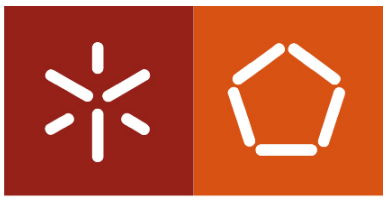

Escola de Engenharia

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Processamento de Linguagens

Trabalho Prático nº 1

Octávio Maia a71369
João Silva a72023
Rui Freitas a72399

5 de Abril de 2016

Conteúdo

Lista de Figuras

Resumo

Este relatório tem como objetivo descrever o processo de desenvolvimento, estruturação e tomadas de decisão resultantes da primeira fase do trabalho prático da Unidade Curricular de Processamento de Linguagens.

Nesta fase, foi decidido abordar o tema sobre o *Normalizador de ficheiros BibTeX*, cujo produto final consiste num interpretador de ficheiros *.bib*.

Palavras chave: BibTeX, HTML, FLEX, Estruturas dados, Expressões regulares.

Capítulo 1

Introdução

De modo a por em prática os conhecimentos adquiridos nas aulas da Unidade Curricular de Processamento de Linguagens, utilizamos a ferramenta FLEX[1], cuja utilização foi coberta extensivamente em ambas as aulas teóricas, bem como as práticas.

Neste relatório iremos expor os nossos métodos de desenvolvimento, bem como as decisões tomadas para concretizar a realização deste trabalho.

1.1 Descrição do Problema e Implementação

Após um debate entre os elementos do grupo sobre qual tema a desenvolver, ficou decidido desenvolver o *Normalizador de ficheiros BibTeX*[2], uma ferramenta de formatação utilizada em documentos LaTeX[3].

Este tem como objetivo o desenvolvimento o desenvolvimento de uma ferramenta que tenha como capacidade a análise de documentos *BibTeX* e a contagem das suas categorias, cuja contagem deverá ser exportada para um ficheiro *HTML*[4], bem como a normalização de texto e a construção de um grafo.

A nossa primeira tarefa teve como base a definição das várias expressões regulares que irão ser utilizadas no âmbito desta primeira fase do trabalho, de forma a filtrar as diversas informações dos ficheiros *.bib* fornecidos.

Após a finalização das expressões regulares tivemos como objetivo o desenvolvimento do código de forma a responder a todas as tarefas fornecidas.

[1]The Fast Lexical Analyzer, http://flex.sourceforge.net/
[2]http://www.bibtex.org/
[3]https://latex-project.org/intro.html
[4]HyperText Markup Language, https://www.w3.org/html/

Capítulo 2

Processamento do BibTeX

2.1 Contagem de categorias *BibTeX*

Nesta primeira alínea é pedido a criação de um ficheiro HTML que contenha a contagem das várias categorias presentes no documento *.bib* fornecido.

2.1.1 Expressões Regulares

As expressões regulares utilizadas nesta fase do exercício foram desenvolvidas para apenas trabalharmos com palavras reservadas do *BibTeX*, ou seja, todas a palavras que são antecedidas pelo caractere @ quando este se encontra no início da linha.

Além disso o grupo decidiu expandir a expressão regular para o caso do @ não ser o primeiro caractere da linha, mas poder ser antecedido por qualquer quantidade de espaços.

Uma vez que queríamos que todas as restantes expressões fossem ignoradas, inserimos a última expressão regular apresentada de forma a contabilizar todos esses casos.

Para contar com o descrito necessitamos de criar o estado de *reservada* onde apenas neste estado procuramos pela primeira palavra uma vez que anteriormente encontramos @ na forma descrita anteriormente, ou seja agora estamos perante uma "tag" do *BibTeX* que deveremos guardar.

```
%x reservada
%%
^[ ]*@ {BEGIN(reservada);}
<reservada>[A-Za-z]+  {addRegisto(yytext);BEGIN(INITIAL);}
.|\n  {;}
```

2.1.2 Estrutura de Dados

De forma a guardar a informação relativa a cada "tag" reservada bem como a quantidade de vezes que esta aparece no ficheiro de *input*, utilizamos uma simples estrutura com 2 "Arrays" em que num guardava a *string* correspondente à "tag" e no outro seria guardado o contador com o numero de vezes que esta aparecia no ficheiro, funcionado assim como um *Hash(String,Int)*.

Foram consideradas a possível existência de 30 "tags" diferente, sendo assim alocadas no inicio do programa estaticamente 30 posições nos dois *arrrays*.

