

Curvas de Hilbert



Fonte: <http://momath.org/home/math-monday-03-22-10>

Niklaus Wirth, *Algorithms and Data Structures*
Prentice Hall, 1986.

http://en.wikipedia.org/wiki/Hilbert_curve

Curvas de Hilbert

As curvas a seguir seguem um certo **padrão regular** e podem ser desenhadas na tela sobre o controle de um programa.

O objetivo é descobrir o **esquema de recursão** para construir tais curvas.

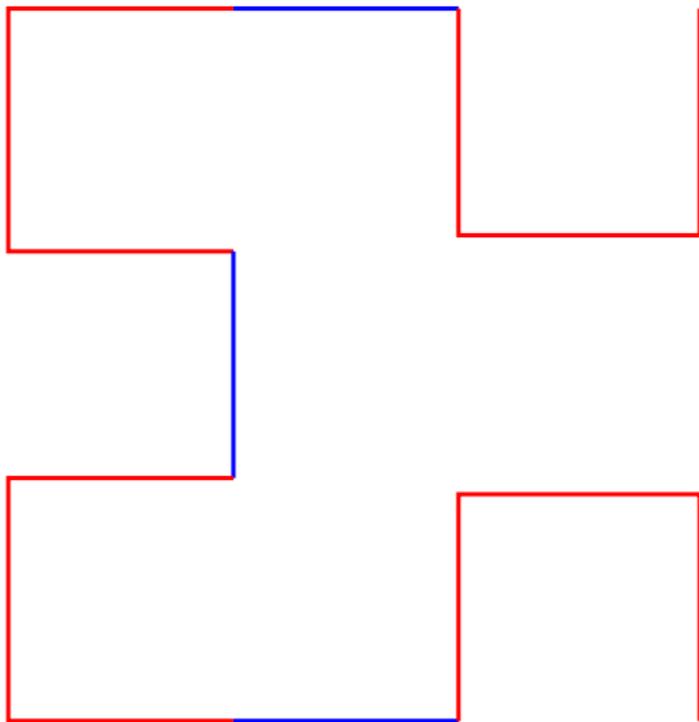
Estes padrões serão chamados de $H_0, H_1, H_2 \dots$

Cada H_i denomina a **curva de Hilbert** de **ordem i** , em homenagem a seu inventor, o matemático *David Hilbert*.

