Université Mohamed Seddik Ben Yahia Jijel

Faculté des Sciences Economiques, Commerciales et Sciences de Gestion

Département des sciences Financières et Comptabilité

Module : Informatique

Algorithmique et Pascal

2ème année LMD, 2021-2022

Réalises par

Enseignante : Hanane layoul

1. Introduction

L'informatique est la science du traitement automatique des informations : le *traitement* est la création, l'ajout, la suppression, la modification, le stockage, etc. ; *automatique* veut dire effectué par machine. L'intérêt de l'ordinateur est sa capacité de manipuler rapidement et sans erreur un grand nombre d'informations : mémoriser des quantités numériques ou alphabétiques, rechercher une quantité mémorisée, comparer ou classer des informations, etc.

Actuellement, l'ordinateur ne peut être utilisé qu'au moyen d'un système d'exploitation. Un système d'exploitation est un ensemble de programmes capables de gérer l'ordinateur. Pour que l'ordinateur puisse résoudre un problème donné, il est nécessaire de lui indiquer la méthode à suivre. Cette méthode est décrite par une suite d'actions à exécuter dans son schéma de fonctionnement. La suite d'action est appelée : un programme informatique qui est exprimé dans un langage de programmation (Pascal, C, Java, …).

2. Définition d'un algorithme

Un algorithme est une suite finie d'instructions nécessaires permettant de résoudre un problème particulier.

- Un algorithme n'est pas un programme.
- Un algorithme est indépendant du langage dans lequel il est implémenté, et de la machine qui exécutera le programme correspondant.
- Il existe plusieurs algorithmes pour résoudre un problème posé.
- Le « langage algorithmique » est un compromis entre un langage naturel et un langage de programmation.
- Une instruction : est un ordre compris par l'ordinateur et qui lui fait exécuté une action

Exemple : lire (A)

3. Programme informatique

Un programme informatique est un algorithme traduit dans un langage de programmation (exemple : Pascal, C, C++, Java,..). Un programme doit être exécuté pour effectuer le traitement souhaité.

• Un programme informatique peut être défini comme une suite d'instructions destinées à être exécutées par un ordinateur.

• Une instruction (ordre) dicte à l'ordinateur l'opération nécessaire qu'il doit effectuer.

4. Programmation

La programmation est la conception d'un algorithme (analyse du problème, le choix de méthode de résolution et le choix des représentations de données) et sa traduction dans un langage de programmation pour être exécuté par une machine.

5. Langage de programmation

Un langage de programmation est un ensemble de termes et de règles de vocabulaire et de grammaire compréhensible par un ordinateur, qui permet de rédiger un programme informatique.

• Un programme écrit dans un langage de programmation est appelé un programme source (ou code source).

• Un programme écrit dans le langage machine est appelé un programme exécutable.

• Un code source ne peut être directement exécuté par la machine.

• Pour être exécuté par une machine, un programme source doit être traduit en langage machine

• Suivant le langage utilisé, un programme peut être interprété ou compilé

Langage interprété: Un programme écrit dans un langage interprété a besoin d'un programme auxiliaire (l'interpréteur) pour traduire au fur et à mesuré les instructions du programme.

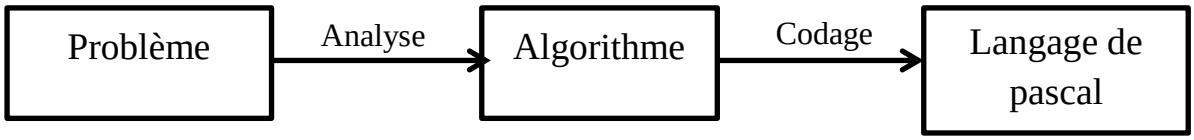
Langage Compilé: Un programme écrit dans un langage compilé va être traduit une fois pour toutes par un programme annexe, appelé compilateur, afin de générer un nouveau fichier contenant des instructions compréhensibles par la machine (c.-`a-d. par le micro-processeur), on dit d'ailleurs que ce fichier est exécutable.

Langage intermédiaire: Certains langages sont en quelque sorte compilés et interprétés, car le programme écrit avec ces langages peut subir une phase de compilation intermédiaire vers un fichier écrit dans un langage qui est diffèrent du programme source et non exécutable (nécessité d'un interpréteur).

