项目说明文档

数据结构课程设计

——算数表达式求解

作 者 姓 名： 赵子昱

学 号： 1951459

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc59371530)

[1.1 项目要求 1](#_Toc59371531)

[1.2 功能分析 1](#_Toc59371532)

[2 设计 2](#_Toc59371533)

[2.1 数据结构设计 2](#_Toc59371534)

[2.2 成员与操作设计 2](#_Toc59371535)

[2.3 系统设计 3](#_Toc59371536)

[3 实现 3](#_Toc59371537)

[3.1 输入表达式并计算**Claculate(…)**功能的实现 3](#_Toc59371538)

[3.1.1 输入表达式并计算功能流程图 4](#_Toc59371539)

[3.1.2 输入表达式并计算功能代码 5](#_Toc59371540)

[3.1.3 输入表达式并计算功能截屏示例 6](#_Toc59371541)

[3.2 进行一次计算**CalculateOnce ()**功能的实现 7](#_Toc59371542)

[3.2.1 进行一次计算功能流程图 8](#_Toc59371543)

[3.2.2 进行一次计算代码 8](#_Toc59371544)

[3.3 幂运算**pow(…)**功能的实现 9](#_Toc59371545)

[3.3.1 幂运算功能代码 9](#_Toc59371546)

[3.4 **Sign**类构造函数的实现 9](#_Toc59371547)

[3.4.1 **Sign**类构造函数代码 10](#_Toc59371548)

[3.6 主函数的实现 10](#_Toc59371549)

[3.6.1 总体系统核心代码 10](#_Toc59371550)

[3.6.2 主函数截屏示例 11](#_Toc59371551)

[4 测试 12](#_Toc59371552)

[4.1 功能测试 12](#_Toc59371553)

[4.1.1 测试1 12](#_Toc59371554)

[4.1.2 测试2 12](#_Toc59371555)

[4.1.3 测试3 12](#_Toc59371556)

[4.1.4 测试4 13](#_Toc59371557)

[4.1.5 测试5 13](#_Toc59371558)

[4.1.6 测试6 13](#_Toc59371559)

[4.2 错误测试 14](#_Toc59371560)

[4.2.1 含有非法字符 14](#_Toc59371561)

[4.2.2 左右括号中内容为空 14](#_Toc59371562)

[4.2.3 左右括号不匹配 14](#_Toc59371563)

[4.2.4 等号位置错误 15](#_Toc59371564)

[4.2.5 运算符不合理 15](#_Toc59371565)

1 分析

1.1 项目要求

* 从键盘上输入中缀算数表达式，包括括号，计算出表达式的值。
* 程序对所有输入的表达式作简单的判断，如表达式有错，能给出适当的提示。
* 支持包括加减，乘除取余，乘方和括号等操作符，其中优先级是等于<括号<加减<乘除取余<乘方。
* 能处理单目运算符：+或-。

1.2 功能分析

项目应实现对表达式的计算功能，包括加减，乘除取余，乘方和括号。

支持的数据类型：

* 整数（多位）
* 小数

若输入的表达式不合理，应给出相应的错误提示，包括以下几条：

* 出现了除了+-\* % / ^ ()=,数字以外的其他字符
* ()匹配不全或者()里面为空
* 表达式没有结束之前出现了=
* +-不是开头,或者前方不是( 一目运算符
* 运算符前方或者后方无数字

# 2 设计

## 2.1 数据结构设计

项目中定义了计算器类（Calculator），封装了表达式的数字、运算符与计算操作。还定义了一个运算符类（Sign），封装了操作符与其优先级。用两个栈分别保存数字与运算符。

## 2.2 成员与操作设计

**运算符类（Sign）**

1. **class** Sign {
2. **friend** **class** Calculator;
3. **public**:
4. Sign() {};
5. Sign(**char** signChar);
6. //初始化优先级
8. **bool** operator<(**const** Sign & rightSign) { **return** Priority < rightSign.Priority; }
9. //优先级比较
11. **char** getSign() { **return** m\_sign; }
12. //取运算符
13. **private**:
14. **char** m\_sign;
15. //运算符
17. **short** Priority;
18. //优先级
19. };

