Processamento de Linguagens
LEI (3º ano)

Trabalho Prático nº 1
(Lex)

Ano lectivo 07/08

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 das estruturas de dados e respectivos algoritmos de manipulação), e de algumas ferramentas de apoio à programação;
- aumentar a capacidade de escrever *Expressões Regulares (ER)* para descrição de *padrões de frases*;
- desenvolver, a partir de ERs, sistemática e automaticamente *Processadores de Linguagens Regulares*, que filtrem ou transformem textos;
- utilizar *geradores de filtros/processadores de texto*, como o Flex

Para o efeito, esta folha contém 5 enunciados, dos quais deverá resolver pelo menos um. O programa desenvolvido será apresentado aos membros da equipa docente, totalmente pronto e a funcionar (acompanhado do respectivo relatório de desenvolvimento) e será defendido por todos os elementos do grupo (3 alunos), em data a marcar.
O **relatório** a elaborar, deve ser claro e, além do respectivo enunciado, da descrição do problema, das decisões que lideraram o desenho e a implementação, deverá conter exemplos de utilização (textos fontes diversos e respectivo resultado produzido). Como é de tradição, o relatório será escrito em LaTeX.

2 Enunciados

Para sistematizar o trabalho que se lhe pede em cada uma das propostas seguintes, considere que deve, em qualquer um dos casos, realizar a seguinte lista de tarefas:

1. Especificar os padrões de frases que quer encontrar no texto-fonte, através de ERs.

2. Identificar as acções semânticas a realizar como reacção ao reconhecimento de cada um desses padrões.

3. Identificar as Estruturas de Dados globais que possa eventualmente precisar para armazenar temporariamente a informação que vai extraindo do texto-fonte ou que vai construindo à medida que o processamento avança.

4. Desenvolver um Processador de Texto para fazer o reconhecimento dos padrões identificados e proceder à transformação pretendida, com recurso ao Gerador Flex.

2.1 Pré-processador para LaTeX ou HTML

desenvolver um documentoem LaTeX ou mesmo em HTML é uma actividade inteligente, e intelectualmente interessante enquanto estruturante das ideias e sistematizante dos processos. Porém o acto de editar o respectivo documento é por vezes fastidioso devido ao peso das marcas (as *tags*) que tem de ser inseridas para anotar o texto com indicações de forma, conteúdo ou formato.

Por isso apareceram editores sensíveis ao contexto que sabendo que se está a escrever um documento LaTeX ou HTML nos facilitam a vida inserindo as ditas marcas, ou anotações. Uma alternativa mais simples mas também muito usada é permitir o uso de anotações mais leves e simples (até de preferência independentes do tipo de documento final) e de pois recorrer ao pré-processamento para substituir essa notação ligeira, abreviada, pelas marcas finais correctas.

Este é o caso do conhecido PPP[1], desenvolvido há alguns anos por José Carlos Ramalho, ou mesmo do mais actual e não menos conhecido sistema Wiki para construção interactiva e via web de páginas HTML.

O que se lhe pede neste trabalho é que, depois de investigar os tais pré-processadores PPP e Wiki, especifique uma sua linguagem de anotação para abreviar a escrita de **formatação** (bold, itálico, sublinhado, vários níveis de títulos) ou em alternativa de **listas de tópicos (items)** não-numerados, numerados ou tipo entradas de um dicionário. Deve, depois, criar, com a ferramenta Flex, um processador que transforme a sua notação em LaTeX ou HTML[2].

2.2 LaTeX importer

Desenvolva um pré-processador que aceite texto LaTeX com mais uma marcação especial `umImport`,

```
\begin{umImport}{gnuplot}
...texto gnuplot....
\end{umImport}
```

e que receba, como parâmetro, um segundo elemento (no exemplo acima gnuplot) indicativo do processador a utilizar.

Como resultado o pré-processador deverá:

[1] Consultar detalhes no manual da linguagem em `http://www.di.uminho.pt/~jcr/AULAS/plc2008/tp1/ppp.html`.

[2] O mais interessante mesmo é que fosse possível escolher a saída final no inicio do próprio texto a pré-processar.

- copiar para um ficheiro auxiliar o texto em causa (marcado pelo novo elemento),
- executar um comando externo que construa uma imagem PDF aplicando a esse ficheiro o processador indicado,
- substituir o texto e a nova marca que o envolve pelo comando `includegraphics` para importar a imagem PDF produzida.

Sugestões: comece por considerar os formatos gnuplot e dot, mas guarde numa tabela os comandos externos a executar para produzir a imagem PDF, de modo a facilitar a definição de novos formatos de importação.

2.3 Wikipedia

Descarregue a Wikipedia-PT (versão XML) e extraia um dicionário PT-EN[3]

Para tal analise os separadores de entradas, os definidores de termo (¡title¿) os definidores de equivalente noutra língua ([[en:...]]), os definidores de categoria ([[categoria:...]] ou [[category:...]]).

No final construa um ficheiro com o seguinte aspecto:

```
PT pal1
EN pal1en
Categoria ...

PT pal2
EN pal2en
Categoria ...
```

Ignore as entradas que não contenham pelo menos PT-EN. Em complemento, verifique se pode extrair outras informações relevantes a acrescentar à saída.

Alternativamente parta de outra Wikipedia-Lingua.xml, ou então do Wiktionary (com diferentes convenções).

2.4 XML to dot

Pretende-se neste trabalho criar e desenhar um grafo de dependências entre os elementos de um documento anotado em XML.

Para isso e dado um ficheiro XML, cada vez que encontrar um elemento X com um subelemento Y, exemplo:

```
<X> ...
    <Y> ... </Y>
    ...
</X>
```

deve gerar uma linha dot-graphviz de modo a que no final se possa obter uma árvore documental com a estrutura de elementos. No final gere a imagem respectiva com o comando dot.

Exemplo de ficheiro dot a gerar:

[3]Cuidado que os tamanhos destes ficheiros são grandes!

```
strict  digraph g {
x -> y ;
a -> b ;
a -> x ;
x -> c ;
x -> c ;
}
```