Université de Bejaia
Faculté de Technologie
Département de Technologie

Année Universitaire 2021-2022

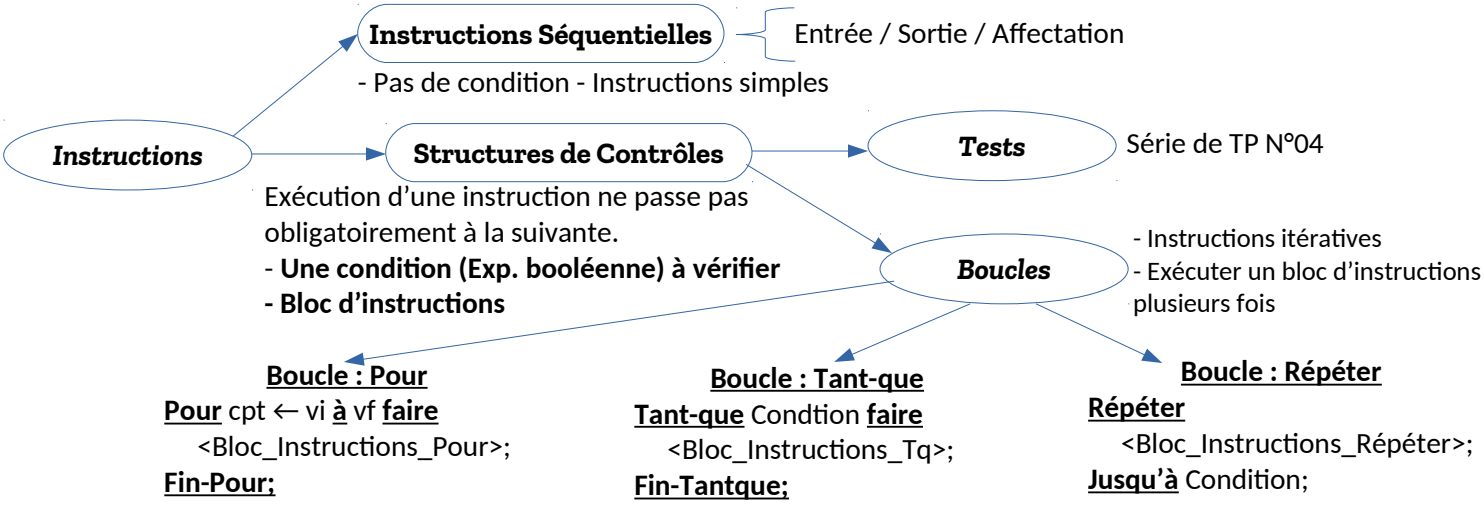
Informatique 1 - Série de TP N°05

Sommaire

Adapté par: Redouane OUZEGGANE
rouzeggane@gmail.com - redouane.ouzeggane@univ-bejaia.dz

TP INFORMATIQUE 1

SÉRIE DE TP N°05 (BOUCLES : POUR - TANT-QUE - RÉPÉTER)



EXERCICE N°01 : ALGORITHMES → PROGRAMME

Soit l'algorithme suivant :

```
Algorithme Exo1;
  Variables
      N, i : entier;
      S : réel;
Début
    {Entrées}
    Lire(N) ;

    {Traitement}
    S ← 0;
    Pour i←1 à N  Faire
        S ← S + (2*i)/(2*i+1);
    Fin-Si

    {Sorties}
    Écrire('La somme S = ', S:0:3) ;
Fin.
```

Questions

1- Traduire l'algorithme en Programme PASCAL, puis compiler et exécuter le programme N = 4?

2- Dérouler le programme pour les N = 4 ?

3- Déduire l'expression générale du résultat S en fonction de N ?

4- Ré-écrire l'algorithme (Programme) en remplaçant la boucle ***Pour*** par la boucle ***Tant-que***.

5- Ré-écrire l'algorithme (Programme) en remplaçant la boucle ***Pour*** par la boucle ***Répéter***.

6- Pour la boucle ***Tant-que***, changer la variable *i* par une variable *j* qui varie de 2 jusqu'à (2*N) avec un pas = 2 ?

Exercice N°02 : Sommes, Produits, ...

- Écrire un algorithme / programme Pascal pour chaque cas suivant :

1) Calculer la somme S = $1^2 + 3^2 + 5^2 + ... + (2*N+1)^2$

2) Calculer le produit P = 1*2*3* ... * N

3) Calculer la somme S = $x + x^2 + x^3 + ... + x^N$

4) Calculer la somme S = $x + x^3/2 + x^5/4! + x^7/6! +$ ... (Nième Terme)

5) Calculer la somme S = $x - x^2 + x^3 - ... \pm x^N$

6) Afficher la table de multiplication d'un entier N entre 1 et 10 (contrôler la valeur de N).

7) Soit A et B deux entier tel-que A < B. Introduire N valeurs entières, et réaliser la somme de valeurs pairs non-nuls et le produit des valeurs impaires.

Solution

EXERCICE N°01 : ALGORITHMES → PROGRAMME

1- Traduire l'algorithme en Programme PASCAL puis compiler et exécuter le programme

Algorithme

```
Algorithme TP5_Exo1;
  Variables
       N, i : entier;
       S : réel;
Début
    {Entrées}
    Lire(N) ;

    {Traitement}
    S ← 0;
    Pour i←1 à N Faire
        S ← S + (2*i) / (2*i+1);
    Fin-Pour

    {Sorties}
    Écrire('La Somme S =  ', S:0:3);
Fin.
```

Programme PASCAL

```
Program TP5_Exo1;
  Var
       N,i : integer;
       S : real;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    {Traitement}
    S := 0;
    For  i := 1 to N do
    begin
        S := S + (2*i) / (2*i+1);
    end;

    {Sorties}
    Writeln('La Somme S = ', S:0:3);
End.
```

