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)

April 3, 2016

Abstract

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. Também é possível encontrar numa seção, no capítulo "Anexo", que possui a resolução de um segundo enunciado, que é a conversão de ficheiros *Zim-Wiki* para slides *Beamer*.

Contents

Chapter 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.

Chapter 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,

```
1  @InProceedings{CPBFH07e,
2    author =     {Daniela da Cruz and Maria Joao Varanda Pereira
3                  and Mario Beron and Ruben Fonseca and
4                  Pedro Rangel Henriques},
5    title =      {Comparing Generators for Language-based Tools},
6    booktitle = {Proceedings of the 1.st Conference on Compiler
7  }
8  year =
9  editor =
10 month =
11  Related Technologies and Applications, CoRTAâ07
12  --- Universidade da Beira Interior, Portugal},
13 {2007},
14 {},
15 {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. Para além destes, na alínea c) o ficheiro de *input* utilizado é referente ao resultado da alínea b) e é tem o nome de *graphTeste.txt*.

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.

Chapter 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

Nesta alínea, optou-se pela utilização de duas listas ligadas, uma para guardar os autores que publicaram com o autor fornecido e outra para guardar os autores de uma publicação para, posteriormente, verificar se o autor principal lá estava contido e, em caso afirmativo, guardar-se na primeira estrutura os outros autores, sendo que, caso um autor já exista na estrutura principal, então incrementava um contador que indica quantas vezes esse autor fez publicações em conjunto com o principal.

O grupo optou por escolher esta estrutura de dados, uma vez que é bastante simples de implentar, tendo como uma das desvantagens a pouca eficiência na procura e inserção de informação. Contudo, como trabalhamos com pouca quantidade de dados (em termos computacionais), chegámos à conclusão que as desvantagens da lista ligada não se iriam fazer sentir na execução do programa.

Chapter 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

Inicialmente, o problema incidiu no pouco conhecimento do grupo acerca da linguagem *Dot*, pelo que foi necessário efetuar um estudo aprofundado sobre a linguagem em questão. Posteriormente, houve uma grande dificuldade em conseguir captar informação do ficheiro lpbib.txt, uma vez que alguns carateres desapareciam e, obviamente, ao fazer depois a comparação com o autor principal, este nunca surgia e, como tal, o grafo resultante seria sempre incorreto. Mas, como o trabalho de filtrar os autores e mudar os seus nomes para o formato "N. Apelido", já havia sido feito, então o grupo optou por receber como input o resultado da alínea b), que foi guardado no ficheiro graphTeste.txt. Desta forma, como os autores estão todos no mesmo formato, sendo que anteriormente uns estavam no formato "Nome Apelido" e outros em "Apelido, Nome", foi mais fácil agora proceder à recolha dos mesmos, aproveitando o trabalho efetuado anteriormente. No Anexo encontra-se um exemplo de um ficheiro *.dot* e o grafo resultante do mesmo. Convém referir que, para proceder à execução desta alínea, deve-se exutar o comando *make ARGS="N. Autor"*. Desta forma o programa irá executar uma *Makefile*, que vai executar o programa com o argumento dado.

Iremos proceder, agora, a uma breve análise às expressões regulares utilizadas nesta alínea e respetivas ações.

