Processamento de Linguagens e Compiladores
(3º ano de LCC - Pedro Rangel Henriques)

Trabalho Prático Nº3 (YACC)

2. Informação Geográfica
Relatório de Desenvolvimento

Francisca Fernandes (A72450)

Francisco Oliveira (A82066)

Maria Luísa Silva (A82124)

25 de Julho de 2021

Resumo

Isto é um resumo do relatório de *Processamento de Linguagens e Compiladores* focado no contexto do trabalho sobre *Informação Geográfica*, os objetivos concretos e os resultados, tanto como as dificuldades atingidas por nós ao longo deste último trabalho.

Conteúdo

Capítulo 1

Introdução

No terceiro trabalho prático, inserido no âmbito da disciplina *Processamento De Linguagens e Compiladores*, o principal objetivo é desenvolver um processador de linguagem segundo o método de *tradução dirigida pela sintaxe*, suportado numa *gramática tradutora(GT)*.

Além disso, o ponto que tem mais foco neste trabalho é, a capacidade de se escrever *gramáticas independentes de contexto(GIC)* para a criação de Linguagens de Domínio Específico(DSL).

1.1 Enunciado

O objectivo deste projecto é construir um site para as localidades de Portugal face à sua hierarquia geográfica definida à custa da linguagem **OrgGeo** abaixo descrita. **OrgGeo** é uma Linguagem de Domínio Específico **(DSL)** que se destina a descrever a organização geo-política de um país, ou sua região. Para o efeito, pretende-se identificar as Localidades de cada Concelho de cada Distrito da região em causa.

A gramática independente de contexto G, abaixo apresentada, define o tipo de linguagem pretendida–cada grupo pode criar a sua variante.
O Símbolo Inicial é **OrgGeo**, os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre apostrofes (sinais de pontuação) e a string nula é denotada por "&"; os restantes serão os Símbolos Não-Terminais.

```
OrgGeo   --> Distritos
Distritos--> IdD DescD '.'
           | Distritos IdD Desc '.'
DescD    --> Concelhos
Concelhos--> Concelho
           | Concelhos ';' Concelho
Concelho --> Locais ':' IdC
Locais   --> Local LstIds
LstIds   --> &
           | ',' Locais
IdD      --> id
IdC      --> id
Local    --> IdL .....
IdL      --> id
```

A informação a incluir em cada Local fica ao cuidado do grupo, sendo que se deve contemplar a possibilidade de fornecer um URL para um site do local; caso tal suceda, no site gerado existirá um link para esse site do local.

Uma possível forma de desenvolver o projecto é definida a seguir: Transforme G numa **gramática tradutora**, GT, reconhecível pelo yacc, para gerar um sítio HTML com uma página por cada Distrito, contendo uma lista de items com os Concelhos e para cada um uma outra lista com suas localidades. Complete a GT anterior de modo a gerar também uma página inicial (a *homepage* do seu sítio) com um título (fixo) do projecto e links para as páginas dos Distritos.

Adicionalmente, estenda o seu processador para incluir a construção duma **Tabela de Localidades** que associe a cada identificador de Localidade os respectivos Concelho e Distrito. Note que podem existir Localidades com nomes repetidos, desde que tal não aconteça no mesmo Concelho. Nomes de Concelhos e de Distritos é que não podem ser duplicados.

Capítulo 2

Análise da Linguagem Desenvolvida

2.1 Descrição informal do problema

Com a realização deste trabalho pretende-se especificar uma gramática de acordo com o nosso enunciado proposto, desenvolver um reconhecedor léxico e sintático para essa linguagem, com recurso a ferramentas geradoras, como o Flex e o Yacc e, construir um gerador de **HTML** que produza a resposta sugerida pelo enunciado do problema.

2.2 Decisões e Problemas de Implementação

Para ser possível criar a **Tabela de Localidades** pedida no enunciado é necessário associar a cada Localidade o seu Concelho e Distrito, para isto decidimos usar uma *HashTable* que associava a cada localidade o concelho e o distrito em que se integra.
Como chave desta *HashTable* era necessário usar algo único à Localidade em questão, infelizmente não pudemos usar o nome da Localidade como identificador devido ao facto de não ser único globalmente(no enunciado é dito que as localidades podem ter nomes repetidos).
Decidimos então criar uma variavél global `loc_id` que é incrementada sempre que o analisador da gramática encontra uma nova Localidade, assim atribuindo um *id* único a cada Localidade.

