Normalizador de Autores em BibTex
&
Processador de Inglês corrente em Flex

Processamento de Linguagens
(3º ano de Curso)
Trabalho Prático nº1 - Parte B
Relatório de Desenvolvimento

José Silva	Pedro Cunha	Gonçalo Moreira
(A74601)	(A73958)	(A73591)

23 de Abril de 2017

Resumo

Documentação do segundo trabalho prático da unidade curricular de "Processamento de Linguagens", o principal foco incide em utilizar o analisador léxico FLEX, para desenvolver processadores de texto. Simplifica o trabalho que seria necessário utilizando diretamente a linguagem de programação C, facilitando o processo. Demonstrando e documentando as soluções propostas pelo grupo de trabalho para os doius problemas escolhidos, termina-se o relatório com uma análise argumentativa sobre a eficiência dessas mesmas soluções.

Conteúdo

Capítulo 1

Introdução

Tal e qual como foi descrito no anterior trabalho prático, os últimos anos trouxeram consigo a inevitabilidade de processar cada vez mais texto. Extrair e editar determinadas linhas, onde certos padrões são bastante evidentes, tornou-se assim algo fundamental. Justificam-se assim as várias ferramentas criadas para o efeito, dentro das quais se destaca o FLEX. Assumindo-se como um programa capaz de gerar analisadores lexicais, juntamente com o uso de expressões regulares e as funcionalidades disponibilizadas pela linguagem de programação C, a utilização do FLEX permite a resolução dos vários problemas relacionados com o que anteriormente foi descrito.

Neste segundo trabalho prático da unidade curricular de "Processamento de Linguagens", através dos meios descritos anteriormente, vão ser realizados dois filtros de texto capazes de processar algumas das características da língua inglesa e de normalizar certos componentes de um ficheiro no formato BibTex. Para além disso, são realizados alguns extras como, por exemplo, a criação de um grafo representativo das ligações entre diversos autores e um página HTML onde são apresentados os diversos verbos no infinitivo não flexionado resultantes do processamento de um texto em inglês.

Estrutura do Relatório

No capítulo 1 faz-se uma pequena introdução ao problema e às ferramentas utilizadas para a resolução deste. Para além disso, é descrita de uma forma breve a estrutura do relatório.

No capítulo 2 faz-se uma análise breve mas mais detalhada do problema escolhido pelo grupo de trabalho.

No capítulo 3 é descrito de uma forma sumarizada o procedimento utilizado para solucionar as várias questões propostas pelos enunciados.

No capítulo 4 são apresentados alguns testes e respetivos resultados para comprovar o respectivo funcionamento das soluções apresentadas.

Finalmente, no capítulo 5 termina-se o relatório com uma síntese do que foi dito, as conclusões e o trabalho futuro.

Capítulo 2

Análise e Especificação

2.1 Normalizador de Autores em BibTex

2.1.1 Descrição informal do problema

É fornecido um ficheiro em BibTex(como input), com várias entradas e diversos autores e editores por entrada. Pretende-se que se desenvolva um "Normalizador"para ler esse mesmo ficheiro e gerar um ficheiro equivalente (em BibTex)com os nomes de todos os editores e autores normalizados. Também se pretende converter todos os carateres com acentos explícitos em caracteres portugueses. A forma de normalização dos nomes e dos acentos explícitos é apresentada em detalhe a baixo, na Especificação dos Requisitos.

2.1.2 Especificação dos Requisitos

Dados

Como já foi referido, é fornecido um ficheiro em BibTex com várias entradas e diversos autores e editores por entrada. Este ficheiro contém, em cada entrada, diversos fields que poderão ou não incluir os fields author e editor. Podem existir diversos tipos de entrada, com número e tipos diferentes de fields.

Cada field pode ocupar uma ou mais linhas, dependendo da informação que o mesmo representa. Na maioria dos casos de editor e author ocupa apenas uma, mas não é regra. Quando existem muitos autores e/ou editores é normal que sejam necessárias mais linhas. Podem também existir acentos explícitos em qualquer field que tenha texto, incluindo nos nomes de editores e autores.

