Processamento de Linguagens (3º ano de MIEI)

Trabalho Prático nº1

Relatório de Desenvolvimento

Henrique Faria (a82200)

João Marques (a81826)

Nuno Rei (a81918)

31 de Março de 2019

Resumo

O presente documento representa o relatório de desenvolvimento do primeiro trabalho prático da unidade curricular *Processamento de Linguagens* do terceiro ano do curso *Mestrado Integrado em Engenharia Informática*. Este trabalho tem como objectivo a realização de um filtro de texto, que filtre e transforme frases que correspondam a padrões definidos através de *Expressões Regulares*. Para tal vai ser utilizado o Flex que permite gerar *filtros de texto em C*.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O *Flex* é um programa que foi escrito por *Vern Paxson* em meados de 1987 e é utilizado para gerar analisadores léxicos. Os ficheiros flex possuem um conjunto de regras, isto é, pares de expressões regulares e código *C*. O flex tem como *output* um ficheiro com código fonte *C*, que depois de compilado e executado procura fazer correspondência com as regras definidas no ficheiro flex, e caso corresponda executa o código associado.

O primeiro trabalho prático desta unidade curricular consiste na utilização do *Flex* para a criação de um filtro de texto. O **enunciado a ser resolvido é o número quatro**, *Wikipedia PT, language links*, no qual é pretendida a realização de uma tabela *CSV* de tradução de títulos do português para outras 6 línguas: francês, inglês, italiano, russo, espanhol e alemão. Para além da criação da tabela é também pretendido que se efetue melhorias no filtro como ignorar entradas com *anos* e *"!"*, retirar prefixos como por exemplo *"Categoria:"*, e outras que se ache pertinente para obter um dicionário mais equilibrado. Por fim deve ser feita uma página html com a estatística de quantas traduções existem para cada língua.

Este documento vai relatar detalhadamente todas as decisões tomadas para a resolução do problema, desde o raciocínio até à implementação da solução, em que será apresentada a codificação e alguns testes que mostram o seu funcionamento.

O documento começa pela Análise e Especificação, no Capítulo 2, onde se faz uma análise detalhada do problema proposto de modo a compreender o que será lido pelo filtro de texto e formalizar uma ideia sobre os resultados que devem ser obtidos. De seguida no Capítulo 3, Concepção/desenho da Resolução, irá ser apresentada as estruturas de dados e algoritmos que foram pensados e que serão utilizados para a resolução. No Capítulo 4, Codificação e Testes, vão ser apresentadas todas as decisões tomadas para implementação, bem como problemas surgidos e possíveis alternativas para aquilo que foi codificado, e ainda testes realizados com os respectivos resultados de forma a mostrar o funcionamento da solução apresentada. Por fim no Capítulo 5 termina-se o relatório com uma síntese do que foi dito, conclusões tiradas, possíveis melhorias para trabalho futuro e principais dificuldades encontradas e como poderiam ser superadas.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

No problema descrito no enunciado **número quatro** é pedido que seja criada uma tabela *CSV* de tradução (para **português**) de títulos de páginas da wikipédia de uma destas seis **línguas**: **inglês, italiano, espanhol, russo, francês e alemão**. Para isto é necessário que seja feito o *parse* de dois ficheiros, fornecidos como *input* de forma a filtrar e obter apenas o necessário para formar a tabela de tradução. Para além da criação da tabela de tradução são ainda apresentados requisitos adicionais de forma a tornar o dicionário de traduções mais equilibrado e é pedido que seja feita uma estatística sobre a linguagem que tem maior número de traduções.

2.2 Especificação dos Requisitos

2.2.1 Dados

Em conjunto com o enunciado do problema são fornecidos dois ficheiros que contém os dados necessários para a sua realização. O formato destes ficheiros é o seguinte:

1) O primeiro é um ficheiro de texto com títulos em português de artigos da *Wikipédia*. Para além do título as entradas são acompanhadas por um identificador do artigo, e os campos são distinguidos pelas marcas $< title >$ e $< id >$, caso seja título ou identificador respectivamente, tal como se pode confirmar pela Figura 2.1.

```
<title>Astronomia</title> <id>220</id>
--
<title>América Latina</title> <id>223</id>
--
<title>Albino Forjaz de Sampaio</title> <id>224</id>
--
<title>Anno Domini</title> <id>226</id>
--
<title>Aquiles</title> <id>228</id>
--
<title>Categoria:Vilas de Luxemburgo (Bélgica)</title> <id>5934005</id>
--
<title>Categoria:Vilas de Flandres Oriental</title> <id>5934006</id>
--
<title>Categoria:Flandres Oriental</title> <id>5934007</id>
--
<title>Categoria:!Esboços sobre cavernas</title> <id>5934008</id>
```

