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： 曹峰源

学 号： 1951328

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 3](#_Toc57652055)

[2. 设计 3](#_Toc57652056)

[2.1 类结构设计 3](#_Toc57652057)

[2.2 类成员 3](#_Toc57652058)

[3. 核心代码功能 4](#_Toc57652059)

[3.1 将中缀表达式转后缀表达式 4](#_Toc57652060)

[3.2 计算功能实现 7](#_Toc57652061)

[3.3 主函数功能 10](#_Toc57652062)

[4. 功能测试 11](#_Toc57652063)

[4.1 正常测试 11](#_Toc57652064)

[4.2 单目运算符测试 12](#_Toc57652065)

[5. 出错测试 12](#_Toc57652066)

[6. 改进方向 12](#_Toc57652067)

1. 分析

我们希望编写一段程序能够实现计算器的功能，但程序无法识辨输入字符的区别，需要人为使它进行判断。所采取的是通过将中缀表达式先转化为后缀表达式，对其进行计算，并且能够满足单目运算符，最后希望能够对算数表达式正误有一定的判断能力。

1. 设计
   1. 类结构设计

为了实现要求，定义了node结点使用模板，另外使用stack类用于后缀表达式的计算。linklist类用于存储输入的字符表达式。最后定义calculator类来进行计算操作。

* 1. 类成员

结点node类：

私有成员：

private:

//data是需要的数据，icp与isp在表达式操作符中缀转后缀时使用

T data = '0';

int isp = 0;

int icp = 0;

node<T>\* next = NULL;

