Lab 8-9

Georgiana Loba

https://github.com/GeorgianaLoba/Formal-Languages-and-Compiler-Design/tree/master/lex-yacc

Statement: Use lex

You may use any version (LEX or FLEX)

1. Write a LEX specification containing the regular expressions corresponding to your language specification - see lab 1
2. Use Lex in order to obtain a scanner. Test for the same input as in lab 1 (p1, p2).

Deliverables: pdf file containing lang.lxi (lex specification file) + demo

Statement: Use yacc

You may use any version (yacc or bison)

1. Write a specification file containing the production rules corresponding to the language specification (use syntax rules from lab1).
2. Then, use the parser generator (no errors)

Deliverables: lang.y (yacc specification file)

Lang.lxi is:

-- on github, the file called: geoslex.txt.bak

```
%{
   #include<stdio.h>
   #include <string.h>
%}

%option noyywrap
%option caseless
```

```
LETTER [a-ZA-Z]

DIGIT [0-9]

NON_ZERO_DIGIT [1-9]

INTEGER [+-]?{NON_ZERO_DIGIT}{DIGIT}*

CHAR \'{LETTER}\'

STRING \"{CHAR}*\"

CONSTANT {INTEGER}|{CHAR}|{STRING}

IDENTIFIER [a-zA-Z][a-ZA-Z0-9_]*


%%


let {printf("%s - as reserved word\n", yytext);}

func {printf("%s - as reserved word\n", yytext);}

returns {printf("%s - as reserved word\n", yytext);}

is {printf("%s - as reserved word\n", yytext);}

or {printf("%s - as reserved word\n", yytext);}

and {printf("%s - as reserved word\n", yytext);}

print {printf("%s - as reserved word\n", yytext);}

while {printf("%s - as reserved word\n", yytext);}

return {printf("%s - as reserved word\n", yytext);}

if {printf("%s - as reserved word\n", yytext);}

else {printf("%s - as reserved word\n", yytext);}

then {printf("%s - as reserved word\n", yytext);}

integer {printf("%s - as reserved word\n", yytext);}

boolean {printf("%s - as reserved word\n", yytext);}
```

```
array {printf("%s - as reserved word\n", yytext);}

true {printf("%s - as reserved word\n", yytext);}

false {printf("%s - as reserved word\n", yytext);}

scan {printf("%s - as reserved word\n", yytext);}


{IDENTIFIER} printf( " - as identifier: %s\n", yytext);

{CONSTANT} printf( " - as identifier: %s\n", yytext);


\{ {printf("%s\n", yytext);}

\} {printf("%s\n", yytext);}

; {printf("%s\n", yytext);}

\( {printf("%s\n", yytext);}

\) {printf("%s\n", yytext);}

\, {printf("%s\n", yytext);}


\+ {printf("%s\n", yytext);}

\- {printf("%s\n", yytext);}

\* {printf("%s\n", yytext);}

\/ {printf("%s\n", yytext);}

\% {printf("%s\n", yytext);}

"=" {printf("%s\n", yytext);}

\== {printf("%s\n", yytext);}

\!= {printf("%s\n", yytext);}

\< {printf("%s\n", yytext);}

\> {printf("%s\n", yytext);}

\<= {printf("%s\n", yytext);}
```

```
\>= {printf("%s\n", yytext);}


%%
void main(argc, argv)
int argc;
char** argv;
{
if (argc > 1)
{
        FILE *file;
        file = fopen(argv[1], "r");
        if (!file)
        {
        fprintf(stderr, "Could not open %s\n", argv[1]);
        exit(1);
        }
        yyin = file;
}


yylex();
}
```

Lang.y is:

-- on github, the file called: geoyacc.y

```
%{
```

```
        #include<stdio.h>
        #include<stdlib.h>
        #define YYDEBUG1
}%


%token LET
%token FUNC
%token RETURNS
%token IS
%token OR
%token PRINT
%token WHILE
%token RETURN
%token IF
%token ELSE
%token THEN
%token INTEGER
%token BOOLEAN
%token TRUE
%token STRING
%token CHAR
%token FALSE
%token SCAN
%token PRINT
%token IDENTIFIER
```

```
%token CONSTANT

%token RELATION

%token COMMA

%token SEMI_COLON

%token OPEN_SQUARE_BRACKET

%token CLOSED_SQUARE_BRACKET

%token OPEN_CURLY_BRACKET

%token CLOSED_CURLY_BRACKET

%token OPEN_ROUND_BRACKET

%token CLOSED_ROUND_BRACKET

%token PLUS

%token MINUS

%token DIV

%token MUL

%token PERCENT

%token EQ

%token NOT_EQ


%start program


%%


program: LET FUNC IDENTIFIER function_arguments RETURNS type
OPEN_CURLY_BRACKET statement_list CLOSED_CURLY_BRACKET SEMI_COLON;

type: INTEGER

        | BOOLEAN

        | STRING
```

```
        | CHAR
        ;
statement_list: statement
        | statemenet_list
        ;
statement: simple_declaration
        | assigned_declaration
        | assignment
        | if_statemenet
        | while_statement
        | output_statement
        | input_statement
        ;
simple_declaration: LET type IDENTIFIER SEMI_COLON;
assigned_declaration: LET IDENTIFIER EQ expression SEMI_COLON;
assignment: IDENTIFIER EQ expression SEMI_COLON;
function_arguments: OPEN_ROUND_BRACKET type IDENTIFIER
CLOSED_ROUND_BRACKET;
if_statement: IF condition THEN OPEN_CURLY_BRACKET statement_list
CLOSED_CURLY_BRACKET ELSE
        OPEN_CURLY_BRACKET statement_list CLOSED_CURLY_BRACKET;
while_statement: WHILE condition THEN OPEN_CURLY_BRACKET statement_list
CLOSED_CURLY_BRACKET;
condition: OPEN_ROUND_BRACKET expression RELATION expression
CLOSED_ROUND_BRACKET;
expression: IDENTIFIER operand IDENTIFIER
        | IDENTIFIER operand CONSTANT
        | CONSTANT operand IDENTIFIER
```

```
        | CONSTANT operand CONSTANT
        ;
operand: PLUS
        | MINUS
        | DIV
        | MUL
        | PERCENT
        ;
output_statement: PRINT STRING SEMI_COLON
        | print IDENTIFIER SEMI_COLON
        ;
input_statement: SCAN type IDENTIFIER SEMI_COLON;
```

%%