```
i. author[ ]*\=[ ]*[{"] {linha = NULL; BEGIN AUTOR;}
ii. <AUTOR>[\t" "\n]+ {}
iii. <AUTOR>[^\t\n]*/"\}," {char *authorName = strdup(yytext); linha = insert(linha, authorName);}
iv. <AUTOR>"\},\n" {autores = getAutores(autores, linha, author); BEGIN INITIAL;}
v. <AUTOR>[^\t\n]+ {char *authorName = strdup(yytext); linha = insert(linha, authorName);}
vi. .|\n {}
vii. <*><<EOF>>{printAutores(); free(linha); free(autores); return 0;}
```

A primeira expressão corresponde à situação em que se encontra o campo "author" de um registo. O programa inicializa a estrutura de dados referente aos autores encontrados num registo e entra no estado "AUTOR". A segunda expressão ocorre já dentro do estado "AUTOR" e serve apenas para limpar carateres que estejam a mais, ou seja, espaços, TAB's ou mudanças de linha. A terceira expressão apanha qualquer carater, exceto mudanças de linha e TAB's, eliminando as aspas, até encontrar uma chaveta para fechar o campo, apanhando assim o nome do autor (dado por *yytext*) e, de seguida efetua a inserção do mesmo na estrutura *linha*. A quarta expressão regular ocorre logo após a anterior e efetua a chamada da função *getAutores*, que verifica se um autor existe na *linha* e, em caso afirmativo, insere os restantes autores na estrutura principal (de nome *autores*) e, de seguida, abandona o estado "AUTOR". A quinta expressão efetua o mesmo trabalho da terceira, mas serve para autores que não sejam os últimos da lista, tendo as mesmas ações. A sexta expressão serve para, quando no estado inicial, ignorar qualquer carater ou mudança de linha. Convém notar que, como esta expressão regular vem depois das anteriores, não serão ignorados os carateres importantes, nomeadamente, os presentes no campo "author". Por fim, a última expressão ocorre aquando do final do ficheiro, sendo que aí serão imprimidos os autores no ficheiro graph.dot e efetadas as devidas limpezas na memória, devido às estruturas criadas.

Após a criação do ficheiro graph.dot, será executado um comando que irá criar um ficheiro de imagem com o grafo resultante. Convém referir, mais uma vez, que todos os comandos importantes estão contidos numa *Makefile*.

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

Para executar os vários testes à aplicação, foi corrida a seguinte *Makefile*:

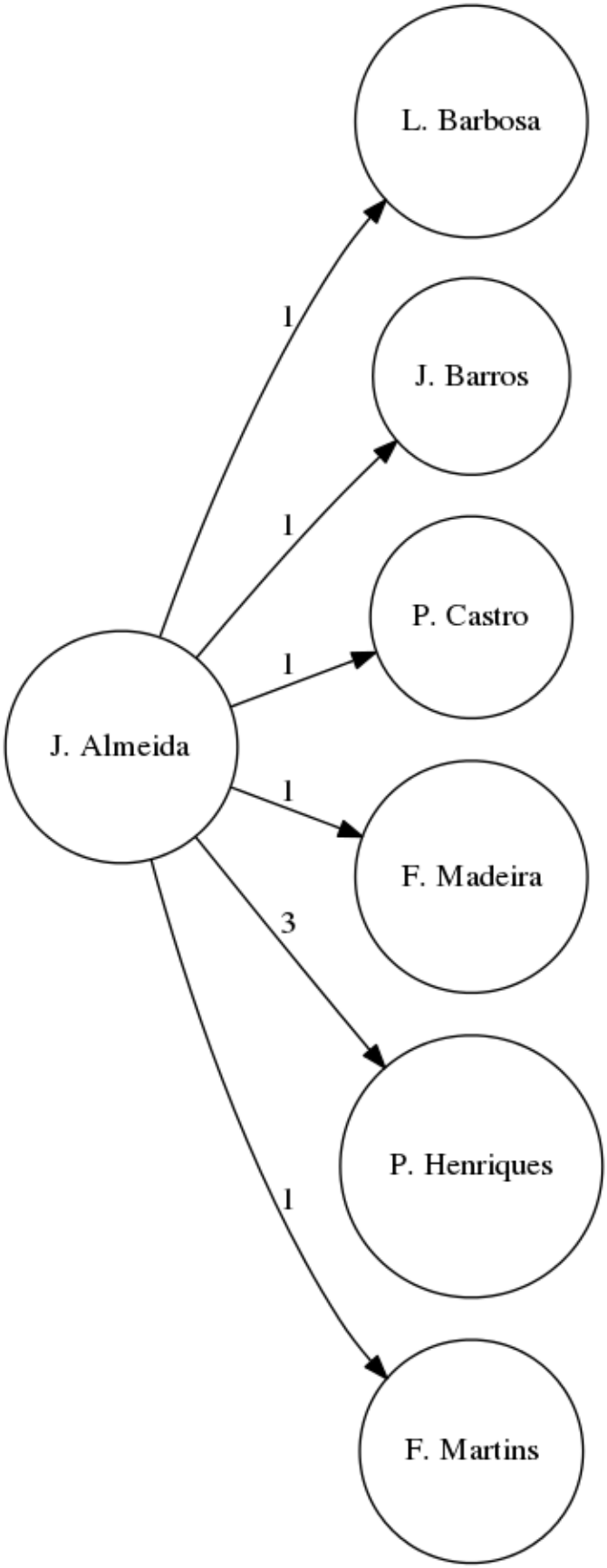
```
all: flexTree compileTree execTree createGraph

