lex/yacc example

Grammar

Prog -> StmtSeq
StmtSeq -> Stmt StmtSeq
StmtSeq -> ε
Stmt -> Id = Expr ;
Expr -> Expr + Term
Expr -> Term
Term -> Term * Factor
Term -> Factor
Factor -> - Factor
Factor -> (Expr)
Factor -> Id
Factor -> INT
Factor -> FLOAT
Id -> Ident

lex/yacc example
lex file

```
%{
#include "yaccExample.h"
#include "y.tab.h"

#define YY_INPUT(buf,result,max_size) \
  { int c = getNextSourceChar(); \
          result = (c == EOF) ? YY_NULL : (buf[0] = c, 1); \
        }

%}


letter [A-Za-z]
digit [0-9]

%%
int     {return INT;}
float           {return FLOAT;}
```

lex/yacc example
lex file

```
{letter}({letter}|{digit})*      {return Ident;}
{digit}{digit}*\.{digit}{digit}* {return FLOATLIT;}
{digit}{digit}*                   {return INTLIT;}
\=                                {return '=';}
\+                                {return '+';}
\*                                {return '*';}
\-                                {return '-';}
\;                                {return ';';}
\(                                {return '(';}
\)                                {return ')';}

[ ]                               {}
\t                                {}
\r                                {}
\n                                {}

.                                 {writeIndicator(getCurrentColumnNum());
                                        writeMessage("Illegal Character in lex");}


%%

int yywrap () {
    return 1;
}
```

lex/yacc example
yacc file

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "SymTab.h"
#include "IOMngr.h"
#include "semantics.h"

extern int yylex(); /* The next token function. */
extern char *yytext;   /* The matched token text.  */
extern int yyleng;      /* The token text length.   */
extern int yyparse();
extern int yyerror(char *);
void dumpTable();

SymTab *table;

%}
```

lex/yacc example
yacc file

```
%union {
  char * string;
  struct Value * val;
}

%type <string> Id
%type <val> Factor
%type <val> Term
%type <val> Expr

%token Ident
%token INT
%token FLOAT
%token INTLIT
%token FLOATLIT
```

lex/yacc example

```
%%

Prog         : Declarations StmtSeq  {dumpTable(); } ;
Declarations : Dec Declarations      { };
Declarations :                       { };
Dec          : INT Id ';'            {declaration($2, 0); };
Dec          : FLOAT Id ';'          {declaration($2, 1); };
StmtSeq      : Stmt StmtSeq          { } ;
StmtSeq      :                       { } ;
Stmt         :  Id '=' Expr ';'      {assignment($1, $3);};
Expr         :  Term '+' Expr        { $$ = add($1, $3); } ;
Expr         :  Term                 { $$ = $1; } ;
Term         :  Factor '*' Term      { $$ = mult($1, $3); } ;
Term         :  Factor                { $$ = $1; } ;
Factor       : '-' Factor            {$$ = unary($2);};
Factor       : '(' Expr ')'          {$$ = $2;};
Factor       :  INTLIT               { $$ = makeInt(yytext); } ;
Factor       :   FLOATLIT           { $$ = makeFloat(yytext); } ;
Factor       :  Id                   {$$ =  getVal($1);}
Id           :  Ident               {$$ = strdup(yytext); }

%%
```

lex/yacc example
yacc file

```
int main(int argc, char *argv[])  {
  table = createSymTab(19);
  openFiles(argv[1], argv[2]);
  yyparse();
}

int yyerror(char *s)  {
  writeIndicator(getCurrentColumnNum());
  writeMessage("Illegal Character in YACC");
    return 1;
}
```

lex/yacc example
semantics.h

```
struct Value {
    int type;
    int intVal;
    double floatVal;
};

struct Value *getVal(char *var);
void dumpTable();
struct Value * add(struct Value * x, struct Value * y);
struct Value * mult(struct Value * x, struct Value * y);
struct Value * unary(struct Value * x);
void declaration(char * id, int type);
void assignment(char *id, struct Value * v);
struct Value * makeInt(char * i);
struct Value * makeFloat(char * i);
```

lex/yacc example
semantics.c

