Algoritmos

+

Estruturas de Dados

=

Programas

Material da Prof. Ana Eliza

Recursividade

Definição:

Um algoritmo é dito recursivo quando chama a si mesmo (**recursão direta**) ou chama uma seqüência de outros algoritmos, e um deles chama novamente o primeiro (**recursão indireta**).

$$\mathbf{R \equiv [C,R]}$$

ou

$$\mathbf{R_1 \equiv [C_1,R_2],\ R_2 \equiv [C_2,R_3],\ ...,\ R_n \equiv [C_n,R_1]}$$

Recursividade

- **Exemplo:**

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

Recursividade

Definição:

- Uma definição resursiva consiste de duas partes:
  - **Solução Trivial ou Base da Recursão**: dada por definição. Não necessita da recursão para ser obtida.

```
if (n == 0 || n == 1) return (1);
```

  - **Solução Geral**: é a solução genérica aplicada a uma parte menor do problema.

```
else return (n * fatorial (n-1));
```

Recursividade

Recursão na Cauda:

- Uma rotina apresenta recursão na cauda se a chamada recursiva está no final de seu código, tendo como única função criar um “looping” que será repetido até que a condição de parada (solução trivial) seja satisfeita.
- Exemplo: Função fatorial recursiva

Recursividade

Recursão na Cauda:

- A recursão na cauda pode ser eliminada se empregarmos, em seu lugar, uma estrutura de repetição que esteja condicionada à expressão de teste usada na versão recursiva.

Recursividade

Exemplo:

```
int fatorial_iterativo (int n)
{     int fat;
      fat = 1;
      while (n > 0)
      {
         fat = fat * n;
         n = n - 1;
      }
      return (fat);
}
```

Recursividade

- Desenvolvimento de Algoritmos Recursivos
  - Encontre a solução geral (solução genericamente aplicável);
  - Encontre a solução trivial (regra de parada);
  - Monte seu algoritmo combinando a solução trivial e a solução geral utilizando de uma estrutura condicional (*if*);
  - Certifique-se que a recursividade sempre terminará.

Recursividade

- Vantagens e Desvantagens
  - Vantagem
    - Torna o algoritmo elegante: claro, simples e conciso.
  - Desvantagem
    - Mais gasto de tempo e espaço
- OBS:

  Um algoritmo iterativo, na maioria das vezes, é mais eficiente do que seu similar recursivo.

Recursividade

Exemplo:

```
void hanoi (int n,char a, char b, char c)
{
   if (n==1)
      printf("Mova disco 1 de %c para %c",a,c);
   else
   {
     hanoi(n-1,a,c,b);
     printf("Mova disco %i de %c para %c",n,a,c);
     hanoi(n-1,b,a,c);
   }
}
```

OBS: Não é recursão na cauda.