2.1.3 Ações executadas

A quando da inicialização do programa as estruturas de dados apresentadas em 2.1.2 de modo garantir a fiabilidade do programa.

Quando é encontrado um *match* na 1º expressão regular responsável declaramos que entramos no estado de palavra reservada.

Uma vez neste estado quando encontramos a 1ª palavra essa será a palavra reservada que queremos do *BibTeX*, ou seja adicionamos-la à estrutura de dados para isso recorrendo à função desenvolvida *addRegisto(char* nome)*.

O funcionamento da função *addRegisto* baseia-se em procurar no 1º *array* por uma *string* igual à que é passada, caso isso ocorra na posição correspondente do *array* com a contagem correspondente è adicionado 1, caso não seja encontrado nenhum "match" adicionamos à primeira posição livre o novo registo.

```
void addRegisto(char* nome){
    int found=0;
    int i;
    for(i=0;i<30 && nomes[i]!=NULL && found==0;i++){
        if(!strcmp(nomes[i],nome)){
            found=1;
            cont[i]++;
        }
    }
    if(nomes[i]==NULL && found==0){
        nomes[i] = strdup(nome);
        cont[i]++;
    }
}
```

Após o término do processamento do ficheiro de *input* é criado um ficheiro *index.html* que contem toda a informação presente na estrutura de dados referida no ficheiro de *input* em forma de lista, possível de ver num *browser*.

2.1.4 Execução do programa

De modo a executar o programa mais facilmente, procedemos à criação de uma simplex *makefile*, de modo a poupar repetições de comandos *(flex, gcc e execução)*.

```
flex tp1.l
gcc -o tp1 lex.yy.c
./tp1 < lib.bib
more index.html
<html>
        <head>
                <title>EXERCICIO 1</title>
        </head>
        <body>
                <h2>Lista Exercico 1</h2>
                        <ul>
                        <li> string -> 31 </li>
                        <li> techreport -> 140 </li>
                        <li> inbook -> 3 </li>
                        <li> book -> 47 </li>
                        <li> phdthesis -> 21 </li>
                        <li> article -> 142 </li>
                        <li> inproceedings -> 209 </li>
                        <li> unpublished -> 15 </li>
                        <li> incollection -> 6 </li>
                        <li> manual -> 13 </li>
                        <li> misc -> 61 </li>
                        <li> proceedings -> 4 </li>
                        <li> mastersthesis -> 2 </li>
                        <li> proceeding -> 1 </li>
                        </ul>
        </body>
</html>
```

Figura 2.1: Execução da primeira alínea do projeto e respetivo output.

Lista Exercico 1

- string -> 31
- techreport -> 140
- inbook -> 3
- book -> 47
- phdthesis -> 21
- article -> 142
- inproceedings -> 209
- unpublished -> 15
- incollection -> 6
- manual -> 13
- misc -> 61
- proceedings -> 4
- mastersthesis -> 2
- proceeding -> 1

Figura 2.2: Página HTML gerada anteriormente.

Através da página HTML gerada podemos verificar as diversas categorias presentes no ficheiro *BibTeX* fornecido, bem como as vezes que estas aparecem.

2.2 Ferramenta de normalização

Na segunda alínea é pedido o desenvolvimento de uma ferramenta de normalização e a capacidade desta fazer *pretty-priting*.

2.2.1 Expressões Regulares

As expressões regulares utilizadas na segunda fase do exercício foram desenvolvidas de forma a coletar informação relativa aos autores de um dado livro, bem como o título do mesmo.

Foram então criados os seguintes campos para guardar informação:

- *char* a[500]* cujo objetivo é o armazenamento dos vários nomes que constituem um nome de um autor.
- *char lixo[5000]* cujo objetivo é guardar a informação não desejada, ou seja, toda a informação exclusive os campos *author* e *title*
- *int titleBOOL=0, authorBOOL=0* cujo objetivo é servir de *flag* para o caso de o campo *title* e o campo *author* terem sido encontrados.

