C++ 语 言 程 序 设 计

实

验

报

告

实验二

姓名: _____方尧_____

学号: _____190410102 ____

班级: 19 自动化 1 班

一 实验项目

- 1、熟悉 C++程序设计
- 2、 掌握 C++基本输入输出方法
- 3、 掌握 C++中 string 类型的使用方法
- 4、实现模板栈功能
- 5、实现表达式中求值

二 实验原理

● 类模板

定义类模板 stack, oper 对象是操作符栈,为 int 型,初始化为 0, position_oper 为其栈顶位置,初始化为 0; number 对象为操作数栈,为 double 型, position_number 为其栈顶位置,初始化为 0;

● 优先级

order[11] = { 0,3,0,3,0,3,0,1,1,2,2 }下标为 1-10 对应 '{}[]()+-*/'

● 数组的预处理及作用说明:

command[length]用于存储接收进来的字符表达式,初始化全为'=',为 char 型 num[length]用于存储提取出的数值,初始化全为 0,为 double 型 brackets[length]用于存放提取出的操作符,初始化为 0,为 int 型,值与字符对应 关系为:

1-'{', 2-'[', 3-'(', 4-'}', 5-']', 6-')', 7-'+', 8-'-', 9-'*', 10-'/', 11-'.'

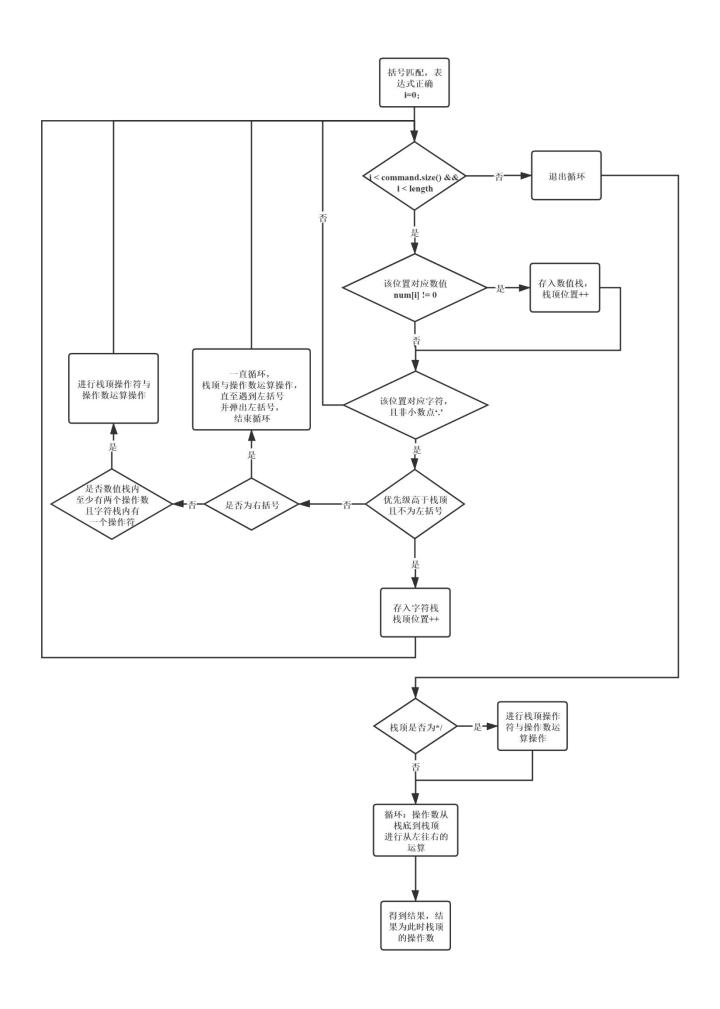
● 用到的 ASCII 码表:

figure 1 ASCII 码表

字符	ASCII 码						
{	123	}	125	0	48	5	53
[91]	93	1	49	6	54
(40)	41	2	50	7	55
+	43	-	45	3	51	8	56
*	42	/	47	4	52	9	57
	46						

1、 说明表达式求值的方法(给出算法的**流程图**与简要说明)

主要利用优先级关系,配合栈类,建立数值栈和操作符栈,以及左括号优先级最高,循环计算表达式。循环完成,判断最后是否为乘除,计算。最后从左至右计算加减,得到结果。实现代码封装在 calcualte result()函数内。