Pedidos

Como primeiro requisito, é pedido que todos os acentos explícitos(e cedilhas) sejam convertidos para caracteres portugueses, exemplos:

Anast\'acia ou Anast{\'a}cia deve ser convertido em: Anastácia.
Gon\c{c}alo ou Gon{\c{c}}alo deve ser convertido em: Gonçalo.
Também é pedido que os nomes de autores e editores sejam normalizados, todos têm que apresentar a mesma forma. A forma requisitada é a seguinte: "Apelido, N1. N2."em que N1 e N2 são possíveis nomes próprios e Apelido é, obviamente, o apelido(ou apelidos em alguns casos). Salienta-se que também é necessário que sejam todos da forma "author/editor = { ... }", por isso casos que usem aspas ou o número errado de espaços também devem ser corrigidos.
Exemplo de normalização:
author="Martini, Ricardo G. and Ara\'ujo, Cristiana and Almeida, Jos\'e Jo\ ao and Henriques, Pedro"
Deve ficar:
author = {Martini, R. G. and Araújo, C. and Almeida, J. J. and Henriques, P.}

2.2 Processador de Inglês corrente

O problema proposto pela equipa docente divide-se em duas partes. É fornecido um ficheiro exemplo contendo texto, em inglês, com vários exemplos das multiplas contrações caracteristicas desta lingua e com vários verbos numa determinada f orma nominal.

Numa primeira parte, pretende-se que se desenvolva um "Normalizador"capaz de gerar um ficheiro equivalente, mas que modifique as várias contrações existentes na gramática inglesa na sua forma normal. Numa segunda parte pretende-se gerar um ficheiro, em formato HTML, que possua todas as ocorrências de verbos no infinitivo não lexionado presentes no ficheiro de input.

2.2.1 Especificação dos Requisitos

Dados

O ficheiro fornecido para a execução das duas etapas, descritas anteriormente, apresenta um formato comum. O programa deve estar preparado para lidar com qualquer tipo de ficheiro, seja ele no formato BibTex, html, ou qualquer outro. As várias contrações podem estar definidas tanto na forma negativa como positiva e encontrar-se em qualquer zona do ficheiro. À semelhança, os verbos e os padrões que os acompanham estão sujeitos às mesmas características.

Pedidos

Como primeiro requisito, é pedido que as contrações sejam convertidos para a sua forma normal. Os casos não são uniformes como podemos ver pelos seguintes exemplos:

I'm - I am

W're - We are

Como segundo requisito, é pedido que sejam retirados, para um ficheiro diferente, todos os verbos no infinitivo não lexionado. Aqui, são três os padrões conhecidos:

- "To accumulate.". A forma verbal é precedida pela forma "to".

- "I might go". A forma verbal é antecedido por can, could, shall, should, will, would, may, might, bem como as suas formas negativas.

- "Do you want to go?". Na forma interrogativa o verbo é precedido por 'do,does, did,can,could,shall, should, will, would, may, might + qualquer outra palavra'. Aqui também, tal como nos anteriores exemplos, as formas negativas.

Capítulo 3

Concepção/desenho da Resolução

3.1 Normalizador de Autores em BibTex

3.1.1 Estruturas de Dados

y

3.1.2 Algoritmos

x

3.2 Processador de Inglês corrente

3.2.1 Estruturas de Dados

Pensando nos requisitos para este projeto é fácil de perceber que as duas partes do trabalho prático vão ter abordagens distintas, no que diz respeito às estruturas de dados. Na primeira parte, onde queremos como output um ficheiro semelhante àquele que recebemos como input, não precisamos de qualquer estrutura de dados auxiliar já que as funcionalidades do flex, da linguagem c e das expressões regulares nos permitem fazer face ao problema.

Por outro lado, no que diz respeito à passagem dos verbos no infinitivo não lexionado para um ficheiro output sem qualquer semelhança ao de input, é necessária uma estrutura de dados. Tal e qual como nas aulas práticas da unidade curricular, fizemos uso da biblioteca GLIB. Para o efeito, escolhemos a implementação de listas ligadas disponibilizada por esta biblioteca. Os factores que mais pesaram na escolha da estrutura de dados foram o facto de pudermos inserir uma quantidade indeterminada de itens e a disponibilização de uma função de inserção ordenada.