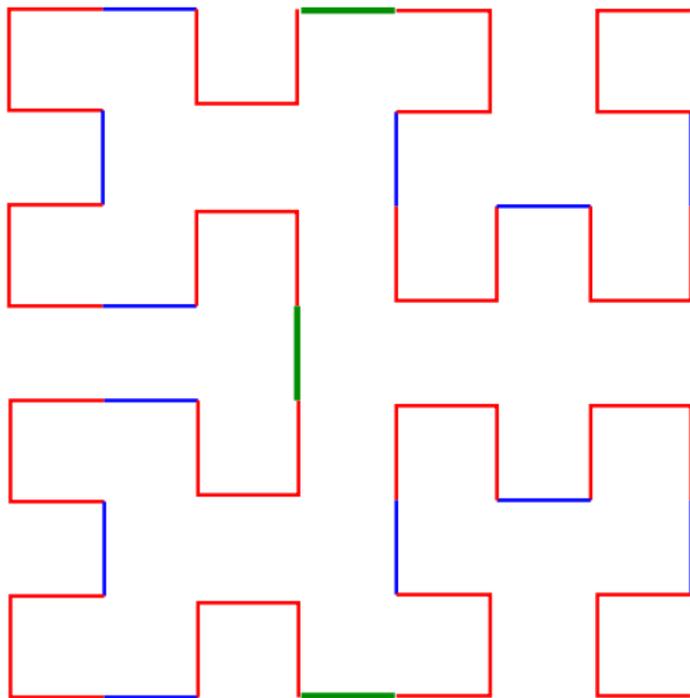
H_1



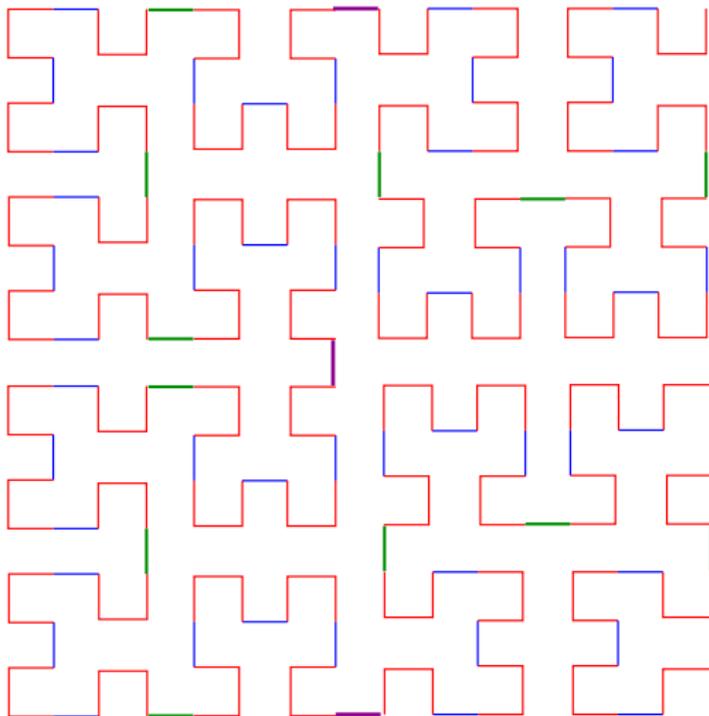
H₂



H_3



H_4



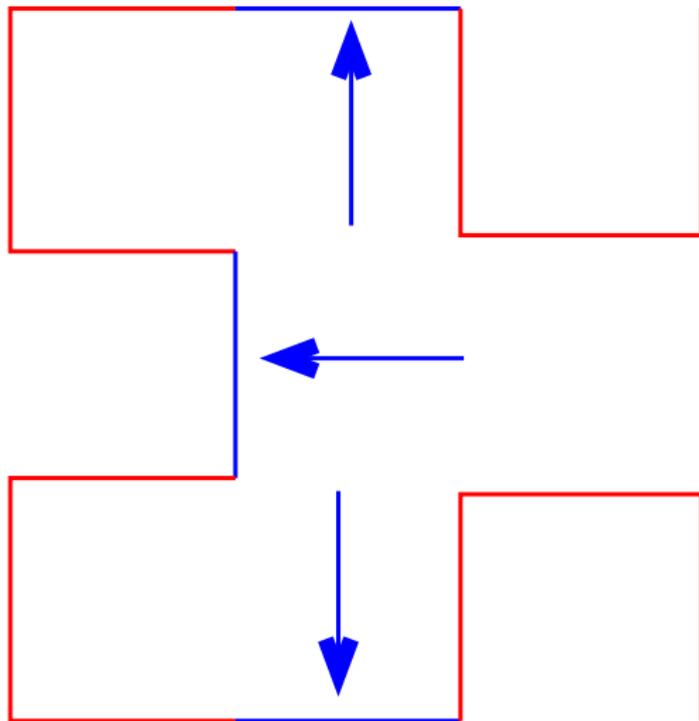
Padrão

As figuras mostram que H_{i+1} é obtida pela composição de 4 instâncias de H_i de metade do tamanho e com a rotação apropriada, ligadas entre si por meio de 3 linhas de conexão.

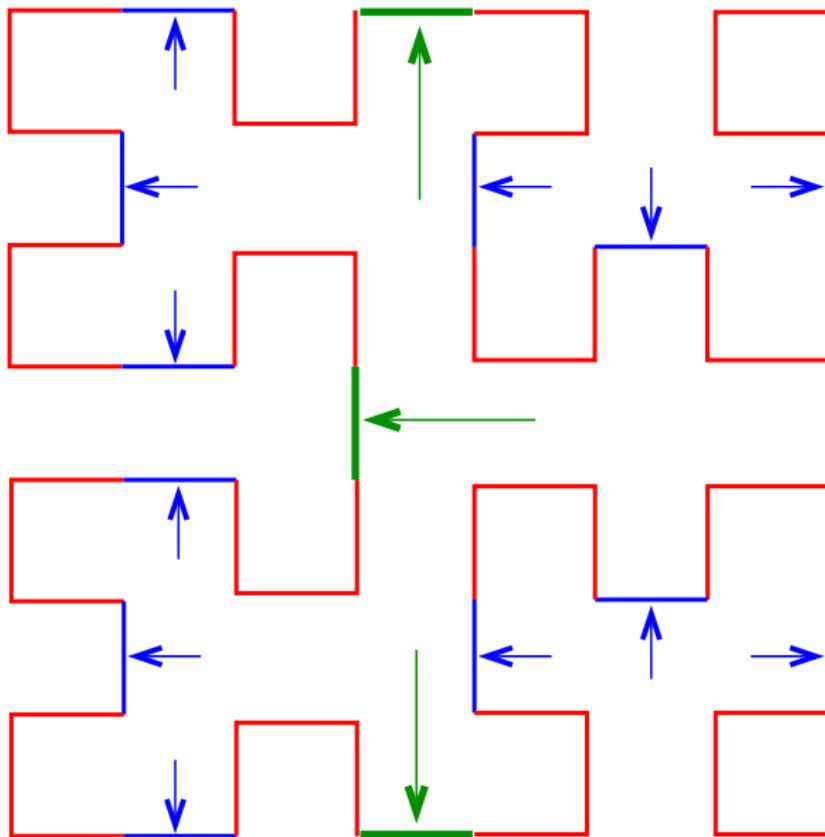
Por exemplo:

- ▶ H_1 é formada por 4 H_0 (vazio) conectados por 3 linhas.
- ▶ H_2 é formada por 4 H_1 conectados por 3 linhas
- ▶ H_3 é formada por 4 H_2 conectados por 3 linhas

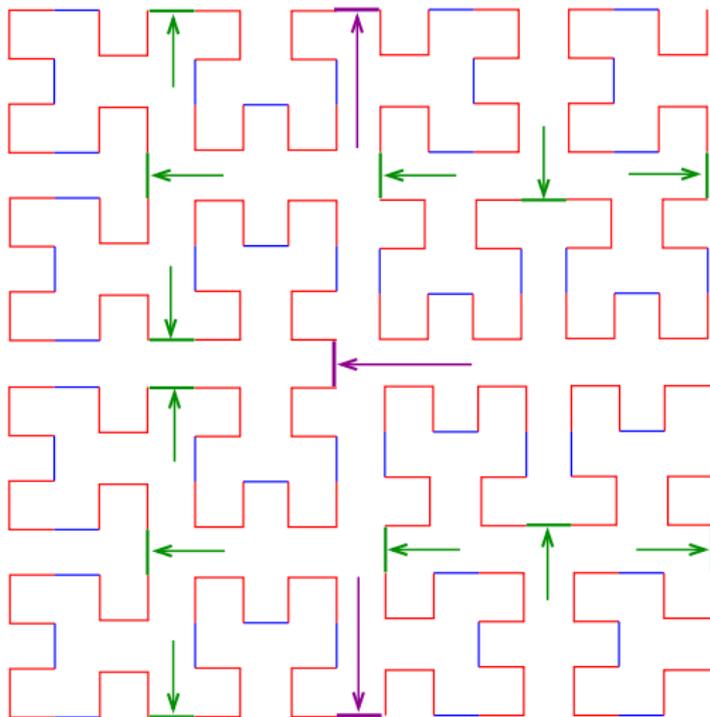
H₂



H_3



H_4



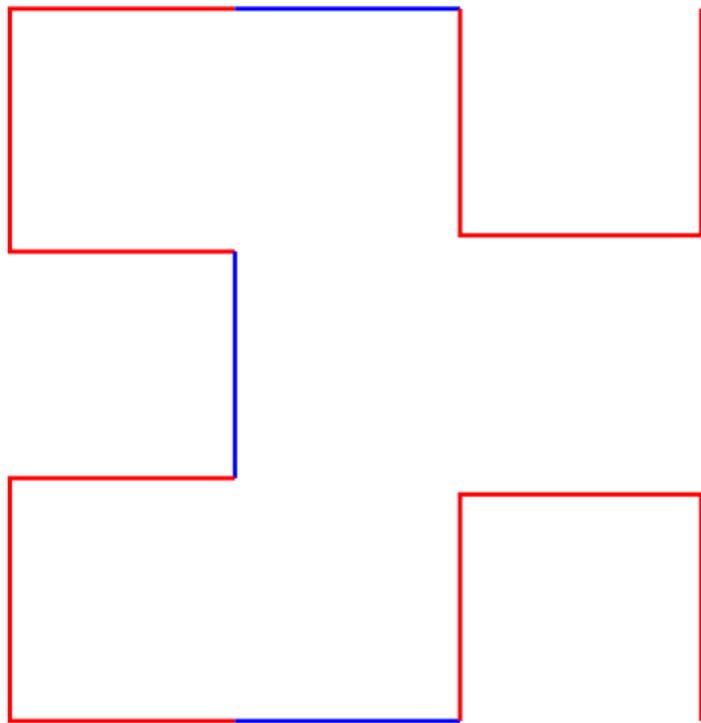
Partes da curva

Para ilustrar, denotaremos as quatro possíveis instâncias por **A**, **B**, **C** e **D**:

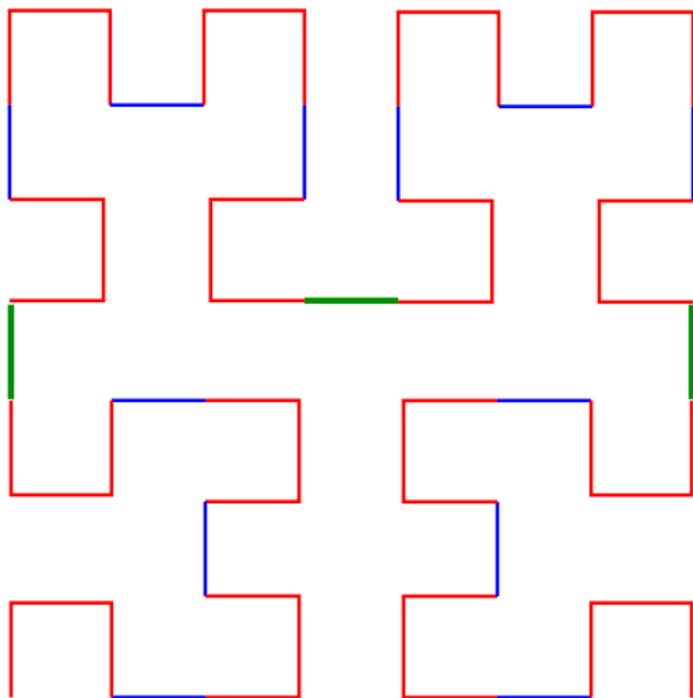
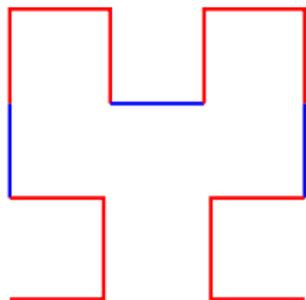
- ▶ **A** será o padrão que tem a “abertura” para **direita**;
- ▶ **B** será o padrão que tem a “abertura” para **baixo**;
- ▶ **C** será o padrão que tem a “abertura” para **esquerda**; e
- ▶ **D** será o padrão que tem a “abertura” para **cima**.

Representaremos a chamada da função que desenha as interconexões por meio das setas \uparrow , \downarrow , \leftarrow , \rightarrow .

