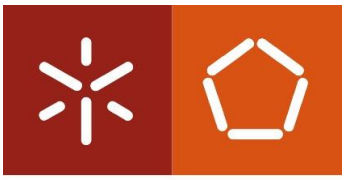


Universidade do Minho

Departamento de Informática

Processamento de Linguagens
Trabalho Prático nº 2

Gonçalo Almeida (A84610)

Emanuel Rodrigues (A84776)

Lázaro Pinheiro (A86788)

28 de Junho de 2020

Conteúdo

1 Introdução **3**

2 Objetivos do trabalho prático **4**

3 Notas iniciais **5**

4 Implementação **6**
4.1 Flex 6
4.1.1 Filtros de Texto Flex 6
4.2 Yacc 7
4.2.1 Gramática 7
4.2.2 HTML 9
4.2.3 Estruturas de dados 9

5 Resultado Final **11**

6 Conclusão **13**

A Flex **14**

B Yacc **15**

Capítulo 1

Introdução

Este projeto foi realizado no âmbito da unidade curricular Processamento de Linguagens e tem como objetivo construir um processador (flex, yacc) que reconheça e valide cadernos de anotação gerando um site HTML. Para isto foi utilizada a ferramenta Flex de modo a processar o texto de entrada, o Yacc para escrever a gramática que define a DSL de caderno de anotações e gerar o site HTML e a biblioteca Glib para o armazenamento dos conteúdos.

Capítulo 2

Objetivos do trabalho prático

- Aumentar a experiência de uso do ambiente Linux , da linguagem imperativa C (para codificação das estruturas de dados e respetivos algoritmos de manipulação), e de algumas ferramentas de apoio à programação;
- Rever e aumentar a capacidade de escrever gramáticas independentes de contexto (GIC) , que satisfaçam a condição LR(), para criar Linguagens de Domínio Específico (DSL);
- Desenvolver processadores de linguagens segundo o método da gramática tradutora (GT) ;
- utilizar geradores de compiladores como o par flex/yacc.

Capítulo 3

Notas iniciais

Dado que o nosso grupo é o 67, usando a formula que se encontra no enunciado ($exe = (NGr\%6) + 1$), foi nos atribuído o trabalho 2, DSL para Cadernos de Anotações em HD.

Capítulo 4

Implementação

4.1 Flex

Com a ajuda da biblioteca Glib definimos uma string para armazenar o conteúdo processado de um documento:

```
GString* conteudo = g_string_new(NULL);
```

4.1.1 Filtros de Texto Flex

De modo a processar o ficheiro de entrada desenvolvemos um ficheiro Flex que faz o parse de todos os campos pedidos para o Yacc.

Conceito

Para filtrar o conceito do documento decidimos capturar o texto que vem a seguir a ===\ \: até encontrar um \n.

Título

Para capturarmos o título do documento, guardamos todo o texto que aparece a seguir a @tit\:\ até a um \n.

Subtítulo

O subtítulo é obtido através da expressão <TEXTO>#\ .+[\n]2

Conteúdo

Para se filtrar o conteúdo usamos a expressão <CONT>.+/\n que vai dar append no array conteúdo do texto encontrado, quando é encontrado um \n no inicio da linha ou um novo subtítulo, o conteúdo e passado ao yacc e o array e apagado.

Triplos

Quando se encontra um @triplos\: vamos entrar na condição de contexto do sujeito do triplo, dentro dela guardamos o sujeito através de <SUJ>\:.+/\n e de seguida entrámos na condição de contexto da relação, nela retiramos o texto da relação e entramos na condição do objeto ou então vamos para a condição da imagem se se encontrar um \:img/\, guardamos o nome da imagem

e voltamos para a relação se for encontrado um ; ou para o sujeito para um \. . No objeto procedemos de maneira semelhante a imagem.

4.2 Yacc

4.2.1 Gramática

A abordagem tomada para o desenvolvimento da gramática que melhor representa lingua- gem a reconhecer foi uma top-down.

