LABORATÓRIO DE PROGRAMAÇÃO AVANÇADA
DÉCIMO TRABALHO PRÁTICO
- - FUNÇÕES RECURSIVAS - -

Este trabalho visa o uso de Funções Recursivas. Só relembrando que uma função recursiva é uma função que se refere a si própria, cuja ideia consiste em utilizar a função que está sendo definida na sua própria definição. Um exemplo clássico é o problema do fatorial, que pode ser definido recursivamente da seguinte forma:

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

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

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 * fatorial (n-1)`.

A execução de uma função recursiva consiste em ir resolvendo sub-problemas 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.

Exemplo

Considere uma versão extremamente primitiva da linguagem C, em que as únicas funções numéricas existentes são **`zero`** e duas funções **`incr`** e **`decr`**, que incrementa/decrementa, respectivamente, seu argumento em uma unidade. Isto implica que as operações >, <, = e similares não podem ser utilizadas. Portanto, assuma que as funções **`zero`**, **`incr`** e **`decr`** já estão definidas, da seguinte forma:

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

int decr (int x)
{
  return (--x);
}

int incr (int x)
{
  return (++x);
}
```

Exemplo – Parte 1:

Nesta linguagem, defina a função **`menor`**, que recebe dois número inteiros positivos e determina se o primeiro argumento é numericamente inferior ao segundo.

Resposta:

```
int menor (int x, int y)
{
  if (zero(y)) {
    return 0;     // retorne FALSO
  }
  else {
    if (zero(x)) {
      return 1;  // retorne VERDADEIRO
    }
    else {
      return menor(decr(x), decr(y));
    }
  }
}
```

Note que, no problema de encontrar o menor, há DOIS casos básicos, isto é, quando **`y=0`**, ou **`x = 1`**. Estamos considerando que 0 significa **FALSO** e 1 significa **VERDADEIRO**.

Exemplo – Parte 2:

Faça uma função main que use a função **`menor`**.

Resposta:

```
void main()
{
  int n1, n2;
  printf ("Entre com 2 numeros: ");
  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);
  }
}
```

EXERCÍCIOS:

Em todos os exercícios resolva o problema, inclua a função main e mais as três funções **`zero`**, **`incr`** e **`decr`**.

1) Considere que nessa versão do C não exista o operador de igualdade. Defina a função recursiva igual, que recebe dois número inteiros positivos e retorne 1, se estes são iguais, ou 0 se não.
2) Até ao momento, essa linguagem apenas trabalha com números inteiros positivos. Admitindo

que as operações **`incr`**, **`decr`** e **`zero`** também funcionam com números negativos, defina a função recursiva **`simétrico`** que recebe um número inteiro positivo e retorna o seu simétrico. Por exemplo, a chamada de **`simetrico(3)`** vai retornar o valor **`-3`**; a chamada de **`simetrico(100)`** vai retornar o valor **`-100`**.

3) Por incrível que pareca é perfeitamente possível definir a soma de dois números inteiros positivos nessa linguagem apenas recorrendo às funções incrementa e decrementa. Defina a função recursiva **`soma`**.
4) Faça uma função recursiva que multiplique dois números inteiros positivos através de somas sucessivas. Por exemplo: 6 * 4 = 4 + 4 + 4 + 4+ 4 + 4).
5) Faça uma função recursiva que receba o valor de N e calcule o somatório: 1 + 1/2 + 1/3 + 1/4 + ... + 1/N.
6) Faça uma função recursiva que gere a sequência dada por:
   F(1) = 1
   F(2) = 2
   F(n) = 2 * F(n - 1) + 3 * F(n - 2)

- - -

Este trabalho deve ser entregue no dia **11/07/2014**. Entretanto, vou liberar a entrega até a data máxima de **14/07/2014** (segunda) até meia-noite.

IMPORTANTE! Após esta data, o trabalho não será mais aceito.

O trabalho deve ser possível ser compilado pelo **GCC** e executado no **Linux**.

Envie para o professor (**xbarretox@gmail.com**) e o monitor (**marrco.santos@gmail.com**).

Coloque no assunto: [**LPAV-TP10**].