Documentation Lab8

https://github.com/Aurelian-Iancu/UBB-Computer-Science/tree/main/Semester5/Formal%20Languages%20and%20Compiler%20Design/Lab8

Lex file:

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

    int lines = 1;
%}

%option noyywrap
%option caseless

DIGIT [0-9]
NON_ZERO_DIGIT [1-9]
INT_CONSTANT [+-]?{NON_ZERO_DIGIT}{DIGIT}*|0
LETTER [a-zA-Z_ă]
SPECIAL_CHAR [ ?:*\^+=.!]
STRING_CONSTANT (\"({LETTER}|{DIGIT}|{SPECIAL_CHAR})*\")
IDENTIFIER {LETTER}({LETTER}|{DIGIT})*
BAD_IDENTIFIER ({DIGIT})+({LETTER})+({LETTER}|{DIGIT})*

%%

"vector"|"char"|int|"string"|"bagă"|"arată"|"oare"|"altfel"|"atunci"|"cattimp"|"fă"|"start"|"oftype" {printf("%s - reserved word\n", yytext);}
```

```
"+"|"-"|"*"|"/"|"=="|"<"|">"|"<="|">="|"mod"|"=" printf("%s - operator\n", yytext);

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

{BAD_IDENTIFIER} {printf("Error at token %s at line %d\n", yytext, lines); exit(1);}

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

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

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

[ \t]+ {}

[\n]+ {++lines;}

. {printf("Error at token %s at line %d\n", yytext, lines); exit(1);}

%%

int main(int argc, char** argv) {
	if (argc > 1)
		yyin = fopen(argv[1], "r");
	else
		yyin = stdin;
	yylex();
}
```

DEMO:

First command

```
C:\Users\Aurelian\Documents\GitHub\Personal\UBB-Computer-Science\Semester5\Formal Languages and Compiler Design\Lab8>flex lang.lxi
```

Second command

```
C:\Users\Aurelian\Documents\GitHub\Personal\UBB-Computer-Science\Semester5\Formal Languages and Compiler Design\Lab8>gcc lex.yy.c
```

Third command:

We run the executable we just created with parameter the name of the file we want to scan. a.exe is the name of the executable and p1.txt is the name of the file.

```
C:\Users\Aurelian\Documents\GitHub\Personal\UBB-Computer-Science\Semester5\Formal Languages and Compiler Design\Lab8>a.exe p1.txt
```

The result is:

```
start - reserved word
{ - separator
a - identifier
oftype - reserved word
int - reserved word
; - separator
bag—â - reserved word
( - separator
a - identifier
) - separator
; - separator
divizor - identifier
oftype - reserved word
int - reserved word
; - separator
divizor - identifier
= - operator
2 - integer constant
; - separator
ok - identifier
oftype - reserved word
int - reserved word
; - separator
ok - identifier
= - operator
0 - integer constant
; - separator
cattimp - reserved word
( - separator
divizor - identifier
< - operator
a - identifier
/ - operator
2 - integer constant
) - separator
f—â - reserved word
{ - separator
oare - reserved word
( - separator
a - identifier
mod - operator
b - identifier
) - separator
atunci - reserved word
```

```
{ - separator
ok - identifier
= - operator
1 - integer constant
; - separator
} - separator
divizor - identifier
= - operator
divizor - identifier
+ - operator
1 - integer constant
; - separator
} - separator
oare - reserved word
( - separator
ok - identifier
== - operator
1 - integer constant
) - separator
atunci - reserved word
{ - separator
arat–â - reserved word
( - separator
"Numarul este prim!" - string constant
) - separator
; - separator
} - separator
altfel - reserved word
{ - separator
arat–â - reserved word
( - separator
"Numarul nu este prim!" - string constant
) - separator
; - separator
} - separator
} - separator
```

The file content of p1.txt:

start

{

```
    a oftype int;
    bagă ( a );
    divizor oftype int;
    divizor = 2;
    ok oftype int;
    ok = 0;
    cattimp ( divizor < a / 2 ) fă {
        oare( a mod b ) atunci {
            ok = 1;
        }
        divizor = divizor + 1;
    }

    oare ( ok == 1 ) atunci {
        arată ( "Numarul este prim!" );
    }
    altfel {
        arată( "Numarul nu este prim!" );
    }
}
```