Cada especificação de um caderno consiste numa lista de símbolos não terminais 'par'.

```
caderno : caderno par
        |
        ;
```

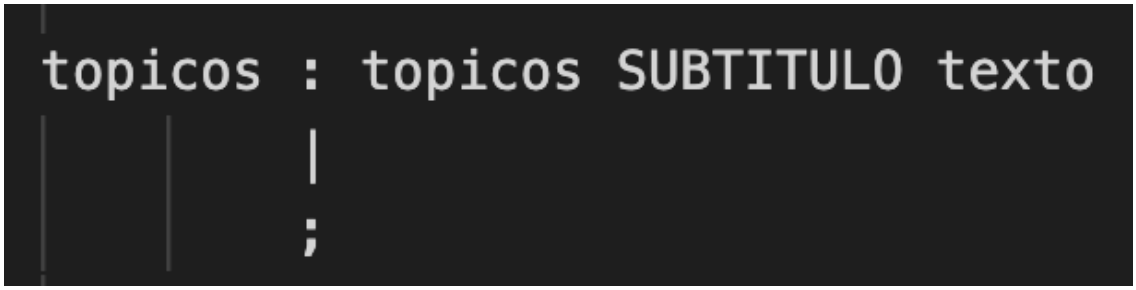
Cada especificação de um 'par' encontra-se essencialmente dividida em duas partes, uma em que se encontra o 'documento' (documento) e outra em que se encontra o 'triplos' (triplos do documento).

```
par : documento triplos
    ;
```

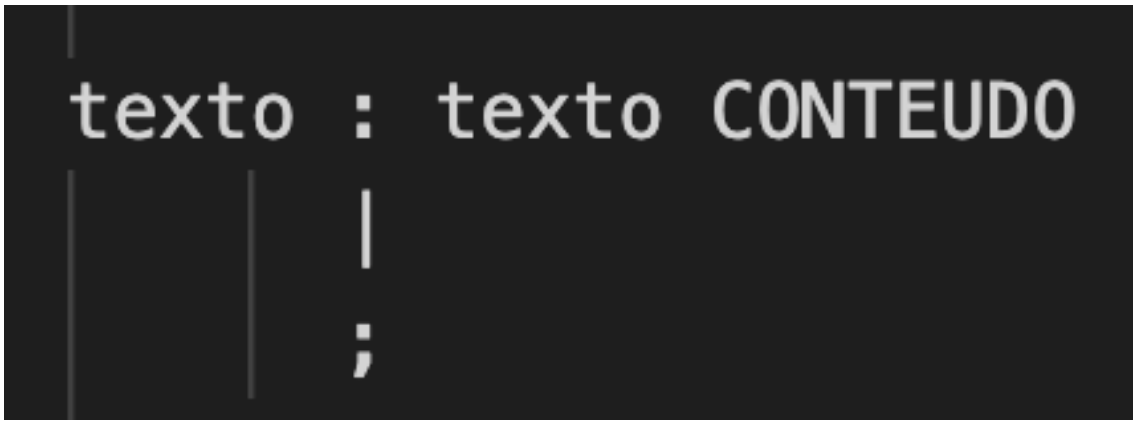
Cada especificação de um 'documento' deriva nos símbolos terminais 'CONCEITO' (respetivo ao conceito em causa) e 'TITULO' (respetivo ao título do conceito), e no símbolo não terminal 'topicos' onde se encontram os tópicos de cada documento.

```
documento : CONCEITO TITULO topicos
          ;
```

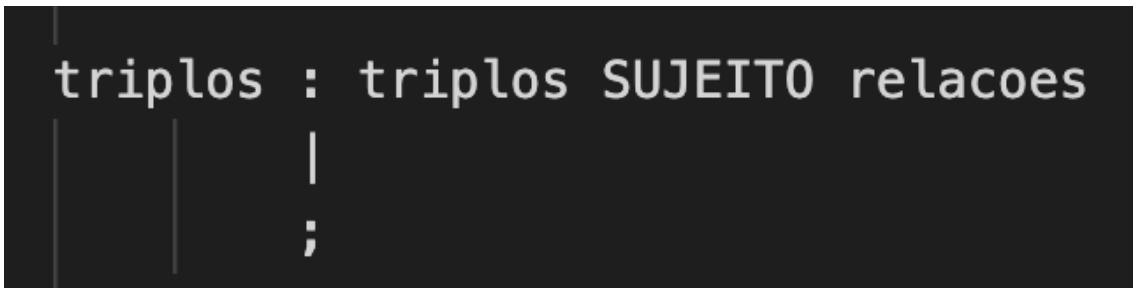
Cada especificação de 'topicos' consiste numa lista de símbolos terminais 'SUBTITULO' (representantes dos subtítulos de cada tópico) e de símbolos não terminais 'texto' (representantes do texto respetivo a cada subtítulo).



