https://github.com/AndreiG23/FLCD

Lex+Yacc

Ghiurcuta Andrei-Bogdan 938pr

Lang.lxi:

```
%{
#include <math.h>
%}

CHARACTER \'[a-zA-Z0-9]\'
INTEGER              ^-?[1-9][0-9]*$
STRING          ^\"[A-Za-z0-9\.\?\!, ]*\"$
constant      {CHARACTER}|{INTEGER}|{STRING}
identifier     ^[a-z][\w]*$

int        printf( "Reserved word: %s\n", yytext);
string     printf( "Reserved word: %s\n", yytext);
array      printf( "Reserved word: %s\n", yytext);
declare    printf( "Reserved word: %s\n", yytext);
read       printf( "Reserved word: %s\n", yytext);
write      printf( "Reserved word: %s\n", yytext);
if         printf( "Reserved word: %s\n", yytext);
else       printf( "Reserved word: %s\n", yytext);
repeat     printf( "Reserved word: %s\n", yytext);
```

```
until      printf( "Reserved word: %s\n", yytext);
for        printf( "Reserved word: %s\n", yytext);
from       printf( "Reserved word: %s\n", yytext);

{identifier}   printf( "Identifier: %s\n", yytext);
{constant}       printf( "Constant: %s\n", yytext );

"+"      printf( "Operator: %s\n", yytext );
"-"      printf( "Operator: %s\n", yytext );
"*"      printf( "Operator: %s\n", yytext );
"/"      printf( "Operator: %s\n", yytext );
"<-"     printf( "Operator: %s\n", yytext );
"="      printf( "Operator: %s\n", yytext );
"!="     printf( "Operator: %s\n", yytext );
"<"      printf( "Operator: %s\n", yytext );
">"      printf( "Operator: %s\n", yytext );
"<="     printf( "Operator: %s\n", yytext );
">="     printf( "Operator: %s\n", yytext );
" "      printf( "Separator: %s\n", yytext );
"{"      printf( "Separator: %s\n", yytext );
"}"      printf( "Separator: %s\n", yytext );
"["      printf( "Separator: %s\n", yytext );
"]"      printf( "Separator: %s\n", yytext );
"("      printf( "Separator: %s\n", yytext );
```

```
")"       printf( "Separator: %s\n", yytext );
":"       printf( "Separator: %s\n", yytext );
";"       printf( "Separator: %s\n", yytext );
","       printf( "Separator: %s\n", yytext );



. printf("Eroare\n");
%%
main( argc, argv )
int argc;
char **argv;
{
   ++argv, --argc; /* skip over program name */
   if ( argc > 0 )
   yyin = fopen( argv[0], "r" );
   else
    yyin = stdin;
   yylex();
}
```

Lang.y:

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

#define YYDEBUG 1
%}



%token int
%token string
%token array
%token declare
%token read
%token write
%token if
%token else
%token repeat
%token until
%token for
%token from

%token identifier
%token constant
```

```
%token ATRIB
%token EQ
%token NE
%token LT
%token LE
%token GT
%token GE
%token NOT
%token ASIGN

%left '+' '-' '*' '/'

%token ADD
%token SUB
%token DIV
%token MOD
%token MUL

%token OPEN_CURLY_BRACKET
%token CLOSED_CURLY_BRACKET
%token OPEN_ROUND_BRACKET
%token CLOSED_ROUND_BRACKET
%token OPEN_RIGHT_BRACKET
```

```
%token CLOSED_RIGHT_BRACKET
%token OPEN_CURLY_BRACKET
%token CLOSED_CURLY_BRACKET
%token OPEN_ROUND_BRACKET
%token CLOSED_ROUND_BRACKET
%token OPEN_RIGHT_BRACKET
%token CLOSED_RIGHT_BRACKET

%token COMMA
%token SEMI_COLON

%start program

%%
program : cmpdStmt
        ;
cmpdStmt: OPEN_CURLY_BRACKET declarations statements
CLOSED_CURLY_BRACKET
        ;
declarations : declaration | declarations declaration
        ;
declaration : declare identifier ATRIB type SEMI_COLON
        ;
type    : typeSimple | arrayType
        ;
```

typeSimple : int | string

;

arrayType : array OPEN_ROUND_BRACKET typeSimple CLOSED_ROUND_BRACKET

;

statements : stmt SEMI_COLON | statements stmt

;

stmt : simpleStmt | structStmt

;

simpleStmt : assignStmt | ioStmt

;

structStmt : ifStmt | loopStmt

;

loopStmt : forStmt | untilStmt

;

op1Exp : ADD exp | SUB exp

;

op2Term : MUL term | DIV term

;

exp : term op1Exp | constant

;

term : factor op2Term

;

factor : OPEN_ROUND_BRACKET exp CLOSED_ROUND_BRACKET | identifier | constant

;

relation : LT | LE | EQ | NE | GT | GE

;

condition : exp relation exp

;

assignStmt : identifier ASIGN exp

;

ioStmt : read identifier | write exp | write identifier

;

ifStmt : if conidtion cmpdStmt

;

untilStmt : repeat cmpdStmt until condition

;

forStmt : for identifier from OPEN_ROUND_BRACKET exp COMMA exp COMMA exp CLOSED_ROUND_BRACKET cmpdStmt

;

constant ; int | char | string | emptyArray

;

emptyArray : OPEN_RIGHT_BRACKET CLOSED_RIGHT_BRACKET

;

%%

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

extern FILE *yyin;

int 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");
}
```