```
%{
#include <stdio.h>
#include <stdlib.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 FUNCTION
%token GO
%token FROM
%token TO
%token LENGTH
%token CHECK
%token OTHERWISE
%token RETURN
%token WRITE
%token READ

%token ID
%token <p_val> CONST_INT

%token INT
%token LIST
%token CHAR
%token STRING
%token BOOLEAN
```

```
%left '+' '-'
%left DIV MOD '*' '/' '%'
%left OR
%left AND

%%
type:           INT
                | CHAR
                | STRING
                | BOOLEAN
                ;

relation:       '<'
                | '>'
                | '<='
                | '>='
                | '=='
                | '<>'
                ;
input:          READ ID';'
                ;

output:         WRITE ID';'
                ;

declaration:    type ID';'
                ;

list_decl:      LIST'['type']' ID';'
                ;

return_decl:    RETURN ID
                | const_int
                ;

assign:         ID '=' ID
                | ID '=' const_int
                ;

const_int:      CONST_INT   {
                        $$ = TIP_INT;
                        push(atof($1));
                                }
```

```
if_stmt: CHECK cond':' stmt';' OTHERWISE':' stmt';'

cond:       logic_expr
            | expression relation expression
            ;
expression: expresie '+' expresie
            | expresie '-' expresie
            | expresie '*' expresie
            | expresie '/' expresie
            | expresie '%' expresie
            | expresie DIV expresie
            | expresie MOD expresie
            ;
logic_expr: logic_expr AND logic_expr
            | logic_expr OR logic_expr
            ;

stmt:       assign
            | input
            | output
            | return_decl
            ;

loop:       GO FROM part_a TO part_b':' stmt';'
part_a:     type assign
            | ID
            ;
part_b:     ID
            | const_int
            ;

func:       FUNCTION ID '('params'):' part_c
part_c:     declaration
            | list_decl
            | input
            | output
            | assign
            | if_stmt
            | loop
            | return_decl
            ;
params:             declaraction
            | list_decl
```

```
                    | ',' params
                    ;

prog:               func
                    ;

%%

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,"\tO.K.\n");
}
```