Processamento de linguagens (3º ano de MIEI)

Trabalho Prático 1

Relatório de Desenvolvimento

Célia Figueiredo (a67637)

Diogo Tavares (a61044)

Gil Gonçalves (a67738)

5 de Abril de 2016

Resumo

Neste relatório será apresentado o desenvolvimento de um filtro de texto com a posterior aplicação sob um ficheiro BibTex, pretende-se que seja produzido um normalizador de ficheiros BibTex.

Implementou-se um filtro que permite fazer a contagem das categorias das referências bibliográficas, também foi implementado um filtro que permite a troca para chavetas do campo que está entre aspas. E ainda foi implementado um filtro que coloca os nomes dos autores escritos no formato "N. Apelido". Por fim, de modo a tornar a leitura mais fácil implementou-se um ferramenta de *pretty-printing*.

Será também mostrado um grafo que ilustra para um dado autor (escolhido pelo utilizador) todos os autores que publicam normalmente com o autor em causa. Utilizou-se a linguagem *Dot* do *GraphViz* esta, que gerou um ficheiro com um grafo de modo a que posteriormente fosse usada uma das ferramentas que processam *Dot* para desenhar o dito grafo de associações de autores.

Conteúdo

Capítulo 1

Introdução

Este trabalho envolverá o desenvolvimento de um normalizador de ficheiros *BibTex*, este é o tema do problema 2.2 do enunciado fornecido.

Enquadramento Utilização de expressões regulares e filtros de texto com o objetivo de produzir novos documentos a partir de padrões existentes no ficheiro de input.

Conteúdo do documento O presente documento contém a explicação do problema, assim como a apresentação das soluções produzidas.

Resultados Os resultados deste desafio serão as alineas pedidas, sendo que serão apresentados em ficheitos *.html* e grafos.

Estrutura do Relatório

Este documentos está dividido em seis partes No capítulo 2 faz-se uma análise detalhada do problema proposto de modo a poder-se especificar as entradas, resultados e formas de transformação do ficheiro *.bib*.

No capítulo 3 serão descritas as estruturas de dados implementadas para a realização do problema descrito. No capítulo 4 serão mostradas as expressões regulares desenvolvidas para a implementação do caso de estudo, assim como os testes realizados e os resultados. No capítulo 5 termina-se o relatório com uma síntese do que foi dito, as conclusões e o trabalho futuro. No fim do documento estará incluido ficheiros anexos com o código desenvolvido.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

O problema escolhido consiste na análise da ferramenta de formatação de citações e referências bibliográficas em documentos LaTeX.
O *BibTex* é uma ferramenta que foi criada em 1985 e utiliza um ficheiro *(.bib)* este que é uma base de dados que contém os dados bibliográficos (autor, título, ano de publicação, etc.) das fontes citadas no documento LaTeX.
Deixámos um exemplo de um excerto de um ficheiro com a extensão *.bib*:

```
@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"
}
```

O ficheiro BibTex contém várias categorias de referência estas inicializadas sempre pelo caracter @, deixamos aqui alguns exemplos de entradas que podem ser encontradas no ficheiro BibTex:

```
@inbook
@misc
@incollection
@inproceedings
@techreport
@unpublished
```

2.2 Especificação do Requisitos

2.2.1 Dados

Os dados fornecidos são um ficheiro *.bib*, este que é um ficheiro com as caracteristicas de um ficheiro *BibTex*.
Cada tipo de categoria tem os seus campos obrigatórios, neste acaso o objetivo de uma das tarefas será pesquisar através do campo **author =** o nome do autor e transformá-lo no formato "N. Apelido".

O nome da categoria é seguida por uma chaveta, e o primeiro campo será o nome para a referência a ser introduzida, os campos de cada categoria são separados por vírgula e a seguir ao campo aparecerá o simbolo igual (=), deixámos um exemplo da sintaxe da categoria @**phdthesis** e os respetivos campos presentes:

```
@phdthesis{Mos75a,
author = "P. D. Mosses",
title = "Mathematical Semantics and Compiler Generation",
year = 1975,
school = "Oxford University",
annote = "compilacao incremental, atributos, ambientes prog"
}
```

É também fornecido o nome de ferramentas de apoio à resolução do problema, neste caso o *Graph Viz*, que permitirá colocar gráficamente a informação dos grafos criados, sendo que tornará as iterações entre os autores mais percetiveis.

