Processamento de Linguagens
MIEI (3º Ano)
# Trabalho Prático Nº 3 (YACC)
Relatório de Desenvolvimento


| Rui Costa | Rafael Silva | Ricardo Pereira |
|---|---|---|
| (79947) | (74264) | (77045) |


10 de Junho de 2019

**Resumo**

O presente relatório foi produzido no âmbito da Unidade Curricular de Processamento de Linguagens do 3º ano do Mestrado Integrado em Engenharia Informática. Este consiste na implementação de um reconhecedor léxico e sintático, através do uso de FLEX e YACC.

# Conteúdo

# Capítulo 1

# Introdução

Este trabalho tem como objetivos o aumento da experiência de uso do ambiente Linux e de o uso de algumas ferramentas que dão suporte à programação, como o gerador de filtros de texto FLEX e ainda o gerador de compiladores baseado em gramáticas tradutoras, como o Yacc. Com a utilização do Flex, tivemos de por à prova os nossos conhecimentos sobre Expressões Regulares, uma vez que,com este foi construímos um analisador léxico, que serviu de suporte para a gramática. Este analisador permitiu que conseguissemos desenvolver as produções necessárias para construir a gramática tradutora que traduzisse o problema proposto. De entre os enunciados disponibilizados, o grupo ficou com a tarefa de desenvolver exercício 4, - *Linguagem para definição de dados genealógicos.*

# Capítulo 2

# Linguagem para definição de dados genealógicos

## 2.1 Contextualização

O compilador ou interpretador de uma linguagem de programação é composto por duas partes, a leitura do programa fonte, descobrindo assim a sua estrutura, e o processamento dessa mesma estrutura, ou seja a geração desse mesmo programa, resultando assim no seu executável, O Lex e o Yacc, em conjunto, geram fragmentos do programa que descobrem a sua estrutura. O lex lê o ficheiro de input e descobre os seus tokens, enquanto que o Yacc encontra a estrutura hierarquica do programa.

## 2.2 Análise de extratos

### 2.2.1 Input

O problema que nos foi proposto, passa pela análise de informação sobre a vida de uma pessoa, contida num ficheiro, e respetiva compactação de forma a que a informação seja perceptível e sem qualquer perda de informação.

```
Manuel da Silva *1977 +2011 [3]  // nome, nascimento, morte, II=3
M  Maria da Silva +2009          // mãe nome, morte da mãe
P  Joaquim Oliveira da Silva     // Pai
MM Joaquina *1930                // mãe da mãe
MP [45]                          // pai da mãe é o #I45, descrito anteriormente
FOTO f.jpg
HIST h1.tex
CC 2000 [2]                      // #I3 casou-se em 2000 com #I8, IF=2
Maria Felisbina *1980 [8]        // Conjugue, nome, nascimento, II=8
F  Serafim da Silva *2004        // Filho (ref. ao CC anterior [3][8])
F  Ana da Silva *2006 [7]{       //
     FOTO f1.jpg                 // dados extra referentes à Ana #I7
     HIST h1.tex
   }
```

Figura 2.1: Exemplo de texto-fonte na linguagem sugerida

### 2.2.2 Output

Depois da análise de ficheiros dados como exemplo, chegamos a conclusão que o ficheiro de output na primeira linha contém sempre a informção relativa à pessoa a quem pertence a história(nome, respetivos eventos, e respetivo id). Também reparámos que cada individuo, se não contém um id na sua identificação, terá de ser identificado por "autn", sendo o n igual ao número de individuos que ja apareceram sem id mais um.

```
#I3 nome Manuel da Silva
#I3 data-nacimento  1977
#I3 data-falecimento 2011
#I3 tem-como-mae #aut1
#aut1 nome Maria da Silva
#aut1 data-nascimento 2009
#I3 tem-como-mae #aut2
#aut2 nome Joaquim Oliveira da Silva
#I3 temo-como-MM #aut3
#aut3 nome Joaquina
#aut3 data-nascimento 1930
#I3 temo-como-MP #I45
#I3 FOTO f.jpg
#I3 HIST h1.tex
#F2 = #I3 #I8
#F2 data-casamento 2000
#I8 nome Maria Felisbina
#I8 data-nascimento 1980
#aut4 nome Serafim da Silva
#aut4 data-nascimento 2004
#F2 tem-como-filho #aut4
#I7 nome Ana da Silva
#I7 data-nascimento 2006
#F2 tem-como-filho #I7
#I7 FOTO f1.jpg
#I7 HIST h1.tex
```

