```c
#define _READ_C
#ifdef _READ_C
/*
*
================================================================================
=========
*       Filename:   reading.c
*    Description:   read input and change it into a reverse polish style
*        Version:   1.1.2
*        Created:   2017.10.26 16:04:47
*      Time Used:   10h
*  Last Modified:   2017.10.27 23:58
*    Last Change:   deal with dots
*         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
extern int execstatus;//"why_calculator.c"

//函数功能：储存输入的表达式(buf)用于格式化输出，并建立一个处理了负号的版本(input)以进行后续计算，同时处理特殊输入指令
int readinput(char *input, char *buf) {//input进行后续计算,buf用于格式化输出
    int i = 0,j=0;
    int minus = 0;

    while (((input[i] = fgetc(stdin))) != '\n')
    {
        buf[j] = input[i];
        if (input[i] == '-') {//非常神秘的没有验证正确性的自己脑子一抽想出来的负数的暴力处理办法
            if (i == 0) {//负号在首位时直接添0
                input[i++] = '0';
                input[i] = '-';
            }
            else if (input[i - 1] == '(') {//负号在左括号后时直接添0
            /*******这个不能和上面的合并，否则读取input[-1]会导致内存错误******/
                input[i++] = '0';
                input[i] = '-';
            }

            else{//负号在数字的后面时添置括号
                input[i++] = '+';
```

```
50                  input[i++] = '(';
51                  input[i++] = '0';
52                  input[i] = '-';
53                  ++minus;
54              }
55          }
56          if (input[i] == '+') {//顺便把正号一起处理了
57              if (i == 0) {//首位添0
58                  input[i++] = '0';
59                  input[i] = '+';
60              }
61              else if (input[i - 1] == '(') {//左括号后添0
62                  input[i++] = '0';
63                  input[i] = '+';
64              }
65          }if (input[i] == '.'&&!(input[i-1]>='0'&&input[i-1]<='9')) {//当然还有
              小数点
66                  input[i++] = '0';
67                  input[i] = '.';//任何情况下没接在数字后面的小数点只要直接变成
                    0.就行了
68
69          }
70          if (input[i] == ' ')
71          {
72              i--;
73              j--;
74          }
75          i++;
76          j++;
77      }
78      buf[j] = '\0';
79      for (int k = 1; k <= minus; k++) {//补全由于处理负数产生的括号
80          input[i++] = ')';
81      }
82      //**********************错误8：无输入*******************************
83      if (input[0]=='\n') {
84          return errorfound(8);
85      }
86      //**************************ERROR8***********************************
87
88      input[i] = '\0';
89
90
91      //特殊命令：
92      //1.exit（或quit）退出程序
93      //2.help 打开帮助界面
94      //3.change 修改精度
95      if (strcmp(input, "quit")==0 || strcmp(input, "exit")==0) {
96          strcpy(input, "");
97          return 0;
98      }
99      else if (strcmp(input, "help") == 0) {
100         outputhelp();
101         return 2;
102     }
103     else if (strcmp(input, "change") == 0) {
```

```
104         preciquest();
105         return 2;
106     }
107     else {
108         return 1;
109     }
110
111
112 }
113
114 //将中缀输入串转换并压入逆波兰栈
115 int translate(const char *input, char *repol) {//input为经部分后处理的输入，repol为逆波兰栈，用于后续运算
116     char stack[MAXSIZE];//in栈用于存符号
117     int now = 0;//input读取符
118     int top = -1;//in栈顶符,读取top为负时表示出现错误
119     int pol = 0;//repol栈写入符
120     while (input[now]!='\0') {
121         //1.括号处理
122         if (input[now] == '(') {//左括号无视优先级直接入栈
123             stack[++top] = input[now++];
124         }
125         else if (input[now] == ')') {//注意右括号本身永远不会存在于入栈中，后续不用判断其存在
126             //左括号上方符号出栈
127             while (stack[top] != '('&&top >= 0) {
128                 repol[pol++] = stack[top--];
129             }
130             //*****************************错误1：右括号多于左括号*********************************
131             if (top < 0) {
132                 repol[pol] = '\0';//结束逆波兰栈的写入，并返回0以请求继续输入
133                 return errorfound(1);
134             }
135             //*************************************ERROR1*************************************
136             --top;//舍弃左括号
137             ++now;//舍弃右括号
138         }
139         //2.读取运算符
140         else if (input[now]=='@'||input[now]=='^') {//最高级运算符@与^（开方/乘方）同级运算符及其上方的全部运算符出栈压入repol，再将本身压入入栈
141             while (top >= 0 && (stack[top] == '@' || stack[top] == '^')) {
142                 repol[pol++] = stack[top--];
143             }
144             stack[++top] = input[now++];
145         }
146         else if (input[now] == '*' || input[now] == '/') {//次高级运算符*与/将同级或高级运算符出栈压入repol,再将本身压入入栈
147             while (top >= 0 && (stack[top] == '@' || stack[top] == '^' ||
148                 stack[top] == '*' || stack[top] == '/')) {
149                 repol[pol++] = stack[top--];
150             }
151             stack[++top] = input[now++];
```

```
152         else if (input[now] == '+' || input[now] == '-') {//低级运算符+与-将左括号上方的全部运算符出栈压入repol,再将本身压入入栈
153             while (top >= 0 && stack[top] != '(') {
154                 repol[pol++] = stack[top--];
155 
156             }
157             stack[++top] = input[now++];
158         }
159         else if((input[now] >= '0'&&input[now] <= '9') || (input[now] == '.')) {//数字（含浮点）直接写入逆波兰栈
160             //此处可考虑用项目 \\codevstest\\sliver1 里写的函数进行功能拓展：其他进制读入
161             //初步想法：读取0x为十六进制，1x~fx对应1~15进制
162             while ((input[now] >= '0'&&input[now] <= '9')||input[now]=='.') {
163                 repol[pol++] = input[now++];
164             }
165             repol[pol++] = ' ';//数字读入结束标识，防止repol中压入连续数字产生混淆
166         }
167         //*****************************错误6：无法识别的输入*************************************
168         else {
169             return errorfound(6);
170         }
171         //*************************************ERROR6*****************************************
172     }
173     while (top >= 0) {//符号压入repol
174             //*************************错误2：左括号多于右括号*************************************
175             if (stack[top] == '(') {
176                 repol[pol++]='\0';//结束逆波兰栈的写入，并返回0以请求继续输入
177                 return errorfound(2);
178             }
179             //*********************************ERROR2*****************************************
180             repol[pol++] = stack[top--];
181         }
182     repol[pol++] = '\0';//逆波兰栈写入完成
183 
184 
185 }
186 
187 #endif //_READ_C
```