**C++ 语 言 程 序 设 计**

实

验

报

告

实 验 二

姓名： 葛旭

学号： 190320517

班级： 自动化5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**

1. 说明表达式求值的方法（给出算法的**流程图**与简要说明）

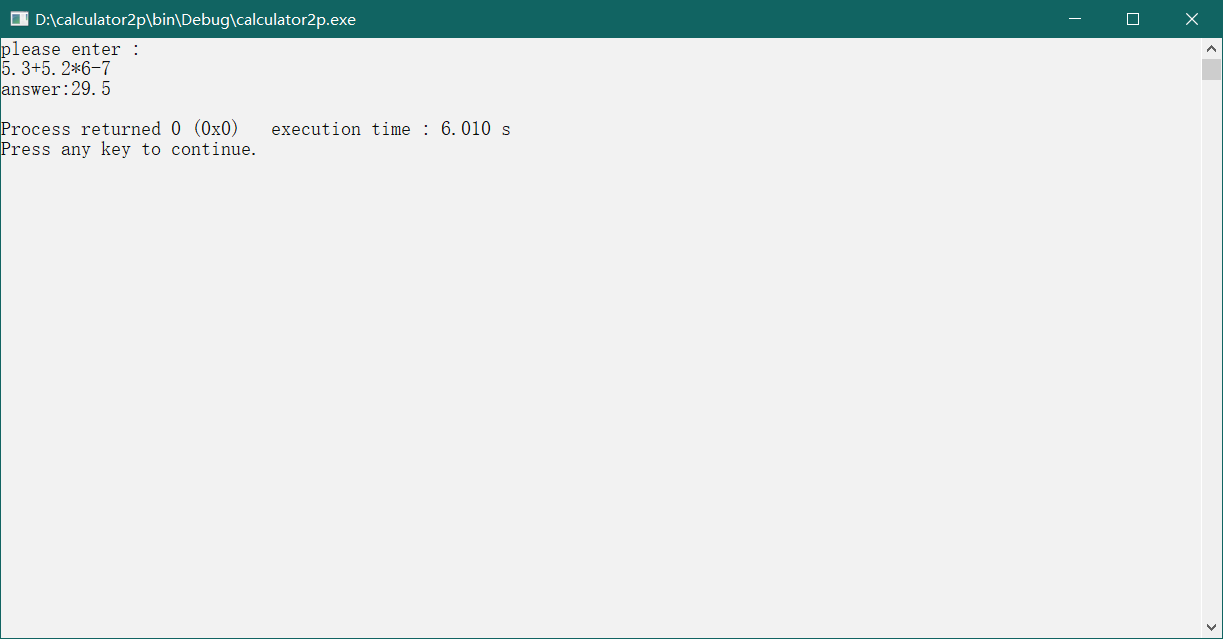


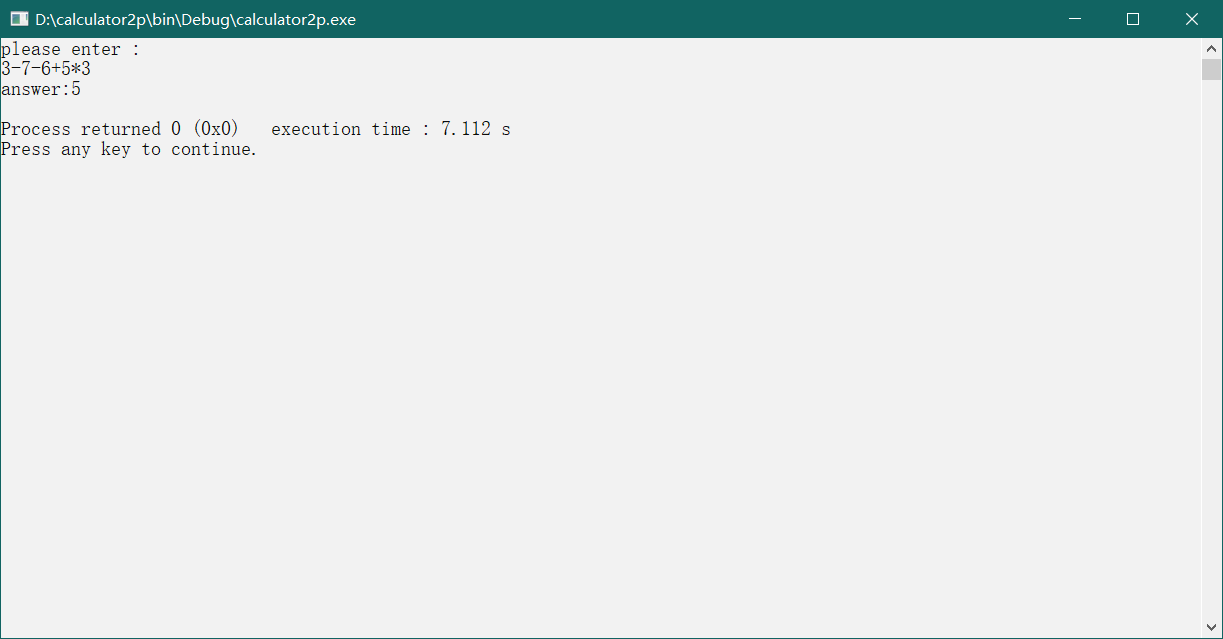


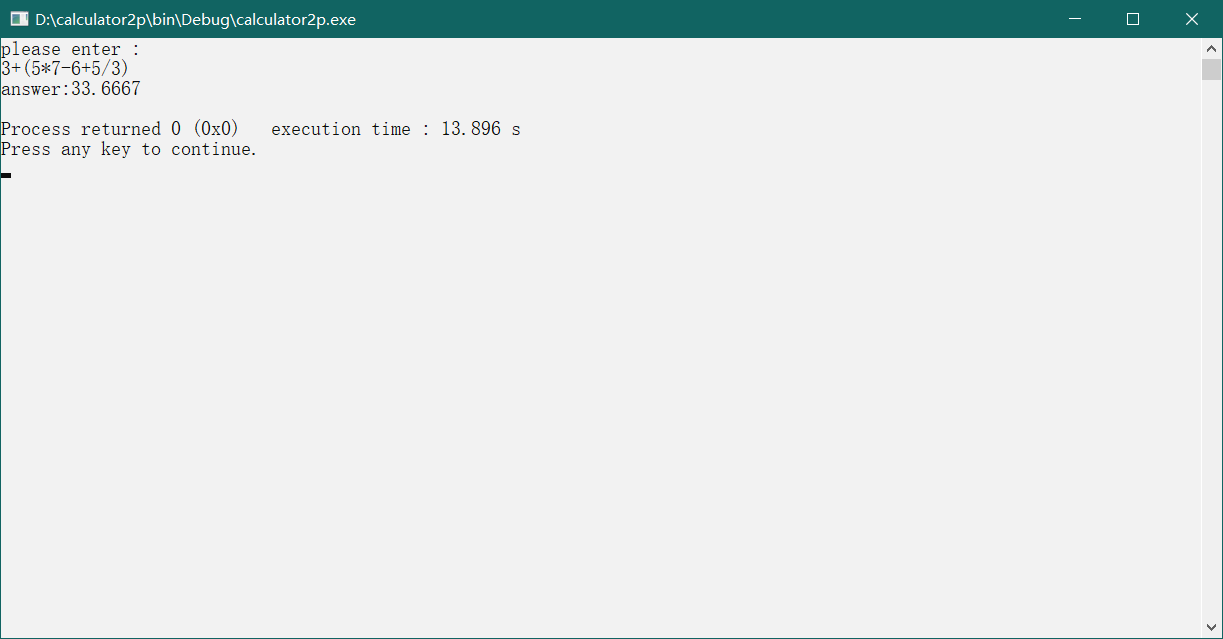


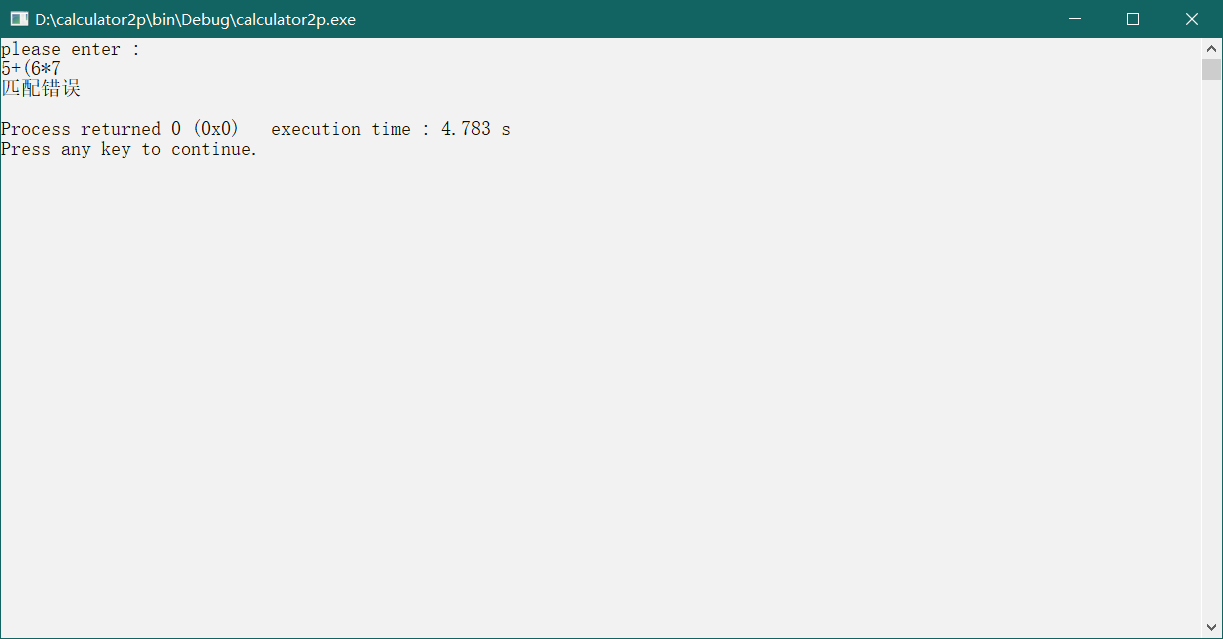
1. 给出程序的源代码，说明关键代码的操作含义，给出运行结果
2. #include <iostream>
3. #include <string>
4. #include <math.h>
6. **using** **namespace** std;
8. **template**<**typename** T>
9. **struct** stackNode
10. {
11. T data;//存储的具体元素
12. stackNode\* next;//指向下一个节点的指针
13. };
15. **template**<**typename** T>
16. **class** MyStack
17. {
18. **private**:
19. unsigned **int** count;//计数目前stack中存储的元素个数
20. unsigned **int** maxSize;//stack中最大存储的元素个数
21. stackNode<T>\* top;//stack头节点的指针
22. **public**:
23. MyStack();//默认构造函数，创建最多容纳10个元素的stack
24. MyStack(unsigned **int** ms);//自定义容纳个数
25. ~MyStack();//析构函数
26. **void** Push(T data);//压栈
27. T Pop(**void**);//出栈
28. T Top(**void**);//获取栈顶元素
29. **bool** isEmpty(**void**);//检查是否为空
30. **bool** notEmpty(**void**);
31. **bool** isfull(**void**);//经检查是否已满
32. };

