Processamento de Linguagens (3º ano de MiEI)

Trabalho Prático 1

Relatório de Desenvolvimento de um Normalizador de ficheiros *BibTex*

Gustavo da Costa Gomes-Aluno
(72223)

José Carlos da Silva Brandão Gonçalves-Aluno
(71223)

Tiago João Lopes Carvalhais-Aluno
(70443)

20 de Março de 2016

Resumo

Isto é um resumo do relatório da unidade curricular Processamento de Linguagens relativamente ao Trabalho Prático 1. Este visa a produção de um Normalizador de ficheiros *BibTex* permitindo a exploração da ferramenta *Flex* acompanhada de uma pequena demonstração de quão poderosa realmente é.

Conteúdo

Capítulo 1

Introdução

Este trabalho envolveu o desenvolvimento de um Normalizador de ficheiros *BibTex*, um dos enunciados disponibilizados, no qual se procede á sua análise tendo em conta os seguintes pontos,

Enquadramento Utilização de Expressões Regulares e Filtros de Texto com o objetivo de produzir novos documentos a partir de padrões existentes num outro ficheiro.

Estrutura do documento Este documento possui um anexo com todo o código produzido em cada uma das alíneas, uma conclusão final que une o exercício e as respectivas soluções elaboradas e ainda capítulos elucidativos de cada tarefa a desempenhar em cada alínea.

Resultados Os resultados serão apreciados nos respectivos capítulos correspondentes a cada uma das alíneas desenvolvidas e cujo ficheiro(ou partes dele) estará(ão) no Anexo correspondente.

Conteúdo do documento Contém a explicação do problema em si, bem como a apresentação das soluções produzidas para colmatar essa situação, auxiliada com documentação e código produzido, presente nos Anexos.

Estrutura do Relatório

Este relatório possui quatro capítulos, uma conclusão,um capítulo extra dedicado a Anexos e a respectiva Bibliografia utilizada durante a realização deste projecto. Os capítulos são Análise e Especificação do problema, Especificação dos Requisitos, Arquitectura da solução para cada um dos sub-problemas indicados no enunciado global que envolverá a explicação das estruturas de dados utilizadas e por fim o capítulo designado Codificação, que incluirá alguns aspectos relevantes de todos os testes realizados para a verificação do correcto funcionamento.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

BibTex é uma ferramenta de formatação de citações bibliográficas em documentos *Latex*, que separa a bibliografia consultada do restante conteúdo. Um exemplo desse ficheiro, com a extensão .bib ilustra-se de seguida,

```
@InProceedings{CPBFH07e,
  author =     {Daniela da Cruz and Maria Joao Varanda Pereira
                and Mario Beron and Ruben Fonseca and
                Pedro Rangel Henriques},
  title =      {Comparing Generators for Language-based Tools},
  booktitle = {Proceedings of the 1.st Conference on Compiler
}
year =
editor =
month =
 Related Technologies and Applications, CoRTAâ07
 --- Universidade da Beira Interior, Portugal},
{2007},
{},
{Jul},
```

Este enunciado consiste em três alíneas, nas quais são requeridas diferentes tarefas a implementar. Na alínea a é pedido que se elabore um documento *HTML* que contenha a contagem de todas as diferentes categorias presentes no documento lpbib.txt, explicado num capítulo vindouro, *Código do Programa*.

Na alínea b é pedido que se desenvolva uma ferramenta de normalização que faça *pretty-printing*, fazendo a indentação correta em cada campo, escrevendo cada autor numa linha e que coloque no início da mesma os campos autor e título, e quando um campo estiver entre aspas modifique para chavetas e se escreva o nome dos autores com o seguinte formato, *N.Apelido*.

Por fim, na alínea c é pedido a construção de um grafo que para um determinado autor, á escolha do utilizador, mostre todos os autores que publicaram com esse autor. Para tal, recorrer-se-á á linguagem Dot do *GraphViz2*, gerando um ficheiro com esse grafo de modo a que possa, posteriormente, desenhar o mesmo através de uma outra ferramenta que faça a leitura desses ficheiros.

