https://github.com/cristiifrim/Formal-Languages-and-Compiler-Design/tree/master/Lab9

prog.l

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "parser.tab.h"
int lines = 1;
%}

%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}|{DIGIT}
id              [a-zA-Z][a-zA-Z0-9_]{0,7}

%%

and {printf("Reserved word: %s\n", yytext); return AND;}
array {printf( "Reserved word: %s\n", yytext); return ARRAY;}
else {printf( "Reserved word: %s\n", yytext); return ELSE;}
for {printf( "Reserved word: %s\n", yytext); return FOR;}
go {printf( "Reserved word: %s\n", yytext); return GO;}
if {printf( "Reserved word: %s\n", yytext); return IF;}
```

```
number {printf( "Reserved word: %s\n", yytext); return NUMBER;}
or {printf( "Reserved word: %s\n", yytext); return OR;}
cin {printf( "Reserved word: %s\n", yytext); return CIN;}
cout {printf( "Reserved word: %s\n", yytext); return COUT;}
string {printf( "Reserved word: %s\n", yytext); return STRING;}
while {printf( "Reserved word: %s\n", yytext); return WHILE;}
xor {printf( "Reserved word: %s\n", yytext); return XOR;}

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

{const} {printf( "Constant: %s\n", yytext ); return CONST;}

":" {printf( "Separator: %s\n", yytext ); return COLON;}
";" {printf( "Separator: %s\n", yytext ); return SEMI_COLON;}
"," {printf( "Separator: %s\n", yytext ); return COMMA;}
"." {printf( "Separator: %s\n", yytext ); return DOT;}
"{" {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_ROUND_BRACKET;}
")" {printf( "Separator: %s\n", yytext ); return CLOSED_ROUND_BRACKET;}
"[" {printf( "Separator: %s\n", yytext ); return OPEN_RIGHT_BRACKET;}
"]" {printf( "Separator: %s\n", yytext ); return CLOSED_RIGHT_BRACKET;}
"=" {printf( "Separator: %s\n", yytext ); return ATRIB;}

"+" {printf( "Operator: %s\n", yytext ); return PLUS;}
"-" {printf( "Operator: %s\n", yytext ); return MINUS;}
"*" {printf( "Operator: %s\n", yytext ); return MUL;}
"/" {printf( "Operator: %s\n", yytext ); return DIV;}
"<" {printf( "Operator: %s\n", yytext ); return LT;}
```

```
">" {printf( "Operator: %s\n", yytext ); return GT;}
"<=" {printf( "Operator: %s\n", yytext ); return LE;}
">=" {printf( "Operator: %s\n", yytext ); return GE;}
"!=" {printf( "Operator: %s\n", yytext ); return NE;}
"==" {printf( "Operator: %s\n", yytext ); return EQ;}
"!" {printf( "Operator: %s\n", yytext ); return NOT;}
">>" {printf( "Operator: %s\n", yytext ); return READ_OP;}
"<<" {printf( "Operator: %s\n", yytext ); return WRITE_OP;}



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

[+-]?0[0-9]+ {printf("Illegal constant: %s at line %d\n", yytext, lines); exit(1);}

[a-zA-Z][a-zA-Z0-9]{8,}    {printf("Illegal size of the identifier at line %d\n", lines); exit(1);}

[0-9~@#$%^][a-zA-Z0-9]*          {printf("Illegal identifier %s at line %d\n", yytext, lines); exit(1);}

\'[a-zA-Z0-9]{2,}\' {printf("Character of length >=2 at line %d\n", lines); exit(1);}

. {printf("Illegal character at line %d\n", lines); exit(1);}
%%
```

parser.y

```
%{
#include <stdio.h>
#include <stdlib.h>

int yyerror(char *s);
```

```
#define YYDEBUG 1
%}

%token AND
%token ARRAY
%token ELSE
%token FOR
%token GO
%token IF
%token NUMBER
%token OR
%token CIN
%token COUT
%token STRING
%token WHILE
%token XOR