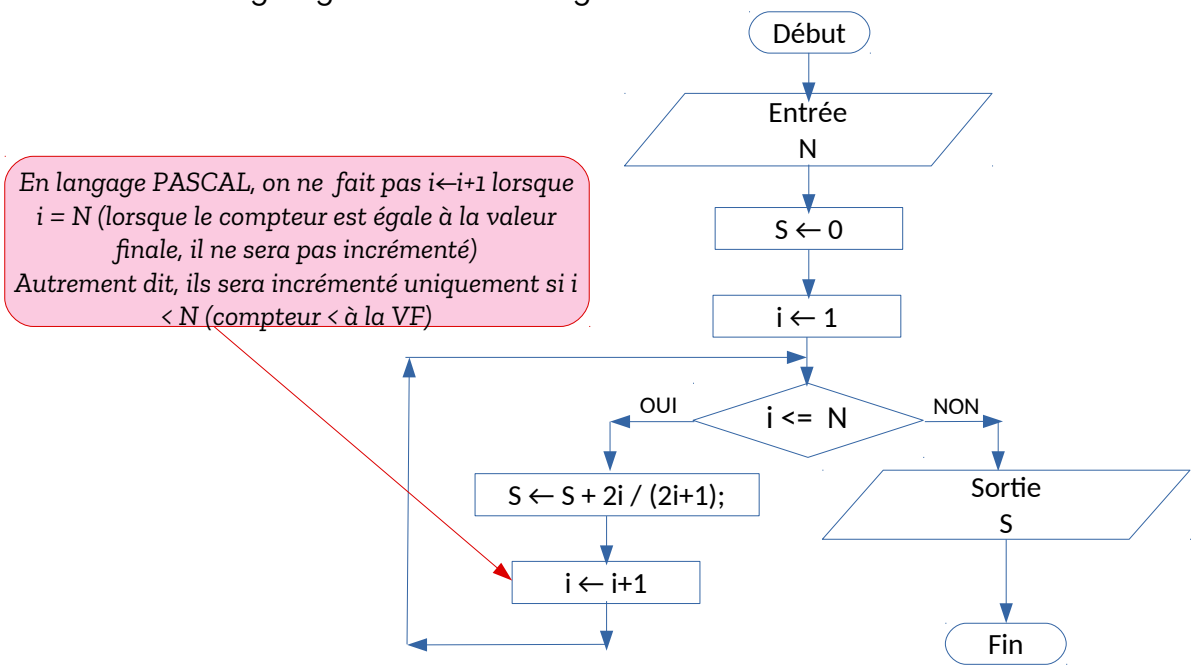
Puisqu'il y a une seule instruction dans l bloc, le **Begin** et **End;** sont facultatifs (optionnels). On alors peut les enlever, .

Vous pouvez exécuter le programme en ligne sur le lien : https://onlinegdb.com/iFU8blcZG

Dans le programme précédent, la boucle For contient une seule instruction, donc, comme mentionné dans le programme ci-dessus, les mots clés **begin** et **end;** de cette boucle sont facultatifs, on peut alors :

```
for  i:=1 to N  do
     S := S + (2*i) / (2*i+1);
```

Pour l'organigramme de cet algorithme :



2- Dérouler l'algorithme / programme Pour N=4

Le déroulement est l'exécution manuelle des instructions et l'affichage de l'impacte de ces instructions sur les valeurs variables de l'algorithme (ou le programme). À travers le déroulement, nous visualisons l'évolution des valeurs des différentes variables, et ça nous permet d'expliquer ou de comprendre le fonctionnement de l'algorithme.

Instructions	***Variables***		
	N	***i***	***S***
Lire (N);	4	/	/
S ← 0;	''	/	0
Pour i=1 S ← S + (2*i) / (2*i+1) = 0 + 2/3 Fin-Pour → automatiquement i devient 2	''	1	2/3 = 0.666
Pour i=2 S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 = 22/15 Fin-Pour → automatiquement i devient 3	''	2	2/3+4/5 = 22/15
Pour i=3 S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 + 6/7 Fin-Pour → automatiquement i devient 4		3	2/3 + 4/5 + 6/7
Pour i=4 S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 + 6/7 + 8/9 Fin-Pour → (en PASCAL, puisque c'est dernière itération de la boucle Pour, i reste 4 (la valeur finale de i) et on quitte la boule)		4	2/3 + 4/5 + 6/7 + 8/9
⇒ Écrire('La somme S = ', S:0:3);	La somme S = 2/3 + 4/5 + 6/7 + 8/9		

3- Déduire l'expression générale du résultat S en fonction de N

À travers le déroulement, nous pouvons aussi déduire l'expression générale du résultat calculé par l'algorithme (*calcul itératif* : par boucle), en fonction des variables d'entrée.

Dans cet exercice, nous avons une seule variable d'entrée qui est N, et une seule variable de sortie qui est S.

Dans le déroulement ci-dessus, nous avons :

Pour i = 1, nous avons S = 2/3 = 2*1 / (2*1+1)

Pour i= 2, nous avons S = 2/3 + 4/5 = 2*1 / (2*1+1) + 2*2 / (2*2+1)

Pour i= 3, nous avons S = 2/3 + 4/5 + 6/7 = 2*1 / (2*1+1) + 2*2 / (2*2+1) + 2*3 / (2*3+1)

…

Pour i quelconque, nous aurons S = 2*1 / (2*1+1) + 2*2 / (2*2+1) + … + 2*i / (2*i+1)

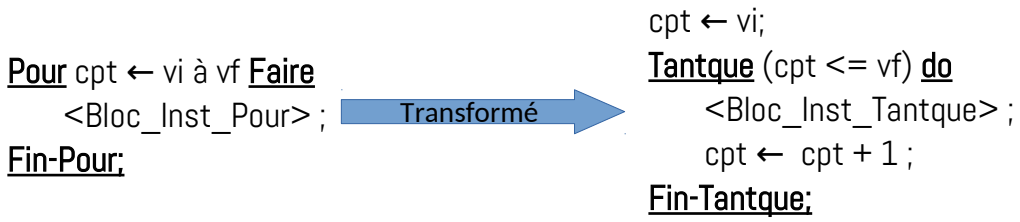
Ainsi, pour i = N, nous aurons : S = S = 2*1 / (2*1+1) + 2*2 / (2*2+1) + … + 2*N / (2*N+1)

On peut généraliser par la formule suivante :

$$S=\sum_{i=1}^{N}\frac{2\times i}{2\times i+1}$$

4- Ré-écrire l'algorithme en remplaçant la boucle Pour par la boucle Tant-que

La boucle Pour possède un compteur, une valeur initiale et une valeur finale. Par contre, la boucle Tant-que possède une condition.



Telque :
ctp : compteur(varible entière)
vi : valeur initiale
vf : valeur finale

En appliquant ceci, nous aurons l'algorithme et le programme suivants :

Algorithme

```
Algorithme TP5_Exo1;
  Variables
      N, i : entier;
      S : réel;
Début
    {Entrées}
    Lire(N) ;

    {Traitement}
    S ← 0;
    i ← 1;
    Tant-que i <= N Faire
        S ← S + (2*i) / (2*i+1);
        i ← i + 1;
    Fin-Tantque

    {Sorties}
    Écrire('La Somme S =  ', S:0:3);
Fin.
```

Programme PASCAL

```
Program TP5_Exo1;
  Var
      N,i : integer;
      S : real;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    {Traitement}
    S := 0;
    i := 1;
    While  i  <= N do
    begin
        S := S + (2*i) / (2*i+1);
        i := i + 1;
    end;

    {Sorties}
    Writeln('La Somme S = ', S:0:3);
End.
```

Pour cette version du programme, voir le lien suivant :
https://onlinegdb.com/IOZa01nyl

Organigramme & Déroulement

Pour l'organigramme de la boucle **Tant-que**, il sera le même que celui de la boucle Pour (voir la page 3).