Figura 2.1: Excerto do ficheiro ptwiki-20190220-pttit_id.txt

2) Para além do ficheiro de texto é ainda fornecido um ficheiro *sql* constituído por uma tabela com triplos do tipo *(id,língua,título)* que também se referem a títulos de artigos da *Wikipédia* mas este já não contém língua portuguesa, mas sim diversas línguas tal como se pode ver na Figura 2.2.

```
-- MySQL dump 10.16  Distrib 10.1.26-MariaDB, for debian-linux-gnu (x86_64)
--
-- Host: 10.64.32.116    Database: ptwiki
-- ------------------------------------------------------
-- Server version   10.1.37-MariaDB

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
......
INSERT INTO `langlinks` VALUES (220,'en','Astronomy'),
(165973,'af','Kategorie:Kultuur in die Verenigde Koninkryk'),
(2497006,'af','Kategorie:Kultuur in die Verenigde State van Amerika'),
(1792530,'af','Kategorie:Kultuur volgens kontinente'),(93682,'af','Kategorie:Kultuur volgens
lande'),....
```

Figura 2.2: Excerto do ficheiro ptwiki-latest-langlinks.sql

2.2.2 Pedidos

De seguida apresentam-se os requisitos propostos no enunciado que a solução deve implementar:

1) Produção de uma tabela em formato CSV de tradução dos títulos numa das seis línguas para português, isto é, extrair os triplos do ficheiro *SQL* que contenham *'fr','en','it','ru','es','de'* no segundo campo, e fazer corresponder com a respectiva tradução que se encontra no ficheiro de texto através do identificador;

2) O filtro deve ignorar as entradas que tenham outras línguas que não as seis referidas no ponto anterior;

3) Ignorar entradas com *datas, "!"* e outros casos que se ache pertinente não processar;

4) Retirar, das entradas, prefixos como *"Categoria:", "Wikipédia:",* e outros para que o dicionário das traduções fique mais equilibrado;

5) Conter a opção de processar apenas um número dado de tuplos e ao fim de processados criar o ficheiro *CSV* e sair;

6) Efetuar uma estatística, em *html*, de quantas traduções existem por língua;

Requisitos adicionais que o grupo se propôs a implementar:

1) Dicionário em *HTML* que permite consultar as traduções das palavras, isto é, poder consultar as traduções das palavras começadas pela letra que desejar desde *A até Z*;

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

Para o armazenamento dos dados considerados relevantes foi criada uma estrutura *DIC_ENTRADA* que possui 7 *char**, cada um correspondente a uma língua (português, alemão, espanhol, francês, inglês, italiano e russo). Será criada uma estrutura destas por cada nova entrada válida lida do ficheiro de texto *ptwiki-20190220-pttit_id.txt* e todas essas entradas são guardadas numa *HashTable* disponibilizada pela biblioteca *glib.h* e *gmodule.h*.

```
typedef struct entrada {
    char* titulo;
    char* ingles;
    char* espanhol;
    char* frances;
    char* italiano;
    char* russo;
    char* alemao;
}*DIC_ENTRADA;
```

3.2 Algoritmos

Para a resolução do problema proposto e já com a estruturas de dados definidas, foram criadas *Expressões Regulares* de forma a corresponder a todas as exprssões relevantes para este trabalho. Devido à necessidade de fazer o parse de dois ficheiros distintos foram criados dois autómatos, um para cada ficheiro, e as respetivas expressões para iniciar esses autómatos de modo a poder posteriormente especificar regras para cada um dos ficheiros e estas não serem *"apanhadas"* pelo outro.

As expressões para iniciar os autómatos são as seguintes:

- <*>"<title>" - Inicia o autómato *F1*;
- <*>"INSERT INTO ‘langlinks‘ VALUES " - Inicia o autómato *F2*;
- <*>(.|\n) - Apanha qualquer outra expressão não apanhada anteriormente.

