****

**实 验 报 告**

**实验课程： 数据结构与算法实验**

**实验题目： 栈与队列验证算术表达式的正确性**

**任课老师： 吴之旭老师**

**学生姓名： 殷骢睿**

**学 号： 5509121041 \_**

**专业班级： 2021级人工智能实验班 \_**

**2022 年 11 月 1日**

** 南昌大学实验报告**

学生姓名： 殷骢睿 学号： 5509121041 专业班级： 人工智能211班

实验类型：□ 验证 □ 综合 ■ 设计 □ 创新 实验日期： 2022.11.1 实验成绩：

1. **实验目的**

(1) 掌握栈与队列的定义；

(2) 掌握栈与队列的基本操作；

1. **实验内容**

1. 验证某算术表达式的正确性，若正确，则计算该算术表达式的值。

(1) 从键盘上输入表达式的值；

(2) 分析该表达式是否合法：

（a）是数字，则判断该数字的合法性。若合法，则压入数据存放堆栈

（b）是规定的运算符，则根据规则进行处理。在处理过程中，将计算该表达式的值。

（c）若是其它字符，则返回错误信息。

(3) 若上述处理过程中没有发现错误，则认为该表达式合法，并打印处理结果；

2. 表达式的计算过程：

1、变量直接输出；

2、遇到“（”，无条件入栈；

3、输入运算优先数是2，3，4，5，时，如栈空则入栈，否则，将其与栈顶元素进行比较，如果优先级大，就进栈，否则就退栈输出；

4、如遇“）”，同时栈顶元为“（”，则退栈，抹去括号。

**三、实验要求**

(1) 程序要添加适当的注释，程序的书写要采用缩进格式。

(2) 程序要具在一定的**健壮性**，即当输入数据非法时，程序也能适当地做出反应。

(3) 程序要做到界面友好，在程序运行时用户可以根据相应的提示信息进行操作，比如如何结束一个多项式的输入。

(4) 根据实验报告模板详细书写实验报告,在实验报告中给出相应的流程图。

(5) 源程序和实验报告打包。实验报告命名为：实验报告3.doc。源程序和实验报告压缩为一个文件（如果定义了头文件则一起压缩），按以下方式命名：学号姓名.rar，如0814101王五.rar。

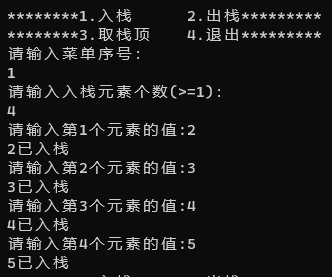
**四、主要实验步骤及程序分析**

**4.1 功能实现操作界面搭建**

如果是数字，首先判断数字的合法性，如果数字是合法的，可以点击“1”进行入栈操作，如果是能够匹配的（如“（）”），则退栈并且进行判断，同时抹去括号，该UI的优点是人机交互友好，可以提示“栈已满”、“栈已空”等信息，以便于用户进行表达式的输入。

（1）功能实现函数

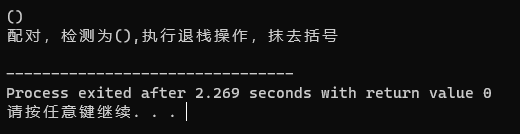
1. //\*功能实现函数
2. **void** menu()
3. {
4. printf("\*\*\*\*\*\*\*\*1.入栈      2.出栈\*\*\*\*\*\*\*\*\*\n");
5. printf("\*\*\*\*\*\*\*\*3.取栈顶    4.退出\*\*\*\*\*\*\*\*\*\n");
6. }
7. //入栈功能函数 调用Push函数
8. **void** PushToStack(SqStack &s)
9. {
10. **int** n;SElemType e;**int** flag;
11. printf("请输入入栈元素个数(>=1)：\n");
12. scanf("%d",&n);
13. **for**(**int** i=0;i<n;i++)
14. {
15. printf("请输入第%d个元素的值:",i+1);
16. scanf("%d",&e);
17. flag=Push(s,e);
18. **if**(flag)printf("%d已入栈\n",e);
19. **else** {printf("栈已满！！！\n");**break**;}
20. }
21. }
22. //出栈功能函数 调用Pop函数
23. **void** PopFromStack(SqStack &s)
24. {
25. **int** n;SElemType e;**int** flag;
26. printf("请输入出栈元素个数(>=1)：\n");
27. scanf("%d",&n);
28. **for**(**int** i=0;i<n;i++)
29. {
30. flag=Pop(s,e);
31. **if**(flag)printf("%d已出栈\n",e);
32. **else** {printf("栈已空！！！\n");**break**;}
33. }
34. }
35. //取栈顶功能函数 调用GetTop
36. **void** GetTopOfStack(SqStack &s)
37. {
38. SElemType e;**bool** flag;
39. flag=GetTop(s,e);
40. **if**(flag)printf("栈顶元素为:%d\n",e);
41. **else** printf("栈已空！！！\n");
42. }