Ao longo da resolução do problema por vezes foi necessário apenas retirar uma certa parte de texto, do total daquilo que é adquirido através do uso de uma certa expressão regular. Assim, foi necessário utilizar a função strtok. Por consequência, definimos um array e uma string capazes de guardar informação temporária e de auxiliar a função descrita anteriormente. Finalmente, foi utilizada uma string para que a impressão de verbos repetidos não se realizasse.

3.2.2 Algoritmos

No que diz respeito à primeira parte do exercício prático, respeitante às contrações gramaticais, três abordagens distintas tiveram que ser realizadas. Numa primeira abordagem, respeitante por exemplo à contração "I'm", a expressão regular utilizada faz matching apenas com o "' "e tudo o que lhe sucede, já que são os únicos elementos que devem ser alterados por a expressão correspondente. Tudo o que antecede, permanece na mesma. O mesmo acontece para algumas das formas negativas como, por exemplo, "Don't". No que diz respeito a contrações como "Won't"ou "Can't", não existe um padrão especifico e por isso a expressão regular que faz matching deve ser alterada na sua totalidade.

Por outro lado, no que diz respeito à segunda parte do exercício prático, o nível de complexidade aumenta. Quando o verbo no infinitivo é precedido por "to", a expressão regular deve fazer matching com o "to", seguido de um espaço, e finalmente seguido do verbo que desejamos. Assim, na altura de guardar o verbo na estrutura de dados devemos redirecionar o apontador três caracteres à frente de modo a que a palavra "to"e o consequente espaço não sejam guardados. Quando o verbo no infinitivo é precedido por "can/could...", a expressão regular é semelhante à descrita anteriormente. Mas se no caso anterior era sempre conhecido o tamanho da palavra que precede o verbo, neste caso em concreto, não. Assim, é necessário utilizar a função strtok de modo a dividir as palavras que fizeram matching por espaços. Feito isto, os passos são semelhantes aos anteriores, com o verbo a ser inserido na estrutura de dados. Finalmente, quando estamos perante uma forma interrogativa, ao contrário dos dois casos anteriores, o verbo vai encontrar-se com maior probabilidade na segunda palavra que sucede às expressões "do/did/does". O procedimento é então bastante semelhante ao caso anterior.

Como sabemos, nem tudo retirado pelas expressões regulares corresponde a verbos. Assim, foram definidas uma série de excepções correspondentes a pronomes, advérbios, alguns padrões não frequentes em verbos no infinitivo, etc de forma a tentar evitar que o que seja imprimido no ficheiro output corresponda ao desejado.

Na parte correspondente à main é tratado tudo o que está relacionado com a estrutura de um ficheiro HTML. Para além disto, são impressos para o ficheiro de output os vários verbos presentes na estrutura de dados, excluindo os repetidos.

Capítulo 4

Codificação e Testes

4.1 Normalizador de Autores em BibTex

4.1.1 Alternativas, Decisões e Problemas de Implementação

xyz

4.1.2 Testes realizados e Resultados

xyzzzz

4.1.3 O programa é executado com ...

4.1.4 Obtém-se o seguinte resultado

4.2 Processador de Inglês corrente

4.2.1 Alternativas, Decisões e Problemas de Implementação

Tal como sugerido pela segunda parte do primeiro exercício prático, o ficheiro output deveria de estar no formato HTML. Tal e qual como no primeiro trabalho prático decidimos que o ficheiro deveria utilizar também estilos escritos em css e incluir um script(em javascript) para tornar possível o aparecimento de listas escondidas. Também foi utilizada a font-awesome para juntar icons que refletem o conteúdo de cada linha apresentada. O objetivo do uso destes recursos é tornar o conteúdo do ficheiro apelativo. Para além disso, poderíamos ter optado por outro tipo de estruturas de dados, como por exemplo, árvores binárias.

