郑凯琳 205220025

项目三：综合报告

实验内容概述

基本功能：

- 判断表达式合法性
- 输出相应结果
- 错误表达式的识别、位置信息，错误类型，高亮等
  - 如果有多个错误，只需要指出一个就行
- 小数保留小数点后 6 位。
- 记录合法算式的计算历史记录，只需要用 txt 文件记录。

框架设计 & 程序流程

创建一个 Complex 类：

- 表示复数，包含实部 real 和虚部 image 两个成员变量
- 运算符重载实现复数的运算：
  - 加法、减法、乘法、除法、n 次幂
  - 输出（友元）—— 使程序输出复数类

```
// 构造函数（Double类型）
Complex(double r = 0.0, double i = 0.0) : real(r), image(i) {}

// 加法运算符重载
Complex operator+(Complex& c);
// 减法运算符重载
Complex operator-(Complex& c);
// 乘法运算符重载
Complex operator*(Complex& c);
// 除法运算符重载
Complex operator/(Complex& c);
// n次幂 Z^n
// 计算复数的n次幂
Complex operator^(int n);

// 重载输出运算符，使程序可以直接输出一个复数类
friend ostream& operator<<(ostream& os, Complex& c);
```

- 其他运算：取模、共轭、幅角主值

```
// 取模 ||
// Gives the magnitude of the two numbers
double mod();

// 共轭 cjg(Z)
// Create the complex conjugate
Complex cjg();

// 辐角主值 arg(Z)
// Gives the angle between real and image
double arg();
```

逆波兰表达式运算：

- 代码中定义了三个辅助函数：
  - int getPriority(char opr)：根据运算符返回优先级，数字越小优先级越高
  - bool isDigit(char c)：判断一个字符是否为数字
  - bool isOperator(char opr)：判断一个字符是否为运算符
- 使用两个栈：oprStack（操作符栈）和 numStack（操作数栈）
- 在主循环中，根据字符的类型进行相应的处理：
  - 如果字符是一个数字，
    - 将连续的数字字符读取为一个完整的数值，包括可能的小数点，
    - 并将其转换为 Complex 对象（实部或虚部），压入 numStack 栈中。
  - 如果字符是虚数单位字符 'i'，
    - 根据其位置和前后字符的类型进行判断，
    - 并将相应的 Complex 对象压入 numStack 栈中。
  - 如果字符是一个运算符，
    - 根据运算符的优先级和栈中已有的运算符进行比较，并进行相应的操作：

1. 如果 oprStack 为空或当前运算符的优先级是 2（即'a'、'c'或'm'），或当前运算符是 '('，则将当前运算符压入 oprStack 栈中。
2. 如果 oprStack 非空且当前运算符不是 ')'，且当前运算符的优先级不高于栈顶运算符的优先级，则将当前运算符压入 oprStack 栈中。
3. 如果 oprStack 非空且当前运算符的优先级高于栈顶运算符的优先级，并且下一个字符不是 '('，且栈顶运算符不是 '('，则将栈顶的运算符弹出，从 numStack 栈中取出两个操作数进行运算，并将结果压入 numStack 栈中，直到当前运算符的优先级不再高于栈顶运算符的优先级或栈顶运算符是 '('。
4. 如果当前字符不是 '('，当前字符不是 ')'，下一个字符是 '('，则将当前运算符压入 oprStack 栈中。
5. 如果字符是 ')'，
   a. 首先处理栈中优先级大于 2 的操作符：
      从 numStack 栈中取出两个操作数和栈顶操作符进行运算，并将结果压入 numStack 栈中。
   b. 然后处理栈顶操作符是 '(' 的情况：
      将其弹出，如果栈顶操作符优先级为 2，则从 numStack 栈中取出一个操作数和栈顶操作符进行运算，并将结果压入 numStack 栈中。
   c. 如果栈顶操作符仍然是 '('，则继续处理。