Pour le déroulement d'algorithme avec la boucle Tant-que, et en prenant la valeur de N = 4, nous aurons le tableau suivant :

Instructions	***Variables***		
	N	***i***	***S***
Lire (N);	4	/	/
S ← 0;	''	/	0
i ← 1;		1	
Tant-que i<=N ⇒ 1<=4 est **True**(Vrai)⇒on fait la boucle S ← S + (2*i) / (2*i+1) = 0 + 2/3 = 2/3 i ← i + 1 = 1 + 1 = 2 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)	''	2	2/3 = 0.666
Tant-que i<=N ⇒ 2<=4 est **True**(Vrai)⇒on fait la boucle S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 i ← i + 1 = 2 + 1 = 3 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)	''	3	2/3+4/5 = 22/15
Tant-que i<=N ⇒ 3<=4 est **True**(Vrai)⇒on fait la boucle S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 + 6/7 i ← i + 1 = 3 + 1 = 4 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)		4	2/3 + 4/5 + 6/7
Tant-que i<=N ⇒ 4<=4 est **True**(Vrai)⇒on fait la boucle S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 + 6/7 + 8/9 i ← i + 1 = 4+ 1 = 5 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)		5	2/3 + 4/5 + 6/7 + 8/9
Tant-que i<=N ⇒ 5<=4 est **False**(Faux)⇒on ne fait pas la boucle ⇒ quitter la boucle tant-que			
⇒ Écrire('La somme S = ', S:0:3);	La somme S = 2/3 + 4/5 + 6/7 + 8/9		

Remarques :

-) Dans les deux déroulements précédents (le déroulement de la boucle **Pour** à la *page 4*, et le déroulement de la boucle **Tant-que** *ci-dessus*), la valeur finale de la variable i est différente :

***)** Dans la boucle **Pour**, la valeur finale de i est 4 (c-à-d la valeur de N)

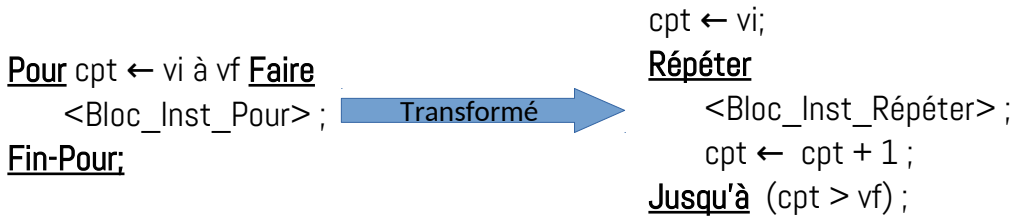
***)** Dans la boucle **Tant-que**, la valeur finale de i est 5 (c-à-d la valeur de N+1)

-) La boucle pour possède un compteur, une valeur initiale et une valeur finale. Le compteur est une *variable entière*, les valeurs initiale et finale sont des valeurs entières.

-) La boucle **Tant-que** est plus puissante que la boucle **Pour** : tout problème qui peut être résolu par **Pour** peut être aussi résolue par la boucle **Tant-que**, et l'inverse n'est pas toujours correct.

5- Ré-écrire l'algorithme en remplaçant la boucle Pour par la boucle Répéter

De la même façon que la boucle Tant-que, la boucle Répéter possède une condition. Sauf que pour cette dernière (la boucle répéter), la condition représente la condition d'achèvement (terminaison) de la boucle. Voici le modèle de transformation de la boucle Pour vers la boucle Répéter :



Telque :
ctp : compteur(varible entière)
vi : valeur initiale
vf : valeur finale

En appliquant ceci, nous aurons l'algorithme et le programme suivants :

Algorithme

```
Algorithme TP5_Exo1;
  Variables
      N, i : entier;
      S : réel;
Début
    {Entrées}
    Lire(N) ;

    {Traitement}
    S ← 0;
    i ← 1;
    Répéter
        S ← S + (2*i) / (2*i+1);
        i ← i + 1;
    Jusqu'à i > N;

    {Sorties}
    Écrire('La Somme S =  ', S:0:3);
Fin.
```

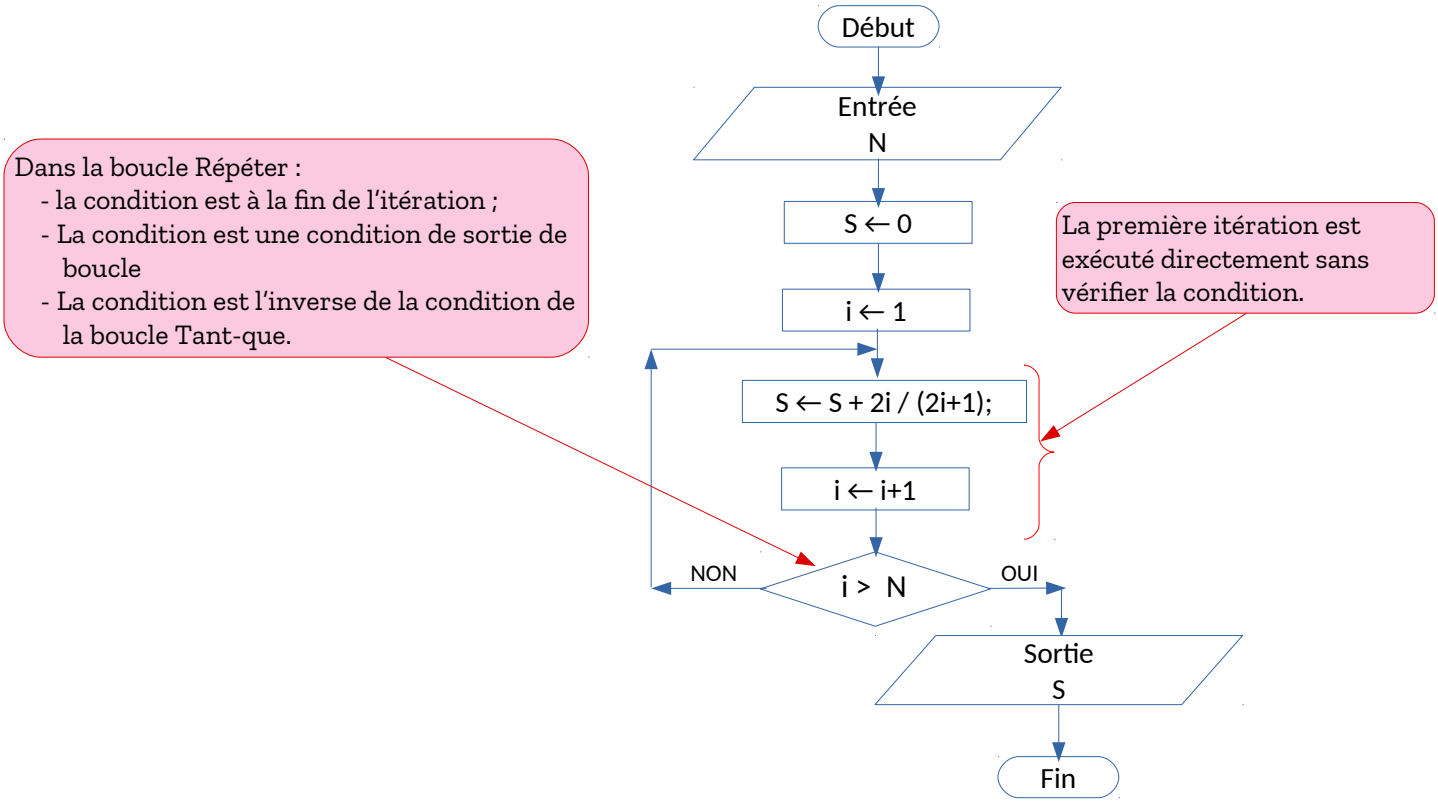
Programme PASCAL

```
Program TP5_Exo1;
  Var
      N,i : integer;
      S : real;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    {Traitement}
    S := 0;
    i := 1;
    Repeat
        S := S + (2*i) / (2*i+1);
        i := i + 1;
    Until  i > N;

    {Sorties}
    Writeln('La Somme S = ', S:0:3);
End.
```

