**C++ 语 言 程 序 设 计**

实

验

报

告

实 验 一

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. 实现字符栈功能
5. 实现表达式中数值与操作符的识别
6. 实现表达式中括号匹配的判断

**二 实验原理**

1. 说明字符串表达式中数值提取与操作符识别的方法（给出算法的**流程图**与简要说明）

操作符识别流程图：



即利用asc||码判断操作符是哪一个。

数值提取流程图：



即设置标志变量flag判断是乘还是除，用con代表除以10的几次方，用num储存未读取完的数值，遇到小数点转变模式，遇到操作符输出数值。

1. 说明表达式中括号匹配的判别方法（给出算法**流程图**与简要说明）

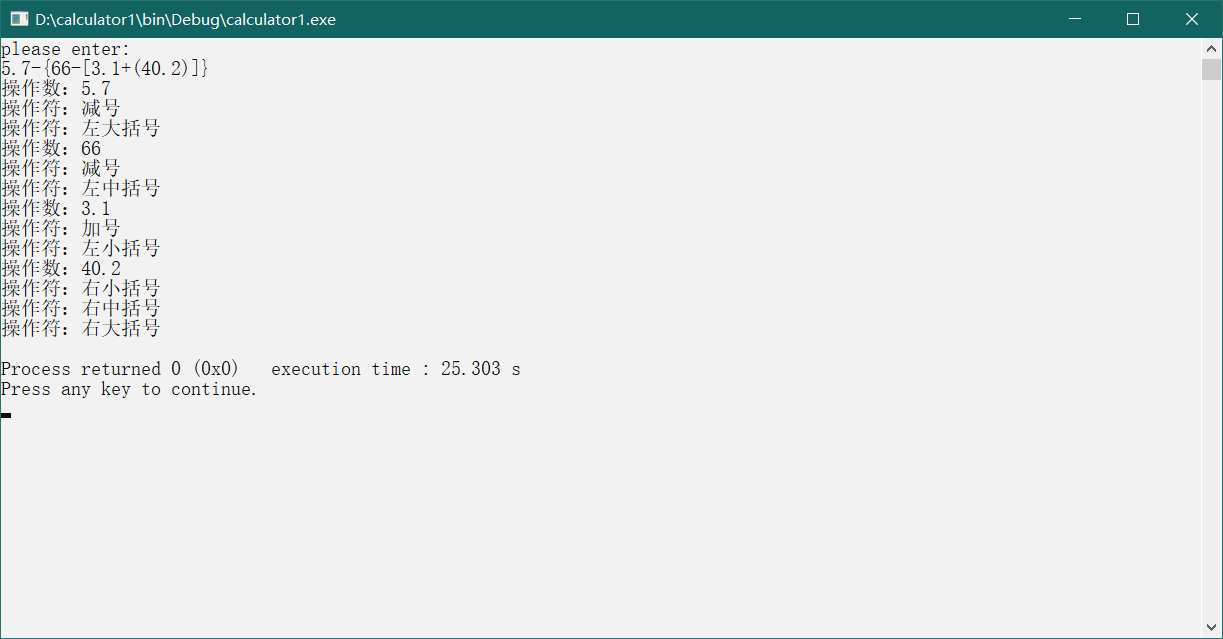


用节点类储存字符型数据，用栈类构造函数进行压栈，出栈，判断是否为空，遇到左括号压栈，右括号先判断是否为空，再弹出栈顶，判断是否匹配。

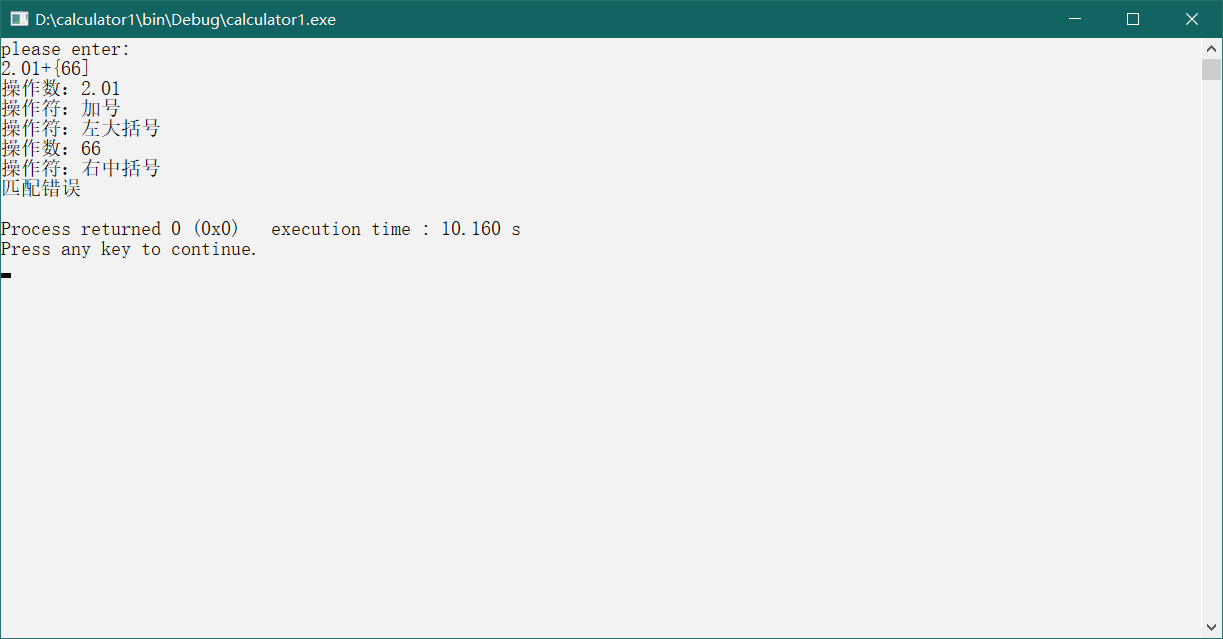
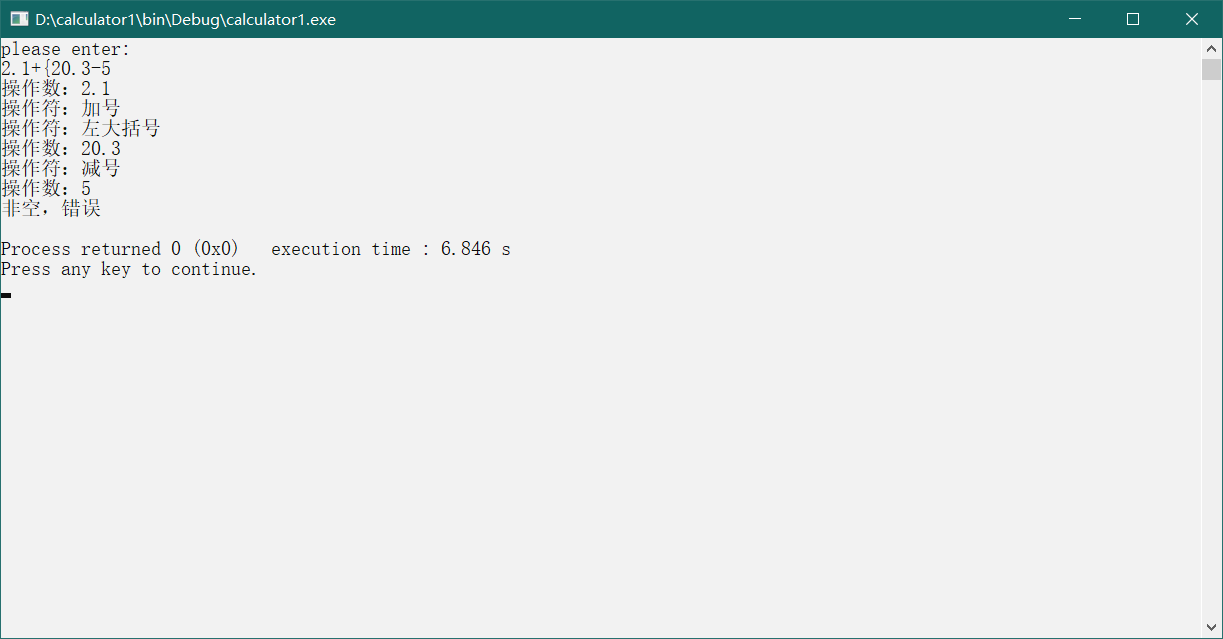
1. 给出程序的源代码，说明关键代码的操作含义，给出运行结果
2. #include <iostream>
3. #include <string>
4. #include <math.h>
6. **using** **namespace** std;
8. **class** Node
9. {
10. **char** date;//节点数据
11. Node \*pNext;//指向下一个节点的指针
12. **friend** **class** Stack;//友元类
13. };
15. **class** Stack
16. {
17. **public**:
18. Stack()//构造器
19. {
20. pTop = **new** Node;
21. pBottom = pTop;
22. pBottom->pNext = NULL;
24. }
25. ~Stack()//析构器