栈stack类：

私有成员：

node<int>\* top;//top指针

公有成员函数：

stack(){}//构造函数

} ~stack() {}//析构函数

void push(int p);//栈弹入函数

int pop();//栈弹出函数

链表linklist类：

私有成员：

node<char>\* first;//头指针

node<char>\* current;//用于移动的current指针

公有成员函数：

linklist(){}//构造函数

~linklist(){}//析构函数

void insert(char p);//用于在链表最后位置加入一个元素

void del();//用于将链表最后一个元素删除，current指针指向删除后链表最后的元素

void establish();//构造链表

node<char>\* getfirst();//这个函数返回链表的头指针

计算calculator类：

私有成员：

stack s;//定义栈s

linklist h;//定义链表h

公有成员函数：

calculator() {};//构造函数

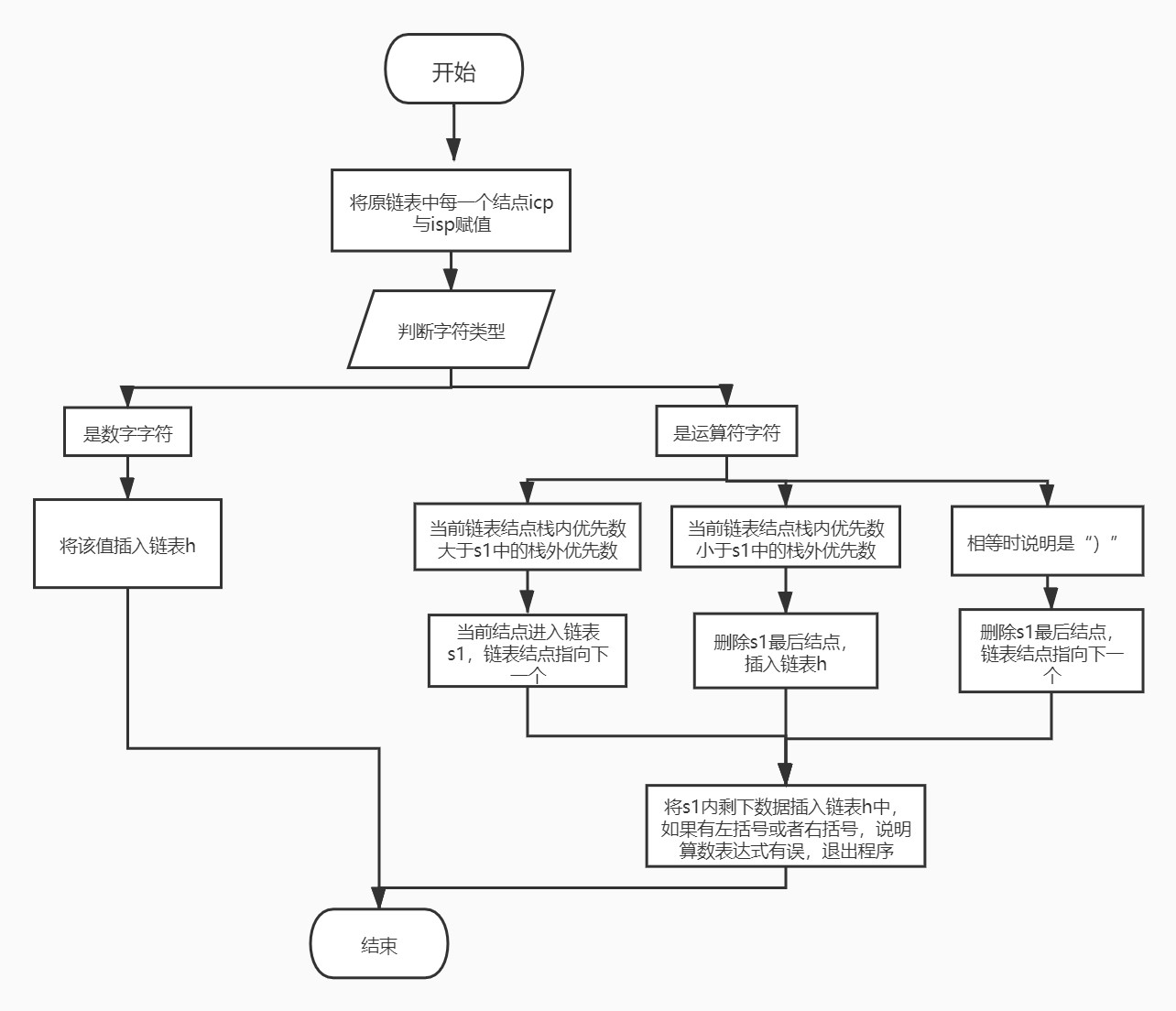
~calculator() {};//析构函数

bool remix(node<char>\* t);//用于将中缀表达式调整为后缀表达式存入链表h

void calculate();//计算表达式的值

1. 核心代码功能
   1. 将中缀表达式转后缀表达式

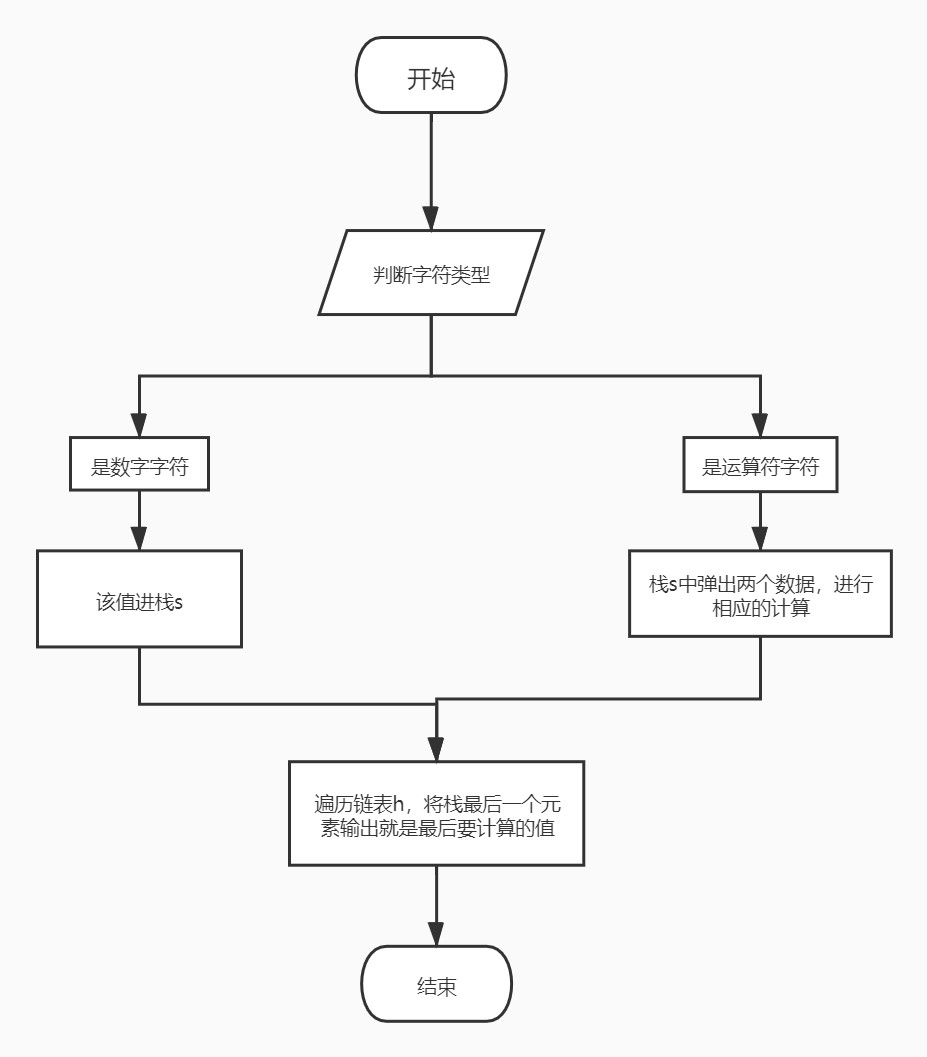
流程图：



代码实现：

1. **bool** calculator::remix(node<**char**>\* t)
2. {
3. node<**char**>\* p;
4. p = t->next;
5. //将每一个结点的icp（栈内优先数）与isp（栈外优先数）赋值
6. **while** (p != NULL)
7. {
8. **if** (p->data <= '9' && p->data >= '0') {}
9. **else**
10. {
11. **switch** (p->data)
12. {
13. **case** '+': {
14. p->icp = 2;
15. p->isp = 3;
16. **break**;
17. }
18. **case**'-': {
19. p->icp = 2;
20. p->isp = 3;
21. **break**;
22. }
23. **case**'\*': {
24. p->icp = 4;
25. p->isp = 5;
26. **break**;
27. }
28. **case**'/': {
29. p->icp = 4;
30. p->isp = 5;
31. **break**;
32. }
33. **case**'%': {
