

constant.c

```
1 #include "constant.h"
2 #include<stdbool.h>
3 #include<string.h>
4 #include<stdio.h>
5 #include <stdlib.h>
6 #include <ctype.h>
7 #include<stdlib.h>
8 #include <ctype.h>
9
10 int COND_OPERATORS_SIZE = 6;
11 char COND_OPERATORS[6][5] = {
12     "lt", "gt", "eq", "neq", "le", "ge"
13 };
14
15 int ARITHMATIC_OPERATORS_SIZE = 6;
16 char ARITHMATIC_OPERATORS[6][5] = {
17     "add", "sub", "mul", "div", "dif", "rem"
18 };
19
20 bool isArithOp(char *ch){
21     for(int i=0; i<ARITHMATIC_OPERATORS_SIZE; i++){
22         if( strcmp(ARITHMATIC_OPERATORS[i], ch) == 0 ) return true;
23     }
24     return false;
25 }
26
27 bool isCondOpValid(char* op){
28     //char arr[6][5] = { "lt", "gt", "eq", "neq", "le", "ge" };
29     for(int i=0; i<COND_OPERATORS_SIZE; i++){
30         if(strcmp(COND_OPERATORS[i],op) == 0) return true;
31     }
32     return false;
33 }
34
35 bool isConditionValid(double left, char* op, double right){
36     if( strcmp(op,"lt",2) == 0) return (left < right);
37     if( strcmp(op,"gt",2) == 0) return (left > right);
38     if( strcmp(op,"eq",2) == 0) return (left == right);
39
40     if( strcmp(op,"le",2) == 0) return (left <= right);
41     if( strcmp(op,"ge",2) == 0) return (left >= right);
42     if( strcmp(op,"neq",2) == 0) return (left != right); // 2 fine also
43 }
44
45
46 double getValue(double left, double right, char *op){
47     // "add", "sub"
48     if( strcmp(op,ARITHMATIC_OPERATORS[0],3) == 0 ) {
49         printf("\nadd %lf %lf\n",left,right);
50         return left+right;
51     }
52     if( strcmp(op,ARITHMATIC_OPERATORS[1],3) == 0 ) return left-right;
53
54     // "mul", "div"
55     if( strcmp(op,ARITHMATIC_OPERATORS[2],3) == 0 ) return left*right;
56     if( strcmp(op,ARITHMATIC_OPERATORS[3],3) == 0 ) return left/right;
57
58     // "dif", "rem"
59     if( strcmp(op,ARITHMATIC_OPERATORS[4],3) == 0 ) return left>right ? left-right : right-
left;
60     if( strcmp(op,ARITHMATIC_OPERATORS[5],3) == 0 ) return ((int)left) % ((int)right);
61
62 }
63
64 char* trim(char *s) {
65     // Trim trailing whitespace
66     char *end = s + strlen(s) - 1;
67     while(end > s && isspace((unsigned char)*end)) {
68         end--;
69     }
```

```
70 *(end + 1) = '\0';
71
72 // Trim leading whitespace
73 while(*s && isspace((unsigned char)*s)) {
74     s++;
75 }
76
77 return s;
78 }
79
80 void processStatement(const char *stmt, char *leftVar, char *varBeforeOp, char *operator,
char *varAfterOp) {
81     char buffer[100];
82     int len = strlen(stmt);
83     int bufPos = 0;
84     int state = 0;
85     // 0: searching for leftVar,
86     // 1: searching for operator,
87     // 2: searching for varAfterOp
88     for (int i = 0; i < len; i++) {
89         char ch = stmt[i];
90
91         // Skip spaces
92         if (isspace(ch) || ch == '=') {
93             if (bufPos > 0) {
94                 buffer[bufPos] = '\0';
95                 switch (state) {
96                     case 0:
97                         strcpy(leftVar, buffer);
98                         state++;
99                         break;
100                     case 1:
101                         if( isArithOp(buffer) ){
102                             strcpy(operator, buffer);
103                             state++;
104                         }
105                     else {
106                         strcpy(varBeforeOp, buffer);
107                     }
108                 }
109                 break;
110             case 2:
111                 strcpy(varAfterOp, buffer);
112                 state++;
113                 break;
114             }
115             bufPos = 0;
116         }
117         continue;
118     }
119     if (isalnum(ch) || ch == '.' ) {
120         buffer[bufPos++] = ch;
121     }
122 }
123
124
125 // Handle any remaining items in the buffer
126 if (bufPos > 0) {
127     buffer[bufPos] = '\0';
128     if(state == 2){
129         strcpy(varAfterOp, buffer);
130     }
131 }
132
133 //printf("leftVar:~%s-, varBeforeOp: ~%s-, operator:~%s-, varAfterOp: ~%s-\n", leftVar,
varBeforeOp, operator, varAfterOp);
134 }
135
```