Université : A.Mira – Bejaia
Faculté : Sciences Exactes
Département : Informatique

Année Universitaire : 2022-2023

Niveau : Première Année Ingénieur

ALSD-1 - Série de TD N°03

Sommaire

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

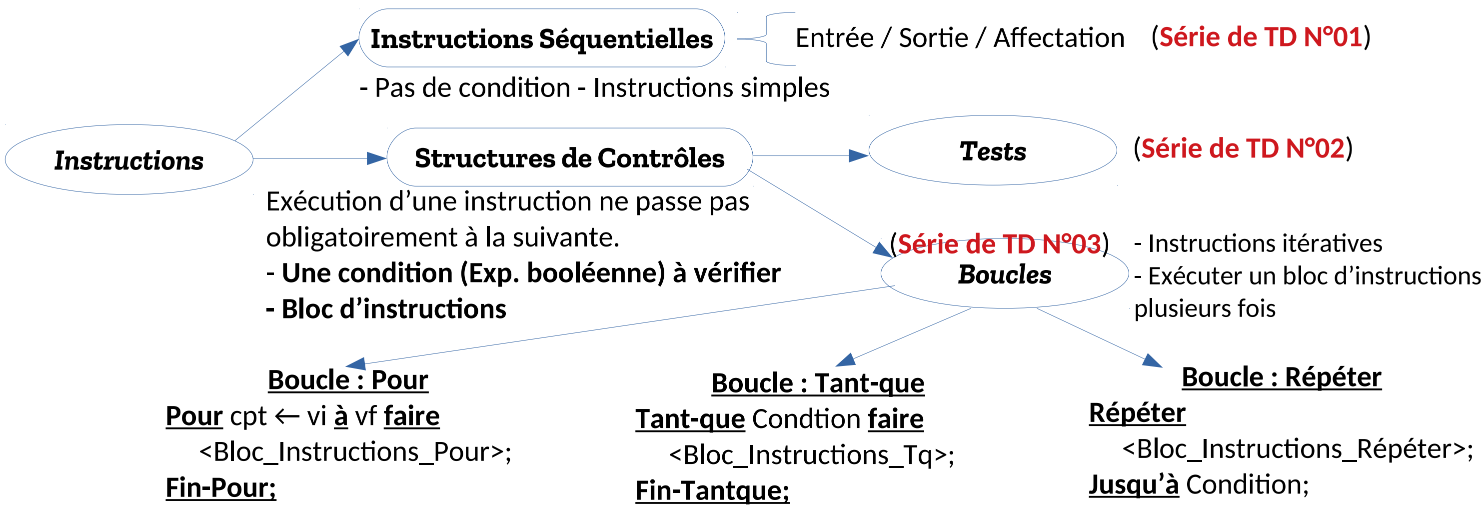
Université : A.Mira – Bejaia | Année Universitaire : 2022-2023

Faculté : Sciences Exactes

Département : Informatique | Niveau : Première Année Ingénieur

ALSD-1

SÉRIE DE TD N°03 (BOUCLES: POUR, TANT-QUE ET RÉPÉTER)



TD ALSD-1

SÉRIE DE TD N°03 (INSTRUCTIONS ITÉRATIVES : POUR, TANT-QUE & RÉPÉTER)

Exercice 01

Écrire un algorithme pour chaque question suivante :

1. Calculer et afficher la somme : 1 + 2 + 3 + 4 + … + n
2. Calculer et afficher le x^n via la formule suivante : $x^n = x*x*x* \ldots$ *(n fois)*
3. Calculer et afficher le factoriel de n (n ≥ 3)
4. Calculer et Afficher le produit A x B sans utiliser l'opération de multiplication (A et B deux entier positifs)

Exercice N°02

Soit N une valeur entière supérieure ou égale à 10. Écrire un algorithme qui permet d'afficher toutes les valeurs positives (>=0) paires qui sont inférieures ou égale à N :

- Dans l'ordre croissant
- Dans l'ordre décroissant

Exercice N°03

- Écrire une algorithme qui permet de calculer la valeur approximative de π tel-que $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots\pm\frac{1}{2\times n+1}$?
- Modifier l'algorithme en s'arrêtant lorsque le terme $\frac{1}{i}$ soit inférieur à ε (epsilon).

