Algorithmique :

Ce fichier est préparé par le groupe **Compil'Court** d'ENSA Agadir.
Rejoignez nous sur discord https://discord.gg/Zy2rwPHA
Let's make ENSA-A great again!

Quelques notions préliminaires :

Algorithme : En littérature, c'est la suite d'opérations élémentaires constituant un schéma de calcul ou de résolution d'un problème donné.
Et pour écrire un, il faut nécessairement passer par la démarche suivante.
Analyser le problème donné et identifier les variables, qui sert à stocker une valeur donnée, il peut être un nombre entier ou réel, une chaine de caractère ou tout simplement un caractère. Ensuite on pense à une méthode judicieuse pour résoudre le problème. Généralement un algorithme est de la forme suivante :

```
Variables : (Entiers, réels, caractères...)
Début :
        instruction 1
        instruction 2
        ....
        ....
Fin
```

Les méthodes pour résoudre un problème sont différentes, mais ils ont une relation avec la nature de ce dernier, on utilise les boucles, les tests, les tableaux... Ainsi que les opérateurs arithmétiques ou logiques, qui se déduisent à partir du problème :
Les opérateurs arithmétiques : L'addition `(+)`, la soustraction `(-)`, la multiplication `(*)`, la division `(/)`, le modulo `(%)`, la puissance `(^)`.
Les opérateurs logiques : `NON`, `ET`, `OU`, `=`, $\neq$, `<`,`>`,`<=`, `>=` et `&`.
Les fonctions : Pour afficher un message à l'utilisateur, on utilise le mot clé (fonction) : `afficher` ou bien `écrire`, par exemple la syntaxe `écrire("Hello World")` affiche dans l'écran d'utilisateur le message **Hello World**, et pour afficher un message suivi d'une valeur stockée dans un variable on écrit : `écrire("message", var1, var2)` où `var1` et `var2` sont deux variables déclarés.
Pour lire les données saisies par l'utilisateur, on utilise le mot clé : `lire` ou bien `saisir`, la syntaxe : `lire(var1)` l'utilisateur doit entrer une donnée selon la nature de `var1`.
Les instructions conditionnelles :
les instructions conditionnelles servent à n'exécuter une instruction ou plusieurs instructions que si une condition est vérifiée.

```
Si (condition) Alors
        Instructions 1
Sinon
        Instructions 2
FinSi
```

```
Si (condition) Alors
        Instructions
FinSi
```

Les boucles : (Les instructions itératives)
La boucle tant que : Elle est utilisée lorsqu'il faut répéter une certaine instruction tant qu'une condition est réalisée :

```
Tantque (condition)
        Instructions
FinTantque
```

La boucle pour : Elle est utilisée lorsqu'on répète des instructions en faisant évoluer un compteur d'une valeur initiale à une finale par un certain pas :

```
Pour (compteur) Allant De (valeur initiale) À (valeur finale) Par Pas (le pas)
        Instructions
FinPour
```

N.B : Si le pas n'est pas mentionnée dans l'algorithme, alors par défaut l'incrémentation se fait en ajoutant 1 à la valeur initiale du compteur.
La boucle jusqu'à : Elle est utilisée lorqu'il faut répéter des instructions jusqu'à qu'une condition soit réalisée.

```
Répéter
        instructions
Jusquà (condition)
```

Un cocktail des exercices :

Écrire un algorithme qui permet de permuter les valeurs de deux variables.

```
Variables : x,y,z : entier
Début :
        x ←1
        y ←2
        z ←x
        x ←y
        y ←z
Fin
```

Écrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le double de ce nombre.

```
Variables : x : entier
Début :
        écrire("Donner le nombre")
        lire(x)
        x ←2*x
        écrire("Le double du nombre est", x)
Fin
```

