Processamento de Linguagens (3ºano de Curso)

Trabalho Prático nº2

Relatório de Desenvolvimento

Carlos Pedrosa (a77320)
David Sousa (a78938)
Manuel Sousa (a78869)

7 de Maio de 2018

Resumo

Este projeto tem como principal objetivo a interação por parte dos alunos com ferramentas de apoio à programação. Neste sentido, permite assim, aumentar a capacidade destes relativamente à escrita de expressões regulares como motor para a filtração e transformação de textos. Em suma, este relatório pretende sumarizar todos os esforços efetuados para alcançar o objetivo proposto, focando assim na utilização da ferramenta *Flex*. Efetivamente, começamos por desenvolver os ficheiros necessários para responder a todas as questões propostas referentes ao BibTex, uma ferramenta de formatação de citações bibliográficas. Passaremos pela criação de vários ficheiros *HTML* terminado na construção de um grafo, usando para o efeito o *GraphViz*.

Conteúdo

Capítulo 1

Introdução

1.1 Estrutura do Relatório

O enunciado foi atribuído de acordo com o menor número de aluno de entre todos os elementos do grupo. Nesse sentido, o presente relatório debruçar-se-à sobre o trabalho número um. Primeiramente, introduziremos o problema proposto, passaremos então pela conceção da solução e por último faremos uma análise crítica a todo o trabalho elaborado. Na secção enunciada como problema proposto, apresentaremos o problema retratando um pouco o formato sobre o qual teremos de extrair informação. Na conceção da solução, iremos abordar a metodologia usada na resposta às questões enunciadas, seremos assim exaustivos na explicação das soluções que apresentamos. Por último, na análise crítica enumeraremos as dificuldades sentidas, referindo ainda, a forma como estas foram ultrapassadas.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

O nosso problema prático aborda BibTex, uma ferramenta de formatação de citações bibliográficas em documentos Latex. Tendo como principal objetivo o de facilitar a separação da base de dados da bibliográficos, apresenta uma estrutura bastante peculiar. A nossa intenção é a de nos aproveitarmos da semântica que BibTex proporciona para assim respondermos às questões que nos são propostas. Sendo assim, foi-nos apresentado um enunciado composto por diversas questões. [1] Cada questão debatia-se com problemas de extração de informação dos ficheiros *exemplo-latin1.bib* e *exemplo-utf8.bib* fornecidos pelos docentes. Para além disso, em cada questão eram abordados desafios de criação de novos formatos de ficheiros, como por exemplo *HTML* e *GraphViz*. [2]

Capítulo 3

Conceção/desenho da Resolução

3.1 Estruturas de Dados

As estruturas de dados são um modo de armazenamento e organização de dados, de modo, que podem ser usadas de forma eficiente, facilitando assim a busca e modificação. Nesse sentido, como era necessária uma estrutura de dados relativamente eficiente optamos por utilizar uma biblioteca do sistema, a GLib. Esta pareceu-nos uma biblioteca bastante adequada pois contém uma série de estruturas já implementadas. Assim, a utilização das estruturas que envolviam a manipulação de *Strings* facilitaram todo o trabalho.

3.2 Respostas às questões enunciadas

- **Alínea a)**

A resolução da alínea a) surgiu de forma bastante natural. Começamos por definir as expressões regulares para a extração correta dos dados. Nesta alínea apenas é necessária a extração das várias categorias em BibTex. Sendo estas identificadas por um "@" desenvolvemos a seguinte expressão regular: @.*/{ . Assim, depois de toda a informação ser extraída era necessário um processamento, que permitisse ir ao encontro do que o enunciado propunha. Visto que o objetivo era a contagem do número de categorias foi necessário o desenvolvimento de funções em C que o permitissem. Surge assim a função *void existeCat(char* pal)* . Esta à medida que o ficheiro vai sendo percorrido verifica se a categoria encontrada é diferente das demais. Caso seja, é guardado num *array* referente às categorias, caso não o seja, soma-se um à posição do *array* que trata dessa mesma categorias. Por fim, o objetivo foi o de construir um *HTML* com os dados obtidos, optamos por construir uma tabela.