Exercice N°04

1– Écrire un algorithme qui indique si un nombre entier positif est premier ou non (un nombre premier est un nombre possédant exactement deux diviseurs) ?

2– Écrire un algorithme qui affiche toutes les nombres premiers entre 1 et N (N ≥ 100) ?

Exercice N°05

- Écrire un algorithme qui permet d'afficher tous les nombres parfaits qui sont entre 1 et N ? tel-que N est nombre entier supérieur ou égale à 10.
- Modifier l'algorithme pour qu'il affiche les nombres parfait en commençant la recherche à partir de N jusqu'à 1 ?

Exercices supplémentaires

1) Écrire un algorithme (avec l'organigramme) qui permet de trouver le PGCD (Plus Grand Commun Diviseur) de deux nombres entiers strictement positifs A et B, en utilisant cette méthode :

i. soit X et Y deux nombres tel-que : X = A et Y = B ;
ii. Si X = Y alors aller à (v);
iii. Si X > Y alors X devient X-Y ; et Aller à (ii) ;
iv. Si X < Y alors Y devient Y – X et Aller à (ii);
v. PGCD de A et B = X (Fin)

Compléter l'algorithme pour calculer aussi PPCM (Plus Petit Commun Multiplicateur) de A et B ?

2) Écrire un algorithme avec le Programme C ainsi que son organigramme (Représentation graphique de l'algorithme) qui permet de rechercher tous les nombre cubiques entre 100 et 999. Un nombre cubique est un nombre est égale à la somme des ces chiffres élevé à la puissance 3. Par exemple : $153 = 1^3 + 5^3 + 3^3$

3) Soit M, valeur entière supérieure ou égale à 25, qui représente une somme d'argent (Montant). Soit a , b (aussi deux valeurs entières strictement positives) qui représentent les valeurs de deux pièces de monnaie. Écrire un algorithme qui permet de chercher et d'afficher Toutes les possibilités pour rendre la monnaie de la somme M en utilisant uniquement les deux pièces a et b (Afficher aussi, à la fin, le nombre de possibilités trouvés).

4) Écrire l'algorithme d'une fiche de paie journalière de N ouvriers (N ≥ 50), d'une entreprise, rémunérés à la tâche. Pour cela, on donne :

- La valeur de cette rémunération par pièces réalisées VP,
- Le salaire brut (SB) est calculé selon le nombre de pièces correctes réalisées pendant la journée (NPC) comme suit :

  Si NPC 100, l'ouvrier touche NPC*VP
  Si NPC > 100, l'ouvrier touche 150*VP
- On enlève à la fin 10% du salaire pour les charges sociales (CS).

Calculer et afficher le salaire journalier brut (SB), les charges sociales (CS) et salaire journalier net (SN).

NB : Salaire brut=salaire totale ; Salaire net = salaire sans les charges sociales.

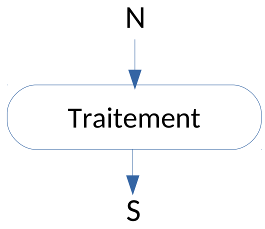
Solution - TD N° 03 ALSD-1

Exercice N°01 : Sommes & Produits

1- Somme S = 1+2+3+....+N

<u>Analyse du problème</u>

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 :

$$S = 1 + 2 + 3 + 4 + \ldots + N$$

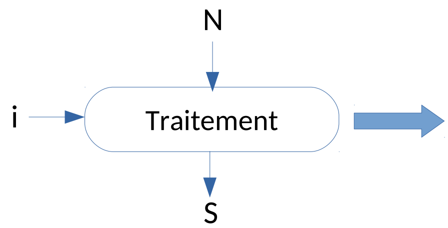
La forme abrégée de cette somme est :

$$S = \sum_{i=1}^{N} i$$

L'égalité : $S = \sum_{i=1}^{N} i$ devient en algorithmique comme suit :

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

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




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

<u>Algorithme & Programme C</u>

Lorsque on regroupe toute l'analyse précédente du problème, nous aurons l'algorithme dans la page suivante :

Algorithme

```
Algorithme TD3_Exo1_Q01;
  Variables
      N, i, S : entier;
Début
     //Entrées
     Lire(N) ;

     //Traitement
     S ← 0;
     Pour i ← 1 à N faire
          S ← S + i;
     Fin-Pour;

     //Sorties
     Écrire(S);
Fin.
```

