https://github.com/cs-ubbcluj-ro/lab-work-computer-science-2024-NikAlien/tree/main/4-Lex-Yacc

Lab 8 – Lex/Yacc

Carp Nicoleta
Gr 931

// specific.lxi

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

%option noyywrap

IDENTIFIER          [#_a-zA-Z][a-zA-Z0-9_]*
NUMBER_CONST    [-]?[1-9]\d*|0
STRING_CONST      \"[^\n\"]*\"

%%

"int"           {printf("Reserved word: %s\n", yytext); return INT;}
"char"              {printf("Reserved word: %s\n", yytext); return CHAR;}
"if"            {printf("Reserved word: %s\n", yytext); return IF;}
"else"          {printf("Reserved word: %s\n", yytext); return ELSE;}
"not"           {printf("Reserved word: %s\n", yytext); return NOT;}
"and"           {printf("Reserved word: %s\n", yytext); return AND;}
"or"            {printf("Reserved word: %s\n", yytext); return OR;}
"cout"              {printf("Reserved word: %s\n", yytext); return COUT;}
"cin"           {printf("Reserved word: %s\n", yytext); return CIN;}
"for"           {printf("Reserved word: %s\n", yytext); return FOR;}

"+"             {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 division;}
"="             {printf("Operator %s\n", yytext); return eq;}
"<="    {printf("Operator %s\n", yytext); return lessOrEqual;}
"=="    {printf("Operator %s\n", yytext); return equal;}
">="    {printf("Operator %s\n", yytext); return moreOrEqual;}
```

```
"<"             {printf("Operator %s\n", yytext); return less;}
">"             {printf("Operator %s\n", yytext); return more;}
">>"    {printf("Operator %s\n", yytext); return rightShift;}
"<<"    {printf("Operator %s\n", yytext); return leftShift;}
"!="    {printf("Operator %s\n", yytext); return different;}
"++"    {printf("Operator %s\n", yytext); return increment;}

"{"             {printf("Separator %s\n", yytext); return leftCurlyBracket;}
"}"             {printf("Separator %s\n", yytext); return rightCurlyBracket;}
"("             {printf("Separator %s\n", yytext); return leftRoundBracket;}
")"             {printf("Separator %s\n", yytext); return rightRoundBracket;}
"["             {printf("Separator %s\n", yytext); return leftBracket;}
"]"             {printf("Separator %s\n", yytext); return rightBracket;}
","             {printf("Separator %s\n", yytext); return comma;}
"."             {printf("Separator %s\n", yytext); return period;}

{IDENTIFIER}        {printf("Identifier: %s\n", yytext); return IDENTIFIER;}
{NUMBER_CONST}          {printf("Number: %s\n", yytext); return NUMBER_CONST;}
{STRING_CONST}          {printf("String: %s\n", yytext); return STRING_CONST;}

[ \t]+                  {/* Ignore whitespace */}
[\n]+                   {currentLine++;}

.                               {printf("Unknown token %s at line %d\n", yytext, currentLine);}
%%
```

// parser.y

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

#define YYDEBUG 1
int yylex(void);
void yyerror(char *s);
%}

%token INT
%token CHAR
%token NOT
%token AND
%token OR
%token IF
```

```
%token ELSE
%token FOR
%token COUT
%token CIN

%token plus
%token minus
%token mul
%token division
%token lessOrEqual
%token moreOrEqual
%token less
%token more
%token equal
%token different
%token eq
%token rightShift
%token leftShift
%token increment



%token leftCurlyBracket
%token rightCurlyBracket
%token leftRoundBracket
%token rightRoundBracket
%token leftBracket
%token rightBracket
%token comma
%token period

%token IDENTIFIER
%token NUMBER_CONST
%token STRING_CONST

%start program


%%

CONST : NUMBER_CONST | STRING_CONST;

program : Declaration_list Cmpstmt;

Declaration_list : Declaration Declaration_list
```

```
            | Declaration
            ;

Declaration : Type IDENTIFIER
          | Type IDENTIFIER eq CONST
          | Type Array_declaration
          ;

Type : INT
    | CHAR
    ;

Array_declaration : IDENTIFIER leftRoundBracket CONST rightRoundBracket
              ;

Cmpstmt : Stmt Cmpstmt
       | Stmt
       ;

Stmt : Assign_stmt
   | IOstmt
   | Struct_stmt
   ;

Assign_stmt : IDENTIFIER eq Expression
          ;

IOstmt : CIN rightShift IDENTIFIER
     | COUT leftShift CONST
     | COUT leftShift IDENTIFIER
     ;

Struct_stmt : For_stmt
         | If_stmt
         ;

If_stmt : IF leftBracket Condition rightBracket leftCurlyBracket Cmpstmt rightCurlyBracket
ELSE leftCurlyBracket Cmpstmt rightCurlyBracket
       | IF leftBracket Condition rightBracket leftCurlyBracket Cmpstmt rightCurlyBracket
       ;

For_stmt : FOR leftBracket For_loop rightBracket leftCurlyBracket Cmpstmt
rightCurlyBracket
       ;
```

```
Condition : Expression relation Expression
        ;

For_loop : INT IDENTIFIER eq CONST period IDENTIFIER For_relation IDENTIFIER period
IDENTIFIER increment
        | INT IDENTIFIER eq CONST period IDENTIFIER For_relation CONST period IDENTIFIER
increment
        ;

For_relation : less
           | lessOrEqual
           | more
           | moreOrEqual
           ;

Expression : Term Math_operators Term
         | Term
         ;

Math_operators : mul
             | minus
             | plus
             | division
             ;

Term : CONST
    | IDENTIFIER
    ;

relation : less
        | lessOrEqual
        | more
        | moreOrEqual
        | equal
        | different
        ;

%%

void yyerror(char *s)
{
  printf("%s\n", s);
}

extern FILE *yyin;
```

```
int main(int argc, char **argv)
{
  if(argc>1) yyin = fopen(argv[1], "r");
  if((argc>2)&&(!strcmp(argv[2],"-d"))) yydebug = 1;
  if(!yyparse()) fprintf(stderr,"\tEverything is okay!!!\n");
}
```

// DEMO

```
carpnicoleta@Vitalies-MacBook-Air Lab8 % ./parser p1.txt
Reserved word: int
Identifier: a
Reserved word: int
Identifier: b
Reserved word: int
Identifier: c
Reserved word: cout
Operator <<
String: "First term -> a:"
Reserved word: cin
Operator >>
Identifier: a
Reserved word: cout
Operator <<
String: "Second term -> b:"
Reserved word: cin
Operator >>
Identifier: b
Reserved word: cout
Operator <<
String: "Sum -> c:"
Reserved word: cin
Operator >>
Identifier: c
Reserved word: if
Separator [
Identifier: a
Operator +
Identifier: b
Operator ==
Number: 1
Separator ]
Separator {
Reserved word: cout
Operator <<
String: "a and b add up to c"
Separator }
Reserved word: else
Separator {
Reserved word: cout
Operator <<
String: "wrong sum"
Separator }
Reserved word: cout
Operator <<
String: "...fin..."
        Everything is okay!!!
carpnicoleta@Vitalies-MacBook-Air Lab8 %
```