- **Alínea b)**

A segunda alínea surgiu de forma semelhante à anterior. Neste caso, tivemos de ter atenção que a quantidade de informação a ser recolhida era maior. Assim, guardamos dados referente ao código, ao título e aos autores dos vários artigos. Usamos assim, como expressões regulares:

```
[Aa][Uu][Tt][Hh][Oo][Rr]\ *=.*/,
```

```
\ [Tt][Ii][Tt][Ll][Ee]\ *=.*/,
```

Sendo que cada tag se encontra separada por uma vírgula foi esta a condição fulcral em que nos baseamos. É ainda importante realçar o cuidado que tivemos para que o "match" da expressão regular não engloba-se o booktitle também indicado no artigo. Para além disso, o "." faz

"match" até ao final da linha, uma vez que se encontra conjugado com o asterisco(0 ou muitos). Deste modo, conseguimos extrair sem problemas casos em que existam vírgulas na nomeação dos diferentes títulos. Por último associamos o resultado desta alínea com o resultado da alínea anterior, obtendo assim uma tabela com diferentes hiperligações que permitem obter a informação associada a cada categoria.

- **Alínea c)**

A alínea c) já exigiu um maior processamento dos documentos fornecidos. Consideramos que na criação de um índice de autores, este deveria surgir associado aos códigos identificando, assim, os respetivos registos em que estes surgem. No entanto, um artigo pode possuir vários autores, separados pela palavra "and" ou simplesmente pelo sinal de pontuação ",". Assim, separamos cada um destes criando, deste modo, um índice mais verossímil. Todo este processamento foi efetuado com auxílio da biblioteca *GLib*. Para além disso, referir que utilizamos o identificador «EOF» que permitiu executar algumas funções no final do ficheiro. É ainda importante mencionar que procuramos explorar as potencialidades do *Flex* pelo que o processamento da informação era feito à medida que esta ia sendo extraída. Contudo, é de sublinhar que os dados necessitavam de tratamento, por esse facto se justifica as funções implementadas em C. Como era necessário um padrão que facilita-se futuras ferramentas de procura optamos por usar o símbolo de maior (>) para a separação entre o nome do autor e os códigos dos arquivos onde este publicou.

- **Alínea d)**

Na alínea d) mais uma vez extraímos os dados necessários. Nesta alínea era pedida a construção de um grafo que demonstrasse a interação dos vários autores nos vários artigos. Sendo que um autor pode publicar com um outro um elevado número de vezes decidimos agilizar o processo, definindo um número mínimo para o qual era considerado habitual a cooperação dos autores com o que será passado como argumento. Para o efeito, definimos um número mínimo para dar resposta a este problema. Assim, simplificaremos o grafo que será desenhado tornando mais percetível os seus nodos. A extração dos autores foi semelhante à efetuada na alínea anterior, no entanto, estes agora, são guardados num *HashTable* associado ao número de vezes que determinado autor publica com os demais.

Capítulo 4

Testes

4.1 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (com o ficheiro *exemplo-utf8.bib*) e os respetivos resultados obtidos:

- **Alínea a)**

Tabela de Categorias

Categoria	Repetições
@string	2
@inproceedings	50
@techreport	11
@article	10
@InProceedings	59
@phdthesis	1
@MastersThesis	1
@inProceedings	2
@InCollection	3
@Book	1
@Incollection	2
@Article	23
@misc	1
@INPROCEEDINGS	1

Figura 4.1: Ficheiro produzido para a alínea a).

