Lex-Yacc Laboratory

Guliciuc Ștefan, 934/1

Github link: https://github.com/stefan99x/FLCD/tree/master/Lab08_LEX

lang.lxi

```
%{
#include <stdio.h>
#include <string.h>
#include "y.tab.h"

int currentLine = 1;
%}

%option noyywrap
%option caseless

DIGIT         [0-9]
NZ_DIGIT      [1-9]
ZERO          [0]
NUMBER        {NZ_DIGIT}{DIGIT}*
SIGN          [+]|[-]
INTEGER               {ZERO}|{NUMBER}|{SIGN}{NUMBER}
SIGNER_INTEGER {SIGN}{NUMBER}
SPECIAL_CHAR
"_"|"."|","|";"|":"|"?"|"!"|"@"|"/"|"("|")"|"-"|"+"|"="|"{"|"}"|"*"|"["|
"]"|"$"|"%"|"^"|" "
CHAR          {DIGIT}|{SPECIAL_CHAR}|[a-zA-Z]
CHARACTER      "'"{CHAR}"'"
STRING         [\"]{CHAR}*[\"]
CONSTANT              {STRING}|{INTEGER}|{CHARACTER}
IDENTIFIER            [a-zA-Z][a-zA-Z0-9_]*
```

```
%%


"&&" {printf("%s\n",yytext);return AND;}
"||" {printf("%s\n",yytext);return OR;}
ton {printf("%s\n",yytext);return NOT;}
fi {printf("%s\n",yytext);return IF;}
esle {printf("%s\n",yytext);return ELSE;}
file {printf("%s\n",yytext);return ELIF;}
elihw {printf("%s\n",yytext);return WHILE;}
rof {printf("%s\n",yytext);return FOR;}
daer {printf("%s\n",yytext);return READ;}
etirw {printf("%s\n",yytext);return WRITE;}
regetni {printf("%s\n",yytext);return INTEGER;}
gnirts {printf("%s\n",yytext);return STRING;}
rahc {printf("%s\n",yytext);return CHAR;}
margopr {printf("%s\n",yytext);return PROGRAM;}
noitcnuf {printf("%s\n",yytext);return FUNCTION;}
loop {printf("%s\n",yytext);return BOOL;}
nruter {printf("%s\n",yytext);return RETURN;}
true {printf("%s\n",yytext);return TRUE;}
false {printf("%s\n",yytext);return FALSE;}



{CONSTANT} {printf("%s\n",yytext);return CONSTANT;}
{IDENTIFIER} {printf("%s\n",yytext);return IDENTIFIER;}
{NZ_DIGIT} {printf("%s\n",yytext);return NON_ZERO_DIGIT;}
{DIGIT} {printf("%s\n",yytext);return NUMBER_DIGIT;}


; {printf("%s\n",yytext);return SEMI_COLON;}
"," {printf("%s\n",yytext);return COMMA;}
\t {printf("%s\n",yytext);return DOT;}
\{ {printf("%s\n",yytext);return OPEN_CURLY_BRACKET;}
```

```
\} {printf("%s\n",yytext);return CLOSED_CURLY_BRACKET;}
\[ {printf("%s\n",yytext);return OPEN_SQUARE_BRACKET;}
\] {printf("%s\n",yytext);return CLOSED_SQUARE_BRACKET;}
\( {printf("%s\n",yytext);return OPEN_ROUND_BRACKET;}
\) {printf("%s\n",yytext);return CLOSED_ROUND_BRACKET;}


\+ {printf("%s\n",yytext);return PLUS;}
\- {printf("%s\n",yytext);return MINUS;}
\* {printf("%s\n",yytext);return MUL;}
\/ {printf("%s\n",yytext);return DIV;}
\% { printf("%s\n",yytext);return PERCENT;}
\< {printf("%s\n",yytext); return LT;}
\> { printf("%s\n",yytext);return GT;}
\<= {printf("%s\n",yytext); return LE;}
\>= { printf("%s\n",yytext);return GE;}
"=" { printf("%s\n",yytext);return ATRIB;}
\== { printf("%s\n",yytext);return EQ;}
\!= { printf("%s\n",yytext);return NOT_EQ;}


[\n\r] {currentLine++;}
[ \t\n]+ {}


[a-zA-Z_0-9][a-zA-Z0-9_]* {printf("%s - illegal identifier found at line
%d\n", yytext, currentLine);}
\'[a-zA-Z0-9]*\' {printf("%s - illegal char at line %d, did you mean
string?\n", yytext, currentLine);}
[\"]{CHAR}* {printf("%s - illegal string constant at line, you forgot to
close it %d\n", yytext, currentLine);}


. {printf("%s - illegal token found at line %d\n",yytext, currentLine);}


%%
```

lang.y :