35. **template**<**typename** T>
36. MyStack<T>::MyStack()
37. {
38. count = 0;
39. maxSize = 30;
40. top = NULL;
41. }
43. **template**<**typename** T>
44. MyStack<T>::MyStack(unsigned **int** ms)
45. {
46. count = 0;
47. maxSize = ms;
48. top = NULL;
49. }
51. **template**<**typename** T>
52. MyStack<T>::~MyStack()
53. {
54. stackNode<T>\* p = **new** stackNode<T>;
55. **for** (unsigned **int** i = count; i >0; i--)
56. {
57. p = top;
58. top = p->next;
59. **delete** p;
60. }
61. }
63. **template**<**typename** T>
64. **void** MyStack<T>::Push(T data)
65. {
66. count++;
67. **if** (isfull())
68. {
69. cout << "Error: This Stack is full" << endl;
70. cout << "File Path =" << \_\_FILE\_\_ << endl;
71. cout << "Function Name =" << \_\_FUNCTION\_\_ << endl;
72. cout << "Line =" << \_\_LINE\_\_ << endl;
73. }
74. stackNode<T>\* node = **new** stackNode<T>;
75. node->data = data;
76. node->next = top;
77. top = node;
78. }
80. **template**<**typename** T>
81. T MyStack<T>::Pop(**void**)
82. {
83. T nowdata;
84. **if** (isEmpty())
85. {
86. cout << "Error: Stack is empty" << endl;
87. cout << "File Path =" << \_\_FILE\_\_ << endl;
88. cout << "Function Name =" << \_\_FUNCTION\_\_ << endl;
89. cout << "Line =" << \_\_LINE\_\_ << endl;
90. **return** 1;
91. }
92. stackNode<T>\* p = top;
93. nowdata=top->data;
94. top = top->next;
95. **delete** p;
96. count--;
97. **return** nowdata;
98. }
100. **template**<**typename** T>
101. T MyStack<T>::Top(**void**)
102. {
103. **return** top->data;
104. }
106. **template**<**typename** T>
107. **bool** MyStack<T>::isEmpty(**void**)
108. {
109. **return** count == 0 ? **true** : **false**;
110. }
112. **template**<**typename** T>
113. **bool** MyStack<T>::notEmpty(**void**)
114. {
115. **return** count == 0 ? **false** : **true**;
116. }
118. **template**<**typename** T>
119. **bool** MyStack<T>::isfull(**void**)
120. {
121. **return** count > maxSize ? **true** : **false**;
122. }
124. **int** compare(**char** top,**char** now)
125. {
126. **int** toplevel=0;
127. **int** nowlevel=0;
128. **if**(top=='+'||top=='-')
129. {
130. toplevel=1;
131. }
132. **else** **if**(top=='\*'||top=='/')
133. {
134. toplevel=2;
135. }
136. **if**(now=='+'||now=='-')
137. {
138. nowlevel=1;
139. }
140. **else** **if**(now=='\*'||now=='/')
141. {
142. nowlevel=2;
143. }
144. **if**(nowlevel>toplevel)
145. {
146. **return** 2;
147. }
148. **else** **if**(nowlevel==toplevel)
149. {
150. **return** 1;
151. }
152. **else**
153. {
154. **return** 0;
155. }
156. }
158. **bool** notdod(**char** c)
159. {
160. **if**((**int**)c>=48&&(**int**)c<=57)
161. {
162. **return** **false**;
163. }
164. **else** **if**(c=='.')
165. {
166. **return** **false**;
167. }
168. **else**
169. {
170. **return** **true**;
171. }
172. }
174. **double** calculate(**double** num1,**char** op,**double** num2)
175. {
176. **if**(op=='+')
177. **return** num1+num2;
178. **else** **if**(op=='-')
179. **return** num1-num2;
180. **else** **if**(op=='\*')
181. **return** num1\*num2;
182. **else**
183. **return** num1/num2;
184. }
186. **int** main()
187. {