- **Alínea b)**

@inproceedings	ABBN98	Almeida, J.J. and Barbosa, L.S. and Barros, J.B. and Neves, L.F.	Adapting Museum Structures for the Web: No Changes Needed!
@inproceedings	Gis99	Jorge Rocha and Ana Silva and Ricardo Henriques and J.J. Almeida and Pedro Henriques	Algebraic specification of documents
@inproceedings	RPA99	Jorge Rocha and Tiago Pedroso and J.J. Almeida	Formal Methods for {GI} Systems Development
@inproceedings	RSea99	Jorge Gustavo Rocha and Ana Silva and J.J. Almeida and Mario Ricardo Henriques and Pedro Rangel Henriques	{MAPit} - A tool set for automatically generation of {HTML} Maps

Figura 4.2: Ficheiro produzido para a alínea b).

- **Alínea c)**

Para a alínea C usamos uma ferramenta de procura que permitiu demonstrar que a forma como escrevemos o índice torna a pesquisa facilitada.

```
Pedro Rangel Henriques >    RSea99 RRH02 ORH06 RMHV06 RMHCV06 BH98 RAH98 RARH98 GRH06 JGRH04 JGRH03 GRH04 GRH05a RH98a MSH05 ALHF02
VH01 RMHVC06 BHVU07d BHVU07c BHVU07b BHVU07a CHLB07a BHVU06a BHVU06b BHVUM06 BHVU08 BHV06 BCVHU08 BUHV08 OVH05 OHV06 GH07b GH07a
GFH08 FGH08 FH08 LPRH07 LPRH07-TM LRHGT08 LGFSSAH08 LMMVRH08 CHV08ja FPCH08jb PMCH08j CHV07 CPH07f CH07g CH07h CHP08i CH07a CLH07c
CH07d CFPBH07d CHP08a CHP08b CPH08c FPCH08d PMCH08e PMCH08f OPCH09a FCHGD09a ORH09a LPH09a LARH09 OPHC09a OPHCC09 MKCHCPO09 OHCP09
FCHGD09b BHVU09 CBHP09 cruz09 OPHCC2010 KOMPCCH2010
Rui Vilela >    linguateca xata05:fs
Elisabete Cunha >   CHLB07a
```

Figura 4.3: Ficheiro produzido para a alínea c).

Assim, por exemplo se quiséssemos obter o número de vezes que o docente Pedro Rangel Henriques publica apenas teríamos de fazer:

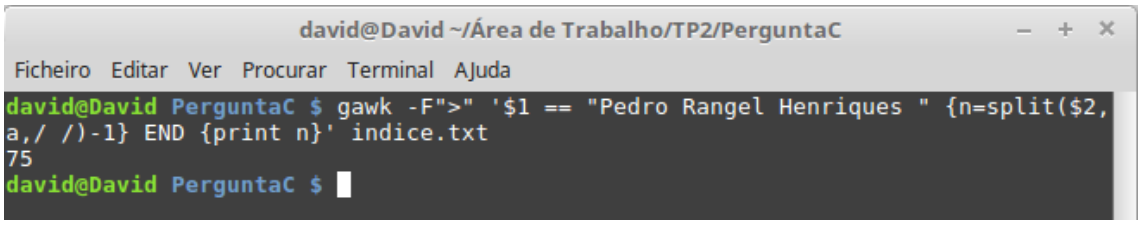

Figura 4.4: Comando exemplo usado para a alínea c).

- **Alínea d)**

Para efeito de teste usamos o nome "Pedro Rangel Henriques" pelo que o grafo produzido foi o seguinte:

