***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\_PARENTHESIS ;

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