**图1 功能实现操作界面搭建示意图**

（2）实现匹配

1. **ool** Match(**char** exp[],**int** n)//匹配
2. {
3. **int** i=0;
4. **char** e;
5. **bool** match=**true**;
6. LiStack \*st;
7. initStack(st);
8. **while**(i<n&&match)//扫描所有字符
9. {//小括号
10. **if**(exp[i]=='(')
11. Push(st,exp[i]);//进栈
12. **else** **if**(exp[i]==')')
13. {
14. **if**(GetTop(st,e)==**true**)
15. {
16. **if**(e!='(')//栈顶不是（时不匹配
17. match=**false**;
18. **else**
19. Pop(st,e);//匹配成功出栈
20. }
21. **else** match=**false**;//取不到栈顶元素时不匹配



**图2 匹配示意图**

**4.2主要具体步骤（完整代码见附录）**

(1) 所用数据结构的定义及其相关说明（相关结构体或类的定义及其含义）

1. //定义一个栈类型的结构体
2. **typedef** **struct** SqStack
3. {
4. SElemType \* base;        /\* 在栈构造之前和销毁之后，base的值为NULL \*/
5. SElemType \* top;         /\* 栈顶指针\*/
6. **int** stacksize;           /\* 当前已分配的存储空间，以元素为单位\*/
7. }SqStack;

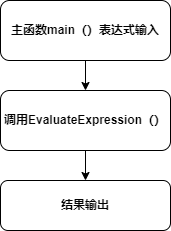
(2) 自定义函数的名称及其功能说明

1. Status InitStack(SqStack &S)   /\* 构造一个空栈S \*/
2. Status GetTop(SqStack &S, SElemType &e)  /\* 若栈不空，则用e返回S的栈顶元素，并返回e；否则返回ERROR \*/
3. Status Push(SqStack &S, SElemType e)   /\* 插入元素e为新的栈顶元素\*/
4. Status Pop(SqStack &S, **char** &e) /\* 若栈不空，则删除S的栈顶元素，用e返回其值，并返回e；否则返回ERROR \*/
5. **char** Precede(**char** t1,  **char** t2) /\* 判断两符号的优先关系\*/
6. **int** In(**char** c) /\* 判断c是否为运算符\*/
7. **char** Operate(**char** a,  **char** theta,  **char** b) //进行运算的函数
8. **char** EvaluateExpression() //输入运算表达式并计算