**计算器类（Calculator）**

1. **class** Calculator {
2. **public**:
3. Calculator() { SignStack.Push('@'); }
4. //将‘@’推入运算符栈（作为哨兵）
6. **bool** Claculate(**double** &result);
7. //输入表达式并计算，返回是否计算成功，result为运算结果
8. **private**:
9. Stack<**double**> Number;
10. //存数字
12. Stack<Sign> SignStack;
13. //存运算符
15. **bool** CalculateOnce();
16. //进行一次计算（处理一个运算符）
18. **double** pow(**double** &base, **double** &power);
19. //幂运算
20. };

## 2.3 系统设计

系统主体为一个循环，循环内定义一个Calculator类，让用户输入表达式并计算，若计算成功则输出结果，不成果输出相应的错误提示，最后让用户输入是否继续，若继续则进下一循环，不继续跳出循环，结束程序。

# 3 实现

## 3.1 输入表达式并计算**Claculate(…)**功能的实现

此函数为整个项目的主要函数，需要完成逐字符输入，根据输入的字符判断应进行操作，表达式不合理时给出错误提示。进行计算的思路是将当前操作符的优先级与上一操作符对比，若当前的优先级高则将其入栈，否则从栈中弹出两个数进行计算，并将结果入栈。

实现过程：首先让用户输入第一个字符，因为第一个字符可能是单目运算符，所以先单独判断是否是‘+’或‘-’，若是则向操作符栈（**SignStack**）中推入该操作符，并向数字栈中推入0（项目中处理单目运算符的方法是在单目运算符前补0，将其变为双目运算符）。

接下来输入下一个字符并进入循环，若该字符是数字，则继续输入字符，直到输入的不是数字或‘.’，并将前面的数字处理为double型推入数字栈（**Number**）。