Posteriormente deparamo-nos com outra dificuldade no preenchimento da informação da **Tabela de Localidades** uma vez que no momento em que é executado a ação semântica associada a uma Localidade(Local), ainda não temos informação do concelho e distrito em que essa Localidade se enquadra devido ao Analisador Gramatical(*Yacc*) funcionar de um modo 'BottomUp', ou seja, as regras gramaticais do `Concelho` e `Distritos` só são aplicadas após a construção de todos os Nodos Não-Terminais que as compõem.

Conseguimos resolver este problema atribuindo o Concelho de cada Localidade já encontrada(cujo Concelho ainda não tenha sido preenchido) na execução da ação semântica da regra do Concelho que sabemos que só é executado após todas as suas localidades terem sido encontradas pelo *parser*. O método foi o mesmo para atribuir os Distritos às Localidades.

Capítulo 3

Implementação/ desenho da Resolução e Codificação

3.1 Gramática

Usando como base a gramática sugerida no enunciado, começamos por definir os símbolos terminais e não terminais e só depois criar as produções para a formação da linguagem.

Os **Símbolos Terminais** são os separadores de contéudo (como `.:;,=`) e também `ID` e `LINK`.

Com isto em mente, obtivemos a seguinte gramática:

```
OrgGeo   --> Distritos
Distritos--> IdD DescD '.'
           | Distritos IdD Desc '.'
DescD    --> Concelhos
Concelhos--> Concelho
           | Concelhos ';' Concelho
Concelho --> Locais ':' IdC
Locais   --> Local LstIds
LstIds   --> &
           | ',' Locais
IdD      --> ID
IdC      --> ID
Local    --> IdL '=' LocalHyperlink
           | IdL
IdL      --> ID
LocalHyperlink --> LINK
```

3.1.1 Código desenvolvido

O código final relativo a, quer seja o **Analisador Léxico(Flex)** como o **Gerador de Parsers(Yacc)** estão nos anexos deste relatório.

3.1.2 Analisador Léxico

O nosso analisador léxico foi relativamente trivial, visto que apenas vamos encontrar nomes, links de locais e separadores de hierarquia.

Usamos apenas uma expressão regular para obter o valor ASCII correspondente a cada símbolo.

Capítulo 4

Testes e Resultados

Após alguns testes extremamente simples, um ex-colega de curso mencionou que tinha conhecimento de um documento com todas as localidades de Portugal e voluntariou-se para criar um tradutor para a nossa linguagem.
Assim conseguimos criar o exemplo perfeito para o que nos foi pedido.

Distritos

- LEIRIA
- PORTO
- GUARDA
- EVORA
- VIANA DO CASTELO
- SANTAREM
- FARO
- VILA REAL
- BEJA
- BRAGA
- BRAGANCA
- COIMBRA
- VISEU
- LISBOA
- CASTELO BRANCO
- AVEIRO
- PORTALEGRE
- SETUBAL

Tabela de Localidades

- **AMOR** - Concelho: LEIRIA Distrito: LEIRIA
- **ARRABAL** - Concelho: LEIRIA Distrito: LEIRIA
- **BAROSA** - Concelho: LEIRIA Distrito: LEIRIA
- **AZOIA** - Concelho: LEIRIA Distrito: LEIRIA
- **CARANGUEJEIRA** - Concelho: LEIRIA Distrito: LEIRIA
- **CARVIDE** - Concelho: LEIRIA Distrito: LEIRIA
- **BOA VISTA** - Concelho: LEIRIA Distrito: LEIRIA
- **COIMBRAO** - Concelho: LEIRIA Distrito: LEIRIA
- **BARREIRA** - Concelho: LEIRIA Distrito: LEIRIA
- **CORTES** - Concelho: LEIRIA Distrito: LEIRIA
- **COLMEIAS** - Concelho: LEIRIA Distrito: LEIRIA
- **MACEIRA** - Concelho: LEIRIA Distrito: LEIRIA
- **LEIRIA** - Concelho: LEIRIA Distrito: LEIRIA
- **MILAGRES** - Concelho: LEIRIA Distrito: LEIRIA
- **MARRAZES** - Concelho: LEIRIA Distrito: LEIRIA
- **MONTE REDONDO** - Concelho: LEIRIA Distrito: LEIRIA
- **MONTE REAL** - Concelho: LEIRIA Distrito: LEIRIA
- **PARCEIROS** - Concelho: LEIRIA Distrito: LEIRIA
- **ORTIGOSA** - Concelho: LEIRIA Distrito: LEIRIA
- **REGUEIRA DE PONTES** - Concelho: LEIRIA Distrito: LEIRIA

Figura 4.1: Exemplo Final - Homepage

Figura 4.2: Exemplo Final - Página de Braga

Capítulo 5

Conclusão

Com este trabalho fomos capazes de aumentar o nosso conhecimento sobre desenho e implementação de processadores para além do que foi lecionado nas aulas, melhorando também, o nosso conhecimento em outras vertentes importantes no ramo da programação.