6. Langage Pascal

6.1. Définition

Le Pascal est un langage de programmation compilé, il tient son nom du mathématicien français Balaise Pascal, il a été conçu au début des années 70 par N. Wirth, il est l'un des langages de programmation les plus utilisés dans le milieu scolaire pour enseigner les notions de base de la programmation, car c'est un langage facile à apprendre et il possède des instructions très claires et bien structurées. Il est basé essentiellement sur les **algorithmes**.



Comment programmer en Pascal ?

A. Edition: Un programme source Pascal peut être écrit par n'importe quel éditeur de texte ou dans l'éditeur dédie de l'environnement de programmation. Le programme obtenu est stocké dans un fichier avec l'extension (.pas).

B. Compilation: le résultat de cette étape s'appelle le code objet du programme, le code objet ne peut être crée que si le code source ne contient pas d'erreurs syntaxiques. Dans cette étape, toute erreur de nature syntaxique est localisée par le compilateur. Remarque Un programme qui est syntaxiquement correct, ne veut pas dire qu'il est sémantiquement correct (c.-`a-d. il fait ce qu'on attend de lui). Pour s'assurer qu'un programme est sémantiquement correct, il faut l'exécuter plusieurs fois, dans les différentes situations possibles, surtout dans les cas particuliers.

C. Edition de liens: elle est réalisée par un programme auxiliaire, qui s'appelle éditeur de liens et qui intègre dans le fichier objet le code machine de toutes les fonctions utilisées dans le programme et non définies dans le fichier source. L'éditeur de liens génère un fichier exécutable (avec l'extension .exe) qui peut être chargé en mémoire pour être exécuté.

6.2. La forme générale d'un programme pascal

La structure d'un programme PASCAL est définie selon la figure suivante:

a) **En-tête:** permet d'attribuer un nom au programme.
b) **Partie déclarative :** dit au compilateur la quantité de m´mémoire qui doit être réservée au besoin du programme. Elle annonce également tout ce qui va être donc utilisé dans le corps du programme.

c) **Corps du programme (programme principal) :** commence avec la ligne

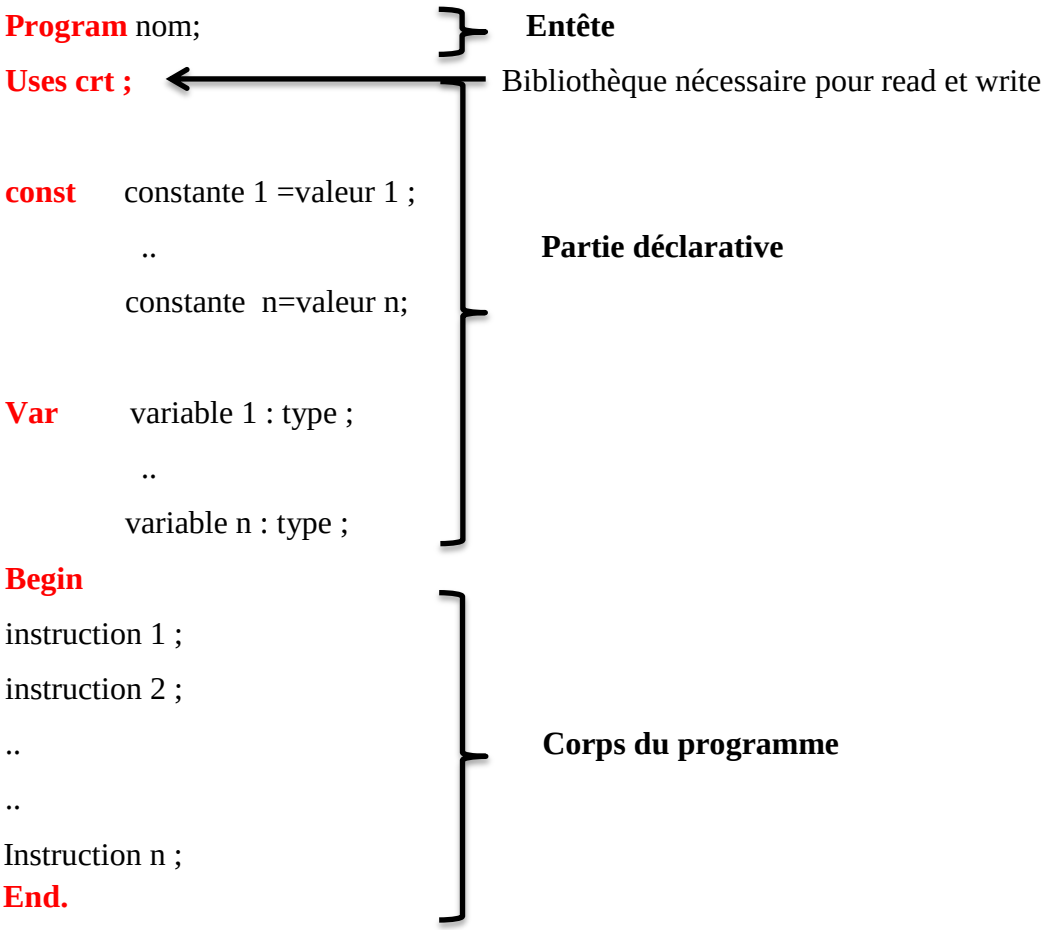
begin

et se termine par la ligne

end.

