```
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
#define YYDEBUG 1

#define TIP_INT 1
#define TIP_REAL 2
#define TIP_CAR 3

double stiva[20];
int sp;

void push(double x)
{ stiva[sp++]=x; }

double pop()
{ return stiva[--sp]; }

%}

%union {
	int l_val;
	char *p_val;
}

%token identifier
%token number_const
%token string_const
```

```
%token INT

%token STRING

%token IF

%token THEN

%token ELSE

%token WHILE

%token EXECUTE

%token ARRAY

%token READ

%token WRITE

%token coma

%token semicolon

%token leftSquareBracket

%token rightSquareBracket

%token leftRoundBracket

%token rightRoundBracket

%token startCurlyBracket

%token endCurlyBracket

%token plus

%token minus

%token multiply

%token division

%token equal

%token lessThan

%token lessThanOrEqual

%token equalTo

%token greaterThan

%token greaterThanOrEqual

%token different
```

```
%token modulo

%token CHAR


%start program


%%

program: startCurlyBracket declist cmpdstmt endCurlyBracket ;

declist: declaration | declaration declist ;

declaration: type identifier semicolon ;

vartype: INT | STRING | CHAR ;

arraydecl: ARRAY leftSquareBracket vartype rightSquareBracket leftRoundBracket number_const
rightRoundBracket ;

type: vartype | arraydecl ;

cmpdstmt: startCurlyBracket stmtlist endCurlyBracket ;

stmtlist: stmt | stmt stmtlist ;

stmt: simplestmt semicolon | structstmt ;

simplestmt: assignstmt | iostmt ;

assignstmt: identifier equal expression ;

expression: expression plus term | expression minus term | term ;

term: term multiply factor | term division factor | term modulo factor | factor ;

factor: leftRoundBracket expression rightRoundBracket | list ;

list: identifier | number_const | string_const;

iostmt: READ leftRoundBracket identifier rightRoundBracket | WRITE leftRoundBracket list
rightRoundBracket ;

structstmt: cmpdstmt | ifstmt | whilestmt ;

ifstmt: IF leftRoundBracket condition rightRoundBracket THEN stmt ELSE stmt | IF leftRoundBracket
condition rightRoundBracket THEN stmt ;

elsestmt: IF leftRoundBracket condition rightRoundBracket THEN stmt ELSE stmt ;

whilestmt: WHILE leftRoundBracket condition rightRoundBracket EXECUTE stmt ;

condition: expression relation expression ;
```

```
relation: lessThan | lessThanOrEqual | equalTo | greaterThan | greaterThanOrEqual | different ;
%%

yyerror(char *s)
{
  printf("%s\n", s);
}

extern FILE *yyin;

main(int argc, char **argv)
{
  if (argc > 1)
    yyin = fopen(argv[1], "r");
  if ( (argc > 2) && ( !strcmp(argv[2], "-d") ) )
    yydebug = 1;
  if ( !yyparse() )
    fprintf(stderr,"\t No error\n");
}
```