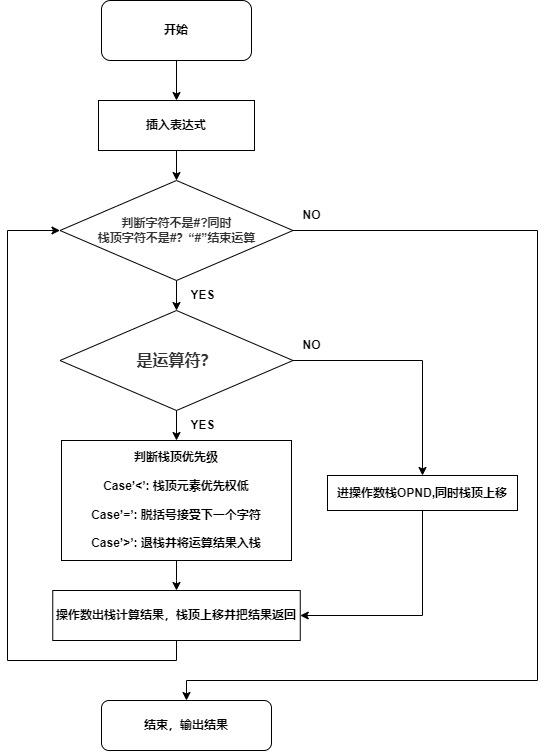
(3) 主要功能算法的时间复杂度

1. Status InitStack(SqStack &S)                          O(1);
2. Status GetTop(SqStack &S, SElemType &e)            O(1);
3. Status Push(SqStack &S, SElemType e)                 O(1);
4. Status Pop(SqStack &S, **char** &e)                      O(1);
5. **char** Precede(**char** t1,  **char** t2)                        O(1);
6. **int** In(**char** c)                                         O(1);
7. **char** Operate(**char** a,  **char** theta,  **char** b)             O(1);
8. **char** EvaluateExpression()                             O(n);

**4.3 流程图**

****

**图3 总体思路规划图**

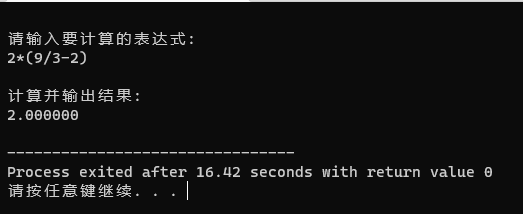
****

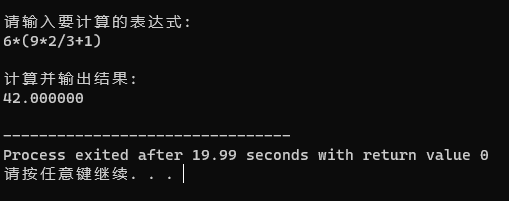
**图4 利用栈验证算术表达式正确性完整逻辑流程图**

**4.4 程序结果**

输入：需要计算的表达式

输出：相加多项式，各多项式的项式和最高次幂。





**图5 利用栈验证算术表达式正确性程序结果图**

**五、编写代码时遇到的问题及实验心得体会**

**遇到的问题：**

(1) 所面对的问题直接想到运用栈和队列仍有难度，需要通过不断练习来进一步完善。

(2) 思考代码逻辑速度比较慢，需要继续花时间去提高。

**实验心得体会：**

(1) 通过编写算术表达式的程序实现了利用栈进行算术表达式求解。利用栈和队列分别存储运算符和字符然后比较优先级实现运算输出结果。老师的题目很难，也让我知道了自己理论能力和编程能力的不足，但是经过长时间的独立思考，觉得还是非常有成就感的。

