Processamento de Linguagens (3º ano de Curso)

Trabalho Prático 3

Relatório de Desenvolvimento

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

9 de Junho de 2018

Resumo

Este projeto tem como principal objetivo a interação por parte dos alunos com ferramentas de apoio a programação. Neste sentido, permite assim, aumentar a capacidade destes relativamente à escrita de gramáticas, bem como permitiu aprofundar conhecimentos relativos a yacc e novamente a flex (com a necessidade da criação de um analisador léxico). Em suma, neste relatório descrevemos todos os esforços efetuados para dar resposta ao enunciado proposto. Nesse sentido, o presente relatório aborda num primeiro momento a conceção do ficheiros yacc e flex, e um segundo momento, a análise dos resultados obtidos. Destes, é de realçar a criação de um grafo utilizando a linguagem *dot*.

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 exemplificando um pouco o que nos foi pedido. 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 *dot*, uma linguagem de descrição de grafos em texto puro. Nesse sentido, o enunciado propunha, através de uma análise de uma base de dados de emigrantes [1] a construção de um grafo. Assim, deverá ser permitida uma navegação visual sobre esse repositório de conhecimento. Deverá ser permitido, ao selecionar um nodo, passar para uma página web a qual contem informação sobre esse elemento. Os nodos serão de dois tipos. Um dos nodos representará os emigrantes e toda a informação referente. O outro, será referente a uma operação, sendo que cada uma possui uma *label* com a designação de "fez"ou "participa". Para dar resposta ao pretendido, deverá ser elaborado uma gramática em YACC e o respetivo analisador léxico.

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 contem uma série de estruturas já implementadas. Assim, a utilização das estruturas que envolviam a manipulação de Hash Tables facilitaram todo o trabalho.

3.2 Resolução do enunciado

De acordo com o que era pedido no enunciado, dividimos a conceção da solução em duas partes. Uma primeira em que o objetivo foi tornar o trabalho útil, que pudesse ser aplicado na vida real. A segunda parte prende-se com o que é pedido pelo enunciado do projeto, isto é, com a criação de uma linguagem que permitisse gerar vários grafos, tornando mais facilmente interpretável um determinado conjunto de dados.

- **Primeira parte**

Para que o projeto pudesse ser utilizado pela comunidade em geral, pensamos que as informações dos diversos emigrantes pudessem ser incluídas num ficheiro Excel (tal como o exemplo indicado), mais propriamente numa extensão deste, um ficheiro *CSV*. Assim, sempre que fosse necessário introduzir novas informações, estas poderiam ser introduzidas por qualquer um, de modo fácil e intuitivo, apenas através do preenchimento das colunas com os novos dados. Posteriormente, uma vez que o ficheiro se encontrava escrito em *CSV*, desenvolvemos um programa com o auxílio da ferramenta *GAWK*, apenas transformando todo o documento segundo a linguagem que definimos na fase seguinte.

```
BEGIN { FS = "," }

{ if (NR == 1) next; }

{ print("emigrante {\n\t"$1";\n\t"$2";\n\t"$3";\n\t"$4";\n\t"$5";\n\t"$6";\n\t"$7";\n\t"$8";\n}\n"); }

END { }
```

Destino	Nome	CC	Idade	Profissão	Sexo	Origem	Ano_Partida
France	Marylynne Took	12842191	25	Project Manager	F	Kenwood	2002
Yemen	Scarface Willshaw	12842192	41	Legal Assistant	M	Menomonie	1996
American Samoa	Celle Vellender	12842193	67	Software Consultant	F	Kim	2010
Russia	Gweneth Brazear	12842194	18	Sales Associate	F	Arkansas	2011
Russia	Rachele Pfeffle	12842195	36	Help Desk Operator	F	Artisan	2003
China	Carce Moreinu	12842196	57	Office Assistant II	M	Lerdahl	2009
Philippines	Quinlan Greville	12842197	51	Recruiting Manager	M	Grover	1996
France	Prisca Braunstein	12842198	77	Account Executive	F	Ridge Oak	2003
Serbia	Alvie Ciobutaro	12842199	95	General Manager	M	Banding	2009
Thailand	Florentia Haw	12842200	89	Account Executive	F	Schurz	1990
China	Bancroft Dallicoat	12842201	63	Internal Auditor	M	Crowley	1993
China	Angelia Wombwell	12842202	17	Research Nurse	F	Hudson	1995