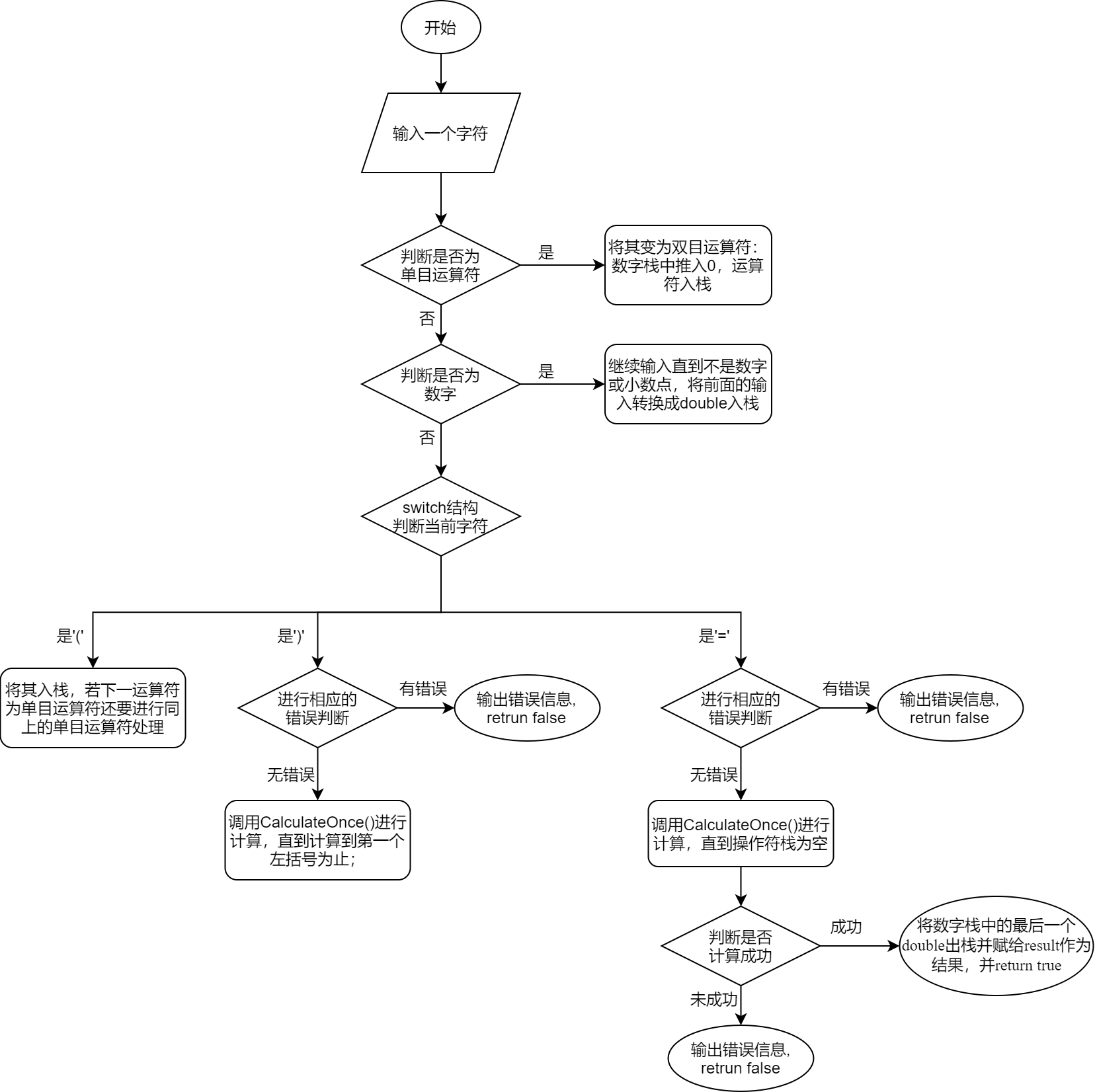
若该字符是左括号则将其推入运算符栈并输入下一字符，若下一字符是单目运算符还要进行同上的单目运算操作，最后进入下一循环（continue）。

若该字符是右括号，判断左括号的个数是否为0，若为0说明右括号多于左括号，直接给出错误提示并return false，再判断操作符栈中栈顶是否为右括号，若是说明存在空的括号，直接给出错误提示并return false，否则调用CalculateOnce()进行计算，直到计算到第一个左括号为止；

若该字符是等于号，判断下一个字符是否为‘\n’，若不是说明等号位置不在表达式末尾，直接给出错误提示并return false，再判断左括号数是否为零，若不是说明左括号数多于右括号数，直接给出错误提示并return false，否则调用CalculateOnce()进行计算直到操作符栈为空，若计算不成功说明有非法字符，直接给出错误提示并return false，若计算成功将数字栈中的最后一个double出栈并赋给result作为结果，并return true。