flexTree:
	flex tp1C.fl

compileTree:
	gcc lex.yy.c lista.c -ll -o tp1C

execTree:
	./tp1C < graphTeste.txt
```

E, de seguida, correr o seguinte comando:

```
make ARGS="J. Almeida"
```

De seguida, seguem abaixo o ficheiro *graph.dot* e o grafo respetivo:

```
digraph finite_state_machine {
    rankdir=LR;
    size="20,20"
node [shape = circle]; "J. Almeida" node [shape = circle]; "L. Barbosa" node [shape = circle
    ]; "J. Barros" node [shape = circle]; "P. Castro" node [shape = circle]; "F. Madeira"
    node [shape = circle]; "P. Henriques" node [shape = circle]; "F. Martins"
"J. Almeida""J. Almeida" -> "L. Barbosa" [label = "1"];"J. Almeida" -> "J. Barros" [label =
    "1"];"J. Almeida" -> "P. Castro" [label = "1"];"J. Almeida" -> "F. Madeira" [label =
    "1"];"J. Almeida" -> "P. Henriques" [label = "3"];"J. Almeida" -> "F. Martins" [label =
    "1"];
}
```


L. Barbosa
J. Barros
P. Castro
J. Almeida
F. Madeira
P. Henriques
F. Martins
1
1
1
1
3
1

Chapter 5

Conclusão

Como pudemos verificar ao longo da realização deste trabalho prático, o *Flex* é uma ferramenta bastante útil para efetuar um *parsing* a um ficheiro que contenha informação que possa ser útil em alguma situação.

Em relação ao trabalho efetuado, o grupo demonstra-se satisfeito com o resultado final do mesmo, apesar das dificuldades sentidas em alguns momentos. Conseguimos, numa primeira fase, proceder à contagem de todas as categorias existentes no ficheiro *BibTex* e, de seguida, efetuar a normalização dos campos e desenvolver uma ferramenta de *pretty-printing*, que embelezou o ficheiro inicial, sendo que o resultado desta tarefa revelou-se extremamente importante para a realização da última fase, que consistiu, na criação de um grafo de todos os autores que efetuaram publicações com um autor passado como argumento.

Para além destas tarefas, ainda optamos pela realização de mais uma tarefa (explicada em anexo), que consistiu na criação de *slides* em *Beamer*, utilizando para tal a ferramenta *Zim-wiki*. Mais uma vez e, à semelhança da tarefa anterior, os resultados gerados foram bastante satisfatórios, o que nos deixa bastante satisfeitos.

Futuramente, pretendemos aperfeiçoar os nossos conhecimento acerca desta e de outras ferramentas de *parsing* de documentos, a fim de estarmos habilitados a construir programas ainda mais sofisticados e de melhor qualidade.

Appendix 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

Ficheiro *tp1C.fl*:

```
%{

#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "lista.h"

char *author;
GRAPH autores;
GRAPH linha;

void printLinha(GRAPH linha);
void printAutores();

