Documentation LEX

myscannerLab8.lxi file:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void yyerror(char *msg);
%}

%option noyywrap

IDENTIFIER    [a-zA-Z_][a-zA-Z0-9_]*
INT_CONST     0|[-]?[1-9][0-9]*
DOUBLE_CONST   [-]?[1-9][0-9]*([.][0-9]+)|[-]?0[.][0-9]+
STRING_CONST   [\"][a-zA-Z0-9 ]*[\"]



%%

"read"|"write"|"if"|"else"|"whileLoop"|"forLoop"|"boolean"|"int"|"double"|"string"
{printf("Reserved words: %s\n", yytext);}
"+"|"-"|"*"|"/"|"%"|"<"|"<="|"=="|"!="|">="|">"|"="       {printf("Operator: %s\n", yytext);}
"{"|"}"|"["|"]"|"("|")"|","|";"       {printf("Separator: %s\n", yytext);}
{IDENTIFIER}    {printf("Identifier %s\n", yytext);}
{INT_CONST}     {printf("Integer constant: %s\n", yytext);}
{STRING_CONST}   {printf("String constant: %s\n", yytext);}
{DOUBLE_CONST}    {printf("Double constant: %s\n", yytext);}

[ \t]+   {}
[ \n]+   {}

%%

void yyerror(char *msg){
        fprintf(stderr,"%s\n",msg);
        exit(1);
}

void main(argc, argv)
int argc;
char** argv;
{
if(argc > 1)
{
        FILE *file;
        file = fopen(argv[1], "r");
        if(!file)
        {
```

```
                fprintf(stderr, "Could not open %s\n", argv[1]);
                exit(1);
        }
        yyin = file;
}
yylex();
}
```

1. Overview

The myscannerLab8.lxi file is a Lex (Flex) program designed to recognize and classify tokens in a simple programming language. It recognizes keywords, operators, separators, identifiers, integer constants, string constants, and character constants.

2. Preparation

Before compiling and running the program, ensure that you have a command prompt opened at the location of the myscannerLab8.lxi file. Navigate to the directory using the cd command:

$ cd path/to/directory

Replace path/to/directory with the actual path where your myscannerLab8.lxi file is located. Once you are in the correct directory, proceed with the compilation and execution steps as described in the next sections.

3. Compilation

Compliation:

To compile the Lex file, use the following command:

$flex myscannerLab8.lxi

```
D:\FACULTATE\Materiale facultate 2023-2024\LFTC\Labs\Lab8>flex myscannerLab8.lxi
```

Generate Executable:

After compiling, you will obtain the lex.yy.c file. Use the following command to generate the executable:

gcc lex.yy.c

```
D:\FACULTATE\Materiale facultate 2023-2024\LFTC\Labs\Lab8>gcc lex.yy.c
```

Running the Program:

Run the compiled program by providing an input file as an argument:

a.exe p1.txt
Or
a.exe p2.txt
Or
a.exe p3.txt

Replace p1.txt with the path to your desired input file.

4. Token Recognition

Reserved Words

Recognizes reserved keywords such as read, write, if, else, whileLoop, forLoop, boolean, int, double, string.

Operators

Identifies arithmetic and relational operators: +, -, *, /, %, <, <=, ==, !=, >=, >,=.

Separators

Recognizes separators including {, }, [,], (,), „, ;.

Identifiers

Matches identifiers according to the regular expression [a-zA-Z_][a-zA-Z0-9_]*.

Integer Constants

Matches integer constants using the regular expression 0|[-]?[1-9][0-9]*.

String Constants

Recognizes string constants enclosed in double quotes, e.g., "Hello, World!", using the regular expression [\"][a-zA-Z0-9]*[\"]

Double Constants

Matches double constants using the regular expression [-]?[1-9][0-9]*([.][0-9]+)|[-]?0[.][0-9]+

```
D:\FACULTATE\Materiale facultate 2023-2024\LFTC\Labs\Lab8>a.exe p1.txt
Reserved words: int
Identifier array
Separator: [
Integer constant: 1
Separator: ,
Integer constant: 2
Separator: ,
Integer constant: 3
Separator: ,
Integer constant: 4
Separator: ,
Integer constant: 5
Separator: ]
Separator: ;
Reserved words: int
Identifier i
Separator: ;
Reserved words: int
Identifier sum
Separator: ;
Reserved words: int
Identifier n
Separator: ;
Reserved words: double
Identifier average
Separator: ;
Identifier i
Operator: =
Integer constant: 0
Separator: ;
Identifier sum
Operator: =
Integer constant: 0
Separator: ;
Identifier n
Operator: =
Integer constant: 5
Separator: ;
Reserved words: whileLoop
Separator: (
Identifier i
Operator: <
Identifier n
Separator: )
Separator: {
Identifier sum
Operator: =
Identifier sum
Operator: +
Identifier array
Separator: [
Identifier i
Operator: +
Integer constant: 1
Separator: ]
Separator: ;
Identifier i
Operator: =
Identifier i
Operator: +
```