26. {
27. **delete** pTop;
28. }
30. **void** push(**char** val)//压栈
31. {
32. Node \*pNew = **new** Node;
33. pNew->date=val;
34. pNew->pNext=pTop;
35. pTop = pNew;
36. }
38. **void** traverse()//遍历
39. {
40. Node \*p=pTop;
41. **while**(p!=pBottom)
42. {
43. cout<<p->date<<endl;
44. p=p->pNext;
45. }
46. }
48. **bool** isempty()//判空
49. {
50. **if**(pTop==pBottom)
51. {
52. **return** **true**;
53. }
54. **else**
55. {
56. **return** **false**;
57. }
58. }
60. **char** top()//顶部出栈
61. {
62. **char** s;
63. **if**(isempty())
64. {
65. **return** **false**;
66. }
67. **else**
68. {
69. Node \*p=pTop;
70. s = p->date;
71. pTop = p->pNext;
72. **delete** p;
73. p=NULL;
75. **return** s;
76. }
77. }
79. **void** clearall()//清空栈数据
80. {
81. Node \*p=pTop;
82. Node \*q=NULL;
84. **while**(p!=pBottom)
85. {
86. q=p->pNext;
87. **delete** p;
88. p = q;
89. }
90. pTop=pBottom;
91. }
93. **private**:
94. Node \*pTop;//栈顶
95. Node \*pBottom;//栈底

