**江苏科技大学**

**实 验 报 告**

课 程： 编译原理

课 题： 实验3·逆波兰式的产生及计算

学 院： 计算机学院

学 号： 182210710119

姓 名： 陈四贵

班 级： 1822107101

指导老师： 肖 寒

# 一、实验目的

通过完成非后缀式表示的算术表达式转换为用逆波兰式表示的算术表达式，并计算用逆波兰式来表示的算术表达式的值，是学生了解逆波兰式的构成方式及其计算方法。

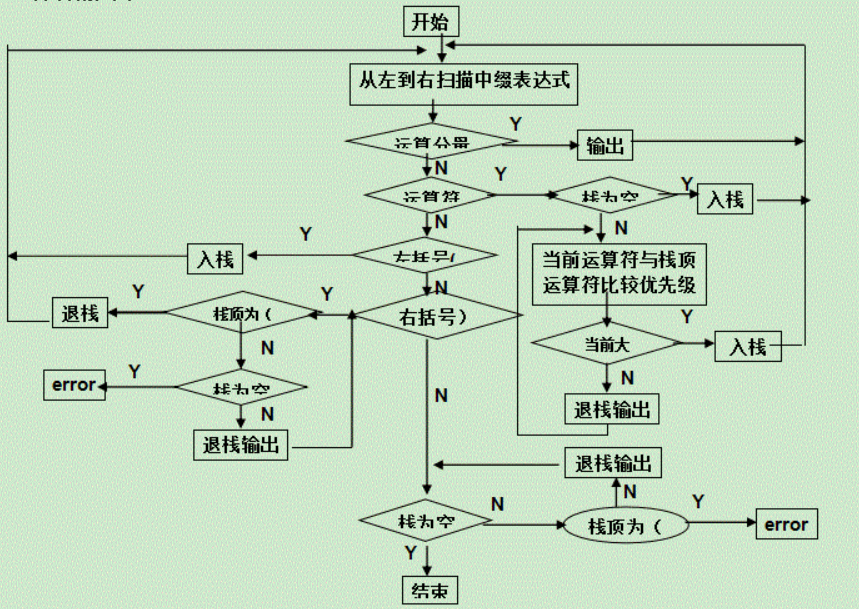
# 二、设备与环境

windows 10环境、visual studio 2017

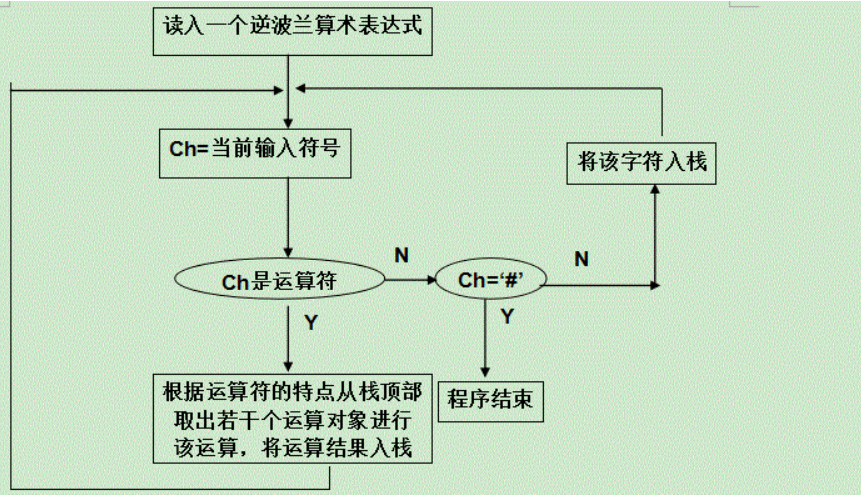
# 三、实验要求

将用非后缀式表示的算术表达式转换为用逆波兰式来表示的算术表达式，并计算用逆波兰式表示的算术表达式的值。

# 四、实验原理

1、算法分析·生成逆波兰式流程图

2、算法分析·逆波兰式的计算流程图

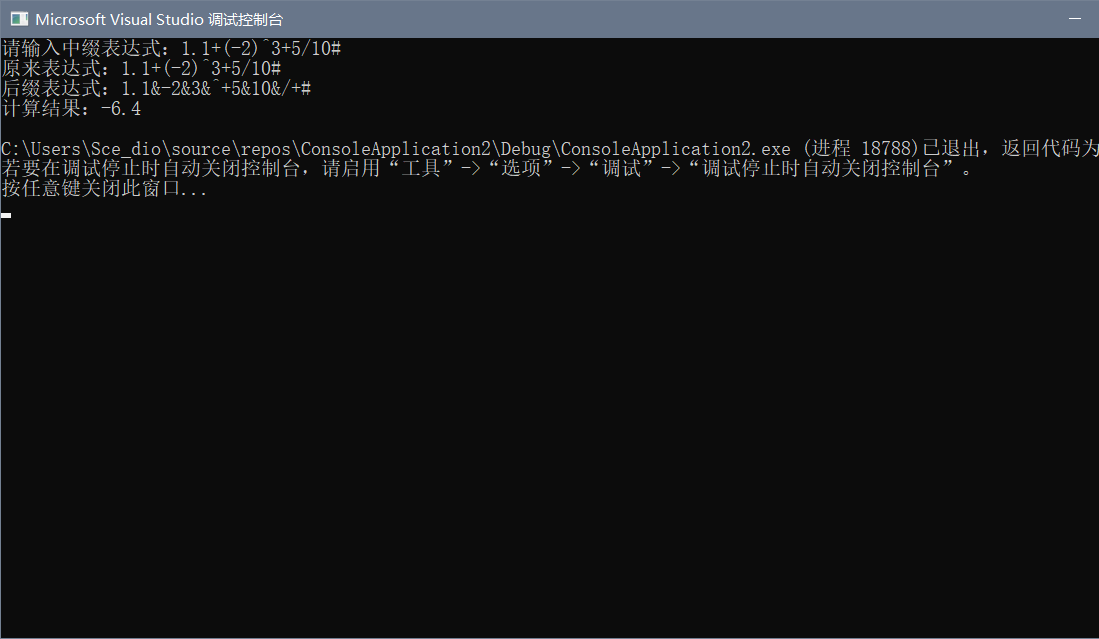


# 五、程序测试与调试

1、测试用例

1.1+(-2)^3+5/10#

2、运行结果参考



# 六、程序源码

1. #include<iostream>
2. #include<stdio.h>
3. #include<stdlib.h>
4. #include<algorithm>
5. #include<cctype>
6. #include<cstring>
7. **using** **namespace** std;
9. **char** str[50];   //用于存放原来的表达式
10. **int** top;    //栈顶指针
11. **char** stack[50];     //定义栈，用于计算逆波兰式
12. **char** ex[50];    //存放后缀表达式
13. **double** \_stack[50]; //定义栈，用于计算逆波兰式子
14. **int** flag[50];   //用于区分+、-号的含义，0表示运算符，1表示正负号
16. //生成逆波兰式
17. **void** NiBolan()
18. {
19. memset(flag, 0, **sizeof**(flag));    //flag初始值设为0
20. **char** ch = str[0];
21. **int** i = 1, t = 0;
22. top = 0;
24. **while** (ch != '#')
25. {
26. **switch** (ch)
27. {
28. **case** '(':
29. top++;
30. stack[top] = ch;
31. **break**;
32. **case** ')':
33. **while** (stack[top] != '(')
34. {
35. ex[t] = stack[top];
36. top--;
37. t++;
38. }
39. top--;
40. **break**;
41. **case** '^':
42. **while** (stack[top] == '^')  //设置^运算符优先级为最高
43. {
44. ex[t] = stack[top];
45. top--;
46. t++;
47. }
48. top++;
49. stack[top] = ch;
50. **break**;
51. **case** '+':
52. **case** '-':
53. //当ch为+、-号是，若前面相邻字符不是')'或数字且后面相邻字符是数字时表示正负号
54. **if** (isdigit(str[i]) && !isdigit(str[i - 2]) && str[i - 2] != ')')
55. {
56. flag[t] = 1;  //标记符号为正负号
57. ex[t++] = ch;
58. ch = str[i++];
59. **while** ((ch >= '0'&&ch <= '9') || ch == '.')  //判别小数点
60. {
61. ex[t] = ch;
62. t++;
63. ch = str[i];
64. i++;
65. }