(2) 这次编程坚决没有上网搜索现成的方案，锻炼了自己的意志力和克服困难的能力，很感谢老师的这次考验。

(3) 算术表达式求解运用了栈和队列相应知识，我从中深刻体会到了栈和队列的深刻意义，能够灵活运用栈和队列来运用到其他领域。

**六、实验源码**

**附件一：Stack.cpp**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
5. #define MaxSize 9999
7. //用于存放运算符的栈的数据类型定义及基本操作：
9. //数据类型定义
10. #define ElemType1 char
12. **typedef** **struct** {
13. ElemType1 data[MaxSize];
14. **int** top;
15. } SqStack1;
17. //运算符栈的初始化操作：
18. **void** InitStack\_1(SqStack1 &S) {
19. memset(S.data, '\0', MaxSize);
20. S.top = -1;
21. }
23. //取运算符栈的栈顶元素:
24. **bool** GetTop\_1(SqStack1 &S, ElemType1 &x) {
25. **if** (S.top == -1)
26. **return** **false**;
27. x = S.data[S.top];
28. **return** **true**;
29. }
31. //运算符栈入栈操作:
32. **bool** Push\_1(SqStack1 &S, ElemType1 x) {
33. **if** (S.top == (MaxSize - 1))
34. **return** **false**;
35. S.data[++S.top] = x;
36. **return** **true**;
37. }
39. //运算符栈出栈操作:
40. **bool** Pop\_1(SqStack1 &S, ElemType1 &x) {
41. **if** (S.top == -1)
42. **return** **false**;
43. x = S.data[S.top--];
44. **return** **true**;
45. }
47. //用于存放操作数的栈的数据类型定义及基本类型操作
49. //数据类型定义
50. #define ElemType2 float
52. **typedef** **struct** {
53. ElemType2 data[MaxSize];
54. **int** top;
55. } SqStack2;
57. //操作数栈的初始化操作：
58. **void** InitStack\_2(SqStack2 &S) {
59. S.top = -1;
60. }
62. //取操作数栈的栈顶元素:
63. **bool** GetTop\_2(SqStack2 &S, ElemType2 &x) {
64. **if** (S.top == -1)
65. **return** **false**;
66. x = S.data[S.top];
67. **return** **true**;
68. }
70. //操作数栈入栈操作:
71. **bool** Push\_2(SqStack2 &S, ElemType2 x) {
72. **if** (S.top == (MaxSize - 1))
73. **return** **false**;
74. S.data[++S.top] = x;
75. **return** **true**;
76. }
78. //操作数栈出栈操作:
79. **bool** Pop\_2(SqStack2 &S, ElemType2 &x) {
80. **if** (S.top == -1)
81. **return** **false**;
82. x = S.data[S.top--];
83. **return** **true**;
84. }
86. //用于判断运算符优先级的高低，优先级高于或等于就返回true，否则为false
87. **bool** Priority(**char** a, **char** b) {
88. **if** ((a == '+' || a == '-') && (b == '+' || b == '-' || b == '\*' || b == '/')) {
89. **return** **true**;
90. } **else** **if** ((a == '\*' || a == '/') && (b == '\*' || b == '/')) {
91. **return** **true**;
92. }
93. **return** **false**;
94. }
96. //用于两个操作数之间的计算
97. ElemType2 CalculateTwo(ElemType2 n1, ElemType2 n2, ElemType1 t) {
98. //n1是右操作数,n2是左操作数
99. **if** (t == '+')
100. **return** n1 + n2;
101. **if** (t == '-')
102. **return** n2 - n1;
103. **if** (t == '/')
104. **return** n2 / n1;
105. **if** (t == '\*')
106. **return** n2 \* n1;
107. }
109. //计算中缀表达式的算法:
110. **void** Calculate(ElemType1 str[], ElemType2 &e) {
111. SqStack1 S1;//S1是运算符栈
112. SqStack2 S2;//S2是操作数栈
113. InitStack\_1(S1);//初始化S1
114. InitStack\_2(S2);//初始化S2
115. **float** n = 0;
116. ElemType2 n1, n2;
117. ElemType1 t;
118. **for** (unsigned **int** i = 0; i < strlen(str); i++) {
119. **if** (str[i] >= '0' && str[i] <= '9') {
120. **while** (1) {
121. n \*= 10;
122. n += (str[i] - '0');
123. **if** (str[i + 1] >= '0' && str[i + 1] <= '9')
124. i++;
125. **else**
126. **break**;
127. }
128. Push\_2(S2, n);
129. n = 0;
130. } **else** **if** (str[i] != ')') {
131. **while** (GetTop\_1(S1, t) && t != '(' && Priority(str[i], t)) {
132. Pop\_1(S1, t);
133. Pop\_2(S2, n1); //n1是右操作数
134. Pop\_2(S2, n2); //n2是左操作数
135. Push\_2(S2, CalculateTwo(n1, n2, t));
136. }
137. Push\_1(S1, str[i]);
138. } **else** {
139. **while** (Pop\_1(S1, t)) {
140. **if** (t == '(')
141. **break**;
142. Pop\_2(S2, n1); //n1是右操作数
143. Pop\_2(S2, n2); //n2是左操作数
144. Push\_2(S2, CalculateTwo(n1, n2, t));
145. }
146. }
147. }
148. **while** (Pop\_1(S1, t) && Pop\_2(S2, n1) && Pop\_2(S2, n2)) {
149. Push\_2(S2, CalculateTwo(n1, n2, t));
150. }
151. Pop\_2(S2, e);
152. }
154. **int** main() {
155. ElemType1 str[MaxSize];
156. ElemType2 e;
157. memset(str, '\0', MaxSize);
159. printf("\n请输入要计算的表达式:\n");
160. scanf("%s", str);
161. printf("\n计算并输出结果:\n");
162. Calculate(str, e);
163. printf("%f\n", e);
164. **return** 0;
165. }