Cada especificação de 'texto' consiste numa lista de símbolos não terminais 'CONTEUDO' (representantes de cada parágrafo do respetivo texto).

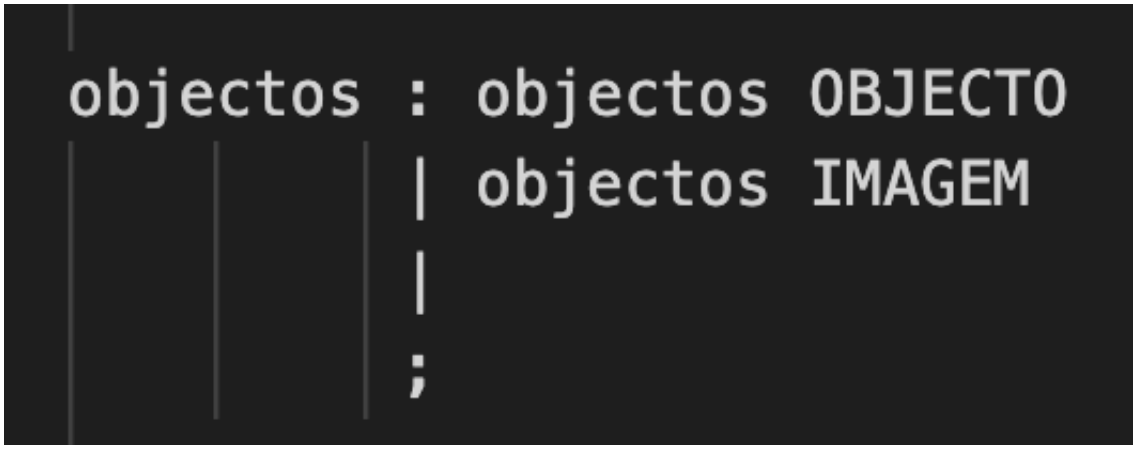


Cada especificação de 'triplos' consiste numa lista de símbolos terminais 'SUJEITO' (representantes dos sujeitos dos triplos) e de símbolos não terminais 'relacoes' (representantes das relações respetivas a cada sujeito).



Cada especificação de 'relacoes' consiste numa lista de símbolos terminais 'RELACAO' (representantes das relações dos sujeitos) e de símbolos não terminais 'objectos' (representantes das objetos respetivos a cada relação).

Cada especificação de 'objectos' consiste numa lista de símbolos terminais 'OBJECTO' (representantes de cada objeto da respetiva relação) ou de símbolos terminais 'IMAGEM' (representantes de cada imagem do respetivo sujeito).



4.2.2 HTML

Para uma melhor navegação nos ficheiros gerados, foi criado um índex (index.html) que contêm as referências para todos os conceitos encontrados no ficheiro de entrada.

O HTML gerado para cada sujeito/objeto apenas consiste no seu título, imagens e botão para regressar ao índex gerado.

O HTML gerado para cada conceito (sendo também sujeito/objeto ou não) consiste nos mesmos elementos referidos anteriormente mais todos os seus tópicos e a listagem de todos os seus triplos, estes contendo as referências para os sujeitos e objetos.

É de notar que as imagens são todas guardadas em memória e apenas adicionadas ao ficheiro no final do parse do ficheiro de entrada de modo a que estas não fiquem espalhadas, mas sim agrupadas numa classe com propriedades independentes.

4.2.3 Estruturas de dados

Como foi referido acima, as imagens são guardadas em memória. Para tal foram definidas algumas estruturas de dados. Algumas destas estruturas são referentes à biblioteca Glib que facilitou o seu manipulamento.

A estrutura 'Imagens' armazena, para cada sujeito, as respetivas imagens e o número destas.

```c
typedef struct imagens {
    char* sujeito;
    int numImgs;
    char* imgs[NUM_MAX_IMAGES];
} *Imagens;
```

A estrutura 'buff' é um GArray que armazena todas as estruturas 'Imagens' referidas anteriormente.