- 2、给出程序的源代码,说明关键代码的操作含义,给出运行结果
- A. 源代码见附录。
- B. 操作及代码说明:

reset()函数,每次输入命令前重置 command,num,brackets

former()函数,数值整数部分操作函数

later(),数值小数部分操作函数

spl()函数,识别操作符,并将操作符信息存入 brackets 中;调用 former()和 later()函数,识别提取数值,并存入 num 中

cal()函数,判断括号匹配,若错误,输出错误信息

主要利用字符与 ASCII 码间的关系,识别字符串中的运算符以及数值。

outputformat()函数,格式化输出错误信息

calcualte_result()函数,括号匹配正确后的表达式计算函数

calcualte simple()函数,实现二操作数加减乘除

● 定义类模板栈

```
    template < class type >

2. class stack
3. {
4. public:
5.
        stack(type a)
6.
7.
             for (int i = 0; i < length; i++)</pre>
8.
9.
                 link[i] = a;
10.
11.
12.
        type link[length];
13. };
```

● 建立操作符栈和操作数栈

```
14. //操作符栈
15. stack <int> oper(0);
16. int position_oper = 0;
17. //操作数栈
18. stack <double> number(0);
19. int position_number = 0;
```

● 栈顶操作符与操作数计算

● 从左至右计算

C. 运行结果

运行结果均正确

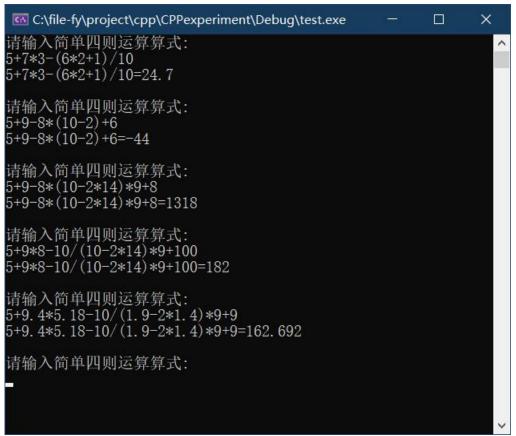


figure 1 运行结果截图

三 实验总结与建议

实验实施过程: 先构思; 画框架图; 编写伪代码; 用实际代码实现; 运行调试; debug 直至无错误并得到预期结果。

实验解决方案: 定义栈类,建立操作数(double型)和操作符栈(int型),实现运算符计算。

附录:源代码

```
28. #include<iostream>
29. #include<math.h>
30. #include<string>
31. #define length 100
32. using namespace std;
33.
34. template<class type>
35. class stack
36. {
37. public:
38.
        stack(type a)
39.
40.
            for (int i = 0; i < length; i++)</pre>
41.
42.
                link[i] = a;
43.
44.
45.
        type link[length];
46.};
47.
48. void outputformat(string command, double num[], int brackets[], int n = length);
49. void spl(string command, double num[], int brackets[]);
50. void calcualte(string command, double num[], int brackets[]);
51. double calcualte_result(string command, double num[], int brackets[]);
52. double calcualte_simple(int n, double a, double b);
53. void reset(string command, double num[], int brackets[]);
54. int former(int i, string command, double num[]);
55. int later(int i, int j, string command, double num[]);
57. int main()
58. {
59.
        string command;
60.
        double num[length] = { 0 };
        int brackets[length] = { 0 }; //1-{,2-[,3-(,4-},5-],6-)
61.
62.
        for (;;)
63.
64.
            reset(command, num, brackets);
65.
            cout << "请输入简单四则运算算式:" << endl;
66.
            cin >> command;
67.
            spl(command, num, brackets);
68.
            calcualte(command, num, brackets);
69.
            cout << endl;</pre>
70.
71.
        return 0;
72.}
73. void reset(string command, double num[], int brackets[])
74. {
75.
        int i, j;
76.
        for (int i = 0; i < length; i++)</pre>
77.
78.
            command = "";
79.
            num[i] = 0;//num, brackets 默认初值为 0,'0'
80.
            brackets[i] = 0;
81.
82.}
83. void spl(string command, double num[], int brackets[])
84. {
85.
        for (int i = 0; i < command.size() && i < length; i++)</pre>
86.
87.
            switch (command[i])
88.
            {
89.
            case '{':
90.
                brackets[i] = 1;
91.
                break;
92.
            case '}':
```

```
93.
                brackets[i] = 2;
94.
                break;
95.
            case '[':
96.
                brackets[i] = 3;
97.
                break;
98.
            case ']':
99.
                brackets[i] = 4;
100.
                 break;
101.
             case '(':
102.
                 brackets[i] = 5;
                 break;
103.
104.
             case ')':
105.
                 brackets[i] = 6;
106.
                 break;
             case '+':
107.
108.
                 brackets[i] = 7;
109.
                 break;
             case '-':
110.
111.
                 brackets[i] = 8;
112.
                 break;
             case '*':
113.
114.
                 brackets[i] = 9;
115.
                 break;
             case '/':
116.
117.
                 brackets[i] = 10;
                 break;
118.
119.
             case '.':
120.
                 brackets[i] = 11;
121.
                 break;
122.
             }
123.
124.
         for (int i = 0; i < command.size() && i < length; i++)</pre>
125.
126.
             int j, k;
127.
             if (int(command[i]) >= 48 && int(command[i]) <= 57)</pre>
128.
                 num[i] = command[i] - '0';
129.
130.
                 j = former(i, command, num);
131.
                 k = later(i, j, command, num);
132.
                 if (k != 0)
133.
                     i = k;
134.
                 else
135.
136.
                      i = j;
                     cout << "wrong\n";</pre>
137.
138.
                      /*错误输出*/
139.
140.
             }
141.
142.}
143. //小数点前数字处理
144.int former(int i, string command, double num[])
145.{