34. p->icp = 4;
35. p->isp = 5;
36. **break**;
37. }
38. **case**'^': {
39. p->icp = 6;
40. p->isp = 7;
41. **break**;
42. }
43. **case**'(': {
44. p->icp = 8;
45. p->isp = 1;
46. **break**;
47. }
48. **case**')': {
49. p->icp = 1;
50. p->isp = 8;
51. **break**;
52. }
53. }
54. }
55. p = p->next;
56. }
57. p = t->next;
58. linklist s1;//建立一个链表s1，用于临时存放数据
59. **while** (p != NULL)
60. {
61. //对中缀表达式进行遍历，如果是数值，就进入h链表
62. **if** (p->data <= 57 && p->data >= 48)
63. {
64. h.insert(p->data);
65. p = p->next;
66. }
67. //如果是运算字符的情况
68. **else**
69. {
70. //如果当前结点栈内优先数大于s1中的栈外优先数，进s1栈
71. **if** (p->icp > s1.current->isp)
72. {
73. s1.insert(p->data);
74. s1.current->icp = p->icp;
75. s1.current->isp = p->isp;
76. p = p->next;
77. }
78. //如果小于，s1当前退栈,h进栈
79. **else** **if** (p->icp < s1.current->isp)
80. {
81. **char** d = s1.current->data;
82. s1.del();
83. h.insert(d);
84. }
85. //等于的情况下说明是括号出现了，将其删除，后缀表达式中不会出现括号
86. **else**
87. {
88. s1.del();
89. p = p->next;
90. }
91. }
92. }
93. //将s1剩余的元素删掉进入h栈中，如果出现有括号则说明输入的表达式有误
94. **while** (s1.current != s1.first)
95. {
96. **if** (s1.current->data=='('||s1.current->data==')')
97. {
98. **return** **false**;
99. }
100. **char** d = s1.current->data;
101. h.insert(d);
102. s1.del();
103. }
104. **return** **true**;
105. }
     1. 计算功能实现

