Lab 09 documentation

Github link:
https://github.com/timoteicopaciu/LFCD/tree/main/Lab_08

Lang.lxi

```
%{
    #include <stdio.h>
    #include "parser.tab.h"
    int lineNumber = 1;
    int correct = 1;
    int badLine = 0;
%}
%option noyywrap

digit                  [0-9]
nonZeroDigit           [1-9]
letter                 [a-zA-Z]
character_constant     [']([a-zA-Z_?! ])*[']
numerical_constant     [-]?{nonZeroDigit}{digit}*|0
constant               {character_constant}|{numerical_constant}
identifier             {letter}|{letter}({letter}|{digit}|_)*?


%%

"main" {printf( "%s - reserved word\n", yytext ); return MAIN; }
"define" {printf( "%s - reserved word\n", yytext ); return DEFINE;}
"Integer" {printf( "%s - reserved word\n", yytext ); return INTEGER;}
"Char" {printf( "%s - reserved word\n", yytext ); return CHAR;}
"while" {printf( "%s - reserved word\n", yytext ); return WHILE;}
"for" {printf( "%s - reserved word\n", yytext ); return FOR;}
"if" {printf( "%s - reserved word\n", yytext ); return IF;}
"else" {printf( "%s - reserved word\n", yytext ); return ELSE; }
"in.Integer" {printf( "%s - reserved word\n", yytext ); return IN_INTEGER;}
"in.Chars" {printf( "%s - reserved word\n", yytext ); return IN_CHARS;}
"out" {printf( "%s - reserved word\n", yytext ); return OUT;}

{identifier} {printf( "Identifier: %s\n", yytext ); return IDENTIFIER;}
```

```
{constant}  {printf( "Constant: %s\n", yytext ); return CONSTANT;}

"[" {printf("%s - as separator\n", yytext); return LEFT_SQUARE_PARENTHESIS;}
"]" {printf("%s - as separator\n", yytext); return RIGHT_SQUARE_PARENTHESIS; }
"{" {printf("%s - as separator\n", yytext); return LEFT_CURLY_PARENTHESIS; }
"}" {printf("%s - as separator\n", yytext); return RIGHT_CURLY_PARENTHESIS;}
"(" {printf("%s - as separator\n", yytext); return LEFT_ROUND_PARENTHESIS;}
")" {printf("%s - as separator\n", yytext); return RIGHT_ROUND_PARENTHESIS;}
";" {printf("%s - as separator\n", yytext); return SEMI_COLON;}
"," {printf("%s - as separator\n", yytext); return COMMA;}

"+" {printf("%s - as operator\n", yytext); return PLUS;}
"-" {printf("%s - as operator\n", yytext); return MINUS;}
"*" {printf("%s - as operator\n", yytext); return MULTIPLY;}
"/" {printf("%s - as operator\n", yytext);return DIVISION;}
"%" {printf("%s - as operator\n", yytext); return MOD;}
">>" {printf("%s - as operator\n", yytext); return IN_SIGN;}
"<=" {printf("%s - as operator\n", yytext); return LESS_OR_EQUAL_THAN; }
">=" {printf("%s - as operator\n", yytext); return GREATER_OR_EQUAL_THAN;}
"==" {printf("%s - as operator\n", yytext);return EQUAL;}
"!=" {printf("%s - as operator\n", yytext); return DIFFERENT;}
"=" {printf("%s - as operator\n", yytext); return ASSIGNMENT;}
"<" {printf("%s - as operator\n", yytext); return LESS_THAN;}
">" {printf("%s - as operator\n", yytext); return GREATER_THAN;}


[ \t]+      {}

[\n]+   {++lineNumber;}

. {correct = 0; badLine = lineNumber; printf("Incorrect:%s\n", yytext);}


%%
```