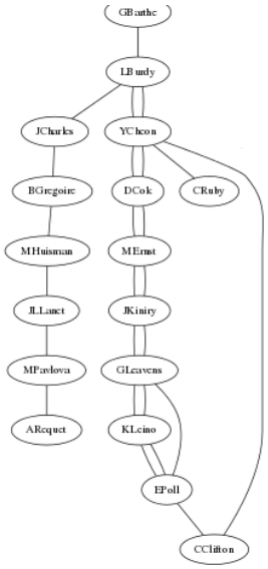

Figura 2.1: Exemplo de geração de um grafo com os nomes dos autores com recurso à ferramenta *Graph Viz*

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

3.1.1 Algoritmo alinea a)

Nesta alinea optou-se por utilizar listas ligadas porque vai guardando as categorias e o respetivo contador à medida que se encontra um padrão no ficheiro *.bib*.
Decidimos que as categorias que se diferenciavam apenas nas letras maiúsculas ou minúsculas pertenceriam à mesma categoria.

```
void* procura(char* c){
int flag = 1;

Est* aux;
aux = est;
while(aux!=NULL && flag && aux->cat != NULL){
if(!strcmp(minusculas(aux->cat), minusculas(c+1))){
aux->i++;
flag=0;
}
aux=aux->next;
}

if(flag){
Est* novo = (Est*) malloc(sizeof(Est));
novo->cat = strdup(c+1);
novo->i = 1;
novo->next = est;
est = novo;
}
}
```

3.1.2 Algoritmo alinea b1)

Utilizámos dois arrays, um que guarda a informação que é lida, e outro que guarda a informação tratada.

```
void trata()
```

3.1.3 Algoritmo alínea b2)

Utilizamos arrays strtoken para apanhar o que está entre chavetas ou aspas strstr para cortar a string por autores

3.1.4 Algoritmo alinea c)

Nesta alínea utilizámos arrays

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

4.1.1 Expressões Regulares

Para garantir que uma secção do código do ficheiro input ficassse protegida de todas as acções accionadas por outras expressões regulares declaramos o estado %x.

Alinea a)

Nesta alinea é pedido que se faça a contagem das categorias existentes, e no final se produza um ficheiro em formato HTML como o nome e as respetivas contagens.
Para a realização deste desafio é necesário procurar a categoria inicializada pelo caractere @ até encontrar a chaveta de abertura ({). A categoria encontrada será inserida na estrutura através da função *procura*, se a categoria já existir adiciona mais uma entrada ao contador, senão cria uma nova e insere-a na estrutura. Todas as categorias que obdecerem à expressão regular são capturadas com o *yytext*.

```
@[^{]* {procura(yytext);}
```

O restante texto é ignorado com a seguinte expressão:

```
.|\n {;}
```

Alínea b1)

Esta alínea está dividida em duas partes. A primeira pede que sempre que um campo está entre aspas estas sejam trocadas por chavetas e o nome dos autores deverá apresentar o formato "N. Apelido".
Para apanhar o conteúdo do campo **author =** , este que pode iniciar-se com chaveta de abertura ou aspas, sugerimos os seguintes filtros:

```
"author ="[ ]+"\""  {BEGIN (AUT);}
"author ="[ ]+"\{"  {BEGIN (AUT);}
```

Quando num autor se encontrar caracteres especiais no meio do nome tais como chavetas ou aspas, insere esses caracteres no array.

```
<AUT>"\}"  {inser(yytext); }
<AUT>"\""  {inser(yytext); }
```

Implementámos os filtros que nos permitem separar os nomes dos autores, estes que podem estra separados por vírgulas ou **and**.

```
<AUT>"\,"[ ]+  {trata();funcao();}
<AUT>[ ]+"and"[ ]+  {trata();funcao();}
<AUT>[ ]+"and"(\n)  {trata();funcao();}
```

Estes filtros permitem-nos determinar quando um autor termina, tanto com aspas seguida de vírgula ou chaveta seguida de vírgula.

```
<AUT>"\"""\,"     {trata();funcao();print();tmp=NULL;BEGIN  INITIAL;}


<AUT>"\}""\,"     {trata();funcao();print();tmp=NULL;BEGIN INITIAL;}
```

Este filtro lê tudo o que está no à frente do campo autor:

```
<AUT>[ '-}] {inser(yytext); }
\"[a-zA-Z]  {printf("{%c", yytext[1]);}
[a-zA-Z]\"  {printf("%c}", yytext[0]);}
```

Alínea c)