%token ID
%token CONST

%token ATRIB
%token EQ
%token NE
%token LE
%token GE
%token LT
%token GT
%token NOT

%token DOT

%left '+' '-' '*' '/'

%token PLUS
%token MINUS
%token DIV
%token MOD
%token MUL

%token OPEN_CURLY_BRACKET
%token CLOSED_CURLY_BRACKET
%token OPEN_ROUND_BRACKET
%token CLOSED_ROUND_BRACKET
%token OPEN_RIGHT_BRACKET
%token CLOSED_RIGHT_BRACKET
```

```
%token READ_OP
%token WRITE_OP

%token COMMA
%token SEMI_COLON
%token COLON
%token SPACE

%start program

%%
program : GO cmpdstmt
          ;
declaration :  type arrayDecl ID
              ;
type :  NUMBER | STRING
             ;
arrayDecl : /*Empty*/ | ARRAY OPEN_RIGHT_BRACKET CONST CLOSED_RIGHT_BRACKET
          ;
cmpdstmt : OPEN_CURLY_BRACKET stmtlist CLOSED_CURLY_BRACKET
          ;
stmtlist :  stmt stmtTemp
          ;
stmtTemp : /*Empty*/ | stmtlist
          ;
stmt :  simplstmt SEMI_COLON | structstmt
    ;
simplstmt :  assignstmt | iostmt | declaration
          ;
structstmt :  cmpdstmt | ifstmt | whilestmt | forstmt
             ;
ifstmt :  IF OPEN_ROUND_BRACKET boolean_condition CLOSED_ROUND_BRACKET cmpdstmt tempIf
      ;
tempIf : /*Empty*/ | ELSE cmpdstmt
      ;
forstmt :  FOR forheader cmpdstmt
      ;
forheader :  OPEN_ROUND_BRACKET assignstmt SEMI_COLON boolean_condition SEMI_COLON
assignstmt CLOSED_ROUND_BRACKET
           ;
whilestmt :  WHILE OPEN_ROUND_BRACKET boolean_condition CLOSED_ROUND_BRACKET cmpdstmt
           ;
assignstmt :  ID ATRIB expression
            ;
```

```
expression : arithmetic2 arithmetic1
            ;
arithmetic1 : PLUS arithmetic2 arithmetic1 | MINUS arithmetic2 arithmetic1 | /*Empty*/
             ;
arithmetic2 : multiply2 multiply1
             ;
multiply1 : MUL multiply2 multiply1 | DIV multiply2 multiply1 | /*Empty*/
            ;
multiply2 : OPEN_ROUND_BRACKET expression CLOSED_ROUND_BRACKET | CONST | ID |
IndexedIdentifier
            ;
IndexedIdentifier :  ID OPEN_RIGHT_BRACKET expression CLOSED_RIGHT_BRACKET
                        ;
iostmt :  CIN READ_OP ID idTemp | COUT WRITE_OP ID idTemp | COUT WRITE_OP CONST
       ;
idTemp: /*Empty*/ | OPEN_RIGHT_BRACKET expression CLOSED_RIGHT_BRACKET
    ;
condition : expression GT expression |
            expression GE expression |
            expression LT expression |
            expression LE expression |
            expression EQ expression |
            expression NE expression
             ;
boolean_condition : condition boolean_cond_temp
                        ;
boolean_cond_temp : /*Empty*/ | AND boolean_condition | OR boolean_condition | XOR
boolean_condition
                       ;
%%

int yyerror(char *s)
{
          printf("Error: %s\n",s);
}

extern FILE *yyin;

int main(int argc, char** argv) {
   if (argc > 1)
   {
      yyin = fopen(argv[1], "r");
   }
   if(!yyparse())
   {
```

```
    fprintf(stderr, "\tOK\n");
  }
  return 0;
}
```

P1.txt

```
go {
  number a;
  number b;
  number c;
  a=+32;
  a=-15;
  b=1;
  c=154;
  number d;
  number acb;
  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;
```