2.2 Especificação do Requisitos

Neste trabalho, o objectivo é estimular a utilização de um ambiente *Linux*, da linguagem imperativa *C* e de outras ferramentas de apoio para a resolução de problemas de um modo diferente do habitual, que seria tentar resolver tudo apenas utilizando uma linguagem de programação e, para tal, visam o estudo e o desenvolvimento de Expressões Regulares, bem como, a sua manipulação por forma a atingir o resultado pretendido. Essas expressões são fundamentais para encontrar os padrões para os quais se irá tomar uma ação, que será a transformação do texto, filtrando ou removendo esses. Como auxiliar na realização de filtros de texto recorrer-se-á á utilização de um gerador designado, *Flex*.

Na realização deste problema é necessário concluir uma lista de tarefas que são, especificar os padrões de frases que se quer encontrar no texto fonte, através de Expressões Regulares, identificar as acções semânticas a realizar como reacção ao reconhecimento de cada um desses padrões, identificar as estruturas de dados globais que possa eventualmente precisar para armazenar temporariamente a informação que se vai extraindo do texto fonte ou que se vai construindo á medida que o processamento avança e por fim desenvolver um filtro de texto para fazer o reconhecimento dos padrões identificados e proceder à transformação pretendida, com recurso ao gerador *Flex*.

2.2.1 Dados

Os dados fornecidos são o ficheiro lpbib.txt, a ser abordado num capítulo posterior, a definição e utilização da ferramenta *Flex* e a definição de um ficheiro *BibTex*, isto é, as suas características.

É também fornecido o nome de ferramentas de apoio á resolução do problema, sendo neste problema, a ferramenta *GraphViz2*, que permitirá colocar graficamente a informação dos grafos criados, sendo que as interações entre os autores se tornam mais perceptíveis.

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

3.1.1 Alínea a

Nesta alínea optou-se por utilizar uma *hashtable* como estrutura de dados auxiliar que vai guardando as categorias e o respectivo contador á medida que se vai encontrando um padrão no ficheiro lpbib.txt, ou seja, a acção ao padrão, que permite encontrar as categorias todas ao longo de todo o contéudo.

Esta estrutura segue a lógica de *Open Adressing*, isto é, através de uma função de *hash* é encontrada a posição onde se irá inserir a categoria capturada pela expressão regular e no caso de essa estar já ocupada vai tentar inserir na posição seguinte, e se chegar á última reinicia, visto que é *circular*. Apenas se implementou funções essenciais, *inserir*, *remover*, *procurar* e *imprimir* o conteúdo desta estrutura e gerar o conteúdo do ficheiro *HTML* pedido. Para verificar se uma posição já possui ou não conteúdo basta verificar a *etiqueta* associada e designada por *state*, no Anexo encontra-se o conteúdo integral da implementação desta estrutura de dados.

3.1.2 Alínea b

3.1.3 Alínea c

Capítulo 4

Codificação

4.1 Alternativas, Decisões e Problemas de Implementação

4.1.1 Alínea a

Um dos problemas de implementação passou por conseguir contabilizar as categorias de forma independente, isto é, após uma análise do documento fonte lpbib.txt verificou-se a existência de categorias que contem exatamente os mesmos caracteres mas escritos de diferentes formas. Um exemplo disso é a categoria *inproceedings* que pode também se encontrar como *InProceedings* e como *INPROCEEDINGS*. Então o grande desafio foi separar esta categoria em três, porque apesar de terem os mesmos caracteres, elas representam a mesma categoria mas de forma independente visto que na contagem destas se pretende ter um resultado que mostre o que realmente está no ficheiro fonte.

Na fase de implementação surgiram pequenos problemas relacionados com a expressão regular que foi definida por forma a filtrar apenas a categoria. Visto que essa estava delimitada por dois caracteres, o '@' e o ''. Por vezes *yytext* não continha o conteúdo correto, o que revelava que a expressão regular ainda não estava a funcionar corretamente.

Após esse problemas estarem resolvidos conseguiu-se produzir o ficheiro *HTML* sem nenhuma dificuldade, apenas se efetuou a impressão do conteúdo da estrutura de dados que foi armazenando as categorias e atualizando os seus contadores com a indentação e os *headers* que permitem visualizar o ficheiro obtido num *browser*. Esse ficheiro *HTML* produzido chama-se indexA.html e pode ser visto no Anexo A.1, bem como o filtro de texto produzido, recorrendo á ferramenta *Flex*, neste documento.

