《数据结构与算法分析》
上机实验实验报告

姓名：Cantjie

班级：

学号：

联系方式：

cantjie@cantjie.com

Github:www.github.com/cantjie

题目一：

（来源：http://ica.openjudge.cn/dg1/2）（题目有误，实为波兰表达式）

描述

*逆波兰表达式是一种把运算符前置的算术表达式，例如普通的表达式 2 + 3 的逆波兰表示法为+ 2 3。逆波兰表达式的优点是运算符之间不必有优先级关系，也不必用括号改变运算次序，例如(2 + 3) * 4 的逆波兰表示法为* + 2 3 4。本题求解逆波兰表达式的值，其中运算符包括+ - * /四个。*

输入

输入为一行，其中运算符和运算数之间都用空格分隔，运算数是浮点数。

输出

输出为一行，表达式的值。

可直接用 printf("%f\n", v)输出表达式的值 v。

样例输入

** + 11.0 12.0 + 24.0 35.0*

样例输出

1357.000000

解题思路：

利用函数递归调用。当遇到数字时，将字符串转化成数字，当遇到四则运算符号，则进行运算。

代码：

```
#include<stdio.h>
#include<stdlib.h>
double RPolishExpVal() {
    char str[50];
    scanf("%s", str);
    switch (str[0])
    {
    case '*': {return(RPolishExpVal() * RPolishExpVal()); break; }
    case '+': {return(RPolishExpVal() + RPolishExpVal()); break; }
    case '-': {return(RPolishExpVal() - RPolishExpVal()); break; }
    case '/': {return(RPolishExpVal() / RPolishExpVal()); break; }
    default:return atof(str);
    }
}

int main()
{
    //逆波兰表达式求值 Reverse Polish expression
    printf("%f\n", RPolishExpVal());
    return 0;
```

```
21. }
```

题目二:

(来源：http://ica.openjudge.cn/dg3/solution/11562783/)

描述

当你站在一个迷宫里的时候，往往会被错综复杂的道路弄得失去方向感，如果你能得到迷宫地图，事情就会变得非常简单。
*假设你已经得到了一个 n*m 的迷宫的图纸，请你找出从起点到出口的最短路。*

输入

第一行是两个整数 n 和 m(1<=n,m<=100)，表示迷宫的行数和列数。
接下来 n 行，每行一个长为 m 的字符串，表示整个迷宫的布局。字符'.'表示空地，'#'表示墙，'S'表示起点,'T'表示出口。

输出

输出从起点到出口最少需要走的步数。

样例输入

3 3
S#T
.#.
...

样例输出

6

解题思路：

从起点开始向四周可以走的搜索，然后把起点到这里的长度写在这个格里同时将这个格子加入队列，以实现之后对这个点的访问。如果格子里已经有数了则取较小的一个。最后终点的数就是长度。

代码：

```
1.  #include<stdio.h>
2.  #include<stdlib.h>
3.  #include<queue>
4.  #define INT_MAX 2147483647
5.  #define MAX_MAZE_SIZE 100
6.  //http://ica.openjudge.cn/dg3/solution/11562783/
7.  using namespace std;
8.
9.  class Node {
10. public:
11.     int x;
12.     int y;
13.     ////bool Node::operator!=(const Node & another_node) const;
14. };
```

```
15.
16. class Maze {
17. private:
18.     int maze[MAX_MAZE_SIZE][MAX_MAZE_SIZE];
19.     Node start;
20.     Node end;
21.     int col_size;
22.     int row_size;
23. public:
24.     Maze();
25.     int ShortestWay();
26. };
27.
28.
29.
30. int main() {
31.     int step;
32.     Maze *MyMaze = new Maze();
33.     step = MyMaze->ShortestWay();
34.     printf("%d", step);
35. }
36.
37. Maze::Maze() {
38.     int m, n;
39.     int i = 0;
40.     int j = 0;
41.     char temp;
42.     scanf("%d %d", &n, &m);
43.     row_size = n;
44.     col_size = m;
45.     for (i = 0; i < n; i++) {
46.         for (j = 0; j < m; j++) {
47.             scanf("%c", &temp);
48.             maze[i][j] = temp;
49.             switch (temp)
50.             {
51.             case '#':
52.                 maze[i][j] = -1;
53.                 break;
54.             case '.':
55.                 maze[i][j] = INT_MAX;
56.
57.                 break;
58.             case '\n':
```

```
                j--;
                break;
            case 'S':
                maze[i][j] = 0;
                start.x = i;
                start.y = j;
                break;
            case 'T':
                maze[i][j] = INT_MAX;
                end.x = i;
                end.y = j;
                break;
            default:
                break;

            }
        }
    }
}

int Maze::ShortestWay() {
    int i = 0, j = 0;
    int count = 0;
    Node curr;
    Node next;
    queue <Node>MyQueue;
    curr = start;
    while (curr.x != end.x || curr.y != end.y) {
        count = maze[curr.x][curr.y] + 1;
        if (curr.x + 1 < row_size && maze[curr.x + 1][curr.y] > count) {
            maze[curr.x + 1][curr.y] = count;
            next.x = curr.x + 1;
            next.y = curr.y;
            MyQueue.push(next);
        }
        if (curr.x - 1 >= 0 && maze[curr.x - 1][curr.y] > count) {
            maze[curr.x - 1][curr.y] = count;
            next.x = curr.x - 1;
            next.y = curr.y;
            MyQueue.push(next);
        }
         if (curr.y + 1 < col_size && maze[curr.x][curr.y + 1] > count) {
             maze[curr.x][curr.y + 1] = count;
             next.x = curr.x;
```

```
103.             next.y = curr.y + 1;
104.             MyQueue.push(next);
105.         }
106.         if (curr.y - 1 >= 0 && maze[curr.x][curr.y - 1] > count) {
107.             maze[curr.x][curr.y - 1] = count;
108.             next.x = curr.x;
109.             next.y = curr.y - 1;
110.             MyQueue.push(next);
111.         }
112.         curr = MyQueue.front();
113.         MyQueue.pop();
114.     }
115.     return maze[end.x][end.y];
116. }
117.
118. //bool Node::operator!=(const Node & another_node) const
119. //{
120. //  if (another_node.x == x && another_node.y == y) {
121. //      return false;
122. //  }
123. //  else {
124. //      return true;
125. //  }
126. //}
```

