Universidade do Minho

Trabalho Prático I

Mestrado Integrado em Engenharia Informática

Processamento de Linguagens.

2º Semestre — 2018/2019

a77523 - Alexandre Martins
a74814 - João Bernardo Freitas
a74806 - João Amorim

.

30 de Março 2019
Braga

Resumo

Este projeto tem como objetivo a interação dos alunos com a ferramenta *Flex* permitindo o aumento da capacidade destes relativamente à escrita de expressões regulares para filtrar e transformar textos. Este relatório pretende sumarizar os esforços efetuados para alcançar o objetivo proposto utilizando a ferramenta *Flex*.

Contents

1 Introdução

1.1 Estutura do Relatório

O enunciado foi atribuído de acordo com o menor número de aluno de todos os elementos do grupo. Como tal o relatório debruçar-se-à sobre o enunciado número cinco **Wiki Quotes: provérbios**.

Começaremos por introduzir o problema proposto, passaremos para a concepção da solução e, finalmente, iremos realizar uma análise crítica ao trabalho elaborado.

Na secção enunciada como **Problema Proposto** apresentaremos o problema retratando o formato sobre o qual iremos ter de extrair informação.

Na secção **Solução**, iremos abordar os métodos utilizados na resposta ao enunciado.

Finalmente, na secção **Conclusão** enumeraremos as dificuldades sentidas na resolução do enunciado, mencionando como foram ultrapassadas.

2 Problema Proposto

2.1 Descrição do problema

O nosso problema prático aborda **HTML**, *Hyper Text Markup Language*, que é uma linguagem de marcação utilizada na construção de páginas na Web. O nosso objectivo é utilizar a semântica peculiar que **HTML** proporciona para responder às questões que nos foram propostas.
Foi-nos então apresentado um enunciado com as seguintes questões:

- Criar uma lista citações com respectivo autor;
- Criar uma lista de provérbios;
- Criar uma listagem de provérbios "adulterados" e seu original;
- Apresentar estatísticas relativas ao trabalho efectuado.

Para a resolução destas questões foi nos fornecido, pelos docentes, o seguinte ficheiro:
ptwikiquote-20190301-pages-articles.xml.bz2.

3 Solução

3.1 Resposta às questões do enunciado

- **Alínea 1)**

Para a solução da *alínea 1)*, é necessário retirar todas as citações presentes no ficheiro *ptwikiquote-20190301-pages-articles.xml* com o seu respectivo autor, para tal, começamos por determinar como é que as citações estão identificadas. Após leitura, verificamos que as citações estão definidas,inicialmente, por:

```
*&quot;
- &quot;
* ''&quot;
&quot;
```

E terminam com

```
&quot;
```

Como tal, tivemos de desenvolver quatro expressões regulares de forma a extrair todas as citações e seus conteúdos.

```
^(\*)(&quot;).+(&quot;)
^(\-[ ])(&quot;).+(&quot;)
^(\*[ ]\'+)(&quot;).+(&quot;)
^(\*[ ])(&quot;).+(&quot;)
```

Após extração das citações reparamos que era necessário a remoção dos caracteres correspondentes ao início e fim de uma citação, dependendo do tipo.

```
^(\*)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+7);printf("%s \n ", quote);}
^(\-[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);printf("%s \n ", quote);}
^(\*[ ]\'+)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+10);printf("%s \n ", quote);}
^(\*[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);printf("%s \n ", quote);}
```

Tendo em conta os requisitos das próximas alíneas e a falta de autores das citações decidimos definir várias **Start Conditions** para facilitar a extração do nome dos autores e provérbios.