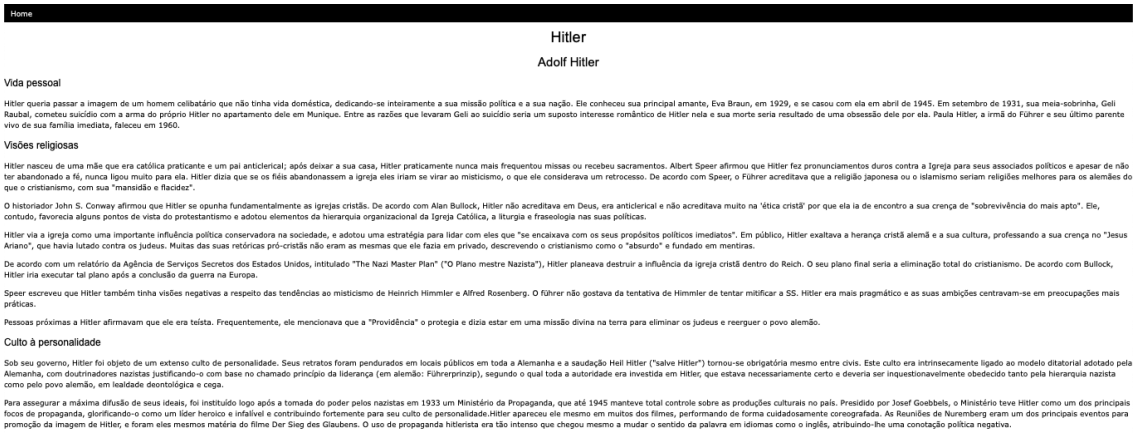
```c
GArray *buff;
```

A estrutura 'imgs' é um array que armazena as imagens do sujeito a ser tratado momentariamente. O inteiro 'imageCount' é referente ao número de imagens nesta estrutura.

```c
char* imgs[NUM_MAX_IMAGES];
int imageCount = 0;
```

Capítulo 5

Resultado Final



Home

Hitler

Adolf Hitler

Vida pessoal

Hitler queria passar a imagem de um homem celibatário que não tinha vida doméstica, dedicando-se inteiramente a sua missão política e a sua nação. Ele conheceu sua principal amante, Eva Braun, em 1929, e se casou com ela em abril de 1945. Em setembro de 1931, sua meia-sobrinha, Geli Raubal, cometeu suicídio com a arma do próprio Hitler no apartamento dele em Munique. Entre as razões que levaram Geli ao suicídio seria um suposto interesse romântico de Hitler nela e sua morte seria resultado de uma obsessão dele por ela. Paula Hitler, a irmã do Führer e seu último parente vivo de sua família imediata, faleceu em 1960.

Visões religiosas

Hitler nasceu de uma mãe que era católica praticante e um pai anticlerical; após deixar a sua casa, Hitler praticamente nunca mais frequentou missas ou recebeu sacramentos. Albert Speer afirmou que Hitler fez pronunciamentos duros contra a Igreja para seus associados políticos e apesar de não ter abandonado a fé, nunca ligou muito para ela. Hitler dizia que se os fiéis abandonassem a igreja eles iriam se virar ao misticismo, o que ele considerava um retrocesso. De acordo com Speer, o Führer acreditava que a religião japonesa ou o islamismo seriam religiões melhores para os alemães do que o cristianismo, com sua "mansidão e flacidez".

O historiador John S. Conway afirmou que Hitler se opunha fundamentalmente as igrejas cristãs. De acordo com Alan Bullock, Hitler não acreditava em Deus, era anticlerical e não acreditava muito na 'ética cristã' por que ela ia de encontro a sua crença de "sobrevivência do mais apto". Ele, contudo, favorecia alguns pontos de vista do protestantismo e adotou elementos da hierarquia organizacional da Igreja Católica, a liturgia e fraseologia nas suas políticas.

Hitler via a igreja como uma importante influência política conservadora na sociedade, e adotou uma estratégia para lidar com eles que "se encaixava com os seus propósitos políticos imediatos". Em público, Hitler exaltava a herança cristã alemã e a sua cultura, professando a sua crença no "Jesus Ariano", que havia lutado contra os judeus. Muitas das suas retóricas pró-cristãs não eram as mesmas que ele fazia em privado, descrevendo o cristianismo como o "absurdo" e fundado em mentiras.

