# Compiladores

Universidade de Trás-os-Montes e Alto Douro

2º Ano Eng.Informática

# Anuário de Medicamentos

Relatório de Desenvolvimento

Bernardo Meneses 74116 Andreia Queirós 73997
Adriana Paiva 73489 Tiago Fernandes 73701

22 de Janeiro 2021

**Resumo**

No âmbito da disciplina 'Compiladores' leccionada pela professora Teresa Perdicoúlis , foi-nos solicitado que implementássemos um sistema de consulta desses medicamentos , que seja acessivel a qualquer farmácia através de um browser HTML.

A finalidade deste projeto é desenvolver ...

Torna-se crucial a leitura deste trabalho uma vez que nos dá noções sobre uma aplicação complexa , sem privilegios especiais , que traduz uma descrição numa linguagem fonte numa descrição equivalente numa liguagem destino : um compliador. A fim de compreender melhor o funcionamento de um processador de linguagens foi-nos proposta a implementação de um reconhecedor léxicoe de um reconhecedor sintático usando a linguagem C. De forma a conhecer melhor as ferramentas auxiliares para o desenvolvimento automático dos processadores de linguagens foi nos sugerida a utilização do Lex e do Yacc , de modo a gerar a partir destes um reconhecedor léxico e sintático, respetivamente.

# Conteúdo

# Capítulo 1

# Introdução

No âmbito da unidade curricular 'Compiladores' orientada pela professora Teresa Perdicoúlis foi-nos proposto implementar um sistema de consulta de medicamentos , que seja acessivel a qualquer farmacia atraves de um browser HTML.

Neste trabalho iremos começar por desenvolver um analisador léxico (AL) usando a linguagem Lex para este devolver a informação envolvida no lote de medicamentos a considerar.

De seguida iremos desenvolver um analisador sintático(AS), isto lê a informação devolvida pelo AL e verifica se a informação em causa respeita as regras da gramática utilizada.

Também incluimos um analisador semântico (ASem), este descodifica o significado da frase, e, então, valida se a informação em causa cumpre as condições em causa q a torne semanticamente válida.

Estes dois últimos (AS,ASem), irão ser desenvolvidos em Yacc

# Capítulo 2

# Análise e Especificação

## 2.1 Descrição informal do problema

Este projeto tem a finalidade da implementação para auxiliar o Instituto Farmacêutico do Ministério de Saude na gestão de medicamentos , podendo ser consultados e que qualquer farmácia tenha acesso atarvés de um browser.

## 2.2 Especificação do Requisitos

### 2.2.1 Dados

Para a realização deste trabalho usamos a aplicação ... com extensões a liguagem C , Yacc e Lex. Fizemos um relatório em Latex usando o Overleaf. Ao longo das aulas fomos aprendendo e apontando material de apoio para a execução deste projeto .Também usamos o relatório fornecido pela docente para orientar o desempenho desta atividade

### 2.2.2 Pedidos

A finalidade deste trabalho é definir uma linguagem para descrever a infromação envolvida no lote de medicamentos a considerar e desenvolver um AL usando o Lex para reconhecer todos os símbolos terminais dessa linguagem e devolver os respetivos códigos. Já a Analise Sintatica e semãntica desenvolvido, em Yacc, vai receber os

símbolos do AL e verificar a sequencia em causa se respeita a derivação,ou produçoes da gramtica, bem como validar se uos simbolos estao semanticamente corretos , fazendo tambem um tradutor , a partir da gramatica tradutora da linguagem a processar.

### 2.2.3 Relações

Para a implementação dos analisadores utliziamos..... com extensoes da Linguagem C , Yacc e Lex para servir de analisador sintatico e léxico. Como editor do texto para o desenvolvimento do trabalho recorremos ao Latex. Auxiliamo-nos tanto aos apontamentos das aulas como o respetivo protocolo .