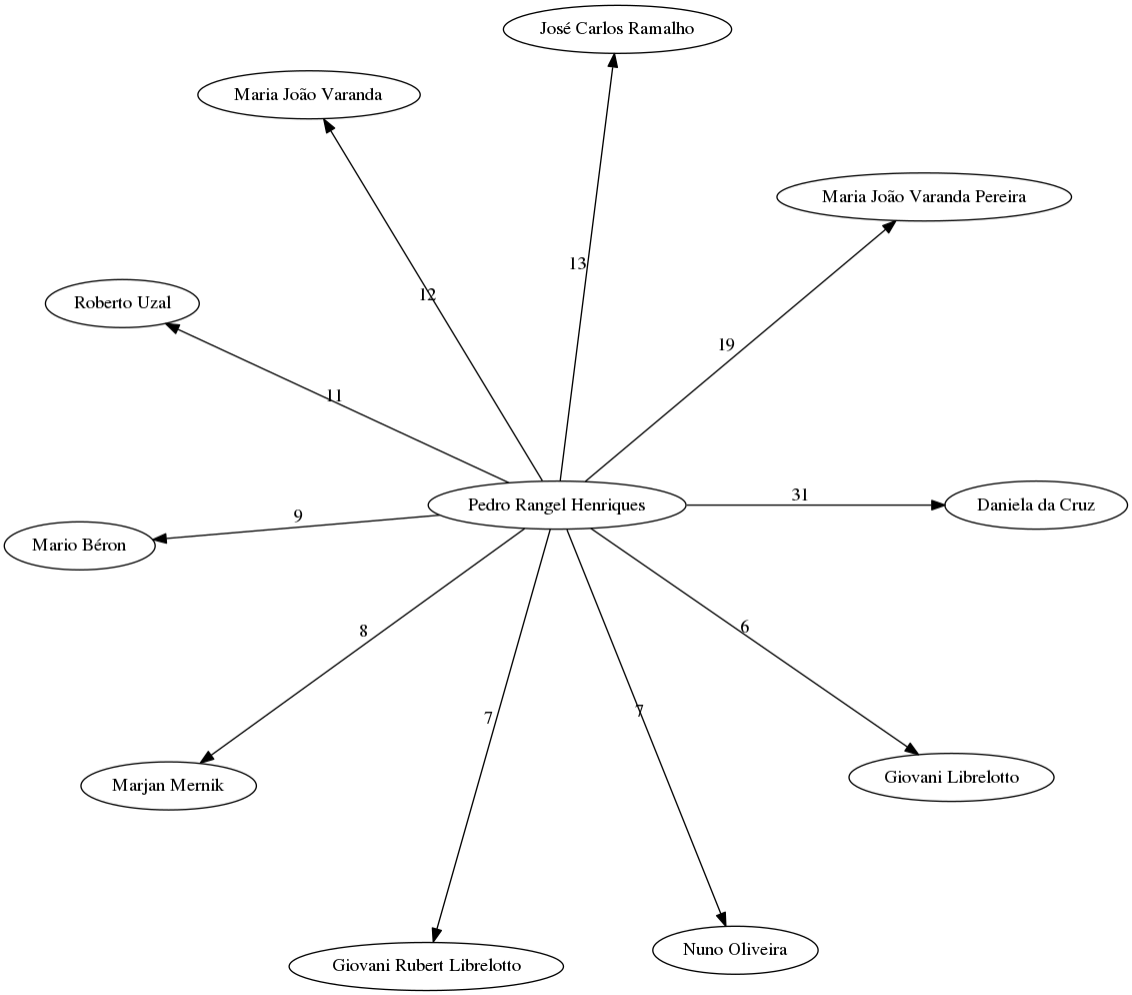

Figura 4.5: Grafo gerado para “Pedro Rangel Henriques” d).

Capítulo 5

Conclusão

De acordo com o trabalho pretendido consideramos que o principal objetivo foi atingido. De facto, foi possível por parte dos alunos um maior contacto com a ferramenta *Flex*.

No entanto, no decorrer da resolução do projeto, foram surgindo alguns desafios que foram ultrapassados. Assim, o principal obstáculo à resolução do trabalho proposto foi a flexibilidade que existe na escrita de um ficheiro *BibTex*. Efetivamente, este tipo de ficheiros, não são muito rigorosos na forma como são desenvolvidos pelo que foi uma tarefa árdua encontrar padrões que permitissem uma correta extração dos dados. Para além disso, a conjugação do *Flex* com a linguagem de programação C (principalmente no uso de *Strings*) também dificultou um pouco todo o processo. Apesar de todos os obstáculos conseguimos desenvolver ficheiros coerentes e concisos que permitiram dar resposta às questões requeridas.