De forma a representar a informação referenciada anteriormente, foi criado o estado *author*, a *categoria* e o *title*.

```
@[A-Za-z]+\{[A-Za-z0-9\:\+]+,
<categoria>[Aa][Uu][Tt][Hh][Oo][Rr][ \t]*=[ \t]*[\"\{]
<author>[ ]+[Aa][Nn][Dd][ ]+
<author>[ ]+[Aa][Nn][Dd][ ]*\n?[ ]*
<author>[ ]*\n[ ]*[Aa][Nn][Dd][ ]+
<author>\{[^ ,]+[^ \"\}]\}
<author>[^ ,]+([^ \"\},\n]|[^ \"\}],)
<author>[ ]*[\"\}],?\n
<author>.|\n
<categoria>[ \n\t][Tt][Ii][Tt][Ll][Ee][ \t]*=[ \t]*\"
<categoria>[ \n\t][Tt][Ii][Tt][Ll][Ee][ \t]*=[ \t]*\{
<title>[^=]*\",?
<title>[^=]*\},?
<categoria>[ ]*=[ ]*\"
<categoria>\",
<categoria>.|\n
<categoria>\}\n
.|\n
```

2.2.2 Estrutura de Dados

Nesta fase sendo mais complexa foram criados os seguintes campos para guardar informação para auxiliar a execução pretendida

- *char* a[500]* cujo objetivo é o armazenamento dos vários nomes que constituem os nomes dos autores.
- *int n* é a variável auxiliar que permite contar o número de nomes que um autor tem, por exemplo para "Jose Antonio" o valor de n seria 2.
- *int numeroAutores* variável que guarda o numero total de autores que participaram num livro.
- *int numeroNomesAutores[100]* é o *array* onde é guardado o numero de nomes correspondentes aos autores em a, de modo a iterar de nome completo em nome completo.
- *int bool=0* São guardados os nomes normalizados dos autores de uma publicação.

2.2.3 Ações executadas

- `@[A-Za-z]+\{[A-Za-z0-9:\+]+`

  Expressão regular que marca o inicio de um registo de um dado bibliográfico, onde se encontram os campos dos registos que queremos processar.

- `<categoria>[Aa][Uu][Tt][Hh][Oo][Rr][ \t]*=[ \t]*[\"\{]`

  Expressão regular que marca o inicio de um registo de um conjunto de autores, para isso faz reset as variáveis e estruturas de dados para este novo registo de autores.

- `<author>[ ]+[Aa][Nn][Dd][ ]+`
  `<author>[ ]+[Aa][Nn][Dd][ ]*\n?[ ]*`
  `<author>[ ]*\n[ ]*[Aa][Nn][Dd][ ]+`

  Expressões regulares que marcam a delimitação dos nomes dos distintos autores de uma obra, registando no array que representa o numero de nomes que o autor em analise tem. Indicamos também que a obra tem mais que um autor, e o numero de nomes do próximo autor será inicializado a 0.

- `<author>\{[^ ,]+[^ \"\}]\}`
  `<author>[^ ,]+([^ \"\},\n]|[^ \"\}],)`
  `<author>[ ]*[\"\}],?`

  Expressões regulares que marcam o registo de um nome presente num autor, guardando-os no array auxiliar a onde os nomes de todos os autores são guardados.

- `<author>[ ]*[\"\}],?`

  Expressões regulares onde encontramos o final do campo correspondente aos autores da obra em análise. Após passamos para o contexto de *categoria* e depois registamos no ficheiro do *grafo* todas as ligações presentes na obra em analise, para isso recorremos à função *void registaAuthors()*.

- `<categoria>[ \n\t][Tt][Ii][Tt][Ll][Ee][ \t]*=[ \t]*\"`
  `<categoria>[ \n\t][Tt][Ii][Tt][Ll][Ee][ \t]*=[ \t]*\{`

  Expressão regular que marca o início do registo de um titulo, iniciando assim um novo contexto.

2.2.4 Execução do programa

De modo a executar o programa mais facilmente, procedemos à criação de uma simplex *makefile*, de modo a poupar repetições de comandos *(flex, gcc e execução)*.

```
@Misc{CH05a,
author = { D. Cruz and
        P. Henriques },
 title = {LISS - Language of Integers, Sequences and Sets},
    howpublished = {Talk to the gEPL, Dep. Inform\'atica / Univ. Minho},
    year = 2005,
    month = {Oct.},
    annote = {compilacao, ga, geracao codigo, VM"
    }
```

Figura 2.3: Output da segunda alínea.