98. };

101. **int** main()
102. {
103. Stack A;
104. string expression="null";
105. **char** s;
106. **char** s1;
107. **double** num=0;
108. **int** con=0;
109. **int** flag=0;//定义flag标志变量检查是小数点前还是小数点后
110. cout << "please enter:" << endl;
111. cin>>expression; //输入表达式
112. **for**(unsigned **int** i=0;i<**sizeof**(expression);i++)//遍历每一个字符
113. {
114. s=expression[i];//保存当前字符
115. **switch**((**int**)s)//检索asc||码
116. {
117. **case** 40:
118. //(
119. cout<<"操作符：左小括号"<<endl;
120. A.push(s);
121. num=0;//读取到操作符将之前读取到的数据清零
122. con=0;//读取到操作符将小数点后除的倍数清零
123. **break**;
124. **case** 41:
125. //)
126. cout<<"操作符：右小括号"<<endl;
127. **if**(A.isempty())
128. {
129. cout<<"匹配错误"<<endl;
130. **return** 0;
131. }
132. **else**
133. {
134. s1=A.top();
135. **if**(s1!='(')
136. {
137. cout<<"匹配错误"<<endl;
138. **return** 0;
139. }
140. num=0;
141. con=0;
142. }
143. flag=0;
144. **break**;
145. **case** 91:
146. //[
147. cout<<"操作符：左中括号"<<endl;
148. A.push(s);
149. num=0;
150. con=0;
151. flag=0;
152. **break**;
153. **case** 93:
154. //]
155. cout<<"操作符：右中括号"<<endl;
156. **if**(A.isempty())
157. {
158. cout<<"匹配错误"<<endl;
159. **return** 0;
160. }
161. **else**
162. {
163. s1=A.top();
164. **if**(s1!='[')
165. {
166. cout<<"匹配错误"<<endl;
167. **return** 0;
168. }
169. num=0;
170. con=0;
171. }
172. flag=0;
173. **break**;
174. **case** 123:
175. //{
176. cout<<"操作符：左大括号"<<endl;
177. A.push(s);
178. num=0;
179. con=0;
180. flag=0;
181. **break**;
182. **case** 125:
183. //}
184. cout<<"操作符：右大括号"<<endl;
185. **if**(A.isempty())
186. {
187. cout<<"匹配错误"<<endl;
188. **return** 0;
189. }
190. **else**
191. {
192. s1=A.top();
193. **if**(s1!='{')
194. {
195. cout<<"匹配错误"<<endl;
196. **return** 0;
197. }
198. num=0;
199. con=0;
200. }
201. flag=0;
202. **break**;
203. **case** 42:
204. //\*
205. cout<<"操作符：乘号"<<endl;
206. num=0;
207. con=0;
208. flag=0;
209. **break**;
210. **case** 43:
211. //+
212. cout<<"操作符：加号"<<endl;
213. num=0;
214. con=0;
215. flag=0;
216. **break**;
217. **case** 45:
218. //-
219. cout<<"操作符：减号"<<endl;
220. num=0;
221. con=0;
222. flag=0;
223. **break**;
224. **case** 47:
225. // /
226. cout<<"操作符：除号"<<endl;
227. num=0;
228. con=0;
229. flag=0;
230. **break**;
231. **case** 48:
232. //0
233. **if**(flag==0)
234. {
235. num=num\*10+0;
236. }
237. **else**
238. {
239. num=num+(**float**)0/pow(10,con);
240. con++;//下一次多除以10
241. }
242. //如果下一个是操作符直接输出数值
243. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
244. {
245. cout<<"操作数："<<num<<endl;
246. }
247. **break**;
248. **case** 49:
249. //1
250. **if**(flag==0)
251. {
252. num=num\*10+1;
253. }
254. **else**
255. {
256. num=num+(**float**)1/pow(10,con);
257. con++;
258. }
259. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