Programme C

```c
#include <stdio.h>

int main() {
     int N, i, S;

     //Entrées
     printf("Donner la valeur de N  :") ;
     scanf("%d", &N) ;
     //Traitement
     S = 0;
     for  (i=1 ; i<=N ; i++) {
          S = S + i;
     }
     //Sorties
     printf("La Somme S = %d", S);
}
```

Le programme C correspondant est disponible sur le lien : https://onlinegdb.com/SMkxzYqvpi

Vous pouvez modifier le programme pour qu'il affiche les sommes intermédiaires pour chaque itération de la boucle **pour** (***Voir le lien : https://onlinegdb.com/pcvRhXOJs***):

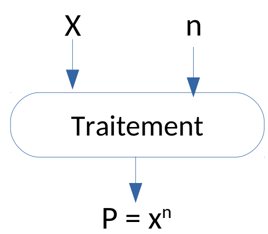
Programme C

```c
#include <stdio.h>

int main() {
     int N, i, S;

     //Entrées
     printf("Donner la valeur de N  :") ;
     scanf("%d", &N) ;
     //Traitement
     S = 0;
     for  (i=1 ; i<=N ; i++) {
          S = S + i;
          printf("Pour i=%d , S = %d\n", i, S);
     }
     //Sorties
     printf("La Somme S = %d", S);
}
```

2- $x^n = x*x*x*x*....*x$

Analyse du problème

La figure ci-contre, illustre les variables d'entrée et de sortie pour calculer x^n (On met $P = x^n$):

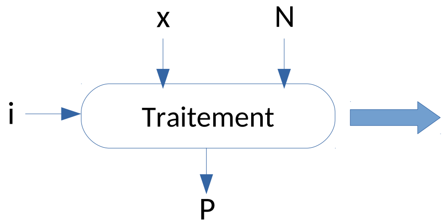


Tel-que : P = x * x * x ..... (n fois)

On peut écrire : P = $\prod_{i=1}^{N} x$. Cette égalité devient en algorithmique :

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

Le schéma Entrée / Traitement / Sortie devient :




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

En ce qui concerne les types de données :

x : réel, N : entier , i : entier et P est de type réel.

Algorithme & Programme C

En regroupant tous les éléments de l'analyse ci-dessus, nous aurons l'algorithme et le programme ci-dessous :

Algorithme	Programme C

<pre>Algorithme TD3_Exo1_Q02;
  Variables
     N, i : entier;
     x, P : réel ;
Début
    //Entrées
    Lire(x, N) ;
 	
    //Traitement
    P ← 1;
    Pour i ← 1 à N faire
        P ← P * x;
    Fin-Pour;
 	
    //Sorties
    Écrire(P);
Fin.</pre>
	<pre>#include <stdio.h>
	
	int main() {
	     int N, i;
	     float x, P;
	
	    //Entrées
	    printf("Donner les valeurs de x et N :") ;
	    scanf("%f %d", &x, &N) ;
	    //Traitement
	    P = 1;
	    for (i=1 ; i<=N ; i++) {
	        P = P * x;
	    }
	    //Sorties
	    printf("X puissance N = %.3f", P);
	}</pre>

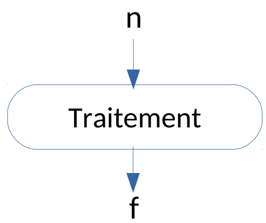
Le programme C correspondant est disponible sur le lien : https://onlinegdb.com/3qAT4P_wJ2

3- Factoriel de N

Analyse du problème

Le factoriel de n est égale à $n!=1\times2\times3\times....\times n$

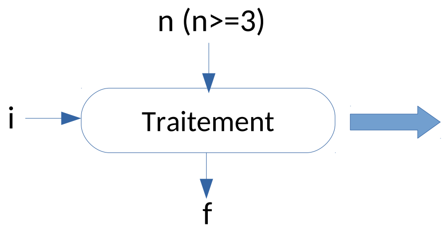
Si on met *f=n!* , on aura le schéma d'entrée et de sortie ci-contre :



On peut écrire : $f=\prod_{i=1}^{N} i$. Cette égalité devient en algorithmique :

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

Le schéma Entrée / Traitement / Sortie devient :




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

Bien évidemment, les variables i, n, et f sont tous des entiers.