Para esta alínea necessitávamos de apenas duas **start conditions**, *META* e *QUOTE*.
META é a **start condition** inicial, que começa sempre que é encontrado o identificador de inicio de página, sendo que o objetivo desta é indicar se a página é para ignorar, para citações ou para provérbios:

```
\<page\>
```

Definimos então a entrada na *META* como:

```
\<page\> {BEGIN META;}
```

De seguida, para extrair apenas citações, que definir que estas estão contidas em páginas cujo o título inclui pelo menos duas palavras começadas por maiúscula.
Ficamos então com a seguinte definição:

```
<META>\<title\>([A-Z][a-z ]+){2,}\<\/title\>
    {yytext[yyleng-8]='\0';autor=strdup(yytext+7);BEGIN QUOTE;}
```

Para ignorar tudo o que não era pertinente para a resolução do enunciado decidimos que sempre que é encontrado o identificador de final de página deve voltar à **start condition** inicial, ignorando todos os caracteres que não fazem parte das citações.
Para tal definimos as seguintes expressões regulares globais:

```
<*>\<\/page\> {BEGIN INITIAL;}
<*>(.|\n) {;}
```

Como, de acordo com enunciado, é necessário fornecer o nome do autor e a respectiva citação decidimos guardar o nome deste na string *autor* e a citação na string *quote*.

Ficamos então com o seguinte algoritmo:

```
    %{
#include <stdio.h>
#include <string.h>

char* autor="";
char* quote="";
%}

%x META
%x QUOTE

%%
\<page\> {BEGIN META;}

<META>\<title\>([A-Z][a-z ]+){2,}\<\/title\>
    {yytext[yyleng-8]='\0';autor=strdup(yytext+7);BEGIN QUOTE;}


<QUOTE>^(\*)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+7);printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\-[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\*[ ]\'+)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+10);printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\*[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);printf("%s:%s \n ", autor, quote);}


<*>\<\/page\> {BEGIN INITIAL;}
<*>(.|\n) {;}
%%


int yywrap(){
   return 1;
}

int main(){
   printf("Inicio da Filtragem\n");
   yylex();
   printf("\nFim da Filtragem\n");
   return 0;
}
```

- **Alínea 2)**

Para a solução da *alínea 2)* era necessário retirar todos os provérbios, para isso, decidimos expandir as **start conditions** de forma a extrair os provérbios. De acordo com o enunciado só devíamos incluir provérbios que estavam contidos em páginas cujo titulo começava por "Provérbios".
Para isso definimos a start condition *PROVERBIO*

```
<META>\<title\>(Proverbio).*\<\/title\> {BEGIN PROVERBIO;}
```

Para a extracção dos provérbios tivemos que, mais uma vez, determinar como é que os provérbios estão identificados. Após analisar novamente o documento, verificamos que todos os provérbios estão definidos de forma semelhante às citações, isto é:

```
*&quot;
* ''&quot;
* &quot;
```

E tal como nas citações terminavam por:

```
&quot;
```

Logo desenvolvemos as seguintes expressões regulares:

```
<PROVERBIO>^(\*)(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+7);printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*[ ]\'+)(&quot;).*(&quot;) {yytext[yyleng-6]='\0';
    quote = strdup(yytext+10);printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*[ ])(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote
    = strdup(yytext+8);printf("PROVERBIO:%s \n ", quote);}
```

- **Alínea 3)**

Para a resolução da *alínea 3)* é necessário extrair todos os provérbios adulterados, para isso, foi necessário a criação de uma última **start condition** à qual chamamos *ADULTERADO*.
Após leitura do ficheiro reparamos que todos os provérbios adulterados eram precedidos pela seguinte linha:

```
** ''' Adulterao :'''
```

Sendo que o início destes estava definido por:

```
*** &quot;
```

E, tal como nas ultimas duas alíneas, acabava por:

```
(&quot;)
```

Para além disso reparamos que a lista de adulterados acabava sempre quando no inicio de uma linha era:

```
* &quot;
```

Que indica um provérbio novo.

Definimos então que os provérbios adulterados estão dentro de páginas de provérbios logo criamos a seguinte expressão regular:

```
<PROVERBIO>^(\*+[ ])(\'{3}Adulterao\:\'{3}) {BEGIN ADULTERADO;}
```

Para extrair os adulterados e voltar á **start condition** *PROVERBIO* criamos duas expressões regulares:

```
<ADULTERADO>^(\*{3}[ ])(&quot;).*(&quot;) {yytext[yyleng-6]='\0';
    quote = strdup(yytext+10);printf("ADULTERADO:%s \n ", quote);}
<ADULTERADO>^(\*[ ])(&quot;) {BEGIN PROVERBIO;}
```