Embora algumas dificuldades que se tenham apresentado ao longo do trabalho consideramos que atingimos os objetivos pertendidos com o mesmo.

Apêndice A

Código Flex

```
1
2 %{
3 #include <stdio.h>
4 #include <stdlib.h>
5 %}
6
7 %%
8 \"[^\"]*\" { yylval.texto = strdup(yytext); return LINK; }
9 [a-zA-Z][a-zA-Z0-9|_| |\(|\)|\'|\-|\/]* { yylval.texto = strdup(yytext); return ID; }
10 [\.\:\;\,\=] { return yytext[0]; }
11 .|\n                          { ; }
12 %%
13
14 int yywrap() {
15     return 1;
16 }
```

Apêndice B

Código Yacc

```
%{
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <glib.h>

/* Declaracoes C diversas */

int endr(char *id);


/* Criar a hash table */

GHashTable* hash_table;
int localidade_id;

char* homepage_html;


typedef struct {
char* locais_html;
char* id_concelho;
} Concelho_info;

typedef struct {
int loc_id;
char* localidade_nome;
char* concelho_nome;
char* distrito_nome;
} Localidade_Info;


int yyerror(const char *s);
void print_distrito_file(const char* distrito_name, const char* conteudo);
void generate_distrito(char** destination, const char* conteudo);
void generate_concelho(char* destination, const char* conteudo);
int register_localidade(Localidade_Info loc);
void preencher_localidades_distrito(char* distrito);
```

```
void preencher_localidades_concelho(char* concelho);
void print_localidade(gpointer key, gpointer value, gpointer str);
int print_localidade_table(char** destination);
void add_homepage_distrito(const char* distrito);

%}

%union {
char* texto;
Concelho_info conselho;
}

%token <texto>TID_DISTRITO
%token <texto>TID_CONCELHO
%token <texto>TID_LOCAL
%token <texto>ID
%token <texto>LINK

%type <texto>DescD
%type <texto>IdD
%type <texto>Concelhos
%type <texto>Locais
%type <texto>IdC
%type <conselho>Concelho
%type <texto>Local
%type <texto>IdL
%type <texto>LstIds
%type <texto>LocalHyperlink

%%

OrgGeo      : Distritos { }
;
Distritos   : IdD DescD '.' {
preencher_localidades_distrito($1);
asprintf(&$2,
"%s<a href='index.html#%s'>Voltar à Pagina Inicial</a>",
$2, $1);
print_distrito_file($1, $2);
add_homepage_distrito($1);
}

    | Distritos IdD DescD '.' {
preencher_localidades_distrito($2);
add_homepage_distrito($2);
asprintf(&$3,
"%s<a href='index.html#%s'>Voltar à Pagina Inicial</a>",
$3, $2);
print_distrito_file($2, $3);
}
    ;

DescD       : Concelhos { generate_distrito(&$$, $1); }
;
```

```
Concelhos   : Concelho {
asprintf(&$$, "<h2>%s</h2>\n%s", $1.id_concelho, $1.locais_html);
}
            | Concelhos ';' Concelho {
asprintf(&$$, "%s\n<h2>%s</h2>\n%s", $1, $3.id_concelho, $3.locais_html);
}
;
Concelho    : Locais ':' IdC {
asprintf(&$$.id_concelho, "%s", $3);
preencher_localidades_concelho($3);
asprintf(&$$.locais_html, "<ol>%s</ol>", $1);
}
;
Locais  : Local LstIds {
if($2 != NULL)
{
asprintf(&$$, "%s<li>%s</li>", $2, $1);
} else {
asprintf(&$$, "<li>%s</li>", $1);
}
}
LstIds  : { $$ = ""; }
| ',' Locais { asprintf(&$$, "%s", $2); }
IdD : ID
IdC : ID
Local : IdL '=' LocalHyperlink {
Localidade_Info info;
asprintf(&info.localidade_nome, "%s", $1);
register_localidade(info);
if($3 != NULL)
{
asprintf(&$$, "<a target='_blank' href=%s>%s</a>", $3, $1);
}
 }
| IdL {
Localidade_Info info;
asprintf(&info.localidade_nome, "%s", $1);
register_localidade(info);
}
IdL : ID
LocalHyperlink : LINK { $$ = strdup($1); }

%%

#include "lex.yy.c"

int endr(char *id) {
    return 0;
}

void print_distrito_file(const char* distrito_name, const char* conteudo)
{
char szFileName[256];
