Processamento de Linguagens I
LESI + LMCC (3º ano)

Trabalho Prático nº 1
(Lex e Yacc)

Ano lectivo 2003/2004

1 Objectivos e Organização

Este trabalho prático tem como principais **objectivos**:

- aumentar a experiência de uso do ambiente linux, da linguagem imperativa C (para codificação, estruturas de dados e respectivos algoritmos de manipulação), e de algumas ferramentas de apoio à programação;
- rever e aumentar a capacidade de escrever *gramáticas independentes de contexto*;
- desenvolver processadores de linguagens segundo o método da *tradução dirigida pela sintaxe*, suportado numa *gramática tradutora*;
- utilizar *geradores de compiladores* como o par lex/yacc

Para o efeito, esta folha contém 6 enunciados, dos quais deverá resolver um.

O trabalho deve ser entregue ao docente a funcionar (acompanhado do respectivo relatório de desenvovlimento) e defendido pelo grupo (3 alunos), em data a marcar.
O trabalho deve ser entregue no 1º dia de aulas a seguir às férias da Páscoa (4ªfeira, 14 de Abril) através da sistema electrónico para recepção de TPs cujo endereço será disponibilizado na página WWW da disciplina (os alunos devem fazer a inscrição do grupo no dito sistema o mais cedo possível).
O **relatório** a elaborar, deve ser claro e, além do respectivo enunciado e da descrição do problema, deverá conter exemplos de utilização e o código do programa. Como é de tradição, o relatório será escrito em LaTeX.

2 Enunciados

2.1 Maquina de Venda de Chocolates

Pretende-se simular o funcionamento de uma máquina de vender chocolates.
Dado o stock no início do dia (nome, preço e quantidade de cada produto disponível), a quantia inicial de trocos e os registos das vendas diárias (nome do chocolate escolhido e a quantia introduzida), o objectivo é calcular: a evolução do stock ao longo do dia; e o dinheiro acumulado.
A máquina não fornecerá nenhum chocolate se: o produto seleccionado não existir em stock; ou o nome do produto não constar da lista de produtos disponíveis na máquina; ou a quantia introduzida for inferior ao preço. Em qualquer um destes casos o movimento de venda é ignorado.
Caso a quantia introduzida não seja exacta, a máquina deverá restituir o excesso. Note que a

quantia acumulada deverá corresponder ao número de chocolates vendidos vezes o preço respectivo.
A máquina tem uma capacidade máxima de produtos disponíveis sendo também limitado o número de produtos de cada tipo.
A animação pretendida deverá mostrar através de desenhos o estado da máquina (stock existente e dinheiro ganho) após cada movimento.

Para resolver este problema pretende-se: que defina uma linguagem para descrever o estado inicial da máquina e os registos de vendas efectuadas durante o dia; e que desenvolva um processador que gere o programa de animação em Java.
Para realizar este projecto considere a seguinte lista de tarefas:

1. Especificar a gramática concreta da linguagem
2. Desenvolver um reconhecedor léxico e sintáctico para essa linguagem
3. Construir o gerador de código para o animador (em Java), associando acções semânticas de tradução às produções da gramática;
4. Proceder à animação executando o *applet* Java gerado.

Para construir a animação use funções de desenho da biblioteca Java.AWT que serão invocadas a partir de um *applet* com uma estrutura fixa. O *template* desse *applet* está disponível na página da disciplina (`http://www.ipb.pt/~mjoao/ProcLing.html`).

2.2 Robot de limpeza

Pretende-se simular o comportamento de um aspirador automático que faz a limpeza de uma unidade industrial rectangular (de dimensões a definir em cada caso), de modo a visualizar-se no monitor as zonas que foram limpas e as que ficaram por aspirar. Os movimentos do aspirador serão sempre em linha recta para a esquerda ou direita, para cima ou para baixo. Em cada deslocação pode ter o sistema de aspiração ligado ou desligado e pode mover-se um número de passos variável (por omissão será 1 passo).
O canto inferior esquerdo do pavilhão é o ponto de repouso, tendo, por isso, coordenadas (0,0); considere que esta é sempre a posição inicial. O robot é ligado automaticamente pelo primeiro comando de deslocamento; será desligado pelo comando de repouso que o coloca na posição inicial.