2.3 Construção de grafo das ligações presentes no ficheiro e de um autor

Na última alínea é pedida a construção de um grafo para um determinado autor, que mostre todos os autores que publicaram artigos com esse autor, recorrendo ao *GraphViz2* e à linguagem *Dot.*

2.3.1 Expressões Regulares

```
@[A-Za-z]+\{[A-Za-z0-9:\+]+,
<categoria>[Aa][Uu][Tt][Hh][Oo][Rr][ \t]*=[ \t]*[\"\{]
<author>[ ]+[Aa][Nn][Dd][ ]+
<author>[ ]+[Aa][Nn][Dd][ ]*\n?[ ]*
<author>[ ]*\n[ ]*[Aa][Nn][Dd][ ]+
<author>\{[^ ,]+[^ \"\}]\}
<author>[^ ,]+([^ \"\},\n]|[^ \"\}],)
<author>[ ]*[\"\}],?
<author>.|\n
<categoria>.|\n
<categoria>\}\n
.|\n
```

2.3.2 Estrutura de Dados

Nesta fase foram usadas as variáveis disponíveis na alínea B bem como as variáveis seguintes:

- *int nParceiros* cujo objetivo é guardar a informação relativa ao numero de parceiros do autor que é passado como autor a procurar.
- *authorParceria[100]* cujo objetivo é guardar os nomes de dos autores que participaram em alguma obra com o autor em causa.

2.3.3 Ações executadas

- `@[A-Za-z]+\{[A-Za-z0-9:\+]+`

  Expressão regular que marca o inicio de um registo de um dado bibliográfico, onde se encontram os campos dos registos que queremos processar.

- `<categoria>[Aa][Uu][Tt][Hh][Oo][Rr][ \t]*=[ \t]*[\"\{]`

  Expressão regular que marca o início de um registo de um conjunto de autores, para isso faz reset as variáveis e estruturas de dados para este novo registo de autores.

- `<author>[ ]+[Aa][Nn][Dd][ ]+`
  `<author>[ ]+[Aa][Nn][Dd][ ]*\n?[ ]*`

`<author>[ ]*\n[ ]*[Aa][Nn][Dd][ ]+`

Expressões regulares que marcam a delimitação dos nomes dos distintos autores de uma obra, registando no array que representa o numero de nomes que o autor em analise tem. Indicamos também que a obra tem mais que um autor, e o numero de nomes do próximo autor será inicializado a 0.

- ```
  <author>\{[^ ,]+[^ \"\}]\}
  <author>[^ ,]+([^ \"\},\n]|[^ \"\}],)
  <author>[ ]*[\"\}],?
  ```

  Expressões regulares que marcam o registo de um nome presente num autor guardando-os no array auxiliar *a* onde os nomes de todos os autores são guardados.

- `<author>[ ]*[\"\}],?`

  Expressões regulares onde encontramos o final do campo correspondente aos autores da obra em análise. Após passamos para o contexto de *categoria* e depois registamos no ficheiro do *grafo* todas as ligações presentes na obra em analise, para isso recorremos à função *void registaAuthors()*.

2.3.4 Execução do programa

A execução deste programa vai gerar dois ficheiros *.dot*. Um deles com todas as ligações presente, e outro apenas com as ligações dos autores escolhidos.

Estes ficheiros serão depois processados pelo programa *dot* de modo a obtermos uma representação visual do grafo das ligações presentes no ficheiro *.bib*.

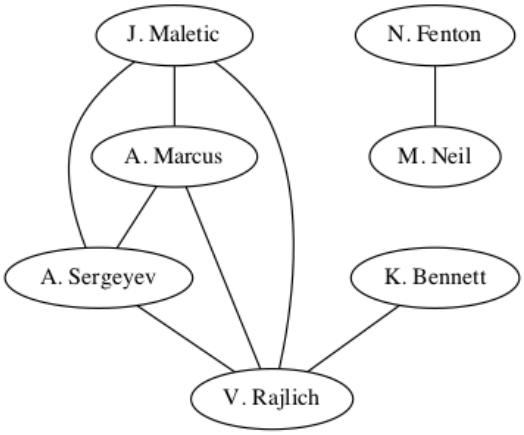

Figura 2.4: Excerto do grafo produzido pela terceira alínea.
  ```

Capítulo 3

Testes realizados

3.1 Alínea A

Para a realização desta alínea realizaram-se várias tentativas até encontrar a expressão regular que permite o correto processamento de texto.

De modo a facilitar a execução do programa criou-se uma *makefile* que faz a compilação do código, bem como a execução do mesmo.

