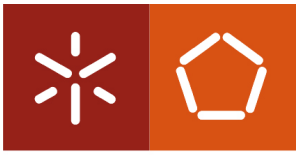

Universidade do Minho

LEI — Licenciatura de Engenharia Informática

Processamento de Linguagens

Trabalho pratico 1

Paulo Araujo - a58925, Orlando Costa - a67705, Rui Oliveira - a67661

Braga, 25 de março de 2015

Conteúdo

Capítulo 1

Introdução

O presente projeto enquandra-se na unidade curricular de Processamento de Linguagnes do curso de Licenciatura em Engenharia Informática da Universidade do Minho. O projeto pretende aumentar as capacidades com as expressões regulares, desenvolvendo processadores de linguagens regulares utilizando o gerador de filtros de texto Flex. Para isso foram selecionados 3 exercícios dentro de um grupo de 8 exercícios, são eles: *2.1 Museu da Pessoa — tratamento de fotografias*, *2.2 Processamento de Entidades Nomeadas (Enamex)* e *2.5 Processamento de ficheiros com Canções*.

Capítulo 2

Museu da Pessoa — tratamento de fotografias

<Pequeno resumo do problema>

2.1 Analise e Especificação

<O que é preciso fazer de forma tecnica, i.e. temos que apanhar isto e acoloutro guardar mas depois temos que >

2.2 Implementação

2.2.1 Estrutura de dados

<estrutura de dados criada>

2.2.2 Filtro de Texto

<ficheiro flex>

2.2.3 Funcionamento

<Como o ficheiro flex funciona e chama poe na estrutura de dados>

2.3 Testes realizados

<alguns exemplos>

Relatório

Capítulo 3

Museu da Pessoa — tratamento de fotografias

Neste problema é pretendido que, a partir de um ficheiro *XML*, seja criado um filtro de texto capaz de interpretar informação relativa às entrevistas feitas para construção o Museu da Pessoa. Com toda essa informação, deverá ser criado um album em *HTML* possuidor de um índice, ordenado alfabeticamente, contendo todos os nomes de pessoas presentes no álbum. Além disso, cada elemento do índice deverá estar referenciado para a página que contem todas as fotos da respectiva pessoa. Relativamente às fotos, essas deverão estar ordenadas cronologicamente e a descrição da foto deverá ser o seu título/cabeçalho.

3.1 Analise e Especificação

Após a análise do que é pedido no enunciado e de alguns dos *Datasets*, foi nos possível verificar qual a informação essencial a retirar do ficheiro *XML*. Além disso, como não é regra que as fotos e pessoas presentes no ficheiro *XML* estejam já na ordem pretendida, será necessário que toda a informação seja armazenada em estruturas de dados. O nome do ficheiro *HTML* resultante, poderá ser dado como argumento e caso não seja, por omissão será *AlbumGerado.html*. Além deste ficheiro, serão gerados tantos ficheiros *HTML* quantas as pessoas presentes no album. Esses ficheiros serão denominados numericamente (1.html,2.html,...) à medida que novos nomes são encontrados. Os ficheiros anteriormente referidos apenas serão gerados no final da leitura e filtragem de ficheiro *XML*.

3.2 Implementação

3.2.1 Estrutura de dados

De forma a dar seguimento ao que foi especificado na secção 3.1, foi necessária a criação de uma estrutura de dados que sirva de suporte aos dados recolhidos. Sendo assim, optamos por criar uma estrutura denominada `photo_node` que terá o formato de uma árvore binária e servirá para

armazenar toda a informação relativa a cada descrição de fotos encontrada no ficheiro *XML*. Nesta estrutura serão guardados o nome do ficheiro que contem a foto, a data e o local em que esta foi tirada, a sua descrição e o nome das pessoas nela presentes. A inserção de fotos nesta árvore será ordenada relativamente à sua data. Além desta estrutura, também criamos outra, denominada `person_node`, com o objectivo que esta guarde toda a informação acerca das pessoas presentes no álbum. Esta estrutura tem o formato de uma lista ligada e guardará informação como o nome da pessoa, assim como o nome do ficheiro da sua página *HTML* e terá ainda um apontador para uma estrutura `photo_node` em que será armazenada toda a informação relacionada com as suas fotos. Finalmente, criamos uma estrutura `Album` que apenas conterá um apontador para uma estrutura `person_node`, onde estará a informação relativa às pessoas nele presente, e um contador que servirá para contar o número de pessoas presentes no álbum. O contador servirá de auxílio na criação dás páginas *HTML* numeradas referidas na secção 3.1.