No que diz respeito à implementação, achamos que a o modo de resolução do problema nunca poderia ser muito diferente daquele realizado. Seria possível acrescentar mais casos que restringissem o aparecimento de outras palavras que não fossem verbos. Mas, por outro lado, a adição de mais exceções poderia levar igualmente à supressão de verbos que estariam correctos a ser apresentados no ficheiro de output.

4.2.2 Testes realizados e Resultados

xyzzzz

4.2.3 O programa é executado com ...

4.2.4 Obtém-se o seguinte resultado

Capítulo 5

Conclusão

Espetacular

Apêndice A

Código do Programa para Normalizador de Autores em BibTex

Lista-se a seguir o código do programa que foi desenvolvido.

../convAcentos.fl

```
%option noyywrap


%%
    FILE *output  = fopen("outAc.bib", "w");

    /* Acentuação */
\{?\\'A\}?     { fprintf(output, "Á"); }
\{?\\'E\}?     { fprintf(output, "É"); }
\{?\\'I\}?     { fprintf(output, "Í"); }
\{?\\'O\}?     { fprintf(output, "Ó"); }
\{?\\'U\}?     { fprintf(output, "Ú"); }
\{?\\~A\}?     { fprintf(output, "Ã"); }
\{?\\'a\}?     { fprintf(output, "á"); }
\{?\\'e\}?     { fprintf(output, "é"); }
\{?\\'i\}?     { fprintf(output, "í"); }
\{?\\'o\}?     { fprintf(output, "ó"); }
\{?\\'u\}?     { fprintf(output, "ú"); }
\{?\\~a\}?     { fprintf(output, "ã"); }
\{?\\c\{c\}\}? { fprintf(output, "ç"); }
\{?\\C\{C\}\}? { fprintf(output, "Ç"); }

    /*
        Print o que sobrou para não se
        perder info do ficheiro original
    */
.|\n { fprintf(output, "%s", yytext); }

%%


int main (int argc, char* argv[]) {
    FILE *input;
    input = fopen(argv[1], "r");
    yyin = input;
    yylex();
    return 0;
```

```
}
```

../normNomes.fl

```
%option noyywrap
%s LIMPAR

%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
%}

L       [a-z]|ú|ù|á|à|ç|é|ê|è|õ|ó|ò|ã|ì|í|\.
LM      [A-Z]|Ú|Ù|Á|À|Ç|É|É|Õ|Ó|Ò|Ã|Ì|Í
T       (author|editor)
AUX     (([\ ]+and)|[\}\"\ ]*))
N       (da|dos|de)[\ ]

%%
    int i = 0, j = 0, x;
    int aspas = 0;
    int otherTag = 0;
    char **nomesProprios, *token;
    char apelido[64], *nomes;

    nomesProprios = malloc(sizeof(char *) * 20);
    FILE *output  = fopen("out.bib", "w");

    for(x = 0; x < 20; x++)
        nomesProprios[x] = malloc(sizeof(char) * 64);

@.*  { fprintf(output, "%s", yytext); }


    /* tags author e editor */
{T}[\ ]*[=][\ ]*["] {
    aspas = 1;
    token = strtok(yytext, " =");
    while(!strcmp(token, ""))
        token = strtok(NULL, " =");
    fprintf(output, "%s = {", token);
    otherTag = 0;
}
    /* tags author e editor */
{T}[\ ]*[=][\ ]*[{] {
    token = strtok(yytext, " =");
    while(!strcmp(token, ""))
        token = strtok(NULL, " =");
    fprintf(output, "%s = {", token);
    otherTag = 0;
}

    /* print outras tags */
[^{T}][\ ]*[=][\ ]*[{"].* {
    otherTag = 1;
    fprintf(output, "%s", yytext);
}