Figura 2.2: Output pretendido correspondente ao exemplo de input anterior

### 2.2.3 Gramática

Nesta secção iremos definir a liguagem dada como solução do problema, analisando todos os constituintes de uma gramática. De acordo com o estudado ao longo do semestre, define-se uma gramática para a representação de uma linguagem imperativa como a junção dos quatro conjuntos <T,N,S,P>, respetivamente símbolos Terminais, Não-Terminais, axioma da gramática e produções.

#### Terminais

Os símbolos terminais são os que podem aparecer como entrada ou saída de uma produção, dos quais não pode derivar mais nenhuma unidade. Por convenção, serão escritos a letra minúscula. As suas definições explicitam adequadamente as suas funções na execução do programa.

```
T = {'exit_comm','identifier','nascimento','falecimento',
'casamento,'nome','parentesco','foto','hist','newline'}
```

### Não-Terminais

Os símbolos terminais são os que podem aparecer como saída de uma produção, dos quais obrigatoriamente deriva uma ou mais unidades. Por convenção foram escritos a letra maiúscula.

```
N = {'LINE', 'PESSOAPRINCIPAL', 'EVENTOS', 'EVENTO', 'PESSOA'}
```

### Axioma

O axioma é a raiz da árvore de derivação, do qual deriva a primeira produção.

```
S = {NGen}
```

### Produções

Uma gramática é definida pelas regras de produção que especificam que símbolos podem substituir outros. Todas as derivações do conjunto de testes fornecido seguem as seguintes regras.

```
P = {
    p1:NGen -> LINE
    p2:NGen -> NGen LINE
    p5:LINE -> PESSOAPRINCIPAL
    p6:LINE -> exit_comm
    p7:LINE -> newline
    p10:PESSOAPRINCIPAL -> nome EVENTOS '['identifier']'
    p11:EVENTOS -> EVENTO
    p12:EVENTOS -> EVENTOS EVENTO
    p13:EVENTOS -> EVENTOS newline EVENTO
    p14:EVENTO -> nascimento
    p15:EVENTO -> falecimento
    p16:EVENTO -> nascimentoIncerto
    p17:EVENTO -> falecimentoIncerto
    p18:EVENTO -> casamento PESSOA
    p19:EVENTO -> evento
    p20:EVENTO -> parentesco PESSOA
    p21:EVENTO -> foto
    p22:EVENTO -> hist
    p23:EVENTO ->
    p24:PESSOA -> nome '['identifier']'
    p25:PESSOA -> nome EVENTO '['identifier']'
}
```

## 2.3 Código

### 2.3.1 Analisador Léxico

```
%option noyywrap

%{
#include "y.tab.h"
%}

ano [0-9]{1,}

%%

"exit"                              {return exit_comm;}
[0-9]+                              {yylval.id=atoi(yytext);return identifier;}
(\-)?(PP|MM|PM|MP|P|M|F)            {yylval.str=strdup(yytext);return parentesco;}
cc\({ano}                           {yylval.ano=atoi(yytext+3); return casamento;}
ev\({ano}                           {yylval.str=strdup(yytext+3); return evento;}
[a-zA-Z]+(\/[a-zA-Z]+)?(\%[0-9]+)?  {yylval.str=strdup(yytext); return nome;}
\*{ano}                             {yylval.ano=atoi(yytext+1); return nascimento;}
\*c{ano}                            {yylval.ano=atoi(yytext+2); return nascimentoIncerto;}
\+{ano}                             {yylval.ano=atoi(yytext+1); return falecimento;}
\+c{ano}                            {yylval.ano=atoi(yytext+2); return falecimentoIncerto;}
.*\.jpg                             {yylval.str=strdup(yytext); return foto;}
.*\.tex                             {yylval.str=strdup(yytext); return hist;}
"\n"                                {return newline;}
[\[\]]                              {return yytext[0];}
.                                   {;}

%%
```

