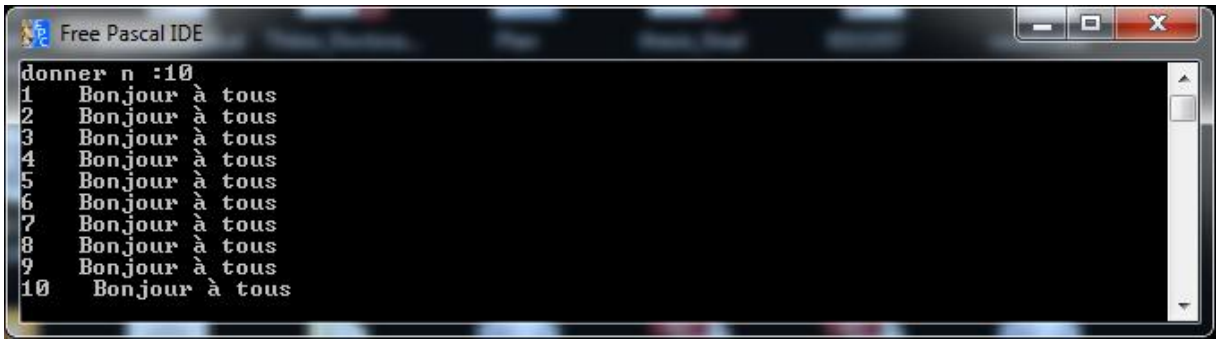
1. <compteur> est une variable de type ordinal (entier ou caractère)
2. Le pas est par defaut de 1.
3. Si le pas est de (-1), la syntaxe en Pascal devient :
   for <compteur>:=<valeur_finale> **downto** <valeur_initiale> **do**
   begin
     <instruction(s)>;
   End ;
4. Le changement de la valeur du compteur est automatique ; le pas est de 1 ou de -1.

Exemple

Algorithme Affichage
Variables i,n :entier
Debut
Ecrire ('donner n :')
lire(n)
Pour i←1 à n faire
Ecrire (i, ' Bonjour à tous')
Finpour ;
Fin.

```
Program Affichage ;
Uses wincrt ;
Var i ,n : integer ;
Begin
Write('donner n :');
Read(n);
For i:= 1 to n do
Begin
Writeln(i, ' Bonjour à tous') ;
End ;
End.
```

A l'exécution on aura le résultat suivant pour n=10 :



Exercice 1 : écrire un algorithme/programme pascal qui calcule somme suivante : $s = \sum_{i=1}^{n} i = 1+2+3+.....+n$

Algorithme somme
Variables i, n, s : entier
Debut
Ecrire ('donner n :') ;
Lire (n)
S ←0 ;
Pour i←1 à n faire
s←s+i
finpour ;
ecrire ('somme =', s) ;
Fin.

```
Program somme;
Uses wincrt ;
Var i ,n, s : integer ;
Begin
Write('donner n :');
Read(n);
S:=0;
For i:= 1 to n do
Begin
S:=s+i ;
End ;
Write(' somme =', s);
End.
```

Déroulement pour n=5

Instructions	n	i	s	Affichage
Ecrire ('donner n :') ;				donner n :
Lire (n)	5			
S ←0 ;			0	
Pour i←1 s←s+i		1	1	
Pour i←2 s←s+i		2	3	
Pour i←3 s←s+i		3	6	
Pour i←4 s←s+i		4	10	
Pour i←5 s←s+i		5	15	
ecrire ('somme =', s) ;				Somme=15

4.2.2 Boucle Tantque

Dans cette boucle, le nombre d'itérations dépend d'une condition, il peut être connu comme il peut dépendre de l'utilisateur.

Syntaxe

En algor

Tantque condition **faire**
    inst1 ;
    inst2 ;
    …
Fintantque ;

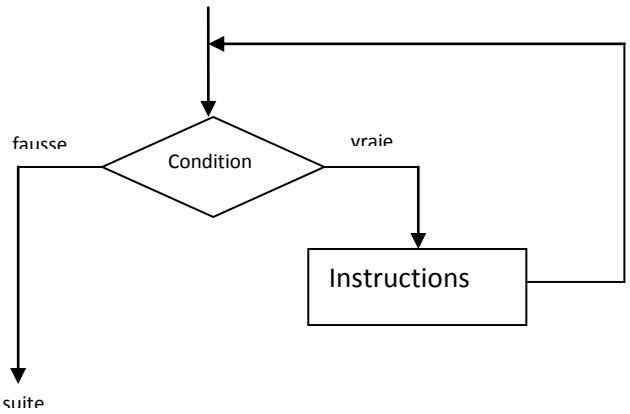
En pascal

While condition **do**
Begin
  Inst1 ;
  Inst2 ;
…..
End ;

Fonctionnement

1- On évalue la condition
2- Si la condition est vraie, on exécute les instructions puis on boucle et on répète 1 et 2 jusqu'à ce que la condition soit fausse.
3- Si la condition est fausse, les instructions ne seront pas exécutées et la boucle sera terminée.