3.2.2 Filtro de Texto

Um dos objetivos deste trabalho prático é a utilização de geradores de filtro de texto, como o *Flex*. Sendo assim, foi criado um ficheiro Flex que permite encontrar determinados padrões de expressões regulares e executar uma determinada acção para cada uma delas.

No ficheiro referido, podemos encontrar algumas instruções em `linguagem C` como a inclusão de ficheiros de cabeçalhos (headers, .h) e a declaração de variáveis. De seguida, são definidas as expressões regulares e as respetivas acções que se pretendem realizar no caso da identificação positiva do referido padrão:

`\<foto[ \t]+[a-zA-Z]+=\".*\"` A partir da análise do ficheiro *XML* exemplificado no enunciado, foi possível verificar que a descrição de uma foto começa com o nome do ficheiro que a contem da seguinte forma: `<foto ficheiro="ficheiro.jpg">`. Sendo assim, quando este padrão é encontrado, é inicializado um `photo_node` com o nome encontrado entre as aspas. O nome é obtido retirando da expressão apenas o que se encontra à frente da primeira ocorrência de aspas e atrás da ocorrência seguinte.

`\<quando[ \t]+[a-zA-Z]+=\"[0-9]{4}(.|-)[0-9]{2}(.|-)[0-9]{2}` De seguida pretendemos encontrar a data em que a foto foi tirada. A partir do exemplo do enunciado, verificamos que a data é descrita da seguinte forma: `<quando data="1961-01-15"/>`. Sendo assim, pretendemos encontrar todas as expressões no formato referido, com a possibilidade que a data esteja separada por pontos em vez de traços. A obtenção da data faz-se retirando apenas o que se encontra à frente da primeira ocorrência de aspas.

`\<quem>[ \t\na-zA-ZÀ-û,\"0-9;:]+` Um padrão essencial a encontrar é o nome das pessoas presentes na foto. Esses nomes devem estar descritos da seguinte forma:
`<quem>Ana de Lourdes de Oliveira Chamine; Antonio Oliveira Machado</quem>`
Nesta fase, optamos por não separar ainda os nomes por tokens porque será mais útil fazê-lo apenas quando a descrição da foto estiver completa. Sendo assim, apenas retiramos tudo o que esteja entre `'>'` e `'<'` e adicionamos à estrutura que contem a descrição da foto.

`\<onde>[ \t\n0-9a-zA-ZÀ-û.,;:\"]+<` Continuando com a análise do exemplo do enunciado, verificamos que uma foto pode ter descrito o local em que esta foi tirada:
`<onde>Casa Machado, Afurada, Vila Nova de Gaia</onde>`
Ora, quando este padrão é encontrado, só é necessário guardar o que está entre `'>'` e `'<'`, assim como no caso anterior.

`\<facto>[ \t\n0-9a-zA-ZÀ-û.,;:\"]+<` Finalmente, o último campo necessário para a descrição da foto é o facto que esta representa: `<facto> Os noivos cortam o bolo de casamento</facto>`. O que é essencial retirar neste caso é o mesmo que nos casos anteriores, ou seja, apenas o que se encontra entre '>' e '<'.

`\<\/foto>` Sempre que é encontrado o padrão que finaliza a descrição de uma foto, `</foto>`, é necessário adicionar o `photo_node` criado a todos os respetivos `person_node` das pessoas que se encontram na descrição da foto. É nesta fase que os nomes das pessoas presentes na foto é separado e identificado. Finalmente adiciona-se a descrição da foto a todas as pessoas nela presente.

`.|\n` Este padrão apenas é utilizado para indicar que sempre que é encontrado qualquer outro `byte` ou `\n`, deve ser ignorado.