Para resolver este problema pretende-se: que defina uma linguagem para programar o robot; e que desenvolva um processador que gere o programa de animação em Java. Essa linguagem terá que permitir definir as dimensões do pavilhão industrial. O processador deve validar os movimentos do robot de modo a não deixar ultrapassar os limites.
Para realizar este projecto considere a seguinte lista de tarefas:

1. Especificar a gramática concreta da linguagem
2. Desenvolver um reconhecedor léxico e sintáctico para essa linguagem
3. Construir o gerador de código para o animador (em Java), associando acções semânticas de tradução às produções da gramática;
4. Proceder à animação executando o *applet* Java gerado.

Para construir a animação use funções de desenho da biblioteca Java.AWT que serão invocadas a partir de um *applet* com uma estrutura fixa. O *template* desse *applet* está disponível na página da disciplina (`http://www.ipb.pt/~mjoao/ProcLing.html`).

2.3 Anuário dos Medicamentos brancos

Para auxiliar o Instituto Farmacêutico do Ministério da Saúde na gestão do novo lote de medicamentos brancos, pretende-se criar um sistema de consulta, acessível a partir de qualquer farmácia via um browser HTML.
Esse sistema deve mostrar a informação agrupada por: classe de medicamentos no Symposium Terapêutico (uma página por classe, com os medicamentos ordenados alfabeticamente); ou por fabricante (uma página única, com os medicamentos agrupados por fabricante).
Sobre cada medicamento é fornecida a seguinte documentação: nome, código, classe, composição química, preço recomendado, fabricantes disponíveis e lista de medicamentos de marca equivalentes (respectivo nome e fabricante).

Para resolver este problema pretende-se: que defina uma linguagem para descrever a informação envolvida no lote de medicamentos a considerar; e que desenvolva um processador que gere as páginas em HTML para implementar o sistema de consulta pedido.
Essa linguagem terá que permitir definir inicialmente o ano a que o Symposium Terapêutico diz respeito e a lista das classes de medicamentos. O processador deve validar que não haja definições repetidas de medicamentos e que a classe associada a cada medicamento exista na lista.
Para realizar este projecto considere a seguinte lista de tarefas:

1. Especificar a gramática concreta da linguagem
2. Desenvolver um reconhecedor léxico e sintáctico para essa linguagem
3. Construir o gerador de HTML, associando acções semânticas de tradução às produções da gramática;
4. Proceder à consulta abrindo em Netscape ou IE o ficheiro HTML gerado.

2.4 Construção automática de páginas pessoais

Pretende-se criar uma linguagem muito simples para construção automática de páginas pessoais em HTML. Para muitos utilizadores de Internet é ainda difícil a construção da sua página pessoal. Com esta nova linguagem, composta por um número reduzido de comandos (para permitir a introdução de nome, identificação, contactos, títulos, sub-títulos, items, palavras com links, palavras em destaque, etc), o utilizador poderá programar a sua página pessoal sem recorrer a HTML.

Para resolver este problema pretende-se: que defina uma linguagem para descrever a informação envolvida na página pessoal; e que desenvolva um processador que gere o código HTML dessa página.
O sistema deve considerar alguns campos de informação obrigatórios e outros opcionais e deverá notificar o utilizador quando surgirem problemas relacionados com a falta de campos obrigatórios.

Para realizar este projecto considere a seguinte lista de tarefas:

1. Especificar a gramática concreta da linguagem
2. Desenvolver um reconhecedor léxico e sintáctico para essa linguagem
3. Construir o gerador de HTML, associando acções semânticas de tradução às produções da gramática;
4. Proceder à consulta abrindo em Netscape ou IE o ficheiro HTML gerado.

2.5 Mensagens de Email

Considere que, no âmbito de uma aplicação para tratamento e gestão de mensagens de correio electrónico de um grupo de pessoas, se pretende criar um *Address Book* para cada utilizador.