```
1 p1:
2         flex tp1.l
3         gcc -o tp1 lex.yy.c
4         ./tp1 < lib.bib
5         more index.html
```

A página HTML resultante desta invocação pode ser verificada na imagem 2.2

3.2 Alínea B

Para a realização desta alínea foi criada a *makefile* proposta a seguir que faz a compilação do código, bem como a execução do mesmo.

```
1 p2:
2         flex tp2.l
3         gcc -o tp2 lex.yy.c
4         ./tp2 < lib.bib
```

Após a execução deste comando, será impresso no terminal o resultado do *pretty-printing* resultante.

3.3 Alínea C

Para a realização desta alínea foi criada a seguinte *makefile* que faz a remoção dos grafos já existentes, caso existam.

Após remover os ficheiros faz a compilação do código

```
p3:
        if [ -f "graf.dot" ]; then rm graf.dot; fi;
        if [ -f "grafAuthor.dot" ]; then rm grafAuthor.dot; fi;
        flex tp3.l
        gcc -o tp3 lex.yy.c
        ./tp3 ${ARGS} < lib.bib
```

De modo a produzir o grafo relativo a um autor em específico, neste exemplo será o autor P. Henriques, deverá ser executado o comando *make p3 ARGS="P. Henriques"*.

Caso executemos apenas *make p3* irá ser produzido um grafo geral, ou seja, irá representar as relações de todos os autores.

```
graph:
        if [ -f "graf.dot" ]; then dot -Tpng graf.dot -o graf.png;
           fi;
        if [ -f "grafAuthor.dot" ]; then dot -Tpng grafAuthor.dot -
           o grafAuthor.png; fi;
```

Após correr o comando mencionado anteriormente (*make p3* ou *make p3 ARGS="..."*) podemos executar o comando *make graph* de forma a gerar a imagem em formato PNG.

Capítulo 4

Conclusão

Neste projeto, o principal objetivo consistiu na aplicação da linguagem *FLex*, e a construção de *ER's* funcionais e eficazes para o propósito das mesmas.

Este conhecimento foi posto em prática, recorrendo ao tema posposto sobre *BibTeX*, sendo dividido em 3 partes uma de identificação de expressões reservadas, uma segunda parte de normalização do ficheiro *BibTex* e por fim a construção de um grafo em *Dot* para representar as ligações dos autores em grafos. Isto levou o grupo a realizar um estudo prévio sobre o que iríamos abordar, de forma a garantir que tudo fosse efetuado da forma correta.

Após o início do projeto, podemos afirmar que este fluiu de forma natural graças aos diversos conhecimentos adquiridos em ambas aulas práticas, bem como nas teóricas.

Devido a este grau de familiaridade com a linguagem *C* e a linguagem de programação onde o programa *Flex* se encontra embebido, podemos afirmar que desenvolvemos um projeto que cumpre os requerimentos propostos e cuja implementação foi feita de forma correta.

Assim sendo, o grupo foi capaz de atingir os objetivos propostos, o que nos deixou satisfeitos com o produto final desenvolvido.

Capítulo 5

Anexos

5.1 Ficheiro BibTeX

Para a realização deste projeto, utilizamos o ficheiro *BibTeX* disponibilizado no endereço `http://di.uminho.pt/~prh/lp.bib`

Este ficheiro irá seguir a tipologia proposta pelo seguinte excerto do ficheiro *.bib*:

```
@techreport{BW83a,
    author = "Manfred Broy and Martin Wirsing",
    title = "Generalized Heterogeneous Algebras and Partial
       Interpretations",
    year = 1983,
    month = Feb,
    institution = "Institut fur Informatik, TUM",
    note = "(draft version)",
    annote = "espec algebrica"
    }

