Github: https://github.com/Alin-St/FLCD

This is the content of my lang.y file (specifications for yacc program):

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

#define YYDEBUG 1

int yyerror(const char *s);
%}

%token INT;
%token BOOL;
%token INTLIST;
%token IF;
%token WHILE;
%token READ;
%token WRITE;

%token ASSIGN;
%token PLUS;
%token MINUS;
%token MUL;
%token DIV;
%token MOD;
%token LT;
%token LE;
%token EQ;
%token GT;
%token GE;
%token AND;
%token OR;
%token ADD;
%token GET;

%token BRACKETOPEN;
%token BRACKETCLOSE;
%token PARANT_OPEN;
%token PARANT_CLOSE;
%token SEMICOLON;
%token END_BLOCK;
%token BEGIN_BLOCK;
%token ENDL;
```

```
%token IDENTIFIER;
%token INTEGER;
%token STRING;
%token BOOLEAN;


%start program

%%
program : stmt         { printf("program -> stmt\n"); }
        | stmt program { printf("program -> stmt program\n"); }
        ;

stmt : declaration_stmt { printf("stmt -> declaration_stmt\n"); }
     | assignment_stmt  { printf("stmt -> assignment_stmt\n"); }
     | if_stmt          { printf("stmt -> if_stmt\n"); }
     | while_stmt       { printf("stmt -> while_stmt\n"); }
     | read_stmt        { printf("stmt -> read_stmt\n"); }
     | write_stmt       { printf("stmt -> write_stmt\n"); }
     | list_add_stmt    { printf("stmt -> list_add_stmt\n"); }
     ;

declaration_stmt : type IDENTIFIER SEMICOLON { printf("declaration_stmt -> type IDENTIFIER;\
                 ;

type : INT     { printf("type -> int\n"); }
     | BOOL    { printf("type -> bool\n"); }
     | INTLIST { printf("type -> int_list\n"); }
     ;

assignment_stmt : IDENTIFIER ASSIGN expression SEMICOLON { printf("assignment_stmt -> IDENTI
           ;

expression : constant                                 { printf("expression -> constant\n"); }
           | IDENTIFIER                               { printf("expression -> identifier\n");
           | expression compound_operator expression { printf("expression -> expression comp
           | PARANT_OPEN expression PARANT_CLOSE      { printf("expression -> (expression)\n"
           ;

constant : INTEGER { printf("constant -> INTEGER\n"); }
         | BOOLEAN { printf("constant -> BOOLEAN\n"); }
         | STRING  { printf("constant -> STRING\n"); }
         ;

compound_operator : PLUS   STRING  { printf("compound_operator -> +\n"); }
                  | MINUS  STRING  { printf("compound_operator -> -\n"); }
```

```
                    | MUL    STRING  { printf("compound_operator -> *\n"); }
                    | DIV    STRING  { printf("compound_operator -> /\n"); }
                    | MOD    STRING  { printf("compound_operator -> %%\n"); }
                    | LT     STRING  { printf("compound_operator -> <\n"); }
                    | LE     STRING  { printf("compound_operator -> <=\n"); }
                    | EQ     STRING  { printf("compound_operator -> =\n"); }
                    | GT     STRING  { printf("compound_operator -> >\n"); }
                    | GE     STRING  { printf("compound_operator -> >=\n"); }
                    | AND    STRING  { printf("compound_operator -> and\n"); }
                    | OR     STRING  { printf("compound_operator -> or\n"); }
                    | GET    STRING  { printf("compound_operator -> .get\n"); }
                    ;

if_stmt : IF PARANT_OPEN expression PARANT_CLOSE compound_stmt { printf("if_stmt -> if (expr
        ;

compound_stmt : BRACKETOPEN program BRACKETCLOSE { printf("compound_stmt -> { program }\n");
              ;

while_stmt : WHILE PARANT_OPEN expression PARANT_CLOSE compound_stmt { printf("while_stmt ->
           ;

read_stmt : READ IDENTIFIER SEMICOLON { printf("read_stmt -> read identifier;\n"); }
          ;

write_stmt : WRITE IDENTIFIER SEMICOLON { printf("write_stmt -> write identifier;\n"); }
           ;

list_add_stmt : IDENTIFIER ADD expression SEMICOLON { printf("list_add_stmt -> identifier .a
              ;
%%

int yyerror(const char *s) {
    printf("%s\n",s);
    return 0;
}

extern FILE *yyin;

int main(int argc, char** argv) {
    if (argc > 1)
        yyin = fopen(argv[1], "r");
    if (!yyparse())
        fprintf(stderr, "\tOK\n");
}
```

Example

Input:

```
int a;
int b;
int c;

read a;
read b;
read c;

int min;
min <- a;

if (b < min)
{
    min <- b;
}

if (c < min)
{
    min <- c;
}

write min;
```

Output:

```
reserved word: int
type -> int
identifier: a
separator: ;
declaration_stmt -> type IDENTIFIER;
stmt -> declaration_stmt
reserved word: int
type -> int
identifier: b
separator: ;
declaration_stmt -> type IDENTIFIER;
stmt -> declaration_stmt
reserved word: int
type -> int
identifier: c
separator: ;
declaration_stmt -> type IDENTIFIER;
stmt -> declaration_stmt
```

```
reserved word: read
identifier: a
separator: ;
read_stmt -> read identifier;
stmt -> read_stmt
reserved word: read
identifier: b
separator: ;
read_stmt -> read identifier;
stmt -> read_stmt
reserved word: read
identifier: c
separator: ;
read_stmt -> read identifier;
stmt -> read_stmt
reserved word: int
type -> int
identifier: min
separator: ;
declaration_stmt -> type IDENTIFIER;
stmt -> declaration_stmt
identifier: min
operator: <-
identifier: a
expression -> identifier
separator: ;
assignment_stmt -> IDENTIFIER <- expression;
stmt -> assignment_stmt
reserved word: if
separator: (
identifier: b
expression -> identifier
operator: <
identifier: min
syntax error
```