Algorithme (f=n!)

Algorithme

Algorithme TD3_Exo1_Q03;
  Variables
      n, i, f : entier;
Début
    //Entrées
    Répéter
        Lire(n) ;
    Jusqu'à (n>=3); //Contrôler la valeur de n
 	
    //Traitement
    f ← 1;
    Pour i ← 1 à N faire
        f ← f * i;
    Fin-Pour;
 	
    //Sorties
    Écrire("Le factoriel f = ", f);
Fin.

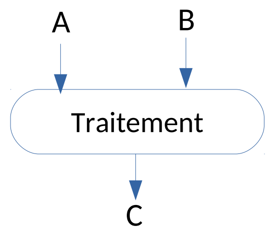
4- Calculer A*B

Analyse du problème

Puisque A et B sont des entiers positifs, on peut écrire :

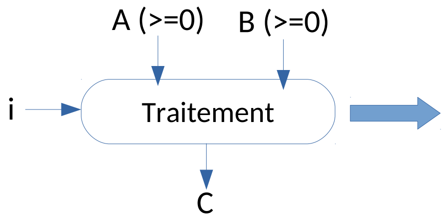
$$A \times B = \sum_{i=1}^{B} A = \sum_{i=1}^{A} B$$

A*B est égale à la somme des termes A, B fois (ou la somme des termes B, A fois). Si on met C=A*B , on aura le schéma d'entrée et de sortie ci-contre :




```
C ← 0;
Pour i ← 1 à B Faire
   C ← C + A;
Fin-Pour;
```

Le schéma Entrée / Traitement / Sortie devient :




```
C ← 0;
Pour i ← 1 à B Faire
   C ← C + A;
Fin-Pour;
```

Bien évidemment, les variables i, A, B, et C sont tous des entiers.

Algorithme (C=A*B)

Algorithme

```
Algorithme TD3_Exo1_Q04;
  Variables
      A, B, C, i : entier;
Début
    //Entrées
    Répéter
        Lire(A) ;
    Jusqu'à (A>=0); //Contrôler la valeur de A

    Répéter
        Lire(B) ;
    Jusqu'à (B>=0); //Contrôler la valeur de B

    //Traitement
    C ← 0;
    Pour i ← 1 à A faire
        C ← C + B;
    Fin-Pour;

    //Sorties
    Écrire("Le produit AxB = ", C);
Fin.
```

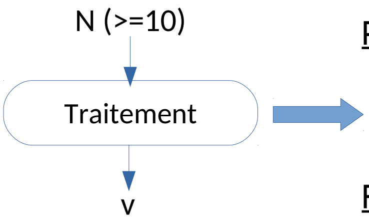
Exercice N°02 : Afficher les valeurs positives paires

1- Dans l'ordre croissant

Analyse du problème

On veut afficher les valeurs positives (>=0) parie et qui sont inférieures à N, tel-que N est un entier ≥ 10. On parcours toutes les valeurs de 0 à N, et on vérifier chacune d'elle si elle est paire ou non. Un valeur paire est une valeur divisible par 2 (reste de division égale à 0) :

Le schéma suivant récapitule ce qui a été ci-dessus expliqué :




```
Pour v ← 0 à N Faire
   si (v mod 2 = 0) alors
      écrire (v) ;
   Fin-Si;
Fin-Pour;
```

Algorithme

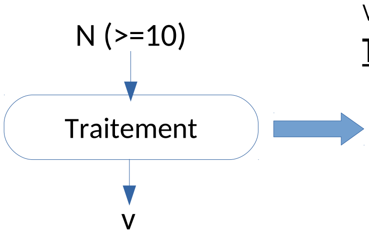
Algorithme

```
Algorithme TD3_Exo2_Q01;
  Variables
      N, v : entier;
Début
    //Entrées
    Répéter
        Lire(N) ;
    Jusqu'à (N>=10); //Contrôler la valeur de N

    //Traitement & Sortie
    Pour V ← 0 à N faire
       Si (V mod 2 = 0) alors
           Écrire (V);
       Fin-Si;
    Fin-Pour;
Fin.
```