sprintf(szFileName, "%s.html", distrito_name);
```

```
FILE* f = fopen(szFileName, "w+");
fprintf(f, "%s", conteudo);
fclose(f);
}

void generate_distrito(char** destination, const char* conteudo)
{
asprintf(destination,
 "<html>\n"
 "<head>\n"
 "</head>\n"
 "<body>\n"
 "%s\n"
 "</body>\n"
 "</html>", conteudo);
}

int yyerror(const char *s) {
    fprintf(stderr, "ERRO: %s \n", s);
    return 0;
}

int register_localidade(Localidade_Info loc)
{
Localidade_Info* locinfo_dynamic = malloc(sizeof(Localidade_Info));
asprintf(&locinfo_dynamic->localidade_nome, "%s", loc.localidade_nome);
locinfo_dynamic->concelho_nome = NULL;
locinfo_dynamic->distrito_nome = NULL;
//asprintf(&locinfo_dynamic->concelho_nome, "%s", loc.concelho_nome);
//asprintf(&locinfo_dynamic->distrito_nome, "%s", loc.distrito_nome);
locinfo_dynamic->loc_id = localidade_id++;
g_hash_table_insert(hash_table,
(gpointer)&locinfo_dynamic->loc_id,
locinfo_dynamic);
}

void preencher_localidade_concelho(gpointer key, gpointer value, gpointer user)
{
const char* concelho = (const char*)user;
Localidade_Info* loc_info = (Localidade_Info*)value;
if(loc_info->concelho_nome == NULL)
{
asprintf(&loc_info->concelho_nome, "%s", concelho);
}
}


void preencher_localidades_concelho(char* concelho)
{
g_hash_table_foreach(hash_table, preencher_localidade_concelho, concelho);
}

void preencher_localidade_distrito(gpointer key, gpointer value, gpointer user)
{
```

```
const char* distrito = (const char*)user;
Localidade_Info* loc_info = (Localidade_Info*)value;
if(loc_info->distrito_nome == NULL)
{
asprintf(&loc_info->distrito_nome, "%s", distrito);
}
}

void preencher_localidades_distrito(char* distrito)
{
g_hash_table_foreach(hash_table, preencher_localidade_distrito, distrito);
}

void print_localidade(gpointer key, gpointer value, gpointer str)
{
Localidade_Info* loc_info = (Localidade_Info*)value;
char** destination = (char**)str;
asprintf(destination, "%s\n<li><b>%s</b> - Concelho: %s Distrito: %s\n",
*destination, loc_info->localidade_nome,
loc_info->concelho_nome, loc_info->distrito_nome);
}

void add_homepage_distrito(const char* distrito)
{
asprintf(&homepage_html, "%s<li><a id='%s' href='%s.html'>%s</a></li>\n",
homepage_html, distrito, distrito, distrito);
}

int build_localidade_table(char** destination)
{
char* localidade_string;
asprintf(&localidade_string, "<ul>");
g_hash_table_foreach(hash_table, print_localidade, &localidade_string);
asprintf(destination, "%s</ul>", localidade_string);
}

int main()
{
hash_table = g_hash_table_new(g_int_hash, g_int_equal);
localidade_id = 0;
asprintf(&homepage_html, "<html>\n"
"<head><title>OrgGeo</title></head>\n"
"<body>\n"
"<h1> Distritos </h1>\n"
"<ul>");

    yyparse();
asprintf(&homepage_html, "%s\n"
"</ul>\n", homepage_html);

char* localidades_table = NULL;
build_localidade_table(&localidades_table);

// Falta construir página inicial
```

```
asprintf(&homepage_html, "%s\n<h2>Tabela de Localidades</h2>\n%s\n",
homepage_html, localidades_table);
asprintf(&homepage_html, "%s\n</body>\n</html>", homepage_html);
puts(homepage_html);
free(localidades_table);
    return(0);
}
```

Apêndice C

Makefile

```
start:
	flex scanner.l
	yacc -d -v grammar.y
	gcc `pkg-config --cflags --libs glib-2.0` -o tradutor y.tab.c -l glib-2.0

clear:
	rm -f *.html
	rm -f tradutor
	rm -f y.tab.c
	rm -f lex.yy.c
	rm -f y.output
	rm -f y.tab.h

import:
	python importar.py > OrgGeo.txt

generate:
	./tradutor < OrgGeo.txt > index.html
```