Écrire un algorithme qui vous demande de saisir votre nom puis votre prénom et qui affiche ensuite votre nom complet.

```
Variables : nom, prénom, nom_complet : chaine de caractères
Début :
        écrire("Saisir votre nom")
        lire(nom)
        écrire("Saisir votre prénom")
        lire(prénom)
        nom_complet ←nom&prénom
        écrire("Votre nom nom complet est :" nom_complet)
Fin
```

Écrire un algorithme qui permet d'afficher la valeur absolue d'un nombre réel.
On peut traiter ce problème en deux versions différentes.

```
Variables : x,y : réel
Début :
        écrire("Donner un nombre réel")
        lire(x)
        y←x
        Si (x<0) Alors
                y ←-x
        FinSi
        écrire("La valeur absolue de", x,"est", y)
Fin
```

```
Variables : x : réel
Début :
        écrire("Donner un nombre réel")
        lire(x)
        Si (x<0) Alors
                écrire("La valeur absolue de", x,"est", -x)
        Sinon
                écrire("La valeur absolue de", x,"est", -x)
        FinSi
Fin
```

Écrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui teste et affiche s'il est divisible par 3.

```
Variables : n : entier
Début :
        écrire("Donner un nombre entier")
        lire(n)
        Si (n%3=0) Alors
                écrire(n,"est divisible par 3")
        Sinon
                écrire(n,"n'est pas divisible par 3")
        FinSi
Fin
```

Écrire un algorithme qui permet d'identifier si un entier est nul, positif ou négatif.
Ce problème peut être traité en deux versions différentes

```
Variables : n : entier
Début :
        écrire("Donner un nombre entier")
        lire(n)
        Si (n=0) Alors
                écrire("Le nombre est nul")
        Sinon
                Si (n<0) Alors
                        écrire("Le nombre est négatif")
                Sinon
                        écrire("Le nombre est positif")
                FinSi
        FinSi
Fin
```

```
Variables : n : entier
Début :
        écrire("Donner un nombre entier")
        lire(n)
        Si (n=0) Alors
                écrire("Le nombre est nul")
        FinSi
        Si (n<0) Alors
                écrire("Le nombre est négatif")
        FinSi
        Si (n>0)
                écrire("Le nombre est positif")
        FinSi
Fin
```

Le prix de photocopies dans une reprographie varie selon le nombre demandé: $0,5$ dhs pour un nombre de copies inférieur à 10, $0,4$ dhs pour un nombre compris entre 10 et 20 et $0,3$ dhs au-delà. Écrire un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées, qui calcule et affiche le prix à payer.

```
Variables : prix, nbr : réel
Début :
        écrire("Donner le nombre de copies demandé")
        lire(nbr)
        Si (nbr<10) Alors
                prix ←nbr*0,5
        Sinon
                Si (nbr>10 ET nbr<20) Alors
                        prix ←nbr*0,4
                Sinon
                        prix ←nbr*0,3
                FinSi
        FinSi
        écrire("Le montant à payer est :", prix)
Fin
```

Écrire un algorithme qui demande à l'utilisateur d'entrer un nombre compris entre 1 et 3 jusqu'à ce que la réponse convienne.
Ce problème peut être traité en utilisant les deux boucles 'Tant que' et 'Jusqu'à'.

```
Variables : N : entier
Début :
        écrire("Donner un nombre")
        lire(N)
        Tantque (N<1 OU N>3)
                écrire("Essayer encore une fois")
                lire(N)
        FinTantque
        écrire("Bien saisie")
Fin
```

Commentaire : Regarder bien que l'instruction `écrire("Essayer encore une fois")` et celle `lire(N)` se répète tant que le nombre saisie vérifie les conditions `N<1 OU N>3`, lorsqu'elles ne sont plus vérifiées, on quitte la boucle et on exécute `écrire("Bien saisie")`.

```
Variables : N : entier
Début :
        Répéter
                écrire("Donner un nombre")
                lire(N)
        Jusquà (N>1 ET N<3)
        écrire("Bien saisie")
Fin
```

Commentaire : De même ici, les instructions qui permet de lire et saisir la valeur de `N` se répètent jusqu'à la condition `N>1 ET N<3` soit vérifiée, on quitte la boucle ensuite, et on exécute `écrire("Bien saisie")`.

Écrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : `Plus petit!`, et inversement, `Plus grand!` si le nombre est inférieur à 10.

```
Variables : N : entier
Début :
        Répéter
                écrire("Donner un nombre")
                lire(N)
                Si (N>20) Alors :
                        écrire("Plus petit!")
                Sinon
                        écrire("Plus grand!")
                FinSi
        Jusquà (N>1 ET N<3)
        écrire("Bien saisie")
Fin
```

Écrire un algorithme qui demande à l'utilisateur d'entrer deux nombres, puis l'informer si leur produit est nul, négatif ou positif. Ne calculer pas le produit!

```
Variables : x,y : entier
Début :
        écrire("Saisir deux nombres")
        lire(x,y)
        Si (x=0 ET y=0) Alors
                écrire("Le produit est nul")
        Sinon
                Si ((x<0 ET y>0) OU (x>0 ET y<0))
                        écrire("Le produit est négatif")
                Sinon
                        écrire("Le produit est positif")
                FinSi
        FinSi
Fin
```

Écrire un algorithme permettant de lire un nombre entier n puis de calculer son factoriel. On rappelle que :

$$n! = \prod_{i=1}^{n} i = 1 \times 2 \times \cdots \times n$$

```
Variables : n,s,i : entier
Début :
        écrire("Saisir un nombre")
        lire(n)
        s← 1
        Si (n=0) Alors
                écrire("n!=",s)
        Sinon
                Pour (i) Allant De (1) À (n)
                        s← s*i
                FinPour
                écrire("n!=",s)
        FinSi
Fin
```

Écrire un algorithme qui permet d'identifier le premier réel n, vérifiant la somme de 1 à n dépasse strictement 100, c'est à dire :

$$\sum_{i=0}^{n} > 100$$

```
Variables : s,i : entier
Début :
        i← 0
        s←0
        Tantque (s<= 100)
                i← i+1
                s← s+i
        FinTantque
        écrire("la valeur cherché de n est:",i)
Fin
```

Écrire un algorithme qui permet de calculer et d'afficher les premières valeurs de n et m produisant une somme s strictement supérieur à 5, on donne :

$$s = \frac{1}{4} + \frac{3}{7} + \frac{5}{10} + \cdots + \frac{n}{m}$$

```
Variables : s,m,n : entier
Début :
        s← 0
        n← 1
        m← 4
        Tantque (s<= 5)
                n← n+2
                m← m+3
                s← s+n/m
        FinTantque
        écrire("la valeur cherché de n est:",n,"et celle de m est",m)
Fin
```

Écrire un algorithme qui permet de calculer la puissance d'un nombre réel x^n (x et n donnés par l'utilisateur).
On peut résoudre ce problème en utilisant trois démarches, on va utiliser **la boucle pour**. On peut le résoudre aussi en utilisant les deux boucles **la boucle jusqu'à** et **Tant que**.

```
Variables : x, x_n : réel
            i, n : entier
Début :
       écrire("Donner un nombre réel")
       lire(x)
       écrire("Donner un nombre entier")
       lire(n)
       x_n← 1
       Pour (i) Allant De (1) À (n)
              x_n←x_n*x
       FinPour
       écrire(x,"à la puissance",n,"est",x_n)
Fin
```

Écrire un algorithme qui demande à l'utilisateur un nombre de départ et qui affiche par la suite les dix nombres qui le suivent.

```
Variables : n,i : entier
Début :
       écrire("Donner un nombre")
       lire(n)
       Pour (i) Allant De (1) À (10)
              n ←n+1
              écrire("n+",i,"est",n)
       FinPour
Fin
```