@inbook{Val90a,
    author = "Jos\'e M. Valen\c{c}a",
    title = "Processos, {O}bjectos e {C}omunica\c{c}\~ao
                ({O}p\c{c}\~ao I - {MCC})",
    chapter = 2,
    year = 1990,
    month = Oct,
    publisher = gdcc,
    address = um,
    annote = "programacao oobjectos, proc comunicantes, espec
       formal"
    }
```

5.2 Código Flex

5.2.1 Alínea A

De seguida apresentamos uma cópia do programa do ficheiro *tp1.l* desenvolvido em *Flex*, de modo a responder à alínea A.

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

char*  nomes[30];
int cont[30];

char* lowerCASE(char * str){
        int i;
        for(i = 0; str[i]!='\0'; i++){
                str[i] = tolower(str[i]);
        }
        return str;
}

void addRegisto(char* nome){
        int found=0;
        int i;
        for(i=0;i<30 && nomes[i]!=NULL && found==0;i++){
                if(!strcmp(nomes[i],nome)){
                        found=1;
                        cont[i]++;
                }
        }
        if(nomes[i]==NULL && found==0){
                nomes[i] = strdup(nome);
                cont[i]++;
        }
}

%}
%x reservada
%%
^[ ]*@          {BEGIN(reservada);}
<reservada>[A-Za-z]+   {char* aux = lowerCASE(yytext);addRegisto(aux);BEGIN(INITIAL);}
.|\n    {;}

%%
int yywrap() {return 1;}
int main(){
        int i;
        for(i=0;i<30;i++){
                nomes[i]=NULL;
                cont[i]=0;
        }
        yylex();

        FILE* fp =fopen("index.html","w");

        fprintf(fp, "<html>\n\t<head>\n\t\t<title>EXERCICIO 1</title>\n\t</head>\n\t<body>\n \t\t<h2
             >Lista Exercico 1</h2>\n\t\t\t<ul>\n");

        for(i=0;i<30 && nomes[i]!=NULL ;i++){

                fprintf(fp, "\t\t\t<li> %s -> %d </li>\n", nomes[i],cont[i]);
        }

        fprintf(fp, "\t\t\t</ul>\n\t</body>\n</html>\n");
        fclose(fp);

        return 0;
}
```

5.2.2 Alínea B

De seguida apresentamos uma cópia do programa do ficheiro *tp2.l* desenvolvido em *Flex*, de modo a responder à alínea B.

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


char* a[500];
int n=0;
int tot=0;
int numeroAutores=0;
int numeroNomesAutores[100];
char lixo[5000];

int bool=0;
int titleBOOL=0;
int authorBOOL=0;

char* removeVirgula(char* name){
        name[strlen(name)-1]='\0';
        return name;
}

char* retiraEnter(char* lixo){
        int i = 0;
        while(lixo[0]=='\n'){lixo++;}
        i=strlen(lixo);
        while(lixo[i-1]=='\n'){lixo[i-1]='\0';i--;}

        return lixo;
}
void registaAuthors(){
        char** aux = a;
        char* autor;
        int i,j;

        numeroNomesAutores[numeroAutores]=n;
        for(i=0; i<=numeroAutores;i++){
                if(numeroAutores==0){
                        //0 ou 1 autores
                        if(tot==0){
                                printf("author = { },\n");
                        }else{
                                if(tot==1){
                                        printf("author = { %s },\n",aux[0]);
                                }else{
                                        if(aux[0][strlen(aux[0])-1]==','){
                                                printf("author = { %c. %s },\n",aux[1][0],
                                                    removeVirgula(aux[0]));
                                        }else{
                                                printf("author = { %c. %s },\n",aux[0][0],aux[
                                                    numeroNomesAutores[i]-1]);
                                        }
                                }
                        }
                }else{
                        if(i==0){
                                if(numeroNomesAutores[i]==1){
                                        printf("author = { %s and\n",aux[0]);
                                }else{
                                        if(aux[0][strlen(aux[0])-1]==','){
                                                printf("author = { %c. %s and\n",aux[1][0],
                                                    removeVirgula(aux[0]));
                                        }else{
                                                printf("author = { %c. %s and\n",aux[0][0],aux[
                                                    numeroNomesAutores[i]-1]);
                                        }
                                }

                        }else{
                                if(i==numeroAutores){
                                        if(numeroNomesAutores[i]==1){
                                                printf("\t%s },\n",aux[0]);
                                        }else{
                                                if(aux[0][strlen(aux[0])-1]==','){
                                                        printf("\t%c. %s },\n",aux[1][0],
                                                            removeVirgula(aux[0]));
                                                }else{
                                                        printf("\t%c. %s },\n",aux[0][0],aux[
                                                            numeroNomesAutores[i]-1]);
                                                }
                                        }

                                }else{
                                        if(numeroNomesAutores[i]==1){
                                                printf("\t%s and\n",aux[0]);
```