66. i--;
67. ex[t] = '&';
68. t++;
69. }
70. **else**
71. {
72. **while** (top != 0 && stack[top] != '(')
73. {
74. ex[t] = stack[top];
75. top--;
76. t++;
77. }
78. top++;
79. stack[top] = ch;
80. }
81. **break**;
82. **case** '\*':
83. **case** '/':
84. **while** (stack[top] == '\*' || stack[top] == '/' || stack[top] == '^')    //运算符^优先级高于\*和/
85. {
86. ex[t] = stack[top];
87. top--;
88. t++;
89. }
90. top++;
91. stack[top] = ch;
92. **break**;
93. **case** ' ':
94. **break**;
95. **default**:
96. **while** ((ch >= '0'&&ch <= '9') || ch == '.')  //判别小数点
97. {
98. ex[t] = ch;
99. t++;
100. ch = str[i];
101. i++;
102. }
103. i--;
104. ex[t] = '&';
105. t++;
106. }
107. ch = str[i];
108. i++;
109. }
110. **while** (top != 0)
111. **if** (stack[top] != '(')
112. {
113. ex[t] = stack[top];
114. t++;
115. top--;
116. }
117. **else**
118. {
119. printf("error");
120. top--;
121. exit(0);
122. }
123. ex[t] = '#';
124. ex[t + 1] = '\0';
125. printf("后缀表达式：%s\n", ex);
127. }
129. //计算逆波兰式
130. **void** Calculate()
131. {
132. **char** ch = ex[0];
133. **int** t = 0;
134. top = -1;
136. **while** (ch != '#')
137. {
138. **if** (ch == '&') {
139. ch = ex[++t];
140. **continue**;
141. }
142. **switch** (ch)
143. {
144. **case** '+':
145. **if** (flag[t]) //'+'表示正号
146. {
147. ch = ex[++t];
148. **double** d = 0;
149. **while** (ch >= '0'&&ch <= '9')
150. {
151. d = 10.0\*d + **double**(ch - '0');
152. ch = ex[++t];
153. }
154. **if** (ch == '.')     //判断是否为小数
155. {
156. ch = ex[++t];
157. **double** k = 1.0;
158. **while** (ch >= '0'&&ch <= '9')
159. {
160. d = d + **double**(ch - '0') / (10.0\*k);
161. k = k + 1.0;
162. ch = ex[++t];
163. }
164. }
165. top++;
166. \_stack[top] = d;
167. }
168. **else**
169. {
170. \_stack[top - 1] = \_stack[top - 1] + \_stack[top];
171. top--;
172. t++;
173. }
174. **break**;
175. **case** '-':
176. **if** (flag[t]) //'-'表示负号
177. {
178. ch = ex[++t];