2- Dans l'ordre décroissant

Analyse du problème

C'est la même solution que la précédente, sauf que la valeur de v varie de N à 0 avec un pas = -1. Dans l'algorithmique, on utilisera la boucle **Tant-que** avec la décrémentation de la variable v : V← V-1 ; comme indiqué dans le schéma c-dessous :




```
v ← N;
Tant-que v >=0 Faire
   si (v mod 2 = 0) alors
      écrire (v) ;
   Fin-Si;
   v ← v-1;
Fin-Tantque;
```

Algorithme

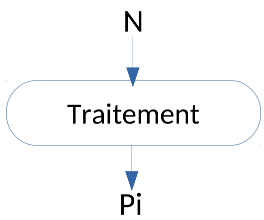
```
Algorithme
Algorithme TD3_Exo2_Q02;
  Variables
     N, v : entier;
Début
    //Entrées
    Répéter
       Lire(N) ;
    Jusqu'à (N>=10); //Contrôler la valeur de N

    //Traitement & Sortie
    v ← N;
    Tant-que v >=0 Faire
      Si (v mod 2 = 0) alors
         écrire (v) ;
      Fin-Si;
      v ← v-1;
    Fin-Tant-que;
Fin.
```

Exercice N°03 : Valeur approximative de π

1- Calcul jusqu'à (2n+1)

Analyse du problème

Le schéma d'E/S de l'algorithme est :



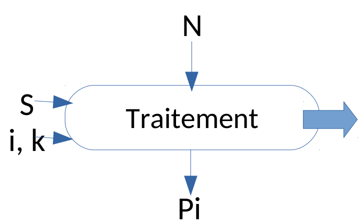
Dans l'exercice, nous la formule suivante : $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots\pm\frac{1}{2\times n+1}$.

On met $S=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots\pm\frac{1}{2\times n+1}=\sum_{0}^{n}\frac{k\times 1}{2\times i+1}$ / k prends les valeurs 1, -1 alternativement. Ansi, pour calculer S, on écrit :

```
S ← 0;
K ← 1 ;
for i ←0 à n Faire
   S ← S + k* 1 / (2*i+1);
   k ← -k;
Fin-Pour;
```

Pour calculer la valeur de Pi, on écrit : Pi ← 4*S;

En regroupant tous les éléments ci-dessus, nous obtiendrons l'algorithme:




```
S ← 0;
K ← 1 ;
for i ←0 à n Faire
   S ← S + k* 1 / (2*i+1);
   k ← -k;
Fin-Pour;
Pi ← 4*S ;
```

Algorithme

Algorithme

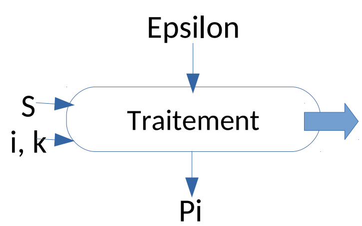
```
Algorithme TD3_Exo2_Q01;
  Variables
      N, k, i : entier;
      S, Pi : réel ;
Début
    //Entrées
    Lire(N) ;

    //Traitement
    S ← 0 ;
    K ← 1 ;
    Pour i ← 0 à N faire
       S ← S + k / (2*i+1);
       k ← - k;
    Fin-Pour;
    Pi ← 4*S ;

    //Sortie
    écrire(Pi) ;
Fin.
```

Le programme C correspondant à l'algorithme ci-dessus : https://onlinegdb.com/M7wcHd72C

2- Calcul jusqu'à un terme inférieure à epsilon

Analyse du problème

La différence par rapport à la solution précédente est qu'on ne connaît pas le nombre d'itérations, on doit calculer jusqu'à trouver un terme inférieure à epsilon (petite valeur). Dans ce cas nous utilisons soit la boucle Tant-que ou la boucle Répéter, comme suit :




```
S ← 0;
K ← 1 ;
i ← -1:
Répéter
   i ← i+1;
   S ← S + k* 1 / (2*i+1);
   k ← -k; i ← i+1;
Jusqu'à (1/(2i+1) < epsilon);
Pi ← 4*S ;
```

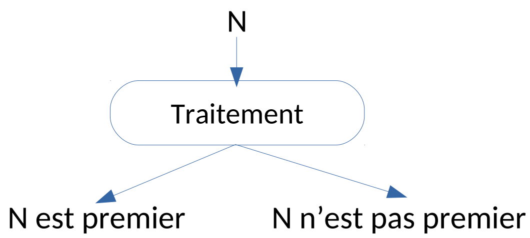
Lien du programme C : https://onlinegdb.com/gphHoFPOM

Exercice N°04 : Nombres premiers

1- Indiquer si un nombre est premier ou non

Analyse du problème

L'algorithme, pour entier positif N, indique si N est premier ou non. Donc, le schéma d'entrée / sortie est comme suit :



L'algorithme affiche soit N est premier ou bien N n'est pas premier.