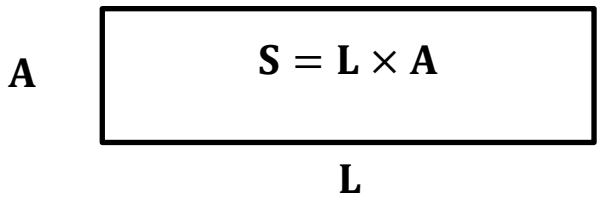
Elle contient tous les instructions nécessaires (exécutables) qui, traduites par le compilateur, seront ensuite exécutées.

Schématiquement on a la structure suivante :



6.3. Exemples de programmes écrits en Pascal

Exemple 1 : écrire un programme pascal qui permet de calculer la **surface** d'un rectangle.




```
Program surface;
Uses crt;
Var   S, L, A: real;
Begin
writeln ('Entrez la longueur L');
readln (L);
writeln ('Entrez la largeur L');
readln (A);
S: = L*A;
Writeln ('la surface est', S);
End.
```

Exemple 2 : écrire un programme pascal qui permet d'afficher sur l'écran la somme de deux nombres obtenus par une lecture au clavier.

```
Program somme;
Uses crt;
Var   A, B, S: real;
Begin
writeln ('Donnez un nombre : ');
readln (A);
Writeln ('Donnez un deuxième nombre : ');
readln (B);
S: = A+B;
Writeln ('la somme des deux nombres =', S);
```

End.

6.4. Déclaration des données:

En pascal on a des données simples et des données structurées.

6.4.1. Les constantes :

Une constante est un objet qui ne peut pas changer de valeur lors de l'exécution d'un programme. Sa déclaration se fait grâce au mot « Const » de la façon suivante :

Const nom de constante = sa valeur ;

• On utilise le symbole = dans les déclarations de constantes.

Exemple:

Const Pi= 3,14 ; TVA= 0,17 ; Max= 44 ;

6.4.2. Les variables: Une variable est un objet qui peut changer de valeur lors de l'exécution d'un programme. Sa déclaration se fait grâce au mot « Var » de la façon suivante :

Var nom de variable : type de variable;

Exemple:

Var a: integer; F= real; d= char;

6.4.3. Les types de variables

Le pascal supporte 5 types de variables de base.

Type	Exemple	Opération
Integer (entier)	3, 40, 200	+, *, /
Real (réel)	-12, 4,6, 9	+, *, /, sin
Char (caractère)	''A'' ''C''	comparaison
String (chaine de caractère)	'' Bonjour Mohamed ''	Comparaison, longueur
Boolean (booléen)	true, false	And, or, not

6.5. Les opérations principales dans le pascal :

Les opérateurs de calcul	**Exemple**
Addition: +	8+3 vaut 11
Soustraction : –	20 - 7= 13
Multiplication : *	4*8= 32
Division réelle : /	7 / 2 vaut 3,5
Division entière: Div	(7) div (2) vaut 3
Reste de la division entière : Mod	(7) mod (2) vaut 1

Les opérateurs logiques	**Exemple**
Supérieur : >	15 > 12
Inferieur : <	7 < 20
Supérieur ou égal : >=	6 >=5
Inférieur ou égal : <=	13<=13
Égal: =	3 =3
N'est pas égal : <>	8 <>5

6.6. Les fonctions prédéfinis (standard) dans le pascal :

Les fonctions prédéfinis (standard)			
Notion mathématique	**Fonction en pascal**	**signification**	**Exemple**
$\|X\|$	Abs $\|X\|$	Valeur absolue de X	Abs $\|-10\|= 10$
X^2	Sqr (X)	Carré de X	Sqr (4)=16
$\sqrt{X}$	Sqrt (X)	Racine de X	Sqrt (25)= 5
Sin (x)	Sin (X)	Sinus de X	
Cos (x)	Cos (X)	Cosinus de X	
e^x	Exp (X)	Exponentiel de X	Exp(0)=1
Log(X)	Ln (X)	Logarithme de X	
Arrondi (X)	Round(X)	Arrondi de X à l'entier le plus proche	Round (8.4)=8

