%{

/\* includes, C defs \*/

#include <stdio.h>

extern int yylineno;

void yyerror(const char\* msg);

int yylex();

%}

Em relação ao trecho acima a variável yylineno permite reportar a linha de um determinado erro sintático.

A função yyerror permite reportar o erro sintático eventualmente ocorrido (inclusive a linha, usando yylineno).

/\* declare tokens \*/

%token ID

%token INT

%token VOID

%token CONST

%token NUM

%token RETURN

/\*%token return\*/

%precedence RETURN

%precedence '='

%left '+' '-'

%left '\*' '/'

%right '('

No trecho acima são declarados alguns dos tokens que serão usados pelo analisador léxico (serão incluídos nele por meio do arquivo “e1.tab.h” – criado após do comando “bison -d e1.y).

Também são definidas regras de precedência para alguns tokens.

Um pouco depois são reproduzidas as regras sintáticas definidas pelo exercício, algumas delas reproduzidas abaixo:

%%

program: declaration-list compound-stmt

;

declaration-list: declaration-list declaration |

declaration

;

declaration: var-declaration |

const-declaration

;

Por fim o analisador sintático tem uma função “main” (reproduzida abaixo) que que chama o analisador sintático (**yyparse()** e guarda na variável **result** se o análise foi realizada com sucesso (retorno 0) ou não (retorno diferente de zero):

int main(void){

//extern int yydebug;

//yydebug = 1;

int result;

result = yyparse();

if(result == 0)

printf("Sucesso na realização da análise sintática.\n Retorno de yyparse() vale: %d\n", result);

else{

printf("Insucesso na realização da análise sintática.\n Retorno de yyparse() vale: %d\n", result);

}

return 0;

}