### 

### 3.1.1 输入表达式并计算功能流程图



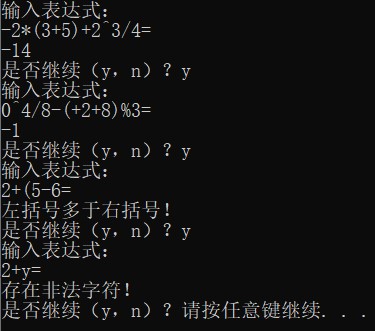
### 

### 3.1.2 输入表达式并计算功能代码

1. **bool** Calculator::Claculate(**double** &result) {
2. **char** ch;
3. **int** n = 0;//左括号的个数
4. cin >> ch;
5. **if** (ch == '+' || ch == '-') {
6. //单目运算符处理
7. Number.Push(0);
8. SignStack.Push(ch);
9. cin >> ch;
10. }
11. **while** (1) {
12. **if** (ch >= '0'&&ch <= '9') {
13. //数字处理
14. **double** num = 0;
15. **do** { num = num \* 10 + ch - 48; } **while** (cin >> ch, ch >= '0'&&ch <= '9');
16. **if** (ch == '.') {
17. **double** i = 10;
18. **while** (cin >> ch, ch >= '0'&&ch <= '9') {
19. num = num + (ch - 48) / i;
20. i \*= 10;
21. }//while
22. }//if
23. Number.Push(num);
24. }//if
25. **switch** (ch) {
26. **case** '(': {
27. n++;
28. SignStack.Push(ch);
29. cin >> ch;
30. **if** (ch == '+' || ch == '-') {
31. //单目运算符处理
32. Number.Push(0);
33. SignStack.Push(ch);
34. cin >> ch;
35. }
36. **continue**;
37. }
38. **case** ')': {
39. **if** (n-- <= 0) {
40. cout << "右括号多于左括号！" << endl;
41. **return** **false**;
42. }
43. **if** (SignStack.getFirst().getSign() == '(') {
44. cout << "存在空的括号！" << endl;
45. **return** **false**;
46. }
47. **while** (SignStack.getFirst().getSign() != '(') {
48. **bool** b = CalculateOnce();
49. **if** (!b) {
50. cout << "不合理的运算符！" << endl;
51. **return** **false**;
52. }
53. }
54. SignStack.Pop();
55. **break**;
56. }
57. **case** '=': {
58. **if** (cin.get() != '\n') {
59. cout << "等号位置错误！" << endl;
60. **return** **false**;
61. }
62. **if** (n != 0) {
63. cout << "左括号多于右括号！" << endl;
64. **return** **false**;
65. }
66. **while** (SignStack.getFirst().getSign() != '@') {
67. **bool** b = CalculateOnce();
68. **if** (!b) {
69. cout << "不合理的运算符！" << endl;
70. **return** **false**;
71. }
72. }
73. result = Number.Pop();
74. **return** **true**;
75. }
76. **default**: {
77. Sign sign(ch);
78. **if** (sign.Priority == 6) {               //其他字符的优先级为6
79. cout << "存在非法字符！" << endl;
80. **return** **false**;
81. }
82. **while** (!(sign < SignStack.getFirst())) {//当前优先级不高就计算
83. **bool** b = CalculateOnce();
84. **if** (!b) {
85. cout << "不合理的运算符！" << endl;
86. **return** **false**;
87. }
88. }
89. SignStack.Push(sign);
90. }
91. }//switch
92. cin >> ch;
93. }
94. }

