Fundamentos de Programação

Aula Extra - Recursão

Arnaldo Barreto Vila Nova

Recursão

- Definição
- Exemplos
- Estrutura
- Prós e Contras

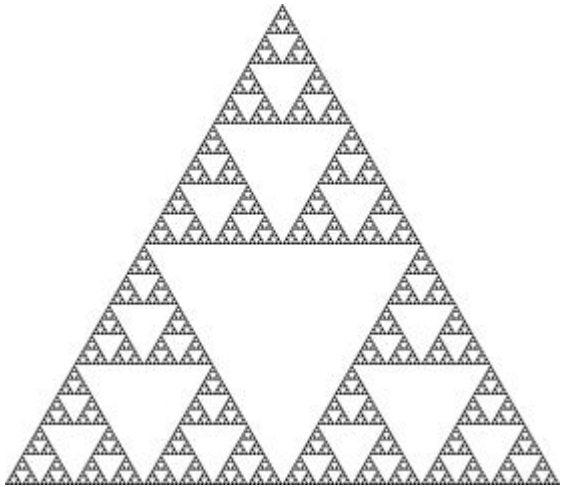
Recursão - Definição

- Uma função é dita recursiva quando dentro de seu código existe uma chamada para si mesma

Recursão - Definição

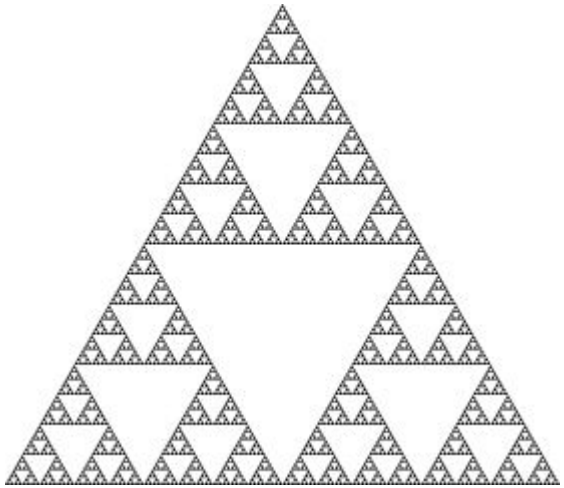
- Uma função é dita recursiva quando dentro de seu código existe uma chamada para si mesma
- A técnica de recursão visa quebrar um problema em partes menores dele mesmo (recorrências)

Recursão - Exemplos



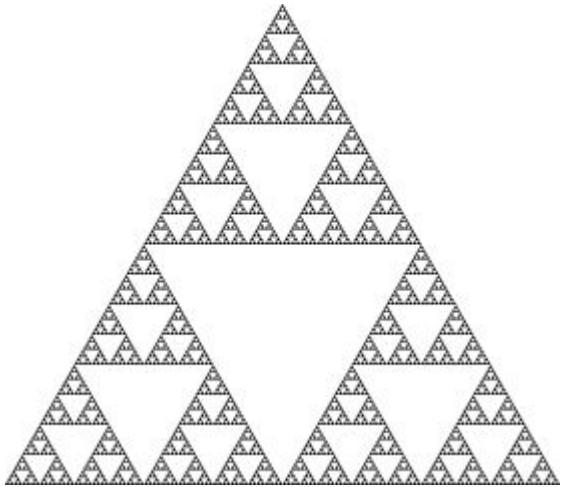
O triângulo de Sierpinsk é um exemplo clássico de recursão.

Recursão - Exemplos



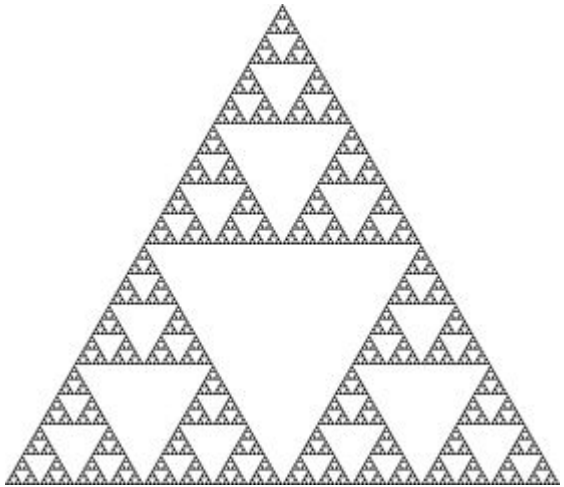
Começando de um triângulo inicial, cada lado é dividido na metade para formar 3 triângulos.