De acordo com um relatório da Agência de Serviços Secretos dos Estados Unidos, intitulado "The Nazi Master Plan" ("O Plano mestre Nazista"), Hitler planeava destruir a influência da igreja cristã dentro do Reich. O seu plano final seria a eliminação total do cristianismo. De acordo com Bullock, Hitler iria executar tal plano após a conclusão da guerra na Europa.

Speer escreveu que Hitler também tinha visões negativas a respeito das tendências ao misticismo de Heinrich Himmler e Alfred Rosenberg. O führer não gostava da tentativa de Himmler de tentar mitificar a SS. Hitler era mais pragmático e as suas ambições centravam-se em preocupações mais práticas.

Pessoas próximas a Hitler afirmavam que ele era teísta. Frequentemente, ele mencionava que a "Providência" o protegia e dizia estar em uma missão divina na terra para eliminar os judeus e reerguer o povo alemão.

Culto à personalidade

Sob seu governo, Hitler foi objeto de um extenso culto de personalidade. Seus retratos foram pendurados em locais públicos em toda a Alemanha e a saudação Heil Hitler ("salve Hitler") tornou-se obrigatória mesmo entre civis. Este culto era intrinsecamente ligado ao modelo ditatorial adotado pela Alemanha, com doutrinadores nazistas justificando-o com base no chamado princípio da liderança (em alemão: Führerprinzip), segundo o qual toda a autoridade era investida em Hitler, que estava necessariamente certo e deveria ser inquestionavelmente obedecido tanto pela hierarquia nazista como pelo povo alemão, em lealdade deontológica e cega.

Para assegurar a máxima difusão de seus ideais, foi instituído logo após a tomada do poder pelos nazistas em 1933 um Ministério da Propaganda, que até 1945 manteve total controle sobre as produções culturais no país. Presidido por Josef Goebbels, o Ministério teve Hitler como um dos principais focos de propaganda, glorificando-o como um líder heroico e infalível e contribuindo fortemente para seu culto de personalidade.Hitler apareceu ele mesmo em muitos dos filmes, performando de forma cuidadosamente coreografada. As Reuniões de Nuremberg eram um dos principais eventos para promoção da imagem de Hitler, e foram eles mesmos matéria do filme Der Sieg des Glaubens. O uso de propaganda hitlerista era tão intenso que chegou mesmo a mudar o sentido da palavra em idiomas como o inglês, atribuindo-lhe uma conotação política negativa.

Conceitos

- Salazar
- Hitler
- Jong-un

Triplos

- Hitler
  - a
    - Person
    - Ditador
    - Político
    - Führer
    - Militar
    - Escritor
    - Artista
  - Name
    - Adolf Hitler
  - birth_place
    - Braunau am Inn
  - nationality
    - Austríaco
    - Alemão
  - viveu_em
    - Berlim
    - Braunau am Inn
  - partidos_politicos
    - Partido Alemão dos Trabalhadores
    - Partido Nazi
  - religião
    - Visão religiosa de Adolf Hitler
- EvaBraun
  - casada_com
    - Hitler
    - name
    - Eva Braun
- KlaraHitler

Imagens

Capítulo 6

Conclusão

A realização deste trabalho prático permitiu compreender a grande utilidade da ferramenta Flex no tratamento de informação e o Yacc permite a especificação de uma hierarquia complexa de relacionamentos entre conceitos. Com recurso a esta é gerado um ficheiro html que resulta numa página web. Esta página web permite o link com outras páginas web , permitindo assim a consulta de conceitos, de triplos e também a visualização das imagens.

Numa primeira abordagem o grupo não percebeu o que estava a ser requerido no enunciado, contudo a sessão de dúvidas facilitou bastante este processo. Em suma, obtivemos uma melhor compreensão e prática relativamente às expressões regulares bem como consolidamos o conhecimento sobre gramáticas adquirido no decorrer das aulas da U.C. . É necessário frisar que com este trabalho aprofundamos o nosso conhecimento teórico e prático relativamente à linguagem de programação html, considerando assim que os requisitos propostos foram cumpridos.

