Série d'exercices : LES SOUS-PROGRAMMES

Exercice n°1 :

1) Factorielle

Faire une fonction **facto(n)** qui renvoie ***n!*.**

2) Puissance

Faire une fonction **puiss(x,n)** qui renvoie $\mathbf{x^n}$.

3) Exponentielle

$$e^x \simeq 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

Faire une fonction **expn(x,n)** qui calcule la valeur approchée de $\mathbf{e^x}$ en faisant appel aux fonctions **facto** et **puiss**.

Exercice n°2 :

1-Ecrire une procédure à trois paramètres entiers qui fait la somme des deux premiers et range cette valeur dans le troisième.

2-Transformer la procédure précédente en fonction.

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \cdots$$

Exercice n°3 :

Faire une fonction ***calcpi*** qui calcule la valeur approchée de π en s'arrêtant lorsque le terme **1 / x** est plus petit que ε.

Faire un programme qui lit ε réel, appelle ***calcpi*** puis affiche le résultat.

Exercice n°4 :

Soit la suite récurrente *U* suivante définie par

$$\begin{cases} U_1 = 4 \\ U_2 = 2 \\ U_3 = 1 \\ U_n = \sqrt{2U_{n-3} + U_{n-1}} + U_{n-2} \end{cases}$$

Ecrire une procédure algorithmique nommée ***PSuite*** qui calcule le *n-ième* terme de *U*.

Exercice n°5 :

A partir d'un entier n strictement positif, on cherche à se ramener à 1, pour cela on applique la méthode suivante : **« Suite de SYRACUS »**

- Si n est pair on le remplace par $\frac{n}{2}$
- Si n est impair on le remplace par $3n+1$

Et ainsi de suite jusqu'à ce que l'on arrive à 1.

Ecrire une procédure algorithmique ***SYRACUS***, permettant de compter et d'afficher le nombre d'itérations nécessaires pour le ramener un entier n strictement positif à 1.

Exercice n°6 : « Algorithmes des babyloniens »

Supposons que l'on cherche à calculer une valeur approchée de la racine carrée $\sqrt{a}$ du nombre a. On considère la suite récurrente suivante :

$$\begin{cases} u_0 = 1 \\ u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right) \end{cases}$$

Le calcul des termes s'arrête lorsque la différence entre deux termes consécutifs soit inférieure à 10^{-8}.

Ecrire une procédure algorithmique ***BABYLONE*** qui permet de calculer puis d'afficher la racine carrée d'un réel positif a donné.

Exercice n°7 : « Suite de Feigenbaum »

On considère la suite récurrente (suite de Feigenbaum) définie par :

$$\begin{cases} u_0 \in \square \\ u_{n+1} = a u_n \left(1 - u_n\right) \end{cases} \quad \text{où } a \text{ est un nombre réel, } a \in [0,4]$$

Ecrire une procédure algorithmique **FEIGNBAUM**, qui à partir d'un entier n calcule et affiche la valeur de u_n.

Exercice n°8 : « Algorithme à Euclide »

L'algorithme dû à Euclide est un algorithme qui détermine le PGCD de deux nombres naturels a et b.

Le processus de l'algorithme est le suivant :

- ✓ Si un des nombres est nul, l'autre est le PGCD.
- ✓ Sinon il faut soustraire le plus petit du plus grand et laisser le plus petit inchangé.
- ✓ Puis, recommencer ainsi avec la nouvelle paire jusqu'à ce qu'un des deux nombres soit nul.
- ✓ Dans ce cas, l'autre nombre est le PGCD.

Ecrire une procédure permettant calculer et afficher le PGCD de deux nombres a et b en utilisant l'algorithme à l'Euclide.

Exercice n°9 : « Polynômes de TChebychev »

Soit $T_n \in IR_n[X]$ défini par :

$$T_0 = 1,\ T_1 = X \ \text{ et } \forall n \geq 2,\ T_n = 2XT_{n-1} - T_{n-2}.$$

Ecrire une procédure permettant de calculer le $n^{ème}$ polynôme de Tchebychev avec n donné tel que $n \geq 2$.

Exercice n°10:

- On appelle ***diviseur propre*** de n, un diviseur quelconque de n, n exclu; Exemple : 6 = 1 + 2 + 3
- Un entier est dit ***parfait*** s'il est égal à la somme de tous ses diviseurs propres
- On appelle nombre ***premier*** tout entier naturel supérieur a 1 qui possède exactement deux diviseurs, lui-même et 1 ;
- Les nombres premiers a tel que : $(a+n+n^2)$ est premier pour tout n tel que $0 \leq n \leq (a-1)$ sont appelées ***nombres chanceux.***

Travail demandé :

1. Ecrire une procédure nommée, ***SOMDIV*** qui calcule et renvoie dans ***sd*** la somme des diviseurs propres d'un entier n donnée.
2. Ecrire les instructions algorithmiques (on ne demande pas d'écrire un algorithme principal) permettant d'afficher les nombres parfaits de l'intervalle [2,1000].
3. Ecrire une fonction nommée, ***estpremier,*** à résultat booléen permettant de vérifier si un entier ***e*** donné est premier ou non.
4. Ecrire une fonction nommée, ***estchanceux,*** à résultat booléen permettant de vérifier si un entier ***a*** donné est chanceux ou non.