- **Alínea 4)**

Para esta alínea decidimos que era necessário a adição de vários contadores, nomeadamente, o número de páginas, citações, provérbios, provérbios adulterados e autores lidos.

```
    int a=0; //autores
    int q=0; //quotes
    int p=0; //proverbios
    int ad=0; //adulterados
    int pa=0; //paginas
```

Alterando as linhas acima de forma a incrementar estes contadores temos:

```
    \<page\> {pa++;BEGIN META;}

<META>\<title\>([A-Z][a-z ]+){2,}\<\/title\>
    {yytext[yyleng-8]='\0';autor=strdup(yytext+7);a++;BEGIN QUOTE;}
<META>\<title\>(Provrbio).*\<\/title\> {BEGIN PROVERBIO;}


<QUOTE>^(\*)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+7);q++;printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\-[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);q++;printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\*[ ]\'+)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+10);q++;printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\*[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);q++;printf("%s:%s \n ", autor, quote);}

<PROVERBIO>^(\*)(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+7);p++;printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*[ ]\'+)(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+10);p++;printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*[ ])(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);p++;printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*+[ ])(\'{3}Adulterao\:\'{3}) {BEGIN ADULTERADO;}

<ADULTERADO>^(\*{3}[ ])(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote
    = strdup(yytext+10);ad++;printf("ADULTERADO:%s \n ", quote);}
<ADULTERADO>^(\*[ ])(&quot;) {BEGIN PROVERBIO;}
```

```
int main(){
   printf("Inicio da Filtragem\n");
   yylex();
   printf("Numero total de autores processados:%d \n",a);
   printf("Numero total de citacoes processadas:%d \n",q);
   printf("Numero total de proverbios processados:%d \n",p);
   printf("Numero total de proverbios adulterados processados:%d \n",ad);
   printf("Numero total de paginas processadas:%d \n",pa);
   printf("\nFim da Filtragem\n");
   return 0;
}
```

3.2 Opcional

Após a filtragem do ficheiro *ptwikiquote-20190301- pages-articles.xml* reparamos que existiam links e parênteses retos que estavam a mais, para os remover decidimos criar um segundo filtro que tem como único objetivo "limpar" o output do filtro original.

A utilização deste segundo filtro é puramente opcional, sendo que, este fica definido por quatro expressões regulares.

Expressão regular que remove os links.

```
(&lt;ref&gt;).*(&lt;\/ref&gt;)  {}
```

Expressão regular que remove os parênteses rectos.

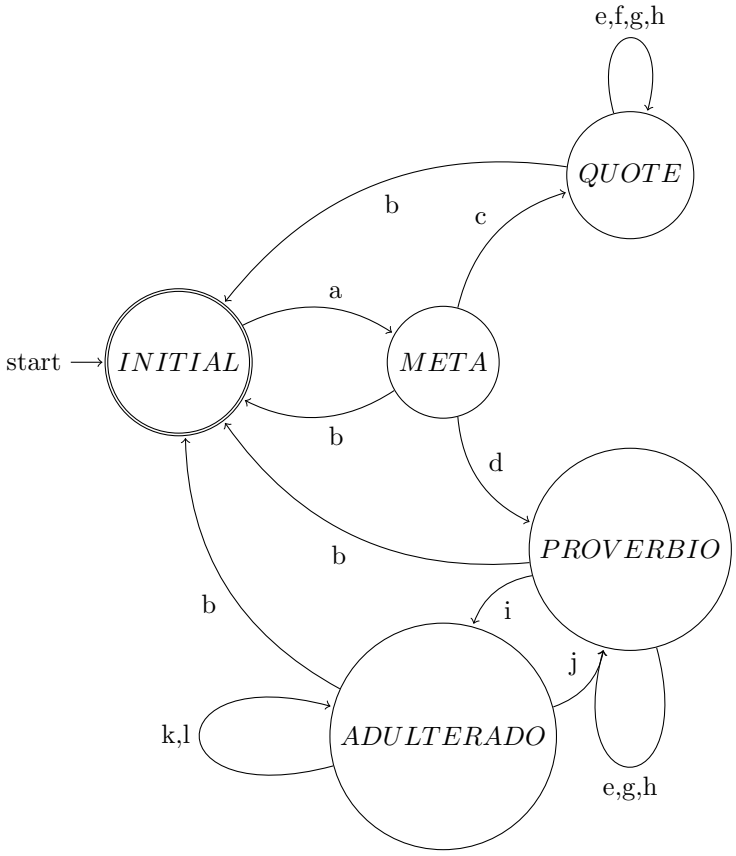
```
(\[)                {}
(\])                {}
```

Expressão regular que remove `&quot` presentes no meio de uma citação.