Para terminar falta proceder á análise das expressões regulares utilizadas e a respectiva acção a tomar quando estas forem encontradas.

```
i. @[a?zA?Z]+\{
ii. .|\n
```

A expressão regular ii. é para filtrar todo o texto, mas é absorvente e por isso é preciso muito cuidado com a sua utilização e ,associado a esta, a acção de fazer *print* que é a acção por defeito, quando se fornece

```
{}
```

A expressão regular i. é a expressão que foi desenvolvida para a resolução do problema pedido, contagem das categorias, que exige inicialmente a captação das categorias e apenas das categorias presentes no ficheiro lpbib.txt, que contem muita mais informação. O objectivo desta é encontrar todos os padrões que estejam contidos entre os caracteres '@' e '', visto que essa é a definição de categoria num ficheiro *BibTex*. E como as categorias apenas podem conter letras temos de restringir os padrões encontrados entre esses dois caracteres ao facto de que só podem ter letras, quer minúsculas quer maiúsculas, daí '[a?zA?Z]+'. O símbolo '+' refere-se á possibilidade de encontrar uma

ou mais ocorrências, isto é, entre esses dois caracteres encontra-se apenas uma ou mais letras. Visto que não existe nenhuma restrição quanto á forma da palavra categoria, isto é, por exemplo a exigência de começar por letra maiúscula e seguida apenas de letras minúsculas não é necessário efetuar mais nenhuma restrição ao padrão que permitirá filtrar as categorias.

Associado a esta, última expressão regular, está a acção de copiar o conteúdo de *yytext+1*, que apenas considera o texto após o '@' até *yyleng-2*, onde está o '' para um *char** local que será então inserido na estrutura de dados através da instrução *insertTable(ht, str, (int ?) count) ;* sendo que *count* é um contador inicial que apenas serve para iniciar o contador na estrutura de dados. Esta instrução está codificada por forma a verficar logo se a categoria já existe ou não, e caso exista apenas incrementa a ocorrência dessa categoria e ignora o parâmetro *count* recebido. E caso não exista procede então ao inicio do contador com valor de *count* recebido e insere na estrutura de dados.

Por fim é necessário criar o ficheiro *HTML* pretendido com o conteúdo da estrutura de dados que foi sendo atualizada até se chegar ao fim do ficheiro lpbib.txt e para tal na função *main* do ficheiro tp1A.l recorreu-se á chamada da função *printHashTable (ht) ;* que foi codificada no ficheiro *hashtable.c* por forma a criar o ficheiro *HTML* com a formatação necessária, produzindo, desse modo, o resultado final desta alínea.

4.1.2 Alínea b

4.1.3 Alínea c

4.2 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (valores introduzidos) e os respectivos resultados obtidos: lpbib.txt nas diferentes alíneas deste problema.

4.2.1 Alínea a

Para este problema fizeram-se testes sucessivos até encontrar a expressão regular que permitisse armazenar apenas a categoria na estrutura de dados evitando desse modo que a manipulação de *strings* fosse feita através de funções definidas na linguagem *C*, tal como, *strtok* ou outra semelhante, uma vez que o objectivo é a utilização de expressões regulares que façam todo o trabalho de manipulação de texto.
Os testes realizados passaram todos por correr a seguinte *makefile*

```
1 all: 1 2
2
3 1:
4                 flex tp1.l
5
6 2:
7                 gcc lex.yy.c hashtable.c -ll -o tp1
```

E de seguida efetuar

```
./tp1 < lpbib.txt
```

E verificando se o resultado obtido era realmente o resultado pretendido. Antes de se chegar ao resultado correcto teve-se de corrigir a situação de contagem de categorias com os mesmos caracteres mas que representavam categorias distintas porque o que acontecia era, para cada uma dessas categorias o contador estava a incluir as outras, mas só mostrava as ocorrências de uma dessas categorias. Por exemplo, para a categoria *INPROCEEDINGS* verifica-se que no ficheiro fonte lpbib.txt apenas ocorre cinco vezes, mas o que acontecia era que se se imprimisse o conteúdo da estrutura de dados ela apresentava apenas essas cinco ocorrências mas o valor do contador não correspondia á contagem real porque estava a incluir os contadores das categorias *InProceedings* e *inproceedings*.