### 2.3.2 Gerador de Compiladores

```
%{
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
char *nomePrincipal,*eventos,*token;
int idP, idE;
FILE *fp;
int count=1;
extern int yylex();
extern int yylineno;
void yyerror(char *s);
%}
```

```
%union{int id;int ano; char *str;char c;}
%start NGen
%token exit_comm
%token <id> identifier
%token <ano> nascimento nascimentoIncerto falecimento falecimentoIncerto casamento evento
%token <str> nome parentesco foto hist
%token <c> newline
%type <str> LINE PESSOAPRINCIPAL EVENTOS EVENTO PESSOA

%%

NGen        : LINE                  {;}
              | NGen LINE               {;}
              ;

LINE        : PESSOAPRINCIPAL       {;}
              | exit_comm               {fclose(fp); exit(EXIT_SUCCESS);}
              | newline                 {;}
              ;

PESSOAPRINCIPAL : nome EVENTOS '['identifier']' {
                                    if(strchr($1,'/')){
                                        char* tmp=strdup($1);
                                        char* token=strtok(tmp,"/");
                                        fprintf(fp,"#I%d nome %s ",$4,token);
                                        token=strtok(NULL,"/");
                                        fprintf(fp,"apelido %s\n",token);
                                    }else
                                        fprintf(fp,"#I%d nome %s\n",$4,$1);
                                    if(strchr($1,'%')){
                                        char* tmp=strdup($1);
                                        char* token2=strtok(tmp,"%");
                                        token2=strtok(NULL,"%");
                                        fprintf(fp,"Existem %s pessoas com este nome\n",token2);
                                    }
                                    eventos=malloc(sizeof(char)*strlen($2)+1);
                                    strcpy(eventos,$2);
                                    if(eventos){
                                        char * token=strtok(eventos,"$");
                                        while(token!=NULL){
                                            fprintf(fp, "#I%d %s\n",$4,token);
                                            token=strtok(NULL,"$");
                                        }
                                    }
                                }
                                ;
```

```
EVENTOS         : EVENTO                    { $$=$1; }
                | EVENTOS EVENTO            { asprintf(&$$,"%s%s",$1,$2); }
                | EVENTOS newline EVENTO    { asprintf(&$$,"%s%s",$1,$3); }
                ;

EVENTO  : nascimento        { asprintf(&$$,"$nasceu em %d",$1); }
            | falecimento       { asprintf(&$$,"$morreu em %d",$1); }
            | nascimentoIncerto { asprintf(&$$,"$nasceu cerca de %d",$1); }
            | falecimentoIncerto{ asprintf(&$$,"$morreu cerca de %d",$1); }
            | casamento PESSOA  { asprintf(&$$,"$casou em %d com #aut%d\n%s",$1,count++,$2); }
            | evento            {
                    char* tmp=strdup($1);
                    char* token=strtok(tmp,":");
                    token=strtok(NULL,":");
                    asprintf(&$$,"$evento %s em %s\n",token,$1);
                                }
            | parentesco PESSOA {
        if(!strcmp($1,"PP")) { asprintf(&$$,"$pai-do-pai #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"MM")) { asprintf(&$$,"$mae-da-mae #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"P")) { asprintf(&$$,"$tem-como-pai #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"M")) { asprintf(&$$,"$tem-como-mae #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"MP")) { asprintf(&$$,"$mae-do-pai #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"PM")) { asprintf(&$$,"$pai-da-mae #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"F")) { asprintf(&$$,"$tem-como-filho #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"-PP")) { asprintf(&$$,"$neto(a) #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"-MM")) { asprintf(&$$,"$neto(a) #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"-P")) { asprintf(&$$,"$filho(a) #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"-M")) { asprintf(&$$,"$filho(a) #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"-MP")) { asprintf(&$$,"$neto(a) #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"-PM")) { asprintf(&$$,"$neto(a) #aut%d\n%s",count++,$2); }
        else if(!strcmp($1,"-F") { asprintf(&$$,"$pai/mae #aut%d\n%s",count++,$2); }
        else { printf("parentesco desconhecido"); }
        }
            | foto              { asprintf(&$$,"$%s",$1); }
            | hist              { asprintf(&$$,"$%s",$1); }
            ;

PESSOA      : nome '['identifier']'         { asprintf(&$$,"#aut%d nome %s\n#aut%d id %d",count,$1
                | nome EVENTO '['identifier']'  { asprintf(&$$,"#aut%d nome %s\n#aut%d id %d\n#aut
                ;

%%

void yyerror(char *s){
    printf("erro: %s\nlinha %d",s,yylineno);
    exit(1);
}
```

```
int main(){
    fp=fopen("config.out","a");
    while(1)
        yyparse();
    return 0;
}
```

