Cálculo de Programas

2.° ano das Licenciaturas em
Engenharia Informática (LEI) e Ciências da Computação (LCC)
da Universidade do Minho

2009/10 - Ficha nr.° 7

1. Considere a definição habitual da função de Fibonacci:

$$
\begin{aligned}
&\mathit{fib}\ 0 = 1\\
&\mathit{fib}\ 1 = 1\\
&\mathit{fib}\ (n+2) = \mathit{fib}\ (n+1) + \mathit{fib}\ n
\end{aligned}
$$

   (a) Mostre que essa função é a mesma que a que a seguir se define (em $\mathbb{N}_0$)

$$\mathit{fib} \cdot \mathbf{in} = [\underline{1}\ , f] \tag{1}$$

   com recurso à função auxiliar

$$f \cdot \mathbf{in} = [\underline{1}\ , \mathsf{add} \cdot \langle f, \mathit{fib} \rangle] \tag{2}$$

   para $\mathbf{in} = [\underline{0}\ , \mathsf{succ}]$ e $\overline{\mathsf{add}} = (+)$.

   (b) Recorra à lei da recursividade múltipla na resolução em ordem a x da equação

$$\langle f, \mathit{fib} \rangle \cdot \mathbf{in} = x \cdot (id + \langle f, \mathit{fib} \rangle) \tag{3}$$

   (c) Derive a partir desse cálculo a seguinte implementação linear de fib:

$$
\begin{aligned}
&\mathit{fib}\ n = b\\
&\quad \mathbf{where}\ (a, b) = \mathit{for}\ \mathit{loop}\ (1, 1)\ n\\
&\quad\quad \mathit{loop}\ (a, b) = (a + b, a)
\end{aligned}
$$

2. Considere o seguinte fragmento de Haskell

$$
\begin{aligned}
&\mathit{par} = \pi_1 \cdot \mathit{parimpar}\\
&\mathit{impar} = \pi_2 \cdot \mathit{parimpar}\\
&\mathit{parimpar} = \mathit{for}\ \mathsf{swap}\ (\mathsf{True}, \mathsf{False})
\end{aligned}
$$

   onde as definições das funções que testam se um número inteiro não-negativo é par ou ímpar se baseiam num ciclo-for a duas variáveis. Usando a lei da recursividade múltipla mostre que impar e par são as funções

$$
\begin{aligned}
&\mathit{impar}\ 0 = \mathsf{False}\\
&\mathit{impar}\ (n+1) = \mathit{par}\ n
\end{aligned}
$$

   e

$$
\begin{aligned}
&\mathit{par}\ 0 = \mathsf{True}\\
&\mathit{par}\ (n+1) = \mathit{impar}\ n
\end{aligned}
$$

3. Quantos quadrados se podem desenhar numa folha de papel quadriculado com $n \times n$ quadrículas? A resposta é dada pela função $nq\ n = \sum_{i=1,n} i^2$ isto é, em Haskell,

$$nq\ 0 = 0$$
$$nq\ (n+1) = (n+1) \uparrow 2 + nq\ n$$

Será possível escrever nq sob a forma de um ciclo-for?

A resposta a esta questão é afirmativa usando a lei de recursividade múltipla estendida a três funções (ver ficha anterior). Começa-se por definir a função $bnm\ \ n$ = $(n+1) \uparrow 2$ e fazer a respectiva substituição em nq:

$$nq\ 0 = 0$$
$$nq\ (n+1) = bnm\ n + nq\ n$$

Logo nq e sq ficam mutuamente recursivas. De seguida expande-se bnm nas suas duas cláusulas:

$$bnm\ 0 = 1$$
$$bnm\ (n+1) = 2 * n + 3 + bnm\ n$$

A ocorrência do termo $2 * n + 3$ em $bnm\ \ (n+1)$ obriga-nos a repetir o processo acima até que a referida lei possa ser aplicada.

Faça-o, definindo $lin\ n = 2*n+3$ e continuando o raciocínio, mostrando finalmente que, aplicando recursividade múltipla ao sistema definido pelas três funções nq, bnm e lin, se obtém a versão linear de nq que se segue:

$$
\begin{array}{l}
nq\ n = \mathbf{let}\ (a,b,c) = aux\ n\ \mathbf{in}\ a \\
\quad \mathbf{where} \\
\qquad aux\ 0 = (0,1,3) \\
\qquad aux\ (n+1) = \mathbf{let}\ (a,b,c) = aux\ n\ \mathbf{in}\ (a+b, b+c, 2+c)
\end{array}
$$

Já agora, aqui está o programa em C que pode derivar do código Haskell que acima se calculou:

```
int ns(int x)
{
int n=0;int s=1;int l=3;int i;
for (i=1;i<x+1;i++) {n+=s;s+=l;l+=2;}
return n;
};
```