Parser.y

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

#define YYDEBUG 1
%}
```

```
%token IDENTIFIER
%token CONSTANT
%token MAIN
%token DEFINE
%token INTEGER
%token CHAR
%token WHILE
%token FOR
%token IF
%token ELSE
%token IN_INTEGER
%token IN_CHARS
%token OUT
%token PLUS
%token MINUS
%token MULTIPLY
%token DIVISION
%token MOD
%token IN_SIGN
%token LESS_OR_EQUAL_THAN
%token GREATER_OR_EQUAL_THAN
%token EQUAL
%token DIFFERENT
%token ASSIGNMENT
%token LESS_THAN
%token GREATER_THAN
%token LEFT_CURLY_PARENTHESIS
%token RIGHT_CURLY_PARENTHESIS
%token LEFT_SQUARE_PARENTHESIS
%token RIGHT_SQUARE_PARENTHESIS
%token LEFT_ROUND_PARENTHESIS
%token RIGHT_ROUND_PARENTHESIS
%token SEMI_COLON
%token COMMA


%start program

%%


program : MAIN LEFT_CURLY_PARENTHESIS declarationList stmtList RIGHT_CURLY_PARENT
HESIS ;
declarationList : declaration | declaration declarationList ;
identifierList : IDENTIFIER SEMI_COLON | IDENTIFIER COMMA identifierList ;
```

```
declaration : DEFINE type identifierList;
type : mainTypes | arraysDecl ;
mainTypes : INTEGER | CHAR ;
arraysDecl : mainTypes LEFT_SQUARE_PARENTHESIS CONSTANT RIGHT_SQUARE_PARENTHESIS ;

vectorItem : IDENTIFIER LEFT_SQUARE_PARENTHESIS IDENTIFIER RIGHT_SQUARE_PARENTHESIS | IDENTIFIER LEFT_SQUARE_PARENTHESIS CONSTANT RIGHT_SQUARE_PARENTHESIS ;
item : IDENTIFIER | CONSTANT | vectorItem ;
operator : PLUS | MINUS | MULTIPLY | DIVISION | MOD ;
expression : item operator expression | item operator item | item | LEFT_ROUND_PARENTHESIS item operator expression RIGHT_ROUND_PARENTHESIS | LEFT_ROUND_PARENTHESIS item operator item RIGHT_ROUND_PARENTHESIS ;

RELATION : LESS_THAN | LESS_OR_EQUAL_THAN | EQUAL | DIFFERENT | GREATER_OR_EQUAL_THAN | GREATER_THAN ;

stmtList : stmt | stmt stmtList ;
stmt : assignStmt| inStmt | outStmt | ifStmt | whileStmt | forStmt ;
assignStmt : IDENTIFIER ASSIGNMENT expression SEMI_COLON | vectorItem ASSIGNMENT expression SEMI_COLON;
inStmt : IN_INTEGER IN_SIGN IDENTIFIER SEMI_COLON | IN_CHARS IN_SIGN IDENTIFIER SEMI_COLON | IN_CHARS IN_SIGN vectorItem SEMI_COLON | IN_INTEGER IN_SIGN vectorItem SEMI_COLON ;
outStmt : OUT LEFT_ROUND_PARENTHESIS CONSTANT RIGHT_ROUND_PARENTHESIS SEMI_COLON | OUT LEFT_ROUND_PARENTHESIS IDENTIFIER RIGHT_ROUND_PARENTHESIS SEMI_COLON;
ifStmt : IF LEFT_ROUND_PARENTHESIS condition RIGHT_ROUND_PARENTHESIS LEFT_CURLY_PARENTHESIS stmtList RIGHT_CURLY_PARENTHESIS | IF LEFT_ROUND_PARENTHESIS condition RIGHT_ROUND_PARENTHESIS LEFT_CURLY_PARENTHESIS stmtList RIGHT_CURLY_PARENTHESIS ELSE LEFT_CURLY_PARENTHESIS stmtList RIGHT_CURLY_PARENTHESIS ;
whileStmt : WHILE LEFT_ROUND_PARENTHESIS condition RIGHT_ROUND_PARENTHESIS LEFT_CURLY_PARENTHESIS stmtList RIGHT_CURLY_PARENTHESIS ;
forStmt : FOR LEFT_ROUND_PARENTHESIS IDENTIFIER COMMA expression COMMA expression COMMA CONSTANT RIGHT_ROUND_PARENTHESIS LEFT_CURLY_PARENTHESIS stmtList RIGHT_CURLY_PARENTHESIS ;
condition : expression RELATION expression ;

