```c
#define _CALCU_C
#ifdef _CALCU_C
/*
*
================================================================================
=========
*       Filename:   calculating.c
*    Description:   funs about calculating
*        Version:   1.1.2
*        Created:   2017.10.26 16:05:58
*      Time Used:   10h
*  Last Modified:   2017.10.28 00:05
*    Last Change:   One error judgement deleted
*         Author:   伍瀚缘(Tree Wu), why2000@hust.edu.cn
*        Company:   Huazhong University of Science and Technology
*
================================================================================
=========
*/


#include<why_calculator.h>
//#define NOW 1
#ifdef NOW//调试用

int main(void) {


    return 0;
}

#endif //NOW
extern int execstatus;//"why_calculator.c"

//函数功能：计算结果
double calculate(char *repol) {//repol为填充完毕的逆波兰栈
    double stack[MAXSIZE];//计算栈
    char dnum[MAXSIZE];//读取浮点数
    int pol = 0;//repol读取符
    int top = -1;//栈顶符
    while (repol[pol] != '\0') {
        if ((repol[pol] >= '0'&&repol[pol] <= '9')||repol[pol]=='.') {//读取数
          字字符串
            int now = 0;
            while (repol[pol] != ' ') {
                dnum[now++] = repol[pol++];
            }
            dnum[now++] = '\0';
            stack[++top] = numbertrans(dnum);//数字字符串的转换
            pol++;
        }
        else {
            //*************************错误7：运算符过多
              ***************************
            if (top == 0) {
                return errorfound(7);
```

```
51              }
52              //**************************ERROR7*********************************
                *
53              switch (repol[pol++]) {
54              case '@'://开方运算，中缀中后读取的（stack[top]）为开方的次数，先
                读取的为被开方的数,目前不支持对负数开方
55                  if (stack[top-1]>=0) {//被开方数为负时要求开方次数为奇整数
56                      stack[top - 1] = pow(stack[top - 1], 1.0 / stack[top]);
57                      --top;
58                      break;
59                  }
60                  else if ((isint(stack[top])&&(int)stack[top]%2&&stack[top-1]
                    <0)) {
61                      stack[top - 1] = -pow(-stack[top - 1], 1.0 / stack[top]);
62                      --top;
63                      break;
64                  }
65
66                  //************************错误3：开非奇整数次方时底数小于零
                    ****************************
67                  else {
68                      return errorfound(3);
69                  }
70                  //**********************************ERROR3*********************
                    ********************
71
72              case '^'://幂运算，后读取的为指数，先读取的为底数
73                  if (stack[top-1]>=0) {
74                      stack[top - 1] = pow(stack[top - 1], stack[top]);
75                      --top;
76                      break;
77                  }
78                  else if (isint(1.0 / stack[top]) && ((int)(1.0 / stack[top]) %
                     2) && stack[top - 1] < 0) {
79                      stack[top - 1] = pow(stack[top - 1], stack[top]);
80                      --top;
81                      break;
82                  }
83                  //************************错误4：结果为虚数的幂运算
                    ****************************
84                  else {
85                      return errorfound(4);
86                  }
87                  //*******************************ERROR4*********************
                    ************
88
89              case '*'://乘法运算
90                  stack[top - 1] *= stack[top];
91                  --top;
92                  break;
93              case '/'://除法运算，后读取的为除数，先读取的为被除数
94                  if (stack[top]) {
95                      stack[top - 1] /= stack[top];
```

```
96                      --top;
97                      break;
98                  }
99                  //************************错误5：分母为0************************************
100                 else {
101                     return errorfound(5);
102                 }
103                 //*****************************ERROR5*************************************
104             case '+'://加法运算
105                 stack[top - 1] += stack[top];
106                 --top;
107                 break;
108             case '-'://减法运算
109                 stack[top - 1] -= stack[top];
110                 --top;
111                 break;
112             }
113
114         }
115
116     }
117     return stack[top];
118 }
119
120 //函数功能：整数判断
121 int isint(double n) {//
122     if (n >= 0)
123         if ((n - (int)n) < 1e-15 || (n - (int)n) < -0.999999999999999) return 1;
124         else return 0;
125     else
126         if (-(n - (int)n) < 1e-15 || -(n - (int)n) < -0.999999999999999) return 1;
127         else return 0;
128
129 }
130
131 //函数功能：将字符串形式的浮点数转换成double类型
132 double numbertrans(const char *dnum) {//dunm为字符串形式的浮点数
133     double intpart = 0, floatpart = 0, flo = 1;//整数部分，小数部分，浮点位
134     int now = 0;
135     double result = 0;
136     while (dnum[now] >= '0' && dnum[now] <= '9')
137     {
138         intpart = intpart * 10 + dnum[now++] - '0';//整数部分读取：乘十加n
139     }
140
141     if (dnum[now] == '.') //小数部分读取
142     {
143         while (dnum[++now] != '\0')
144         {
145             //*****************************错误9：多重小数点*************************
```

```
146             if (dnum[now] == '.') {
147                 return errorfound(9);
148             }
149             //**********************************ERROR9**********************************
                *****
150             floatpart = floatpart * 10 + dnum[now] - '0';
151             flo *= 10;
152         }
153     }
154 
155 
156     result = intpart + floatpart / flo;
157     return result;
158 }
159 #endif //_CALCU_C
160 
```