TP2

Exercice 1.
Écrire un programme `racines` qui demande à l'utilisateur trois réels `a, b` et `c`, qui calcule les racines du polynôme $ax^2 + bx + c$ et qui les écrit à l'écran.
Attention à bien prendre en compte tous les cas possibles et accompagner l'affichage des résultats d'un message parlant.
Conseil : Commencer par programmer le cas où il y a deux racines réelles puis intégrer les autres cas un à un.

Exercice 2.
Créer un répertoire `Exercice2` et se placer à l'intérieur. Dans un fichier `exo2.f90` écrire un programme qui calcule une approximation de la limite quand $n \to \infty$ de la série $\sum_n \frac{1}{n^3}$. Ce programme écrira à l'écran la valeur de la limite et le nombre d'itérations effectuées pour l'atteindre.

Exercice 3.
Soit le complexe $j = \frac{-1}{2} + i\frac{\sqrt{3}}{2}$. Ecrire un programme qui calcule la somme des j^n avec $n = 20, 2009, 2010$. Imprimer à l'écran les différents résultats.

Exercice 4.
Créer un répertoire `Exercice4` et se placer à l'intérieur.
Dans un fichier `exo4.f90` :

1. Définir un type dérivé `cercle` qui contient un tableau de réels de taille 2 nommé `centre` et un réel `rayon`
2. Déclarer une variable de type cercle nommé `c` et initialiser-la.
3. Imprimer à l'écran le champ `rayon` de `c`
4. Si $cercle1$ est une variable de type cercle, quel est le type de $c\%centre(1)$?

Exercice 5.
Le crible d'Ératosthène permet d'obtenir les nombres premiers inférieurs à un entier `n` donné.
Son principe est relativement simple : on parcourt les entiers de 2 à `n` et dès qu'on trouve un nombre premier, on raye tous ses multiples inférieurs à `n`.
Écrire un programme `eratosthene` qui permet d'obtenir les nombres premiers de 1 à `n` :

1. On commencera par fixer `n`=100. On utilisera un tableau de booléens `premier` de taille `2:n` initialisé à vrai. En parcourant les éléments de ce tableau, programmer le crible "naïvement". Afficher la liste des nombres premiers.

2. Améliorer le programme précédent en réduisant la recherche aux nombres entre 2 et la racine carrée de `n`.
3. Modifier le programme pour réaliser le crible pour `n` choisi par l'utilisateur.

Exercice 6.
Créer un répertoire `Exercice6` et se placer à l'intérieur. Dans un fichier `exo6.f90` en utilisant le développement limité de la fonction exponentielle, écrire un programme qui calcule une approximation du nombre irrationnel e.

Exercice 7.

1. Soient les matrices A et B de taille 4*4 définies par : $A_{ij} = (-4)^i + 6 * j$ et $B_{ij} = 3 + 5 * i * j$
   Dans un fichier `matrice.f90`,
   (a) Calculez la somme de A et B et stockez-la dans un tableau nommé `matsom`
   (b) Calculez le produit de A par B et stockez-le dans un tableau nommé `matprod`
   (c) Imprimez à l'écran les résultats en présentant les matrices lignes par lignes comme on a l'habitude de les écrire en maths.
2. Dans un fichier `matricedynamique.f90`,
   (a) Demandez à l'utilisateur de donner un entier `n` et lisez-le au clavier.
   (b) Initialisez des matrices A et B définies par $A_{ij} = (-4)^i + 6 * j$ et $B_{ij} = 3 + 5 * i * j$
   (c) Calculez la somme de A et B et stockez-la dans un tableau nommé `matsom`
   (d) Calculez le produit de A par B et stockez-le dans un tableau nommé `matprod`
   (e) Imprimez à l'écran les résultats en présentant les matrices lignes par lignes comme on a l'habitude de les écrire en maths.

Exercice 8. Écrire un programme tri_bulle qui permet de trier les éléments d'un vecteur par ordre croissant puis décroissant.
Le tri à bulle consiste à comparer 2 éléments consécutifs du tableau et à les intervertir si nécessaire. On parcourt tout le tableau et si à la fin du parcours au moins une interversion a été effectuée, on recommance.

Déclarer trois tableaux de réels `tab, tab_c` et `tab_d` de taille `n`, fixé dans un premier temps à 10.
Remplir `tab` par des nombres "aléatoires" en utilisant l'instruction `Call random_Number(tab)`.
Puis remplir les tableaux `tab_c` et `tab_d` en effectuant deux tris à bulle.
Afficher les trois tableaux à l'écran.

Exercice 9.

1. Dans un fichier som1.f90 écrire un programme qui demande à l'utilisateur un entier `n` et un réel `k`, qui calcule le réel `Skn` défini par `Skn`=$\sum_{i=1}^{n} i^k$ et qui affiche le résultat à l'écran.
2. Dans un fichier som2.f90 écrire un programme qui demande à l'utilisateur un entier `k`, qui calcule le réel *simple précision* `Sk` et qui affiche le résultat à l'écran.
   `Sk` est défini par `Sk`=$\sum_{i=1}^{\infty} i^k$. Pour l'obtenir, on utilisera un deuxième réel simple précision

`Sk1` qui contient la somme partielle `Sk(n+1)` et calculer jusqu'à ce que `Sk` et `Sk1` soient différents ou que `Sk`$> 10^{10}$.

3. Dans un fichier som3.f90, définir un type dérivé `som` qui comprend 3 tableaux de taille 17 :
   — un tableau de réels `exposant`,
   — un tableau de réels `somme`,
   — un tableau de booléens `converge`.
   Puis, remplir une variable `masomme` de type `som` de la manière suivante :
   — `exposant` contient les nombres de -2 à 2 par pas de 0.25,
   — `somme`(i) =`Sk` calculé comme à la question précédente avec `k=exposant`(i),
   — `converge`(i) est vrai si `somme`(i) $\leq 10^{10}$ et faux sinon.
   Puis pour chaque élément i de `exposant` afficher à l'écran :
   — "La somme pour l'exposant" `exposant(i)` "vaut" `somme(i)` si `converge(i)` est vrai,
   — "La somme pour l'exposant" `exposant(i)` "diverge" sinon.