Lab 9 – yacc

```
%{
#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;
        bool b_val;
}

%token defvar
%token deflist
%token if
%token else
%token loop
%token input
%token print

%token id
%token <p_val> const_int
%token <p_val> const_double
%token <p_val> const_char
%token const_str
%token <b_val> const_bool

%token int
%token char
%token str
%token bool
%token double

%left and
%left or
%left not
```

```
%left '+'
%left '-'
%left '*'
%left '/'
%left '%'
%left "=="
%left '='
%left '<'
%left '>'
%left "<="
%left ">="
%left "+="
%left "-="
%left "++"
%left "--"
%left "**"

%type <l_val> expr_stat factor_stat constanta
%%

math_operator: '+' | '-' | '*' | '/' | '%' | "**" | "<=" | ">=" | "+=" | "-=" | "++" | "--" | "**" ;
relational_operator: "==" | "!=" | '<' | "<=" | '>' | ">=" ;
boolean_operator: and | or | not ;
type: int | str | char | double | bool ;
list: deflist identifier "[]" ':' type ';' ;
variable: defvar identifier [':' type] '{' ',' identifier [':' type] '}' ';' ;
number: int | double ;
mathematical_expression: number '{' math_operator number '}'
relational_operand: identifier | int | double | mathematical_expression
relational_expression: relational_operand relational_operator relational_operand
mathematical_or_relational_expression: mathematical_expression
                                                                | relational_expression
                                                                ;
expression: (mathematical_expression|relational_expression) '{' boolean_operator expression '}' ;
condition: expression relation expression ;
assignment: identifier '=' expression ';' ;
identifier_or_type: id | type
input_output_statement: input '(' identifier_or_type '{' ',' identifier_or_type '}' ')' ';'
                                    | print '(' identifier_or_type '{' ',' identifier_or_type '}' ')' ';' ;

simple_statement: assignment | input_output_statement ;
compound_statement: simple_statement '{' ';' compound_statement '}' ;
statement: compound_statement | if_statement | loop_statement ;
if_statement: if condition '{' statement '}' [ else '{' statement '}' ] ;
loop_statement: loop expression { '{' statement '}' } ;

%%


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");
}
```