Écrire un algorithme qui demande un nombre de départ, et qui ensuite affiche sur l'écran sa table de multiplication de 1 à 10 :

```
Variables : n,i: entier
Début :
       écrire("Donner un nombre")
       lire(n)
       Pour (i) Allant De (1) À (10)
              écrire("n*",i,"=",n*i)
       FinPour
Fin
```

Les tableaux : Ensemble de donné de même type (réels, entiers, caractères...), pour cela les tableaux sont des variables, et pour les déclarer on utilise :

```
Variables : T[i][j]: (entier, réel...)
```

Où `i` est le nombre de lignes, et `j` est le nombre de colonnes du tableau `T`. On peut considérer ce tableau comme une matrice tel que :

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1j} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \cdots & a_{ij} \end{bmatrix}$$

Il faut mentionner que l'emplacement de a_{11} est `T[0][0]`, le tableau commence son positionnement de la position 0. C'est-à-dire : a_{ij} est `T[i-1][j-1]`.
Pour remplir et afficher un tableau simple de taille choisie par l'utilisateur on écrit l'algorithme suivant :

```
Variables : i, n: entier
          T[n]: réel
Début :
       écrire("Donner la taille de votre tableau")
       lire(n)
       Pour (i) Allant De (0) À (n-1)
              T[i]← 0
       FinPour
       Pour (i) Allant De (0) À (n-1)
              écrire("La valeur de la case",i+1,"est",T[i])
       FinPour
Fin
```

Pour permettre à l'utilisateur d'entrer les éléments de son tableau et de l'afficher, on utilise dans la première boucle les instructions `écrire("Donner l'élément de la case", i+1)` et `lire(T[i])`.
Comme le présente l'algorithme suivant :

```
Variables : i, n: entier
          T[n]: réel
Début :
       écrire("Donner la taille de votre tableau")
       lire(n)
       Pour (i) Allant De (0) À (n-1)
              écrire("Donner l'élément de la case", i+1)
              lire(T[i])
       FinPour
       Pour (i) Allant De (0) À (n-1)
              écrire("La valeur de la case",i+1,"est",T[i])
       FinPour
Fin
```

Écrire un algorithme qui permet à l'utilisateur d'entrer les notes de 9 matières, et l'afficher, ensuite on calcule la moyenne de ces notes.

```
Variables : i, T[9], moy: entier
Début :
       moy ←0
       Pour (i) Allant De (0) À (8)
              écrire("Donner la note", i+1)
              lire(T[i])
              moy ←moy+T[i]
       FinPour
       Pour (i) Allant De (0) À (8)
              écrire("La note de",i+1,"matière est",T[i])
       FinPour
       moy ←moy/9
       écrire("La moyenne est :",moy)
Fin
```

Écrire un algorithme qui permet à l'utilisateur d'entrer un nombre quelconque de valeurs, qui doivent être stockées dans un tableau.
L'utilisateur doit donc entrer les valeurs de son tableau, une fois qu'il termine ceci, le programme affiche le nombre des valeurs positives et négatives.

```
Variables : N, T[N], cpt1, cpt2, i: entier
Début :
        écrire("Donner la taille du tableau")
        lire(N)
        Pour (i) Allant De (0) À (N-1)
                écrire("Donner la valeur de la case",i+1)
                lire(T[i])
        FinPour
        cpt1← 0
        cpt2←0
        Pour (i) Allant De (0) À (N-1)
                Si (T[i]<0) Alors
                        cpt1← cpt1+1
                Sinon
                        cpt2← cpt2+1
                FinSi
        FinPour
    écrire("Le nombre des valeurs positives :", cpt2,"et négatives", cpt1)
Fin
```