```

```
56      /* Final com virgula */
57  [\}\"][\,] {
58      if(aspas)
59          fprintf(output, "},");
60      else
61          fprintf(output, "%s", yytext);
62      aspas = 0;
63  }
64
65      /* Final com \n */
66  [\}\"]/[\n] {
67      if(aspas)
68          fprintf(output, "}");
69      else
70          fprintf(output, "%s", yytext);
71      aspas = 0;
72  }
73
74      /* Apelidos + nomes proprios guardados */
75  ([\ ]{N})?{LM}{L}+/([\,]|([\ ]+and|[\}\"\ ]*)) {
76      if(!otherTag) {
77          if(i == 0)
78              sprintf(apelido,"%s", yytext);
79          else
80              sprintf(apelido,"%s, ", yytext);
81
82          for(j = 0; j < i; j++)
83              strcat(apelido, nomesProprios[j]);
84
85          i = 0;
86          fprintf(output, "%s", apelido);
87          BEGIN LIMPAR;
88      }
89      else fprintf(output, "%s", yytext);
90  }
91
92      /* Guardar nomes próprios atuais */
93  [A-ZÁÉÍÓÚ]/({L}+({LM}{L}+)*([\ ]+{N}?{LM}{L}+){AUX} {
94      if(!otherTag) {
95          sprintf(nomesProprios[i], "%s.", yytext);
96          i++;
97          BEGIN LIMPAR;
98      }
99      else fprintf(output, "%s", yytext);
100 }
101
102     /* Print dos 'and' */
103 [\ \n\r]and[\ \n\r] { fprintf(output, " and "); }
104
105     /* Tratar casos de nomes do tipo: X. X. YYYY */
106 ({LM}{L}+[\ ]?)+[\,][\ ]*{LM}{L}+([\ ]+{LM}{L}+)* {
107     if(!otherTag) {
108         token = strtok(yytext, ",");
109         nomes = strtok(NULL, ",");
110         nomes = strtok(nomes, " ");
111         fprintf(output, "%s, ", token);
112         while(nomes != NULL) {
113             int ascii = nomes[1];
114
```

```
            //Nao é letra (<65 >122) e nao é ponto (46)
            //Serve pra encontrar casos com acento
            if((ascii < 65 || ascii > 122) && ascii != 46)
                fprintf(output, "%c%c.", nomes[0], nomes[1]);
            else
                fprintf(output, "%c.", nomes[0]);
            nomes = strtok(NULL, " ");
        }
    }
    else fprintf(output, "%s", yytext);
}


    /* Limpar restos dos matches de nomes proprios */
<LIMPAR>{L}|[\ ] { }
<LIMPAR>\n {
    fprintf(output, "%s", yytext);
    BEGIN(INITIAL);
}

    /*
        Print o que sobrou para não se
        perder info do ficheiro original
    */
.|\n { fprintf(output, "%s", yytext); }


%%


int main (int argc, char* argv[]) {
    FILE *input;
    input = fopen(argv[1], "r");
    yyin = input;
    yylex();
    return 0;
}
```

../genGraph.fl

```
%option noyywrap

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

char* multi_tok(char *input, char *delimiter);
%}

L       [a-z]|ú|ù|á|à|ç|é|è|õ|ó|ò|ã|ì|í|\.
LM      [A-Z]|Ú|Ù|Á|À|Ç|É|É|Õ|Ó|Ò|Ã|Ì|Í
N       (da|dos|de)[\ ]
NAME    {N}?({LM}{L}+)+([\,][\ ]({LM}[\.])+)?

%%

  FILE *output  = fopen("Graph/graph.dot", "w");
  char **names, *token;
```

```
20    char format[64] = "\"%s\" -- \"%s\"\n";
21    int flag = 0, i, j;
22    int start = 1;
23    names = (char **) malloc(sizeof(char *) * 20);
24
25  (author|editor) { flag = 1; }
26
27  {NAME}([\ ]+and[\ ]+{NAME})* {
28    if(start) {
29      fprintf(output, "strict graph G{\n");
30      fprintf(output, "ranksep=\"1.0 equally\"\n");
31      fprintf(output, "bgcolor=lightsteelblue4;\n");
32      fprintf(output, "edge [color=grey94];\n");
33      fprintf(output, "node [shape=box,style=filled,color=grey];\n");
34      start = 0;
35    }
36
37    if(flag) {
38      i = 0;
39      token = multi_tok(yytext, " and ");
40      names[i++] = strdup(token);
41      token = multi_tok(NULL, " and ");
42      while(token) {
43        j = 0;
44        while(j < i)
45          fprintf(output, format, names[j++], token);
46        names[i++] = strdup(token);
47        token = multi_tok(NULL, " and ");
48      }
49    }
50
51    flag = 0;
52
53  }
54
55  \n|. {}
56
57  <<EOF>>   fprintf(output, "}\n"); return 0;
58
59
60  %%
61
62  char* multi_tok(char *input, char *delimiter) {
63      static char *string;
64      if (input != NULL)
65          string = input;
66
67      if (string == NULL)
68          return string;
69
70      char *end = strstr(string, delimiter);
71      if (end == NULL) {
72          char *temp = string;
73          string = NULL;
74          return temp;
75      }
76
77      char *temp = string;
78
```

```
79      *end = '\0';
80      string = end + strlen(delimiter);
81      return temp;
82  }
83
84  int main (int argc, char* argv[]) {
85      FILE *input;
86      input = fopen(argv[1], "r");
87      yyin = input;
88      yylex();
89      return 0;
90  }
```