Nesta alínea é pedido que seja construido um grafo que mostre para dado autor todos os autores que publicam com ele.
Para tal é necessário que seja feita a recolha de todos os nomes dos autores, depois para associá-los utiliza-se a expressão *and* que nos indica quais os autores que teem publicações conjuntas.
Este filtro tal como o anterior serve para apanhar o conteúdo do campo **author =** para os dois casos possiveis no ficheiro *.bf*.

```
"author ="[ ]*"\""  {BEGIN (AUT);}
"author ="[ ]*"\{"  {BEGIN (AUT);}

<AUT>"\}"             {;}


<AUT>"\,"[ ]+         {trata();}
<AUT>[ ]+"and"[ ]+    {trata();}
<AUT>[ ]+"and"(\n)    {trata();}


<AUT>"\"""\,"         {trata();print();tmp=NULL;BEGIN INITIAL;}
<AUT>"\}""\,"         {trata();print();tmp=NULL;BEGIN INITIAL;}
<AUT>[A-Za-z]           {inser(yytext);}
```

4.1.2 Makefile

O principal objetivo da Makefile é facilitar a compilação e execução do programa. Para isso criamos o seguinte ficheiro:

4.2 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos e respectivos resultados obtidos:

4.2.1 Testes e Resultados alinea a)

Utilizámos este ficheiro de teste para testar as diferentes opções que apareciam no ficheiro

```
@book{
@book{
@cenas{
@book{
@coisas{
@Book{
@book{
@BOOK{
      @BOOK{
       @Book{
      @author{

@AuThor{
@AUTHOR{
```

Mostrámos de seguida o ficheiro *.html* produzido:

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

4.2.2 Testes e Resultados alinea b)

4.2.3 Testes e Resultados alinea c)

Capítulo 5

Conclusão

Na alínea a pressupusemos que as categorias com o mesmo nome e que se diferenciavam apenas em maiúsculas ou minusculas pertenciam à mesma categoria. Pois assim não estariamos a repetir informação.

A alinea c foi implementada pensando que o grafo seria criado com todos os autores. Porém após uma leitura mais atenta verificou-se que seria pedido que o utilizador escolhesse o autor e assim gerar o grafo com os autores que publicam diretamente com ele.

Apêndice A

Código do Programa

Lista-se a seguir o código que foi desenvolvido para a alínea a:

```
%{

typedef struct Est{
        char* cat;
        int i;
        struct est *next;
}Est;

Est* est;

void* novo(){
        est =(Est*) malloc(sizeof(Est));
        est->next = NULL;
        est->cat = NULL;
        est->i = 0;
}

char* minusculas(char* d){
        char* novo = strdup(d);
        int j=0;
        while(novo[j] != '\0'){
                novo[j]= tolower(novo[j]);
                j++;
        }
        return novo;
}

void* procura(char* c){
        int flag = 1;

                Est* aux;
                aux = est;
                while(aux!=NULL && flag && aux->cat != NULL){
                        if(!strcmp(minusculas(aux->cat), minusculas(c+1))){
                                aux->i++;
                                flag=0;
                        }
                        aux=aux->next;
                }

        if(flag){
```

```
                Est* novo = (Est*) malloc(sizeof(Est));
                novo->cat = strdup(c+1);
                novo->i = 1;
                novo->next = est;
                est = novo;
        }
}

%}

%%
@[^{]*  {procura(yytext);}
.|\n    {;}

%%

void print(){
        FILE* fp;
        fp = fopen("ficheiro.html", "w");
        fprintf(fp, "<html>");
        while(est->next!=NULL){
                fprintf(fp, "%s -> %d<br>", est->cat, est->i);
                est = est->next;
        }
        fprintf(fp, "<\html>");
        fclose(fp);
}

int yywrap() {return 1;}
int main(){
        est = novo();
        yylex();
        print();
        free(est);
        return 0;

}
```