Em suma, o trabalho foi bastante benéfico para todos os elementos do grupo, sendo que permitiu entender os pressupostos da ferramenta *Flex*.

Apêndice A

Código dos Programas

Lista-se a seguir o código dos programas que foram desenvolvidos. Como já referido, ao longo da resolução dos ficheiros *Flex*, foi necessária a criação de funções em C que permitissem moldar a informação que ia sendo extraída. No entanto, sendo que o foco do trabalho era a ferramenta *Flex*, apenas incluímos em anexo as respetivas funções C desenvolvidas.

- **Alínea a)**

```
%option noyywrap

%{
#include <string.h>

char* cat[20];
int count[20];

void existeCat(char* pal);
void printInfo();
%}


%%

\@.*/\{      { existeCat(yytext); }

.|\n         { }

<<EOF>>      { printInfo(); return 0; }

%%


void existeCat(char* pal)
{
    int flag = 1;
    int i = 0;

    while (cat[i] != NULL && flag) {
        if (strcmp(pal, cat[i]) == 0) {
            count[i]++;
            flag = 0;
        }
        i++;
    }

    if (flag == 1) {
        cat[i] = (char*)malloc(sizeof(char) * 15);
        count[i]++;
        strcpy(cat[i], pal);
```

```
    }
}

void printInfo()
{
    int i = 0;
    FILE* f = fopen("HTML/tabelaCat.html", "w");

    fprintf(f, "<h1> Tabela de Categorias </h1>");
    fprintf(f, "<html><head><style>\n");
    fprintf(f, "table { border-collapse: collapse; width 100%%; }\n");
    fprintf(f, "th, td { text-align: left; padding: 8px; }\n");
    fprintf(f, "tr:nth-child(even){ background-color: #f2f2f2 }\n");
    fprintf(f, "th { background-color: #3A80F2; color: white; }\n");
    fprintf(f, "</style><meta charset='ISO-88591'> </head><body>\n");
    fprintf(f, "<table><tr> <th>Categoria</th> <th>Repetições</th> </tr>");


    while (cat[i] != NULL) {
        char* aux = (char*)malloc(sizeof(char) * 50);
        strcpy(aux, "HTML/");
        strcat(aux, cat[i] + 1);
        strcat(aux, ".html");

        FILE* f1 = fopen(aux, "a");

        fprintf(f1, "<h1 style='color:blue;'> <i> Lista  </i> </h1> \n");
        fprintf(f1, "<html><head><title>Lista </title><meta charset='ISO-88591'> </head> \n");

        if (i == 0) {
            fprintf(f, "<tr><td> %s </td> <td> %d </td></tr>", cat[i], count[i]);
        } else {
          fprintf(f, "<tr><td><a href = %s> %s </a> </td> <td> %d </td>", aux + 5, cat[i], count[i]);
          fprintf(f1, "<html><head><style>\n");
          fprintf(f1, "table { border-collapse: collapse; width 100%%; }\n");
          fprintf(f1, "th, td { text-align: left; padding: 8px; }\n");
          fprintf(f1, "tr:nth-child(even){ background-color: #f2f2f2 }\n");
          fprintf(f1, "th { background-color: #3A80F2; color: white; }\n");
          fprintf(f1, "</style><meta charset='ISO-88591'> </head><body>\n");
          fprintf(f1, "<table><tr> <th>Categoria</th> <th>Código</th> <th>Autores</th> <th>Título</th> </tr>");
        }

        free(aux);
        i++;
    }

    fprintf(f, "</table></body></html>");
}

int main(int argc, char** argv)
{
    int i = 0;

    for (i = 0; i < 20; i++) {
        cat[i] = NULL;
        count[i] = 0;
    }

    yylex();
}
```