A base de dados de *Address Book* deve registar o "username"dos utilizadores do sistema e cada *Address Book* deve conter o endereço de correio electrónico de todos aqueles com quem o utilizador se corresponde.
Nesse contexto, uma mensagem de correio electrónico é constituída por:

- um cabeçalho, um corpo e anexos;
- no cabeçalho é indicada a data e a hora do envio da mensagem, o email do remetente, a lista de destinatários a quem é enviada, a lista de pessoas a quem se quer dar conhecimento e o assunto abordado na mensagem;
- o corpo é constituído por um ou mais parágrafos de texto;
- cada um dos anexos é constituído por uma referência a um ficheiro externo (o seu nome, ou path).

Cada endereço referido na mensagem é formado por um par que começa pelo nome da pessoa (escrito entre apóstrofes) seguido pelo endereço propriamente dito (trabalhe com versões simplificadas, mas válidas). O "username"dos utilizadores válidos do sistema é retirado do campo "remetente", enquanto que o dos seus amigos (a colocar no *Address Book*) é extraído dos campos "destinatário"e "com cópia a".

Para resolver este problema pretende-se que: defina uma linguagem para descrever um conjunto de mensagens de correio electrónico (cada uma com o formato acima descrito); e que desenvolva um processador que gere comandos em SQL para carregar as tabelas da base de dados com os endereços pedidos. O processador deve ainda fazer alguma estatística, indicando o total de mensagens processadas, o movimento total diário e movimento total por "remetente", bem como deve validar que não haja endereços repetidos na mesma mensagem.
Para realizar este projecto considere a seguinte lista de tarefas:

1. Especificar a gramática concreta da linguagem
2. Desenvolver um reconhecedor léxico e sintáctico para essa linguagem
3. Construir o gerador de SQL, associando acções semânticas de tradução às produções da gramática;
4. Proceder à consulta dos utilizadores válidos e seus *Address Book's*, abrindo a base de dados em Access.

2.6 Provas de Orientação

A Orientação, na sua variante pedestre, é um desporto com grande desenvolvimento em Portugal. Basicamente, um atleta recebe um mapa que tem um percurso desenhado; nesse percurso, há uma série de *pontos quentes* assinalados que o atleta terá que *visitar* pela ordem em que estão numerados; no fim do percurso, quando retorna ao ponto de partida, é classificado em função do número de pontos visitados e do tempo gasto.Em cada prova os concorrentes são agrupados em escalões etários, aos quais correspondem percursos diferentes.

Para ajuda à organização de provas pretende-se desenvolver um sistema que permita definir os percursos. Neste contexto, apenas se pretende tratar da componente relacionada com a definição da prova, a qual é feita através da especificação da lista de percursos que a compõem, de modo a que se possa calcular o comprimento de cada um e visualizar os trajectos.

Para resolver este problema pretende-se que: defina uma linguagem para descrever os percursos de uma prova; e que desenvolva um processador que calcule o comprimento de cada percurso

e gere instruções em GraphViz para desenhar os percursos. Essas instruções de desenho serão posteriormente usadas pelo interpretador de GraphViz para visualizar o grafo dos percursos da prova (será disponibilizada informação sobre GraphViz na página da disciplina).
A linguagem pedida deve permitir que se comece por identificar todos os pontos quentes do terreno onde a prova vai decorrer, sendo que cada ponto será identificado por uma sigla e pelas suas duas coordenadas. Depois a linguagem servirá para definir cada percurso, indicando o seu nome, o escalão etário a que se destina e a lista de pontos (referidos pelas respectivas siglas) que o compõem. Esta lista é que determina a ordem de visita.
Recorde-se que a distância entre dois pontos pode ser calculada pela fórmula:

$$dist = \sqrt{|y2 - y1|^2 + |x2 - x1|^2}$$

Para realizar este projecto considere a seguinte lista de tarefas:

1. Especificar a gramática concreta da linguagem
2. Desenvolver um reconhecedor léxico e sintáctico para essa linguagem
3. Construir o gerador de GraphViz, associando acções semânticas de tradução às produções da gramática;
4. Proceder à consulta dos pontos quentes existentes e percursos construídos executando o código Graphviz.