Recursão - Exemplos



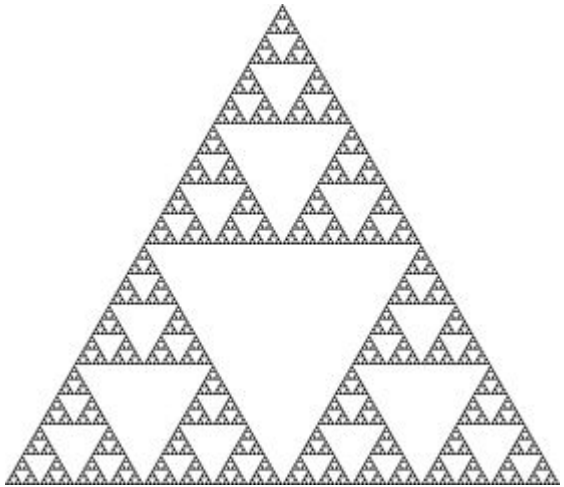
Depois, para cada triângulo é feita a mesma coisa, até atingir o número de subdivisões desejada

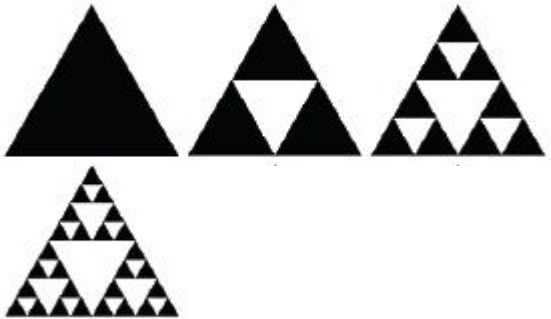
Recursão - Exemplos



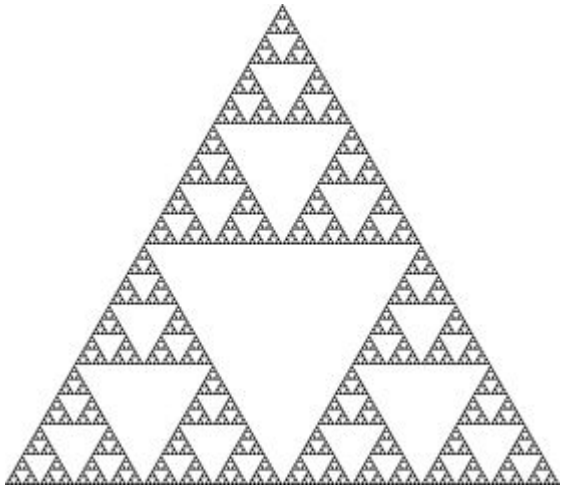
Depois, para cada triângulo é feita a mesma coisa, até atingir o número de subdivisões desejada

