**FLCD Lab. 9**

***lang.y (YACC)***

%{

#include <stdio.h>

#include <stdlib.h>

#define YYDEBUG 1

#define TYPE\_INT 1

#define TYPE\_STRING 2

%}

%token MAIN

%token RETURN

%token READ

%token WRITE

%token TRUE

%token FALSE

%token EQUALS

%token LESS\_EQUAL

%token LESS\_THAN

%token GREATER\_EQUAL

%token GREATER\_THAN

%token IF

%token ELSE

%token FOR

%token ID

%token NOT\_ID

%token INT

%token STRING

%token CONST\_INT

%token CONST\_STRING

%token CONST\_SIR

%token PLUS

%token MINUS

%token MUL

%token DIV

%token MOD

%token EQ

%token SEMICOLON

%token OPEN\_BRACE

%token CLOSED\_BRACE

%token OPEN\_BRACKET

%token CLOSED\_BRACKET

%token OPEN\_SQUARE\_BRACKET

%token CLOSED\_SQUARE\_BRACKET

%token UNRECOGNIZED

%start program

%%

program: INT MAIN OPEN\_BRACKET CLOSED\_BRACKET OPEN\_BRACE statementList CLOSED\_BRACE

;

relation: EQUALS

| LESS\_EQUAL

| LESS\_THAN

| GREATER\_EQUAL

| GREATER\_THAN

;

operand: ID

| CONST\_INT

| CONST\_STRING

| ID OPEN\_SQUARE\_BRACKET ID CLOSED\_SQUARE\_BRACKET

| ID OPEN\_SQUARE\_BRACKET CONST\_INT CLOSED\_SQUARE\_BRACKET

;

operator: PLUS

| MINUS

| MUL

| DIV

| MOD

| EQ

;

statementList: statement

| statement statementList

;

statement: simpleStatement SEMICOLON

| structuredStatement

;

simpleStatement: declaration

| io

| assignment

| condition

| return

;

declaration: INT ID

| STRING ID

| INT ID EQ CONST\_INT

| STRING ID EQ CONST\_STRING

| INT ID OPEN\_SQUARE\_BRACKET ID CLOSED\_SQUARE\_BRACKET

| STRING ID OPEN\_SQUARE\_BRACKET ID CLOSED\_SQUARE\_BRACKET

| INT ID OPEN\_SQUARE\_BRACKET CONST\_STRING CLOSED\_SQUARE\_BRACKET

| STRING ID OPEN\_SQUARE\_BRACKET CONST\_STRING CLOSED\_SQUARE\_BRACKET

;

io: READ ID

| READ ID OPEN\_SQUARE\_BRACKET ID CLOSED\_SQUARE\_BRACKET

| READ ID OPEN\_SQUARE\_BRACKET CONST\_INT CLOSED\_SQUARE\_BRACKET

| WRITE ID

| WRITE CONST\_INT

| WRITE CONST\_STRING

| WRITE ID OPEN\_SQUARE\_BRACKET ID CLOSED\_SQUARE\_BRACKET

| WRITE ID OPEN\_SQUARE\_BRACKET CONST\_INT CLOSED\_SQUARE\_BRACKET

;

assignment: ID EQ expression

| ID EQ ID

| ID EQ CONST\_INT

| ID EQ CONST\_STRING

;

expression: operand operator operand

| ID operator ID

| ID operator CONST\_INT

| ID operator CONST\_STRING

| CONST\_INT operator CONST\_INT

| CONST\_STRING EQ CONST\_STRING

;

condition: operand relation operand

| expression relation operand

| operand relation expression

| expression relation expression

;

return: RETURN CONST\_INT

;

structuredStatement: if

| for

;

if: IF OPEN\_BRACKET condition CLOSED\_BRACKET OPEN\_BRACE statementList CLOSED\_BRACE

;

for: FOR OPEN\_BRACKET assignment SEMICOLON condition SEMICOLON assignment CLOSED\_BRACKET OPEN\_BRACE statementList CLOSED\_BRACE

%%

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,"\tO.K.\n");

}

***lang.lxi (LEX)***

%{

#include "syn.tab.h"

%}

%option noyywrap

ID [A-Za-z][A-Za-z0-9]\*

NOT\_ID [^ ][A-Za-z][A-Za-z0-9]\*

INTEGER [+-]?(0|[1-9][0-9]\*)

STRING \"[A-Za-z0-9 \_\.,:;!\?\'#]\*\"

%%

{INTEGER} {return CONST\_INT;}

{STRING} {return CONST\_STRING;}

main {return MAIN;}

return {return RETURN;}

read {return READ;}

write {return WRITE;}

true {return TRUE;}

false {return FALSE;}

equals {return EQUALS;}

lessEqual {return LESS\_EQUAL;}

lessThan {return LESS\_THAN;}

greaterEqual {return GREATER\_EQUAL;}

greaterThan {return GREATER\_THAN;}

if {return IF;}

else {return ELSE;}

for {return FOR;}

int {return INT;}

string {return STRING;}

{ID} {return ID;}

{NOT\_ID} {return NOT\_ID;}

"+" {return PLUS;}

"-" {return MINUS;}

"\*" {return MUL;}

"/" {return DIV;}

"%" {return MOD;}

"=" {return EQ;}

";" {return SEMICOLON;}

"{" {return OPEN\_BRACE;}

"}" {return CLOSED\_BRACE;}

"(" {return OPEN\_BRACKET;}

")" {return CLOSED\_BRACKET;}

"[" {return OPEN\_SQUARE\_BRACKET;}

"]" {return CLOSED\_SQUARE\_BRACKET;}

[ \t\n]+ /\* eat up whitespace \*/

. {return UNRECOGNIZED;}

%%

**DEMO’S**

**Syntactically Correct**

* output: O.K.
* code:

int main ( )

{

int no = 0 ;

int no2 = 0 ;

int no3 = 0 ;

int minimum = 0 ;

read no ;

read no2 ;

read no3 ;

minimum = no ;

if ( minimum lessThan no2 ) {

minimum = no2 ;

}

if ( minimum lessThan no3 ) {

minimum = no3 ;

}

write minimum ;

return 0 ;

}

**Syntactically Incorrect**

* missing semicolon after first declaration
* output: syntax error
* code:

int main ( )

{

int no = 0

int no2 = 0 ;

int no3 = 0 ;

int minimum = 0 ;

read no ;

read no2 ;

read no3 ;

minimum = no ;

if ( minimum lessThan no2 ) {

minimum = no2 ;

}

if ( minimum lessThan no3 ) {

minimum = no3 ;

}

write minimum ;

return 0 ;

}