No autómato *F1* criado para o primeiro ficheiro são criadas várias expressões, algumas expressões não são tratadas para remover os casos em que os dados tem *anos* ou *"!"*, outras são tratadas de forma mais elaborada para remover excessos como *"Categoria:"*. Estas expressões são utilizadas para apanhar os títulos, existem ainda as expressões para apanhar os identificadores dos respetivos títulos.

As expressões utilizadas para este primeiro autómato são:

- "<title>".*{NUMEROS} - Apanha os títulos que contêm *anos*.
- "<title>".*[!|?] - Apanha os títulos que contenham *"!"* ou *"?"*.
- "<title>".*:{LETRAS}+ - Apanha os títulos que contenham excessos como *"Categoria:"*.
- "<title>"{LETRAS}+ - Apanha os títulos que não tenham excessões.
- "<id>"{NUMEROS}+ - Apanha todos os identificadores.
- (.|\n) - Apanha qualquer outra expressão não apanhada anteriormente.

Do mesmo modo que foi criado o autómato *F1* para o ficheiro *ptwiki-20190220-pttit_id.txt* foi criado o autómato *F2* para o ficheiro *ptwiki-latest-langlinks.sql*. Da mesma forma foram criadas *Expresões Regulares* para apanhar as expressões pretendidas. As expressões geradas foram as seguintes:

- (\({NUMEROS}+",'de','"{LETRAS}+"')")? - Apanha as traduções em alemão.
- (\({NUMEROS}+",'en','"{LETRAS}+"')")? - Apanha as traduções em inglês.
- (\({NUMEROS}+",'es','"{LETRAS}+"')")? - Apanha as traduções em espanhol.
- (\({NUMEROS}+",'fr','"{LETRAS}+"')")? - Apanha as traduções em francês.
- (\({NUMEROS}+",'it','"{LETRAS}+"')")? - Apanha as traduções em italiano.
- (\({NUMEROS}+",'ru','"[^']*"')")? - Apanha as traduções em russo.

Depois destas expressões serem apanhadas individualmente invoca-se a função *faztraducao(char* ,char*)* que recebe o tuplo apanhado e a respetiva língua. Dentro dessa funcção é invocada uma outra para obter o identificador desse tuplo, denominada de *getId(char*)*. Caso esse identificador corresponda a uma posição da *HashTable* é retirada a tradução do tuplo através de duas funções, uma que retira os excessos como *"Categoria:"* e outra que obtém a tradução final depois de removido o excesso, estas funções são a *retiraExcesso(char*)* e *getdados(char*)* respetivamente. Por fim é adicionada à estrutura *DIC_ENTRADA* que se encontra na posição referida anteriormente a tradução retirada na respetiva linguagem, a adição é feita através do método *addLang(DIC_ENTRADA, char*)* que verifica qual é a língua da tradução e coloca na respetiva língua.

3.3 Compilação e Execução

Para facilitar a execução do projeto foi criada uma makefile. O modo mais simples para executar o projeto é ter os ficheiros *makefile*, *1tp1.fl*, *createFiles.c*, *ptwiki-20190220-pttit_id.txt* e *ptwiki-latest-langlinks.sql* na mesma diretoria e no terminal executar apenas *make*. Quando executado desta forma existe um limite muito alto de tuplos, o que permite fazer um dicionário completo. No caso de se querer criar um limite de tuplos têm de ser executado 3 comandos (garantindo que *makefile*, *1tp1.fl* e *createFiles.c* estão na mesma

diretoria), primeiro o comando *make alternativa*, em segundo o *./tp1 (nºtuplos) ptwiki-20190220-pttit_id.txt ptwiki-latest-langlinks.sql* e por último *make last*. No caso de os ficheiros *.txt* e *.sql* estarem noutra diretoria é necessário especificar a diretoria onde se encontram.

Depois de um dos métodos utilizados anteriormente pode ser visualizado no seu WebBrowser o ficheiro *indice.html* que mostra todas as traduções realizadas e as respetivas estatísticas.

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

Neste trabalho poderiamos ter implementado as condições de parser do flex num único autómato, no entanto, optamos por uma alternativa que não só torna o código mais limpo como ainda o torna mais eficiente. Nesta alternativa, dividimos o parser do flex em 2 autómatos. Um para o primeiro ficheiro e um para o segundo, desta forma não só é mais fácil perceber quais as condições aplicadas a cada um, como são menos condições a terem de ser verificadas nos mesmos, fazendo com que este programa ganhe em eficiência.