-Méthode 01 : compter le nombre de diviseurs

Un nombre premier est un nombre qui possède exactement deux diviseurs. Donc, une première solution consiste à compter le nombre de diviseurs d'un nombre entier N.

Soit Nbd le nombre de diviseurs de N, il est calculé par le traitement ci-dessous (***Traitement 01***) :

```
// Traitement 01
Nbd ← 0;
Pour i←1 à N faire
  Si (N mod i = 0) Alors
     Nbd ← Nbd + 1 ;
  Fin-Si;
Fin-Pour;
```

Par la suite, on vérifie le nombre de diviseurs s'il est égale à deux ou non, et on décide à base de ça si N est premier ou non. Voir le traitment 02 ci-dessous :

```
// Traitement 02;
Si (Nbd = 2) Alors
     Écrire("N est premier") ;
Sinon
     Écrire("N n'est pas premier") ;
Fin-Si;
```

En regroupant ce qui a été ci-dessus indiqué, on aura l'algorithme suivant.

```
Algorithme TD3_Exo2_Q01;
Variables
    N, Nbd, i : entier;
Début
  //Entrées
  Lire(N) ;
  //Traitement
  Nbd ← 0 ;
  Pour i ← 1 à N faire
    Si (N mod i = 0) alors
        Nbd ← Nbd + 1;
    Fin-Si;
  Fin-Pour;

  //Sortie
  Si (Nbd = 2) alors
    écrire("N est premier");
  Sinon
    écrire( "N n'est pas premier");
  Fin-Si;
Fin.
```

Le programme C correspondant à l'algorithme ci-dessus est sur le lien : https://onlinegdb.com/oUUUOmNEU

Une amélioration de l'algorithme précédent, est de compter le nombre de diviseurs entre 2 et N/2, et si ce nombre est égale à 0, donc, N est premier, sinon, il n'est pas premier. (Juste il faut prendre en considération le cas particulier des valeurs 0, 1) :

```
Algorithme TD3_Exo2_Q01;
Variables
    N, Nbd, i : entier;
Début
  //Entrées
  Lire(N) ;
  //Traitement
  Nbd ← 0 ;
  Pour i ← 2 à (N div 2) faire
    Si (N mod i = 0) alors
        Nbd ← Nbd + 1;
    Fin-Si;
  Fin-Pour;

  //Sortie
  Si (Nbd = 0 ET N>=2 ) alors
     écrire("N est premier");
  Sinon
     écrire( "N n'est pas premier");
  Fin-Si;
Fin.
```

Le lien suivant : https://onlinegdb.com/GfSI6iJw1 contient l'implémentation de l'algorithme en langage C.

-Méthode 02 : Chercher le premier diviseur entre 2 et N/2

Une autre méthode consiste à chercher s'il y a un diviseur entre 2 et N/2, dès qu'on trouve le premier diviseur on arrête la recherche et on affiche que N n'est pas premier. Sinon (il n y a pas de diviseurs entre 2 et N/2) N est premier. Le fragment algorithme suivant illustre la solution :

```
R ← (N-1) * N; //Si N = 0 ou N = 1 on aura R = 0, donc  0 et 1 ne sont pas premier ...
i←2
Tant-que ( (i<= N div 2) ET (R <> 0)) Faire
    R ← N mod i;
    i ← i+1 ;
Fin-Tant-que;
Si (R=0) alors
    écrire ("N n'est premier") ;
Sinon
    écrire("N est premier") ;
Fin-Si;
```

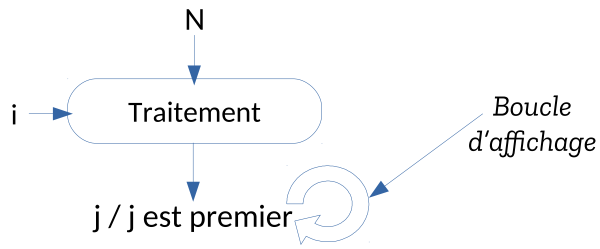
Le code C de la partie traitement ci-dessus est sur le lien : https://onlinegdb.com/Dx9757nhI

2- Afficher tous les nombres premier entre 1 et N

Analyse du problème

En se basant sur la solution de la première question, nous allons élaborer un algorithme qui permet d'afficher toutes les valeurs premiers entre 1 et N / N>=10.

