Processamento de Linguagens
LEI (3º ano)

1º Ficha Prática

Ano Lectivo de 07/08

1 Objectivos

Este ficha prática contém exercícios para serem resolvidos nas aulas teórico-práticas com vista a sedimentar os conhecimentos relativos a:

- Motivação para o uso de Expressões Regulares como forma de especificar padrões a pesquisar em textos —recurso a utilitários de Linux que seguem essa abordagem;
- uso de Expressões Regulares para definir (gerar) Linguagens Regulares;
- uso de Expressões Regulares para desenvolver programas eficientes, baseados em algoritmos standard guiados por Autómatos Finitos Deterministas, para reconhecer Linguagens Regulares;
- uso de Autómatos Deterministas Reactivos, para processar Linguagens Regulares, isto é para desencadear Acções específicas ao reconhecer frases que derivam de Padrões (definidos com base em ERs) —princípio da Programação baseada em regras *Condição-Reacção*;
- geração automática de programas a partir de especificações formais;
- uso da ferramenta Flex, disponível em ambiente Linux, para geração automática de processadores de linguagens regulares, nomeadamente para criação de *Filtros de Texto* ou de *Analisadores Léxicos*.

2 Pesquisa de Padrões em Texto

Dado um conjunto de ficheiros de texto para teste, usa o utilitário egrep para

a) Procurar em "teste1.txt"todas as linhas com marcas HTML e só essas linhas.

b) Procurar em "teste1.txt"todas as linhas com marcas HTML encostadas ao inicio da linha.

c) Procurar em "teste1.txt"todas as linhas com marcas HTML encostadas ao fim da linha.

d) Procurar em "teste1.txt"as linhas com marcas HTML que contenha a letra "H"(tipo HTML ou HREF).

e) Procurar em "teste1.txt"todas as linhas que contenham o algarismo 1; teste variantes do género: no fim da linha, na casa das dezenas, no inicio do número, etc.

f) Procure na colecção de testes todos os ficheiros que contenham a palavra Daniela; teste variantes do género: esse nome independentemente de começar por minúscula ou maiúscula; Daniela no header; Pedro Henriques, etc.

3 Filtros de Texto

Para introduzir a ferramenta de geração de programas FLex baseada em especificações com Expressões Regulares, e para sedimentar os conhecimentos que adquiriu sobre autómatos deterministas reactivos como suporte à construção de programas eficientes, propõem-se alguns exercícios, para resolver dentro ou fora da aula, que visam a criação de programas autónomos para *filtrar textos* (FT).

3.1 Processador de Questionários

Suponha que ao fim de cada entrevista um Repórter produz um texto com as perguntas e respostas, distinguindo umas das outras porque as perguntas começam com "EU:", no início da linha, e as respostas começam com "ELE:", também no início da linha.
Nesse contexto, pretende-se desenvolver um FT para processar os questionários que:

a) simplesmente retire do texto original as tais marcas "EU:" e "ELE:", devolvendo todo o resto da entrevista sem qualquer alteração.
Melhore o filtro, de modo a tratar as marcas, quer estejam escritas em maiúsculas, quer em minúsculas;

b) substitua a marca "EU" pela palavra "Entrevistador" e a marca "ELE" por "Entrevistado";

c) substitua a marca "EU" pelo nome do entrevistador e a marca "ELE" pelo nome do entrevistado, supondo que no início encontrará as respectivas definições (ordem irrelevante) na forma "EU=nome." ou "ELE=nome."[1]

3.2 Expansor de Abreviaturas

Quando se retiram apontamentos, ou de uma forma geral, se tem de escrever muito depressa, é hábito usar abreviaturas que correspondam a uma ou mais palavras vulgares e longas.
Suponha que criou esse costume e resolveu inserir nos seus textos as ditas abreviaturas (2 ou mais letras) precedidas pelo carácter "\". Por exemplo: "\qq"(qualquer), ou "\mb"(muito bom), ou ainda "\sse"(se e só se).
Desenvolva, então:

a) um FT que lhe devolva o texto original mas com todas as abreviaturas (que definiu à partida) devidamente expandidas;

b) melhore o seu filtro de modo a contemplar ainda o tratamento do carácter "/" no fim de uma palavra, representando o sufixo "mente", e o carácter "~"no início de uma palavra, representando o prefixo "in". Uma palavra pode conter ambos os caracteres, um no início e outro no fim (pense na abreviatura da palavra "infelizmente");

c) outra melhoria que poderia introduzir no seu filtro era contemplar a possibilidade de definir abreviaturas dentro do próprio texto, na forma "\def:abrev=termo-expandido;". Pense como o fazer e nas implicações que tal requisito teria no seu filtro original.[2]

3.3 Somador de Números

Construa um Filtro de Texto que adicione todos os números dum texto e que, no final, imprima a sua soma (no ficheiro de saída não deve aparecer nenhum caracter do texto de entrada).
Evolua o seu processador no sentido de:

[1] Alínea proposta para pensar fora da aula.
[2] Alínea proposta para pensar fora da aula.

a) Escrever a soma sempre que encontre o carácter '='.

b) Só comece a somar quando detectar o carácter '+' e deixe de somar quando este carácter voltar a aparecer.

3.4 Documento anotado em XML

Como sabe um Documento XML é um texto vulgar semeado de anotações, ou marcas, que são identificadores especiais (designados por *elementos XML*) intercalados entre os caracteres "<" e ">". Num documento XML bem formado, a cada *marca de abertura* corresponderá uma *marca de fecho*, que tem o mesmo identificador, mas que começa por "</" terminando na mesma em ">".
Desenvolva um filtro de texto (FT) que receba um documento XML e:

a) devolva o texto original, após ter retirado todas as marcas;

b) conte o número de *marcas de abertura* e o número de *marcas de fecho*, indicando *erro* sempre que se verifique um desequilíbrio entre ambas[3];

c) verifique a concordância entre as *marcas de abertura* e as *marcas de fecho*, isto é, garanta que as marcas se fecham por ordem inversa que se abrem[4]. No fim produza uma listagem, ordenada alfabeticamente, de todos os elementos distintos encontrados.

3.5 Normalizador de Emails

Os Emails escritos à moda PRH caracterizam-se por terem todas as palavras começadas por letras minúsculas, à excepção dos nomes próprios e siglas.
Desenvolva, então:

a) um FT que normalize o texto, *capitalizando* (escrevendo a letra inicial em maiúsculas) todas as palavras no início de cada frase. Além da primeira palavra do texto, uma frase começa depois de um '.', '?' ou '!', seguido de zero ou mais espaços, eventualmente um ou mais fim-de-linha;

b) complete a especificação anterior de modo a que o seu *normalizador de emails prh* conte também todos os nomes próprios (palavras começadas por maiúsculas) e siglas (palavras formadas só por maiúsculas, uma ou mais) encontradas no texto original.

[3]Aqui apenas se pede que detecte o erro por contagem e não atendendo ao *identificador do elemento* em causa em cada marca.

[4]Mas agora tomando em atenção o *identificador do elemento* em causa em cada marca.