

<https://github.com/RavaszTamas/FormalLanguagesLaboratories/tree/main/lab8>

%{

#include <math.h>

#include <stdio.h>

#define ID 1

#define CONST 2

#define osszead 3

#define kivon 4

#define szoroz 5

#define oszt 6

#define modulo 7

#define kisebb 8

#define kisebbvagyegyenlo 9

#define egyenlo 10

#define nagyobbvagyegyenlo 11

#define nagyobb 12

#define nemegyenlo 13

#define novel 14

#define csokken 15

#define kapja 16

#define ha 17

#define kulonben 18

#define karakter 19

#define karakterlanc 20

#define amig 21

#define boolean 22

#define egesz 23

#define tomb 24

#define dupla 25

```
#define visszater 26

#define ismeteld 27

#define Kezd 28

#define Vegez 29

#define allj 30

#define valassz 31

#define eset 32

#define alapertelmezett 33

#define konstans 34

#define open_square_bracket 35

#define closed_square_bracket 36

#define open_curly_bracket 37

#define closed_curly_bracket 38

#define open_bracket 39

#define closed_bracket 40

#define semicolon 41

#define coma 42

#define colon 43


int current_line = 0;


%}


%option noyywrap


DIGIT      [0-9]
STRING     ["].*["]
INTEGER     [--]?[1-9][0-9]*|0
REAL_NUMBER {INTEGER}+"."{DIGIT}*
```

CONSTANT_VALUE {STRING}|{INTEGER}|{REAL_NUMBER}

IDENTIFIER [a-zA-Z][a-zA-Z0-9_]{0,256}

%%

"osszead" {printf("Reserved word: %s\n", yytext);return összead;}

"kivon" {printf("Reserved word: %s\n", yytext);return kivon;}

"szoroz" {printf("Reserved word: %s\n", yytext);return szoroz;}

"oszt" {printf("Reserved word: %s\n", yytext);return oszt;}

"modulo" {printf("Reserved word: %s\n", yytext);return modulo;}

"kisebb" {printf("Reserved word: %s\n", yytext);return kisebb;}

"kisebbvagyeegyenlo" {printf("Reserved word: %s\n", yytext);return kisebbvagyeegyenlo;}

"egyenlo" {printf("Reserved word: %s\n", yytext);return egyenlo;}

"nagyobbvagyegyenlo" {printf("Reserved word: %s\n", yytext);return nagyobbvagyegyenlo;}