188. MyStack<**char**> C;//字符模板栈
189. MyStack<**double**> D;//数值模板栈
190. **int** flag=0;//标志变量
191. **int** con=0;//除以10的倍数
192. **int** asc=0;//当前字符的asc码
193. **double** num=0;//储存当前数字
194. string expression="null";
195. **char** s;
196. cout<<"please enter :"<<endl;
197. cin>>expression;
198. **for**(unsigned **int** i=0;i<**sizeof**(expression);i++)
199. {
200. s=expression[i];
201. asc=(**int**)s;
202. **if**(asc>=48&&asc<=57)//对数字进行处理
203. {
204. **if**(flag==0)
205. {
206. num=num\*10+asc-48;
207. }
208. **else**
209. {
210. num=num+(**float**)(asc-48)/pow(10,con);
211. con++;
212. }
213. **if**(notdod(expression[i+1]))
214. {
215. D.Push(num);
216. num=0;
217. con=0;
218. flag=0;
219. }
220. }
221. **else** **if**(asc==46)//遇到小数点进入除法模式
222. {
223. flag=1;
224. con++;
225. }
226. **else** **if**(s=='(')
227. {
228. C.Push(s);
229. }
230. **else** **if**(s==')')
231. {
232. **if**(C.Top()=='(')//遇到左括号弹出
233. C.Pop();
234. **else**
235. {
236. **do**{
237. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));//将数字弹出
238. **if**(C.isEmpty())//若空则弹出
239. {
240. cout<<"匹配错误"<<endl;
241. **break**;
242. }
243. }**while**(C.Top()!='(');
244. **if**(C.Top()=='(')//弹出括号
245. C.Pop();
246. }
247. }
248. **else** **if**(s=='+')
249. {
250. **if**(C.isEmpty())
251. C.Push(s);
252. **else** **if**(C.Top()=='(')
253. C.Push(s);
254. **else**
255. {
256. **do**{
257. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
258. **if**(C.isEmpty())
259. {
260. **break**;
261. }
262. }**while**(C.Top()!='(');
263. C.Push(s);
264. }
266. }
267. **else** **if**(s=='-')
268. {
269. **if**(C.isEmpty())
270. C.Push(s);
271. **else** **if**(C.Top()=='(')
272. C.Push(s);
273. **else**
274. {
275. **do**{
276. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
277. **if**(C.isEmpty())
278. {
279. **break**;
280. }
281. }**while**(C.Top()!='(');
282. C.Push(s);
283. }
284. }
285. **else** **if**(s=='\*')
286. {
287. **if**(C.isEmpty())
288. C.Push(s);
289. **else** **if**(C.Top()=='(')
290. C.Push(s);
291. **else** **if**(compare(C.Top(),s)==1)
292. {
293. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
294. C.Push(s);
295. }
296. **else** **if**(compare(C.Top(),s)==2)
297. C.Push(s);
298. }
299. **else** **if**(s=='/')
300. {
301. **if**(C.isEmpty())
302. C.Push(s);
303. **else** **if**(C.Top()=='(')
304. C.Push(s);
305. **else** **if**(compare(C.Top(),s)==1)
306. {
307. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
308. C.Push(s);
309. }
310. **else** **if**(compare(C.Top(),s)==2)
311. C.Push(s);
312. }
313. }
315. **if**(C.notEmpty())
316. {
317. **while**(C.Top()!='(')
318. {
319. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
320. **if**(C.isEmpty())
321. {
322. **break**;
323. }
324. }
325. **if**(C.Top()=='(')
326. {
327. cout<<"匹配错误"<<endl;
328. **return** 0;
329. }
331. }
332. num=D.Pop();
333. cout<<"answer:"<<num<<endl;
334. **return** 0;
335. }









**三 实验总结与建议**

（总结实验实施过程，说明实验过程中遇到的问题与解决方案；提出实验环节的建议）

一开始没有调整好弹出是程序段的代码，经常产生访问越界，也就是访问节点的头指针和判空是不能同时进行的，如果空了的话再去访问头指针就是访问一个不存在的地址，因此程序经常崩溃。

在调整好判空与判断头指针数据的顺序后程序终于能顺利进行。Debug发挥了重要的作用。