Lista-se a seguir o código que foi desenvolvido para a alínea b:

```
%{
#include<string.h>
char  nomes[1000];
int i=0;
char *tmp;


void inser(char *texto) {
nomes[i]=*texto;
i++;

}

void trata() {
      nomes[i]='\0';

   int variavel =1;

  if(nomes[variavel]==' ') {
   char kill[1000]; int o=0;
```

```
21      while(nomes[variavel]==' ') {variavel++;}
22      while(nomes[variavel]!='\0') {
23           kill[o]=nomes[variavel];
24           o++;
25          variavel++;
26  }       kill[o]='\0';
27      o=0;
28      while(kill[o]!='\0') {
29           nomes[o]=kill[o];
30        o++;
31     }
32
33   nomes[o]='\0';
34
35  }
36
37     int n= strlen(nomes);
38  int j=n-1;
39
40
41  while(nomes[j]==' '){j--;}
42
43  while(nomes[j]!=' ' && j!=0){
44  j--;
45  }
46  int k=1;
47  nomes[k]='.';
48  k++;
49
50  if(j!=0) {
51  while(nomes[j]!='\0') {
52
53          nomes[k]=nomes[j];
54         j++;
55         k++;
56  }
57
58  nomes[k]='\0';
59  i=0;
60
61  }
62  else {
63  nomes[k]='\0';i=0;
64  }
65
66
67  }
68
69  void funcao() {
70
71  if (tmp) {
72
73  i=0;
74     strcat(tmp," and ");
75  int n=strlen(tmp);
76  int j=0;
77  tmp[n]=' ';
78  n++;
79
```

```
80 for (j=0;nomes[j]!='\0';j++) {
81
82     tmp[n]=nomes[j];
83     n++;
84
85 }
86 tmp[n]='\0';
87
88 }
89
90 else {
91 i=0;
92 int j=0;
93 tmp=(char *)malloc(1000);
94
95     for(j=0;nomes[j]!='\0';j++) {
96                 tmp[j]=nomes[j];
97
98   }
99           tmp[j]='\0';
100
101 }
102
103
104 }
105
106
107 void print() {
108
109 char h[1000]="author = {";
110
111 strcat(h,tmp);
112 int n=strlen(h);
113 h[n]='}';
114 n++;
115 h[n++]=',';
116
117 h[n]='\0';
118 printf("%s",h);
119
120 }
121
122
123
124
125
126
127 %}
128
129
130 %x AUT
131
132 %%
133
134
135 "author ="[ ]*"\""   {BEGIN (AUT);}
136 "author ="[ ]*"\{"   {BEGIN (AUT);}
137
138
```

```
<AUT>"\}"   {inser(yytext); }
<AUT>"\""   {inser(yytext); }
<AUT>"\,"[ ]+   {trata();funcao();}
<AUT>[ ]+"and"[ ]+   {trata();funcao();}
<AUT>[ ]+"and"(\n)   {trata();funcao();}
<AUT>"\"""\,"       {trata();funcao();print();tmp=NULL;BEGIN INITIAL;}
<AUT>"\}""\,"       {trata();funcao();print();tmp=NULL;BEGIN INITIAL;}
<AUT>[ '-}] {inser(yytext); }

\"[a-zA-Z]        {printf("{%c", yytext[1]);}
[a-zA-Z]\"        {printf("%c}", yytext[0]);}


%%

int yywrap() {return 1;}
int main(){
        yylex();

  free(tmp);
        return 0;
}
```

Lista-se a seguir o código que foi desenvolvido para a alínea c:

```
%{
#include<string.h>
char  nomes[1000];
int i=0;
char *tmp;
FILE *fp;


void inser(char *texto) {
   nomes[i]=*texto;
    i++;
}

void trata() {
   nomes[i]='\0';
  if(tmp) {
   i=0;

    int n =strlen(tmp);
    tmp[n++]='-';
    tmp[n++]='-';
    tmp[n]='\0';
    strcat(tmp,nomes);
   }

 else {
    tmp=(char *) malloc(1000);
    strcat(tmp,nomes);
    i=0;

   }

}

void print() {
```

```
 i=0;
    fprintf(fp,"%s",tmp);
    fprintf(fp,"\n");

}



void abrir(){

        fp = fopen("ficheiro.dot", "w");
        fprintf(fp, "graph {");
}

void fechar() {


 fprintf(fp, "}");
        fclose(fp);

}




%}

%x AUT



%%

"author ="[ ]*"\""   {BEGIN (AUT);}
"author ="[ ]*"\{"   {BEGIN (AUT);}
<AUT>"\}"              {;}
<AUT>"\,"[ ]+          {trata();}
<AUT>[ ]+"and"[ ]+     {trata();}
<AUT>[ ]+"and"(\n)     {trata();}
<AUT>"\"""\,"          {trata();print();tmp=NULL;BEGIN INITIAL;}
<AUT>"\}""\,"          {trata();print();tmp=NULL;BEGIN INITIAL;}
<AUT>[A-Za-z]            {inser(yytext);}



%%


int yywrap() {return 1;}
int main(){
       abrir();
     yylex();
        fechar();
   free(tmp);
        return 0;
}
```