Decidimos também utilizar uma *HashTable* para guardar tanto as palavras em português como em inglês, espanhol, italiano, francês, russo e alemão. Desta forma, como a cada palavra está associado um índice, é mais fácil aceder às mesmas através dele e associar traduções.

Após a obtenção das palavras em português e das respetivas traduções disponíveis no segundo ficheiro decidimos implementar um dicionário em html, no qual apresentamos algumas estatísticas relativamente à quantidade de palavras presentes no dicionário em cada língua bem como o número de palavras começadas por cada letra do alfabeto. Dentro de cada entrada do dicionário (letra pela qual uma palavra começa), é apresentada uma lista de palavras começadas por esta letra na qual se pode carregar para visualizar as traduções disponíveis neste dicionário para as palavras em português. Para realizar isto foi necessário criar ficheiros html para cada entrada no dicionário bem como para o menu do nosso dicionário e escrever o código html a partir da nossa aplicação. No início começamos por fechar o descritor de saída do programa e para cada palavra do nosso dicionário abrir um novo descritor de saída para o ficheiro correspondente à inicial da palavra portuguesa, usando depois um printf para escrever nesse ficheiro. Seguidamente, passamos para um array de apontadores para ficheiros para não os criarmos em cada iteração. Ainda assim, não era suficientemente eficiente e portanto acabamos por utilizar um array de FILE* para fazer uso do fopen evitando assim fechar e abrir descritores de saída a cada iteração.

Por último tivemos problemas com os caracteres russos que apareciam desformatados no ficheiro *dictionary.txt* onde guardavamos todas as palavras do nosso dicionário em formato txt. Para resolver este problema, utilizamos o comando: *iconv -c -t utf8 dictionary.txt ¿ dic.txt* que nos faz a conversão destes caracteres russos desformatados para formatados, colmatando assim o problema. Convem referir que nos Mac este comando gera um erro ao ser executado numa makefile, assim, enquanto os usuários do linux podem correr o comando *make last*, para executar este comando os usuários do Mac têm de corre-lo à mão no terminal.

4.2 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (valores introduzidos) e os respectivos resultados obtidos:

No primeiro caso usamos o comando *make* para compilar e executar o programa por nós. Desta forma esta cria os ficheiros html, o dictionary.txt, o dic.txt, compila e corre o programa 1tp1.fl criando o executável e executando-o passando-lhe os ficheiros a serem consultados. Note-se que para executar esta operação os ficheiros têm de estar todos na mesma diretoria do makefile. Por defeito é atribuído um valor à variável tuplos que representa o número de palavras nas línguas usadas para fazer traduções a serem comparadas com os índices correspondentes às palavras em português. O output correspondente apresenta algumas estatísticas referentes ao dicionário bem como entradas para cada letra do alfabeto. Na imagem seguinte temos um exemplo de uma entrada do dicionário onde se mostram as traduções disponíveis. Note-se que nem todas as palavras em português têm traduções disponíveis, isto deve-se à falta de traduções no ficheiro .sql.

```
henrique:PL$ make
rm -f lex.yy.c
rm -f tp1
rm -f output.txt
rm -f *.html
rm -f dic.txt
rm -f dictionary.txt
gcc createFiles.c -o files
./files
rm files
flex -8 -f 1tp1.fl
gcc -I/usr/local/Cellar/glib/2.60.0_1/include/glib-2.0 -I/usr/local/Cellar/glib/2.60.0_1/lib/gl
ib-2.0/include -I/usr/local/opt/gettext/include -I/usr/local/Cellar/pcre/8.43/include lex.yy.c
-o tp1 -L/usr/local/Cellar/glib/2.60.0_1/lib -L/usr/local/opt/gettext/lib -lglib-2.0 -lintl
time ./tp1 ptwiki-20190220-pttit_id.txt ptwiki-latest-langlinks.sql
       32.90 real        26.10 user         3.94 sys
rm tp1
```

Figura 4.1: Execução do comando make

Dicionário

Estatísticas:

Número de palavras Portuguesas do Dicionário: 1996736

Número de traduções para Alemão: 322068

Número de traduções para Espanhol: 378351

Número de traduções para Frances: 372220

Número de traduções para Inglês: 557946

Número de traduções para Italiano: 347867

Número de traduções para Russo: 415709

Escolha uma entrada do dicionário para aceder á lista de palavras disponíveis:

- A (nº de palavras: 174254)
- B (nº de palavras: 94542)
- C (nº de palavras: 215010)
- D (nº de palavras: 77971)
- E (nº de palavras: 97983)
- F (nº de palavras: 86816)
- G (nº de palavras: 68279)
- H (nº de palavras: 52278)
- I (nº de palavras: 53990)
- J (nº de palavras: 62782)
- K (nº de palavras: 26453)
- L (nº de palavras: 111927)
- M (nº de palavras: 134523)
- N (nº de palavras: 52810)
- O (nº de palavras: 43721)
- P (nº de palavras: 183360)
- Q (nº de palavras: 7352)
- R (nº de palavras: 89467)

Figura 4.2: Página principal do Dicionário html gerado

Figura 4.3: Primeira entrada do dicionário correspondente à letra 'a'

Para um segundo teste no qual nós escolhemos o número de tuplos a serem comparados temos um comando *make alternativa* que apenas cria os html e os dictionary.txt e dic.txt para além de compilar o programa 1tp1.fl.

Neste caso podemos especificar na execução do programa o número de tuplos a serem comparados, também podemos usar esta alternativa caso um ou mais dos ficheiros a serem utilizados para a construção do dicionário não se encontrem na mesma diretoria.

Caso não especifiquemos o numero de tuplos estes são atribuidos automáticamente tomando o valor de 10000000.

Podemos observar a figura 4.2 pois não houve mudanças nas estatísticas do nosso dicionário, isto porque os tuplos não foram especificados, tendo, tanto neste teste como no anterior, ficado om um valor de 10 milhões, sendo estas duas execuções iguais em resultado.

```
henrique:PL$ make alternativa
rm -f lex.yy.c
rm -f tp1
rm -f output.txt
rm -f *.html
rm -f dic.txt
rm -f dictionary.txt
gcc createFiles.c -o files
./files
rm files
flex -8 -f 1tp1.fl
gcc -I/usr/local/Cellar/glib/2.60.0_1/include/glib-2.0 -I/usr/local/Cellar/glib/2.60.0_1/lib/gl
ib-2.0/include -I/usr/local/opt/gettext/include -I/usr/local/Cellar/pcre/8.43/include lex.yy.c
-o tp1 -L/usr/local/Cellar/glib/2.60.0_1/lib -L/usr/local/opt/gettext/lib -lglib-2.0 -lintl
henrique:PL$ time ./tp1 ptwiki-20190220-pttit_id.txt ptwiki-latest-langlinks.sql

real    0m28.923s
user    0m24.637s
sys     0m3.208s
```

Figura 4.4: Execução do comando *make alternativa* seguida da execução do programa sem expecificação dos tuplos

De seguida apresentamos a alternativa na qual se utiliza a expecificação de tuplos.

Neste caso podemos verificar que o índice gerado apresenta 600 traduções para alemão, isto deve-se ao facto de que um tuplo é uma palavra de outra língua que corresponde a uma palavra portuguesa no dicionário, para além disso tratam-se apenas de traduções para alemão pois no ficheiro de extensão sql as palavras estão ordenadas por língua e como alemão é a primeira lingua que aparece e tem mais de 600 palavras correspondentes às portuguesas apenas aparecem estas. Cada entrada do dicionário apresenta as mesmas palavras em português que as dos testes anteriores, a diferença está no número de traduções disponíveis em cada entrada bem como as línguas para as quais existe tradução, dispensando assim a sua apresentação.

```
henrique:PL$ make alternativa
rm -f lex.yy.c
rm -f tp1
rm -f output.txt
rm -f *.html
rm -f dic.txt
rm -f dictionary.txt
gcc createFiles.c -o files
./files
rm files
flex -8 -f 1tp1.fl
gcc -I/usr/local/Cellar/glib/2.60.0_1/include/glib-2.0 -I/usr/local/Cellar/glib/2.60.0_1/lib/gl
ib-2.0/include -I/usr/local/opt/gettext/include -I/usr/local/Cellar/pcre/8.43/include lex.yy.c
-o tp1 -L/usr/local/Cellar/glib/2.60.0_1/lib -L/usr/local/opt/gettext/lib -lglib-2.0 -lintl
henrique:PL$ time ./tp1 600 ptwiki-20190220-pttit_id.txt ptwiki-latest-langlinks.sql

real    0m23.675s
user    0m20.621s
sys     0m2.518s
```