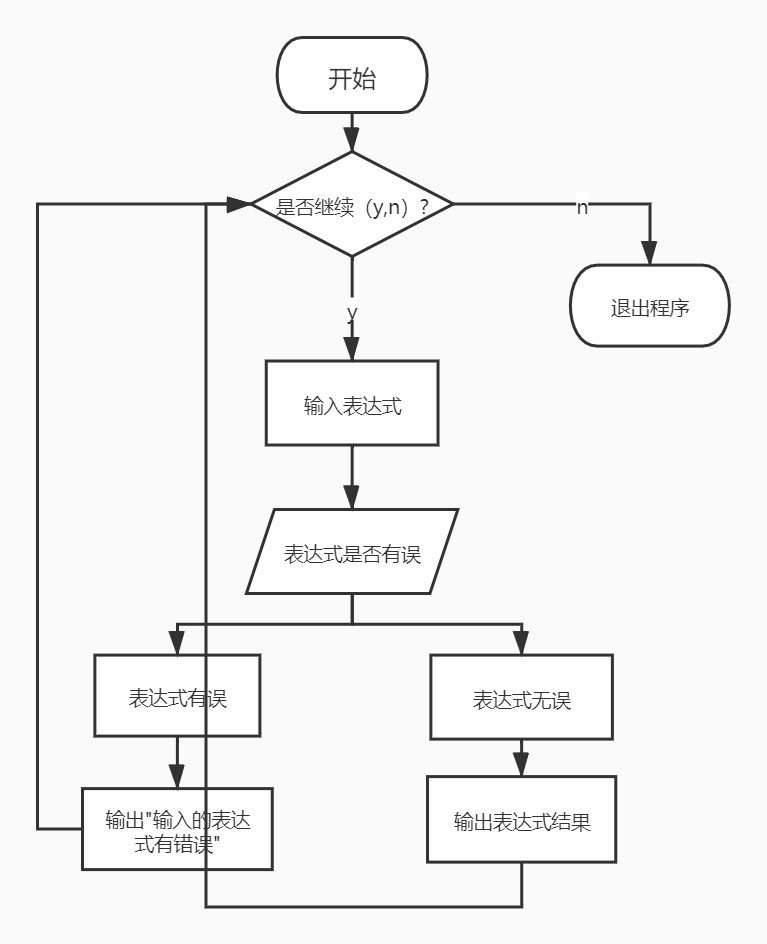
流程图：



代码实现：

1. **void** calculator::calculate()//用于计算得到的后缀表达式
2. {
3. node<**char**>\* p = **new** node<**char**>;
4. p = h.first->next;
5. **while** (p != NULL)
6. {
7. //如果是运算符，则将栈s中两个元素弹出，进行相应的计算
8. **if** (p->data == '+' || p->data == '-' || p->data == '\*' || p->data == '/' || p->data == '%' || p->data == '^')
9. {
10. **switch** (p->data)
11. {
12. **case** '+':
13. {
14. **int** number1 = s.pop(), number2 = s.pop();
15. **int** number3;
16. number3 = number1 + number2;
17. s.push(number3);
18. **break**;
19. }
20. **case** '-':
21. {
22. **int** number1 = s.pop(), number2 = s.pop();
23. **int** number3;
24. number3 = number2 - number1;
25. s.push(number3);
26. **break**;
27. }
28. **case** '\*':
29. {
30. **int** number1 = s.pop(), number2 = s.pop();
31. **int** number3;
32. number3 = number1 \* number2;
33. s.push(number3);
34. **break**;
35. }
36. **case** '/':
37. {
38. **int** number1 = s.pop(), number2 = s.pop();
39. **int** number3;
40. number3 = number2 / number1;
41. s.push(number3);
42. **break**;
43. }
44. **case** '%':
45. {
46. **int** number1 = s.pop(), number2 = s.pop();
47. **int** number3;
48. number3 = number2 % number1;
49. s.push(number3);
50. **break**;
51. }
52. **case** '^':
53. {
54. **int** number1 = s.pop(), number2 = s.pop();
55. **int** number3;
56. number3 =pow( number2 ,number1);