Como resultado final obteve-se o ficheiro *HTML* com o formato pretendido, que é categoria x e o valor do contador para essa categoria x. Pode ser verificada abrindo o ficheiro lpbib.txt e procurando uma categoria qualquer e verificar que o número de ocorrências coincidem e para visualizar o aspecto da solução apenas é necessário abrir o indexA.html com um *browser* qualquer.

4.2.2 Alínea b

4.2.3 Alínea c

Capítulo 5

Conclusão

Síntese do Documento.
Estado final do projecto; Análise crítica dos resultados.
Trabalho futuro.

Apêndice A

Código do Programa

Lista-se a seguir um excerto do ficheiro ***BibTex*** que foi utilizado para demonstrar o funcionamento, correto, do código desenvolvido para a resolução do problema. Ficheiro esse que foi disponibilizado em `http://di.uminho.pt/~prh/lp.bib`

```
@string{ eth = "Institut fur Informatik, ETH Zurich" }

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

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

A.1 Alínea a

O código do programa desenvolvido em *Flex*, tal como está no ficheiro fonte *tp1A.l* encontra-se de seguida.

```
%{
        #include <stdlib.h>
        #include <stdio.h>
        #include <string.h>
  #include "hashtable.h"
        HashTable ht;
        int count=1;
        int position=0;
%}



%%

@[a-zA-Z]+\{                    {
                        char* str = (char *) malloc(sizeof(char)*1000);
                                                    strcpy(str,yytext+1);
                                                    str[yyleng-2] ='\0';
                                                                        insertTable(
                                                                            ht, str,
                                                                             (int *)
                                                                            count);
                                                }

.|\n                                            {}



%%
int main(){
        initializeTable(ht);
        int s;
   s=yylex();
        while(s){printf("%d",s);s=yylex();}
        printHashTable(ht);
        printf("\n");
        return 0;
}
```

Apesar de existirem librarias disponíveis com estruturas de dados já implementadas, tomou-se a liberdade de reutilizar uma libraria já produzida numa unidade curricular anterior em vez de utilizar *hsearch.h* disponível em *glib2.h*.

O código da estrutura de dados utilizada na resolução da alínea *a* deste problema foi desenvolvido na linguagem *C* e apresenta-se de seguida o código na integra, que pode ser encontrado no ficheiro fonte *hashtable.h*. Relembra-se que as operações de inserir, remover e procura nesta estrutura de dados encontra-se em *hashtable.c*.

Ficheiro *.h*,

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define HASHSIZE 31
6 #define KEYTYPE_SIZE 30
7 #define EMPTY 0
8 #define DELETED -1
9 #define FULL 1
10
11 //Open  Addressing - Linear Probing (insertTable) and Quadratic Probing (insertTable2)
12 //After hash_function calculated the position if it is not available it try insert in the
       next position with
13 //label "empty" or "deleted". "deleted" is required when retrieving data only stops when
       founds "empty"
14
15 typedef char Keytype[KEYTYPE_SIZE];
16 typedef void *Info;
17 typedef struct entry{
18   int state;
19   Keytype key;
20   Info info;
21 }Entry;
22
23 typedef Entry* HashTable[HASHSIZE];
24
25 int hash_function(Keytype key);
26 void initializeTable(HashTable ht);
27 void clearTable(HashTable ht);
28 void insertTable(HashTable ht, Keytype k, Info i);
29 int retrieveTable(HashTable ht,Keytype k);
30 void deleteTable(HashTable ht, Keytype k);
31 void printHashTable(HashTable ht);
```

