Paradigmas de Linguagens de Programação
Exame Escrito

Centro de Informática - UFPE
19 de junho de 2009

Questão 1 [2,0] Defina uma função ***bag*** que recebe uma lista de elementos e retorna uma lista de pares, onde o primeiro elemento de cada par é um elemento da lista original e o segundo é o número de ocorrências deste elemento. Nesta segunda lista, cada elemento só ocorre uma vez. Por exemplo***, bag [a,b,a,c,a,b] = [(a,3),(b,2),(c,1)]***

Questão 2 [2,0] a) Defina as funções auxiliares necessárias e descreva uma propriedade que estabelece que o tamanho da lista original é igual à soma dos segundos elementos dos pares da lista gerada. Por exemplo, a lista da questão anterior tem tamanho 6, que equivale a soma dos elementos 3 + 2 + 1 da segunda lista.
b) prove esta propriedade para o caso base.

Questão 3 [2,0] Quais das linguagens apresentadas no curso são, ***estritamente***, linguagens de programação? Com todas as que são linguagens de programação é possível desenvolver programas que expressam exatamente a mesma classe de problemas? Ou umas podem expressar soluções para problemas que outras não permitem?

Questão 4 [4,0] Estenda a Linguagem Imperativa 2 (veja BNF em anexo) permitindo funções e chamadas de funções (de primeira ordem), considerando que o corpo de uma função é uma expressão e que a chamada é uma expressão e não um comando.

a) Defina a BNF para a linguagem redefinida, destacando apenas o que mudar.
b) Defina a interface do ambiente de execução com todos os métodos necessários à implementação de um interpretador para a linguagem redefinida. Explique suas decisões de projeto e os atributos necessários à implementação da interface do ambiente, mas NÃO precisa implementar os métodos.
c) Implemente as classes relacionadas à declaração e chamada de uma função (não precisa implementar o checa tipo).

Apêndice 1. 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