```
#include "SymTab.h"
#include "IOMngr.h"
#include "semantics.h"
extern SymTab *table;

struct Value * getVal(char *var) {
  int x = findName(table, var);
  if (x) return  getCurrentAttr(table);;
  writeIndicator(getCurrentColumnNum());
  writeMessage("Variables must be declared before they are used");
  exit(1);
}

void dumpTable() {
  struct Value *x;
  int more = startIterator(table);
  while (more) {
        x = (struct Value *) getCurrentAttr(table);
        if (x ->type == 0)
            printf("%10s%8s%10d\n", getCurrentName(table), "INT", x->intVal);
        else
            printf("%10s%8s%10.2f\n", getCurrentName(table), "FLOAT", x->floatVal);
    more = nextEntry(table);
  }
}
```

lex/yacc example
semantics.c

```
struct Value * add(struct Value * x, struct Value * y) {
    struct Value * result = (struct Value *) malloc(sizeof(struct Value));
    if (x->type == 0 && y->type == 0) {
        result->type = 0;
        result->intVal = x->intVal + y->intVal;
    } else {
        result->type = 1;
        if (x->type == 0 && y->type == 1)
            result->floatVal = x->intVal + y->floatVal;
        else if (x->type == 1  && y->type == 0)
            result->floatVal = x->floatVal + y->intVal;
        else
            result->floatVal = x->floatVal + y->floatVal;
    }
    return result;
}
```

lex/yacc example
semantics.c

```
struct Value * mult(struct Value * x, struct Value * y) {
    struct Value * result = (struct Value *) malloc(sizeof(struct Value));
    if (x->type == 0 && y->type == 0) {
        result->type = 0;
        result->intVal = x->intVal * y->intVal;
    } else {
        result->type = 1;
        if (x->type == 0 && y->type == 1)
            result->floatVal = x->intVal * y->floatVal;
        else if (x->type == 1  && y->type == 0)
            result->floatVal = x->floatVal * y->intVal;
        else
            result->floatVal = x->floatVal * y->floatVal;
    }
    return result;
}
```

lex/yacc example
semantics.c

```
void declaration(char * id, int type) {
    enterName(table, id);
    struct Value * result = (struct Value *) malloc(sizeof(struct Value));
    if (type == 0) {
        result->type = 0;
        result->intVal = 0;
    } else {
        result->type = 1;
        result->floatVal = 0.0;

    }
    setCurrentAttr(table, (void *) result);
}
```

lex/yacc example
semantics.c

```
void assignment(char *id, struct Value * v) {
    int x = findName(table, id);
    if (x) {
        struct Value *attr = getCurrentAttr(table);
        if (attr->type == 0 && v->type == 1) {
            v->type = 0;
            v->intVal = (int) v->floatVal;
        } else if (attr->type == 1 && v->type == 0){
            v->type = 1;
            v->floatVal = v->intVal;
        }
        setCurrentAttr(table, (void *) v);
    } else {
        writeIndicator(getCurrentColumnNum());
        writeMessage("Variables must be declared before they are used");
        exit(1);
    }
}
```

lex/yacc example
semantics.c

```
struct Value * unary(struct Value * x) {
    struct Value * result = (struct Value *) malloc(sizeof(struct Value));
    if (x->type == 0) {
        result->type = 0;
        result->intVal = -x->intVal;
    } else {
        result->type = 1;
        result->floatVal = -x->floatVal;
    }
    return result;
}
```

lex/yacc example
semantics.c

```
struct Value * makeInt(char * i) {
    struct Value * result = (struct Value *) malloc(sizeof(struct Value));
    result->type = 0;
    result->intVal = atoi(i);
    return result;
}

struct Value * makeFloat(char * i) {
    struct Value * result = (struct Value *) malloc(sizeof(struct Value));
    result->type = 1;
    result->floatVal = atof(i);
    return result;
}
```