Exercícios : Programação III

Paul Crocker 23 de Março de 2001

Recursividade

Nos seguintes exercícios escreva a função e um programa main (um "*driver"*) para a testar :

1) A função n-factorial : **n !** = { **1** if **n**=0 ; else **n(n-1)!** }

2) A função exponencial : $x^n = \begin{Bmatrix} 1 & \text{if n = 0} \\ x(x^{\lfloor n/2 \rfloor})^2 & \text{if n impar} \\ (x^{\lfloor n/2 \rfloor})^2 & \text{in n par} \end{Bmatrix} \quad n \in N^+,\ x \in \Re$

3) A função para implementar o algoritmo de Euclides que permite calcular o maior divisor comum de dois números.

O Algoritmo : d = mcd(x,y):

```
if       x > y then     d = mcd( x -y , y )
else if  y > x then     d = mcd( x , y-x)
else                    d = x
```

4) Uma função recursiva para imprimir uma cadeia de caracteres de elementos 0 .. n em ordem inverso. Por exemplo : Ler string "sempre" e passar para função cujo output seria: "erpmes".

O protótipo desta função é : void ex4(char s[]) ;

A ideia de recursividade é a seguinte :
se o comprimento do string, n, for maior que um
então chame outra vez a função mas com o string reduzido em tamanho , quer dizer com o apontador do string apontando para a próxima caracter da cadeia →ex4(s+1).
A condição terminal é testar o comprimento do string (*strlen*) e se este for igual a 1, então paramos. Depois regressamos pelo *stack* das funções chamadas imprimindo o primeiro caracter da cadeia s[0].

Nota: a função standard *strlen* não conte o terminador do string '\0'

5) Uma função recursiva para converter um string de dígitos para o número inteiro correspondente.
Por exemplo para o string input "5274" a função devolverá o inteiro 5274.
Lembrem-se que (int)'5' - (int)'0' é o número 5 !
A ídeia de recursividade é a seguinte :
5274 = 527(10)+4 : 527 = (52)10+7 : 52 = (5)10+2 : 5 = 5 stop

6*) Escreva um programa que contenha duas funções. Ambos devem devolver o n-ésimo número de Fibonnaci. Uma função utilizará iteração: int fibIt (int n); e a outra recursividade:
int fibRec(int n); .

a) Usando as funções e tipos predefinidos na biblioteca standard <time.h> meça o tempo que cada função leva para calcular o numero de Fibonnaci para n=5,10,15, ….. ,40.
b) Qual é o mínimo número *n* que provoca um "*stack overflow error"* na função recursiva ?

Notas:

I.) trabalhe com inteiros **long int** e não com inteiros mais simples do tipo **int** !
II.) Meça o tempo em segundos do tempo normal e em tempo do processador (ver informação sobre os tipos : **time_t** e **clock_t)**
III.) Guarde, as respostas das partes (a) e (b) num ficheiro.
IV.) Se tiver falta de recursos (informátios ou de paciencia !) para obter a resposta à parte (b) então volte a utilizar o tipo **int** em vez do tipo **long int** para ver se isto ajuda.

Formal Grammers

A função seguinte reconhece se um string está dentro da linguagem {números binários} ou não.

```
#include <string.h>
typedef enum { FALSE , TRUE } boolean;
/*
pre    : S cadeia de caracteres
post :  TRUE ou FALSE conforme S é um membro de linguagem com a
          gramática  S ->0, S ->1, S->0S,  S->1S  (os números binários)
*/
boolean dentroLing ( char s[] )            /* ou o equivalente char *s */
{
    boolean reconhece;
    if ( strlen(s) == 1 )
            reconhece = ( s[0] == '0' || s[0] == '1') ? TRUE : FALSE;
    else if  ( s[0] == '0' || s[0] == '1')
            reconhece = dentroLing( s +  1 ) ;
    else
            reconhece = FALSE;
    return reconhece;
}
```

1) Implemente esta função e escreva a parte main() para testar a sua implementação.

Para cada um dos seguintes exercícios 2-5 faça o seguinte:

a) Descreva a linguagem que as produções geram no vocabulário {0, 1, S }
b) Mostre como gerir o "string" **b**
c) Escreva uma função recursiva booleana que devolva TRUE (1) se um string está dentro da linguagem e FALSE (0) se não. Teste a sua função !

2) S -> 0S
S-> 0
b. 000

3) S->01S
S->01
b. 0101

4) S->0S0
S->1
b.0001000

5) S->0S0
S->1S1
S->1 **b.**0011100

6) Usando a descrição BNF dum identifacador de C escreva uma função recursiva booleana que devolva TRUE se um string é um identificador valido em C ou não. Teste a sua função !

dica : utilize as seguintes funções: *int isalpha(int c) e int isalnum(int c) etc.; (#include <ctype.h>*) e cuidado com o facto do primeiro caracter não poder ser um número ! (uma solução seria escrever duas funções, apenas uma recursiva)

7) Escreva uma função recursiva booleana para testar se um string está dentro da linguagem das "cadeias de bits que não contêm 11. Teste a sua função ! Mostre o "parse tree" para a palavra 0101

8) Desenvolva a gramática para a linguagem, que contêm todos os strings de bits que não contêm 00. Mostre o "parse tree" para a palavra 01101.A seguir escreva uma função recursiva booleana para testar se um string está dentro desta linguagem ou não.Teste a sua função !

9*) Desenvolva a gramática para a linguagem, que contêm todos os strings de bits que não contêm 10. Mostre o "parse tree" para a palavra 00111. A seguir escreva uma função recursiva booleana para testar se um string está dentro desta linguagem ou não.Teste a sua função !

10*) Desenvolva a gramática para a linguagem L, que contêm todos os strings de caracteres que são capicuas. Mostre o "parse tree" para a palavra *abccba*. A seguir escreva uma função recursiva booleana para testar se um string está dentro desta linguagem ou não.Teste a sua função !

11) Seja o conjunto sintatico <S> definido na descrição BNF por <S> :: = r L | r <S> L | <S><S>
Quais dos seguintes fazem parte deste conjunto? a) rr LL r L b) r LL rrr L L c) rrLrLrLLrLrrL