3.2.3 Funcionamento

Antes do filtro de texto ser aplicado ao ficheiro *XML* é invocada a função `init` que inicializa uma variável `static Album`. Inicializada esta variável, é possível aplicar o filtro de texto. À medida que a informação do ficheiro *XML* é filtrada, são invocadas funções que tratam e guardam na estrutura a informação recolhida. Sempre que é encontrada uma expressão que defina o início de uma descrição de uma foto, a informação relativa ao nome da foto é recolhida e é invocada a função `initPhoto` que trata de alocar o espaço necessário para a informação que virá a ser recolhida posteriormente. Do mesmo modo, sempre que é encontrada uma expressão que identifique a data, a localização, a descrição ou as pessoas presentes na foto, são invocadas funções capazes de tratar e armazenar essa informação. As funções referidas são `setDate, setLoc, setFact e setWho`, respetivamente. Sempre que não seja possível identificar uma destas características anteriores, estas ficam com os seus valores por defeito, ou seja, caso uma foto não tenha a tag `<quem>`, o seu campo correspondente na estrutura ficará preenchido com `"Desconhecidos"`. Finalmente, quando é encontrada a expressão que define o final da descrição de uma foto, é invocada a função que trata de inserir a informação da nova foto na estrutura `Album`. Depois de tratada toda a informação, são invocadas funções auxiliares que criam e preenchem o(s) ficheiro(s) *HTML* necessários.

3.3 Testes realizados

<alguns exemplos>

Possíveis exemplos de aplicação deste filtro de texto:

- `cat\ <inputFile>\ |\ ./play\ <output1.html>\`
- `cat\ <inputFile>\ |\ ./play\`

Capítulo 4

Processamento de ficheiros com Canções

Neste problema era pretendido que fosse criado um filtro de texto que interpretasse ficheiros com letras de musica, e fosse gerado um ficheiro *latex* para cada musica encontrada. Ainda existe a particularidade de cada ficheiro com musicas poder conter mais do que uma musica, neste caso deve ser criado 2 ficheiros *latex*.

4.1 Analise e Especificação

Existem varias questões que são deixadas em aberto no enunciado que iram ser especificadas nesta secção. O programa irá ler do *stardard* input e os nomes dos ficheiros *latex* que irão ser gerados podem ser recebidos por argumento, caso contrario os nomes assumido utilizam numeração, começando em 0 até à n-ésima musica interpretada. Uma vez que não se sabe a ordem pela qual os cabeçalhos estão nos ficheiros a ser interpretados, o mais seguro será guardar toda a musica em memoria e só imprimir para o ficheiro *latex* depois do fim da musica. Após a analise dos *Datasets* verificou-se a existência de campos no cabeçalho que não são utilizados por o programa, ou seja todos os possíveis campos no cabeçalho devem ser ignorados. Durante a analise também foi verificada a existência de anotações em algumas musicas que serviriam para apresentar as pautas, o nosso programa irá tentar ignorar as marcas e assim tentar apenas imprimir a letra da musica do ficheiro *latex*. Existe ainda outro cuidado na criação do ficheiro *latex* que é a utilização na musica de caracteres especiais no *latex*.

4.2 Implementação

4.2.1 Estrutura de dados

De forma a complementar o enunciado na secção 4.1, foi criada uma estrutura principal chamada `Music`, onde se guarda a informação geral da musica temporariamente até esta ser imprimida para um ficheiro.

Nesta estrutura irá-se guardar o titulo, o nome do autor entre outros campos do cabeçalho necessários, e também a letra da musica.

A letra da musica é guardada numa lista ligada onde cada nodo é um linha da letra e é representada pela estrutura `MusicLine`. A estrutura pode ser encontrada em anexo 5.3.2.

4.2.2 Filtro de Texto

Para a filtragem do texto foram criadas varias expressões regulares, o ficheiro pode ser encontrado em anexo (5.3.1).

As primeiras expressões regulares, do tipo `^title:.+` servem para apanhar os cabeçalhos que serão necessários, para alem do `title` existe mais as seguinte: `from`, `author`, `lyrics`, `music` e `singer`, todas com equivalentes. De forma a ignorar qualquer outro campo do cabeçalho que não tivesse sido previsto foi ainda criada a seguinte expressão regular: `^[a-zA-Z]+:.+`.

Quanto à detenção da letra da musica existem duas expressões regulares: uma para apanhar uma linha da lírica, outra para apanhar as linhas em branco entre os poemas, que são respetivamente: `[ ].*` e `$^\n`.