%%

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

extern FILE *yyin;
```

```
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,"\t Good !!!\n");
}
```

Demo: - good

Run output for p1.txt

p1.txt

```
main{
        define Integer x , y , copy_x , p ;
        y = 0;
        p = 1;
        in.Integer>> x;
        copy_x = x;
        while(x != 0){
                y = y + x % 10 * p;
                p = p * 10;
                x = x / 10;
        }

        if(y == copy_x){
                out('The_integer_is_palindrome!');
        }
        else{
                out('The_integer_is_not_palindrome!');
        }
}
```

Output:

```
main - reserved word
{ - as separator
define - reserved word
Integer - reserved word
Identifier: x
, - as separator
Identifier: y
, - as separator
Identifier: copy_x
, - as separator
Identifier: p
; - as separator
Identifier: y
= - as operator
Constant: 0
; - as separator
Identifier: p
= - as operator
Constant: 1
; - as separator
in.Integer - reserved word
>> - as operator
Identifier: x
; - as separator
Identifier: copy_x
```

```
= - as operator
Identifier: x
; - as separator
while - reserved word
( - as separator
Identifier: x
!= - as operator
Constant: 0
) - as separator
{ - as separator
Identifier: y
= - as operator
Identifier: y
+ - as operator
Identifier: x
% - as operator
Constant: 10
* - as operator
Identifier: p
; - as separator
Identifier: p
= - as operator
Identifier: p
* - as operator
Constant: 10
; - as separator
Identifier: x
= - as operator
Identifier: x
/ - as operator
Constant: 10
; - as separator
} - as separator
if - reserved word
( - as separator
Identifier: y
== - as operator
Identifier: copy_x
) - as separator
{ - as separator
out - reserved word
( - as separator
Constant: 'The_integer_is_palindrome!'
) - as separator
; - as separator
} - as separator
else - reserved word
{ - as separator
out - reserved word
( - as separator
Constant: 'The_integer_is_not_palindrome!'
) - as separator
; - as separator
} - as separator
} - as separator
    Good !!!
```

Demo: - with error

Run output for p1.txt

p1.txt

```
main{
        define Integer x , y , copy_x , p ;
        y = 0;
        p = 1;
        in.Integer>> x;
        copy_x = x;
        while(x != 0){
                y = y + x % 10 * p
                p = p * 10;
                x = x / 10;
        }

        if(y == copy_x){
                out('The_integer_is_palindrome!');
        }
        else{
                out('The_integer_is_not_palindrome!');
        }
}
```

Output:

```
main - reserved word
{ - as separator
define - reserved word
Integer - reserved word
Identifier: x
```

```
, - as separator
Identifier: y
, - as separator
Identifier: copy_x
, - as separator
Identifier: p
; - as separator
Identifier: y
= - as operator
Constant: 0
; - as separator
Identifier: p
= - as operator
Constant: 1
; - as separator
in.Integer - reserved word
>> - as operator
Identifier: x
; - as separator
Identifier: copy_x
= - as operator
Identifier: x
; - as separator
while - reserved word
( - as separator
Identifier: x
!= - as operator
Constant: 0
```

```
) - as separator
{ - as separator
Identifier: y
= - as operator
Identifier: y
+ - as operator
Identifier: x
% - as operator
Constant: 10
* - as operator
Identifier: p
Identifier: p
syntax error
```