***Lab 08 documentation***

Github link: <https://github.com/timoteicopaciu/LFCD/tree/main/Lab_08>

**Lang.lxi**

%{

    #include <stdio.h>

    int lineNumber = 1;

    int correct = 1;

    int badLine = 0;

%}

%option noyywrap

digit               [0-9]

nonZeroDigit        [1-9]

letter              [a-zA-Z]

character\_constant  [']([a-zA-Z\_?! ])\*[']

numerical\_constant  [-]?{nonZeroDigit}{digit}\*|0

constant            {character\_constant}|{numerical\_constant}

identifier          {letter}|{letter}({letter}|{digit}|\_)\*?

%%

"main" {printf( "%s - reserved word\n", yytext );}

"define" {printf( "%s - reserved word\n", yytext );}

"Integer" {printf( "%s - reserved word\n", yytext );}

"Char" {printf( "%s - reserved word\n", yytext );}

"while" {printf( "%s - reserved word\n", yytext );}

"for" {printf( "%s - reserved word\n", yytext );}

"if" {printf( "%s - reserved word\n", yytext );}

"else" {printf( "%s - reserved word\n", yytext );}

"in.Integer" {printf( "%s - reserved word\n", yytext );}

"in.Chars" {printf( "%s - reserved word\n", yytext );}

"out" {printf( "%s - reserved word\n", yytext );}

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

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

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

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

"{" {printf("%s - as separator\n", yytext);}

"}" {printf("%s - as separator\n", yytext);}

"(" {printf("%s - as separator\n", yytext);}

")" {printf("%s - as separator\n", yytext);}

";" {printf("%s - as separator\n", yytext);}

"," {printf("%s - as separator\n", yytext);}

"+" {printf("%s - as operator\n", yytext);}

"-" {printf("%s - as operator\n", yytext);}

"\*" {printf("%s - as operator\n", yytext);}

"/" {printf("%s - as operator\n", yytext);}

"%" {printf("%s - as operator\n", yytext);}

">>" {printf("%s - as operator\n", yytext);}

"<=" {printf("%s - as operator\n", yytext);}

">=" {printf("%s - as operator\n", yytext);}

"==" {printf("%s - as operator\n", yytext);}

"!=" {printf("%s - as operator\n", yytext);}

"=" {printf("%s - as operator\n", yytext);}

"<" {printf("%s - as operator\n", yytext);}

">" {printf("%s - as operator\n", yytext);}

[ \t]+      {}

[\n]+   {++lineNumber;}

. {correct = 0; badLine = lineNumber; printf("Incorrect:%s\n", yytext);}

%%

void main(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();

if(correct == 1)

    printf("Correct program!");

else

    printf("Incorrect program at line %d!", badLine);

}

**Demo:**

Run output for p1.txt

**p1.txt**

main{

define Integer x , y , copy\_x , p ;

y = 0;

p = 1;

in.Integer>> x;

copy\_x = x;

while(x != -0){

y = y + (x % 10) \* p;

p = p \* 10;

x = x / 10;

}

if(y == copy\_x){

out('The\_integer\_is\_palindrome!');

}

else{

out('The\_integer\_is\_not\_palindrome!');

}

}

**Output:**

main - reserved word

{ - as separator

define - reserved word

Integer - reserved word

Identifier: x

, - as separator

Identifier: y

, - as separator

Identifier: copy\_x

, - as separator

Identifier: p

; - as separator

Identifier: y

= - as operator

Constant: 0

; - as separator

Identifier: p

= - as operator

Constant: 1

; - as separator

in.Integer - reserved word

>> - as operator

Identifier: x

; - as separator

Identifier: copy\_x

= - as operator

Identifier: x

; - as separator

while - reserved word

( - as separator

Identifier: x

!= - as operator

- - as operator

Constant: 0

) - as separator

{ - as separator

Identifier: y

= - as operator

Identifier: y

+ - as operator

( - as separator

Identifier: x

% - as operator

Constant: 10

) - as separator

\* - as operator

Identifier: p

; - as separator

Identifier: p

= - as operator

Identifier: p

\* - as operator

Constant: 10

; - as separator

Identifier: x

= - as operator

Identifier: x

/ - as operator

Constant: 10

; - as separator

} - as separator

if - reserved word

( - as separator

Identifier: y

== - as operator

Identifier: copy\_x

) - as separator

{ - as separator

out - reserved word

( - as separator

Constant: 'The\_integer\_is\_palindrome!'

) - as separator

; - as separator

} - as separator

else - reserved word

{ - as separator

out - reserved word

( - as separator

Constant: 'The\_integer\_is\_not\_palindrome!'

) - as separator

; - as separator

} - as separator

} - as separator

Correct program!