%}

%x AUTOR

%%

                        autores = insert(NULL, author);
author[ ]*\=[ ]*[{"]    {linha = NULL; BEGIN AUTOR;}
<AUTOR>[\t" "\n]+       {}
<AUTOR>[^\t\n]*/"\},"   {char *authorName = strdup(yytext); linha = insert(linha, authorName);}
<AUTOR>"\},\n"          {autores = getAutores(autores, linha, author); BEGIN INITIAL;}
<AUTOR>[^\t\n]+         {char *authorName = strdup(yytext); linha = insert(linha, authorName);}
.|\n                    {}
<*><<EOF>>              {printAutores(); free(linha); free(autores); return 0;}

%%

void printAutores() {
        FILE *fp = fopen("graph.dot", "w");
        GRAPH aux = autores;
        fprintf(fp, "digraph finite_state_machine {\n");
        fprintf(fp, "    rankdir=LR;\n");
        fprintf(fp, "    size=\"20,20\"\n");
        while(aux != NULL) {
                fprintf(fp, "node [shape = circle];");
                fprintf(fp, " \"%s\" ", aux->author);
                aux = aux->next;
        }
        aux = autores;
        while(aux != NULL) {
                if(strncmp(author, aux->author, strlen(author)) == 0) fprintf(fp, "\n\"%s\"", author);
                else fprintf(fp, "\"%s\" -> \"%s\" [label = \"%d\"];", author, aux->author, aux->num);
                aux = aux->next;
```

```
51          }
52          fprintf(fp, "\n}");
53          fclose(fp);
54  }
55
56  int main(int argc, char *argv[]) {
57          if(argc == 3) {
58                  int s;
59                  char *firstName = argv[1];
60                  char *lastName = argv[2];
61                  char *result = malloc(strlen(firstName) + strlen(lastName) + 2);
62                  strcpy(result, firstName);
63                  strcat(result, " ");
64                  strcat(result, lastName);
65                  author = strdup(result);
66                  s=yylex();
67                  while(s) {
68                          printf("%d",s);
69                          s=yylex();
70                  }
71                  free(result);
72          }
73          return 0;
74  }
```

Ficheiro *lista.c*:

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <errno.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <ctype.h>
7  #include "lista.h"
8
9  GRAPH insert(GRAPH g, char *author) {
10         if(g == NULL) {
11                 g = (GRAPH) malloc(sizeof(struct grafo_autores));
12                 g->author = strdup(author);
13                 g->num = 1;
14                 g->next = NULL;
15         }
16         else if(strncmp(author, g->author, strlen(author)) == 0) g->num++;
17         else g->next = insert(g->next, author);
18         return g;
19 }
20
21 GRAPH getAutores(GRAPH autores, GRAPH linha, char *author) {
22         GRAPH aux = linha;
23         while(aux != NULL) {
24                 if(strncmp(author, aux->author, strlen(author)) == 0) {
25                         aux = linha;
26                         while(aux != NULL) {
27                                 if(strncmp(author, aux->author, strlen(author)) != 0)
                                        autores = insert(autores, aux->author);
28                                 aux = aux->next;
29                         }
```

```
30                          break;
31                  }
32                  else aux = aux->next;
33          }
34          return autores;
35  }
```

Ficheiro *lista.h*:

```
1  #ifndef LISTA_H
2
3  #define LISTA_H
4
5  typedef struct grafo_autores {
6          char *author;
7          int num;
8          struct grafo_autores *next;
9  } GA;
10
11 typedef struct grafo_autores *GRAPH;
12
13 GRAPH insert(GRAPH g, char *author);
14 GRAPH getAutores(GRAPH autores, GRAPH linha, char *author);
15
16 #endif
```

A.4 Extra- Zim to Beamer

Decidiu-se também resolver o útlimo enunciado que fiz respeito á conversão de ficheiros *Zim-wiki* para slides *Beamer*, que é uma variação de *LaTex*, cujos slides são criados consoante o tipo de *Header* lido (apenas os tipos 3,4,5 permitem criar novo slide).