Apêndice A

Flex

```
%{
    #include<glib.h>
    #include "y.tab.h"
    void yyerror(char* s);
%}

%option noyywrap yylineno

%x TEXTO CONT SUJ REL OBJ IMG

%%
                                        GString* conteudo = g_string_new(NULL);

===\ \:.+/\n                            { yylval.string = strdup(yytext+5); return CONCEITO; }

@tit\:\ .+/\n                           { BEGIN TEXTO;  yylval.string = strdup(yytext+6); return TITULO; }

<TEXTO>@triplos\:                       { BEGIN SUJ; }
<TEXTO>#\ .+[\n]{2}                     { BEGIN CONT; yytext[strlen(yytext)-2] = '\0'; yylval.string = strdup(yytext+2); return SUBTITULO; }
<TEXTO>\n                               ;

<CONT>.+/\n                             { g_string_append(conteudo, yytext); }
<CONT>^\n                               { yylval.string = strdup(conteudo->str); g_string_erase(conteudo, 0, -1); return CONTEUDO; }
<CONT>\n/(#|@)                          { BEGIN TEXTO; yylval.string = strdup(conteudo->str); g_string_erase(conteudo, 0, -1); return CONTEUDO; }
<CONT>\n                                { g_string_append(conteudo," "); }

<SUJ>\n/===\ \:                         { BEGIN 0; }
<SUJ>\:.+/\n                            { BEGIN REL; yylval.string = strdup(yytext+1); return SUJEITO; }
<SUJ>\n                                 ;

<REL>\:img/\                            { BEGIN IMG; }
<REL>a/\                                { BEGIN OBJ; yylval.string = strdup(yytext); return RELACAO; }
<REL>\:[^ ]*/\                          { BEGIN OBJ; yylval.string = strdup(yytext+1); return RELACAO; }
<REL>[\n;, ]                            ;

<OBJ>\.                                 { BEGIN SUJ; }
<OBJ>;                                  { BEGIN REL; }
<OBJ>\:[^ ,;.]*/(\ |,|;|\.)             { yylval.string = strdup(yytext+1); return OBJECTO; }
<OBJ>\".*/\"                            { yylval.string = strdup(yytext+1); return OBJECTO; }
<OBJ>[ ,\n\"]                           ;

<IMG>\.                                 { BEGIN SUJ; }
<IMG>;                                  { BEGIN REL; }
<IMG>\".*/\"                            { yylval.string = strdup(yytext+1); return IMAGEM; }
<IMG>[ ,\n\"]                           ;

.|\n                                    ;

%%
```