题目三：

（来源：http://bailian.openjudge.cn/practice/4132）

(目前样例输入可以通过而测评出现 runtime error)

描述

求一个可以带括号的小学算术四则运算表达式的值

输入

一行，一个四则运算表达式。''表示乘法，'/'表示除法*

输出

一行，该表达式的值，保留小数点后面两位

样例输入

输入样例 1:

3.4

输入样例 2:

7+8.3

输入样例 3:

3+4.5(7+2)*(3)*((3+4)*(2+3.5)/(4+5))-34*(7-(2+3))*

样例输出

输出样例 1:

3.40

输出样例 2:
15.30
输出样例 3:
454.75

解题思路：

利用栈，创建一个操作数栈和一个运算符栈。当当前运算符的优先级高于栈顶运算符优先级时，将该运算符入栈；若当前运算符优先级低于栈顶运算符时，则弹出栈顶运算符和两个操作数进行计算，并把计算结果压栈，最后当运算符栈为空时，操作数栈中剩下的一个元素即为最终结果。

```
double cal(char opt, double num1, double num2) {
    switch (opt)
    {
    case '+':
        return num1 + num2;
        break;
    case '-':
        return num1 - num2;
        break;
    case '*':
        return num1*num2;
        break;
    case '/':
        return num1 / num2;
        break;
    default:
        return 0;
        break;
    }
}

int getPriority(char opt) {
    {
        int priority;
        if (opt == ')')
            priority = 3;
        else if (opt == '*' || opt == '/')
```

```
            priority = 2;
        else if (opt == '+' || opt == '-')
            priority = 1;
        else if (opt == '(')
            priority = 0;
        return priority;
    }
}

float NormExpVal() {
    char exp[300];
    bool num_tag = 0;
    int curr = 0;
    int priority = 0;
    double result;
    double temp;
    double num1, num2;
    char opt;
    stack <double>NumStack;
    stack <char>OptStack;
    scanf("%s", exp);

    while (exp[curr] != '\0') {
        if ((exp[curr] >= 48 && exp[curr] <= 57) || exp[curr] == '.') {
            if (num_tag == 0) {
                temp = atof(exp + curr);
                num_tag = 1;
                NumStack.push(temp);
            }
        }
        else {
            if (getPriority(exp[curr]) == 3) {
                opt = OptStack.top();
                OptStack.pop();
                while (getPriority(opt) != 0) {
                    num2 = NumStack.top();
                    NumStack.pop();
                    num1 = NumStack.top();
                    NumStack.pop();
                    temp = cal(opt, num1, num2);
                    NumStack.push(temp);

                    opt = OptStack.top();
                    OptStack.pop();
```

```
72.                 }
73.             }
74.             else {
75.                 //getPriority(exp[curr]) == 0
76.                 if (!OptStack.empty()) {
77.                     if (getPriority(exp[curr]) > getPriority(OptStack.top())
    || exp[curr] == '(') {
78.                         OptStack.push(exp[curr]);
79.
80.                     }
81.                     else {
82.                         opt = OptStack.top();
83.                         OptStack.pop();
84.                         num2 = NumStack.top();
85.                         NumStack.pop();
86.                         num1 = NumStack.top();
87.                         NumStack.pop();
88.                         temp = cal(opt, num1, num2);
89.                         NumStack.push(temp);
90.                         if (getPriority(exp[curr]) <= getPriority(OptStack.t
   op())) {
91.                             opt = OptStack.top();
92.                             OptStack.pop();
93.                             num2 = NumStack.top();
94.                             NumStack.pop();
95.                             num1 = NumStack.top();
96.                             NumStack.pop();
97.                             temp = cal(opt, num1, num2);
98.                             NumStack.push(temp);
99.                         }
100.                         OptStack.push(exp[curr]);
101.                     }
102.                 }
103.                 else {
104.                     OptStack.push(exp[curr]);
105.                 }
106.             }
107.             num_tag = 0;
108.         }
109.         curr++;
110.     }
111.     while (!OptStack.empty()) {
112.         opt = OptStack.top();
113.         OptStack.pop();
```

```
            num2 = NumStack.top();
            NumStack.pop();
            num1 = NumStack.top();
            NumStack.pop();
            temp = cal(opt, num1, num2);
            NumStack.push(temp);
        }
        return NumStack.top();
    }
```

其他任务：

一、 迷宫动态版

利用 DFS 和栈实现

动图、代码见附件

二、 迷宫 DFS 版

代码见附件