```
79                                              }else{
80                                                      if(aux[0][strlen(aux[0])-1]==','){
81                                                              printf("\t%c. %s and\n",aux[1][0],
                                                                    removeVirgula(aux[0]));
82                                                      }else{
83                                                              printf("\t%c. %s and\n",aux[0][0],aux[
                                                                    numeroNomesAutores[i]-1]);
84                                                      }
85                                              }
86
87                                      }
88                              }
89                              aux=aux+numeroNomesAutores[i];
90                      }
91              }
92  }
93
94  %}
95
96  %x author categoria title
97  %%
98  @[A-Za-z]+\{[A-Za-z0-9\:\+]+,    {
99              BEGIN(categoria);
100             printf("%s\n",yytext);
101             tot=0;
102             bool=0;titleBOOL=0;authorBOOL=0;
103             lixo[0]='\0';
104 }
105
106 <categoria>[Aa][Uu][Tt][Hh][Oo][Rr][ \t]*=[ \t]*[\"\{]   {
107             BEGIN(author);
108             numeroAutores=0;
109             tot=0;n=0;
110             authorBOOL=1;
111 }
112
113 <author>[ ]+[Aa][Nn][Dd][ ]+     {
114             numeroNomesAutores[numeroAutores]=n;
115             numeroAutores++;
116             n=0;
117 }
118
119 <author>[ ]+[Aa][Nn][Dd][ ]*\n?[ ]*              {
120             numeroNomesAutores[numeroAutores]=n;
121             numeroAutores++;
122             n=0;
123 }
124 <author>[ ]*\n[ ]*[Aa][Nn][Dd][ ]+               {
125             numeroNomesAutores[numeroAutores]=n;
126             numeroAutores++;
127             n=0;
128 }
129
130 <author>\{[^ ,]+[^ \"\}]\} {
131             a[tot]=strdup(yytext);
132             n++;tot++;
133 }
134 <author>[^ ,]+([^ \"\},\n]|[^ \"\}],) {
135             a[tot]=strdup(yytext);
136             n++;tot++;
137 }
138
139 <author>[ ]*[\"\}],?\n   {
140             BEGIN(categoria);
141             registaAuthors();
142 }
143 <author>.|\n     {;}
144
145
146 <categoria>[ \n\t][Tt][Ii][Tt][Ll][Ee][ \t]*=[ \t]*\"    {
147             BEGIN(title);
148             if(authorBOOL){
149                     printf("%s",yytext);
150             }else{
151                     strcat(lixo,yytext);
152             }
153             titleBOOL=1;
154 }
155
156 <categoria>[ \n\t][Tt][Ii][Tt][Ll][Ee][ \t]*=[ \t]*\{    {
157             BEGIN(title);
158             if(authorBOOL){
159                     printf("%s",yytext);
160             }else{
161                     strcat(lixo,yytext);
162             }
163             titleBOOL=1;
164 }
165
166 <title>[^=]*\",?                 {
167             BEGIN(categoria);
168             if(yytext[strlen(yytext)-1]==','){
169                     yytext[strlen(yytext)-2]='}';
```

