Exame de Linguagens de Programação 2

5 de Julho de 2002 (Duração: 2H30M)

I

1. [2] Indique o valor dos programas Oby seguintes:

   (a)
   ```
   let comp = fun f,g -> fun x -> f(g(x))
    in let f = fun x -> fun y -> x+y
        in comp(f(1),fun x -> x*2)(3)
   ```

   (b)
   ```
   letrec f = fun x -> f(x-1)
       in let foo = fun x -> fun y -> x
           in foo(fun x -> x+1)(f(2))(3)
   ```

2. [2] Considere os programas acima:

   (a) Explique detalhadamente o funcionamento do programa 1(a), apresentando o valor do ambiente no momento em que se inicia a avaliação da expressão `x+y`.

   (b) Indique qual o valor do programa 1(b) se em vez da regra de avaliação de parâmetros *call-by-value* usual for adoptada a regra *call-by-name*.

II

Considere a linguagem Oby estudada nas aulas. A linguagem Oby é muito poderosa apesar de muito simples, pois permite "programar" praticamente todas as construções das linguagens de programação modernas. Em particular, vimos que é simples definir classes de objectos programando-as como funções geradoras de objectos. Por exemplo uma classe de contadores pode ser definida em Oby como

```
let counterclass = ( fun initial_value ->
     let c = new initial_value in
       [
         inc = fun -> c := !c+1;
         dec = fun -> c := !c-1;
         get = fun -> !c
       ]
     ) in ...
```

1. [2] Numa tradução deste tipo, o nome `c` representa um atributo privado de cada objecto, só acessível através dos métodos do mesmo. Explique porque `c` só é acessível dentro do corpo dos métodos.

2. [1] Indique detalhadamente qual o valor (estrutura de dados) resultante de avaliar a expressão `counterclass(2)` no contexto da declaração acima apresentada, indicando correctamente os ambientes associados a cada fecho (a cada *closure*).

3. [2] Nas linguagens de programação baseadas em classes, é possível definir nas classes não só variáveis locais a cada objecto, mas também variáveis (ou atributos de classe). Estas variáveis são usualmente declaradas como `static` (por exemplo em C++ ou Java).

   Indique como poderia representar em Oby classes com variáveis de classe privadas, isto é, apenas acessíveis aos métodos dos objectos da classe.

   Exemplifique definindo uma classe contador com a funcionalidade da apresentada acima, mas com uma variável de classe `glo` e os métodos adicionais `inc_all` e `dec_all` e `get_all` que a permitem incrementar, decrementar e consultar *a partir de qualquer objecto da classe*.

III

A maior parte das linguagens de programação permitem definir valores obtidos não apenas por agregação de um tuplo de valores mas também por selecção de um valor de entre um conjunto de alternativas. Por exemplo, temos os tipos enumerados em C++, os tipos soma da linguagem ML, os registos com variante da linguagem Pascal, etc. Este exercício visa a introdução de tipos soma na linguagem Oby, obtendo a linguagem ObyS.

Uma *alternativa* é uma expressão da forma

$$id\{E_1, E_2, \ldots, E_n\}$$

onde id é uma etiqueta (um identificador) e os E_i são expressões quaisquer da linguagem ObyS. Eis alguns exemplos de alternativas:

```
nil{}
cons{1,nil{}}
node{x,1+2,y}
```

A única operação definida sobre alternativas é a construção de análise de casos `case`, que é uma expressão da forma:

$$\begin{array}{l}\texttt{case } E \texttt{ of}\\ \quad l_1\{x_1^1, \ldots, x_{n_1}^1\} = E_1\\ \quad \cdots\\ \quad l_k\{x_1^k, \ldots, x_{n_k}^k\} = E_k\\ \texttt{end}\end{array}$$

Os identificadores $x_1^i, \ldots, x_{n_i}^i$ são ocorrências ligantes de variáveis (cujo âmbito abrange cada expressão E_i), os l_i são etiquetas, e E e os E_i são expressões quaisquer.

A semântica da construção `case` é a seguinte: primeiro, é avaliada a expressão E, que deve produzir como resultado uma alternativa construída com base numa das etiquetas $l_1, \cdots, l_k$ (se assim não for, deverá ocorrer um erro de execução). Assim, podemos supor que o valor de E é a alternativa $l_j\{v_1, \cdots, v_m\}$, sendo l_j a etiqueta da alternativa associada à expressão E_j. Seguidamente, será avaliada a expressão E_j num contexto (ambiente) onde o valor de cada identificador x_i^j estará associado ao valor v_i existente na posição correspondente da alternativa. O resultado de tal avaliação será o valor devolvido finalmente pela expressão `case`.

Por exemplo, usando as alternativas `nil{}` e `cons{a,b}` como construtores podemos definir listas: `cons{1,cons{2,nil{}}}` representa a lista dos dois inteiros 1 e 2. E uma função que soma os elementos de listas representadas dessa forma por

```
letrec sum = fun l ->
  case l of
    nil{} = 0
    cons{x,y} = x+sum(y)
  end
```

1. [2] Indique detalhadamente como adicionar valores alternativa ao interpretador de Oby.
2. [3] Indique detalhadamente como adicionar a construção `case` ao interpretador de Oby.
3. [1] Usando alternativas para definir os construtores da sintaxe abstracta, defina em ObyS um interpretador para a linguagem de expressões aritméticas `CALC` estudada nas teóricas e definida pela sintaxe abstracta seguinte:

$$\begin{array}{lcl} num & : & integer \rightarrow \texttt{CALC}\\ add & : & \texttt{CALC} \times \texttt{CALC} \rightarrow \texttt{CALC}\\ sub & : & \texttt{CALC} \times \texttt{CALC} \rightarrow \texttt{CALC}\\ mul & : & \texttt{CALC} \times \texttt{CALC} \rightarrow \texttt{CALC}\\ div & : & \texttt{CALC} \times \texttt{CALC} \rightarrow \texttt{CALC}\end{array}$$