L'algorithme accepte N comme entrée, et affiche toutes les valeurs premiers entre 1 et N, comme élaborer sur le schéma suivant :



La partie Traitement sera comme suit :

```
Pour j←1 à N faire
    R ← (j-1) * j;
    i←2
    Tant-que ( (i<= j div 2) ET (R <> 0)) Faire
        R ← j mod i;
        i ← i+1 ;
    Fin-Tant-que;
    Si (R=0) alors // donc i est premier => afficher i
        écrire (i) ;
    Fin-Si;
Fin-Pour ;
```

L'algorithme complet sera comme suit (voir le lien : https://onlinegdb.com/gxb1nrkvv) :

Algorithme

```
Algorithme TD3_Exo4_Q04;
  Variables
      N, j, i, R : entier;
Début
    //Entrées
    Répéter
        Lire(N) ;
    Jusqu'à (N>=100);

    //Traitement & Sortie
    Pour j←1 à N faire
       R ← (j-1) * j;
       i←2
       Tant-que ( (i<= j div 2) ET (R <> 0)) Faire
           R ← j mod i;
           i ← i+1 ;
       Fin-Tant-que;
       Si (R <> 0) alors // donc j est premier => afficher j
           écrire (j) ;
       Fin-Si;
    Fin-Pour ;
Fin.
```

Exercice N°05 : Nombres parfaits

1- Afficher les nombres parfaits entre 1 et N

Analyse du problème

Une nombre parfait est un nombre qui égale à la somme des diviseurs propres d'un nombre N. Les diviseurs propres d'un entier N est les diviseurs de N >=1 et < N (c'est à dire entre 1 et N / 2).

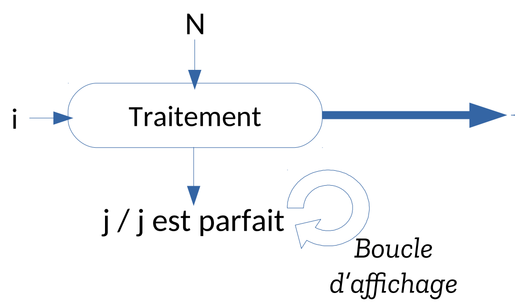
Donc, pour savoir si N est parfait, on réalise la somme des diviseurs de N qui sont entre 1 et N/ 2, comme suit :

```
S ← 0;
Pour i←1 à (N div 2) faire
   Si (N mod  i = 0) alors //i  est un diviseur de N
       S ← S + i;
   Fin-Si;
Fin-Pour ;
```

Par la suite, on compare cette somme à N :

```
S ← 0;
Si (S = N) alors
     écrire ("N est un nombre parfait") ;
Sinon
     écrire ("N n'est pas un nombre parfait") ;
Fin-Si;
```

Ce qui est demandé dans cette question est d'afficher les valeurs parfaits entre 1 et N, et le schéma d'E/S et la partie traitement de ce problème sera comme suit :




```
Pour j←1 à N faire
   S ← 0;
   Pour i←1 à j / 2 faire
      S ← S + i;
   Fin-Pour;

   Si (s = j) alors
      Écrire (j);
   Fin-Si;
Fin-Pour;
```

Voir le lien suivant pour le programme C correspondant : https://onlinegdb.com/_ZH7r4gDv
Et aussi le lien : https://onlinegdb.com/OFaH9b3b6

Remarque :

Les exercices supplémentaires seront traités par les étudiants. Si un étudiant résout un exercice, il peut voir avec l'enseignant du cours, TD ou TP pour lui vérifier sa solution.

Bon Courage

&

Travaillez bien.

Cours Elearning :
https://elearning.univ-bejaia.dz/course/view.php?id=2749

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

La chaîne Youtube :
https://www.youtube.com/c/AlgoProgrammation1èreAnnéeTechnologie

La playlist sur le langage C :
https://youtube.com/playlist?list=PLwHHAvorm5F-tL9EXDEH0miOKmAj7iUTU

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