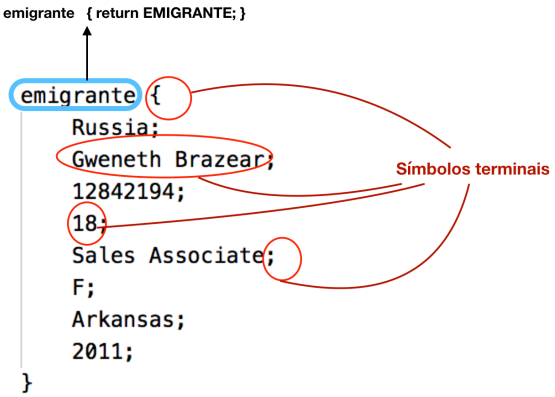
```
emigrante {
    France;
    Marylynne Took;
    12842191;
    25;
    Project Manager;
    F;
    Kenwood;
    2002;
}

emigrante {
    Yemen;
    Scarface Willshaw;
    12842192;
    41;
    Legal Assistant;
    M;
    Menomonie;
    1996;
}
```

Figura 3.1: Conversão de csv para a linguagem definida

- **Segunda parte**

Nesta parte, criamos uma linguagem conforme as necessidades do problema. Esta linguagem é caracterizada pelo seu aspeto simples e conciso. De facto, o emigrante, possui uma estrutura semelhante a uma estrutura de dados em C, em que as informações deste se encontram delimitadas por chaveta. Cada informação, é então separada pelo ponto e vírgula. O mesmo se verifica no caso de uma operação. Posteriormente, deu-se a conversão de cada um dos elementos num grafo multi-nível. Consideramos que o primeiro nível conteria um determinado conjunto de países, depois, já num outro nível temos os emigrantes nesses países e por fim ao carregar sobre um emigrante, obtemos todas as suas informações bem como as operações que este fez ou participou. Assim, criamos todas as funções que consideramos necessárias para que fosse possível esta transição.




```
Z : S
;

S :
  | S EMIGRANTE '{' Dados '}'
  | S OPERACAO '{' Infs '}'
;

Dados :
      | Dados f ';'
;

Infs :
      | Infs f ';'
;

f : STR
  | f STR
;
```

Figura 3.2: Exemplificação de um ficheiro de input.

```
S :

  | S EMIGRANTE '{' Dados '}'
```

```
{ }

{
  LPersonalData* pd = createLPersonalData();
  insertHTCountries(pd);
  Events* ev = createEvent(pd->id, pd->name);
  insertHTEvents(ev);
}
```

Figura 3.3: Funções necessárias à criação do grafo.

Capítulo 4

Testes

4.1 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (dados introduzidos) e os respetivos resultados obtidos:

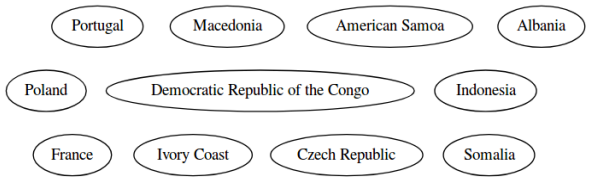


Figura 4.1: Exemplificação de um ficheiro: 1º nível.

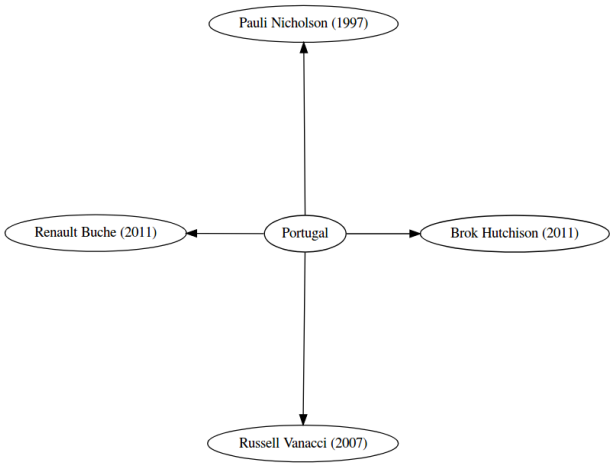


Figura 4.2: Exemplificação de um ficheiro: 2º nível.

