https://github.com/Natan-Gabriel/FLCD/tree/master/lex

Name: Tiutiu Natan-Gabriel,937

DOCUMENTATION LEX

This is how to generate the final result:

lex spec.lxi

gcc lex.yy.c -o my_lex -ll

./my_lex < p1.txt (or p2.txt or p3.txt)

The lang.lxi file contains:

```
%{
//#include "sspascal.tab.h"
%}

%option noyywrap
%option caseless


DIGIT          [0-9]
NONZERODIGIT    [1-9]
LETTER         [A-Za-z]
CONST          {NONZERODIGIT}*({DIGIT})*
INTEGER         "0"|[+-]{0,1}{CONST}
```

```
STIRNG          \"("-"|":"|[a-zA-Z0-9 ])*\"

IDENTIFIER      {LETTER}({LETTER}|{DIGIT})*

OPERATORS       [+-*/%<>=!]|"-"|"<="|">="|"!="|"&&"|"||"|"=="

SEPARATORS      [()[]{};]|" "




%%



"+"|"-"|"*"|"/"|"%"|"<"|"<="|">="|">"|"!="|"="|"&&"|"||"|"!"|"=="
{printf("An operator: %s \n", yytext) ;}



"("|")"|"["|"]"|"{"|"}"|";"|" "      {printf("A separator: %s \n", yytext) ;}

"\n"      {}


char|int|string|boolean|array|for|while|if|else|elif|of|program|read|print
{printf("A reserved word: %s \n", yytext) ;}
```

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

{INTEGER}       {printf("An integer: %s \n", yytext) ;}
{STIRNG}        {printf("A string: %s \n", yytext) ;}

.                           {printf( "Unrecognized character: %s\n", yytext );}

%%
```

The output for running “./my_lex < p1.txt” will be

```
A reserved word: int
A separator:
An identifier: a
A separator: ;
An identifier: a
A separator:
An operator: =
A separator:
An integer: 7
A separator: ;
A reserved word: int
A separator:
An identifier: b
```

A separator: ;

An identifier: b

An operator: =

An integer: 7

A separator: ;

A reserved word: int

A separator:

An identifier: c

A separator: ;

An identifier: c

A separator:

An operator: =

A separator:

An integer: 7

A separator: ;

A reserved word: if

A separator: (

An identifier: a

An operator: >=

An identifier: b

A separator:

An operator: &&

A separator:

An identifier: a

An operator: >=

An identifier: c

A separator:)

A separator: {

A separator:

A reserved word: print

A separator: (

A string: "the maximum number is a"

A separator:)

A separator: ;

A separator: }

A reserved word: elif

A separator: (

An identifier: b

An operator: >=

An identifier: a

A separator:

An operator: &&

A separator:

An identifier: b

An operator: >=

An identifier: c

A separator:)

A separator: {

A separator:

A reserved word: print

A separator: (

A string: "the maximum number is b"

A separator:)

A separator: ;

A separator: }

A reserved word: else

A separator: {

A separator:

A reserved word: print

A separator: (

A string: "the maximum number is c"

A separator:)

A separator: ;

A separator: }

A separator: ;