Paradigmas de Linguagens de Programação

Exame Escrito

Centro de Informática - UFPE

9 de junho de 2011

Questão 1 [2,0] Defina uma função ***insert*** que recebe dois argumentos (um inteiro, n, e uma lista ordenada de inteiros, L, e inclui n em L de forma que a lista resultante continua ordenada e preserva todos os elementos de L, além de n. Por exemplo, ***insert(5,[2,3,5,8]) = [2,3,5,5,8]***. Defina uma outra função, ***sort***, usando ***insert***, que recebe uma lista de inteiros e retorna uma lista ordenada, em ordem crescente, com exatamente os mesmos elementos da lista original. Por exemplo***, sort([9,5,7,3,4,6,8,9]) = [3,4,5,6,7,8,9,9]*.**

Questão 2 [2,0] a) Defina uma propriedade da função ***sort*** que garante que o tamanho (#) da lista ordenada é o mesmo da lista original.
b) prove esta propriedade, podendo assumir que #(x:xs) = 1 + #(xs) = #(insert(x,xs)).

Questão 3 [2,0] Explique a diferença entre *encapsulamento* e *information hiding*. Exemplifique importantes construções de linguagens modernas que surgiram graças a estes conceitos?

Questão 4 [1,0] Assinale com V (para verdadeiro) ou F (para falso):

( ) Estretégias de avaliação (innermost/outermost) podem impactar a terminação de programas
( ) LI1 possui a propriedade de ***transparência referencial***, já que não possui procedimentos
( ) Assumindo que a função ***add : int -> int -> int*** está na forma *currificada (curried)*, então ***(add(2))(3)*** é uma expressão válida
( ) Passagem de parâmetro por referência e valor-resultado possuem eficiências distintas, mas sempre produzem o mesmo resultado

Questão 5 [3,0] Estenda a linguagem LI1 com os seguintes comandos:

a) Comando de atribuição simultânea (***x1,...,xn = e1,..., en***) que avalia as expressões ***e1,...en*** e atribui os respectivos valores, simultaneamente, às variáveis ***x1,...,xn***. Duas restrições devem ser obedecidas: as listas de variáveis e de expressão devem ter o mesmo tamanho; a lista de variáveis não pode conter variáveis repetidas.
b) Comando condicional com guardas. No comando

```
if
   e1 -> c1
[] e2 -> c2
...
[] en -> cn
fi
```

cada ***ei*** é uma expressão booleana (condição) e ***ci*** um comando. A execução do comando ***if... fi*** escolhe aleatoriamente uma das condições verdadeiras e executa o comando correspondente; se todas as condições forem falsas, o comando bloqueia.

A implementação de ambos os comandos deve considerar os métodos de avaliação e checa tipo, bem como todas as classes auxiliares necessárias à completa implementação. Partcilularmente:

1) Defina a BNF para a linguagem redefinida, destacando apenas o que mudar.
2) Explique o que precisaria mudar no parser e se é necessária alguma mudança nos ambientes de compilação e execução.
3) Implemente as novas classes que se mostrem necessárias e modifique as existentes quando necessário.

Boa Sorte

Apêndice 1. BNF de LI1.

Programa ::= Comando

Comando ::= Atribuicao
| ComandoDeclaracao
| While
| IfThenElse
| IO
| Comando ";" Comando
| Skip

Skip ::=

Atribuicao ::= Id ":=" Expressao

Expressao ::= Valor | ExpUnaria | ExpBinaria | Id

Valor ::=ValorConcreto

ValorConcreto ::= ValorInteiro | ValorBooleano | ValorString

ExpUnaria ::= "-" Expressao | "not" Expressao | "length" Expressao

ExpBinaria ::= Expressao "+" Expressao
| Expressao "-" Expressao
| Expressao "and" Expressao
| Expressao "or" Expressao
| Expressao "==" Expressao
| Expressao "++" Expressao

ComandoDeclaracao :: = "{" Declaracao ";" Comando "}"

Declaracao ::= DeclaracaoVariavel | _DeclaracaoComposta

DeclaracaoVariavel ::= "var" Id "=" Expressao

DeclaracaoComposta ::= Declaracao "," Declaracao

While ::= "while" Expressao "do" Comando

IfThenElse ::= "if" Expressao "then" Comando "else" Comando

IO ::= "write" "(" Expressao ")" | "read" "(" Id ")"