Github Link: https://github.com/dckrck/flcd_lab9

Spec.y

```
%{
#include <stdio.h>
#include <stdlib.h>

#define YYDEBUG 1
%}

%token plus
%token minus
%token multiplication
%token division
%token modulo
%token lessThan
%token lessThanOrEqual
%token equal
%token moreThan
%token moreThanOrEqual
%token doubleEqual
%token notEqual
%token increment
%token decrement

%token leftBracket
%token rightBracket
%token leftCurlyBracket
%token rightCurlyBracket
%token leftRoundBracket
%token rightRoundBracket
%token colon
%token semicolon
%token comma
%token apostrophe
%token quote

%token IF
%token ELSE
%token READ
%token WRITE
%token VAR
%token WHILE
%token FOR
%token BREAK
%token RETURN
%token NOT
%token IN
%token CONTINUE
%token ARRAY
```

```
%token TRUE
%token FALSE
%token AND
%token OR

%token IDENTIFIER
%token INT_CONST
%token STRING_CONST
%token CHAR_CONST
%token POSITIVE_NUMBER

%start program

%%

program : comp_stmt

stmt : var_stmt semicolon | list_stmt semicolon | assign_stmt semicolon | return_stmt semicolon |
increment_decrement_stmt semicolon | read_stmt semicolon | write_stmt semicolon | if_stmt |
while_stmt
stmt_list : stmt | stmt stmt_list
comp_stmt : stmt_list

int_operator : plus | minus | multiplication | division | modulo
int_increment_decrement : increment | decrement
int_type : INT_CONST | IDENTIFIER
int_exp : int_type | int_type int_operator int_type

string_type : quote STRING_CONST quote | IDENTIFIER
string_exp : string_type | string_type plus string_type

bool_type : TRUE | FALSE
bool_exp : bool_type | bool_type plus bool_type

exp : int_exp | string_exp | bool_exp | IDENTIFIER

increment_decrement_stmt : IDENTIFIER int_increment_decrement
array_stmt : ARRAY VAR IDENTIFIER leftBracket rightBracket
var_stmt : VAR IDENTIFIER | array_stmt | VAR IDENTIFIER equal exp | VAR IDENTIFIER
equal IDENTIFIER | VAR IDENTIFIER equal read_stmt
identifier_list : comma IDENTIFIER | comma IDENTIFIER identifier_list
list_stmt : VAR IDENTIFIER identifier_list
assign_stmt : IDENTIFIER equal exp | IDENTIFIER equal read_stmt | IDENTIFIER equal
IDENTIFIER
if_stmt : IF leftRoundBracket comp_cond rightRoundBracket leftCurlyBracket comp_stmt
rightCurlyBracket | IF leftRoundBracket comp_cond rightRoundBracket leftCurlyBracket
comp_stmt rightCurlyBracket ELSE leftCurlyBracket comp_stmt rightCurlyBracket
while_stmt : WHILE leftRoundBracket comp_cond rightRoundBracket leftCurlyBracket
comp_stmt rightCurlyBracket
return_stmt : RETURN exp | RETURN
read_stmt : READ leftRoundBracket rightRoundBracket
```

write_stmt : WRITE leftRoundBracket IDENTIFIER rightRoundBracket | WRITE leftRoundBracket exp rightRoundBracket

relation : lessThan | lessThanOrEqual | doubleEqual | notEqual | moreThanOrEqual | moreThan
logical : AND | OR
comp_cond: cond | cond logical comp_cond
cond : exp relation exp

%%

```
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");
}
```

---

commands:

```
lex specif.lxi
yacc -d parser.y
gcc lex.yy.c y.tab.c -o result -lfl
```

input p1.txt:

```
var first, second, third;

first = read();
second = read();
third = read();

var max_num = first;

if(second > max_num)
{
	max_num = second;
}

if(third > max_num)
{
	max_num = third;
}
```

```
write(max_num);
```

output: ./result p1.txt

```
KEYWORD: var
IDENTIFIER: first
SEPARATOR ,
IDENTIFIER: second
SEPARATOR ,
IDENTIFIER: third
SEPARATOR ;
IDENTIFIER: first
OPERATOR: =
KEYWORD: read
SEPARATOR (
SEPARATOR )
SEPARATOR ;
IDENTIFIER: second
OPERATOR: =
KEYWORD: read
SEPARATOR (
SEPARATOR )
SEPARATOR ;
IDENTIFIER: third
OPERATOR: =
KEYWORD: read
SEPARATOR (
SEPARATOR )
SEPARATOR ;
KEYWORD: var
IDENTIFIER: max_num
OPERATOR: =
IDENTIFIER: first
SEPARATOR ;
KEYWORD: if
SEPARATOR (
IDENTIFIER: second
OPERATOR: >
IDENTIFIER: max_num
SEPARATOR )
SEPARATOR {
IDENTIFIER: max_num
OPERATOR: =
IDENTIFIER: second
SEPARATOR ;
SEPARATOR }
KEYWORD: if
SEPARATOR (
IDENTIFIER: third
OPERATOR: >
IDENTIFIER: max_num
```

SEPARATOR)
SEPARATOR {
IDENTIFIER: max_num
OPERATOR: =
IDENTIFIER: third
SEPARATOR ;
SEPARATOR }
KEYWORD: write
SEPARATOR (
IDENTIFIER: max_num
SEPARATOR)
SEPARATOR ;
O.K.