Pour cette version du programme, voir le lien suivant : https://onlinegdb.com/1Tw8LpiVx

Organigramme & Déroulement

Nous allons voir, dans ce qui suit, l'organigramme (logigramme) de l'algorithme ainsi que son déroulement, avec la boucle Répéter. Ceci permet à voir la différence entre les boucles **Pour**, **Tant-que** et **Répéter**.



Pour le déroulement pour N = 4 :

Instructions	***Variables***		
	N	***i***	***S***
Lire (N);	4	/	/
S ← 0;	''	/	0
i ← 1;		1	
Répéter S ← S + (2*i) / (2*i+1) = 0 + 2/3 = 2/3 i ← i + 1 = 1 + 1 = 2 **Jusqu'à** i>N ⇒ i>N ⇒ 2>4 est **False** ⇒ on ne quitte pas la boucle	''	2	2/3 = 0.666
Répéter S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 i ← i + 1 = 2 + 1 = 3 **Jusqu'à** i>N ⇒ i>N ⇒ 3>4 est **False** ⇒ on ne quitte pas la boucle	''	3	2/3+4/5 = 22/15
Répéter S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 + 6/7 i ← i + 1 = 3 + 1 = 4 **Jusqu'à** i>N ⇒ i>N ⇒ 4>4 est **False** ⇒ on ne quitte pas la boucle		4	2/3 + 4/5 + 6/7
Répéter S ← S + (2*i) / (2*i+1) = 2/3 + 4/5 + 6/7 + 8/9 i ← i + 1 = 4+ 1 = 5 **Jusqu'à** i>N ⇒ i>N ⇒ 5>4 est **True** ⇒ on quitte la boucle		5	2/3 + 4/5 + 6/7 + 8/9
⇒ Écrire('La somme S = ', S:0:3);	La somme S = 2/3 + 4/5 + 6/7 + 8/9		

6- Utiliser la boucle Tant-que avec un pas = 2

Nous avons trouvé l'expression générale du résultat S, dans la question N°3, comme suit :

$$S=\sum_{i=1}^{N}\frac{2\times i}{2\times i+1}$$

Nous savons que l'indice i (le compteur i) varie avec un Pas = 1. C'est à dire que i prendra les valeurs : 1, 2, 3, ..., N. Nous effectuons un changement de variable comme suit :

j = 2*i.

Ainsi, la somme S devient :

$$S=\frac{2}{2+1}+\frac{4}{4+1}+\ldots+\frac{2\times N}{2\times N+1}=\sum_{j=2/pas=2}^{2\times N}\frac{j}{j+1}$$

Avec un pas = 2 pour la variable j. La variable j prendra les valeurs suivantes :

$$2,\ 4,\ 6,\ \ldots,\ 2N$$

Nous aurons cette transformation :

```
S ← 0;
i ← 1;
Tant-que i <= N Faire
    S ← S + 2*i / (2*i+1);
    i ← i + 1;
Fin-Pour;
```

Transformé

```
S ← 0;
J ← 2;
Tant-que j <= (2*N) Faire
    S ← S + J / (J+1);
    J ← J + 2;
Fin-Pour;
```

En appliquant ceci, nous aurons l'algorithme et le programme suivants :

Algorithme

```
Algorithme TP5_Exo1;
  Variables
      N, j : entier;
      S : réel;
Début
    {Entrées}
    Lire(N) ;

    {Traitement}
    S ← 0;
    j ← 2;
    Tant-que (j <= 2*N) faire
        S ← S + j / (j+1);
        j ← j + 2;
    Jusqu'à i > N;

    {Sorties}
    Écrire('La Somme S =  ', S:0:3);
Fin.
```

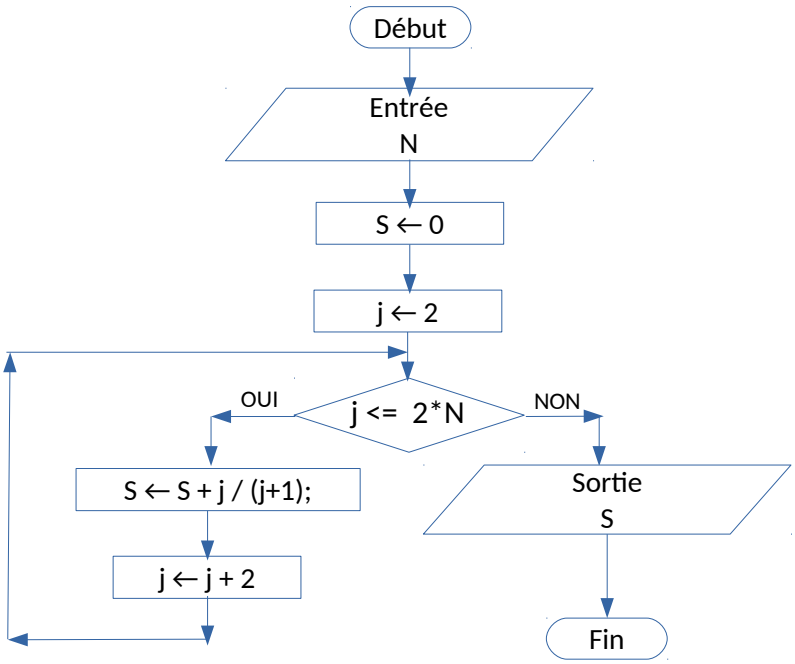
Programme PASCAL

```
Program TP5_Exo1;
  Var
      N, j : integer;
      S : real;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    {Traitement}
    S := 0;
    j := 2;
    While  j <= (2*N) do
    begin
        S := S + j / (j+1);
        j := j + 2; {Pas = 2}
    end;

    {Sorties}
    Writeln('La Somme S = ', S:0:3);
End.
```

