Lab8 - Lex

> Github link: https://github.com/DiaconuAna/Formal-Languages-and-Compiler-Design/tree/main/Lab8 - lex%26yacc

▼ `lang.lxi`

```
%option noyywrap

%{
#include <math.h>
%}

DIGIT     [0-9]
WORD    \"[a-zA-Z0-9]*\"
INTEGER   [+-]?[1-9][0-9]*
CHARACTER   \'[a-zA-Z0-9]\'
constant  {WORD}|{INTEGER}|{CHARACTER}
identifier  [a-zA-Z][a-zA-Z0-9]*

%%

in           printf( "Reserved word: %s\n", yytext);
out          printf( "Reserved word: %s\n", yytext);
begin        printf( "Reserved word: %s\n", yytext);
number       printf( "Reserved word: %s\n", yytext);
string       printf( "Reserved word: %s\n", yytext);
character    printf( "Reserved word: %s\n", yytext);
if           printf( "Reserved word: %s\n", yytext);
end          printf( "Reserved word: %s\n", yytext);
end_if       printf( "Reserved word: %s\n", yytext);
end_for      printf( "Reserved word: %s\n", yytext);
while        printf( "Reserved word: %s\n", yytext);
end_while    printf( "Reserved word: %s\n", yytext);
else         printf( "Reserved word: %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( "Assignment Operator: %s\n", yytext );
"+"        printf( "Operator: %s\n", yytext );
"-"        printf( "Operator: %s\n", yytext );
"*"        printf( "Operator: %s\n", yytext );
"/"        printf( "Operator: %s\n", yytext );
"mod"      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 );
"and"      printf( "Operator: %s\n", yytext );
"or"       printf( "Operator: %s\n", yytext );
"not"      printf( "Operator: %s\n", yytext );

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

[ \t]+
[\n]+

[+-]?0[0-9]*                    printf("Illegal integer at line\n");
[0-9]+[a-zA-Z_]+[a-zA-Z0-9_]*   printf("Illegal identifier\n");
\'[a-zA-Z0-9]{2,}\'             printf("Character of length >=2 at line\n");
.                               printf("Lexical error\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();
}
```

▼ p1.txt

```
begin:
number a;
number b;
number c;
number min;
in a;
in b;
in c;
min <- a;
if (b < min):
min <- b;
end_if
if (c < min):
min <- c;
end_if
out min;
end
```

▼ p1_out.txt

```
Reserved word: begin
Separator: :
Reserved word: number
Identifier: a
Separator: ;
Reserved word: number
Identifier: b
Separator: ;
Reserved word: number
Identifier: c
Separator: ;
Reserved word: number
Identifier: min
Separator: ;
Reserved word: in
Identifier: a
Separator: ;
Reserved word: in
Identifier: b
Separator: ;
Reserved word: in
Identifier: c
Separator: ;
Identifier: min
Assignment Operator: <-
Identifier: a
Separator: ;
Reserved word: if
Separator: (
Identifier: b
Operator: <
Identifier: min
Separator: )
Separator: :
Identifier: min
Assignment Operator: <-
Identifier: b
Separator: ;
Reserved word: end_if
Reserved word: if
Separator: (
Identifier: c
Operator: <
Identifier: min
Separator: )
Separator: :
Identifier: min
Assignment Operator: <-
Identifier: c
Separator: ;
Reserved word: end_if
Reserved word: out
Identifier: min
Separator: ;
Reserved word: end
```

▼ p2.txt

```
begin:
  number a;
  number div;

  in a;

  for div<-1,a,1:
    if div mod a = 0:
      out div;
    end_if
  end_for
end
```

▼ p2_out.txt

```
Reserved word: begin
Separator: :
Reserved word: number
Identifier: a
Separator: ;
Reserved word: number
Identifier: div
Separator: ;
Reserved word: in
Identifier: a
Separator: ;
Identifier: for
Identifier: div
Assignment Operator: <-
Constant: 1
Separator: ,
Identifier: a
Separator: ,
Constant: 1
Separator: :
Reserved word: if
Identifier: div
Operator: mod
Identifier: a
Operator: =
Constant: 0
Separator: :
Reserved word: out
Identifier: div
Separator: ;
Reserved word: end_if
Reserved word: end_for
Reserved word: end
```