- **Alínea b)**

```
%option noyywrap

%{
```

```
int flag = 0;
int indice = 0;
char* cat[200];
char* chave[200];
char* autor[200];
char* titulo[200];
int x = 0;
void printInfo();
int contaN(char* palavra);
void separaString(char* str, char* vSep);
void printHTML();
%}


%%

\@[^ ]*/,                           { separaString(yytext, "{"); }

[Aa][Uu][Tt][Hh][Oo][Rr]\ *=.*/,    {
                                      int sum = contaN(yytext);
                                      int tam = strlen(yytext);

                                      if (flag == 0) {
                                        strncpy(autor[indice], (yytext + sum + 1), (tam - sum - 2));
                                        flag = 1;
                                      }

                                      if (flag == 2) {
                                        strncpy(autor[indice], (yytext + sum + 1), (tam - sum - 2));
                                        indice++;
                                        flag = 0;
                                      }
                                    }

\ [Tt][Ii][Tt][Ll][Ee]\ *=.*        {
                                      int sum = contaN(yytext);
                                      int tam = strlen(yytext);

                                      if (flag == 0) {
                                        strncpy(titulo[indice], (yytext + sum + 1), (tam - sum - 2));
                                        flag = 2;
                                      }

                                      if (flag == 1) {
                                        strncpy(titulo[indice], (yytext + sum + 1), (tam - sum - 2));
                                        indice++;
                                        flag = 0;
                                      }
                                    }

.|\n                                { }

<<EOF>>                             { printHTML(); return 0; }

%%


void printHTML()
{
    int flag = 1;
    int i = 0;

    FILE* f;

    for (i = 0; i < indice; i++) {
        char* aux = (char*)malloc(sizeof(char) * 50);
        strcpy(aux, "./HTML/");
```

```
        strcat(aux, cat[i] + 1);
        strcat(aux, ".html");

        f = fopen(aux, "a");

fprintf(f, "<tr><td> %s </td> <td> %s </td> <td> %s </td> <td> %s </td> </tr>\n", cat[i] , chave[i], autor[i], titulo[i]);

        /*
        fprintf(f, "<p><b> Categoria:</b> %s </p>", cat[i]);
        fprintf(f, "<p><b> Código:</b> %s </p>", chave[i]);
        fprintf(f, "<p><b> Título:</b> %s </p>", titulo[i]);
        fprintf(f, "<p><b> Autores:</b> %s </p>", autor[i]);
        fprintf(f, "<hr>");
        */

        fclose(f);
    }
    fprintf(f,"</table></body></html>");
}


void printInfo()
{
    int i = 0;

    while (i < indice)  {
        printf("Cat: %s \n || Codigo: %s \n || Autor: %s \n || Titulo: %s \n", cat[i], chave[i], autor[i], titulo[i]);
        i++;
    }
}

void separaString(char* str, char* vSep)
{
    char* token;
    token = strstr(str, vSep);

    int x = token - str;

    str[x] = '\0';

    strcpy(cat[indice], str);
    strcpy(chave[indice], token + 1);
}

int contaN(char* palavra)
{
    int i = 0;

    while (palavra[i] != '\0') {
        if (palavra[i] != '{' && palavra[i] != '"') {
            i++;
        } else {
            break;
        }
    }

    return i;
}


int main(int argc, char** argv)
{
    int i = 0;

    for (i = 0; i < 200; i++) {
        cat[i] = (char*) malloc(sizeof(char) * 15);
        chave[i] = (char*) malloc(sizeof(char) * 10);
```

```
        autor[i] = (char*) malloc(sizeof(char) * 200);
        titulo[i] = (char*) malloc(sizeof(char) * 150);
    }

    yylex();
}
```