Pour cette version du programme, voir le lien suivant : https://onlinegdb.com/SitPrNWCO

Organigramme & Déroulement

Pour l'organigramme de la boucle Tant-que avec un Pas = 2, ça sera comme suit :



Pour le déroulement, avec N = 4, ça sera :

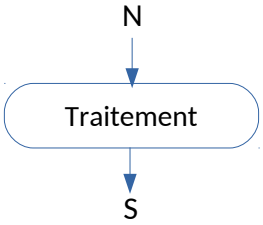
Instructions	***Variables***		
	N	***j***	***S***
Lire (N);	4	/	/
S ← 0;	''	/	0
j ← 2;		2	
Tant-que j<=2*N ⇒ 2<=8 est **True**(Vrai)⇒on fait la boucle S ← S + j/ (j+1) = 0 + 2/3 = 2/3 j ← j + 2 = 2 + 2 = 4 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)	''	4	2/3 = 0.666
Tant-que j<=2*N ⇒ 4<=8 est **True**(Vrai)⇒on fait la boucle S ← S + j/ (j+1) = 2/3 + 4/5 j ← j + 2 = 4 + 2 = 6 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)	''	6	2/3+4/5 = 22/15
Tant-que j<=2*N ⇒ 6<=8 est **True**(Vrai)⇒on fait la boucle S ← S + j/ (j+1) = 2/3 + 4/5 + 6/7 j ← j + 2 = 6 + 2 = 8 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)	''	8	2/3 + 4/5 + 6/7
Tant-que j<=2*N ⇒ 8<=8 est **True**(Vrai)⇒on fait la boucle S ← S + j/ (j+1) = 2/3 + 4/5 + 6/7 + 8/9 j ← j + 2 = 8 + 2 = 10 Fin-Tantque ⇒ essayer de refaire l'itération (boucler)	''	10	2/3 + 4/5 + 6/7 + 8/9
Tant-que j<=2*N ⇒10<=8 est **False**(Faux)⇒on ne fait pas la boucle ⇒ quitter la boucle tant-que	''	''	''
⇒ Écrire('La somme S = ', S:0:3);	La somme S = 2/3 + 4/5 + 6/7 + 8/9		

EXERCICE N°02 : SOMMES, PRODUITS, DIVERS PROBLÈMES DE BOUCLE

1- Calculer la somme S = $1^2 + 3^2 + 5^2 + ... + (2*N+1)^2$

Analyse du problème

La première étape est de recenser les différentes variables de l'algorithme à réaliser, les variables d'entrée et les variables de sortie. L'algorithme doit calculer la variable S, donc S est une variable de sortie. Pour calculer S nous devons savoir la valeur de N, donc, N est une variable d'entrée, comme illustré dans la figure ci-dessous :



La deuxième étape est d'écrire la somme S sous format abrégée, comme suit :

$$S = 1^2 + 3^2 + 5^2 + 7^2 + ... + (2*N+1)^2$$

Nous savons que chaque nombre impair s'écrit sous forme (2*k+1) / k est un entier, donc on aura :

$$S = (2*0+1)^2 + (2*1+1)^2 + (2*2+1)^2 + (2*3+1)^2 + ... + (2*N+1)^2.$$

La forme abrégée de cette somme est :

$$S = \sum_{i=0}^{N} (2*i+1)^2$$

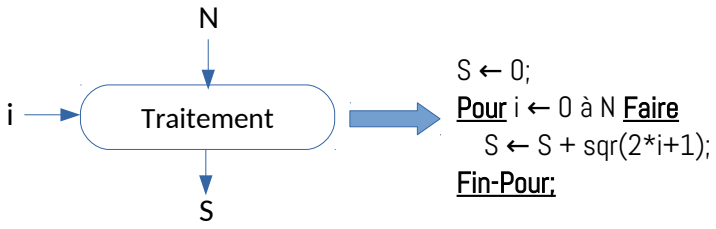
On remarque que :

pour i=0 ⇒ $(2*i+1)^2 = 1^2$
pour i=1 ⇒ $(2*i+1)^2 = 3^2$
pour i=2 ⇒ $(2*i+1)^2 = 5^2$
pour i=3 ⇒ $(2*i+1)^2 = 7^2$
....
pour i=N ⇒ $(2*i+1)^2 = (2*N+1)^2$

L'égalité : $S = \sum_{i=0}^{N} (2*i+1)^2$ devient en algorithmique comme suit :

```
S ← 0;
Pour i ← 0 à N Faire
   S ← S + sqr(2*i+1);
Fin-Pour;
```

Selon l'algorithme ci-dessus, nous aurons besoin d'une variable i, comme compteur pour la boucle **Pour**. Le schéma Entrée / Traitement / Sortie devient :



Lorsque on regroupe toute l'analyse précédente du problème, nous aurons l'algorithme suivant :

Algorithme

```
Algorithme TP5_Exo2_Q01;
  Variables
      N, i, S : entier;
Début
    {Entrées}
    Lire(N) ;

    {Traitement}
    S ← 0;
    Pour i ← 0 à N faire
        S ← S + sqr(2*i+1);
    Fin-Pour;

    {Sorties}
    Écrire(S);
Fin.
```

Programme PASCAL

```
Program TP5_Exo2_Q01;
  Var
      N, i, S : integer;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    {Traitement}
    S := 0;
    For  i:=0 to N  do
        S := S + sqr(2*i+1);

    {Sorties}
    Writeln('La Somme S = ', S);
End.
```

Le programme Pascal ci-dessus est disponible sur le lien : https://onlinegdb.com/NOvDwxcVp

2- Calculer le produit P = 1*2*3* … * N

Analyse du problème

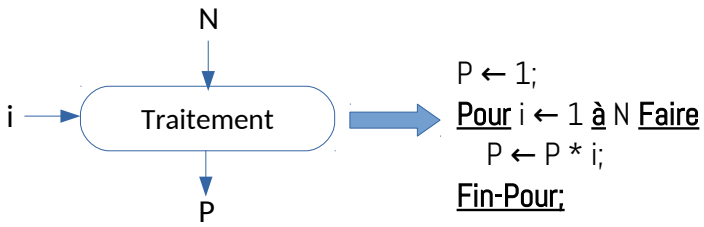
La forme abrégée du produit P est :

$$P = 1 * 2 * .... *N = \prod_{i=1}^{N} i$$

La formule $P = \prod_{i=1}^{N} i$ devient en algorithmique comme suit :

```
P ← 1;
Pour i ← 1 à N Faire
   P ← P * i;
Fin-Pour;
```

Et le schéma Entrée / Traitement / Sortie sera comme illustré dans la figure ci-dessous :



N.B. : Le produit P = 1*2*3*….*N est dit la factoriel de N, et on écrit P = N!