```
}
```

Output

```
Reserved word: go

Separator: {

Reserved word: number

Identifier: a

Separator: ;

Reserved word: number

Identifier: b

Separator: ;

Reserved word: number

Identifier: c

Separator: ;

Identifier: a

Separator: =

Constant: +32

Separator: ;

Identifier: a

Separator: =

Constant: -15

Separator: ;

Identifier: b

Separator: =

Constant: 1

Separator: ;

Identifier: c

Separator: =
```

Constant: 154

Separator: ;

Reserved word: number

Identifier: d

Separator: ;

Reserved word: number

Identifier: acb

Separator: ;

Reserved word: cin

Operator: >>

Identifier: a

Separator: ;

Reserved word: cin

Operator: >>

Identifier: b

Separator: ;

Reserved word: cin

Operator: >>

Identifier: c

Separator: ;

Reserved word: number

Identifier: max

Separator: ;

Reserved word: if

Separator: (

Identifier: a

Operator: >

Identifier: b

Reserved word: and

Identifier: a

Operator: >

Identifier: c

Separator:)

Separator: {

Identifier: max

Separator: =

Identifier: a

Separator: ;

Separator: }

Reserved word: if

Separator: (

Identifier: b

Operator: >

Identifier: a

Reserved word: and

Identifier: b

Operator: >

Identifier: c

Separator:)

Separator: {

Identifier: max

Separator: =

Identifier: b

Separator: ;

Separator: }

Reserved word: if

Separator: (

Identifier: c

Operator: >

Identifier: a

Reserved word: and

Identifier: c

Operator: >

Identifier: b

Separator:)

Separator: {

Identifier: max

Separator: =

Identifier: c

Separator: ;

Separator: }

Reserved word: cout

Operator: <<

Identifier: max

Separator: ;

Separator: }

OK

P2.txt

```
go {
   number array[10] arr;
   number size;
   cin >> size;
   number sum;
   sum=0;
   for (i=0; i < size; i=I + 1) {
      cin>>arr[i];
```

```
        if (arr[i]>0) {
            sum = sum + arr[i];
        }
    }
    cout<<sum;
}
```

output

```
Reserved word: go
Separator: {
Reserved word: number
Reserved word: array
Separator: [
Constant: 10
Separator: ]
Identifier: arr
Separator: ;
Reserved word: number
Identifier: size
Separator: ;
Reserved word: cin
Operator: >>
Identifier: size
Separator: ;
Reserved word: number
Identifier: sum
Separator: ;
Identifier: sum
Separator: =
```

Constant: 0

Separator: ;

Reserved word: for

Separator: (

Identifier: i

Separator: =

Constant: 0

Separator: ;

Identifier: i

Operator: <

Identifier: size

Separator: ;

Identifier: i

Separator: =

Identifier: i

Operator: +

Constant: 1

Separator:)

Separator: {

Reserved word: cin

Operator: >>

Identifier: arr

Separator: [

Identifier: i

Separator:]

Separator: ;

Reserved word: if

Separator: (

Identifier: arr

Separator: [

Identifier: i

Separator:]

Operator: >

Constant: 0

Separator:)

Separator: {

Identifier: sum

Separator: =

Identifier: sum

Operator: +

Identifier: arr

Separator: [

Identifier: i

Separator:]

Separator: ;

Separator: }

Separator: }

Reserved word: cout

Operator: <<

Identifier: sum

Separator: ;

Separator: }

OK

P1err.txt

```
go {
   number 5$a;
   number b;
```

```
    number c;
    cin >> 5$a;
    cin >> b;
    cin >> c;
    number max;
    if(5$a > b and 5$a > c){
        max = 5$a;
    }
    if(b > 5$a and b > c){
        max=b;
    }
    if(c > 5$a and c > b){
        max=c;
    }
    string message;
    message='number is;
    cout<<message;
    cout<<max;
}
```

Output

```
Reserved word: go
Separator: {
Reserved word: number
Constant: 5
Error: syntax error
```