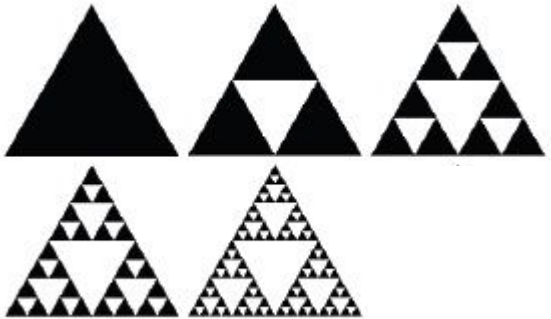
Recursão - Exemplos



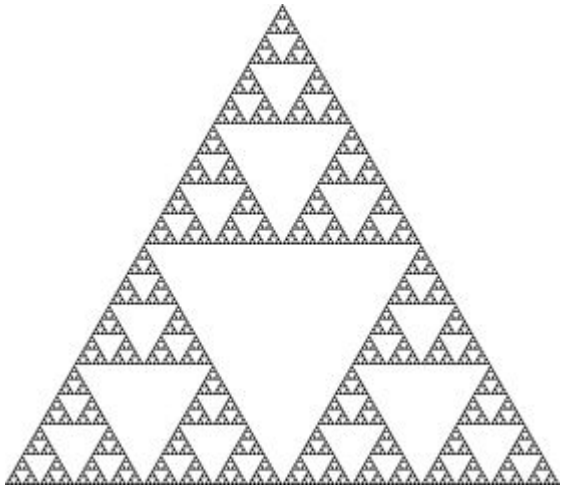


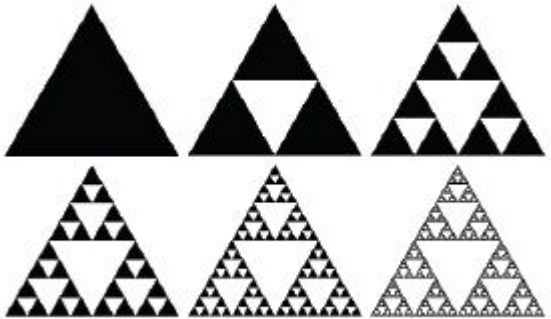
Recursão - Exemplos





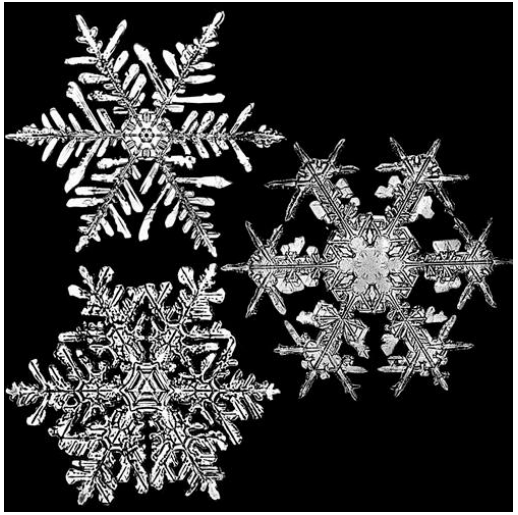
Recursão - Exemplos





Recursão - Exemplos

Na natureza também existem diversos exemplos de recursão, como a formação de flocos de neve, o formato de plantas, o crescimento de cascos dos caracóis...



Recursão - Exemplos

- Um exemplo numérico simples: Fatorial

```
n!   = 1 * 2 * 3 * … * n-1 * n
```

Recursão - Exemplos

- Um exemplo numérico simples: Fatorial

```
  n!   = 1 * 2 * 3 * … * n-1 * n
(n-1)! = 1 * 2 * 3 * … * n-1
```

Recursão - Exemplos

- Um exemplo numérico simples: Fatorial

```
  n!   = 1 * 2 * 3 * … * n-1 * n
(n-1)! = 1 * 2 * 3 * … * n-1
```

Esse cálculo direto é a “forma iterativa” de resolução

Recursão - Exemplos

- Um exemplo numérico simples: Fatorial

```
  n!   = 1 * 2 * 3 * … * n-1 * n
(n-1)! = 1 * 2 * 3 * … * n-1

  n!   = n * (n-1)!
```

Mas podemos pensar dessa forma: para calculcar n!, preciso calcular antes o (n-1)!

Recursão - Exemplos

- Um exemplo numérico simples: Fatorial

```
  n!   = 1 * 2 * 3 * … * n-1 * n
(n-1)! = 1 * 2 * 3 * … * n-1

  n!   = n * (n-1)!
(n-1)! = (n-1) * (n-2)!
```

Para calculcar (n-1)!, preciso calcular antes o (n-2)!... e assim por diante até chegarmos no 1! ou 0! que, por definição, tem resultado 1.

Recursão - Exemplos

- Um exemplo numérico simples: Fatorial

```
  n!   = 1 * 2 * 3 * … * n-1 * n
(n-1)! = 1 * 2 * 3 * … * n-1

  n!   = n * (n-1)!
(n-1)! = (n-1) * (n-2)!
```

Essa segunda forma de pensar é o que chamamos de recursão ou recursividade. Para resolver um problema, resolvemos primeiro uma parte menor dele.

Recursão - Exemplos

- Fatorial (iterativo)

```
///Em código C teríamos

int fatorial = 1;
for (i = 1; i<=n; i++)
{
    fatorial = fatorial *i;
}
printf(“%d”, fatorial);
```

Recursão - Exemplos

- Fatorial (recursivo)

```
  n!   = n * (n-1)!
(n-1)! = (n-1) * (n-2)!

int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

Recursão - Exemplos

- Fatorial (recursivo)

```
  n!   = n * (n-1)!
(n-1)! = (n-1) * (n-2)!
```

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

Podemos ver que dentro da função chamamos ela mesma

Recursão - Exemplos

- Fatorial (recursivo)

```
  n!   = n * (n-1)!
(n-1)! = (n-1) * (n-2)!
```

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

Podemos ver que dentro da função chamamos ela mesma, para uma parte menor

Recursão - Exemplos

- Fatorial (recursivo)