En exploitant le schéma d'entrée/Traitement/Sortie, on aura l'algorithme et le programme pascal suivant :

Algorithme

```
Algorithme TP5_Exo2_Q02;
  Variables
      N, i, P : entier;
Début
    {Entrées}
    Lire(N) ;

    {Traitement}
    P ← 1;
    Pour i ← 0 à N faire
        P ← P * i;
    Fin-Pour;

    {Sorties}
    Écrire(P);
Fin.
```

Programme PASCAL

```
Program TP5_Exo2_Q02;
  Var
      N, i, P : integer;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    {Traitement}
    P := 1;
    For  i:=0 to N  do
        P := P * i;

    {Sorties}
    Writeln('La Produit P (factoriel) = ', P);
End.
```

Le programme Pascal ci-dessus est disponible sur le lien : https://onlinegdb.com/qhIgQmcv5

3- Calculer la somme $S = x+x^2+x^3+ \ldots + x^n$

Analyse du problème

Pour calculer la valeur de S, nous devons connaître la valeur de n et celle de x. Donc, S dépend de *n* et x.

La forme abrégée de la somme S est :

$$S = x + x^2 + x^3 + \ldots. x^n = \sum_{i=1}^{N} x^i$$

La formule $S = \sum_{i=1}^{N} x^i$ devient en algorithmique comme suit :

```
S ← 0;
Pour i ← 1 à N Faire
   S ← S + Xi;
Fin-Pour;
```

Pour la puissance x^i, on utilise une variable $P=x^i$ (Changement de variable).

Tel-que : pour i=1, p=x, pour i=2, $p=x^2$, … pour i=n, $p=x^n$. Et on utilise la récurrence de la puissance : $x^i = x^{i-1} * x$ pour i>=1.

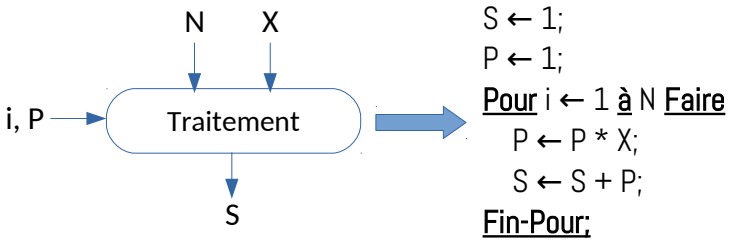
Donc, l'algorithme précédent devient :

```
S ← 0;
P ← 1;
Pour i ← 1 à N Faire
   P ← P * X;
   S ← S + P;
Fin-Pour;
```

ou bien :

```
S ← 0;
P ← X;
Pour i ← 1 à N Faire
   S ← S + P;
   P ← P * X;
Fin-Pour;
```

Et le schéma Entrée / Traitement / Sortie sera comme illustré dans la figure ci-dessous :



En exploitant le schéma d'entrée/Traitement/Sortie, on aura l'algorithme et le programme pascal suivant :

Algorithme

```
Algorithme TP5_Exo2_Q03;
  Variables
      N, i : entier;
      X, P, S : réel;
Début
    {Entrées}
    Lire(N, X) ;

    {Traitement}
    S ← 0;
    P ← 1;
    Pour i ← 0 à N faire
        P ← P * X;
        S ← S + P;
    Fin-Pour;

    {Sorties}
    Écrire(S);
Fin.
```

Programme PASCAL

```
Program TP5_Exo2_Q03;
  Var
      N, i : integer;
      X, P, S : real;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    Write('Donner la valeur de X  :') ;
    Read(X) ;

    {Traitement}
    S := 0;
    P := 1;
    For  i:=0 to N  do
    begin
        P := P * X;
        S := S + P;
    end;

    {Sorties}
    Writeln('La somme S  = ', S:0:3);
End.
```

Le programme Pascal ci-dessus est disponible sur le lien : https://onlinegdb.com/TYvn17JFt1

4- Calculer la somme S = x + $x^3/2 + x^5/4! + x^7/6! + ... + (N^{ième}$ Terme)

Analyse du problème

Pour calculer la valeur de S, nous devons connaître la valeur de n et celle de x. Donc, S dépend de *n* et *x*.

La forme abrégée de la somme S est :

$$S = x + x^3/2 + x^5/4! + x^7/6! + ... (N^{ième} \text{ Terme})$$

$$S = x^1/0! + x^3/2! + x^5/4! + x^7/6! + ... (N^{ième} \text{ Terme}) = \sum_{i=0}^{N-1} x^{2*i+1}/(2*i)!$$

i=0 ⇒ le premier terme, i=1 ⇒ le deuxième terme, ..., i=(N-1) ⇒ le $N^{ième}$ terme.

La formule $S = \sum_{i=0}^{N-1} x^{2*i+1}/(2*i)!$ devient en algorithmique comme suit :

```
S ← 0;
Pour i ← 0 à (N-1) Faire
   S ← S + X^(2*i+1) / (2*i)!;
Fin-Pour;
```

La même idée que la somme précédente, on procédera au changement de variable :
On met $P = X^{(2*i+1)}$ et F = (2*i)!

Pour la variable P, elle prendra les valeurs suivantes : $X^1, X^3, X^5, X^7, ..., X^{2*i+1}, ...$

P sera initialisé à 1 et pour une valeur de P à l'itération i, à l'itération (i+1) P aura la valeur $P*X^2$:

$$X, X*X^2 = X^3, X^3*X^2 = X^5, X^5*X^2 = X^7, ...$$

Pour la valeur de F, elle prendra les valeurs suivantes : 0!, 2!, 4!, 6!, ... (2*i)!, ...