- 循环结束后，处理剩余的操作符：
  - 从 numStack 栈中取出相应的操作数和栈顶操作符进行运算，并将结果压入 numStack 栈中，直到 oprStack 栈为空或 numStack 中的操作数少于 2 个。
- 最后，返回 numStack 栈顶的元素作为计算结果。

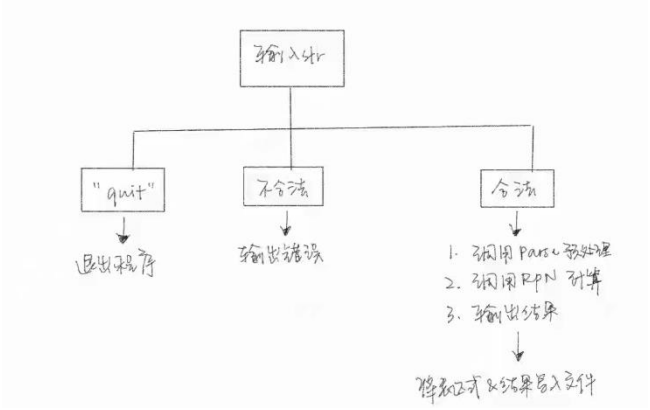
```
// 运算符优先级
int getPriority(char c);

// 判断运算符是否为操作数
bool isDigit(char c);
// 判断运算符是否为操作符
bool isOperator(char op);

// 运算
Complex Calculate(Complex c1, Complex c2, char op);
// 逆波兰表达式运算
Complex RPN(string exp);
```

功能设计：

- 设计一个函数 errorPrint，用于在控制台中打印错误信息并对错误位置进行标记。
  - s，表示待打印的字符串；pos，表示错误位置的索引；error，表示错误类型
  - 将控制台文本颜色设置为红色

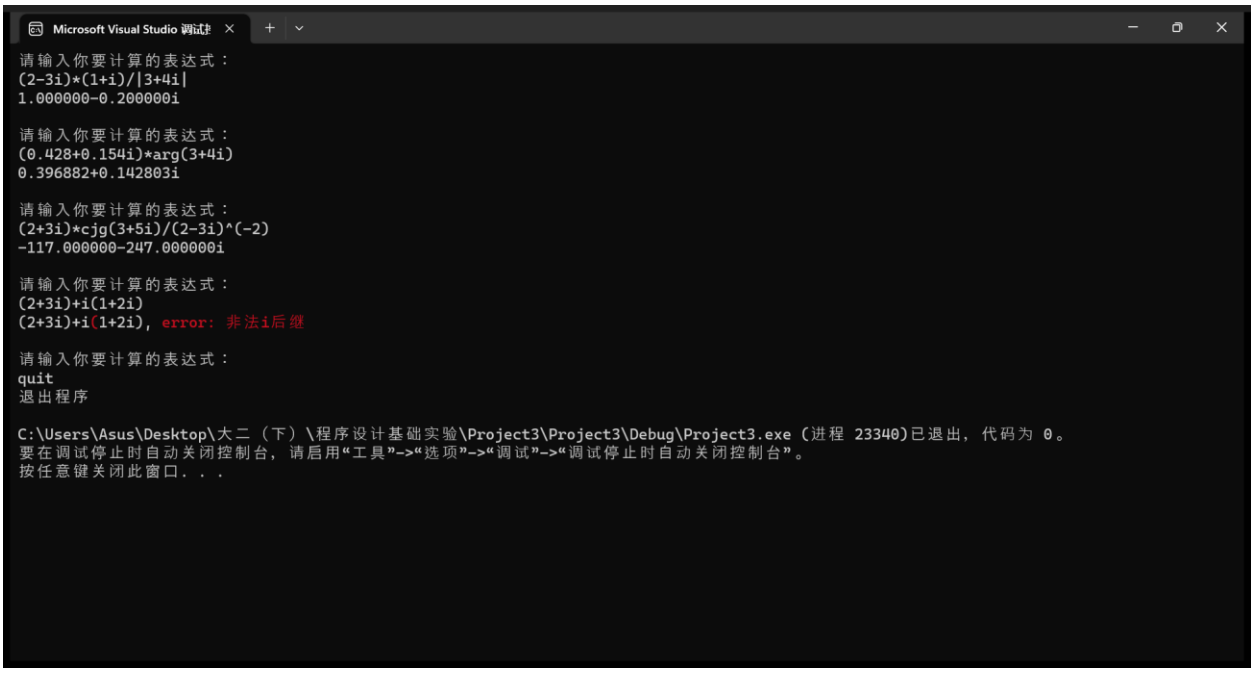
    ```
    #include <windows.h>
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 4); // 红色
    ```
- 设计一个函数 Check，用于检查用户输入的表达式是否合法。
  - 开头不合法
  - 结尾不合法
  - 非法操作符后继(+-*/)
  - 非法实数后继
  - 非法 i 后继
  - 括号内的结果必须是实数
  - 非法 (后继
  - 非法) 后继
  - 括弧不匹配
- 设计一个函数 Parse，用于解析用户输入的表达式。
  - 去除空格
  - 将 arg 转换为 a
  - 将 cjg 转换为 c
  - 将 mod 转换为 m

