编译原理 yacc 实验

实验目的

用 yacc 和 lex 编写一个实现简单计算器功能的语法分析器

经验总结

1. 一开始输入1+2总是报错，但是只输入12，13等等又不报错，后来发现是自己 lex 中 num 这个 token 的正则表达式写错了。以后要学会利用 printf 来发现问题。
2. 为了让计算器支持浮点数，需要把 YYSTYPE 定义成 double 类型，但是在 lex 文件和yacc 文件中都加入这一行后, yylval 还是 int 类型. 后来发现在 lex 文件中这条 define 语句一定要在其他 include 语句的上面

```
%{
    #define YYSTYPE double
    #include "y.tab.h"
    #include <string.h>
    #include <stdio.h>
%}
```

如果放在下面的话，y.tab.h 中就会先把 YYSTYPE 定义成 int，然后声明 extern YYSTYPE yylval，后面再 define 成 double就晚了

实验代码

calculator.lex 文件

```
%{
    #define YYSTYPE double
    #include "y.tab.h"
    #include <string.h>
    #include <stdio.h>
%}

digit [0-9]
number (0)|([1-9]{digit}*)|(0\.{digit}+)|([1-9]{digit}*.{digit}+)
blanks ([ \t]+)
%%
{number}   { yylval = atof(yytext); return NUM; }
\+         { return PLUS; }
\-           { return MINUS; }
\*           { return MUL; }
\/           { return DIV; }
\^           { return POW; }
\(           { return LPAREN; }
\)           { return RPAREN; }
{blanks} ;
\n           { return '\n'; }
.            { return yytext[0]; }

%%

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

calculator.y 文件

```
%{

#define YYSTYPE double
#include <stdio.h>
#include <math.h>

int yyerror(char *msg);
extern int yylex(void);
extern FILE *yyin;

%}
%token NUM PLUS MINUS MUL DIV POW LPAREN RPAREN LINEEND
%start file
```

```
%left PLUS MINUS
%left MUL DIV
%left POW
%left UMINUS

%%

file: file command
    | command
    ;
command: exp '\n' { printf("result: %lf\n", $1); }
        | error '\n' { printf("Invalid expression!\n"); }
        ;
exp: NUM            { $$ = $1; }
    | exp PLUS exp  { $$ = $1 + $3; }
    | exp MINUS exp { $$ = $1 - $3; }
    | exp MUL exp   { $$ = $1 * $3; }
    | exp DIV exp   { $$ = $1 / $3; }
    | exp POW exp   { $$ = pow($1, $3); }
    | LPAREN exp RPAREN { $$ = $2; }
    | MINUS exp %prec UMINUS { $$ = -$2; }
    ;

%%
int main(int argc, char *argv[])
{
    if(argc < 2) {
        yyin = stdin;
    } else {
        yyin = fopen(argv[1], "r");
    }
    return yyparse();
}

int yyerror(char *msg) {
    printf("Error encountered: %s\n", msg);
}

```

Makefile

```
TARGET=calculate
ALL:
    lex calculate.lex
    yacc -d calculate.y
    gcc y.tab.c lex.yy.c -o $(TARGET) -lm

```