Paradigmas de Linguagens de Programação
Exame Escrito

Centro de Informática - UFPE
14 de agosto de 2007

Questão 1 [1,5] Defina uma função *listaInfinita* que, quando aplicada a um inteiro, retorna uma lista infinita e consecutiva de inteiros, conforme exemplo a seguir:

listaInfinita (n) = [n, n+1, n+2, ...]

Defina uma outra função, como na aula de revisão ☺, que obtém os k primeiros elementos de uma lista, como em:

getElem (k, [e1, e2, ..., ek, ek+1, ...]) = , [e1, e2, ..., ek]

Questão 2 [1,5] A expressão *getElem(5,listaInfinita(50)) sempre termina com sucesso? Se sim, qual o resultado produzido? Em qualquer dos casos, justifique sua resposta.*

Questão 3 [2,0] Considere o seguinte programa imperativo:

proc *Inc* (**vr** *x*) = {*x := x + 1*};
{**var** *y =0*; **call** *Inc(y)*; ***write****(y)*}

onde **vr** representa o mecanismo de pssagem de parâmetro por ***valor-resultado***.

a) [0,5] Qual o resultado produzido pelo programa?
b) [1,0] Com base no princípio da correspondência, reescreva o programa acima eliminando a declaração do procedimento e substituindo a chamada pelo corpo do procedimento, de forma que o novo programa possua a mesma semântica (comportamento) do original.
c) Substituindo o mecanismo ***valor-resultado*** por ***referência*** alteraria o comportamento do programa? Explique se estes dois mecanismos sempre produzem o mesmo resultado.

Questão 4 [1,0] Boas práticas de projeto de linguagens de programação incluem uniformidade de tratamento dos construtores da linguagem. Explique e exemplifique um dos princípios de projeto uniforme apresentados no livro texto.

Questão 5 [4,0] Estenda a Linguagem Imperativa 2 (veja BNF em anexo) permitindo que procedimentos possam ter parâmetros do tipo valor-resultado.

a) Defina a BNF para a linguagem redefinida, destacando apenas o que mudar.
b) Analise se a interface do ambiente de execução precisa der modificada e, se for o caso, qual a mudança necessária.
c) Implemente a classe relacionada à chamada de procedimento, apenas o método executar (não precisa implementar o método que faz verificação de tipos).

Apêndice. BNF de LI2.

Programa ::= Comando

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

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 | DeclaracaoProcedimento
| DeclaracaoComposta
DeclaracaoVariavel ::= "var" Id "=" Expressao
DeclaracaoComposta ::= Declaracao "," Declaracao
DeclaracaoProcedimento ::= "proc" Id "(" [ListaDeclaracaoParametro] ")" "{"
Comando "}"
ListaDeclaracaoParametro ::= Tipo Id | Tipo Id "," ListaDeclaracaoParametro
Tipo ::= "string" | "int" | "boolean"
While ::= "while" Expressao "do" Comando
IfThenElse ::= "if" Expressao "then" Comando "else" Comando
IO ::= "write" "(" Expressao ")" | "read" "(" Id ")"
ChamadaProcedimento ::= "call" Id "(" [ListaExpressao] ")"
ListaExpressao ::= Expressao | Expressao, ListaExpressao

Classes Auxiliares

AmbienteCompilacaoImperativa2

AmbienteExecucaoImperativa2
ContextoCompilacaoImperativa2
ContextoExecucaoImperativa2
ListaValor
Procedimento
DefProcedimento
ProcedimentoJaDeclaradoException
ProcedimentoNaoDeclaradoException