Exemple : écrire un programme qui permet de calculer et afficher le carré d'un nombre A

```
Program square;
 Uses crt;
Var   A, car: integer;
Begin
writeln ('donnez un nombre A');
readln (A);
car:= sqr (A);
Writeln (' le carré de A est', car);
End.
```

6.7. Les instructions de base

6.7.1. L'affectation : l'affectation permet d'attribuer une valeur ou une expression à une variable de **même type**. Elle est réalisée en Pascal grâce à l'opérateur **« := ».**

Exemple 1: On suppose que X est une variable entière

	Avant	Affectation	Après
X	??	X :=8;	X 8
X	8	X :=X+5;	X 13

Si X est de type réel et y de type entier, alors :

X :=Y ; possible

Y :=X ; impossible

Exemple 2: quelle sont les valeurs de variables A, B et C après l'exécution des instructions suivantes :

A: = 10;

B: = 5;

B: = A-3;

C: = B –A;

Solution

	A	B	C
A: = 10;	**10**	**-**	**-**
B: = 5;	**10**	**5**	**-**
B: = A-3;	**10**	**7**	**-**
A: = B-A;	**10**	**7**	**-3**

Les valeurs de A, B et C après exécution sont : **A=10, B= 7, C=-3**

6.7.2. L'instruction d'entrée :

La **lecture** (ou **entrée** d'informations) standard se fait au clavier à l'aide des instructions **read** ou **readln** si le tampon de lecture change après exécution.

Syntaxe:

Read (A);

Read (B);

Read (A, B);

Readln (A);

6.7.3. L'instruction de sortie :

La sortie standard d'information ou écriture se fait à l'écran grâce aux instructions **write** ou **writeln**. Le writeln ajoute un saut de ligne après l'écriture.

Syntaxe :

Write (A);

Writeln (C);

6.8. Structures de contrôle conditionnelles

La structure conditionnelle permet à un programme de modifier son traitement en fonction d'une ou plusieurs conditions, il existe **trois** formes de structures conditionnelles :

- Forme simple
- Forme alternative
- Forme généralisé

6.8.1. Forme simple : La structure conditionnelle est dite à forme simple lorsque le traitement dépond d'une condition. Si la condition est évaluée à « vrai », le programme est exécuté.

Si l'élève aura une moyenne annuelle générale >= 10 **Alors** il passera à l'année suivante. **Si …….. Alors …… Finsi**

Syntaxe:

Analyse et Algorithme	**Pascal**

Si (Condition) **Alors**
Instruction 11
Instruction 12
…..
Instruction 1n
FinSi
	If (Condition) **Then**
	Begin
	Instruction 11;
	Instruction 12;
	…..;
	Instruction 1N ;
	End ;

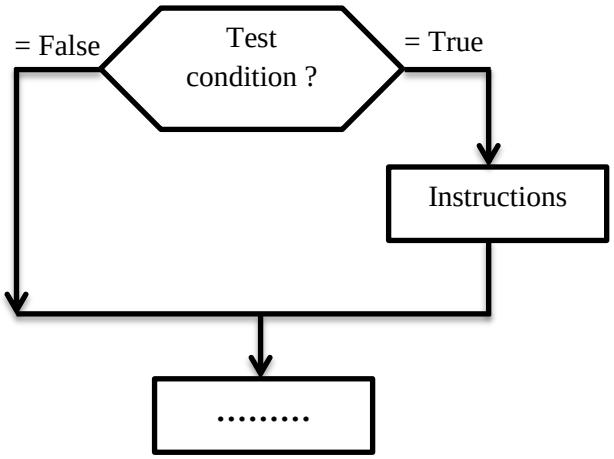
• Chaque traitement comporte un ou plusieurs instructions.

• Dans le cas où le traitement est composé de plusieurs instructions, if faut ajouter les délimites « **Begin** » et « **end** ».

Exécution :

• La condition est tout d'abord ´évaluée,

• les instructions (ou les actions) ne sont exécutées que si la valeur de la condition est égale à True.



6.8.2. Forme alternative: La structure conditionnelle est dite à forme simple lorsque le traitement dépond d'une condition à deux états. Si la condition est évaluée à « vrai » le premier traitement est exécuté. Si la condition est évaluée à « faux », le second traitement est exécuté.