Ficheiro *Zim-Wiki* escrito em .txt,

```
====== Home ======
Created Tuesday 22 March 2016!
This is a test!


===== Header 2 =====
Should be a new slide!!
Next items should produce a **List in Bold** with three different bullets!

* item 1
* item 2
* item 3
      1. item 3
            a. item 3a


==== Header 3 ====
Should be a new slide!!
Now it should be appearing //check boxes in italic//
=== head 4 ===
Should not be a new __slide underlined__!!

[ ] empty
[*] checked wright
        [*] sub checked wright
        [ ] sub empty
[x] checked wrong


==== Header 3 ====
Should be a new slide!!

== head 5 ==
Should not be a new slide!!

Now an example of ~~strike through~~

To finish this presentation an image must be shown!
An import of an image!!!

{{./atom.jpg}}


====== End ======
Now an example of verbatim working ''int main() {return(0);} in verbatim''!!!
```

Ficheiro *Ficheiro .l* com as expressões regulares que permitem resolver o problema,

```
%{
        #include <stdio.h>
        int firstSlide=0;
        FILE *fp;
%}
%%
^[=]{6}[ ]{1}[a-zA-Z0-9 ]+[ ]{1}[=]{6}          {
                                                    if(firstSlide==0){
                                                    fp=fopen("zim2beamer.tex","w
                                                        ");
                                                    fprintf(fp,"%cdocumentclass{
                                                        beamer}\n %chypersetup{
                                                        pdfstartview={Fit}}\n %
                                                        cusetheme{Madrid}\n
                                                        ",92,92,92);
                                                    fprintf(fp,"%cusecolortheme{
                                                        beaver}\n %cusepackage{
                                                        pifont}\n %cusepackage{
                                                        cancel}\n %cbegin{
                                                        document}\n
                                                        ",92,92,92,92);
                                                    fprintf(fp,"%ctitle[Teste
                                                        Beamer]{The Real Deal}\n
                                                         %csubtitle{Example of
                                                        Beamer}\n",92,92);
                                                    fprintf(fp,"%cauthor[Ze
                                                        Carlos]\n",92);
                                                    fprintf(fp,"{F.~Author %
                                                        cinst{1} }\n %cinstitute
                                                        [University of Minho]{\n
                                                         %cinst{1}\n",92,92,92);
                                                    fprintf(fp,"Department of
                                                        Computer Science%c%c\n
                                                        ",92,92);
                                                    fprintf(fp,"University of
                                                        Minho\n}\n %cdate[PT
                                                        2016]{Try number 3,
                                                        2016}\n %csubject{
                                                        Computer Science}\n
                                                        ",92,92);
                                                    }
                                                    yytext[yyleng-7]='\0';
                                                        fprintf(fp,"\n%cbegin{
                                                        frame}\n\t%cframetitle{%
                                                        s}\n",92,92,yytext+7);
                                                        firstSlide++;

^[=]{5}[ ]{1}[a-zA-Z0-9 ]+[ ]{1}[=]{5}          {
                                                if(firstSlide==0){
                                                fprintf(fp,"%cdocumentclass{beamer}\
                                                    n %chypersetup{pdfstartview={Fit
                                                    }}\n %cusetheme{Madrid}\n
                                                    ",92,92,92);
```

```
23                                                  fprintf(fp,"%cusecolortheme{beaver}\
                                                        n %cusepackage{pifont}\n %
                                                        cusepackage{cancel}\n %cbegin{
                                                        document}\n",92,92,92,92);
24                                                  fprintf(fp,"%ctitle[Teste Beamer]{
                                                        The Real Deal}\n %csubtitle{
                                                        Example of Beamer}\n",92,92);
25                                                  fprintf(fp,"%cauthor[Ze Carlos]\n
                                                        ",92);
26                                                  fprintf(fp,"{F.~Author %cinst{1} }\n
                                                         %cinstitute[University of Minho
                                                        ]{\n %cinst{1}\n",92,92,92);
27                                                  fprintf(fp,"Department of Computer
                                                        Science%c%c\n",92,92);