"nagyobb" {printf("Reserved word: %s\n", yytext);return nagyobb;}

"nemegyenlo" {printf("Reserved word: %s\n", yytext);return nemegyenlo;}

"novel" {printf("Reserved word: %s\n", yytext);return novel;}

"csokken" {printf("Reserved word: %s\n", yytext);return csokken;}

"kapja" {printf("Reserved word: %s\n", yytext);return kapja;}

"ha" {printf("Reserved word: %s\n", yytext);return ha;}

"kulonben" {printf("Reserved word: %s\n", yytext);return kulonben;}

"karakter" {printf("Reserved word: %s\n", yytext);return karakter;}

"karakterlanc" {printf("Reserved word: %s\n", yytext);return karakterlanc;}

"amig" {printf("Reserved word: %s\n", yytext);return amig;}

"boolean" {printf("Reserved word: %s\n", yytext);return boolean;}

"egesz" {printf("Reserved word: %s\n", yytext);return egesz;}

"tomb" {printf("Reserved word: %s\n", yytext);return tomb;}

"dupla" {printf("Reserved word: %s\n", yytext);return dupla;}

"visszater" {printf("Reserved word: %s\n", yytext);return visszater;}

```

"ismeteld"    {printf( "Reserved word: %s\n", yytext );return ismeteld;}
"Kezd" {printf( "Reserved word: %s\n", yytext );return Kezd;}
"Vegez" {printf( "Reserved word: %s\n", yytext );return Vegez;}
"allj"    {printf( "Reserved word: %s\n", yytext );return allj;}
"valassz"    {printf( "Reserved word: %s\n", yytext );return valassz;}
"eset" {printf( "Reserved word: %s\n", yytext );return eset;}
"alapertelmezett"    {printf( "Reserved word: %s\n", yytext ); return alapertelmezett;}
"konstans"    {printf( "Reserved word: %s\n", yytext ); return konstans;}
{IDENTIFIER}    {printf( "Identifier: %s\n", yytext ); return ID;}
{CONSTANT_VALUE}    {printf( "Constant: %s\n", yytext ); return CONST;}
"["    {printf( "Separator: %s\n", yytext );return open_square_bracket;}
"]"    {printf( "Separator: %s\n", yytext );return closed_square_bracket;}
"{"    {printf( "Separator: %s\n", yytext );return open_curly_bracket;}
"}"    {printf( "Separator: %s\n", yytext );return closed_curly_bracket;}
"("    {printf( "Separator: %s\n", yytext );return open_bracket;}
")"    {printf( "Separator: %s\n", yytext );return closed_bracket;}
";"    {printf( "Separator: %s\n", yytext );return semicolon;}
","    {printf( "Separator: %s\n", yytext );return coma;}
":"    {printf( "Separator: %s\n", yytext );return colon;}
"{"[^\\n]*"}"    /* eliminate the comments in the code*/ {}
[ \\t]+    /* eliminate the spaces in the code */ {}
[\\n]+    {++current_line;}
[a-zA-Z][a-zA-Z0-9]{256,}    {printf("Illegal size of the identifier %s at line %d\n",yytext,
current_line); return -1;}
[0-9][a-zA-Z0-9]{0,256} {printf("Illegal identifier %s at line %d\n",yytext, current_line); return -1;}
.    {printf("Illegal symbol %s at line %d\n",yytext,current_line); return -1;}
%%

```

```

int main(int argc, char **argv)

```

```

{
    ++argv;
    --argc; /* skip over program name */
    if ( argc > 0 )
        yyin = fopen( argv[0], "r" );
    else
        yyin = stdin;

    int tokenid;

    tokenid = yylex();
    while(tokenid)
    {
        printf("Obtained token %d\n",tokenid);
        if(tokenid == -1)
        {
            return -1;
        }
        tokenid = yylex();
    }
    return 0;
}

```

p3.txt

Kezd

```

    egesz a,b,c;

    olvas(a);

    olvas(b);

    amig ( a nemegyenlo 0 ){
        ures kapja second_number;
    }

```

```
        second_number kapja first_number modulo second_number;  
        first_number kapja ures  
};
```

```
ha(a egyenlo 1){  
    kiir("relative primes")  
}  
kulonben{  
    kiir("not relative primes")  
}
```

Vegez

./my_lex < p3.txt

Reserved word: Kezd

Obtained token 28

Reserved word: egesz

Obtained token 23

Identifier: a

Obtained token 1

Separator: ,

Obtained token 42

Identifier: b

Obtained token 1

Separator: ,

Obtained token 42

Identifier: c

Obtained token 1

Separator: ;

Obtained token 41

Identifier: olvas

Obtained token 1

Separator: (

Obtained token 39

Identifier: a

Obtained token 1

Separator:)

Obtained token 40

Separator: ;

Obtained token 41

Identifier: olvas

Obtained token 1

Separator: (

Obtained token 39

Identifier: b

Obtained token 1

Separator:)

Obtained token 40

Separator: ;

Obtained token 41

Reserved word: amig

Obtained token 21

Separator: (

Obtained token 39

Identifier: a

Obtained token 1

Reserved word: nemegyenlo

Obtained token 13

Constant: 0

Obtained token 2

Separator:)

Obtained token 40

Separator: {

Obtained token 37

Identifier: ures

Obtained token 1

Reserved word: kapja

Obtained token 16

Identifier: second_number

Obtained token 1

Separator: ;

Obtained token 41

Identifier: second_number

Obtained token 1

Reserved word: kapja

Obtained token 16

Identifier: first_number

Obtained token 1

Reserved word: modulo

Obtained token 7

Identifier: second_number

Obtained token 1

Separator: ;

Obtained token 41

Identifier: first_number

Obtained token 1

Reserved word: kapja

Obtained token 16

Identifier: ures

Obtained token 1

Separator: }

Obtained token 38

Separator: ;

Obtained token 41

Reserved word: ha

Obtained token 17

Separator: (

Obtained token 39

Identifier: a

Obtained token 1

Reserved word: egyenlo

Obtained token 10

Constant: 1

Obtained token 2

Separator:)

Obtained token 40

Separator: {

Obtained token 37

Identifier: kiir

Obtained token 1

Separator: (

Obtained token 39

Constant: "relative primes"

Obtained token 2

Separator:)

Obtained token 40

Separator: }

Obtained token 38

Reserved word: kulonben

Obtained token 18

Separator: {

Obtained token 37

Identifier: kiir

Obtained token 1

Separator: (

Obtained token 39

Constant: "not relative primes"

Obtained token 2

Separator:)

Obtained token 40

Separator: }

Obtained token 38

Reserved word: Vegez

Obtained token 29