Documentation

Github link: https://github.com/Gherghel-Vlad/FLCD/tree/main/Lab8

The flex specification:

%{

#include <math.h>

#include <stdlib.h>

#include <string.h>

%}

%option noyywrap

%option yylineno

digit [0-9]

nndigit [1-9]

letter [a-zA-Z\_]

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

number [+-]?{digit}|{nndigit}{digit}\*

BOOLEAN\_FORMAT (true|false)

CHAR\_FORMAT (\'.\'?)

STRING\_FORMAT (".\*")

CONSTANT\_FORMAT ({BOOLEAN\_FORMAT}|{CHAR\_FORMAT}|{STRING\_FORMAT}|{number})

%{

int idsCount = 0, constsCount = 0, line = 1;

char ids[100][100];

char consts[100][100];

void printIdConst(char\* word, char dict[][100], int\* count) {

int i, nr = -1;

for (i = 0; i < \*count; i++) {

if (strcmp(word, dict[i]) == 0) {

nr = i;

break;

}

}

if (nr == -1) {

strcpy(dict[\*count], word);

\*count = \*count + 1;

nr = \*count - 1;

}

printf("%s | %d \n", word, nr);

}

%}

%%

for {printf("%s | -1\n", yytext);}

if {printf("%s | -1\n", yytext);}

in {printf("%s | -1\n", yytext);}

input {printf("%s | -1\n", yytext);}

print {printf("%s | -1\n", yytext);}

else {printf("%s | -1\n", yytext);}

while {printf("%s | -1\n", yytext);}

do {printf("%s | -1\n", yytext);}

int {printf("%s | -1\n", yytext);}

char {printf("%s | -1\n", yytext);}

string {printf("%s | -1\n", yytext);}

bool {printf("%s | -1\n", yytext);}

list {printf("%s | -1\n", yytext);}

"..." {printf("%s | -1\n", yytext);}

"<=>" {printf("%s | -1\n", yytext);}

"==" {printf("%s | -1\n", yytext);}

"<=" {printf("%s | -1\n", yytext);}

">=" {printf("%s | -1\n", yytext);}

"!=" {printf("%s | -1\n", yytext);}

and {printf("%s | -1\n", yytext);}

or {printf("%s | -1\n", yytext);}

"+=" {printf("%s | -1\n", yytext);}

"-"= {printf("%s | -1\n", yytext);}

"\*=" {printf("%s | -1\n", yytext);}

"/=" {printf("%s | -1\n", yytext);}

">" {printf("%s | -1\n", yytext);}

"<" {printf("%s | -1\n", yytext);}

"[" {printf("%s | -1\n", yytext);}

"]" {printf("%s | -1\n", yytext);}

"{" {printf("%s | -1\n", yytext);}

"}" {printf("%s | -1\n", yytext);}

";" {printf("%s | -1\n", yytext);}

":" {printf("%s | -1\n", yytext);}

"(" {printf("%s | -1\n", yytext);}

")" {printf("%s | -1\n", yytext);}

"," {printf("%s | -1\n", yytext);}

"+" {printf("%s | -1\n", yytext);}

"-" {printf("%s | -1\n", yytext);}

"\*" {printf("%s | -1\n", yytext);}

"/" {printf("%s | -1\n", yytext);}

"=" {printf("%s | -1\n", yytext);}

"%" {printf("%s | -1\n", yytext);}

"\"" {printf("%s | -1\n", yytext);}

"'" {printf("%s | -1\n", yytext);}

{identifier} {printIdConst(yytext, ids, &idsCount);}

{CONSTANT\_FORMAT} {printIdConst(yytext, consts, &constsCount);}

[ \t] ;

\n {line++;}

\r\n {line++;}

. {printf("ERROR for %s on line %d\n", yytext, line);}

%%

int lexic\_main(char\* filename)

{

yyin = fopen(filename, "r");

if(!yyin) {

return 1;

}

printf("TOKEN | ST\_POS \n");

yylex();

fclose(yyin);

printf("\nST\n");

printf("ID\n");

int j = 0;

for(j = 0; j < idsCount; j++) {

printf("%s %d\n",ids[j],j);

}

printf("CONST\n");

for(j = 0; j < constsCount; j++) {

printf("%s %d\n",consts[j],j);

}

return 0;

}

For p1.txt:

a = 0;

b = 0;

c = 0;

input(a);

input(b);

input(c);

mx = 0;

if (a > mx) {

mx = a;

}

if (b > mx) {

mx = b;

}

if (c > mx) {

mx = c;

}

print(mx);

The result of running the scanner on it is:

TOKEN | ST\_POS

a | 0

= | -1

0 | 0

; | -1

b | 1

= | -1

0 | 0

; | -1

c | 2

= | -1

0 | 0

; | -1

input | -1

( | -1

a | 0

) | -1

; | -1

input | -1

( | -1

b | 1

) | -1

; | -1

input | -1

( | -1

c | 2

) | -1

; | -1

mx | 3

= | -1

0 | 0

; | -1

if | -1

( | -1

a | 0

> | -1

mx | 3

) | -1

{ | -1

mx | 3

= | -1

a | 0

; | -1

} | -1

if | -1

( | -1

b | 1

> | -1

mx | 3

) | -1

{ | -1

mx | 3

= | -1

b | 1

; | -1

} | -1

if | -1

( | -1

c | 2

> | -1

mx | 3

) | -1

{ | -1

mx | 3

= | -1

c | 2

; | -1

} | -1

print | -1

( | -1

mx | 3

) | -1

; | -1

ST

ID

a 0

b 1

c 2

mx 3

CONST

0 0