28                                                  fprintf(fp,"University of Minho\n}\n
                                                         %cdate[PT 2016]{Try number 3,
                                                        2016}\n %csubject{Computer
                                                        Science}\n",92,92);
29                                                  }
30                                                  yytext[yyleng-6]='\0';fprintf(fp,"\n
                                                        %cbegin{frame}\n\t%cframetitle{%
                                                        s}\n",92,92,yytext+6);firstSlide
                                                        ++;
31
32  ^[=]{4}[ ]{1}[a-zA-Z0-9 ]+[ ]{1}[=]{4}         {
33                                                      if(firstSlide==0){
34                                                  fprintf(fp,"%cdocumentclass{beamer}\n %
                                                        chypersetup{pdfstartview={Fit}}\n %
                                                        cusetheme{Madrid}\n",92,92,92);
35                                                  fprintf(fp,"%cusecolortheme{beaver}\
                                                        n %cusepackage{pifont}\n %
                                                        cusepackage{cancel}\n %cbegin{
                                                        document}\n",92,92,92,92);
36                                                  fprintf(fp,"%ctitle[Teste Beamer]{
                                                        The Real Deal}\n %csubtitle{
                                                        Example of Beamer}\n",92,92);
37                                                  fprintf(fp,"%cauthor[Ze Carlos]\n
                                                        ",92);
38                                                  fprintf(fp,"{F.~Author %cinst{1} }\n
                                                         %cinstitute[University of Minho
                                                        ]{\n %cinst{1}\n",92,92,92);
39                                                  ffprintf(fp,"Department of Computer
                                                        Science%c%c\n",92,92);
40                                                  fprintf(fp,"University of Minho\n}\n
                                                         %cdate[PT 2016]{Try number 3,
                                                        2016}\n %csubject{Computer
                                                        Science}\n",92,92);
41                                                  }
42                                                  yytext[yyleng-5]='\0';fprintf(fp,"\n
                                                        %cbegin{frame}\n\t%cframetitle{%
                                                        s}\n",92,92,yytext+5);firstSlide
                                                        ++;
43
44  \n\n\n                                      {fprintf(fp,"%cend{frame}\n",92);}
45  ^[=]{3}[ ]{1}[a-zA-Z0-9 ]+[ ]{1}[=]{3}             {yytext[yyleng-4]='\0';fprintf(fp,"\n%s\n",
        yytext+4);}
```

```
^[=]{2}[ ]{1}[a-zA-Z0-9 ]+[ ]{1}[=]{2}          {yytext[yyleng-3]='\0';fprintf(fp,"\n%s\n",
    yytext+3);}
\{\{.*\}\}
                                                {yytext[yyleng-2]='\0';fprintf(fp,"\n%cbegin{
    figure}[!ht]\n\t %ccentering\n\t %cincludegraphics[width=0.25 %ctextwidth]{%s} \n\t %
    ccaption{Atom}\n %cend{figure}\n",92,92,92,92,yytext+4,92,92);}
\*.+\*
                                                {yytext[yyleng-2]='\0';fprintf(
    fp,"%ctextbf{%s}\n",92,yytext+2);}
\_.+\_
                                                {yytext[yyleng-2]='\0';fprintf(
    fp,"\n%cunderline{%s}\n",92,yytext+2);}
[/]{2}[a-zA-z0-9 ]+[/]{2}
            {yytext[yyleng-2]='\0';fprintf(fp,"\n%ctextit{%s}\n",92,yytext+2);}
[~]{2}[a-zA-z0-9 ]+[~]{2}
            {yytext[yyleng-2]='\0';fprintf(fp,"\n%ccancel{%s}\n",92,yytext+2);}
[']{2}.+[']{2}
                                        {yytext[yyleng-2]='\0';fprintf(fp,"\n%cend{frame}\n%
    cbegin{frame}[fragile]\n%cbegin{verbatim}\n%s\n%cend{verbatim}\n%cend{frame}\n
    ",92,92,92,yytext+2,92,92);}
[0-9]\.[ A-Za-z0-9]+
                    {fprintf(fp,"\n%cbegin{itemize}\n\n%cbegin{itemize}\n%citem %s\n%cend{
    itemize}\n%cend{itemize}\n",92,92,92,yytext,92,92);}
[a-z]\.[ ]{1}[A-Za-z0-9]+
            {fprintf(fp,"\n%cbegin{itemize}\n\n%cbegin{itemize}\n%citem %s\n%cend{itemize}\n
    %cend{itemize}\n",92,92,92,yytext,92,92);}
^[A-Z][A-Za-z0-9 \!]+
                    {fprintf(fp,"%s\n",yytext);}
\[[ ]\].*
                                        {fprintf(fp,"\n%cding{112} %s %cpar\n",92,yytext+4,92);}
.\[[ ]\].*
                                        {fprintf(fp,"%chspace{5ex}%cding{112} %s %cpar\n",92,92,
    yytext+4,92);}
.\[\*\].*
                                        {fprintf(fp,"%chspace{5ex}%cverb|%ccheckmark%s %cpar\n
    ",92,92,92,yytext+4,92);}
\[\*\].*
                                        {fprintf(fp,"%cverb|%ccheckmark %s %cpar\n",92,92,yytext
    +4,92);}
\[[X|x]\].*
                                {fprintf(fp,"%cverb|%cxmark X %s %cpar\n",92,92,yytext+4,92);}
\n\*[ ]{1}[a-zA-Z0-9 ]+
            {fprintf(fp,"\n%cbegin{itemize}\n%citem %s\n%cend{itemize}\n",92,92,yytext+3,92)
    ;}
.|\n
                                                                {}

%%
int main(){
        int s;
  s=yylex();
        while(s){fprintf(fp,"%d",s);s=yylex();}
        fprintf(fp,"%cend{document}\n",92);
        fclose(fp);
        return 0;
}
```