Si il fait beau **Alors** j'irai au Cinéma **Sinon** je reste à la maison.

Si …… **Alors** ……**Sinon**…… **Finsi**

Syntaxe:

Analyse et Algorithme	Pascal

Si (Condition) **Alors**
Instruction 11
Instruction 12
…..
Instruction 1n
Traitement 1
Sinon
Instruction 21
Instruction 22
…..
Instruction 2n
Traitement 2
FinSi
	If (Condition) **Then**
	Begin
	Instruction 11;
	Instruction 12;
	…..;
	Instruction 1N ;
	End
	Else
	Begin
	Instruction 21;
	Instruction 22;
	…..;
	Instruction 2N ;
	End;

Le ; avant else est interdit

Exécution :

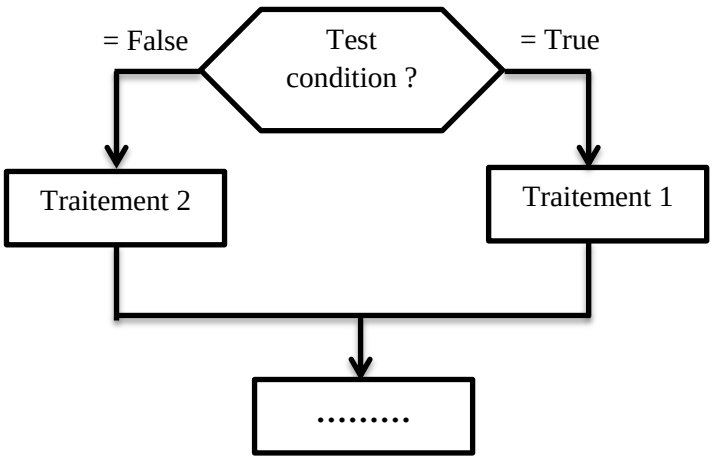
• La condition est tout d'abord évaluée,

• Si le résultat du calcul égale à true,

le bloc d'instructions 1 est exécuté,

• Si le résultat du calcul égale à false,

le bloc d'instructions 2 est exécuté.



Exercice: Ecrire un programme qui permet de calculer et afficher la moyenne d'un étudiant qui contient 4 modules puis l'informer si l'étudiant est admissible ou échoué.

```
Program moyenne;
Uses crt;
Var   N1, N2, N3, N4, Moy: real;
Begin
writeln ('donnez les notes');
readln (N1,N2,N3,N4);
Moy: = (N1+N2+ N3+ N4)/4;
```

```
Writeln (‘ La moyenne est’, Moy);
If   Moy >= 10  then
Writeln (‘Etudiant admissible’)
Else
Writeln (‘Etudiant échoué’);
Readkey;
End.
```

6.8.3. Forme généralisée: La structure conditionnelle est dite à forme généralisée lorsque le problème présente plus que deux cas de traitements possibles. L’exécution d’un traitement entraine automatiquement la non-exécution des autres traitements.

Syntaxe:

Analyse et Algorithme	**Pascal**

Si (Condition 1) **Alors**
Traitement 1
Sinon Si (Condition 2) **Alors**
Traitement 2
Sinon Si (Condition 3) **Alors**
Traitement 3
……..
Sinon Si (Condition N-1) **Alors**
Traitement 3
Sinon Si (Condition N) **Alors**
Traitement N
FinSi
	If (Condition 1) **Then**
	Traitement 1
	Else If (Condition 2) **Then**
	Traitement 2
	Else If (Condition 3) **Then**
	Traitement 3
	……
	Else If (Condition N-1) **Then**
	Traitement N-1
	Else
	Traitement N

Exercice : Ecrire un programme qui demande deux nombres à l’utilisateur et l’informe ensuite si le produit de ces nombres est positif, négatif ou nul.

```
Program produit;
Uses crt;
```

```
Var   A, B: integer;
Begin
writeln (‘ Entrer les deux nombres A et B’);
readln (A, B);
If (A=0) or (B=0) then
Writeln (‘ Le produit est nul’)
Else
If (A>0 and B>0) or (A<0 and B<0) then
Writeln (‘Le produit est positif’)
Else
Writeln (‘Le produit est négatif’) ;
Readkey;
End.
```

6.9. Structures de contrôle itératives (les boucles):

Une Structure de contrôle itérative est une structure qui permet de répéter un traitement plusieurs fois pour les mêmes données.

On distingue trois types de boucles : la boucle **for**, la boucle **while** et la boucle **repeat**.