Tal como dito na analise (4.1), existem algumas anotações no meio da letra da musica que eram necessárias ser retiradas. Para isso foram criadas as seguintes expressões regulares:

- `{abc}(.|\n)*{abcclose}` para retirar a pauta da musica.
- `[ ].*` que ignora as notas no meio dos poemas (pois estas tem um espaço no inicio).

Ainda assim estas duas expressões regulares não eram suficientes e na deteção de uma linha da letra, antes de guardar a linha, passa-se a linha por duas funções: `takeOffAnotations` e `takeOffUnderSccore`. Em que a primeira tira anotações que estao na mesma linha, e a segunda tira os caracteres '_' que estão no meio da linha.

4.2.3 Funcionamento

De forma a perceber melhor o funcionamento do autómato esta secção irá fazer a ponte entre o filtro de texto (4.2.2) e a estrutura de dados (4.2.1).

À medida que o autómato recolhe os campos do cabeçalho da musica, guarda a informação, com as funções de `append`. Como por exemplo `appendAuthor`, `appendLyrics`, entre outras. Estas funções guardam os campos na variável `Music`.

Enquanto que as linhas da letra são guardadas através das funções `appendLine` e `appendWhiteLine`.

Quando é detetado o inicio de uma nova letra, através de expressão regular, é executado `commitCheckNext()` que escreve a letra que esta atualmente na variável `Music` para o ficheiro *latex*, neste ponto caso seja detetado a falta de algum item obrigatório então a escrita para o ficheiro é cancelada. De seguida a variável `Music` é reiniciada para a musica seguinte com a função `Start()`.

Na escrita do ficheiro *latex* a letra é escrita entre as *tags* da *latex* de *Verbatim* para evitar erros no *latex* por falta de caracteres escape.

4.2.4 Testes realizados

Estão documentados neste secção 3 testes realizados ao autómato, utilizando com input os ficheiro que estão em anexo: 5.3.4, 5.3.4 e 5.3.4.

Teste nº 1

Após a utilização do autómato no ficheiro 5.3.4, este gerou o output (5.3.4). Este ficheiro não tem nenhuma situação excecional, é um caso normal.

Teste nº 2

Após a utilização do autómato no ficheiro 5.3.4, este gerou o output (5.3.4). Este ficheiro tem duas situações excecionais, o carácter '_' no meio de palavras e notas musicais no fim das frases. Podemos verificar no output que apenas tem a letra da musica.

Teste nº 3

Após a utilização do autómato no ficheiro 5.3.4, este gerou o output (5.3.4). Este ficheiro tem uma situação excecional, antes da letra da musica tem as tags `<abc>...</abc>` com anotações de notas musicas. Podemos verificar que no output já não está presente.

Um exemplo de uma possível utilização do autómato é:
`cat <inputFile> | ./play <output1.tex> <output2.tex> ...`

Capítulo 5

Anexos

5.1 Museu da Pessoa — tratamento de fotografias

5.1.1 Filtro de Texto

5.1.2 Estrutura de dados

5.1.3 Cabeçalho ficheiro C

5.2 Processamento de Entidades Nomeadas (Enamex)

5.2.1 Filtro de Texto

5.2.2 Estrutura de dados

5.2.3 Cabeçalho ficheiro C

5.3 Processamento de ficheiros com Canções

5.3.1 Filtro de Texto

5.3.2 Estrutura de dados

```
typedef struct sMusicLine {
    char* line;
    struct sMusicLine* next;
} MusicLine;

typedef struct sMusic {
    char* _Title;
```

```
    char* _From;
    char* _Author;
    char* _Lyrics;
    char* _Music;
    char* _Singer;

    MusicLine* poem;
    MusicLine* poemEnd;
    int error;
} Music;
```

5.3.3 Cabeçalho ficheiro C

5.3.4 Testes

Input teste 1

```
title: Amêndoa Amarga
lyrics: José Carlos Ary dos Santos
music: Alain Oulman
singer: Amália Rodrigues

Port ti falo
e ninguém pensa
mas eu digo
minha amêndoa, meu amigo
meu irmão
meu tropel de ternura
minha casa
meu jardim de carência
minha asa.

Por ti vivo
e ninguém pensa
mas eu sigo
um caminho de silvas
e de nardos
uma intensa ternura
que persigo
rodeada de cardos
por tantos lados.

Por ti morro
e ninguém sabe
mas eu espero
o teu corpo que sabe
a madrugada
o teu corpo que sabe
```

```
a desespero

ó minha amarga amêndoa
desejada.

ó minha amarga amêndoa
desejada.
```

