Lex-Yacc Laboratory

Popa Alex Ovidiu, 936/1

https://github.com/alexovidiupopa/flcd/tree/main/lex-yacc

Commands:

lex specif.lxi

gcc lex.yy.c -o exe –ll

./exe < p1.txt

Specif.lxi:

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

%option noyywrap
%option caseless

DIGIT           [0-9]
WORD            \"[a-zA-Z0-9]*\"
NUMBER                  [+-]?[1-9][0-9]*|0$
CHARACTER       \'[a-zA-Z0-9]\'
CONST           {WORD}|{NUMBER}|{CHARACTER}
ID              [a-zA-Z][a-zA-Z0-9_]{0,7}

%%
```

```
and {printf("Reserved word: %s\n", yytext);}
array    {printf( "Reserved word: %s\n", yytext);}
else     {printf( "Reserved word: %s\n", yytext);}
for      {printf( "Reserved word: %s\n", yytext);}
go       {printf( "Reserved word: %s\n", yytext);}
if       {printf( "Reserved word: %s\n", yytext);}
number          {printf( "Reserved word: %s\n", yytext);}
or       {printf( "Reserved word: %s\n", yytext);}
cin      {printf( "Reserved word: %s\n", yytext);}
cout     {printf( "Reserved word: %s\n", yytext);}
string   {printf( "Reserved word: %s\n", yytext);}
while    {printf( "Reserved word: %s\n", yytext);}
xor      {printf( "Reserved word: %s\n", yytext);}

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

{CONST}          {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( "Separator: %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 );}
"+"       {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( "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( "Operator: %s\n", yytext );}
">>"      {printf( "Operator: %s\n", yytext );}
"<<"      {printf( "Operator: %s\n", yytext );}



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

[+-]?0[0-9]* {printf("Illegal constant at line %d\n", lines);}

[a-zA-Z][a-zA-Z0-9]{8,}{printf("Illegal size of the identifier at line %d\n", lines);}

[0-9~@#$%^][a-zA-Z0-9]{0,7}{printf("Illegal identifier at line %d\n", lines);}
```

```
\'[a-zA-Z0-9]{2,}\' {printf("Character of length >=2 at line %d\n", lines);}

%%
```

P1.txt

```
go{
number a;
number b;
number c;
a=+2;
a=-10;
b=+0;
c=154;
number bCBCA123;
number acb;
number 123abc;
cin>>a;
cin>>b;
cin>>c;
number max ;
if(a>b and a>c){
max = a;
}
if(b>a and b>c){
```

```
max=b;
}
if(c>a and c>b){
max=c;
}
cout<<max;
cout<<"ok";
cout<<"alex;
cout<<'asd';
}
```