Ficheiro *.c*,

```
#include "hashtable.h"


int hash_function(Keytype key){
  int i=0,sum=0;
  for(;i<KEYTYPE_SIZE-1;i++){ //ends with '\0' so we must subtract one iteration
      corresponding to that char
    sum+=key[i];
  }
  return (sum%HASHSIZE);
}

void initializeTable(HashTable ht){
  int i=0;
  for(;i<HASHSIZE;i++){
    Entry* new = (Entry*) malloc(sizeof(struct entry));
    new->state=EMPTY;
    strncpy(new->key,"Empty",KEYTYPE_SIZE);
    new->info=NULL;
    ht[i]=new;
  }
}

void clearTable(HashTable ht){
  int i=0;
  for(;i<HASHSIZE;i++) free(ht[i]);
}

void insertTable(HashTable ht, Keytype k, Info i){
  int position=hash_function(k);
  int found=retrieveTable(ht,k);
  if(found!=0){ // k jÃ¡ existe
    ht[found]->info=ht[found]->info + 1;
  }
  else{ //k nÃ£o existe
  if(ht[position]->state!=FULL){
    strncpy(ht[position]->key,k,KEYTYPE_SIZE);
    ht[position]->info=i;
    ht[position]->state=FULL;
  }
  else{
    //try to insert in the next and so on --- Linear Probing
    while(ht[position]->state==FULL) position=(position+1)%HASHSIZE;
    //it founded an EMPTY or DELETED
    strncpy(ht[position]->key,k,KEYTYPE_SIZE);
    ht[position]->info=i;
    ht[position]->state=FULL;
  }
  }
}

//retrieving with linear probing from the initial position
int retrieveTable(HashTable ht,Keytype k){
  int position=hash_function(k);
```

```
54    Entry* aux;
55    int res=0;
56    for(;ht[position]->state!=EMPTY && position<HASHSIZE;position=(position+1)%HASHSIZE){
57      if(strncmp(ht[position]->key,k,KEYTYPE_SIZE)==0){
58        aux=ht[position];res=position;
59      }
60    }
61    return res;
62  }
63
64  void deleteTable(HashTable ht, Keytype k){
65    int position=hash_function(k),i=0;
66    for(;ht[position]->state!=EMPTY && i<HASHSIZE;i++){
67      if(strncmp(ht[position]->key,k,KEYTYPE_SIZE)==0){
68        ht[position]->state=DELETED;
69        strncpy(ht[position]->key,"Deleted",KEYTYPE_SIZE);
70        ht[position]->info=NULL;
71      }
72    }
73  }
74
75  void printHashTable(HashTable ht){
76    int i=0;
77    Entry* aux;
78    FILE * fp = fopen("index.html","wr");
79    fprintf(fp,"<HMTL>\n\t<BODY>\n");
80    fprintf(fp,"<h1> BibTex File </h1>\n");
81    for(;i<HASHSIZE;i++){
82      aux=ht[i];
83      if(aux->state!=EMPTY && aux->state!=DELETED){
84        fprintf(fp,"\t\t<li>Categoria: %s, Contagem: %d\n",aux->key,(int) aux->info);
85      }
86    }
87    fprintf(fp,"\t<BODY/>\n<HTML/>");
88  }
```

O ficheiro com a resolução da alínea *a* é apresentado em baixo, e o seu conteúdo está no ficheiro *indexA.html*.

```
<HMTL>
        <BODY>
                <li>Categoria: MISC, Contagem: 1
                <li>Categoria: TechReport, Contagem: 2
                <li>Categoria: mastersthesis, Contagem: 2
                <li>Categoria: techreport, Contagem: 138
                <li>Categoria: proceeding, Contagem: 1
                <li>Categoria: ARTICLE, Contagem: 1
                <li>Categoria: MISC, Contagem: 1
                <li>Categoria: unpublished, Contagem: 15
                <li>Categoria: INPROCEEDINGS, Contagem: 5
                <li>Categoria: phdthesis, Contagem: 21
                <li>Categoria: string, Contagem: 31
                <li>Categoria: incollection, Contagem: 6
                <li>Categoria: manual, Contagem: 13
                <li>Categoria: BOOK, Contagem: 2
                <li>Categoria: InProceedings, Contagem: 20
                <li>Categoria: inbook, Contagem: 3
                <li>Categoria: inproceedings, Contagem: 184
                <li>Categoria: book, Contagem: 44
                <li>Categoria: misc, Contagem: 39
                <li>Categoria: proceedings, Contagem: 4
                <li>Categoria: article, Contagem: 131
                <li>Categoria: Article, Contagem: 10
                <li>Categoria: Misc, Contagem: 20
                <li>Categoria: Book, Contagem: 1
        <BODY/>
<HTML/>
```

A.2 Alínea b

A.3 Alínea c

Bibliografia

[1] *V. Aho, Alfred* , *S. Lam,Monica* , *Sethi,Ravi* and *D. Ullman,Jeffrey* Second Edition, *Compilers Principles Techniques and Tools*.

[2] ShareLatex examples and tutorials, `https://www.sharelatex.com`