```
  n!   = n * (n-1)!
(n-1)! = (n-1) * (n-2)!
```

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

→ essa é a condição de parada

Recursão - Exemplos

executando fatorial(3)

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

Recursão - Exemplos

executando fatorial(3)
→ 3 == 0

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
      →   3 == 0
          return 3 * fatorial(2);
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);
executando fatorial(2)
     →   2 == 0
         return 2 * fatorial(1);
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →  3 == 0
        return 3 * fatorial(2);
executando fatorial(2)
     →  2 == 0
        return 2 * fatorial(1);
executando fatorial(1)
     →  1 == 0
        return 1 * fatorial(0);
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);
executando fatorial(2)
     →   2 == 0
         return 2 * fatorial(1);
executando fatorial(1)
     →   1 == 0
         return 1 * fatorial(0);
executando fatorial(0)
     →   0 == 0
         return 1;
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);
executando fatorial(2)
     →   2 == 0
         return 2 * fatorial(1);
executando fatorial(1)
     →   1 == 0
         return 1 * fatorial(0);
executando fatorial(0) (1)
     →   0 == 0
         return 1;
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);
executando fatorial(2)
     →   2 == 0
         return 2 * fatorial(1);
executando fatorial(1)
     →   1 == 0
         return 1 * fatorial(0);(1)
executando fatorial(0) (1)
     →   0 == 0
         return 1;
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);
executando fatorial(2)
     →   2 == 0
         return 2 * fatorial(1);
executando fatorial(1) (1)
     →   1 == 0
         return 1 * fatorial(0);(1)
executando fatorial(0) (1)
     →   0 == 0
         return 1;
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);
executando fatorial(2)
     →   2 == 0
         return 2 * fatorial(1);(2)
executando fatorial(1) (1)
     →   1 == 0
         return 1 * fatorial(0);(1)
executando fatorial(0) (1)
     →   0 == 0
         return 1;
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);
executando fatorial(2) (2)
     →   2 == 0
         return 2 * fatorial(1);(2)
executando fatorial(1) (1)
     →   1 == 0
         return 1 * fatorial(0);(1)
executando fatorial(0) (1)
     →   0 == 0
         return 1;
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
executando fatorial(3)
     →   3 == 0
         return 3 * fatorial(2);(6)
executando fatorial(2) (2)
     →   2 == 0
         return 2 * fatorial(1);(2)
executando fatorial(1) (1)
     →   1 == 0
         return 1 * fatorial(0);(1)
executando fatorial(0) (1)
     →   0 == 0
         return 1;
```

Recursão - Exemplos

```
int fatorial(int n)
{
    if (n==0)
        return 1;
    return n*fatorial(n-1);
}
```

```
void main()
{
    int n;
    printf("Digite um inteiro:");
    scanf("%d",&n);
    if (n<0)
        printf("Erro! n negativo");
    else
        printf("n! = %d", fatorial(n));
}
```

Recursão - Estrutura

- Cuidado com possível loop infinito

Recursão - Estrutura

- Cuidado com possível loop infinito
- 2 passos:
    - Definir condição de parada (solução básica)

Recursão - Estrutura

- Cuidado com possível loop infinito
- 2 passos:
  - Definir condição de parada (solução básica)
  - Garantir que a cada chamada da função ela se aproxima da condição de parada, chegando nela eventualmente

Recursão - Estrutura

```
tipo funcao(parametros)
{
    //teste de parada
    //processamento (se houver)
    //chamada recursiva
}
```

Recursão - Estrutura

- Exemplo: Somar os n primeiros inteiros positivos

```
int soma(int n)
{
    if (n==1)
        return 1;
    return n+soma(n-1);
}
```

Recursão - Prós e Contras

✓ Mais elegante e mais claro

Recursão - Prós e Contras

✓ Mais elegante e mais claro

✘ Consome mais memória e é mais lento (no geral)

Recursão - Prós e Contras

- ✓ Alguns problemas são tão mais fáceis de resolver de modo recursivo que acaba valendo a pena os contras
- ✓ Exemplo: O famoso problema da **Torre de Hanoi** é praticamente impossível de resolver iterativamente, mas é resolvido recursivamente com 5 ou 6 linhas de código.

Exercícios

- Leia dois inteiros (x e y) e calcule a soma do menor até o maior usando recursão
- Leia dois inteiros (x e y) e calcule x^y de forma recursiva
- Use recursão para calcular um termo n da Sequência de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 ...
  Ex.: fibonacci(9) retorna 21
  fibonacci(7) retorna 8