### 2.3.3 Makefile

```
virtual: lex.yy.o y.tab.o
        gcc -o ngen y.tab.o lex.yy.o -ll
        ./ngen < config.in
        rm *.o *.c *.h

y.tab.o: y.tab.c
        gcc -c y.tab.c

y.tab.c y.tab.h: ngen.y
        yacc -d ngen.y

lex.yy.c: ngen.l y.tab.h
        flex ngen.l
```

## 2.4 Análise de resultados

### 2.4.1 Teste 1

Num primeiro teste utilizamos o seguinte input:

```
Ricardo/Pereira%2 *1975 +c2089
FOTO foto.jpg
HIST hist.tex
cc(2001) Ana [1]
P Antonio [3]
M Isabel [4] [2]
exit
```

Foi gerado o seguinte output:

```
#I2 nome Ricardo apelido Pereira%2
Existem 2 pessoas com este nome
#I2 nasceu em 1975
#I2 morreu cerca de 2089
#I2 FOTO foto.jpg
#I2 HIST hist.tex
#I2 casou em 2001 com #aut1
#aut1 nome Ana
#aut1 id 1
#I2 tem-como-pai #aut2
```

```
#aut2 nome Antonio
#aut2 id 3
#I2 tem-como-mae #aut3
#aut3 nome Isabel
#aut3 id 4
```

### 2.4.2 Teste 2

Como no primeiro teste foi gerado o output pretendido, neste segundo teste utilizamos um input mais complexo:

```
Mario/Rodrigues *1992 +2102
FOTO fotoCV.jpg
HIST historia.tex
cc(2014) Carla [9]
P Joao [10]
-P Bruno [13]
M Ana [22]
MM Rita +1933 [1]
PP Rafael [90] [66]
exit
```

Foi gerado o seguinte output:

```
#I66 nome Mario apelido Rodrigues
#I66 nasceu em 1992
#I66 morreu em 2102
#I66 FOTO fotoCV.jpg
#I66 HIST historia.tex
#I66 casou em 2014 com #aut1
#aut1 nome Carla
#aut1 id 9
#I66 tem-como-pai #aut2
#aut2 nome Joao
#aut2 id 10
#I66 filho(a) #aut3
#aut3 nome Bruno
#aut3 id 13
#I66 tem-como-mae #aut4
#aut4 nome Ana
#aut4 id 22
#I66 mae-da-mae #aut5
#aut5 nome Rita
#aut5 id 1
#aut5
#I66 morreu em 1933
#I66 pai-do-pai #aut6
#aut6 nome Rafael
#aut6 id 90
```

# Capítulo 3

# Conclusão

Findo o último trabalho da UC, podemos afirmar que a solução atingida poderia ser melhorada, pois foi desde o início desenhada uma solução que se provou não seguir à regra o pressuposto no enunciado. Isto deveu-se em parte a pequenas incongruências no mesmo, no que diz respeito à morfologia do input, comparada com o exemplo dado. Daí, o grupo partiu para uma solução que cobre grande parte das situações que são apresentadas, tendo a conclusão divergido do esperado. Ainda assim, o trabalho serviu o propósito do aprofundamento do conhecimento na matéria, proporcionando obstáculos que tentámos ultrapassar.