Github link:

flex -o lang.c lang.l

gcc -o lang lang.c

./lang p1.txt

Lab 8 – lex:

```
%option noyywrap
%{
#include <stdio.h>
%}


identifier ([a-zA-Z])([a-zA-Z]|[0-9])*

integer (0|([-+]?([1-9][0-9]*)))

string [a-zA-Z ]*

constchar "\'"[a-zA-Z ]"\'"

conststring \"{string}\"

separator " "|"("|")"|"["|"]"|"{"|"}"|";"|":"

whitespace " "|"\n"|"\t"|"\r"

operator "+"|"-"|"*"|"/"|"="|"<"|"<="|">"|">="|"!="|"+="|"-="|"*="|"/="

reserved
"array"|"char"|"const"|"string"|"DECLARE"|"DO"|"ELSE"|"FOR"|"if"|"INTEGER"|"MAIN"|"OF"|"READ
"|"THEN"|"VAR"|"WHILE"|"WRITE"


invalid_id {integer}{identifier}

invalid_string (\'{string}\')|(\'{string})|({string}\')|(\"{string})|({string}\")

invalid_integer ([-+]?0{integer}|[-+]0)

error ({invalid_id}|{invalid_string}|{invalid_integer}|.)


%%
```

```
{whitespace} ;

{separator} printf("Separator: %s\n",yytext);

{operator}  printf("Operator: %s\n",yytext);

{reserved} printf( "A reserved word: %s\n", yytext);

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

{integer} printf("Constant number: %s\n",yytext);

{constchar} printf("Constant char: %s\n",yytext);

{conststring} printf("Constant string: %s\n",yytext);

{error} printf("Error: %s\n", yytext);


%%


int main( argc, argv )

int argc;

char **argv;

{

    ++argv, --argc;

   if ( argc > 0 )

   yyin = fopen( argv[0], "r" );

   else

    yyin = stdin;

   yylex();

}


DECLARE a integer;

DECLARE b integer;

READ(a);

READ(b);
```

WHILE (a != b)

{

IF (a > b)

{

a = a - b;

}

ELSE

{

b = b - a;

}

}

WRITE(a);

Output:

```
antoniosuciu@DESKTOP-9P5Q7EA:/mnt/d/FACULTATE/AN3SEM1/FLCD/Labs$ cd Lab\ 9\ lex/
antoniosuciu@DESKTOP-9P5Q7EA:/mnt/d/FACULTATE/AN3SEM1/FLCD/Labs/Lab 9 lex$ flex -o lang.c lang.l
antoniosuciu@DESKTOP-9P5Q7EA:/mnt/d/FACULTATE/AN3SEM1/FLCD/Labs/Lab 9 lex$ gcc -o lang lang.c
antoniosuciu@DESKTOP-9P5Q7EA:/mnt/d/FACULTATE/AN3SEM1/FLCD/Labs/Lab 9 lex$ ./lang p1.txt
A reserved word: DECLARE
Identifier: a
Identifier: integer
Separator: ;
A reserved word: DECLARE
Identifier: b
Identifier: integer
Separator: ;
A reserved word: READ
Separator: (
Identifier: a
Separator: )
Separator: ;
A reserved word: READ
Separator: (
Identifier: b
Separator: )
Separator: ;
A reserved word: WHILE
Separator: (
Identifier: a
Operator: !=
Identifier: b
Separator: )
Separator: {
Identifier: IF
Separator: (
Identifier: a
Operator: >
Identifier: b
Separator: )
Separator: {
Identifier: a
```