### 

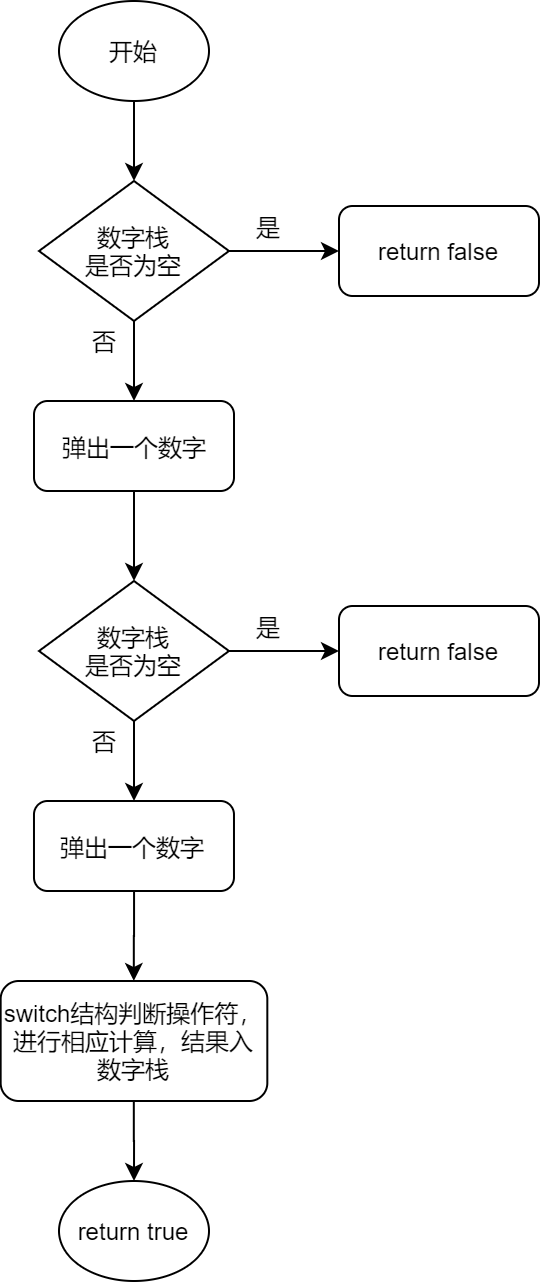
### 3.1.3 输入表达式并计算功能截屏示例



## 3.2 进行一次计算**CalculateOnce ()**功能的实现

* 从数字栈中弹出两个数字，从操作符栈中弹出两个操作符进行计算
* 计算结果入数字栈，return true
* 若数字栈为空说明表达式有错误，retrun false
* 被**Calculate(…)**函数调用

### 3.2.1 进行一次计算功能流程图



### 3.2.2 进行一次计算代码

1. **bool** Calculator::CalculateOnce() {
2. **if** (Number.IsEmpty()) {
3. **return** **false**;
4. }
5. **double** num2 = Number.Pop();
6. **if** (Number.IsEmpty()) {
7. **return** **false**;
8. }
9. **double** num1 = Number.Pop();
10. **switch** (SignStack.Pop().getSign()) {
11. **case** '+':
12. Number.Push(num1 + num2);
13. **break**;
14. **case** '-':
15. Number.Push(num1 - num2);
16. **break**;
17. **case** '\*':
18. Number.Push(num1 \* num2);
19. **break**;
20. **case** '/':
21. Number.Push(num1 / num2);
22. **break**;
23. **case** '%':
24. Number.Push((**int**)num1 % **int**(num2));
25. **break**;
26. **case** '^':
27. Number.Push(pow(num1, num2));
28. **break**;
29. }
30. **return** **true**;
31. }

## 3.3 幂运算**pow(…)**功能的实现

* base为底数，power为指数
* 被**CalculateOnce ()**函数调用

### 3.3.1 幂运算功能代码

1. **double** Calculator::pow(**double** &base, **double** &power) {
2. **double** result = 1;
3. **if** (base == 0) { **return** 0; }
4. **if** (power > 0) {
5. **while** (power-- > 0) { result \*= base; }
6. }
7. **else** **if** (power < 0) {
8. **while** (power++ < 0) { result /= base; }
9. }
10. **return** result;
11. }

## 3.4 **Sign**类构造函数的实现

* **signChar**为操作符
* 在Calculator的构造函数中，将‘@’推入操作符栈，作为哨兵元素

### 3.4.1 **Sign**类构造函数代码

1. Sign::Sign(**char** signChar) :m\_sign(signChar) {
2. **switch** (signChar)
3. {
4. **case** '+':
5. Priority = 3;
6. **break**;
7. **case** '-':
8. Priority = 3;
9. **break**;
10. **case** '\*':
11. Priority = 2;
12. **break**;
13. **case** '/':
14. Priority = 2;
15. **break**;
16. **case** '%':
17. Priority = 2;
18. **break**;
19. **case** '^':
20. Priority = 1;
21. **break**;
22. **case** '@':
23. Priority = 5;
24. **break**;
25. **case** '(':
26. Priority = 4;
27. **break**;
28. **case** ')':
29. Priority = 4;
30. **break**;
31. **default**:
32. Priority = 6;
33. **break**;
34. }
35. }