Écrire un algorithme qui à partir des deux tableaux suivants : `[4,8,7,9,1,5,4,6]` et `[7,6,5,2,1,3,7,4]` on obtient un nouvel tableau dont ses éléments est la somme des deux premiers tableaux.

```
Variables : T1[8],T2[8],T3[8], i: entier
Début :
        T1 ←[4, 8, 7, 9, 1, 5, 4, 6]
        T2 ←[7, 6, 5, 2, 1, 3, 7, 4]
        Pour (i) Allant De (0) À (7)
                T3[i]←T1[i]+T2[i]
        FinPour
        Pour (i) Allant De (0) À (7)
                écrire("La case",i+1,"est",T3[i])
        FinPour
Fin
```

Afin d'obtenir la valeur maximale d'un tableau et son indice.

```
Variables : N, T[N], i, max, indice: entier
Début :
        écrire("Donner la taille du tableau")
        lire(N)
        Pour (i) Allant De (0) À (N-1)
                écrire("Donner la valeur de la case", i+1)
                lire(T[i])
        FinPour
        max← T[0]
        indice←0
        Pour (i) Allant De (0) À (N-1)
                Si (T[i]>max) Alors
                        max← T[i]
                        indice←i
        FinPour
        écrire("Le maximum de ce tableau est", max,"dans la position",indice)
Fin
```

Écrire un algorithme qui permet à l'utilisateur de taper 5 entiers qui seront stockés dans le tableau. Le programme doit trier le tableau par ordre croissant, et l'afficher par la suite :

```
Variables : T[5], i,j, temp: entier
Début :
        Pour (i) Allant De (0) À (4)
                écrire("Donner la valeur de la case", i+1)
                lire(T[i])
        FinPour
        Pour (i) Allant De (0) À (3)
                Pour (j) Allant De (0) À (4)
                        Si (T[i]>T[j]) Alors
                                Temp← T[j]
                                T[j]←T[i]
                                T[i]←Temp
                        FinSi
                FinPour
        FinPour
        écrire("Le nouveau tableau est")
        Pour (i) Allant De (0) À (4)
                écrire("La case", i+1,"est", T[i])
        FinPour
Fin
```

Maintenant essayons de permuter deux tableaux de tailles 5

```
Variables : T1[5],T2[5],Temp[5],i:entier
Début :
        Pour (i) Allant De (0) À (4)
                écrire("Donner la valeur de la case", i+1,"du premier tableau")
                lire(T1[i])
                écrire("Donner la valeur de la case", i+1,"du deuxième tableau")
                lire(T2[i])
        FinPour
        écrire("Avant la permutation")
        Pour (i) Allant De (0) À (4)
                écrire("T1[",i+1,"]est",T1[i])
        FinPour
        Pour (i) Allant De (0) À (4)
                écrire("T2[",i+1,"]est",T2[i])
        FinPour
        Pour (i) Allant De (0) À (4)
                Temp[i]← T1[i]
                T1[i]← T2[i]
                T2[i]← Temp[i]
        FinPour
        écrire("Après la permutation")
        Pour (i) Allant De (0) À (4)
                écrire("T1[",i+1,"]est",T1[i])
        FinPour
        Pour (i) Allant De (0) À (4)
                écrire("T2[",i+1,"]est",T2[i])
        FinPour
Fin
```

Pour les matrices on suit la même démarche, en utilisant seulement une boucle imbriquée.

```
Variables :M, N, T[M][N]n i, j:entier
Début :
        écrire("Donner le nombres de lignes")
        lire(M)
        écrire("Donner le nombre de colonnes")
        lire(N)
        Pour (i) Allant De (0) À (M-1)
                Pour (j) Allant De (0) À (N-1)
                        écrire("Donner la valeur de T[",i,"][",j,"]")
                        lire(T[i][j])
                FinPour
        FinPour
        Pour (i) Allant De (0) À (M-1)
                Pour (j) Allant De (0) À (N-1)
                        écrire("La valeur de T[",i,"][",j,"]", T[i][j])
                FinPour
        FinPour
Fin
```