Apêndice B

Código do Programa para Processador de Inglês corrente

../procIngles.fl

```
%option noyywrap
%{
#include <glib.h>
#include <string.h>
#include <ctype.h>
GSList* list;
GSList* iterator;
FILE *outV;
FILE *Contracoes;
%}

L           [a-z]|ú|ù|á|à|ç|é|è|õ|ó|ò|ã|ì|í
LM          [A-Z]|Ú|Ù|Á|À|Ç|É|É|Õ|Ó|Ò|Ã|Ì|Í
SU          i|you|he|she|it|we
OB          me|him|her[s]?|it[s]?|your[s]?|us|them
PP          my|his|mine|their[s]?|our[s]?
AV          how|then|more|also|here|there|still|never|when|really|back|just|by
C           for|and|nor|but|or|yet|so|if|after|before|even|in|once|now
CC          only|until|while|when|where|thought|till|whe[a-z]+
RP          that|which|who|whom|whose|whichever|who[a-z]+
A           a|an
DP          the|this|those|these
IP          any|most|none|both|many|several|anybody|anyone|anything|each
IPP         either|everbody|everyone|neither|nobody|nothing|some[a-z]+
O           not|[^a-z]|from|new|with
MV1         [Cc]an[’n]?|[Cc]ould[n’t]?|[Ss]hall|[Ss]hant|[Ss]hould[n’t]?
MV2         [Ww]ill|[Ww]on’t|[Ww]ould[n’t]?|[Mm]ay[n’t]?|[Mm]ight[n’t]?
N           1|2|3|4|5|6|7|8|9|0
NE          one|two|three|four|five|six|seven|eight|nine
AU          [Dd]id[n’t]?|[Dd]o[n’t]?|[Dd]oes[n’t]?
Q           all|some|several|enough|numerous|few|another|same
T           ([a-z]+[ijquv])|([a-z]+[rfz][fz])|([a-z]+(ing|ed|lly|[^s]s|es))
EXCEPT1     {OB}|{SU}|{A}|{O}|{N}+|{NE}([a-z]+)?|{Q}|{AV}|{Q}|{C}
EXCEPT2     {CC}|{RP}|{DP}|{PP}|{IP}|{IPP}|{T}

%%

  Contracoes = fopen("Contracoes.txt", "w");
```

```
39    int j, i = 0;
40    char *token;
41    char* args[10];
42
43
44  'm                      { fprintf(Contracoes, " am"); }
45  're                     { fprintf(Contracoes, " are"); }
46  's                      { fprintf(Contracoes, " is/has"); }
47  've                     { fprintf(Contracoes, " have"); }
48  'll                     { fprintf(Contracoes, " will"); }
49  'd                      { fprintf(Contracoes, " had/would"); }
50  can't                   { fprintf(Contracoes, "cannot"); }
51  won't                   { fprintf(Contracoes, "will not"); }
52  shan't                  { fprintf(Contracoes, "shall not"); }
53  {LM}?{L}+/n[\']t        { fprintf(Contracoes, "%s not", yytext); }
54
55  [Tt]o[\ ]({EXCEPT1}|{EXCEPT2}) {
56
57      fprintf(Contracoes, "%s", yytext);
58
59  }
60
61  [Tt]o[\ ][a-z]+ {
62
63      list = g_slist_insert_sorted(list,
64      strdup(yytext+3),
65      (GCompareFunc)g_ascii_strcasecmp);
66      fprintf(Contracoes, "%s", yytext);
67
68  }
69
70
71  ({MV1}|{MV2})[\ ]({EXCEPT1}|{EXCEPT2}) {
72
73      fprintf(Contracoes, "%s", yytext);
74
75  }
76
77  ({MV1}|{MV2})[\ ][a-z]+ {
78
79      token     = strdup(yytext);
80      args[i]   = strtok(token, " ");
