**lang.y**

%{

#include <stdio.h>

#include <stdlib.h>

int yylex();

int yyerror(char \*s);

#define YYDEBUG 1

%}

%token DEF

%token STRING

%token INT

%token SPACE

%token BACK\_TICK

%token READ

%token PRINT

%token IF

%token ELSE

%token WHILE

%token EXIT

%token GO

%token BYE

%token IDENTIFIER

%token INT\_CONSTANT

%token STRING\_CONSTANT

%token EQ

%token NE

%token LE

%token GE

%token LT

%token GT

%token DBLEQ

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

%token PLUS

%token MINUS

%token DIV

%token MOD

%token MUL

%token POWER

%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 SEMI\_COLON

%token COLON

%token TRUE

%token FALSE

%start program

%%

program : GO tempDecl BYE ;

tempDecl : /\*Empty\*/ | declList tempDecl | stmtList tempDecl ;

declList : declaration | declaration declList ;

declaration : DEF IDENTIFIER COLON type SEMI\_COLON | DEF IDENTIFIER COLON type EQ expression SEMI\_COLON ;

type1 : STRING | INT ;

arraydecl : OPEN\_RIGHT\_BRACKET type1 CLOSED\_RIGHT\_BRACKET ;

type : type1 | arraydecl ;

stmtList : stmt | stmt stmtList ;

stmt : simplstmt | structstmt ;

simplstmt : assignstmt | iostmt ;

assignstmt : IDENTIFIER EQ expression SEMI\_COLON ;

expression : number\_expression | string\_expression ;

number\_expression : number\_expression PLUS term | number\_expression MINUS term | term ;

term : term MUL factor | term DIV factor | term MOD factor | term POWER factor | factor ;

factor : OPEN\_ROUND\_BRACKET number\_expression CLOSED\_ROUND\_BRACKET | IDENTIFIER | INT\_CONSTANT {printf("at int\n");} ;

string\_expression : STRING\_CONSTANT | IDENTIFIER;

iostmt : READ OPEN\_ROUND\_BRACKET IDENTIFIER CLOSED\_ROUND\_BRACKET SEMI\_COLON| PRINT OPEN\_ROUND\_BRACKET string\_expression CLOSED\_ROUND\_BRACKET SEMI\_COLON ;

structstmt : ifstmt | whilestmt | exitstmt ;

body : OPEN\_CURLY\_BRACKET stmtList CLOSED\_CURLY\_BRACKET | stmt ;

ifstmt : IF OPEN\_ROUND\_BRACKET condition CLOSED\_ROUND\_BRACKET body | OPEN\_ROUND\_BRACKET condition CLOSED\_ROUND\_BRACKET body ELSE body ;

whilestmt : WHILE OPEN\_ROUND\_BRACKET condition CLOSED\_ROUND\_BRACKET body ;

exitstmt : EXIT expression SEMI\_COLON ;

condition : expression relation expression ;

relation : LT | LE | DBLEQ | GE | GT | NE ;

%%

yyerror(char \*s)

{

printf("err: %s\n",s);

}

extern FILE \*yyin;

main(int argc, char \*\*argv)

{

if (argc > 1)

yyin = fopen(argv[1], "r");

if (!yyparse())

fprintf(stderr, "\tOK\n");

}

**specif.lxi – updated**

%{

#include<stdio.h>

#include <string.h>

#include "lang.tab.h"

int lines = 0;

%}

%option noyywrap

%option caseless

DIGIT [0-9]

NON\_ZERO\_DIGIT [1-9]

INT\_CONSTANT [-]?{NON\_ZERO\_DIGIT}{DIGIT}\*|0

LETTER [a-zA-Z]

STRING\_CONSTANT [`$]({LETTER}|{DIGIT}|" ")\*[`$]

IDENTIFIER {LETTER}({LETTER}|{DIGIT})\*

%%

true {printf( "Reserved word: %s\n", yytext); return TRUE;}

false {printf( "Reserved word: %s\n", yytext); return FALSE;}

def {printf( "Reserved word: %s\n", yytext); return DEF;}

int {printf( "Reserved word: %s\n", yytext); return INT;}

string {printf( "Reserved word: %s\n", yytext); return STRING;}

read {printf( "Reserved word: %s\n", yytext); return READ;}

print {printf( "Reserved word: %s\n", yytext); return PRINT;}

if {printf( "Reserved word: %s\n", yytext); return IF;}

else {printf( "Reserved word: %s\n", yytext); return ELSE;}

while {printf( "Reserved word: %s\n", yytext); return WHILE;}

exit {printf( "Reserved word: %s\n", yytext); return EXIT;}

GO {printf( "Reserved word: %s\n", yytext); return GO;}

BYE {printf( "Reserved word: %s\n", yytext); return BYE;}

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

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

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

"[" {printf( "Separator: %s\n", yytext ); return OPEN\_RIGHT\_BRACKET;}

"]" {printf( "Separator: %s\n", yytext ); return CLOSED\_RIGHT\_BRACKET;}

"{" {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 COLON;}

";" {printf( "Separator: %s\n", yytext ); return SEMI\_COLON;}

"+" {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 MOD;}

">" {printf( "Operator: %s\n", yytext ); return GT;}

"<" {printf( "Operator: %s\n", yytext ); return LT;}

"<=" {printf( "Operator: %s\n", yytext ); return LE;}

"==" {printf( "Operator: %s\n", yytext ); return DBLEQ;}

"=" {printf( "Operator: %s\n", yytext ); return EQ;}

">=" {printf( "Operator: %s\n", yytext ); return GE;}

"!=" {printf( "Operator: %s\n", yytext ); return NE;}

"^" {printf( "Operator: %s\n", yytext ); return POWER;}

[ \t]+ {}

[\n]+ {lines++;}

. {printf("Error at token %s at line %d\n", yytext, lines); return -1;}

%%

**p1.txt**

GO

def x : int;

def y : int = 7 ;

def z : int = -1 ;

x = y^2 + z^(-2);

print(`The squared hypothenuse is`);

print(x);

BYE

**Output:**

Reserved word: GO

Reserved word: def

Identifier: x

Separator: :

Reserved word: int

Separator: ;

Reserved word: def

Identifier: y

Separator: :

Reserved word: int

Operator: =

Constant: 7

at int

Separator: ;

Reserved word: def

Identifier: z

Separator: :

Reserved word: int

Operator: =

Constant: -1

at int

Separator: ;

Identifier: x

Operator: =

Identifier: y

Operator: ^

Constant: 2

at int

Operator: +

Identifier: z

Operator: ^

Separator: (

Constant: -2

at int

Separator: )

Separator: ;

Reserved word: print

Separator: (

Constant: `The squared hypothenuse is`

Separator: )

Separator: ;

Reserved word: print

Separator: (

Identifier: x

Separator: )

Separator: ;

Reserved word: BYE

OK

**P1-err**

GO

def x : int;

def y = int = 7 ;

def z : int = -1 ;

x = y^2 + z^(-2);

print(`The squared hypothenuse is`);

print(x);

BYE

**Output**

Reserved word: GO

Reserved word: def

Identifier: x

Separator: :

Reserved word: int

Separator: ;

Reserved word: def

Identifier: y

Operator: =

err: syntax error