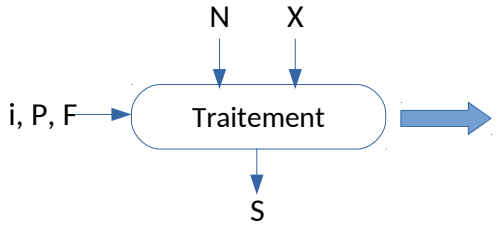
F sera initialisée à 1, et pour une valeur de F à l'itération i, à l'itération (i+1) F aura la valeur F*(2*i+1)*(2*i+2) / nous savons que **(n-1)! * n = n!** :

$$0!, 0!*1*2=2!, 2!*3*4=4!, 4!*5*6=6!, 6!*7*8=8!, ...$$

Donc, la partie traitement précédente devient comme suit :

```
S ← 0;
P ← X;
F ← 1;
Pour i ← 0 à (N-1) Faire
   S ← S + P/ F;
   P ← P*X*X;
   F ← F*(2*i+1)*(2*i+2);
Fin-Pour;
```

Et le schéma Entrée / Traitement / Sortie sera comme illustré dans la figure ci-dessous :




```
S ← 0;
P ← X;
F ← 1;
Pour i ← 0 à (N-1) Faire
   S ← S + P/ F;
   P ← P*X*X;
   F ← F*(2*i+1)*(2*i+2);
Fin-Pour;
```

En exploitant le schéma d'entrée/Traitement/Sortie, on aura l'algorithme et le programme pascal suivant :

Algorithme

```
Algorithme TP5_Exo2_Q04;
  Variables
      N,i, F : entier;
      X, P, S : réel;
Début
    {Entrées}
    Lire(N, X) ;

    {Traitement}
    S ← 0;
    P ← X;
    F ← 1;
    Pour i ← 0 à N faire
        S ← S + P/F;
        P ← P * X*X;
        F ← F * (2*+1) * (2*i+2);
    Fin-Pour;

    {Sorties}
    Écrire(S);
Fin.
```

Programme PASCAL

```
Program TP5_Exo2_Q04;
  Var
      N, i, F : integer;
      X, P, S : real;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    Write('Donner la valeur de X  :') ;
    Read(X) ;

    {Traitement}
    S := 0;
    P := 1;
    F := 1;
    For  i:=0 to N  do
    begin
        S := S + P;
        P := P * sqr(X);
        F := F * (2*i+1) * (2*i+2);
    end;

    {Sorties}
    Writeln('La somme S  = ', S:0:3);
End.
```

Le programme Pascal ci-dessus est disponible sur le lien : https://onlinegdb.com/z5jDvpG1t

5- Calculer la somme $S = x-x^2+x^3- \ldots \pm x^n$

Analyse du problème

La solution est la même que la question N°3, sauf que les termes ont un signe alterné (+1 / -1). On procède à la même analyse (on fait comme s'il n y a pas de signe).

La forme abrégée de la somme S est :

$$S = x + x^2 + x^3 + \ldots. x^n = \sum_{i=1}^{N} x^i$$

La formule $S = \sum_{i=1}^{N} x^i$ devient en algorithmique comme suit :

```
S ← 0;
Pour i ← 1 à N Faire
   S ← S + Xi;
Fin-Pour;
```

Pour la puissance x^i, on utilise une variable $P=x^i$ (Changement de variable).

Tel-que : pour i=1, p=x, pour i=2, $p=x^2$, … pour i=n, $p=x^n$. Et on utilise la récurrence de la puissance : $x^i = x^{i-1} * x$ pour i>=1. Donc on aura : P ← P*X ;

Pour le signe, il suffit d'ajouter l'opération mois unaire (-) à cette affectation, comme suit :

P ← - P * X;

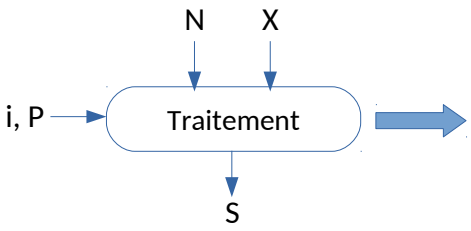
Donc, l'algorithme précédent devient :

```
S ← 0;
P ← -1;
Pour i ← 1 à N Faire
   P ← -P * X;
   S ← S + P;
Fin-Pour;
```

ou bien :

```
S ← 0;
P ← X;
Pour i ← 1 à N Faire
   S ← S + P;
   P ← -P * X;
Fin-Pour;
```

Et le schéma Entrée / Traitement / Sortie sera comme illustré dans la figure ci-dessous :




```
S ← 1;
P ← -1;
Pour i ← 1 à N Faire
   P ← - P * X;
   S ← S + P;
Fin-Pour;
```

En exploitant le schéma d'entrée/Traitement/Sortie, on aura l'algorithme et le programme pascal suivant :

Algorithme

```
Algorithme TP5_Exo2_Q05;
  Variables
      N, i : entier;
      X, P, S : réel;
Début
    {Entrées}
    Lire(N, X) ;

    {Traitement}
    S ← 0;
    P ← -1;
    Pour i ← 0 à N faire
        P ← - P * X;
        S ← S + P;
    Fin-Pour;

    {Sorties}
    Écrire(S);
Fin.
```

Programme PASCAL

```
Program TP5_Exo2_Q05;
  Var
      N, i : integer;
      X, P, S : real;
Begin
    {Entrées}
    Write('Donner la valeur de N  :') ;
    Read(N) ;

    Write('Donner la valeur de X  :') ;
    Read(X) ;

    {Traitement}
    S := 0;
    P := -1;
    For  i:=0 to N  do
    begin
        P := - P * X;
        S := S + P;
    end;

    {Sorties}
    Writeln('La somme S  = ', S:0:3);
End.
```

Le programme Pascal ci-dessus est disponible sur le lien : https://onlinegdb.com/MYN_K3ktwv

6- Afficher la table de multiplication de N (entre 1 et 10).

Analyse du problème

La variable d'entrée est N, doit vérifier une condition : N entre 1 et 10. Autrement dit, N doit vérifier la condition :

(N>=1) ET (N<=10)

Pour réaliser ça en algorithmique, nous utilisons cette technique :

```
Repeat
    Lire(N);
Jusqu'à (N>=1) ET (N<=10);
```

Pour l'affichage, nous allons afficher la valeur i*N, tel-que i varie de 0 jusqu'à 10.