- **Alínea c)**

```
%option yylineno

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

char token[10000];

GString* author;
GString* authors;
GString* code;

GHashTable* indice;

void processaAuthor(char* s);
void printNode(gpointer key, gpointer value, gpointer user_data);
%}


%%

\@[^ ]*/,{ code = g_string_new(strstr(yytext, "{") + 1); }

author[ ]*=.* { processaAuthor(yytext); g_string_erase(code, 0, -1); }

.|\n { }

<<EOF>>{ g_hash_table_foreach(indice, (GHFunc)printNode, NULL); return 0; }

%%


void printNode(gpointer key, gpointer value, gpointer user_data)
{
FILE* f = fopen("indice.txt", "a");

GSList* regs = (GSList*)value;

GString* s = (GString*)key;
fprintf(f, "%s >\t", (char*)s);

while (regs != NULL) {
GString* tmp = regs->data;
fprintf(f, "%s ", tmp->str);

regs = regs->next;
}
fprintf(f,"\n");
}

void cleanToken(char token[], GString* authors)
{
int k = 0;

// Clean token - get only the authors.
while (token[k] != '{' && token[k] != '"') {
```

```
     k++;
    }

k++;

    while ( (token[k] != '}' || token[k] != '"') && token[k + 1] != ',') {
     authors = g_string_append_c(authors, token[k]);
        k++;
    }

    // Insert final char in the authors string.
    authors = g_string_append_c(authors, '\0');
}

void insertAuthor(GString* authors)
{
int i = 0;

// Get all the individual authors.
while (authors->str[i] != '\0') {
if (authors->str[i + 1] == '\0' || (authors->str[i] == ' ' && (authors->str[i + 1] == 'a' || authors->str[i + 1] == ','))){
if (authors->str[i + 1] == '\0') {
author = g_string_append_c(author, authors->str[i]);
author = g_string_append_c(author, authors->str[i + 1]);
}

if (author->str[author->len - 1] == ' ') {
author->str[author->len - 1] = '\0';
} else {
g_string_append_c(author, '\0');
}

gboolean auth = g_hash_table_contains(indice, author->str);
if (auth == FALSE) {
GSList* new_regs = NULL;
new_regs = g_slist_append(new_regs, g_string_new(code->str));

GString* key = g_string_new(author->str);
g_hash_table_insert(indice, key->str, new_regs);
} else {
GSList* old_regs = (GSList*)g_hash_table_lookup(indice, author->str);
GSList* n = g_slist_append(old_regs, g_string_new(code->str));
}

g_string_erase(author, 0, -1);

if (authors->str[i + 1] == '\0') {
i++;
} else {
i += 4;
}
} else {
     author = g_string_append_c(author, authors->str[i]);
}

i++;
}

g_string_erase(author, 0, -1);
}

void processaAuthor(char* s)
{
strcpy(token, s);
cleanToken(token, authors);

    insertAuthor(authors);
```

```
    g_string_erase(authors, 0, -1);
}

int main(int argc, char** argv)
{
authors = g_string_new(NULL);
author = g_string_new(NULL);
indice = g_hash_table_new(g_str_hash, g_str_equal);

yylex();

return 0;
}
```

