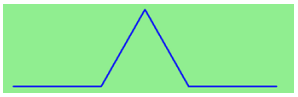
Recursão

- Vimos nas aulas passadas o que é recursão: resolução de um problema usando a solução de um problema semelhante (geralmente de menor dimensão);
- Um fractal é uma figura que tem estruturas semelhantes a si própria;
- Está relacionado com a recursão, i.e., de qualquer coisa definida em termos de si própria;

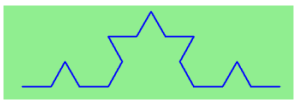
Fractal de Koch

- Ordem 0: linha reta, com uma determinada dimensão:

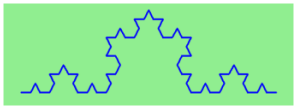
- Ordem 1: em vez da linha reta, quatro segmentos:



- Ordem 2: em vez de cada linha reta, quatro segmentos:



- Ordem 3: repetição do padrão:



Fractal de Koch

- Pensando por ordem inversa:
  1. para desenhar o fractal de ordem 3, desenhar quatro fractais de ordem 2;
  2. para desenhar cada fractal de ordem 2, desenhar quatro fractais de ordem 1;
  3. para desenhar cada fractal de ordem 1, desenhar quatro fractais de ordem 0;
- Em última instância, apenas temos que desenhar fractais de ordem 0.

Programa em Python

```
def koch(t, order, size):
    if order == 0:   # The base case is just a straight line
        t.forward(size)
    else:
        koch(t, order-1, size/3)   # go 1/3 of the way
        t.left(60)
        koch(t, order-1, size/3)
        t.right(120)
        koch(t, order-1, size/3)
        t.left(60)
        koch(t, order-1, size/3)

fred = turtle.Turtle()
wn = turtle.Screen()

fred.color("blue")
wn.bgcolor("green")
fred.penup()
fred.backward(150)
fred.pendown()

koch(fred, 3, 300)

wn.exitonclick()
```


Recursão

- Testar:
  1. correr o programa para valores de `order` diferentes: 0, 1, 2, ...
  2. quando `order` é diferente de zero, `koch` chama-se a si própria quatro vezes;
  3. esta auto-referência é o que constitui a recursão.
- Visão de alto nível
  - Ponto de partida: a função funciona para ordem 0;
  - Acreditando nisso: deixamos de nos preocupar com o detalhe;
  - Para desenhar o fractal de ordem n, apenas temos que assumir que já está a funcionar para ordem $n-1$
- Abstração: ignorar os pequenos problemas enquanto resolvemos o problema grande.
- Semelhante a uma prova por indução, em matemática.

Dados recursivos

- Organização de dados por forma a ficarem mais fáceis de usar: estrutura de dados.
- Problema:
  - Há eleições, e a nossa missão é contar os votos à medida que eles chegam;
  - Os votos chegam de diferentes freguesias, concelhos, e distritos;
  - Os votos são transmitidos umas vezes como a sua soma, outras como uma listas dos subtotais;
  - Decidiu-se guardar os votos como uma lista imbricada de números, que contém:
    - números
    - listas imbricadas de números
  - Exemplo: `[10,[10,2,1,[5],2],[[1,5],10]]`

Definição recursiva de dados

- Esta *definição recursiva* é comum em matemática e em ciência de computadores, descrevendo estruturas de dados recursivas;
- A definição não é circular: em algum ponto, obtém-se uma lista que não contém sublistas.
- A nossa missão é, portanto, a de escrever uma função que some todos os elementos de uma lista imbricada de números.
- Como a estrutura de dados é recursiva, o mais natural é pensarmos numa função também recursiva.

Soma dos elementos de uma lista imbricada de números

1. Identificar casos em que já conhecemos a resposta (casos base):
   - se a lista é vazia, a soma dos seus elementos é zero;
   - se não é vazia, tem um primeiro elemento;
   - se retirarmos o primeiro elemento, o resto é uma lista (com menos um elemento).
2. Para tratarmos do primeiro elemento:
   - se for um inteiro: adicionamo-lo ao resultado retornado para o resto da lista;
   - se for uma lista:
     - primeiro calculamos a sua soma (já o sabemos fazer);
     - depois adicionamo-la ao resultado retornado para o resto da lista.

Código Python

```
def rSum(nestedNumList):
    if nestedNumList == []:
        return 0
    else:
        firstitem = nestedNumList[0]
        if type(firstitem) == type(87): # is an integer
            return firstitem + rSum(nestedNumList[1:])
        else:
            return rSum(firstitem) + rSum(nestedNumList[1:])


print(rSum([]))
print(rSum([1,2,3,4]))
print(rSum([1,2,[4,5,6],7,8]))
```


Noções estudadas esta semana

caso base ramo numa função recursiva que não dá origem a mais chamadas recursivas

chamada recursiva chamada de uma função que já estava a ser executada; pode ser indireta (*f* chama *g*, que chama *h*, que chama *f*)

definição recursiva definição da alguma coisa feita em termos de si própria

estrutura de dados organização dos dados por forma a tornar o seu tratamento mais simples

recursão infinita função que se chama a si própria, sem nunca atingir um caso base

recursão processo de chamar uma função que já está a ser executada

Próximas aulas

- Recursão e fractais.
- Programação orientada a objetos.
- Aplicações.