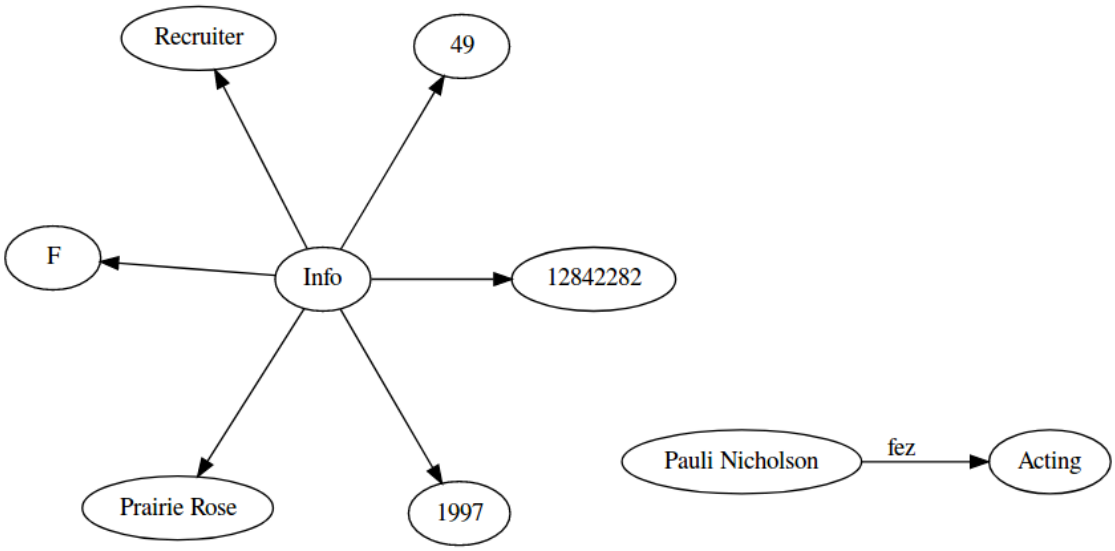


Figura 4.3: Exemplificação de um ficheiro: 3º nível.

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 que diz respeito à criação de um analisador léxico, bem como a ferramenta *YACC* para a criação de linguagens.

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 interpretação do que realmente seria necessário, de acordo com o que era pedido no enunciado. No entanto, o grupo resolveu todo o problema com o método mais adequado, sendo ainda possivel o desenvolvimento de trabalho extra relativamente ao que era pedido no guião. Assim, é percetível o interesse de todo o grupo.

Em suma, o trabalho foi bastante benéfico para todos os elementos do grupo, sendo possível criar uma ferramenta que poderá ter um destino útil no futuro.

Bibliografia

[1] www.museu-emigrantes.or acedido no dia 24 pelas 15h40m.

Apêndice A

Código do Programa

- **Analisador léxico**

```
%option noyywrap yylineno

%%
emigrante            { return EMIGRANTE; }
operacao             { return OPERACAO;  }
[{};]                { return yytext[0]; }
[a-zA-Z0-9]*         { yylval.s = strdup(yytext); return STR; }
[ \n\t&\/\(\)'.]     { }
.                    { yyerror("Invalid char\n"); }
%%
```

- **Programa YACC**

```
%{
#define _GNU_SOURCE
#include "structs.h"

int asprintf(char** strp, const char* fmt, ...);
int yylex();
void yyerror(char* c);
%}

%union  { char* s; }
%token EMIGRANTE OPERACAO STR
%type <s> EMIGRANTE OPERACAO STR Z S Dados Infs f

%%
Z : S                                          { }
;

S :                                            { }
  | S EMIGRANTE '{' Dados '}'                  {
                                                 LPersonalData* pd = createLPersonalData();
                                                 insertHTCountries(pd);
                                                 Events* ev = createEvent(pd->id, pd->name);
                                                 insertHTEvents(ev);
                                                }

  | S OPERACAO '{' Infs '}'                    {
```

```
                                                  LOperations* op = createLOperations();
                                                  insertHTOperations(atoi(lineReader 0]), op);
                                                }
;

Dados :                                           { $$ = ""; }
      | Dados f ';'                               { strcpy(lineReader[indReader], $2); indReader++; }
;

Infs :                                            { $$ = ""; }
     | Infs f ';'                                 { strcpy(lineReader[indReader], $2); indReader++; }
;

f : STR                                           { $$ = $1;  }
  | f STR                                         { asprintf(&$$, "%s %s", $1, $2); }
;
%%

#include "lex.yy.c"

int main() {
    initLineReader();
    initFile();
    countries = g_hash_table_new(g_str_hash, g_str_equal);
    ids = g_hash_table_new(g_int_hash, g_int_equal);
    yyparse();
    /*
    for (int i = 0; i < 8; i++) {
        printf("%s\n", lineReader[i]);
    } */
    g_hash_table_foreach(countries, (GHFunc)printHTCountries, NULL);
    g_hash_table_foreach(ids, (GHFunc)printHTOperations, NULL);
    endFile();
    return 0;
}

void yyerror(char* s) {
    fprintf(stderr, "%s, '%s', line %d \n", s, yytext, yylineno);
}
```

