

# Processamento de Linguagens I

## LESI + LMCC (3<sup>o</sup> ano)

Exame de 1<sup>a</sup> Época – 2<sup>a</sup> Chamada

Data: 10 de Julho de 2006  
Hora: 14:00

Dispõe de 2:30 horas para realizar este exame
---

Um amigo seu é dono de um pequeno ourivesaria, ODume, onde vende: **jóias** de vários tipos—anéis, brincos, colares e alfinetes, em ouro ou prata, lisos ou com pedras preciosas (rubis, ametistas topázios, esmeraldas); **objectos de prata**—jarras, taças, molduras, cigarreiras, candelabros, faqueiros; e **relógios**—pulso, mesa (despertadores) e parede (com ou sem pé)).

Por razões mais que óbvias, os gestores da ODume encomendaram uma série de aplicações informáticas para controlo do stock, das vendas e das encomendas a fabricantes. Porque se licenciaram na UM, os ditos gestores conhecem bem o potencial da informática nos dias que correm, e portanto pretendem que as várias aplicações se interliguem para poderem usar ferramentas de *análise de dados* e *marketing/merchandising*.

As questões deste exame serão colocadas neste contexto.

### Tradução Dirigida pela Semântica

#### Questão 1: gramática de atributos

Escreva uma **Gramática de Atributos**, *GA*, para descrever as jóias.

Cada jóia será descrita, no mínimo, por uma *referência*, o *tipo*, a *data de entrada*, o *fabricante*, o *preço base*, o *peso* (em gramas) da peça relativo ao *material de base* (*ouro* ou *prata*), o *número* e *tipo de pedras preciosas*.

A sua linguagem deve permitir descrever, no início, a lista das matérias primas que podem ser usadas—ouro, prata, pedras preciosas—com o preço unitário (por grama, no caso dos metais nobres), associado a cada uma.

Com essa *GA* pretende-se especificar um processador que gere, em HTML, uma **tabela de preços** das jóias onde constem, no mínimo, a *referência*, o *tipo* e o *preço final* (preço base mais o custo do metal nobre usado mais o custo das pedras preciosas).

O processador a desenvolver deve, ainda, verificar que o nome do material de base e das pedras preciosas está correcto, isto é, faz parte da lista inicial de matérias primas.

Para facilitar a leitura da sua resposta, reúna numa tabela (no início ou no fim) os **atributos herdados e sintetizados, ou intrínsecos** de cada símbolo (NT ou T) de *GA*.

## Tradução Dirigida pela Sintaxe

Para gestão de vendas desenhou-se a gramática independente de contexto  $G$ , abaixo apresentada. O Símbolo Inicial é **Vendas**, os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre apostrofes (sinais de pontuação) e a string nula é denotada por  $\&$ ; os restantes serão os Símbolos Não-Terminais.

```
p1: Vendas --> data LstVds
p2: LstVds --> Venda RLst
p3: RLst --> &
p4:      | Venda LstVds
p5: Venda --> Ref Classe Qt PrU '!'
p6: Ref --> id
p7: Classe --> JOIA
p8:      | OBJP
p9:      | RELOG
p10: Qt --> int
p11: PrU --> real
```

### Questão 2: parsing

Recordando os seus conhecimentos sobre análise sintáctica *Top-Down* e *Bottom-Up*, responda às alíneas seguintes:

- Mostre que  $G$  é **LL(1)**, construindo as Tabelas de Parsing **LL(1)**.
- Escreva a função, pertencente a um parser Recursivo-Descende Puro, para reconhecer o símbolo Não-Terminal **RLst**.
- Supondo que vai analisar a frase

2006-06-18 A01292 JOIA 8 25.00 !

com um parser *Top-Down LL(1)*, desenhe a respectiva *Árvore de Parsing* indicando a ordem de redução dos nodos.

- Verifique, construindo as respectivas tabelas de parsing **ACTION** e **GOTO**, se a gramática apresentada  $G$  é **SLR(1)**.

### Questão 3: gramática tradutora

Transforme  $G$  numa **gramática tradutora**,  $GT$ , reconhecível pelo `yacc`, para: gerar código *Assembly* da Máquina de Stack Virtual MSP (instruções listadas em anexo) de modo a calcular o valor total das vendas de um dia.

## Validação da Componente Prática

### Questão 4: sobre o Trabalhos Práticos

Recorde o seu 2º Trabalho Prático. Diga que tipos estruturados o seu grupo escolheu e implementou, referindo quais as maiores dificuldades enfrentadas na análise semântica e na geração de código.

Indique, também, se incluiu na linguagem algum tipo de instrução de controlo condicional ou cíclica (qual, ou quais). Por fim explique qual a ideia de gerar código para uma Máquina Virtual e descreva sucintamente a estrutura da MV usada.

# Instruções Assembly da Máquina de Stack Virtual MSP

## Grupo 1: Manuseamento de Valores e Endereços

```
PUSH valor ;coloca o valor inteiro no Topo(Stack)
PSHA ender ;coloca o endereço no Topo(Stack)
LOAD ;carrega para o Topo(Stack) o valor que está no endereço contido no Topo(Stack)
LDA ;carrega um endereço no Topo(Stack)
STORE ;armazena na memória o valor que está no Topo(Stack), no endereço contido no Topo(Stack)-1
STRA ;armazena na memória o endereço do Topo(Stack), no endereço contido no Topo(Stack)-1
IN ;lê um inteiro para o Topo(Stack)
INC ;lê um caracter para o Topo(Stack)
OUT ;escreve um inteiro que está no Topo(Stack)
OUTC ;escreve um caracter que está no Topo(Stack)
```

## Grupo 2: Operações Aritméticas e Lógicas

```
ADD ;adiciona dois inteiros
SUB
MUL
DIV
ADDA ;adiciona um inteiro a um endereço
AND ;E lógico
OR ;OU lógico
NOT ;NEGAÇÃO lógica
EQ ;igual
NE ;diferente
LT
LE
GT
GE
```

## Grupo 3: Instruções de Controlo

```
JUMP label ;salta para a instrução de programa etiquetada pela 'label'
JMPF label ;salta para a 'label', se Topo(Stack) for Falso
CALL label ;salta para a subrotina cujo código começa na 'label'
RET ;retorna de uma subrotina
NOOP ;não faz nada
HALT ;termina a execução
```