# Capítulo 3

# Concepção/desenho da Resolução

## 3.1 Estruturas e Dados e Algoritmos

Neste trabalho começamos a implementação do ficheiro lex, uma vez que para nós é a matéria de mais fácil compreensão, onde desenvolvemos os tokens que achamos necessários para a implementação do código Na etapa seguinte, desenvolvemos o ficheiro yacc que se revelou mais complicado e quase um quebra-cabeças, uma vez que não percebiamos muito bem da matéria. Deparados com este situação, decidimos falar com outros grupos com os quais trocámos ideias e, recebemos sugestões cruciais para o desenvolvimento desta parte do trabalho. Com isto, começamos por definir os tokens que iam ser usados no lex e os types, neste caso, alguns grupos que definimos para fazer a análise sintática. Também definimos uma estrutura e dentro dessa estrutura fizemos uma divisão onde acabamos com alguns types como Med que neste caso faz a análise da frase sobre o medicamento em sí que contem types como o nome, a empresa, o preço,... Após fazer isto construímos o Equals e Repetables para este fazer no Med a comparação com os dados existentes numa struct criada para guardar os dados dos medicamentos, onde este quando repete um medicamento irá apresentar no medicamento. No final verificamos os erros do programa e construímos uma função print para colocar no ficheiro de saída os dados pretendidos pelo utilizador. Não conseguimos colocar o acesso ao website em html.

```
%{
#include "y.tab.h"
#include <stdlib.h>
#include <stdio.h>
%}
%option yylineno
%%
[ |\t|\n]+                                  {yylval.string = strdup(yytext); return BRANCOS;}
[1-9]+[0-9]*                                {yylval.num = atoi(yytext); return INT; }
[0-9]+[.][0-9]*                             {yylval.rnum = atof(yytext); return REAL; }
[A-Za-z]+                                   {yylval.string = strdup(yytext); return ID; }
"["                                         {yylval.string = strdup(yytext); return OPEN_SQUARE_BRACK
"]"                                         {yylval.string = strdup(yytext); return CLOSE_SQUARE_BRAC
"("                                         {yylval.string = strdup(yytext); return OPEN_PARENTHESIS;
")"                                         {yylval.string = strdup(yytext); return CLOSE_PARENTHESIS
"{"                                         {yylval.string = strdup(yytext); return OPEN_SQUARE_BRACK
"}"                                         {yylval.string = strdup(yytext); return CLOSE_SQUARE_BRAC
","                                         {yylval.string = strdup(yytext); return COMMA;}
"."                                         {yylval.string = strdup(yytext); return DOT;}
":"                                         {yylval.string = strdup(yytext); return COLON;}
.                                           fprintf(stderr, "%d:Unexpected char: '%s'\n", yylineno, y
%%
int yywrap(void)
{
    return -1;
}
```

```
Symposium: 2022
[Analgesico,Antibiotico,Brufen]
[
/- Medicamento 1 -/
(Moment,1,Analgesico,Paracetamol,4.5,{Roche},{Qualquer-Bial});
/- Medicamento 2 -/
(Antigripine,2,Antibiotico,Acetalilico,6.,{Fabt,Faba,Fabb},{Aga-Fabc,Agb-Fab
/- Medicamento 3 -/
(Brufen,3,Analgesico,Ibuprofeno,400,{Mylan},{Qualquer-detestoCompiladores});
/- Medicamento 4 -/
(Spidifen,20,Analgesico,Ibuprofeno,400,{Zambom},{Qualquer-duyy});
```

```
Ficheiro  Editar  Formatar  Ver  Ajuda
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
extern int yylex();
extern int yylineno;
extern void yyerror(char *);


int nClasses = 0;



int n_Fabricantes = 0;
int ano;

typedef struct med{
        char nome[25], classe[25], comp[20], fabricantes[50], equivalentes[50];
        int codigo, nfabricantes;
        float preco;
} medic;

medic med;
medic **Fabricantes = NULL;




%}
%union
{
    char *string;
}

%start Document