- **Alínea d)**

```
%option yylineno

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

typedef struct cont {
int reps;
char* authorName;
} Cont;

char token[10000];

GString* mainAuthor;
GString* author;
GString* authors;

GList* auths;
GHashTable* contributors;

void writeNodeToFile(GList* auths);
gint compareReps(gconstpointer a, gconstpointer b, gpointer user_data);
void addToList(gpointer key, gpointer value, gpointer user_data);
void extractNodes(GString* authors);
void cleanToken(char token[], GString* authors);
void processaAuthors(char* s);
%}


%%

author[ ]*=.*   { processaAuthors(yytext); }

.|\n            { }

<<EOF>>         {
  g_hash_table_foreach(contributors, (GHFunc)addToList, NULL);
  auths = g_list_sort_with_data(auths, (GCompareDataFunc)compareReps, NULL);

  writeNodeToFile(auths);

  FILE* f = fopen("graph.gv", "a");
  fprintf(f, "}\n");
  fclose(f);
  return 0;
}

%%
```

```
void writeNodeToFile(GList* auths)
{
int i = 0;
    FILE* f = fopen("graph.gv", "a");

while (auths != NULL && i < 10) {
Cont* tmp = (Cont*)auths->data;

fprintf(f, "    \"%s\" -> \"%s\" [label=\"%d\"];\n", mainAuthor->str, tmp->authorName, tmp->reps);

i++;
auths = auths->next;
}

    fclose(f);
}

gint compareReps(gconstpointer a, gconstpointer b, gpointer user_data)
{
Cont* aux_1 = (Cont*) a;
Cont* aux_2 = (Cont*) b;

if (aux_1->reps < aux_2->reps) return 1;
else if (aux_1->reps > aux_2->reps) return -1;
 else return 0;
}

void addToList(gpointer key, gpointer value, gpointer user_data)
{
Cont* c = (Cont*)malloc(sizeof(struct cont));
c->authorName = (char*)malloc(sizeof(char) * 30);

strcpy(c->authorName, (char*)key);
c->reps = GPOINTER_TO_INT(value);

auths = g_list_prepend(auths, c);
}

void extractNodes(GString* authors)
{
int i = 0;

    if (strstr(authors->str, mainAuthor->str) != NULL) {
        // Get all the individual authors.
        while (authors->str[i] != '\0') {
if (authors->str[i + 1] == '\0' || (authors->str[i] == ' ' && (authors->str[i + 1] == 'a' || authors->str[i + 1] == ','))){
             if (authors->str[i + 1] == '\0') {
                    author = g_string_append_c(author, authors->str[i]);
                    author = g_string_append_c(author, authors->str[i + 1]);
               }

               if (author->str[author->len - 1] == ' ') {
                    author->str[author->len - 1] = '\0';
               } else {
                    g_string_append_c(author, '\0');
               }

               // Process the author that was extracted, if it is different from the main one.
if (strcmp(author->str, mainAuthor->str) != 0) {
GString* key = g_string_new(author->str);

gboolean auth = g_hash_table_contains(contributors, author->str);
if (auth == FALSE) {
int count = 1;
g_hash_table_insert(contributors, key->str, GINT_TO_POINTER(count));
```

```
} else {
gpointer value = g_hash_table_lookup(contributors, author->str);
value++;
g_hash_table_insert(contributors, key->str, value);
}
                }

                g_string_erase(author, 0, -1);

                if (authors->str[i + 1] == '\0') {
                    i++;
                } else {
                    i += 4;
                }
            } else {
                author = g_string_append_c(author, authors->str[i]);
            }

            i++;
        }

        g_string_erase(author, 0, -1);
    }
}

void cleanToken(char token[], GString* authors)
{
    int k = 0;

    // Clean token - get only the authors.
    while (token[k] != '{' && token[k] != '"') {
        k++;
    }

    k++;

    while ( (token[k] != '}' || token[k] != '"') && token[k + 1] != ',') {
        authors = g_string_append_c(authors, token[k]);
        k++;
    }

    // Insert final char in the authors string.
    authors = g_string_append_c(authors, '\0');
}

void processaAuthors(char* s)
{
strcpy(token, s);
cleanToken(token, authors);

    extractNodes(authors);

g_string_erase(authors, 0, -1);
}

int main(int argc, char** argv)
{
if (strcmp(argv[1], "") == 0) {
printf("\n=> Error. Main Author not found!\n");
return -1;
}

FILE* f = fopen("graph.gv", "a");
fprintf(f, "digraph G {\n");
fprintf(f, " layout=\"circo\";\n");
fclose(f);
```

```
    mainAuthor = g_string_new(argv[1]);
authors = g_string_new(NULL);
author = g_string_new(NULL);
auths = NULL;
contributors = g_hash_table_new(g_str_hash, g_str_equal);

yylex();

return 0;
}
```

Bibliografia

[1] http://www.bibtex.org/ acedido no dia 3 de Maio pelas 16h03m

[2] https://www.graphviz.org/ acedido no dia 4 de Maio pelas 14h00m