Lex - lab8

github link: https://github.com/914-Mathe-Andrei/lftc/tree/main/lab8

Specification

```
%{
int line = 1;
%}

%option noyywrap

WHITESPACE              [[:space:]]
NEWLINE                 \n

UNDERSCORE              _

UPPERCASE_LETTER        [A-Z]
LOWERCASE_LETTER        [a-z]
LETTER                  ({UPPERCASE_LETTER}|{LOWERCASE_LETTER

ZERO_DIGIT              0
NONZERO_DIGIT           [1-9]
DIGIT                   ({ZERO_DIGIT}|{NONZERO_DIGIT})

OPERATOR                (&&|\|\||==|!=|<=|>=|<|>|\+|-|\*|\/|%
SEPARATOR               (\(|\)|\[|\]|\{|\}|:|;|,|->)
RESERVED_WORD           (int|str|var|const|if|else|while|read

ID                      ({UNDERSCORE}|{LETTER})({UNDERSCORE}|
INT                     ({ZERO_DIGIT}|{NONZERO_DIGIT}{DIGIT}*
STR                     "({LETTER}|{DIGIT})+"

%%

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

```
{RESERVED_WORD}         printf("%s KEYWORD\n", yytext);
{ID}                    printf("%s ID\n", yytext);
{INT}{ID}               { printf("LexicalError (line %d): id
{ZERO_DIGIT}{INT}       { printf("LexicalError (line %d): int
{INT}                   printf("%s INT\n", yytext);
{STR}                   printf("%s STR\n", yytext);
{NEWLINE}               { line++; }
{WHITESPACE}
<<EOF>>                 return 0;
.                       { printf("LexicalError (line %d): unk

%%

int main(int argc, char** argv) {
      yylex();
      return 0;
}
```

Example

▼ **Input**

input > p1.mat

```
fn max(num1: int, num2: int, num3: int) -> int {
    var res: int;

    if (num1 > num2) {
        if (num1 > num3) {
            res = num1;
        } else {
            res = num3;
        }
    } else {
        if (num2 > num3) {
            res = num2;
        } else {
            res = num3;
        }
    }

    return res;
}

fn main() -> int {
    max(1, 2, 3);
    return 1;
}
```

▼ **Output**

output > PIF.out

```
fn KEYWORD
max ID
( SEPARATOR
num1 ID
: SEPARATOR
int KEYWORD
, SEPARATOR
num2 ID
: SEPARATOR
int KEYWORD
, SEPARATOR
num3 ID
: SEPARATOR
int KEYWORD
) SEPARATOR
-> SEPARATOR
int KEYWORD
{ SEPARATOR
var KEYWORD
res ID
: SEPARATOR
int KEYWORD
; SEPARATOR
if KEYWORD
( SEPARATOR
num1 ID
> OPERATOR
num2 ID
) SEPARATOR
{ SEPARATOR
```

```
31  if KEYWORD
32  ( SEPARATOR
33  num1 ID
34  > OPERATOR
35  num3 ID
36  ) SEPARATOR
37  { SEPARATOR
38  res ID
39  = OPERATOR
40  num1 ID
41  ; SEPARATOR
42  } SEPARATOR
43  else KEYWORD
44  { SEPARATOR
45  res ID
46  = OPERATOR
47  num3 ID
48  ; SEPARATOR
49  } SEPARATOR
50  } SEPARATOR
51  else KEYWORD
52  { SEPARATOR
53  if KEYWORD
54  ( SEPARATOR
55  num2 ID
56  > OPERATOR
57  num3 ID
58  ) SEPARATOR
59  { SEPARATOR
60  res ID
```

```
61  = OPERATOR
62  num2 ID
63  ; SEPARATOR
64  } SEPARATOR
65  else KEYWORD
66  { SEPARATOR
67  res ID
68  = OPERATOR
69  num3 ID
70  ; SEPARATOR
71  } SEPARATOR
72  } SEPARATOR
73  return KEYWORD
74  res ID
75  ; SEPARATOR
76  } SEPARATOR
77  fn KEYWORD
78  main ID
79  ( SEPARATOR
80  ) SEPARATOR
81  -> SEPARATOR
82  int KEYWORD
83  { SEPARATOR
84  max ID
85  ( SEPARATOR
86  1 INT
87  , SEPARATOR
88  2 INT
89  , SEPARATOR
90  3 INT
```

```
90  3 INT
91  ) SEPARATOR
92  ; SEPARATOR
93  return KEYWORD
94  1 INT
95  ; SEPARATOR
96  } SEPARATOR
97
```