TP6

Récursivité

I Récursivité

Une fonction récursive est une fonction qui s'appelle elle-même dans le corps de sa définition. Il est alors nécessaire de prévoir un test d'arrêt dans cette fonction sans quoi elle s'appellerait elle-même indéfiniment.
On peut faire le lien avec la notion de récurrence lorsque le paramètre est entier.

- on précise la valeur de $f(0)$
- Pour $n \geqslant 1$, on calcule $f(n)$ à partir de $f(n-1)$.

Une fonction récursive s'écrit donc en deux temps :

- Si le test est vérifié, on renvoie le résultat correspondant.

- Sinon, on lance un appel récursif sur la fonction (avec un nouveau paramètre).

II Exercices

Exercice 1

1. On souhaite écrire une fonction récursive qui calcule $n!$.
   (a) Ecrire une relation entre $n!$ et $(n-1)!$.
   (b) Quel est le test d'arrêt ?
   (c) Ecrire alors une fonction récursive `factrec(n)` permettant de calculer $n!$.
   (d) Détailler l'éxécution de `factrec(5)`.
2. Ecrire une fonction récursive `puissancerec(x,n)` permettant de calculer x^n.
3. Ecrire une fonction récursive `dessinrec(n)` permettant de faire le dessin suivant (n correspondant au nombre de lignes).

```
****
***
**
*
```

Exercice 2

On considère la suite $(u_n)_{n\geqslant 0}$ définie par : $u_0 = 2$ et $u_{n+1} = \sqrt{u_n + \dfrac{1}{u_n}}$.

1. Ecrire une fonction récursive `U(n)` permettant de calculer le terme u_n.
2. Ecrire une fonction itérative `Ubis(n)` permettant de calculer le terme u_n.
3. En utilisant la fonction `time()` du module time, comparer les temps d'éxécution de ces deux fonctions.

Exercice 3

Écrire une fonction récursive `Syracuse(n)` qui prend en entrée un entier n strictement positif et qui affiche tous les termes de la suite de Syracuse à partir de n jusqu'à l'apparition du premier 1.

Rappel : la suite de Syracuse $(u_n)_{n\in\mathbb{N}}$ est définie par son premier terme u_0 dans $\mathbb{N}^*$, et pour tout $n \in \mathbb{N}$,

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair,} \\ 3u_n + 1 & \text{si } u_n \text{ est impair.} \end{cases}$$

Exercice 4

Écrire une version récursive de la recherche dichotomique d'un élément dans une liste triée.

On rappelle que pour rechercher la valeur `v` dans la liste `L`, on la compare avec la valeur `L[m]` située au milieu de la liste (d'indice `m`) :

- Si `L[m] = v`, la valeur `v` est trouvée et on renvoie sa position `m`.
- Si `v < L[m]`, on recommence la recherche dans la première moitié de la liste.
- Sinon, on recommence la recherche dans la seconde moitié de la liste.

On écrira une fonction `recherche_dichotomique(L, v, g, d)` où `L` est une liste supposée triée par ordre croissant, `v` est la valeur recherchée, `g` et `d` étant les indices extrêmes (gauche et droit) de la liste dans laquelle on cherche `v`.

Exercice 5

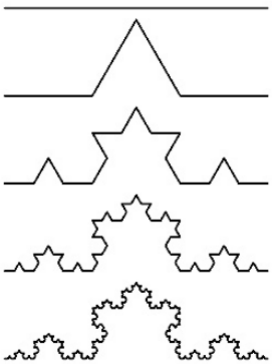
Écrire une fonction `Puissance_Rapide(x,n)` récursive de l'algorithme d'exponentiation rapide qui repose sur les relations suivantes :

- Si n est pair, $x^n = (x^2)^{n//2}$
- Si n est impair, $x^n = x(x^2)^{n//2}$

Fractales

Exercice 6 (*Courbes de Von Koch*)

Le but de cet exercice est de tracer les courbes de Von Koch illustrées ci-dessous.



Construction des courbes de Von Koch.

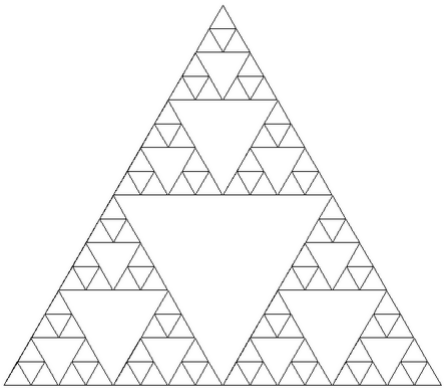
- La courbe de Von Koch d'ordre 1 est un segment de longueur L donnée.
- La courbe de Von Koch d'ordre 2 s'obtient en divisant le segment d'origine en trois et en remplaçant le segment du milieu par un triangle équilatéral sans sa base. Il est donc constitué de quatre segments de longueur $L/3$ comme indiqué sur le deuxième dessin.
- La courbe de Von Koch d'ordre 3 s'obtient en divisant chaque segment de la courbe de Von Koch d'ordre 2 comme décrit à l'étape précédente. Il est donc constitué de seize segments de longueur $L/9$ comme indiqué sur le troisième dessin.
- et ainsi de suite...

Ainsi, chaque "segment" de la courbe de Von Koch d'ordre n est constitué d'une courbe de Von Koch d'ordre $n-1$.

1. Nous utiliserons le module `turtle` qui permet de tracer des figures.
   Télécharger et ouvrir le fichier TP6_Turtle.py. Lire, exécuter et comprendre le code.
2. Écrire une fonction récursive `courbevonkoch(n, L)` qui prend en entrée deux entiers n et L et qui renvoie la courbe de Von Koch d'ordre n de segment initial de longueur L.

Exercice 7 (*Triangles de Sierpinski*)

Le but de cet exercice est de tracer les triangles de Sierpinski illustrées ci-dessous.



Construction des triangles de Sierpinski.

- Le triangle de Sierpinski d'ordre 1 est un triangle équilatéral de côté L donnée.
- On passe des triangles de Sierpinski d'ordre n à ceux d'ordre $n+1$ en reliant les milieux des côtés de
- et ainsi de suite

Écrire une fonction récursive `tiangle(n, L)` qui prend en entrée deux entiers n et L et qui renvoie le triangle de Sierpinski d'ordre n de triangle initial de côté de longueur L.