Demo:

Just type anything and the program will identify the written sequence based on the language specification (if it's an identifier, separator, operator etc).

```
bbfd
Identifier: bbfd
_vfdv
Lexical error
Identifier: vfdv
_vfd
Lexical error
Identifier: vfd
0vf
Illegal identifier
{fv
Separator: {
Identifier: fv
2+3
Constant: 2
Constant: +3
2 + 3
Constant: 2
Operator: +
Constant: 3
int a = 4;
Reserved word: int
Identifier: a
Operator: =
Constant: 4
Separator: ;
```

Code:

```
%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}
IDENIFIER           [a-zA-Z][a-zA-Z0-9_]*

%%

read        printf( "Reserved word: %s\n", yytext);
write         printf( "Reserved word: %s\n", yytext);
if              printf( "Reserved word: %s\n", yytext);
else          printf( "Reserved word: %s\n", yytext);
for             printf( "Reserved word: %s\n", yytext);
while         printf( "Reserved word: %s\n", yytext);
break         printf( "Reserved word: %s\n", yytext);
int             printf( "Reserved word: %s\n", yytext);
string        printf( "Reserved word: %s\n", yytext);
char        printf( "Reserved word: %s\n", yytext);
list            printf( "Reserved word: %s\n", yytext);
return      printf( "Reserved word: %s\n", yytext);
```

```
{IDENIFIER}          printf( "Identifier: %s\n", yytext);
{CONSTANT}           printf( "Constant: %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( "Separator: %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 );
"<"             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 );
"!"             printf( "Operator: %s\n", yytext );
"&&"    printf( "Operator: %s\n", yytext );
"||"      printf( "Operator: %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();
}
```