Recursividade

Profª Simone de Oliveira Santos

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Sumário

Recursividade

É um princípio que permite que problemas sejam definidos em função de instâncias menores do mesmo problema.

A recursão está presente em várias situações e, comumente, é observada em "obras" da natureza.

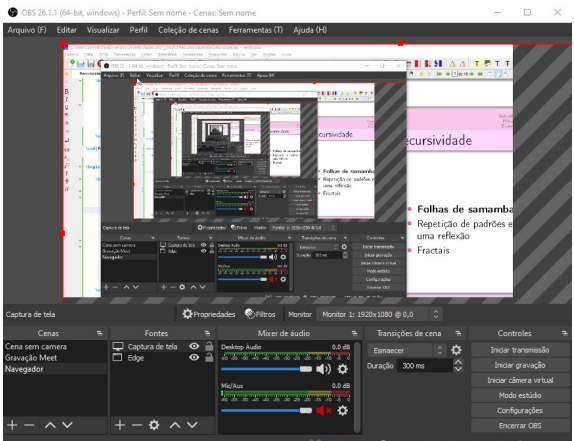
- Folhas de samambaia
- Repetição de padrões em uma reflexão
- Fractais

Recursividade

- **Folhas de samambaia**
- Repetição de padrões em uma reflexão
- Fractais

Recursividade

- Folhas de samambaia
- **Repetição de padrões em uma reflexão**
- Fractais



Recursividade

- Folhas de samambaia
- Repetição de padrões em uma reflexão
- **Fractais**

Recursividade

A recursão está presente também na matemática, vejamos o problema do fatorial.

Definição iterativa do problema:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

Definição recursiva do problema:

$$n! = \begin{cases} 1 & se\ n = 0 \\ n \times (n-1)! & se\ n > 0 \end{cases}$$

Recursividade

- Na computação é possível usar recursão por intermédio de funções recursivas.
- Uma função recursiva é uma função que chama a ela mesmo.
- A recursividade é uma forma interessante de resolver problemas por meio da divisão dos problemas em problemas menores de mesma natureza.
- Se a natureza dos subproblemas é a mesma do problema, o mesmo método usado para reduzir o problema pode ser usado para reduzir os subproblemas e assim por diante.

Recursividade

Se a função vai chamar a si mesmo, quando parar?

Recursividade

Se a função vai chamar a si mesmo, quando parar?

Quando alcançar um caso trivial em que a solução é conhecida.

Recursividade

Se a função vai chamar a si mesmo, quando parar?

Quando alcançar um caso trivial em que a solução é conhecida.

$$n! = \begin{cases} 1 & se\ n = 0 \\ n \times (n-1)! & se\ n > 0 \end{cases}$$

Estratégia recursiva

Uma estratégia recursiva para a solução de um problema consiste em duas partes:

1. O caso base, cuja solução é conhecida;
2. Um método geral que reduz o problema a um ou mais problemas menores (subproblemas) de mesma natureza.

Fatorial de n

Definição recursiva do problema:

$$n! = \begin{cases} 1 & se\ n = 0 \\ n \times (n-1)! & se\ n > 0 \end{cases}$$

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

Pilha de execução

- Para cada chamada de uma função, recursiva ou não, os parâmetros e as variáveis locais são empilhados na pilha de execução.
- Internamente, quando qualquer chamada de função é feita dentro de um programa, é criado um Registro de Ativação na Pilha de Execução do programa

Pilha de execução

- O registro de ativação armazena os parâmetros e variáveis locais da função bem como o "ponto de retorno" no programa ou subprograma que chamou essa função
- Ao final da execução dessa função, o registro é desempilhado e a execução volta ao subprograma que chamou a função

Recursividade

Vantagens da recursão

Torna a escrita do código mais simples e elegante, tornando-o fácil de entender e de manter.

Recursividade

Desvantagens da recursão

- Quando o loop recursivo é muito grande é consumida muita memória nas chamadas a diversos níveis de recursão, pois cada chamada recursiva aloca memória para os parâmetros, variáveis locais e de controle.
- Em muitos casos uma solução iterativa gasta menos memória, e torna-se mais eficiente em termos de performance do que usar recursão.

Prática

1. Faça uma função (de forma iterativa) que mostre a soma dos n primeiros números inteiros.
2. Faça a versão recursiva da função anterior.
3. Faça uma função (de forma iterativa) para calcular a base x elevado a um expoente y.
4. Faça a versão recursiva da função anterior.