题目：一元稀疏多项式计算器

班级：19184115

姓名：

学号：19063140

日期：2020.10.26

### 需求分析：

表达式计算是实现程序设计语言的基本问题之一,也是栈的应用的一个典型例子.设计一个程序,演示用算符优先法对算术表达式求值的过程。

以字符序列的形式从终端输入语法正确的、不含变量的整数表达式。利用教科书表3.1给出的算符优先关系,实现对算术四则混合运算表达式的求值,并仿照教科书的例3-1演示在求值中运算符栈、运算数栈、输入字符和主要操作的变化过程。

### 概要设计：

char priority[][] 存储运算符之间的优先关系

char Procede(char a, char b) 通过映射判断运算符a与运算符b的优先关系

int Operate(int m, int n, char x) 将m,n通过运算符x进行运算

int ExpressionEvaluation(char \*s) 表达式求值

### 调试分析：

1. 遇到的问题
   1. 问题：除数为零

解决方法：特判

1. 时空分析

设表达式长度为n

时间复杂度 O(n)

空间复杂度 O(n)

### 用户使用说明：

输入一行,需要求解的表达式

### 测试结果：

输入

8

1+2+3+4

88-1\*5

1024/4\*8

1024/(4\*8)

(20+2)\*(6/2)

3-3-3

2\*(6+2\*(3+6\*(6+6)))

(((6+6)\*6+3)\*2+6)\*2

8/(9-9)

输出

8

10

83

2048

32

66

-3

312

312

ERROR:divisor is zero

### 附录：

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
5. /\* 数据范围 \*/
6. #define N 100
8. **char** priority[7][7]={
9. {'>','>','<','<','<','>','>'},
10. {'>','>','<','<','<','>','>'},
11. {'>','>','>','>','<','>','>'},
12. {'>','>','>','>','<','>','>'},
13. {'<','<','<','<','<','=','0'}, // 此行"("=")"表示左右括号相遇，括号内运算已完成
14. {'>','>','>','>','0','>','>'}, // "=" 表示整个表达式求值完毕
15. {'<','<','<','<','<','0','='}  // "0"表示不可能出现这种情况 ( 语法错误 )
16. };
18. //Precede 用于判断运算符栈栈顶运算符 a1 与读入运算符 a2 之间的优先关系函数
19. **char** Procede(**char** a, **char** b){   // 建立 pre[][] 到 运算符间的映射关系
20. **int** i, j;
21. **switch**(a) {
22. **case** '+': i = 0; **break**;
23. **case** '-': i = 1; **break**;
24. **case** '\*': i = 2; **break**;
25. **case** '/': i = 3; **break**;
26. **case** '(': i = 4; **break**;
27. **case** ')': i = 5; **break**;
28. **case** '#': i = 6; **break**; // # 是表达式的结束符
29. }
30. **switch**(b) {
31. **case** '+': j = 0; **break**;
32. **case** '-': j = 1; **break**;
33. **case** '\*': j = 2; **break**;
34. **case** '/': j = 3; **break**;
35. **case** '(': j = 4; **break**;
36. **case** ')': j = 5; **break**;
37. **case** '#': j = 6; **break**;
38. }
39. **return** priority[i][j];
40. }
42. **int** Operate(**int** m, **int** n, **char** x){
43. **if** (x == '+') {
44. **return** m+n;
45. } **else** **if** (x == '-') {
46. **return** n-m;
47. } **else** **if** (x == '\*') {
48. **return** m\*n;
49. } **else** {
50. **if** (m == 0) {
51. puts("ERROR:divisor is zero");
52. exit(0);
53. }
54. **return** n/m;
55. }
56. }
58. **int** ExpressionEvaluation(**char** \*s){
59. **static** **int** StackInt[N];
60. **static** **char** StackChar[N];
61. **int** topInt = 0, topChar = 0;
62. **int** len = strlen(s);
63. s[len] = '#'; s[len+1] = '\0';
64. **char** c = s[0], x;
65. **int** k = 1, m, n;
66. StackChar[++topChar] = '#';
67. **while** (c != '#' || StackChar[topChar] != '#') {  //表达式未读完或者运算未完
68. **if** (c >= '0' && c <= '9') {
69. **int** y = 0;
70. **while** (c >= '0' && c <= '9') {  // 读入连续的数字
71. y = y\*10+(c-'0');
72. c = s[k++];
73. }
74. StackInt[++topInt] = y; // 把读进的数字入数字栈
75. } **else** {
76. **switch** (Procede(StackChar[topChar], c)) {
77. **case** '<':  //栈顶元素优先权低
78. StackChar[++topChar] = c;
79. c = s[k++];
80. **break**;
81. **case** '=':
82. --topChar;  // 脱括号
83. c = s[k++];  // 读入下一个字符
84. **break**;
85. **case** '>':  //退栈并将运算结果入栈
86. x = StackChar[topChar--];
87. m = StackInt[topInt--];
88. n = StackInt[topInt--];
89. StackInt[++topInt] = Operate(m,n,x);
90. **break**;
91. }
92. }
93. }
94. **return** StackInt[topInt];
95. }
97. **int** main() {
98. **static** **char** s[N];
99. **while** (~scanf("%s", s)) {
100. printf("%d\n", ExpressionEvaluation(s));
101. }
102. **return** 0;
103. }