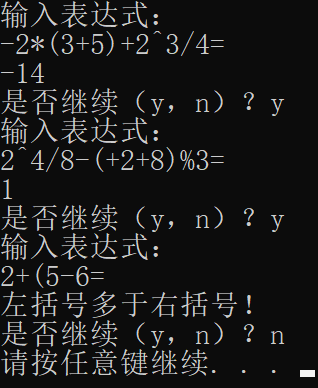
## 3.6 主函数的实现

主函数主体为一个循环，循环内定义一个Calculator类，让用户输入表达式并计算，若计算成功则输出结果，不成果输出相应的错误提示，最后让用户输入是否继续，若继续则进下一循环，不继续跳出循环，结束程序。

### 3.6.1 总体系统核心代码

1. **int** main() {
2. **char** stop = 'y';
3. **bool** b;
4. **double** result;
5. **while** ('y' == stop) {
6. Calculator Calculator1;
7. cout << "输入表达式：" << endl;
8. b = Calculator1.Claculate(result);
9. **if** (b) {
10. cout << result << endl;
11. }
12. **else**{
13. cin.sync();
14. }
15. cout << "是否继续（y，n）？";
16. cin >> stop;
17. }
18. system("pause");
19. **return** 0;
20. }

### 3.6.2 主函数截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 测试1

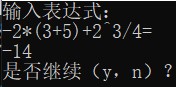
**测试用例**：

-2\*(3+5)+2^3/4=

**预期结果**：

-14

**实验结果**



### 4.1.2 测试2

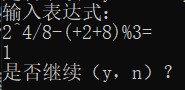
**测试用例**：

-2\*(3+5)+2^3/4=

**预期结果**：

1

**实验结果**

****

### 4.1.3 测试3

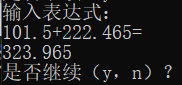
**测试用例**：

101.5+222.465=

**预期结果**：

323.965

**实验结果**

****

### 4.1.4 测试4

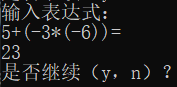
**测试用例**：

5+(-3\*(-6))=

**预期结果**：

23

**实验结果**

****

### 4.1.5 测试5

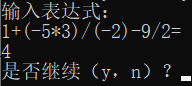
**测试用例**：

1+(-5\*3)/(-2)-9/2=

**预期结果**：

4

**实验结果**

****

### 4.1.6 测试6

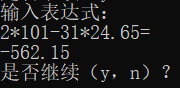
**测试用例**：

2\*101-31\*24.65=

**预期结果**：

-562.15

**实验结果**

****

## 4.2 错误测试

### 4.2.1 含有非法字符

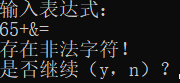
**测试用例：**

65+&=

**预期结果：**

给出错误提示，程序运行正常不崩溃。

**实验结果：**



### 4.2.2 左右括号中内容为空

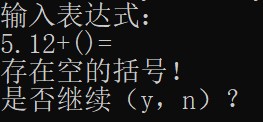
**测试用例：**

5.12+()=

**预期结果：**

给出错误提示，程序运行正常不崩溃。

**实验结果：**

****

### 4.2.3 左右括号不匹配

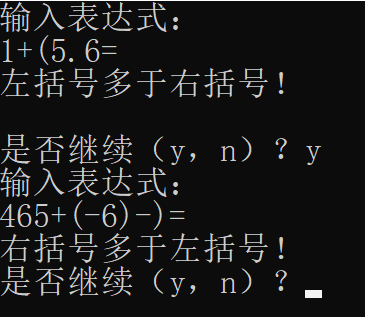
**测试用例：**

1+(5.6=

**预期结果：**

给出错误提示，程序运行正常不崩溃。

**实验结果：**

****

### 4.2.4 等号位置错误

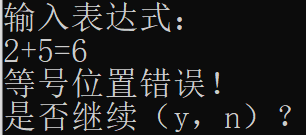
**测试用例：**

2+5=6

**预期结果：**

给出错误提示，程序运行正常不崩溃。

**实验结果：**

****

### 4.2.5 运算符不合理

**测试用例：**

6+%6=

**预期结果：**

给出错误提示，程序运行正常不崩溃。

**实验结果：**