6.9.1. La boucle ‘For’:

• Cette instruction permet de répéter une séquence d’instructions un nombre fixe de fois.

• On utilise la structure pour quand le nombre d’itération est connu à l’avance.

Syntaxe :

Analyse et Algorithme	**Pascal**

Pour (indice ← inf à sup) **faire**
instructions
Fin pour
	For indice := inf to sup **do**
	Begin
	Instructions;
	End;

• **Indice** est une variable entière, souvent appelée **compteur** de la boucle.

• **inf** et **sup** sont, respectivement, les **bornes inferieures** et **supérieures** de la boucle.

Ceux sont deux constantes entières ou deux variables entières initialisées.

• Le bloc d'instructions est répété pour toutes les valeurs de l'indice compris (inclusivement) entre inf et sup.

• La variable indice est augmentée automatiquement d'une unité à chaque passage dans la boucle.

• Cette forme de boucle est utilisée si on connait d'avance le nombre de fois à répéter les instructions.

• Le bloc d'instructions est exécuté (val-sup - val-inf + 1) fois.

• Si le bloc d'instructions ne contient qu'une seule instruction, le "Begin end" devient inutile.

Example : Ecrire un programme pascal qui permet de calculer la somme des nombres de 1 jusqu'à 6 en utilisant la boucle **for**. S= 1+2+3+ ……6

```
Program somme;
Uses crt;
 Var som, i: integer;
Begin
som: = 0 ;
For i: = 1 to 6 do
som : = som + i ;
Writeln ('la somme est', som) ;
Readkey;
End.
```

6.9.2. La boucle 'While':

• Cette structure permet la répétition d'une ou plusieurs instructions tant qu'une condition est satisfaite.

Syntaxe :

Analyse et Algorithme	Pascal

Tant que (condition) **faire**
Instructions
Fin tant que
	While (condition) **do**
	Begin
	Instructions;
	End;

• Condition est une expression booléenne.

• Le bloc d'instructions est répété, tant que la condition est vraie.

• Si la condition est fausse dès le début, le bloc d'instructions n'est jamais exécuté.

• Cette forme de boucle est utilisée si au moment où on écrit la boucle, on ne connait pas le nombre de fois à répéter le bloc d'instructions.

• Les instructions d'une boucle while continuent de s'exécuter tant que la condition n'est pas fausse. Pour éviter une boucle infinie, il faut impérativement ajouter une instruction qui rend la condition fausse à un moment donné.

• Aussi, si le bloc d'instructions ne contient qu'une seule instruction, le "Begin end" devient inutile.

Example : Ecrire un programme pascal qui permet de calculer la somme des nombres de 1 jusqu'à 6 en utilisant la boucle **while**. S= 1+2+3+ ……6

```
Program somme;

Uses crt;

Var  som, i : integer ;

Begin

Som: = 0;

i: = 1 ;
```

while I < = 6 do

Begin

Som: = som + i;

i:= i + 1;

End;

Writeln (‘ la somme est’, som) ;

Readkey;

End.

التنفيد

i= 1 s = s + 1 = 0 + 1 = 1

i = 2 s = s + 2 = 1 + 2 = 3

i = 3 s = s + 3 = 3 + 3 = 6

i = 4 s = s + 4 = 6 + 4 = 10

i= 5 s = s + 5 = 10 + 5 = 15

i=6 s = s + 6= 15 + 6 =21

6.9.3 La boucle ‘Repeat’ :

Cette instruction permet de répéter une séquence d’instructions tant qu’une condition est fausse.

Syntaxe :

Analyse et Algorithme	Pascal
Répéter **Instruction(s);** **Jusqu’à (Condition)**	**Repeat** **instruction(s);** **Until (Condition);**

• Les instructions de la boucle sont répétées jusqu’à ce que la valeur de la condition égale `a vrai.

• Même si la condition est vraie dès le début, les instructions sont au moins exécutées une fois.

• Pas besoin d’encadrer les instructions par un ”begin end”, le ”repeat until” joue déjà ce rôle.

Exemple : Ecrire un programme pascal qui permet de calculer la somme des nombres de 1 jusqu'à 6 en utilisant la boucle **repeat**. S= 1+2+3+ ……6

```
Program   somme;
Uses crt;
Var    som , i : integer ;
Begin
som: = 0 ;
i : = 1;
Repeat
Som: = som + i;
i = i + 1
Until ( i >  6 ) ;
Writeln (' la somme est', som) ;
End.
```