

Funções Recursivas

Funções Recursivas

- É uma função que se refere a si própria.
- A ideia consiste em utilizar a própria função que estamos a definir na sua definição.

Funções Recursivas



Funções Recursivas

- Em todas as funções recursivas existe:
 - Um passo básico (ou mais) cujo resultado é imediatamente conhecido.
 - Um passo recursivo em que se tenta resolver um sub-problema do problema maior.

Funções Recursivas

- Se analisarmos a função fatorial...
 - o caso básico é o teste de igualdade a zero, cujo resultado imediato é 1;
 - e o passo recursivo é $n * \text{fatorial}(n-1)$

Funções Recursivas

fatorial (3)

= 3 * fatorial (2)

= 3 * 2 * fatorial (1)

= 3 * 2 * 1 * fatorial (0)

= 3 * 2 * 1 * 1

= 6

Funções Recursivas

- A execução de uma função recursiva consiste em ir resolvendo subproblemas sucessivamente mais simples até se atingir o caso mais simples de todos, cujo resultado é imediato.
- Os erros mais comuns associados às funções recursivas são, naturalmente:
 - Não detectar os casos simples
 - A recursão não diminuir a complexidade do problema.
- No caso de erro em função recursivas, o mais usual é a recursão nunca parar.

Exercício (1)

- Considere uma versão extremamente primitiva da linguagem C, em que as únicas funções numéricas existentes são **zero** e duas funções que **incrementa** e **decrementa** o seu argumento em uma unidade. Isto implica que as operações **>**, **<**, **=** e similares não podem ser utilizadas. Nesta linguagem, defina a função **menor**, que recebe dois número inteiros positivos e determina se o primeiro argumento é numericamente inferior ao segundo.

Resposta (1)

```
zero (int x)
{
    return (x==0);
}
```

```
incrementa (int x)
{
    return (x+1);
}
```

```
decrementa (int x)
{
    return (x-1);
}
```

Resposta (1)

```
menor (int x, int y)
{
    if (zero(y)) {return 0;}
    else {
        if (zero(x)) {return 1;}
        else {
            return menor(decrementa(x), decrementa(y));
        }
    }
}
```

Resposta (1)

```
main ()
{
    int n1, n2;
    scanf ("%d %d", &n1, &n2);
    if (menor(n1,n2)) {
        printf ("%d eh menor que %d\n", n1, n2);
    }
    else {
        printf ("%d NAO eh menor que %d\n", n1, n2);
    }
}
```

Exercício (2)

- Considere que nessa versão do C não exista o operador de igualdade. Defina a função `igual` que recebe dois número inteiros positivos e determina se estes são iguais ou não.

Exercício (3)

- Até ao momento, essa linguagem apenas trabalha com números inteiros positivos. Admitindo que as operações incrementa, decrementa e zero também funcionam com números negativos, defina a função negativo que recebe um número inteiro positivo e retorna o seu **simétrico**. Assim, pretendemos obter: **simétrico(3) => -3**

Exercício (4)

- É possível definir a soma de dois números inteiros positivos nessa linguagem apenas recorrendo às funções incrementa e decrementa, que somam e subtraem uma unidade, respectivamente. Defina a operação soma.

Exercício (5)

- Multiplicação de dois números naturais, através de somas sucessivas (Ex.: $6 * 4 = 4 + 4 + 4 + 4 + 4 + 4$).

Exercício (6)

- Cálculo de $1 + 1/2 + 1/3 + 1/4 + \dots + 1/N$.

Exercício (7)

- Inversão de uma string.

Exercício (8)

- Gerar a sequência dada por:
 - $F(1) = 1$
 - $F(2) = 2$
 - $F(n) = 2 * F(n - 1) + 3 * F(n - 2)$

Exercício (9)

- Dado valores para m e n , gerar a sequência de Ackerman:
 - $A(m; n) = n + 1$, se $m = 0$
 - $A(m; n) = A(m - 1; 1)$, se $m \neq 0$ e $n = 0$
 - $A(m; n) = A(m - 1, A(m, n - 1))$, se $m \neq 0$ e $n \neq 0$.