Laboratory 8 – FLCD – LEX

Moldovan Vasilica, Group 935/1

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
int no_lines = 0;
%}

%option noyywrap

DIGIT     [0-9]
WORD      ["]([a-zA-Z])*["]
NUMBER    [1-9][0-9]*|0
NR_REAL       {NUMBER}+"."{DIGIT}*
CST           {WORD}|{NUMBER}|{NR_REAL}
ID          [a-zA-Z][a-zA-Z0-9]{0,9}

%%

"individual"   {printf( "Reserved word: %s\n", yytext );}
"decision"     {printf( "Reserved word: %s\n", yytext );}
"char"         {printf( "Reserved word: %s\n", yytext );}
"float"            {printf( "Reserved word: %s\n", yytext );}
"const"        {printf( "Reserved word: %s\n", yytext );}
"parsing"      {printf( "Reserved word: %s\n", yytext );}
"situation"    {printf( "Reserved word: %s\n", yytext );}
"other"        {printf( "Reserved word: %s\n", yytext );}
"come"         {printf( "Reserved word: %s\n", yytext );}
"leave"        {printf( "Reserved word: %s\n", yytext );}
"return"       {printf( "Reserved word: %s\n", yytext );}
"break"        {printf( "Reserved word: %s\n", yytext );}


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

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

":"            {printf( "Separator: %s\n", yytext );}
";"                {printf( "Separator: %s\n", yytext );}
","                {printf( "Separator: %s\n", yytext );}
"."            {printf( "Separator: %s\n", yytext );}
"+"                {printf( "Operator: %s\n", yytext );}
"-"                {printf( "Operator: %s\n", yytext );}
"*"                {printf( "Operator: %s\n", yytext );}
"/"                {printf( "Operator: %s\n", yytext );}
"("                {printf( "Separator: %s\n", yytext );}
")"                {printf( "Separator: %s\n", yytext );}
"["                {printf( "Separator: %s\n", yytext );}
"]"                {printf( "Separator: %s\n", yytext );}
```

```
"<"                {printf( "Operator: %s\n", yytext );}
">"                {printf( "Operator: %s\n", yytext );}
"<="           {printf( "Operator: %s\n", yytext );}
">="           {printf( "Operator: %s\n", yytext );}
"!="           {printf( "Operator: %s\n", yytext );}
"="                {printf( "Operator: %s\n", yytext );}
"=="           {printf( "Separator: %s\n", yytext );}
"||"           {printf( "Separator: %s\n", yytext );}
"&&"           {printf( "Separator: %s\n", yytext );}
"{"                {printf( "Separator: %s\n", yytext );}
"}"                {printf( "Separator: %s\n", yytext );}

["leave" WORD]+         {printf("Printing string: %s\n", yytext);}

[ \t]+                {}

[\n]+  {++no_lines;}

[0-9][a-zA-Z0-9]{0,7}  {printf("Illegal identifier at line %d\n", no_lines);
return -1;}

.                      {printf("Illegal symbol at line %d\n", no_lines);
return -1;}
%%

int main()
{
yylex();
printf("Done");
}
```

P2(No error):

Reserved word: individual

Separator: [

Constant: 100

Separator:]

Printing string: a

Identifier: rr

Separator: ;

Reserved word: decision

Printing string:

Identifier: isSmaller

Printing string:

Operator: =

Printing string:

Identifier: true

Separator: ;

Reserved word: individual

Printing string:

Identifier: n

Separator: ,

Printing string:

Identifier: i

Separator: ;

Reserved word: individual

Printing string:

Identifier: maxNumber

Separator: ;

Reserved word: come

Printing string:

Identifier: n

Separator: ;

Reserved word: come

Printing string:

Identifier: m

Separator: ;

Identifier: i

Printing string:

Operator: =

Printing string:

Constant: 0

Separator: ;

Reserved word: parsing

Printing string:

Separator: (

Printing string:

Identifier: i

Printing string:

Operator: <

Printing string:

Identifier: n

Printing string:

Separator:)

Printing string:

Separator: {

Reserved word: come

Printing string: a

Identifier: rr

Separator: [

Identifier: i

Separator:]

Separator: ;

Reserved word: situation

Printing string:

Separator: (

Identifier: i

Printing string:

Separator: ==

Printing string:

Constant: 0

Separator:)

Printing string:

Identifier: maxNumber

Printing string:

Operator: =

Printing string: a

Identifier: rr

Separator: [

Identifier: i

Separator:]

Separator: ;

Reserved word: other

Printing string:

Separator: {

Reserved word: situation

Printing string:

Separator: (

Printing string:

Identifier: maxNumber

Printing string:

Operator: <

Printing string: a

Identifier: rr

Separator: [

Identifier: i

Separator:]

Printing string:

Separator:)

Identifier: maxNumber

Printing string:

Operator: =

Printing string: a

Identifier: rr

Separator: [

Identifier: i

Separator:]

Separator: ;

Separator: }

Identifier: i

Printing string:

Operator: =

Printing string:

Identifier: i

Printing string:

Operator: +

Printing string:

Constant: 1

Separator: ;

Separator: }

Reserved word: situation

Printing string:

Separator: (

Printing string:

Identifier: maxNumber

Printing string:

Operator: >=

Printing string:

Identifier: m

Printing string:

Separator:)

Identifier: isSmaller

Printing string:

Operator: =

Printing string:

Identifier: false

Separator: ;

Printing string: leave "

Identifier: The

Printing string:

Identifier: maximum

Printing string:

Identifier: number

Printing string:

Identifier: is

Printing string: "

Separator: ;

Printing string: leave

Identifier: maxNumber

Separator: ;

Printing string: leave " a

Identifier: nd

Printing string:

Identifier: is

Printing string:

Identifier: smaller

Printing string:

Identifier: than

Printing string: "

Separator: ;

Printing string: leave

Identifier: m

Separator: ;

Printing string: leave "

Operator: -

Printing string: "

Separator: ;

Printing string: leave

Identifier: isSmaller

Separator: ;

Done

P3(Same program as p2, but with error):

Reserved word: individual

Printing string: a

Identifier: rr

Separator: [

Constant: 100

Separator:]

Separator: ;

Reserved word: decision

Printing string:

Identifier: isSmaller

Printing string:

Operator: =

Printing string:

Identifier: true

Separator: ;

Reserved word: individual

Printing string:

Identifier: n

Separator: ,

Printing string:

Identifier: i

Separator: ;

Reserved word: individual

Printing string:

Identifier: maxNumber

Separator: ;

Reserved word: come

Printing string:

Identifier: n

Separator: ;

Reserved word: come

Printing string:

Identifier: m

Separator: ;

Identifier: i

Operator: =

Constant: 0

Reserved word: parsing

Printing string:

Separator: (

Printing string:

Identifier: i

Printing string:

Operator: <

Printing string:

Identifier: n

Printing string:

Separator:)

Printing string:

Separator: {

Reserved word: come

Printing string: a

Identifier: rr

Separator: [

Identifier: i

Separator:]

Separator: ;

Reserved word: situation

Printing string:

Separator: (

Identifier: i

Printing string:

Separator: ==

Printing string:

Constant: 0

Separator:)

Identifier: maxNumber

Printing string:

Operator: =

Printing string: a

Identifier: rr

Separator: [

Illegal identifier at line 10

Done