- **Programas GAWK**

```
BEGIN { FS = "," }

{ if (NR == 1) next; }

{ print("emigrante {\n\t"$1";\n\t"$2";\n\t"$3";\n\t"$4";\n\t"$5";\n\t"$6";\n\t"$7";\n\t"$8";\n}\n"); }

END { }
```

```
BEGIN { FS = "," }

{ if (NR == 1) next; }

{ print("operacao {\n\t"$1";\n\t"$2";\n\t"$3";\n}\n"); }

END { }
```

- **Código C auxiliar**
  - **structs.h**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <glib.h>

/* Structs */
typedef struct operations {
    char* oper;
    char* description;
    struct operations* next;
} LOperations;

typedef struct events {
    int id;
    char* name;
    LOperations* ops;
} Events;

typedef struct data {
    char* destination;
    char* name;
    int id;
    int age;
    char* occupation;
    char* gender;
    char* source;
    int departureYear;
    struct data* next;
} LPersonalData;

int indReader;
char* lineReader[8];

GHashTable* countries;
GHashTable* ids;


/* Functions */
void initLineReader();

void initFile();

void endFile();

LPersonalData* createLPersonalData();

void insertHTCountries(LPersonalData* pd);

Events* createEvent(int id, char* name);

void insertHTEvents(Events* ev);

LOperations* createLOperations();

void insertHTOperations(int id, LOperations* op);

void printHTCountries(gpointer key, gpointer value, gpointer user_data);

void printHTOperations(gpointer key, gpointer value, gpointer user_data);
```

– **structs.c**

```
#include "structs.h"

void initLineReader() {
    indReader = 0;

    lineReader[0] = (char*)malloc(sizeof(char) * 50);
    lineReader[1] = (char*)malloc(sizeof(char) * 50);
    lineReader[2] = (char*)malloc(sizeof(char) * 10);
    lineReader[3] = (char*)malloc(sizeof(char) * 3);
    lineReader[4] = (char*)malloc(sizeof(char) * 20);
    lineReader[5] = (char*)malloc(sizeof(char) * 10);
    lineReader[6] = (char*)malloc(sizeof(char) * 50);
    lineReader[7] = (char*)malloc(sizeof(char) * 4);
}

LPersonalData* createLPersonalData() {
    LPersonalData* pd = (LPersonalData*)malloc(sizeof(struct data));
    pd->destination = (char*)malloc(sizeof(char) * 50);
    pd->name = (char*)malloc(sizeof(char) * 50);
    pd->occupation = (char*)malloc(sizeof(char) * 20);
    pd->gender = (char*)malloc(sizeof(char) * 10);
    pd->source = (char*)malloc(sizeof(char) * 50);

    strcpy(pd->destination, lineReader[0]);
    strcpy(pd->name, lineReader[1]);
    pd->id = atoi(lineReader[2]);
    pd->age = atoi(lineReader[3]);
    strcpy(pd->occupation, lineReader[4]);
    strcpy(pd->gender, lineReader[5]);
    strcpy(pd->source, lineReader[6]);
    pd->departureYear = atoi(lineReader[7]);

    pd->next = NULL;

    indReader = 0;

    return pd;
}

void insertHTCountries(LPersonalData* pd) {
    LPersonalData* old = (LPersonalData*)g_hash_table_lookup(countries, pd->destination);
    pd->next = old;

    g_hash_table_insert(countries, pd->destination, pd);
}

Events* createEvent(int id, char* name) {
    Events* ev = (Events*)malloc(sizeof(struct events));
    ev->name = (char*)malloc(sizeof(char) * 50);
    ev->ops = NULL;

    ev->id = id;
    strcpy(ev->name, name);

    indReader = 0;

    return ev;
}

void insertHTEvents(Events* ev) {
    gint* key = g_new(gint, ev->id);
    *key = ev->id;

    g_hash_table_insert(ids, key, ev);
}
```

```
LOperations* createLOperations() {
    LOperations* op = (LOperations*)malloc(sizeof(struct operations));
    op->oper = (char*)malloc(sizeof(char) * 10);
    op->description = (char*)malloc(sizeof(char) * 50);

    strcpy(op->oper, lineReader[1]);
    strcpy(op->description, lineReader[2]);
    op->next = NULL;

    indReader = 0;

    return op;
}

void insertHTOperations(int id, LOperations* op) {
    gint* key = g_new(gint, id);
    *key = id;