Figura 4.5: Execução do comando *make alternativa* seguida da execução do programa com expecificação dos tuplos

Dicionário

Estatísticas:

Número de palavras Portuguesas do Dicionário: 1996736

Número de traduções para Alemão: 600

Número de traduções para Espanhol: 0

Número de traduções para Frances: 0

Número de traduções para Inglês: 0

Número de traduções para Italiano: 0

Número de traduções para Russo: 0

Escolha uma entrada do dicionário para aceder á lista de palavras disponíveis:

- A (nº de palavras: 174254)
- B (nº de palavras: 94542)
- C (nº de palavras: 215010)
- D (nº de palavras: 77971)
- E (nº de palavras: 97983)
- F (nº de palavras: 86816)
- G (nº de palavras: 68279)
- H (nº de palavras: 52278)
- I (nº de palavras: 53990)
- J (nº de palavras: 62782)
- K (nº de palavras: 26453)
- L (nº de palavras: 111927)
- M (nº de palavras: 134523)
- N (nº de palavras: 52810)
- O (nº de palavras: 43721)
- P (nº de palavras: 183360)
- Q (nº de palavras: 7352)
- R (nº de palavras: 89467)
- S (nº de palavras: 137867)

Figura 4.6: Indice contruido para 600 túpulos

Capítulo 5

Conclusão

A avaliação feita pelo grupo quanto à solução obtida é bastante positiva, pois todos os pedidos foram concretizados e ainda foi realizado um pedido extra para uma melhor apresentação do trabalho em causa.

Depois de uma breve reflexão sobre o resultado final o grupo de trabalho acha que o trabalho realizado ficou dentro das espectativas, podendo proventura serem melhorados alguns tópicos do mesmo. Uma vez que o *Flex* é uma ferramenta muito poderosa no que toca a fazer *match* de *Expressões Regulares* tudo o que fizer *match* com as expressões criadas pelo grupo ele irá fazer alguma coisa com elas, mas existem alguns casos muito específicos em que as expressões geradas não são suficientes e para apanhar esses casos para tentar ter um programa muito perfeito iria causar uma complexidade extra muito elevada e teriam de ser realizados muitos testes para verificar se este funcionava de forma correta.