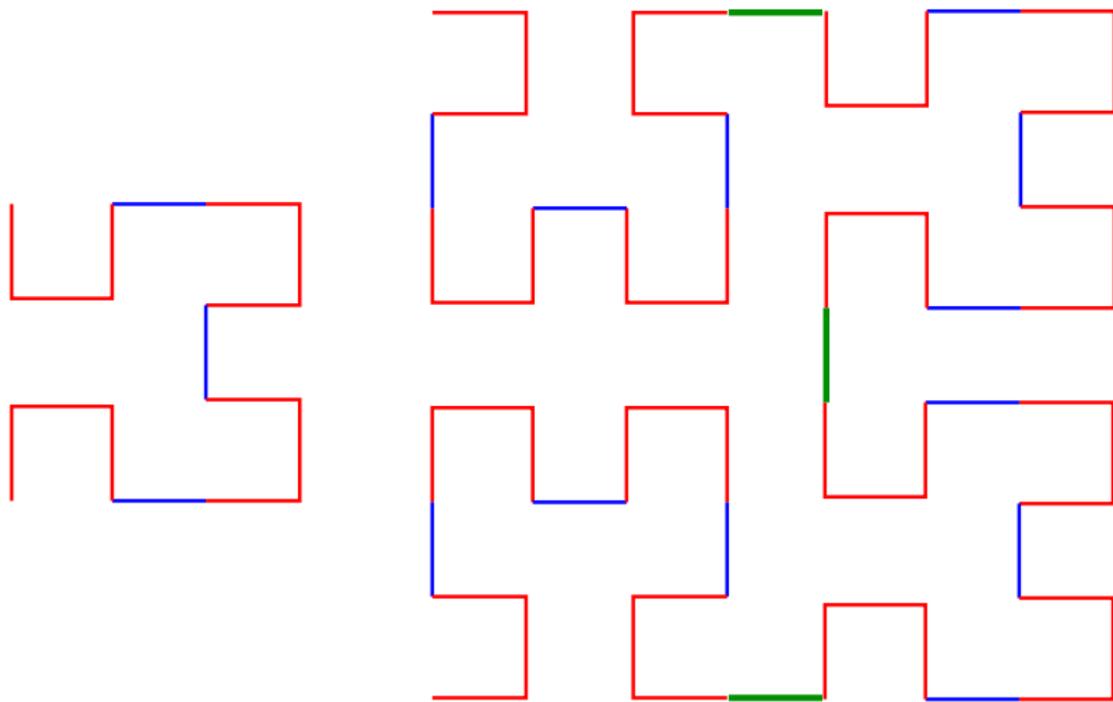
$A_1 \in A_2$



B_2 e B_3



C_2 e C_3



Esquema recursivo

Assim, surge o seguinte esquema recursivo:

$$\begin{array}{l} A_k : D_{k-1} \leftarrow A_{k-1} \downarrow A_{k-1} \rightarrow B_{k-1} \\ B_k : C_{k-1} \uparrow B_{k-1} \rightarrow B_{k-1} \downarrow C_{k-1} \\ C_k : B_{k-1} \rightarrow C_{k-1} \uparrow C_{k-1} \leftarrow D_{k-1} \\ D_k : A_{k-1} \downarrow D_{k-1} \leftarrow D_{k-1} \uparrow C_{k-1} \end{array}$$

Para desenhar os segmentos utilizaremos a chamada de uma função

```
move(pincel, direcao, comprimento)
```

que “**move um pincel**” da posição (x,y) em uma dada **direcao** por um certo **comprimento**.

move()

```
def move(pincel, direcao, comprimento):  
    # pegue a posição do pincel  
    x, y = pincel.pos()  
    # calcule nova posição  
    if direcao == DIREITA:  
        x = x + comprimento  
    elif direcao == ESQUERDA:  
        x = x - comprimento  
    elif direcao == CIMA:  
        y = y + comprimento  
    elif direcao == BAIXO:  
        y = y - comprimento  
    pincel.goto(x, y)
```

A_k

```
def a(k, pincel, comprimento):  
    '''(int, Turtle, int) -> None  
    Desenha a curva  $A_k$  a partir da  
    posição  
    do pincel.  
    '''  
    if k > 0:  
        d(k-1, pincel, comprimento)  
        move(pincel, ESQUERDA, comprimento)  
        a(k-1, pincel, comprimento)  
        move(pincel, BAIXO, comprimento)  
        a(k-1, pincel, comprimento)  
        move(pincel, DIREITA, comprimento)  
        b(k-1, pincel, comprimento)
```

B_k

```
def b(k, pincel, comprimento):  
    '''(int, Turtle, int) -> None  
    Desenha a curva B_k a partir da  
posição  
do pincel.  
    '''  
    if k > 0:  
        c(k-1, pincel, comprimento)  
        move(pincel, CIMA, comprimento)  
        b(k-1, pincel, comprimento)  
        move(pincel, DIREITA, comprimento)  
        b(k-1, pincel, comprimento)  
        move(pincel, BAIXO, comprimento)  
        a(k-1, pincel, comprimento)
```

C_k

```
def c(k, pincel, comprimento):  
    '''(int, Turtle, int) -> None  
    Desenha a curva C_k a partir da  
posição  
do pincel.  
'''  
    if k > 0:  
        b(k-1, pincel, comprimento)  
        move(pincel, DIREITA, comprimento)  
        c(k-1, pincel, comprimento)  
        move(pincel, CIMA, comprimento)  
        c(k-1, pincel, comprimento)  
        move(pincel, ESQUERDA, comprimento)  
        d(k-1, pincel, comprimento)
```

D_k

```
def d(k, pincel, comprimento):  
    '''(int, Turtle, int) -> None  
    Desenha a curva  $D_k$  a partir da  
    posição  
    do pincel.  
    '''  
    if k > 0:  
        a(k-1, pincel, comprimento)  
        move(pincel, BAIXO, comprimento)  
        d(k-1, pincel, comprimento)  
        move(pincel, ESQUERDA, comprimento)  
        d(k-1, pincel, comprimento)  
        move(pincel, CIMA, comprimento)  
        c(k-1, pincel, comprimento)
```