Ficheiro *Ficheiro .tex* com os slides criados a partir de *zim2beamer.txt*, que representa o conteúdo de um ficheiro *Zim-Wiki*.

Para criar os respetivos slides apenas bastará compilar o ficheiro .tex para se obter o ficheiro PDF.

```
\documentclass{beamer}
 \hypersetup{pdfstartview={Fit}}
 \usetheme{Madrid}
\usecolortheme{beaver}
 \usepackage{pifont}
 \usepackage{cancel}
 \begin{document}
\title[Teste Beamer]{The Real Deal}
 \subtitle{Example of Beamer}
\author[Ze Carlos]
{F.~Author \inst{1} }
 \institute[University of Minho]{
 \inst{1}
Department of Computer Science\\
University of Minho
}
 \date[PT 2016]{Try number 3, 2016}
 \subject{Computer Science}

\begin{frame}
        \frametitle{Home}
Created Tuesday 22 March 2016!
This is a test!
\end{frame}

\begin{frame}
        \frametitle{Header 2}
Should be a new slide!!
Next items should produce a
\textbf{List in Bold}

\begin{itemize}
\item item 1
\end{itemize}

\begin{itemize}
\item item 2
\end{itemize}

\begin{itemize}
\item item 3
\end{itemize}

\begin{itemize}

\begin{itemize}
\item 1. item 3
\end{itemize}
\end{itemize}

\begin{itemize}

```

```
\begin{itemize}
\item a. item
\end{itemize}
\end{itemize}
\end{frame}

\begin{frame}
        \frametitle{Header 3}
Should be a new slide!!
Now it should be appearing

\textit{check boxes in italic}

head 4
Should not be a new

\underline{slide underlined}

\ding{112} empty \par
\verb|\checkmark checked wright \par
\hspace{5ex}\verb|\checkmark sub checked wright \par
\hspace{5ex}\ding{112}  sub empty \par
\verb|\xmark X checked wrong \par
\end{frame}

\begin{frame}
        \frametitle{Header 3}
Should be a new slide!!

head 5
Should not be a new slide!!
Now an example of

\cancel{strike through}
To finish this presentation an image must be shown!
An import of an image!!!

\begin{figure}[!ht]
         \centering
         \includegraphics[width=0.25 \textwidth]{atom.jpg}
         \caption{Atom}
 \end{figure}
\end{frame}

\begin{frame}
        \frametitle{End}
Now an example of verbatim working

\end{frame}
\begin{frame}[fragile]
\begin{verbatim}
int main() {return(0);} in verbatim
\end{verbatim}
\end{frame}
\end{document}
```