// tokens here
%token COLON DOT OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET OPEN_PARENTHESIS CLOSE_PARENTHESIS COMM
%token <string> SYMP ID
%token <num> INT
%token <rnum> REAL

%type <string> IdList Name Code Category ChemComp Med MedList Price Repetable Repetables Equal Equals EqualList
%type Classes
```

```
%%
Document: Header Body
        ;


Classes: ID                                     {nClasses++;};
           | Classes COMMA ID           {nClasses++;};
     ;

Header: SYMP COLON INT { ano = $3; }
      ;

Body:ClassHeader ClassBody
    ;



IdList: ID
              | IdList COMMA ID
                {
                    char * newIdent = $3;
                    char * others = $1;
                    strcat(others, ", ");
                    strcat(others, newIdent);
                    $$ = others;
                }
              ;

ClassHeader: OPEN_SQUARE_BRACKET IdList CLOSE_SQUARE_BRACKET
           ;

MedList: Med
             | MedList Med
             ;

ClassBody: OPEN_SQUARE_BRACKET MedList CLOSE_SQUARE_BRACKET // Corresponde a zona onde tem a lista de medicamentos
         ;



Med: OPEN_PARENTHESIS Name COMMA Code COMMA Category COMMA ChemComp COMMA Price COMMA Repetables COMMA Equals CLOSE_PAR
        {
            int i=0, j=0, existeFabricante = 0;
```

```
            med.nome = strdup($2);
            med.codigo = strdup($4);
            med.classe = strdup($6);
            med.comp = strdup($8);
            med.preco = strdup($10);
            med.fabricantes = str_split_comma(strdup($12), &(med.nFabricantes));


            // Adicionar à lista apenas os fabricantes que ainda não tenham aparecid
            for(i=0; i<med.nFabricantes; i++){
                int compare = 0;
                for(j=0; j < n_Fabricantes; j++){
                    if(strcmp(med.fabricantes[i], todosFabricantes[j])==0){
                        compare = 1;
                        break;
                    }
                }
                if(!compare){
                    addString(&Fabricantes, &n_Fabricantes, (med.fabricantes)[i]);
                }
            }
          med.equivalentes = strdup($14);
        }
        ;

Name: ID
    ;

Category: ID
    ;

Repetable: ID
            ;

Code: INT
    ;

ChemComp: ID
        ;

Price: REAL
     ;
```

```
Repetables: OPEN_CURLY_BRACKET IdList CLOSE_CURLY_BRACKET {$$ = $2;}
            ;

EqualList: Equal                                        //Adiciona medicamento a lista de repetidos
               | EqualList COMMA Equal
               {
                    char * newEq = strdup($3);
                    char * others = strdup($1);
                    strcat(others, ", ");
                    strcat(others, newEq);
                    $$ = others;
               }
               ;

Equals: OPEN_CURLY_BRACKET EqualList CLOSE_CURLY_BRACKET { $$ = $2; }
          ;

Equal: Name BRANCOS Repetables       //Diz o numero de medicamentos repetidos
            {
                char * eq = strdup($1);
                strcat(eq, "-");
                strcat(eq, $3);
                $$ = eq;
            }
         ;

%%


void yyerror(char* erro){
   printf("erro: %d: %s: %s\n",yylineno,erro,yytext);
}


void printMedicamentos(){       //Dar Print aos Medicamentos
        int i,j;
        for( i = 0; i < nClasses; i++){
                for( j=0; j < n_fabricantes[i]; j++){
                        printf("Nome: %s\nCod: %d\nCat: %s\nComposicao: %s\nPreco: %.2f\nFabricante: %s\nEquivalentes: %s\n---------
                }
        }
```

```
int main (void) {
        int errado;

        if(errado = yyparse( )){        //ERROR
                return 0;
        }else {                                 //Succe
                printMedicamentos();
                return 0;
        }
}
```