Apêndice B

Yacc

```
%{
    #define _GNU_SOURCE
    #define FALSE 0
    #define TRUE 1
    #define NUM_MAX_IMAGES 5

    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <math.h>
    #include <unistd.h>
    #include <fcntl.h>
    #include <glib.h>

    int yylex();
    void yyerror(char*);

    void fileGenerator(char*, int);
    void addText(char*, char*, char*);
    void generateTriplo (char*);
    void addTriplos(char*, char*);
    char* formatName (char*);
    void writeInIndexHtml(char*, char*);
    void addImages (char*);

    char* indexPath = "base/index.html";

    typedef struct imagens {
        char* sujeito;
        int numImgs;
        char* imgs[NUM_MAX_IMAGES];
    } *Imagens;

    GArray *buff;

    char* imgs[NUM_MAX_IMAGES];
    int imageCount = 0;
%}

%union{
    char* string;
}

%token CONCEITO TITULO SUBTITULO CONTEUDO SUJEITO RELACAO OBJECTO IMAGEM
%type <string> CONCEITO TITULO SUBTITULO CONTEUDO SUJEITO RELACAO OBJECTO IMAGEM topicos texto triplos relacoes objectos documento

%%

caderno : caderno par
        |
        ;

par : documento triplos                             { addTriplos($1, $2); }
    ;

documento : CONCEITO TITULO topicos                 { fileGenerator($1, TRUE); addText($1, $2, $3); }//asprintf(&$$, "%s", $1); }
          ;

topicos : topicos SUBTITULO texto                   { asprintf(&$$, "%s\t\t\t\t<h3>%s</h3>\n%s", $1, $2,$3); }
        |                                           { asprintf(&$$, "");}
        ;

texto : texto CONTEUDO                              { asprintf(&$$, "%s\t\t\t\t<p>%s</p>\n\n", $1, $2);}
      |                                             { asprintf(&$$, "");}
      ;

triplos : triplos SUJEITO relacoes                  { generateTriplo($2); asprintf(&$$, "%s\t\t\t<ul data-role=\"treeview\">\n\t\t\t\t<a href=\"../%s/%s.html\">\n\t\t\t\t\t<h4>%s</h4>\n\t\t\t
        |                                           { asprintf(&$$, ""); }
        ;

relacoes : relacoes RELACAO objectos                { asprintf(&$$, "%s\n\t\t\t\t<ul data-role=\"treeview\">\n\t\t\t\t\t<h5>%s</h5>\n\t\t\t\t\t<ul>\n%s\t\t\t\t</ul>\n\t\t\t\t</ul>
         |                                          { asprintf(&$$, ""); }
         ;

objectos : objectos OBJECTO                         { generateTriplo($2); asprintf(&$$, "%s\t\t\t\t\t<a href=\"../%s/%s.html\">\n\t\t\t\t\t<p>%s</p>\n\t\t\t\t</a>\n", $1, form
         | objectos IMAGEM                          { asprintf(&$$, "%s", $1); asprintf(imgs+imageCount, "%s", $2); imageCount++; }
         |                                          { asprintf(&$$, ""); }
         ;
```

```
%%

void fileGenerator(char* name, int isConcept){
    if(name){
         char *command, *path;
         char* nameFormated = formatName(name);
        asprintf(&path,"base/%s/%s.html", nameFormated, nameFormated);

        if (access(path, F_OK) == -1){
            asprintf(&command,"cd base;mkdir %s;cd %s;touch %s.html", nameFormated, nameFormated, nameFormated);
            system(command);
            free(command);

            FILE* file = fopen(path,"w");
            fprintf(file, "<!DOCTYPE html>\n\t<html lang=\"pt-pt\">\n\t<head>\n\t\t<title>%s</title>\n\t\t<meta charset=\"utf-8\">\n\t\t<meta name=\"viewport\" content=\"width=device-width, i
            fclose(file);
        }

        if(isConcept){
            writeInIndexHtml(path+5, name);
        }

        free(path);
    }
}

void addText(char* conceito, char* subtitulo, char* texto){
    if(conceito && subtitulo && texto){
        char* path;
        asprintf(&path,"base/%s/%s.html",conceito, conceito);
        FILE* file = fopen(path,"a");
        fprintf(file, "\n\t\t<div class=\"documento\">\n\t\t\t<div class=\"cabecalho\">\n\t\t\t\t<h2>%s</h2>\n\t\t\t</div>\n\t\t\t<div class=\"topicos\">\n%s\t\t\t</div>\n\t\t</div>\n", subti
        fclose(file);
        free(path);
    }
}

// Escreve os triplos
void addTriplos (char* conceito, char* texto){
    if(conceito && texto){
        char* path;
        asprintf(&path,"base/%s/%s.html",conceito, conceito);
        FILE* file = fopen(path,"a");
        fprintf(file, "\t\t<div class=\"triplos\">\n\t\t\t<div class=\"cabecalho\">\n\t\t\t\t<h2>Triplos</h2>\n\t\t\t</div>\n%s\t\t</div>\n", texto);
        fclose(file);
        free(path);
    }
}

void generateTriplo (char* name) {

    if (name[strlen(name)-1] == ' ') name[strlen(name)-1] = '\0';

    char* nameFormated = formatName(name);

    char* path;