Exemple :

Donner la valeur de N : 4

La table de multiplication de N est :

0 x 4 = 0

1 x 4 = 4

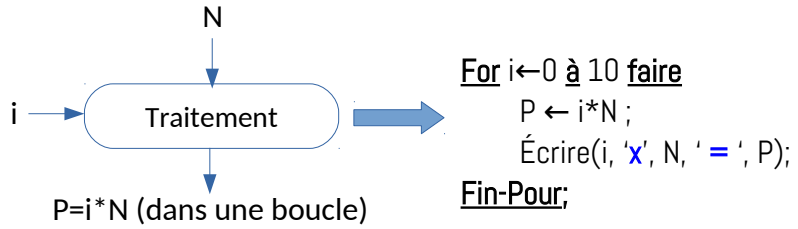
2 x 4 = 8

…

10 x 4 = 40

La partie traitement de l'algorithme est :

```
For i←0 à 10 faire
    P ← i*N ;
    Écrire(i, 'x', N, ' = ', P);
Fin-Pour;
```



La valeur de P=i*N sera affichée dans une boucle Pour qui permet de varier la valeur de i de 0 jusqu'à N.

L'algorithme et le programme PASCAL seront comme suit :

Algorithme

```
Algorithme TP5_Exo2_Q06;
  Variables
       N, i : entier;
       P : entier;
Début
    {Entrées}
    Répéter
        Lire(N) ;
    Jusqu'à (N>=1) ET (N<=10)

    {Traitement & Sortie};
    Pour i ← 0 à 10 faire
        P ← i * N;
        Écrire(i, ' x ', N, ' = ', P);
    Fin-Pour;
Fin.
```

Programme PASCAL

```
Program TP5_Exo2_Q06;
  Var
      N, i, P : integer;
Begin
    {Entrées}
    Repeat
       Write('Donner la valeur de N  :') ;
       Read(N) ;
    Until (N>=1) AND (N<=10);

    {Traitement & Sortie}
    Writeln('LA TABLE DE MULTIPLICATION DE ', N,' EST : ');
    For  i:=0 to 10  do
    begin
        P := i * N;
        Writeln('   ', i:2, ' x ', N:2, ' = ', p:3);
    end;
End.
```

Le programme Pascal ci-dessus est disponible sur le lien : https://onlinegdb.com/ZCcEMW61W

7- Somme de valeurs pairs et le produit des valeurs impaires entre A et B.

Analyse du problème

Nous avons A et B deux valeurs entières, tel-que A<B.

Nous devons introduire N valeurs entières, qui sont entre A et B.

La variables d'entrée :

A, B

N

v1, v2, ..., vn

Nous ne pouvons déclarer les valeurs V1, V2, ..., Vn de cette façons. (Nous devons utiliser un tableau à 1 dimension qui sera l'objet de chapitre 1 de semestre 2). L'idée est de déclarer une seule variable Vi et de la lire plusieurs fois dans une boucle.

Pour les lectures, ça sera comme suit :

```
Repeat
      Lire(A, B);
Jusqu'à (A < B);
Lire(N);

Pour i← 1 à N faire
    Repeter
        Lire(Vi) ;
    Jusqu'à (Vi>=A) ET (Vi<=B) ;
    {Traitement}
Fin-Pour ;
```

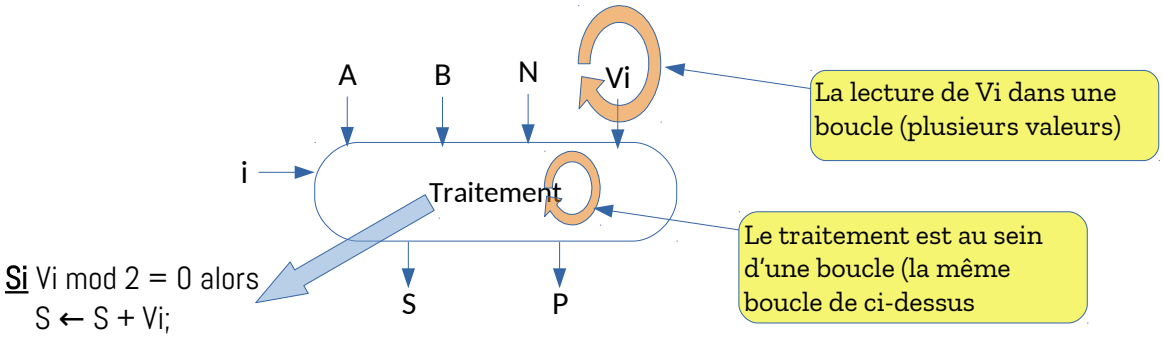
Boucle **Pour** afin de lire N valeurs de Vi

Boucle **Répéter** afin de s'assurer que la valeur de Vi est correcte (entre A et B).

Une fois la valeur de Vi est valide, nous entamons le calcule de la somme S des valeurs pairs et le produit P des valeurs impairs, comme suit :

```
Si Vi mod 2 = 0 alors
   S ← S + Vi;
Sinon
   P ← P * Vi;
Fin-Si;
```

Bien évidemment, il ne faut pas oublier d'initialiser S à 0 et P à 1.




```
Si Vi mod 2 = 0 alors
   S ← S + Vi;
Sinon
   P ← P * Vi;
Fin-Si;
```

Algorithme

```
Algorithme TP5_Exo2_Q07;
  Variables
      A, B, N, i, Vi, P, S : entier;
Début
    {Entrées}
    Répéter
       Lire(A, B) ;
    Jusqu'à (A>B);
    Lire(N) ;

    {Traitement};
    Pour i ← 1 à N faire
        Répéter
            Lire(Vi)
        Jusqu'à (Vi>=A) ET (Vi<=B);
        Si Vi mod 2 = 0 Alors
            S ← S + Vi;
        Sinon
            P ← P * Vi;
        Fin-Si;
    Fin-Pour;
    écrire(S, P)
Fin.
```

Programme PASCAL

```
Program TP5_Exo2_Q06;
  Var
      A, B, N, i, Vi, P, S : integer;
Begin
    {Entrées}
    Repeat
       Write('Donner la valeur de A et B  :') ;
       Read(A, B) ;
    Until (A<B);
    Write('Donenr la valeur de N : ') ;
    Read(N);

    {Traitement}
    For  i:=1 to N  do
    begin
        Repeat   Read(Vi) ; Until (Vi>=A) AND (Vi<=B);
        if (Vi mod 2 = 0) then
            S:= S + Vi
        else
            P := P * Vi;
    end;
    {Sortie}
    Write(S, P) ;
End.
```

Le programme Pascal ci-dessus est disponible sur le lien : https://onlinegdb.com/6J7ndg1qc

Bon Courage

&

Travaillez bien.

Cours Elearning :

https://elearning.univ-bejaia.dz/course/view.php?id=7944

Page facebook :

https://www.facebook.com/InitiationAlgoProgrammation/

La chaîne Youtube :

https://www.youtube.com/c/AlgoProgrammation1èreAnnéeTechnologie

---

Adapté par: Redouane OUZEGGANE

rouzeggane@gmail.com - redouane.ouzeggane@univ-bejaia.dz