179. **double** d = 0;
180. **while** (ch >= '0'&&ch <= '9')
181. {
182. d = 10.0\*d + **double**(ch - '0');
183. ch = ex[++t];
184. }
185. **if** (ch == '.')
186. {
187. ch = ex[++t];
188. **double** k = 1.0;
189. **while** (ch >= '0'&&ch <= '9')
190. {
191. d = d + **double**(ch - '0') / (10.0\*k);
192. k = k + 1.0;
193. ch = ex[++t];
194. }
195. }
196. top++;
197. \_stack[top] = -d;
198. }
199. **else**
200. {
201. \_stack[top - 1] = \_stack[top - 1] - \_stack[top];
202. top--;
203. t++;
204. }
205. **break**;
206. **case** '^':   //运算符为'^'
207. **if** (\_stack[top] == 0)
208. {
209. \_stack[top - 1] = 1;
210. }
211. **else**
212. {
213. **int** temp;
214. temp = \_stack[top - 1];
215. **while** (--\_stack[top])
216. {
217. \_stack[top - 1] \*= temp;
218. }
219. }
220. top--;
221. t++;
222. **break**;
223. **case** '\*':
224. \_stack[top - 1] = \_stack[top - 1] \* \_stack[top];
225. top--;
226. t++;
227. **break**;
228. **case** '/':
229. **if** (\_stack[top] != 0)
230. \_stack[top - 1] = \_stack[top - 1] / \_stack[top];
231. **else**
232. {
233. printf("\n\tchu0error!\n");
234. exit(0);
235. }
236. top--;
237. t++;
238. **break**;
239. **default**:
240. **double** d = 0;
241. **while** (ch >= '0'&&ch <= '9')
242. {
243. d = 10.0\*d + **double**(ch - '0');
244. ch = ex[++t];
245. }
246. **if** (ch == '.') //判断是否为小数
247. {
248. ch = ex[++t];
249. **double** k = 1.0;
250. **while** (ch >= '0'&&ch <= '9')
251. {
252. d = d + **double**(ch - '0') / (10.0\*k);
253. k = k + 1.0;
254. ch = ex[++t];
255. }
256. }
257. top++;
258. \_stack[top] = d;
259. }
260. ch = ex[t];
261. }
262. cout << "计算结果：" << \_stack[top] << endl;
263. //printf("计算结果：%lf\n",\_stack[top]);
264. }
266. **int** main()
267. {
268. printf("请输入中缀表达式：");
269. cin>>str;   //输入原表达式
270. printf("原来表达式：%s\n", str);
271. NiBolan();  //生成逆波兰式
272. Calculate();    //计算逆波兰式
273. **return** 0;
274. }