    asprintf(&path, "base/%s", nameFormated);

    fileGenerator(name, FALSE);

    free(path);
}

char* formatName (char* name) {
    char* nameFormated;
    asprintf(&nameFormated, "%s", name);

    for (int i = 0; nameFormated[i] != '\0'; i++){
        if (nameFormated[i] == ' ') nameFormated[i] = '_';
    }

    return nameFormated;
}

void addImages (char* sujeito){

    if (sujeito[strlen(sujeito)-1] == ' ') sujeito[strlen(sujeito)-1] = '\0';

    if(imageCount > 0){
        int found = 0;
        Imagens imagens;

        for(int i = 0 ; i < buff->len && !found ; i++){
            imagens = g_array_index(buff, Imagens, i);
            if(strcmp(imagens->sujeito, sujeito) == 0){
                found = 1;
                for (int i = imagens->numImgs; i < imagens->numImgs + imageCount && i < NUM_MAX_IMAGES; i++){
                    asprintf(imagens->imgs+i, "\t\t\t<img class=\"imagem\" src=\"%s\" width=\"100\" height=\"100\">", imgs[i]);
                }
                imagens->numImgs += imageCount;
            }
        }

        if(!found){
            imagens = (Imagens)malloc(sizeof(struct imagens));
            asprintf(&imagens->sujeito, "%s", sujeito);
            imagens->numImgs = imageCount;
            for (int i = 0; i < imageCount; i++){
                asprintf(imagens->imgs+i, "\t\t\t<img class=\"imagem\" src=\"%s\" width=\"100\" height=\"100\">", imgs[i]);
            }

            g_array_append_val(buff, imagens);
        }

        imageCount = 0;
    }
```

```
}

void finalizeFiles(){
    char *command;
    asprintf(&command, "cd base; ls -d */ | sed 's/.$//' > output ");
    system(command);
    free(command);

    FILE* f = fopen("base/output", "r");
    char directory[128];
    char* path;
    char* name;
    Imagens imagens;

    if(!f){
        perror("Ocorreu um erro!\n");
    }else{
        while (fgets(directory, 128, f)){
            name = strtok (directory, "\n");
            asprintf(&path, "base/%s/%s.html", name, name);
            FILE* fp = fopen(path, "a");

            int flag = 0;
            for(int i = 0 ; i < buff->len ; i++){
                imagens = g_array_index(buff, Imagens, i);
                if(strcmp(imagens->sujeito, name) == 0){
                    if(flag == 0){
                        fprintf(fp,"\t\t<div class=\"imagens\">\n\t\t\t<div class=\"cabecalho\">\n\t\t\t\t<h2>Imagens</h2>\n\t\t\t</div>\n");
                        flag = 1;
                    }
                    for(int j = 0; j < imagens->numImgs ; j++ ){
                        fprintf(fp,"%s\n", imagens->imgs[j] );
                        free(imagens->imgs[j]);
                    }
                    free(imagens->sujeito);
                }
                flag = 0;
            }
            fprintf(fp, "\n\t</body>\n</html>");
            fclose(fp);
            free(path);
        }
        fclose(f);

        system("cd base; rm -f output");
    }
}


/*
 *  actions in index.html
 */

void generateIndexHtml(){
    system("mkdir base ; cd base ; touch index.html");
    FILE* file = fopen(indexPath,"w");
    fprintf(file, "<!DOCTYPE html>\n\t<html lang=\"pt-pt\">\n\t<head>\n\t\t<title>index</title>\n\t\t<meta charset=\"utf-8\">\n\t\t<style>\n\t\t\tdiv.cabecalho {\n\t\t\t\ttext-align: center;\
    fclose(file);
}

void writeInIndexHtml(char* path, char* name){

    if(path && name){

        char *command;
        asprintf(&command, "cd base;grep -w -c \"%s\" index.html > output", name);
        system(command);
        free(command);

        FILE* f = fopen("base/output", "r");
        char buffer[2];

        fgets(buffer, sizeof(buffer), f);

        if(buffer[0] == '0'){
            FILE* file = fopen(indexPath,"a");
            fprintf(file, "\t\t\t<a href='%s'><li>%s</li></a>\n", path, name);
            fclose(file);
        }

        system("cd base; rm -f output");
    }
}

void finalizeIndexHtml(){
    FILE* file = fopen(indexPath,"a");
    fprintf(file, "\t\t</div>\n\t</body>\n</html>");
    fclose(file);
}


int main(int argc, char* argv[]){
    extern FILE *yyin;
    if(argc > 1){ yyin = fopen(argv[1],"r"); }else{perror("Argumentos Insuficientes!");}
    buff = g_array_new(FALSE, FALSE, sizeof(Imagens));
    generateIndexHtml();
    yyparse();
    finalizeIndexHtml();
    finalizeFiles();
    g_array_free (buff, TRUE);
return 0;
}

void yyerror(char* s){
    extern int yylineno;
    extern char* yytext;
    fprintf(stderr,"linha %d: %s (%s)\n",yylineno,s, yytext);
```

}