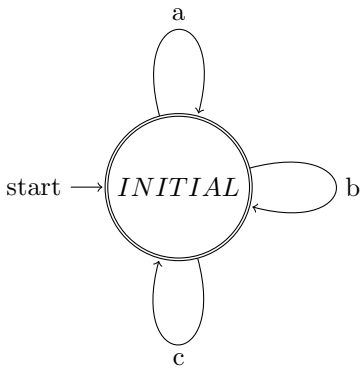
```
(&quot[.;:])           {}
```

3.3 Autómato

3.3.1 TP1



3.3.2 OPCIONAL



4 Conclusão

Visto que o objectivo deste trabalho era aumentar contacto com a ferramenta *Flex*, as is comoma capacidade de escrever *Expressões Regulares* e desenvolver *Processadores de Linguagens Regulares* podemos afirmar que foi um sucesso.
Porém, no decorrer da resolução deste trabalho, foram surgindo alguns desafios que foram ultrapassados, nomeadamente: encontrar todos os identificadores das citações/provérbios, como separar as citações dos provérbios, encontrar o padrão dos provérbios adulterados e definir as respetivas expressões regulares sendo que o maior obstáculo foi descobrir como efetuar a "limpeza" das citações.
Em suma, o trabalho foi bastante benéfico para todos os elementos, sendo que aprofundou o entendimento das expressões regulares e pressupostos da ferramenta *Flex*.

5 Código

5.1 TP1

Código correspondente à solução das quatro alíneas.

```
 %{
#include <stdio.h>
#include <string.h>

char* autor="";
char* quote="";
int a=0;//autores
int q=0;//quotes
int p=0;//proverbios
int ad=0;//adulterados
int pa=0;//paginas

%}

%x META
%x QUOTE
%x PROVERBIO
%x ADULTERADO

%%
\<page\> {pa++;BEGIN META;}

<META>\<title\>([A-Z][a-z ]+){2,}\<\/title\>
    {yytext[yyleng-8]='\0';autor=strdup(yytext+7);a++;BEGIN QUOTE;}
<META>\<title\>(Provrbio).*\<\/title\> {BEGIN PROVERBIO;}


<QUOTE>^(\*)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+7);q++;printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\-[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);q++;printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\*[ ]\'+)(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+10);q++;printf("%s:%s \n ", autor, quote);}
<QUOTE>^(\*[ ])(&quot;).+(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);q++;printf("%s:%s \n ", autor, quote);}

<PROVERBIO>^(\*)(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+7);p++;printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*[ ]\'+)(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+10);p++;printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*[ ])(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote =
    strdup(yytext+8);p++;printf("PROVERBIO:%s \n ", quote);}
<PROVERBIO>^(\*+[ ])(\'{3}Adulterao\:\'{3}) {BEGIN ADULTERADO;}
```

```
<ADULTERADO>^(\*{3}[ ])(&quot;).*(&quot;) {yytext[yyleng-6]='\0'; quote
    = strdup(yytext+10);ad++;printf("ADULTERADO:%s \n ", quote);}
<ADULTERADO>^(\*[ ])(&quot;) {BEGIN PROVERBIO;}

<*>\<\/page\> {BEGIN INITIAL;}
<*>(.|\n) {;}
%%


int yywrap(){
   return 1;
}

int main(){
   printf("Inicio da Filtragem\n");
   yylex();
   printf("Nmero total de autores processados:%d \n",a);
   printf("Nmero total de  citaes  processadas:%d \n",q);
   printf("Nmero total de provrbios processados:%d \n",p);
   printf("Nmero total de provrbios adulterados processados:%d \n",ad);
   printf("Nmero total de pginas processadas:%d \n",pa);
   printf("\nFim da Filtragem\n");
   return 0;
}
```