主函数流程：

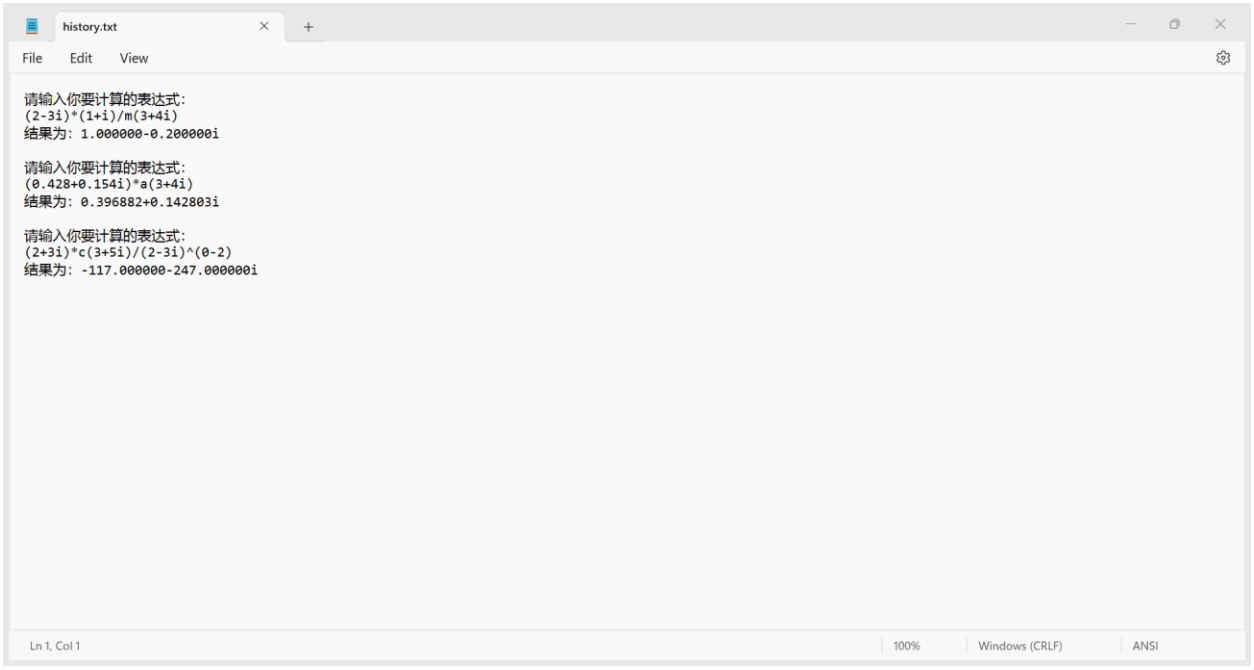


功能运行测试

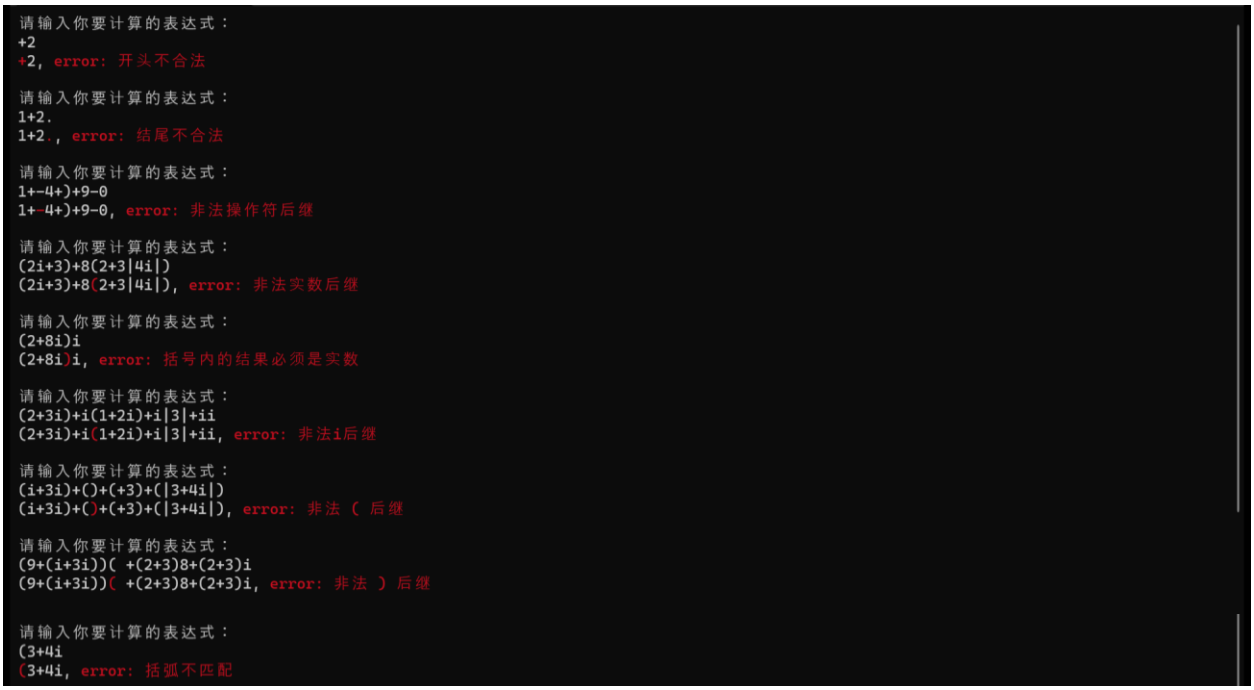
样例测试：



写入文件 history.txt：



不合法输入:


请输入你要计算的表达式：
+2
+2, error: 开头不合法
请输入你要计算的表达式：
1+2.
1+2., error: 结尾不合法
请输入你要计算的表达式：
1+-4+)+9-0
1+-4+)+9-0, error: 非法操作符后继
请输入你要计算的表达式：
(2i+3)+8(2+3|4i|)
(2i+3)+8(2+3|4i|), error: 非法实数后继
请输入你要计算的表达式：
(2+8i)i
(2+8i)i, error: 括号内的结果必须是实数
请输入你要计算的表达式：
(2+3i)+i(1+2i)+i|3|+ii
(2+3i)+i(1+2i)+i|3|+ii, error: 非法i后继
请输入你要计算的表达式：
(i+3i)+()+(+3)+(|3+4i|)
(i+3i)+()+(+3)+(|3+4i|), error: 非法 (后继
请输入你要计算的表达式：
(9+(i+3i))(+(2+3)8+(2+3)i
(9+(i+3i))(+(2+3)8+(2+3)i, error: 非法) 后继
请输入你要计算的表达式：
(3+4i
(3+4i, error: 括弧不匹配