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

    #define YYDEBUG 1
%}

%token AND
%token OR
%token NOT
%token IF
%token ELSE
%token ELIF
%token WHILE
%token FOR
%token READ
%token WRITE
%token INTEGER
%token STRING
%token CHAR
%token PROGRAM
%token FUNCTION
%token BOOL
%token RETURN
%token TRUE
%token FALSE
%token CONSTANT
%token IDENTIFIER
%token NON_ZERO_DIGIT
%token NUMBER_DIGIT
%token SEMI_COLON
%token COMMA
```

```
%token DOT

%token OPEN_CURLY_BRACKET

%token CLOSED_CURLY_BRACKET

%token OPEN_SQUARE_BRACKET

%token CLOSED_SQUARE_BRACKET

%token OPEN_ROUND_BRACKET

%token CLOSED_ROUND_BRACKET

%token PLUS

%token MINUS

%token MUL

%token DIV

%token PERCENT

%token LT

%token GT

%token LE

%token GE

%token ATRIB

%token EQ

%token NOT_EQ



%start program



%%



program : FUNCTION  cmpstmt

       ;



parameters : declaration

          | declaration COMMA parameters

          ;



declaration : type IDENTIFIER
```

```
            ;


type : prim_type
    | arr_type
    ;


prim_type : INTEGER
          | CHAR
          | STRING
          | BOOL
          ;


arr_type : prim_type CLOSED_SQUARE_BRACKET nr OPEN_SQUARE_BRACKET
         ;


nr : NON_ZERO_DIGIT
  | NON_ZERO_DIGIT CLOSED_CURLY_BRACKET NUMBER_DIGIT
OPEN_CURLY_BRACKET
  ;







cmpstmt : CLOSED_CURLY_BRACKET stmtlist OPEN_CURLY_BRACKET
        ;


stmtlist : stmt SEMI_COLON
         | stmt SEMI_COLON stmtlist
         ;


stmt : simpstmt
    | structstmt
```

```
    ;


simpstmt : assignstmt
        | iostmt
        ;


assignstmt : IDENTIFIER ATRIB expression
          ;


expression : expression PLUS term
          | expression MINUS term
          | term
          ;


term : term MUL factor
    | factor
    | term DIV factor
    | term PERCENT factor
    ;


factor : CLOSED_ROUND_BRACKET expression OPEN_ROUND_BRACKET
      | IDENTIFIER
      | arr_type
      ;


iostmt : READ OPEN_ROUND_BRACKET IDENTIFIER CLOSED_ROUND_BRACKET
      | READ OPEN_ROUND_BRACKET STRING CLOSED_ROUND_BRACKET
      | WRITE OPEN_ROUND_BRACKET IDENTIFIER CLOSED_ROUND_BRACKET
      | WRITE OPEN_ROUND_BRACKET STRING CLOSED_ROUND_BRACKET
      ;
```

```
structstmt : ifstmt
           | loopstmt
           | whilestmt
           ;


ifstmt : IF condlist cmpstmt
       | IF condlist cmpstmt ELSE ifstmt
       ;


condition : expression relation expression
          ;


condlist : condition
         | condition operator condlist
         ;


relation : LT
         | GT
         | LE
         | GE
         | EQ
         | NOT_EQ
         ;


loopstmt : FOR loopcond cmpstmt
         ;


loopcond : assignstmt SEMI_COLON condlist SEMI_COLON assignstmt
         ;


whilestmt : WHILE condlist cmpstmt
```

```
            ;


operator : AND
         | OR
         ;

%%


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, "Merge Scarba\n");
}
```

Run code:

1. lex lang.lxi
2. yacc -d lang.y
3. gcc lex.yy.c y.tab.c -o exe -lfl
4. ./exe < p1.txt