Output teste 1

```
\title{ Amêndoa Amarga}
\author{ José Carlos Ary dos Santos, Alain Oulman }
\documentclass[12pt]{article}
\begin{document}
\maketitle
\section*{Letra}
\begin{center}
\begin{verbatim}
Port ti falo
e ninguém pensa
mas eu digo
minha amêndoa, meu amigo
meu irmão
meu tropel de ternura
minha casa
meu jardim de carência
minha asa.

Por ti vivo
e ninguém pensa
mas eu sigo
um caminho de silvas
e de nardos
uma intensa ternura
que persigo
rodeada de cardos
por tantos lados.

Por ti morro
e ninguém sabe
mas eu espero
o teu corpo que sabe
a madrugada
o teu corpo que sabe
a desespero

ó minha amarga amêndoa
desejada.
```

```
ó minha amarga amêndoa
desejada.
\end{verbatim}
\vspace{5mm}
\hfill  Amália Rodrigues
\end{center}
\end{document}
```

Input teste 2

```
title: Amêndoa Amarga
lyrics: José Carlos Ary dos Santos
music: Alain Oulman
singer: Amália Rodrigues

Port ti falo
e ninguém pensa
mas eu digo
minha amêndoa, meu amigo
meu irmão
meu tropel de ternura
minha casa
meu jardim de carência
minha asa.

Por ti vivo
e ninguém pensa
mas eu sigo
um caminho de silvas
e de nardos
uma intensa ternura
que persigo
rodeada de cardos
por tantos lados.

Por ti morro
e ninguém sabe
mas eu espero
o teu corpo que sabe
a madrugada
o teu corpo que sabe
a desespero

ó minha amarga amêndoa
desejada.

ó minha amarga amêndoa
```

desejada.

Output teste 2

```
\title{ * Tejo que levas as águas}
\author{ Manuel da Fonseca, Adriano Correia de Oliveira}
\documentclass[12pt]{article}
\begin{document}
\maketitle
\section*{Letra}
\begin{center}
\begin{verbatim}
Tejo que levas as águas
correndo de par em par
lava a cidade de mágoas
leva as mágoas para o mar

Lava-a de crimes espantos
de roubos, fomes, terrores,
lava a cidade de quantos
do ódio fingem amores

Leva nas águas as grades
de aço e silêncio forjadas
deixa soltar-se a verdade
das bocas amordaçadas

Lava bancos e empresas
dos comedores de dinheiro
que dos salários de tristeza
arrecadam lucro inteiro

Lava palácios vivendas
casebres bairros da lata
leva negócios e rendas
que a uns farta e a outros mata

Tejo que levas as águas
correndo de par em par
lava a cidade de mágoas
leva as mágoas para o mar

Lava avenidas de vícios
vielas de amores venais
lava albergues e hospícios
cadeias e hospitais

Afoga empenhos favores
```

```
vãs glórias, ocas palmas
leva o poder dos senhores
que compram corpos e almas

Leva nas águas as grades
...

Das camas de amor comprado
desata abraços de lodo
rostos corpos destroçados
lava-os com sal e iodo

Tejo que levas nas águas
...
\end{verbatim}
\vspace{5mm}
\hfill  Adriano Correia de Oliveira
\end{center}
\end{document}
```

Input teste 3

```
title: = Raúl tinha um Ioio
singer: Bando dos Gambozinos
lyrics: Manuel António Pina
music: Suzana Ralha
in: "o beco dos gambozinos"
from: jj

<abc>
X: 1
M: 2/4
K: C
Q: 1/4=60
L: 1/8
dc Ad | dc Ad | dc Ad | GG AA |
w:Ra-ul ti-nhaum i-oi-o que io-io-ia-va to-do o di-a
z/2 D/2E/2F/2 G>G | AG F2 |1 z/2 D/2E/2F/2 G>G | AG F2 :|2 z F/2A/2 GG | FE DD |]
w:quan-doo Ra-úl fa-zia ó-ó o i-o-io a-dor-me-cia
</abc>

Raúl tinha um ioio
que ioioiava todo o dia
quando o Raúl fazia ó-ó
o ioio adormecia

E quando o Raúl chorava
porque o ó-ó não vinha
```

```
o ioio embalava
para baixo e para cima

Raúl dormia e sonhava
e quando sonhava sorria
porque o io-io ioioiava
nos sonhos que Raúl via
```