# Capítulo 4

# Codificação e Testes

## 4.1 Alternativas, Decisões e Problemas de Implementação

No final fazendo a compilação do projeto não conseguimos implementar o mesmo pois dá nos dois erros no ficheiro c do lex devido a passagens de informações entre lex e yacc.

Estes erros no ficheiro c do yacc acontecem muitas das vezes devido à confusão de variáveis dentro do projeto, uma vez que estão colocadas em sítios errados ou então uma variável chamada nTrabalho estar no lugar de uma n trabalho.

Estes erros no ficheiro c do yacc podem ter surgido devido ao facto de já existir "warnings" na compilação do yacc.

Apesar de tudo também fizemos um exemplo de entrada de medicamentos, mas como é obvio não foi possível utilizar. Pensámos também que o facto de existirem estes erros derivam bastante de nós termos deixado muito para última a realização do trabalho apesar de que a parte que nos revelou mais trabalhosa e com mais erros ter sido lecionada apenas ainda neste mês.

## 4.2 Testes realizados e Resultados

Como foi referido anteriormente, devido aos erros de implementação não foi possivel compilar os ficheiros.Sendo assim, não conseguimos obter os resultados.

```
     |                                ~~~~~~~~~~~~~~~~~~~^~~
     |                                                   |
     |                                                   char
In file included from med.y:4:
/usr/include/string.h:137:32: note: expected ‘const char *’ but argument is of type ‘char’
  137 | extern int strcmp (const char *__s1, const char *__s2)
      |                    ~~~~~~~~~~~~~~~~^~~~
med.y:109:21: warning: implicit declaration of function ‘addString’ [-Wimplicit-function-declaration]
  109 |                     addString(&Fabricantes, &n_Fabricantes, (med.fabricantes)[i]);
      |                     ^~~~~~~~~~
med.y:112:28: error: assignment to expression with array type
  112 |           med.equivalentes = strdup($14);
      |                            ^
med.y: In function ‘yyerror’:
med.y:167:46: error: ‘yytext’ undeclared (first use in this function); did you mean ‘yylex’?
  167 |     printf("erro: %d: %s: %s\n",yylineno,erro,yytext);
      |                                               ^~~~~~~
      |                                               yylex
med.y: In function ‘printMedicamentos’:
med.y:174:17: error: ‘n_fabricantes’ undeclared (first use in this function); did you mean ‘n_Fabricantes’?
  174 |   for( j=0; j < n_fabricantes[i]; j++){
      |                 ^~~~~~~~~~~~~~~
      |                 n_Fabricantes
med.y:175:231: error: ‘array’ undeclared (first use in this function)
  175 |  Fabricantes[i][j].nome, Fabricantes[i][j].codigo, Fabricantes[i][j].classe, Fabricantes[i][j].comp, array[i][j]
.preco, array[i][j].fabricante, array[i][j].equivalentes);
      |                                                                                                       ^~~~~
```

```
medicamentos.l: In function ‘yylex’:
medicamentos.l:9:8: error: ‘YYSTYPE’ {aka ‘union YYSTYPE’} has no member named ‘num’
    9 | [1-9]+[0-9]*        {yylval.num = atoi(yytext); return INT; }
      |        ^
medicamentos.l:10:8: error: ‘YYSTYPE’ {aka ‘union YYSTYPE’} has no member named ‘rnum’
   10 | [0-9]+[.][0-9]*        {yylval.rnum = atof(yytext); return REAL; }
      |        ^
```

```
Symposium: 2022
[Analgesico,Antibiotico,Brufen]
[
/- Medicamento 1 -/
(Moment,1,Analgesico,Paracetamol,4.5,{Roche},{Qualquer-Bial});
/- Medicamento 2 -/
(Antigripine,2,Antibiotico,Acetalilico,6.,{Fabt,Faba,Fabb},{Aga-Fabc,Agb-Fabh});
/- Medicamento 3 -/
(Brufen,3,Analgesico,Ibuprofeno,400,{Mylan},{Qualquer-detestoCompiladores});
/- Medicamento 4 -/
(Spidifen,20,Analgesico,Ibuprofeno,400,{Zambom},{Qualquer-duyy});
```

# Capítulo 5

# Conclusão

Neste projeto começamos por analisar o problema a que fomos propostos e após uma breve reflexão, dividimos e percebemos os objetivos. Desenvolvemos dois codigos importantes que nos permitiram adquirir um pouco mais de conhecimento acerca do lex e do yacc. . Podemos concluir que o projeto foi de certo modo importante para esta Unidade Curricular.

# Apêndice A

# Equipa

Esta equipa é composta pelos elementos Adriana Paiva, Andreia Queirós, Bernardo Meneses e Tiago Fernandes.

Este trabalho foi feito através da plataforma Discord. De forma a homogeneizar o trabalho e de maneira a todos os elementos entenderem o mesmo da matéria presente no trabalho, em grupo, foi decidido que todas as etapas seriam feitas por todos com partilha de ideias.