260. {
261. cout<<"操作数："<<num<<endl;
262. }
263. **break**;
264. **case** 50:
265. //2
266. **if**(flag==0)
267. {
268. num=num\*10+2;
269. }
270. **else**
271. {
272. num=num+(**float**)2/pow(10,con);
273. con++;
274. }
275. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
276. {
277. cout<<"操作数："<<num<<endl;
278. }
279. **break**;
280. **case** 51:
281. //3
282. **if**(flag==0)
283. {
284. num=num\*10+3;
285. }
286. **else**
287. {
288. num=num+(**float**)3/pow(10,con);
289. con++;
290. }
291. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
292. {
293. cout<<"操作数："<<num<<endl;
294. }
295. **break**;
296. **case** 52:
297. //4
298. **if**(flag==0)
299. {
300. num=num\*10+4;
301. }
302. **else**
303. {
304. num=num+(**float**)4/pow(10,con);
305. con++;
306. }
307. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
308. {
309. cout<<"操作数："<<num<<endl;
310. }
311. **break**;
312. **case** 53:
313. //5
314. **if**(flag==0)
315. {
316. num=num\*10+5;
317. }
318. **else**
319. {
320. num=num+(**float**)5/pow(10,con);
321. con++;
322. }
323. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
324. {
325. cout<<"操作数："<<num<<endl;
326. }
327. **break**;
328. **case** 54:
329. //6
330. **if**(flag==0)
331. {
332. num=num\*10+6;
333. }
334. **else**
335. {
336. num=num+(**float**)6/pow(10,con);
337. con++;
338. }
339. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
340. {
341. cout<<"操作数："<<num<<endl;
342. }
343. **break**;
344. **case** 55:
345. //7
346. **if**(flag==0)
347. {
348. num=num\*10+7;
349. }
350. **else**
351. {
352. num=num+(**float**)7/pow(10,con);
353. con++;
354. }
355. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
356. {
357. cout<<"操作数："<<num<<endl;
358. }
359. **break**;
360. **case** 56:
361. //8
362. **if**(flag==0)
363. {
364. num=num\*10+8;
365. }
366. **else**
367. {
368. num=num+(**float**)8/pow(10,con);
369. con++;
370. }
371. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
372. {
373. cout<<"操作数："<<num<<endl;
374. }
375. **break**;
376. **case** 57:
377. //9
378. **if**(flag==0)
379. {
380. num=num\*10+9;
381. }
382. **else**
383. {
384. num=num+(**float**)9/pow(10,con);
385. con++;
386. }
387. **if**(((**int**)expression[i+1]<46)||((**int**)expression[i+1]>57))
388. {
389. cout<<"操作数："<<num<<endl;
390. }
391. **break**;
392. **case** 46:
393. //.
394. flag=1;
395. con++;
396. **break**;
397. }
398. }
400. **if**(A.isempty()==0)//判断结束后是否为空
401. {
402. cout<<"非空，错误"<<endl;
403. }
404. **return** 0;
405. }







**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）

遇到的问题，不能使用#include<stack>,只能用Node类和Stack类编写动态栈，实现压栈，弹出栈顶等操作。

一开始不知道如何完整的输出操作数，后来加入了判断下一个字符，如果下一个字符是除了小数点的操作符，则直接输出当前存储的操作数，并且将操作数置为0。