Output teste 3

```
\title{ = Raúl tinha um Ioio}
\author{ Manuel António Pina, Suzana Ralha}
\documentclass[12pt]{article}
\begin{document}
\maketitle
\section*{Letra}
\begin{center}
\begin{verbatim}
Raúl tinha um ioio
que ioioiava todo o dia
quando o Raúl fazia ó-ó
o ioio adormecia

E quando o Raúl chorava
porque o ó-ó não vinha
o ioio embalava
para baixo e para cima

Raúl dormia e sonhava
e quando sonhava sorria
porque o io-io ioioiava
nos sonhos que Raúl via
\end{verbatim}
\vspace{5mm}
\hfill  Bando dos Gambozinos
\end{center}
\end{document}
```

Amndoa Amarga

Jos Carlos Ary dos Santos, Alain Oulman

April 1, 2015

Letra

```
Port ti falo
e ningum pensa
mas eu digo
minha amndoa, meu amigo
meu irmo
meu tropel de ternura
minha casa
meu jardim de carncia
minha asa.

Por ti vivo
e ningum pensa
mas eu sigo
um caminho de silvas
e de nardos
uma intensa ternura
que persigo
rodeada de cardos
por tantos lados.

Por ti morro
e ningum sabe
mas eu espero
o teu corpo que sabe
a madrugada
```


Figura 5.1: PDF gerado por o ficheiro latex (teste 1). Pagina 1 de 2

Por ti morro
e ningum sabe
mas eu espero
o teu corpo que sabe
a madrugada

o teu corpo que sabe
a desespero

minha amarga amndoa
desejada.

minha amarga amndoa
desejada.

Amlia Rodrigues

Figura 5.2: PDF gerado por o ficheiro latex (teste 1). Pagina 2 de 2

* Tejo que levas as guas

Manuel da Fonseca, Adriano Correia de Oliveira

April 1, 2015

Letra

Tejo que levas as guas
correndo de par em par
lava a cidade de mgoas
leva as mgoas para o mar

Lava-a de crimes espantos
de roubos, fomes, terrores,
lava a cidade de quantos
do dio fingem amores

Leva nas guas as grades
de ao e silncio forjadas
deixa soltar-se a verdade
das bocas amordaadas

Lava bancos e empresas
dos comedores de dinheiro
que dos salrios de tristeza
arrecadam lucro inteiro

Lava palcios vivendas
casebres bairros da lata
leva negcios e rendas
que a uns farta e a outros mata

Figura 5.3: PDF gerado por o ficheiro latex (teste 2). Pagina 1 de 2

Tejo que levas as guas
correndo de par em par
lava a cidade de mgoas
leva as mgoas para o mar

Lava avenidas de vcios
vielas de amores venais
lava albergues e hospcios
cadeias e hospitais

Afoga empenhos favores
vs glrias, ocas palmas
leva o poder dos senhores
que compram corpos e almas

Leva nas guas as grades
...

Das camas de amor comprado
desata abraos de lodo
rostos corpos destroados
lava-os com sal e iodo

Tejo que levas nas guas
...

Adriano Correia de Oliveira

Figura 5.4: PDF gerado por o ficheiro latex (teste 2). Pagina 2 de 2

= Ral tinha um Ioio

Manuel Antnio Pina, Suzana Ralha

April 1, 2015

Letra

```
Ral tinha um ioio
que ioioiava todo o dia
quando o Ral fazia -
o ioio adormecia

E quando o Ral chorava
porque o - no vinha
o ioio embalava
para baixo e para cima

Ral dormia e sonhava
e quando sonhava sorria
porque o io-io ioioiava
nos sonhos que Ral via
```

Bando dos Gambozinos

Figura 5.5: PDF gerado por o ficheiro latex (teste 3)