Apêndice A

Código do Programa

```
%x F1 F2
%option yylineno
%option noyywrap
LETRAS ([A-Za-z\ ]|ñ|à|á|â|å|ã|ä|ò|ó|ç|õ|é|è|ô|ō|ö|ê|ë|í|ì|ú|ü|ù|ß|Ø|Ú|Ù|Û|Ó|Ô|Ò|Ì|Î|Í
|À|Â|Á|Ê|É|È|ş|’|\-|,|\.)
NUMEROS [0-9]

%%
<*>"<title>"                              {BEGIN F1;}
<*>"INSERT INTO ‘langlinks‘ VALUES "      {BEGIN F2;}
<F1>{
    "<title>".*{NUMEROS}+           {;}
    "<title>".*[!?]                 {;}
    "<title>"[^0-9]*:{LETRAS}+      {int x = 0;
                                     while(yytext[x] != ’:’){x++;}
                                     if(yytext[x+1] == ’ ’ || yytext[x+1] == ’:’) x++;
                                     frase=(char*)malloc((strlen(yytext)-x+1)*sizeof(char));
                                     sprintf(frase,"%s",yytext+x+1);
                                     frase[strlen(yytext)-x] = ’\0’;
                                     e = novaEntrada(frase);
                                     nova = 1;}
    "<title>"{LETRAS}+              {int x = strlen(yytext)-6;
                                     frase = (char*) malloc(x*sizeof(char));
                                     sprintf(frase,"%s",yytext+7);
                                     frase[x-1] = ’\0’;
                                     e = novaEntrada(frase);
                                     nova = 1;}
    "<id>"{NUMEROS}+                {if(nova != 0){
                                     a[0] += 1;
                                     g_hash_table_insert(h,GINT_TO_POINTER(atoi(yytext+4)),e);
                                     }
                                     nova=0;}
    .|\n                            {;}
}
```

```
<F2>{
    (\({NUMEROS}+",'de','"{LETRAS}+"')")?       {faztraducao(yytext+1,"de");}
    (\({NUMEROS}+",'es','"{LETRAS}+"')")?       {faztraducao(yytext+1,"es");}
    (\({NUMEROS}+",'fr','"{LETRAS}+"')")?       {faztraducao(yytext+1,"fr");}
    (\({NUMEROS}+",'en','"{LETRAS}+"')")?       {faztraducao(yytext+1,"en");}
    (\({NUMEROS}+",'it','"{LETRAS}+"')")?       {faztraducao(yytext+1,"it");}
    (\({NUMEROS}+",'ru','"[^']*"')")?           {faztraducao(yytext+1,"ru");}
}
<*>.|\n                                            {;}

%%
DIC_ENTRADA novaEntrada(char* tit){
    DIC_ENTRADA ent = malloc(sizeof(struct entrada));
    ent->titulo = strdup(tit);
    ent->alemao = NULL;
    ent->espanhol = NULL;
    ent->frances = NULL;
    ent->ingles = NULL;
    ent->italiano = NULL;
    ent->russo = NULL;
    return ent;
}

void retiraExcesso(char* str) {
    int j;
    for(j = indice; str[j]!='\0' ; j++)
        if(str[j]==':') break;
    if(str[j]!='\0') {
        j++;
        int i = indice+1;
        for(; str[j]!='\0' ; j++, i++)
            str[i] = str[j];
        str[i] = '\0';
    }
}

int getId(char* str) {
    int i = 0;
    for(; str[i]!=','; i++);
    indice = i;
    char* tmp = malloc(sizeof(char)*(i+1));
    strncpy(tmp,str,i);
    tmp[i] = '\0';
    i = atoi(tmp);
    free(tmp);
    return i;
}
```

```
char* getdados(char* str) {
    char* res = malloc(sizeof(char)*(strlen(str+indice)+1));
    strcpy(res,str+indice);
    return res;
}

void addLang(DIC_ENTRADA dic, char* lang, char* dados) {
    if(strcmp(lang,"de")==0){ dic->alemao = strdup(dados); a[1]+=1;}
    else if(strcmp(lang,"es")==0){ dic->espanhol = strdup(dados); a[2]+=1;}
    else if(strcmp(lang,"fr")==0){ dic->frances = strdup(dados); a[3]+=1;}
    else if(strcmp(lang,"en")==0){ dic->ingles = strdup(dados); a[4]+=1; }
    else if(strcmp(lang,"it")==0){ dic->italiano = strdup(dados); a[5]+=1;}
    else{ dic->russo = strdup(dados); a[6]+=1;}
}

void faztraducao(char* str,char* lang) {
    if(tuplos!=0) {
        str[strlen(str)-2]='\0';
        int id = getId(str);
        DIC_ENTRADA dic = (DIC_ENTRADA) g_hash_table_lookup(h,GINT_TO_POINTER(id));
        if(dic != NULL) {
            indice+=7;
            retiraExcesso(str);
            char* dados = getdados(str);
            addLang(dic,lang,dados);
            tuplos--;
        }
    }
}

void printDicionario(DIC_ENTRADA dic) {
    if(dic->alemao != NULL) fprintf(dicfd,"PT->%s ; DE->%s\n",dic->titulo,dic->alemao);
    if(dic->espanhol != NULL) fprintf(dicfd,"PT->%s ; ES->%s\n",dic->titulo,dic->espanhol);