57. s.push(number3);
58. **break**;
59. }
60. }
61. }
62. //如果是数值，则进s栈
63. **else** {
64. s.push(p->data - 48);
65. }
66. p = p->next;
67. }
68. cout << s.top->data<<endl;
69. }
    1. 主函数功能

流程图：

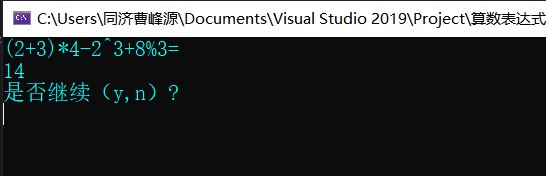


代码实现：

1. **int** main()
2. {
3. linklist g;
4. calculator cal;
5. g.establish();
6. node<**char**>\* j = **new** node<**char**>;
7. j = g.getfirst();
8. **if** (cal.remix(j) == 1)
9. {
10. cal.calculate();
11. }
12. **else**
13. {
14. cout << "输入的表达式有错误" << endl;
15. }
16. cout << "是否继续（y,n）?" << endl;
17. **char** judge;
18. **while** (cin>>judge)//while循环来计算表达式
19. {
20. **if** (judge=='y')
21. {
22. g.establish();
23. node<**char**>\* j;
24. j = g.getfirst();
25. **if** (cal.remix(j)==1)//如果remix返回值为1的话说明表达式没有错误，进行计算
26. {
27. cal.calculate();
28. }
29. **else** {//如果是0，说明表达式有误
30. cout << "输入的表达式有错误" << endl;
31. }
32. }
33. **else**
34. {
35. cout << "运算结束" << endl;
36. **break**;
37. }
38. cout << "是否继续（y,n）?" << endl;
39. }
40. **return** 0;
41. }
42. 功能测试
    1. 正常测试

测试用例：(2+3)\*4-2^3+8%3=

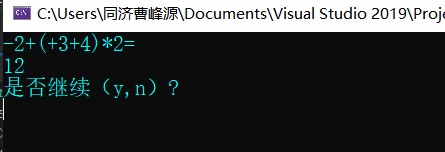
结果：



* 1. 单目运算符测试

测试用例：-2+(+3+4)\*2=

结果：

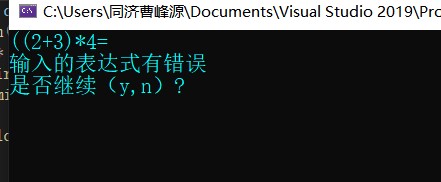


1. 出错测试

该程序只对括号不匹配的情况下进行报错

测试用例：((2+3)\*4=

结果：



1. 改进方向

对于这个程序的实现还有些不足的地方。

首先是没有加入对小数以及多位数的计算，如果想要实现的话需要加入对小数点“.”的判断，想要实现多位数的运算需要对数字进行位数上的判断。

其次是对表达式有误判断功能比较薄弱，只能对括号不匹配的情况进行一些判断，其他情况程序不能做出很好的反映，需要继续改进。