146.
         int j;
         for (j = i + 1; j < command.size() && j < length; j++)
147.
148.
149.
             if (int(command[j]) >= 48 && int(command[j]) <= 57)</pre>
150.
             {
151.
                 num[i] = num[i] * 10 + command[j] - '0';
152.
             }
153.
             else
154.
                 break;
155.
         }
156.
         return j;
157.}
158.
159. //小数点后的数字处理
160.int later(int i, int j, string command, double num[])
161. {
```

```
162.
         int k = 0;
163.
        if (int(command[j]) == 46)
164.
             if (!(command[j + 1] >= 48 \&\& command[j + 1] <= 57))
165.
166.
                 return 0;
167.
             else
168.
             {
169.
                 for (int k = j + 1; k < command.size() && k < length; <math>k++)
170.
171.
                     if (int(command[k]) >= 48 && int(command[k]) <= 57)</pre>
172.
                     {
                         //cout<<num[i]<<"\t"<<command[k]-'0'<<"\t"<<double(pow(10,k-j))<<endl;
173.
174.
                         num[i] = num[i] + (command[k] - '0') / double(pow(10, k - j));
175.
                          //最后为数字结尾
176.
                          if (k == command.size() - 1) return command.size() - 1;
177.
                     }
178.
                     else
179.
                         return k;
180.
                 }
181.
182.
         }
        else
183.
184.
             return j;
185.
        return j;
186.}
187.
188. void calcualte(string command, double num[], int brackets[])
189. {
190.
         int wrong = 0;
         //left 从第二个存储位置开始存
191.
192.
         stack <int> left('0');
193.
         int position = -1;
194.
         //判断括号是否匹配
195.
        for (int i = 0; i < command.size() && i < length; i++)</pre>
196.
197.
             if (brackets[i] == 1 || brackets[i] == 3 || brackets[i] == 5)
198.
199.
                 left.link[position + 1] = brackets[i];
200.
                 position++;
201.
             if (brackets[i] == 2 || brackets[i] == 4 || brackets[i] == 6)
202.
203.
204.
                 if (position == -1) { wrong = i; break; }
205.
                 if (brackets[i] == 2)
206.
207.
                     if (left.link[position] != 1)
208.
209.
                         wrong = i;
210.
                         break;
211.
                     }
212.
                     else
213.
                     {
214.
                          left.link[position] = 0;
215.
                         position--;
216.
217.
                 }
                 if (brackets[i] == 4)
218.
219.
220.
                     if (left.link[position] != 3)
221.
222.
                         wrong = i;
223.
                         break;
224.
                     }
225.
                     else
226.
                     {
227.
                         left.link[position] = 0;
228.
                         position--;
229.
```

```
230.
231.
                 if (brackets[i] == 6)
232.
233.
                     if (left.link[position] != 5)
234.
235.
                         wrong = i;
236.
                         break;
237.
                     }
238.
                     else
239.
                     {
240.
                         left.link[position] = 0;
241.
                         position--;
242.
                     }
243.
                 }
244.
245.
246.
         if (wrong != 0)
247.
        {
248.
             outputformat(command, num, brackets, wrong);
249.
            cout << "匹配错误" << endl;
250.
         }
251.
        else
252.
         {
253.
            if (position == -1)//正确,输出计算结果
254.
255.
                 double result = calcualte result(command, num, brackets);
256.
                 if (result == 999.99)
257.
                     cout << "wrong" << endl;</pre>
258.
                 else
259.
260.
                     cout << command;</pre>
261.
                     cout << "=" << result << endl;</pre>
262.
                 }
263.
             }
264.
             else
265.
             {
                 outputformat(command, num, brackets);
266.
267.
                 cout << "无法匹配" << endl;
268.
             }
269.
270.}
271.double calcualte_result(string command, double num[], int brackets[])
272.{
273.
         int order[11] = { 0,3,0,3,0,3,0,1,1,2,2 };//加减乘除 7,8,9,10
274.
        double result = 999.99;
275.
276.
         //操作符栈
277.
        stack <int> oper(0);
278.
         int position_oper = 0;
279.
        //操作数栈
         stack <double> number(0);
280.
281.
        int position_number = 0;
282.
        for (int i = 0; i < command.size() && i < length; i++)</pre>
283.