5.2 Opcional

Código correspondente ao filtro opcional.

```
%{
#include <stdio.h>
#include <string.h>


%}

%%

(&lt;ref&gt;).*(&lt;\/ref&gt;)  {}
(\[)                    {}
(\])                    {}
(&quot[.;:])              {}
%%


int yywrap(){
   return 1;
}

int main(){
   yylex();
   return 0;
}
```

5.3 MakeFile

Para agilizar o processo de compilação de ambos os filtros criamos o seguinte MakeFile.

```
inicial:
   lex -o parte1.c TP1.l
   lex -o parte2.c TP1_2.l
   gcc -o tp1 parte1.c
   gcc -o limpa parte2.c
   rm parte1.c
   rm parte2.c
```

5.4 Comando

Para fazer com que os dois filtros sejam utilizados utilizamos o seguinte comando.

```
    ./tp1 <ptwikiquote-20190301-pages-articles.xml | ./limpa
```

Anexos

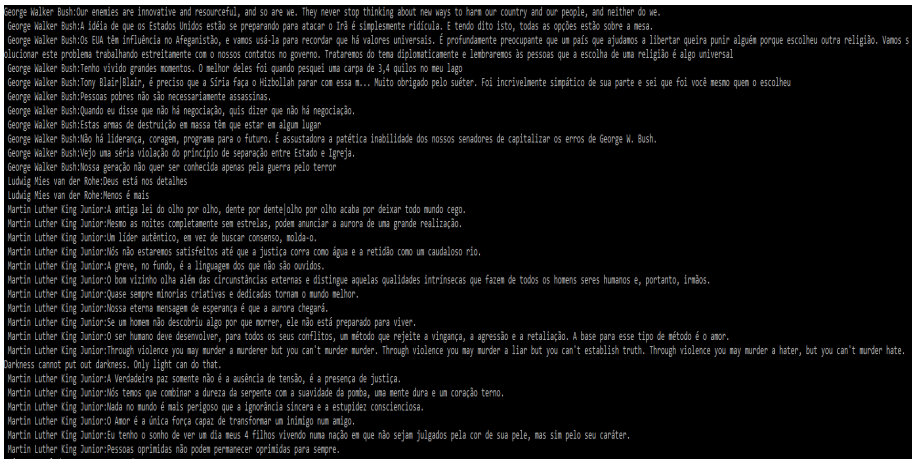


Figure 1: Exemplo do output de citações

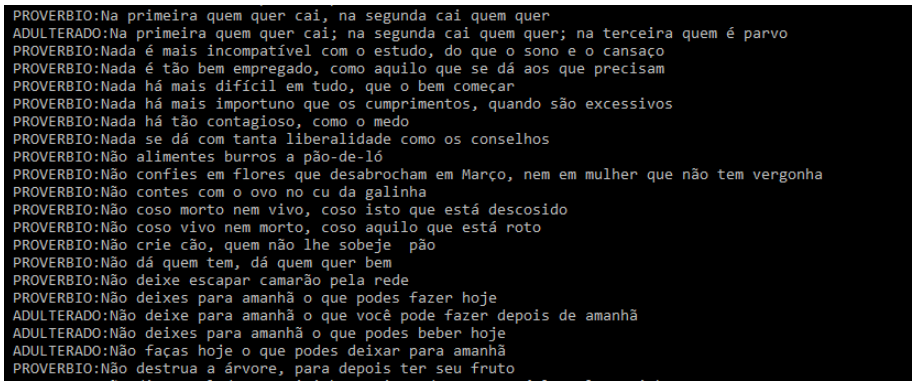


Figure 2: Exemplo do output de provérbios e provérbios adulterados

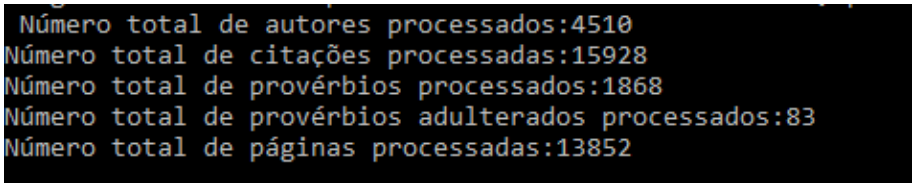


Figure 3: Estatisticas finais