Alguns *prints* do ficheiro PDF final, visto que não fazia sentido incluir um PDF dentro de outro. Pretende-se apenas mostrar três slides que contém a maior parte das características de uma apresentação.

Header 2

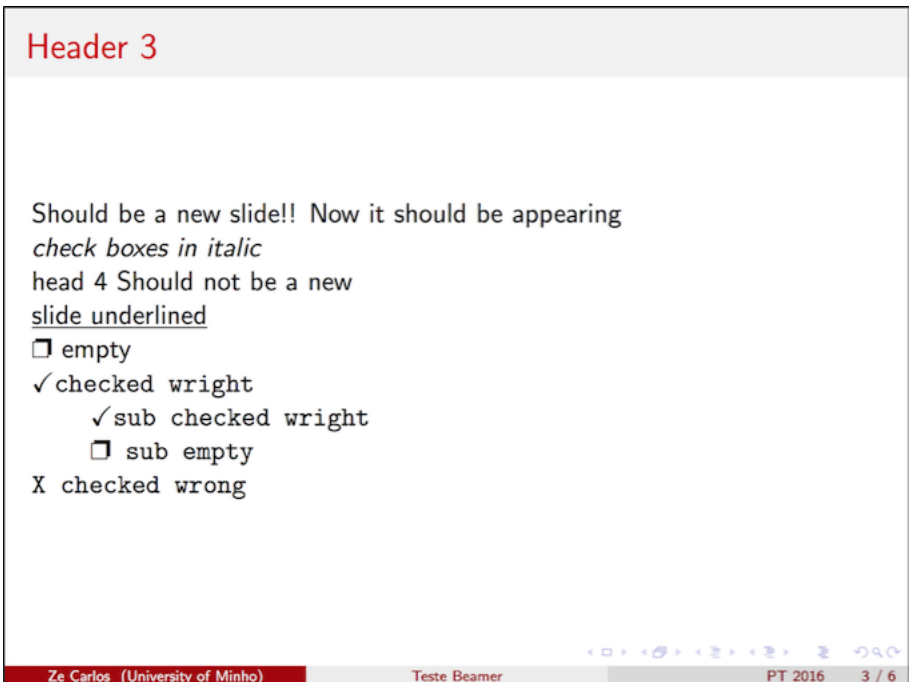
Should be a new slide!! Next items should produce a **List in Bold**

- item 1
- item 2
- item 3
  - 1. item 3
  - a. item

Header 3

Should be a new slide!!
head 5 Should not be a new slide!! Now an example of
~~strike through~~
To finish this presentation an image must be shown! An import of an image!!!

Figure: Atom



Durante a elaboração deste enunciado os testes realizados passaram pela geração do seus ficheiros PDF e ir comparando com o ficheiro .txt que possui o conteúdo de um ficheiro *Zim-Wiki* e observando se os slides estavam de acordo com a sua especificação.

O ficheiro .l contém todo o conjunto de expressões regulares que permitem capturar os padrões que dizem respeito ás regras de escrita de um ficheiro *zim* e as ações que toma, escrever o respectivo conteúdo no ficheiro *LaTex Beamer*.

Os problemas que ocorreram relacionam-se com o o facto de por vezes não estar alinhado mas nada de especial. O maior problema foi apenas o facto de ao fazer "fprintf" para o ficheiro .tex não colocava 'é como tal recorreu-se á utilização do seu valor *ASCII*, que é o número 92 e verificava-se quando surge o símbolo percentagem seguido de 'c'.

Bibliography

[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`