284.
             if (num[i] != 0)//数值数组非 0 存入
285.
286.
                 number.link[position_number + 1] = num[i];
287.
                 position_number++;
288.
289.
            if (brackets[i] != 0 && brackets[i] != 11)//操作符数组非 0 且非.存入
290.
             {
291.
                 if (order[oper.link[position_oper]] < order[brackets[i]] || order[oper.link[p</pre>
   osition_oper]] == 3)//优先级高于现栈顶优先级,压入
292.
293.
                     oper.link[position_oper + 1] = brackets[i];
294.
                     position_oper++;
295.
                 }
296.
                 else
297.
                 {
```

```
298.
                     if (brackets[i] == 2 || brackets[i] == 4 || brackets[i] == 6)
299.
300.
                         for (;;)
301.
                         {
                              if (oper.link[position_oper] == brackets[i] - 1)
302.
303.
304.
                                  position_oper--;
305.
                                  break;
306.
                              }
307.
                              else
308.
                              {
309.
                                  number.link[position number - 1] = calcualte simple(oper.link
    [position_oper], \
310.
                                      number.link[position number - 1], number.link[position nu
    mber]);
311.
                                  position_number--; position_oper--;
312.
313.
314.
                     }
315.
                     else
316.
                         if (position_number > 1)
317.
318.
319.
                             if (position_oper > 0)
320.
                                  number.link[position number - 1] = calcualte simple(oper.link
321.
   [position_oper], \
322.
                                      number.link[position_number - 1], number.link[position_nu
    mber]);
323.
                                  position_number--; position_oper--;
324.
                              }
325.
326.
                         oper.link[position_oper + 1] = brackets[i];
327.
                         position_oper++;
328.
                     }
329.
330.
             }
331.
332.
         if (oper.link[position_oper] == 9 || oper.link[position_oper] == 10)
333.
334.
             number.link[position_number - 1] = calcualte_simple(oper.link[position_oper], \
335.
                 number.link[position_number - 1], number.link[position_number]);
336.
             position_number--; position_oper--;
337.
338.
         for (int i = 1; i <= position_oper; i++)</pre>
339.
340.
             number.link[i + 1] = calcualte_simple(oper.link[i], \
341.
                 number.link[i], number.link[i + 1]);
342.
343.
        result = number.link[position_number];
344.
         return result;
345.}
346.double calcualte_simple(int n, double a, double b)
347.{
348.
         if (n == 7)return a + b;
349.
        if (n == 8)return a - b;
350.
         if (n == 9)return a * b;
351.
        if (n == 10 && b != 0)return a / b;
        return 0;
352.
353.}
354.void outputformat(string command, double num[], int brackets[], int n)
355. {
356.
         for (int i = 0; i < command.size() && i < length && i <= n; i++)</pre>
357.
358.
             if (num[i] != 0)
359.
             {
360.
                 cout << "操作数:\t" << num[i] << endl;
361.
362.
             if (brackets[i] != 0 && brackets[i] != 11)
```

```
363. {
364.
               switch (brackets[i])
365.
               {
366.
                case 1:
                   cout << "操作符:\t 左大括号\t";
367.
368.
                   break;
369.
               case 2:
370.
                   cout << "操作符:\t 右大括号\t";
371.
                   break;
372.
                case 3:
373.
                   cout << "操作符:\t 左中括号\t";
374.
                   break;
375.
                case 4:
                   cout << "操作符:\t 右中括号\t";
376.
377.
                   break;
378.
                case 5:
379.
                   cout << "操作符:\t 左小括号\t";
380.
                   break;
381.
                case 6:
382.
                   cout << "操作符:\t 右小括号\t";
383.
                   break;
384.
               case 7:
385.
                   cout << "操作符:\t 加号\t";
                   break;
386.
387.
                   cout << "操作符:\t 减号\t";
388.
389.
                   break;
390.
                case 9:
391.
                   cout << "操作符:\t 乘号\t";
392.
                   break;
393.
                case 10:
394.
                   cout << "操作符:\t 除号\t";
395.
                   break;
396.
397.
               if (i != n)cout << endl;</pre>
398.
            }
399.
400.}
```