Organigramme



Exercice 2: écrire un algorithme/programme pascal qui calcule somme suivante :

$s = \sum_{i=1}^{n} i = 1+2+3+.....+n$ en utilisant la boucle Tantque.

```
Algorithme somme
Variables i, n, s : entier
Debut
Ecrire ('donner n :')
 Lire (n)
S ←0
 i ←1
Tantque i≤n faire
s←s+i
i←i+1
fintantque;
ecrire ('somme =', s)
Fin.
```

```
Program  somme;
Uses wincrt ;
Var i ,n, s : integer ;
Begin
Write('donner n :');
Read(n);
S:=0;
 i:=1;
while i<= n do
Begin
S:=s+i ;
 i:=i+1;
End ;
Write(' somme =', s);
End.
```

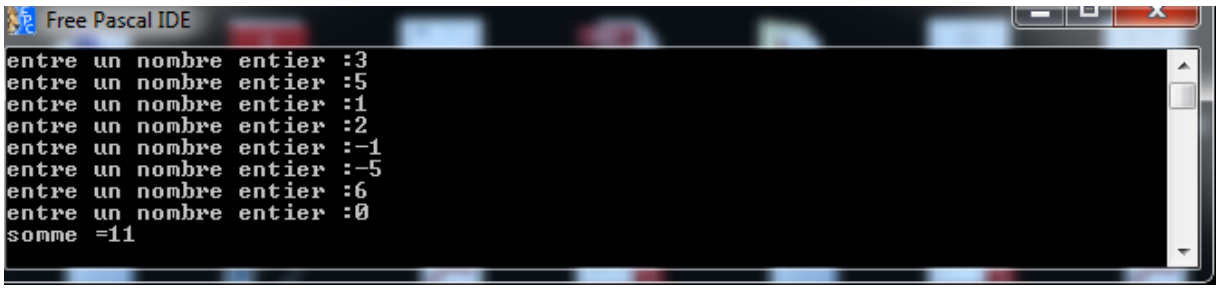
Déroulement pour n=5

Instructions	n	i	s	Affichage
Ecrire ('donner n :') ;				donner n :
Lire (n)	5			
S ←0 ;			0	
i←1		1		
Tantque i≤n vrai s←s+i i←i+1		2	1	
Tantque i≤n vrai s←s+i i←i+1		3	3	
Tantque i≤n vrai s←s+i i←i+1		4	6	
Tantque i≤n vrai s←s+i i←i+1		5	10	
Tantque i≤n vrai s←s+i i←i+1		6	15	
Tantque i≤n faux				
ecrire ('somme =', s) ;				Somme=15

Exercice 3 : écrire un algorithme qui lit une suite de nombres entiers (on ne connait pas leurs nombres) mais la lecture s'arrête dès qu'on introduit un zéro, puis qui calcule et affiche la somme des nombres lus.

```
Algorithme somme
Variables  x, s : entier
Debut
S ←0
Erire (' entrer un nombre entier : ')
 Lire (x)
Tantque  x≠0 faire
s←s+x
Erire ('entrer  un nombre entier : ')
Lire(x)
fintantque;
ecrire ('somme =', s)
Fin.
```

```
Program  somme;
Uses wincrt ;
Var  x, s : integer ;
Begin
S:=0;
Write('entrer un nombre entier :');
Read(x);
while x<> 0 do
Begin
 S:=s+x ;
 Write('entrer un nombre entier :');
Read(x);
End ;
Write(' somme =', s);
End.
```



4.2.3 Boucle répéter

Le nombre d'itérations de cette boucle dépend d'une condition, il peut être connu comme il peut dépendre de l'utilisateur.

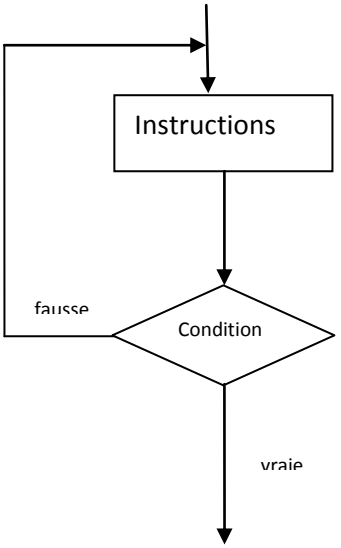
Syntaxe

En algor	**En pascal**
Repeter	**Repeat**
<Instruction (s)>	<Instruction (s)> ;
Jusqu'à < condition> ;	**Until** <condition> ;

Fonctionnement

1- On exécute les instructions.
2- On évalue la condition, Si la condition est fausse, on répète 1 et 2 (on exécute les instructions puis on évalue la condition) jusqu'à ce que la condition soit vraie.
3- Si la condition est vraie, la boucle sera terminée.

Organigramme



Exercice 4: écrire un algorithme/programme pascal qui calcule somme suivante : $s = \sum_{i=1}^{n} i = 1+2+3+.....+n$ en utilisant la boucle repeter.

```
Algorithme somme
Variables i, n, s : entier
Debut
Ecrire ('donner n :')
 Lire (n)
S ←0
 i ←1
repeter
s←s+i
i←i+1
jusqu'à i> n;
ecrire ('somme =', s)
Fin.
```

```
Program  somme;
Uses wincrt ;
Var i ,n, s : integer ;
Begin
Write('donner n :');
Read(n);
S:=0;
 i:=1;
repeat
S:=s+i ;
 i:=i+1;
until i>n;
Write(' somme =', s);
End.
```

Déroulement pour n=5 :

Instructions	n	i	s	Affichage
Ecrire ('donner n :') ;				donner n :
Lire (n)	5			
S ←0 ;			0	
i←1		1		
s←s+i i←i+1 jusqu'à i>n faux		2	1	
s←s+i i←i+1 jusqu'à i>n faux		3	3	
s←s+i i←i+1 jusqu'à i>n faux		4	6	
s←s+i i←i+1 jusqu'à i>n faux		5	10	
s←s+i i←i+1 jusqu'à i>n vrai		6	15	
ecrire ('somme =', s) ;				Somme=15