    Events* ev = (Events*)g_hash_table_lookup(ids, key);
    if (ev != NULL) {
        if (ev->ops != NULL) {
            op->next = ev->ops;
            ev->ops = op;
        } else {
            ev->ops = op;
        }
    }

    g_hash_table_insert(ids, key, ev);
}

void printHTCountries(gpointer key, gpointer value, gpointer user_data) {
    LPersonalData* pd = (LPersonalData*)value;
    FILE* f = NULL;
    FILE* f2 = NULL;
    char* filenameDOT = (char*)malloc(sizeof(char) * 100);
    char* filenameSVG = (char*)malloc(sizeof(char) * 100);
    char s[20];
    char* fname = (char*)malloc(sizeof(char) * 20);

    if (pd != NULL) {
        //printf("----------------%s----------------\n", pd->destination);
        f = fopen("dot_files/countries.dot", "a");
        fprintf(f, "\t\"%s\" [URL=\"%s.svg\"];\n", pd->destination, pd->destination);
        fclose(f);

        strcpy(filenameDOT, "dot_files/");
        strcpy(filenameSVG, "svg_files/");
        strcat(filenameDOT, pd->destination);
        strcat(filenameSVG, pd->destination);
        strcat(filenameDOT, ".dot");
        strcat(filenameSVG, ".svg");

        f2 = fopen(filenameDOT, "a");
        fprintf(f2, "digraph G {\n\tlayout=\"circo\"\n");
    }

    while (pd != NULL) {

        fprintf(f2, "\t\"%s\" -> \"%s (%d)\"\n", pd->destination, pd->name, pd->departureYear);
        fprintf(f2, "\t\"%s (%d)\" [URL=\"%d.svg\"];\n", pd->name, pd->departureYear, pd->id);

        sprintf(s, "%d", pd->id);

        strcpy(fname, "dot_files/");
        strcat(fname, s);
        strcat(fname, ".dot");
```

```
        FILE* f3 = fopen(fname, "a");

        fprintf(f3, "digraph G {\n\tlayout=\"circo\"\n");
        fprintf(f3, "\t\"Info\" -> \"%d\"\n", pd->id);
        fprintf(f3, "\t\"Info\" -> \"%d\"\n", pd->age);
        fprintf(f3, "\t\"Info\" -> \"%s\"\n", pd->occupation);
        fprintf(f3, "\t\"Info\" -> \"%s\"\n", pd->gender);
        fprintf(f3, "\t\"Info\" -> \"%s\"\n", pd->source);
        fprintf(f3, "\t\"Info\" -> \"%d\"\n", pd->departureYear);
        fclose(f3);

        pd = pd->next;

        memset(fname, 0, 20);
    }


    fprintf(f2, "}");
    fclose(f2);

    char* cmd = (char*)malloc(sizeof(char) * 150);
    strcpy(cmd, "dot -Tsvg -o ");
    strcat(cmd, filenameSVG);
    strcat(cmd, " ");
    strcat(cmd, filenameDOT);

    int x;
    if ( ( x = fork() ) == 0 ) {
        execlp("dot", "dot", "-Tsvg", "-o", filenameSVG, filenameDOT, NULL);
    }

    free(filenameDOT);
    free(filenameSVG);
    free(fname);
    free(cmd);
}

void printHTOperations(gpointer key, gpointer value, gpointer user_data) {
    Events* ev = (Events*)value;
    LOperations* ops = ev->ops;
    FILE* f = NULL;
    char* a = (char*)malloc(sizeof(char) * 20);
    char* b = (char*)malloc(sizeof(char) * 20);
    char s[20];

    if (ev != NULL) {
        //printf("----------------%d -> %s----------------\n", ev->id, ev->name);


        sprintf(s, "%d", ev->id);

        strcpy(a, "dot_files/");
        strcpy(b, "svg_files/");
        strcat(a, s);
        strcat(b, s);
        strcat(a, ".dot");
        strcat(b, ".svg");

        f = fopen(a,"a");
    }

    while (ops != NULL) {
        //printf(":: op->%s\n description->%s\n", ops->oper, ops->description);
        fprintf(f, "\t\"%s\" -> \"%s\" [label=\"%s\"];\n", ev->name, ops->description, ops->oper);
        ops = ops->next;
    }
```

```
    fprintf(f,"}\n");
    fclose(f);

    int x;
    if ( ( x = fork() ) == 0 ) {
        execlp("dot", "dot", "-Tsvg", "-o", b, a, NULL);
    }

    free(a);
    free(b);
}

void initFile() {
    FILE* f = fopen("dot_files/countries.dot","a");
    fprintf(f,"digraph G {\n\tlayout=\"circo\"\n");
    fclose(f);
}

void endFile() {
    FILE* f = fopen("dot_files/countries.dot","a");
    fprintf(f,"}\n");
    fclose(f);
}
```