```
170         } else{
171                 yytext[strlen(yytext)-1]='}';
172         }
173         if(authorBOOL){
174                 printf("%s",yytext);
175         }else{
176                 strcat(lixo,yytext);
177         }
178 }
179
180 <title>[^=]*\},? {
181         BEGIN(categoria);
182         if(authorBOOL){
183                 printf("%s",yytext);
184         }else{
185                 strcat(lixo,yytext);
186         }
187
188 }
189
190 <categoria>[ ]*=[ ]*\"          {
191         if(titleBOOL && authorBOOL){
192                 printf(" = {");
193         }else{
194                 strcat(lixo," = {");
195         }
196 }
197
198 <categoria>\",                          {
199         if(titleBOOL && authorBOOL){
200                 printf("},");
201         }else{
202                 strcat(lixo,"\n},");
203         }
204 }
205
206 <categoria>.|\n {
207         if(titleBOOL && authorBOOL ){
208                 if(!bool){
209                         printf("%s%s",retiraEnter(lixo),yytext);
210                         bool=1;
211                 }else{
212                         printf("%s",yytext);
213                 }
214          }else{
215                 strcat(lixo,yytext);
216         }
217 }
218
219 <categoria>\}\n {
220         BEGIN(INITIAL);
221         if(!(titleBOOL && authorBOOL)){
222                 printf("%s",lixo);
223         }
224         printf("}\n\n");
225 }
226
227 .|\n {;}
228
229 %%
230
231 int yywrap() {return 1;}
232 int main(){
233
234         yylex();
235
236         return 0;
237 }
```

5.2.3 Alínea C

De seguida apresentamos uma cópia do programa do ficheiro *tp2.l* desenvolvido em *Flex*, de modo a responder à alínea C.

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

#include <unistd.h>     /* chamadas ao sistema: defs e decls essenciais */
#include <fcntl.h>


char* a[500];
int n=0;
int tot=0;
int numeroAutores=0;
int numeroNomesAutores[100];
int bool=0;
char* new[1000];


int nParceiros=0;
char* authorParceria[100];
char* authorProcura;

char* removeVirgula(char* name){
        name[strlen(name)-1]='\0';
        return name;
}

void registaAuthors(){
        int i,j;
        char** aux = a;
        char buf[100];
        numeroNomesAutores[numeroAutores]=n;
        FILE* fp = fopen("graf.dot","a+");
        FILE* fpAuthor = fopen("grafAuthor.dot","a+");

        for(i=0; i<=numeroAutores;i++){
                if(numeroNomesAutores[i]==1){
                        sprintf(buf,"%s",aux[0]);
                }else{
                        if(aux[0][strlen(aux[0])-1]==','){
                                sprintf(buf,"%c. %s",aux[1][0],removeVirgula(aux[0]));
                        }else{
                                sprintf(buf,"%c. %s",aux[0][0],aux[numeroNomesAutores[i]-1]);
                        }

                }
                new[i]=strdup(buf);
                aux=aux+numeroNomesAutores[i];
        }
        for(i=0;i<=numeroAutores;i++){
                for(j=0;j<=numeroAutores;j++){
                        if(i!=j){
                                fprintf(fp,"\t\"%s\" -- \"%s\";\n",new[i],new[j]);
                                if(!strcmp(new[i],authorProcura)){
                                        fprintf(fpAuthor,"\t\"%s\" -- \"%s\";\n",new[i],new[j]);
                                        authorParceria[nParceiros]=strdup(new[j]);
                                        nParceiros++;
                                }
                        }

                }
        }
        fclose(fp);
        fclose(fpAuthor);
        n=0;
}


%}

%x author categoria graf

%%
@[A-Za-z]+\{[A-Za-z0-9:\+]+,            {
        BEGIN(categoria);
        tot=0;
}

<categoria>[Aa][Uu][Tt][Hh][Oo][Rr][ \t]*=[ \t]*[\"\{]  {
        BEGIN(author);
        numeroAutores=0;
        tot=0;n=0;
}

<author>[ ]+[Aa][Nn][Dd][ ]+ {
```

```
85          numeroNomesAutores[numeroAutores]=n;
86          numeroAutores++;
87          n=0;
88  }
89  <author>[ ]+[Aa][Nn][Dd][ ]*\n?[ ]*      {
90          numeroNomesAutores[numeroAutores]=n;
91          numeroAutores++;
92          n=0;
93  }
94  <author>[ ]*\n[ ]*[Aa][Nn][Dd][ ]+       {
95          numeroNomesAutores[numeroAutores]=n;
96          numeroAutores++;
97          n=0;
98  }
99
100 <author>\{[^ ,]+[^ \"\}]\}       {
101         a[tot]=strdup(yytext);
102         n++;tot++;
103 }
104
105 <author>[^ ,]+([^ \"\},\n]|[^ \"\}],)    {
106         a[tot]=strdup(yytext);
107         n++;tot++;
108 }
109
110 <author>[ ]*[\"\}],?     {
111         BEGIN(categoria);
112         registaAuthors();
113 }
114
115 <author>.|\n     {;}
116
117 <categoria>.|\n {;}
118 <categoria>\}\n {BEGIN(INITIAL);}
119 .|\n     {;}
120
121 %%
122 int yywrap() {return 1;}
123 int main(int argc, char** argv){
124
125         if(argc<=1){
126                 authorProcura=strdup("");
127         }else{
128                 authorProcura=argv[1];
129         }
130
131         FILE* fp = fopen("graf.dot","a+");
132         fprintf(fp,"strict graph authors {\n");
133         fclose(fp);
134
135         FILE* fpAuthor = fopen("grafAuthor.dot","a+");
136         fprintf(fpAuthor,"strict graph authors {\n");
137         fclose(fpAuthor);
138
139         yylex();
140
141         fp = fopen("graf.dot","a+");
142         fprintf(fp,"}");
143         fclose(fp);
144
145         fpAuthor = fopen("grafAuthor.dot","a+");
146         fprintf(fpAuthor,"}");
147         fclose(fpAuthor);
148
149         return 0;
150 }
```