81      args[++i] = strtok(NULL, " ");
82
83      list = g_slist_insert_sorted(list, args[1],
84      (GCompareFunc)g_ascii_strcasecmp);
85      i = 0;
86
87      fprintf(Contracoes, "%s", yytext);
88
89  }
90
91  {AU}[\ ][a-zA-Z]+[\ ]({EXCEPT1}|{EXCEPT2}) {
92
93      fprintf(Contracoes, "%s", yytext);
94
95  }
96
97
```

```
{AU}[\ ][a-zA-Z]+[\ ][a-z]+ {

    token   = strdup(yytext);
    args[i] = strtok(token, " ");

    while(args[i] != NULL)
        args[++i] = strtok(NULL, " ");

    list = g_slist_insert_sorted(list, args[2],
    (GCompareFunc)g_ascii_strcasecmp);
    i = 0;
    fprintf(Contracoes, "%s", yytext);

}


n[\']t          { }
.|\n            { fprintf(Contracoes, "%s", yytext); }

%%


int main (int argc, char* argv[]) {
    FILE *input;
    int z = 0;
    char *auxiliar = malloc(sizeof(char)*100), *aux;
    char format[128], search[128], book[64];
    char arrow[64], dict[256], pencil[64];
    char letter, ultima = ' ';
    strcpy(format, "<a onClick='oL(\"%c\")'><p> Letter %c%s </p></a>\n");
    strcpy(search, "<a href='%s%s'%s><i class='fa fa-search'></i></a></li>\n");
    strcpy(arrow, "<a><i class='fa fa-arrow-right'></i></a>");
    strcpy(dict, "https://dictionary.cambridge.org/dictionary/english/");
    strcpy(pencil, "<i class='fa fa-pencil'> </i>");
    strcpy(book, "<i class='fa fa-book'></i>");

    iterator = NULL;
    list  = NULL;
    input = fopen(argv[1], "r");
    outV  = fopen("verbos.html","w");
    Contracoes = fopen("Contracoes.txt", "w");
    yyin = input;
    yylex();

    fprintf(outV, "<!DOCTYPE html> <html> <head>\n");
    fprintf(outV, "<title> Ingles </title>\n");
    fprintf(outV, "<link rel='stylesheet' href='css/styles.css'>\n");
    fprintf(outV, "<link rel='stylesheet' href='css/fa/css/font-awesome.css'>\n");
    fprintf(outV, "<script src='js/script.js'></script>\n");
    fprintf(outV, "<meta charset='utf-8'/> </head> <body>\n");
    fprintf(outV, "<h1 class='bg-4 title'> Verbs %s</h1>\n",book);


    for (iterator = list; iterator; iterator = iterator->next) {

        if(strcmp(auxiliar,iterator->data) != 0) {
            aux = (char *)iterator->data;

            if(ultima != aux[0]) {
```

```
157                 fprintf(outV, "</ul>");
158                 letter = toupper(aux[0]);
159                 fprintf(outV, format, letter, letter, pencil);
160                 fprintf(outV, "<ul id='%c'>\n", letter);
161                 ultima = aux[0];
162             }
163 
164             fprintf(outV, "<li> %s ", aux);
165             fprintf(outV, "%s", arrow);
166             fprintf(outV, search, dict, aux, "target='_blank'");
167         }
168 
169         auxiliar = iterator->data;
170     }
171     fprintf(outV, "</body> </html>\n");
172     return 0;
173 }
```