    if(dic->frances != NULL) fprintf(dicfd,"PT->%s ; FR->%s\n",dic->titulo,dic->frances);
    if(dic->ingles != NULL) fprintf(dicfd,"PT->%s ; EN->%s\n",dic->titulo,dic->ingles);
    if(dic->italiano != NULL) fprintf(dicfd,"PT->%s ; IT->%s\n",dic->titulo,dic->italiano);
    if(dic->russo != NULL) fprintf(dicfd,"PT->%s ; RU->%s\n",dic->titulo,dic->russo);
}

int main(int argc,char* argv[]) {
    GHashTableIter iter;
    gpointer key,value;
    h = g_hash_table_new(g_direct_hash,g_direct_equal);
    int x,size,i,fd;
    char* f=malloc(sizeof(char)*9);
    FILE* file[26];
    int b[26];
```

```
for(x=0;x<7;x++){a[x] = 0;}
for(x=0;x<26;x++){b[x] = 0;}
if(argc == 4) {tuplos = atol(argv[1]); i=2;}
else {tuplos=10000000;i=1;}
for(; i< argc ; i++) {
 yyin=fopen(argv[i],"r");
    yylex();
    fclose(yyin);
}
FILE* ind = fopen("indice.html","a");
for(x=0;x<26;x++){
   sprintf(f,"%c.html",(97+x));
   file[x] = fopen(f,"a");
}
dicfd = fopen("dictionary.txt","a");
f[1]='\0';
// aqui inserem-se as entradas nos ficheiros
g_hash_table_iter_init (&iter, h);
while (g_hash_table_iter_next (&iter, &key, &value)){
    if(value!=NULL){
       sprintf(f,"%c",tolower(((DIC_ENTRADA)value)->titulo[0]));
       fd=0;
       fd += (int)f[0]-97;
        if(fd < 26 && fd >= 0){
            b[((int)f[0]-97)]+=1;
            fprintf(file[fd],"<button title=\"Clique para mostrar/ocultar\"
                type=\"button\" onclick=\"if(document.getElementById('%s')
                .style.display=='none') {document.getElementById('%s')
                .style.display='block'} else{document.getElementById('%s')
                .style.display='none'}\">",((DIC_ENTRADA)value)->titulo,
                ((DIC_ENTRADA)value)->titulo,((DIC_ENTRADA)value)->titulo);
            fprintf(file[fd],"%s",((DIC_ENTRADA)value)->titulo);
            fprintf(file[fd],"</button>");
            fprintf(file[fd],"<div id=\"");
            fprintf(file[fd],"%s",((DIC_ENTRADA)value)->titulo);
            fprintf(file[fd],"\"style=\"display:none\">");
            fprintf(file[fd],"Traduções diponiveis:<br>");
            fprintf(file[fd],"Português: %s<br>",((DIC_ENTRADA)value)->titulo );
            if(((DIC_ENTRADA)value)->alemao != NULL)
                fprintf(file[fd],"Alemão: %s;<br>",((DIC_ENTRADA)value)->alemao);
            if(((DIC_ENTRADA)value)->espanhol != NULL)
                fprintf(file[fd],"Espanhol: %s;<br>",((DIC_ENTRADA)value)->espanhol);
            if(((DIC_ENTRADA)value)->frances != NULL)
                fprintf(file[fd],"Franços: %s;<br>",((DIC_ENTRADA)value)->frances);
            if(((DIC_ENTRADA)value)->ingles != NULL)
                fprintf(file[fd],"Ingles: %s;<br>",((DIC_ENTRADA)value)->ingles);
            if(((DIC_ENTRADA)value)->italiano != NULL)
```

```
                fprintf(file[fd],"Italiano: %s;<br>",((DIC_ENTRADA)value)->italiano);
            if(((DIC_ENTRADA)value)->russo != NULL)
                fprintf(file[fd],"Russo: %s;<br>",((DIC_ENTRADA)value)->russo);
            fprintf(file[fd],"</div><br>");
            fprintf(file[fd],"\n");
            printDicionario((DIC_ENTRADA)value);
        }
    }
}
close(STDOUT_FILENO);
fprintf(ind,"<html>\n\t<head><meta charset=\'UTF-8\'><style type=\"text/css\">
    a:link {text-decoration:none; color:black; text-align:center;} a:visited
    {text-decoration:none;color:black;text-align:center;} a:hover
    {text-decoration:underline;color:black;text-align:center;} </style>
    </head>\n\t<title>Dicionário</title>\n\t<ul>\n");
fprintf(ind,"<br><h1>Dicionário</h1><br>");
fprintf(ind,"<h3>Estatísticas:</h3><h4>Número de palavras Portuguesas do Dicionário:
    %d</h4><h4>Número de traduções para Alemão: %d </h4><h4>Número de traduções
    para Espanhol: %d </h4><h4>Número de traduções para Franças: %d </h4><h4>Número
    de traduções para Inglês: %d </h4><h4>Número de traduções para Italiano: %d
    </h4><h4>Número de traduções para Russo: %d </h4><p>Escolha uma entrada do
    dicionário para aceder á lista de palavras disponíveis:</p><br><br>",
    a[0],a[1],a[2],a[3],a[4],a[5],a[6]);
for(x=97;x<123;x++)
   fprintf(ind,"\n\t\t<li><a href=\'%c.html\'>%c (nº de palavras: %d)</a></li>",x,
   (x-32),b[(x-97)]);
fprintf(ind,"\n\t</ul>\n</html> ");
return 0;
}
```