constant.c

```c

#include "constant.h"
#include<stdbool.h>
#include<string.h>
#include<stdio.h>
#include <stddef.h>
#include<stdlib.h>
#include <ctype.h>

int COND_OPERATORS_SIZE = 6;
char COND_OPERATORS[6][5] = {
    "lt","gt","eq","neq","le","ge"
};

int ARITHMATIC_OPERATORS_SIZE = 6;
char ARITHMATIC_OPERATORS[6][5] = {
    "add", "sub", "mul", "div", "dif", "rem"
};

bool isArithOp(char *ch){
    for(int i=0; i<ARITHMATIC_OPERATORS_SIZE; i++){
        if( strcmp(ARITHMATIC_OPERATORS[i], ch) == 0 ) return true;
    }
    return false;
}

bool isCondOpValid(char* op){
    //char arr[6][5] = { "lt","gt","eq","neq","le","ge" };
    for(int i=0; i<COND_OPERATORS_SIZE; i++){
        if(strcmp(COND_OPERATORS[i],op) == 0) return true;
    }
    return false;
}

bool isConditionValid(double left, char* op, double right){
    if( strncmp(op,"lt",2) == 0) return (left < right);
    if( strncmp(op,"gt",2) == 0) return (left > right);
    if( strncmp(op,"eq",2) == 0) return (left == right);

    if( strncmp(op,"le",2) == 0) return (left <= right);
    if( strncmp(op,"ge",2) == 0) return (left >= right);
    if( strncmp(op,"neq",2) == 0) return (left != right); // 2 fine also
}


double getValue(double left, double right, char *op){
    // "add", "sub"
    if( strncmp(op,ARITHMATIC_OPERATORS[0],3) == 0 ) {
        printf("\nadd %lf %lf\n",left,right);
        return left+right;
    }
    if( strncmp(op,ARITHMATIC_OPERATORS[1],3) == 0 ) return left-right;

    // "mul", "div"
    if( strncmp(op,ARITHMATIC_OPERATORS[2],3) == 0 ) return left*right;
    if( strncmp(op,ARITHMATIC_OPERATORS[3],3) == 0 ) return left/right;

    // "dif", "rem"
    if( strncmp(op,ARITHMATIC_OPERATORS[4],3) == 0 ) return left>=right ? left-right : right-left;
    if( strncmp(op,ARITHMATIC_OPERATORS[5],3) == 0 ) return ((int)left) % ((int)right);

}

char* trim(char *s) {
    // Trim trailing whitespace
    char *end = s + strlen(s) - 1;
    while(end > s && isspace((unsigned char)*end)) {
        end--;
    }
    *(end + 1) = '\0';

    // Trim leading whitespace
    while(*s && isspace((unsigned char)*s)) {
        s++;
    }

    return s;
}

void processStatement(const char *stmt, char *leftVar, char *varBeforeOp, char *operator, char *varAfterOp) {
    char buffer[100];
    int len = strlen(stmt);
    int bufPos = 0;
    int state = 0;
    // 0: searching for leftVar,
    // 1: searching for operator,
    // 2: searching for varAfterOp

    for (int i = 0; i < len; i++) {
        char ch = stmt[i];

        // Skip spaces
        if (isspace(ch) || ch == '=') {
            if (bufPos > 0) {
                buffer[bufPos] = '\0';
                switch (state) {
                    case 0:
                        strcpy(leftVar, buffer);
                        state++;
                        break;
                    case 1:
                        if( isArithOp(buffer) ){
                            strcpy(operator, buffer);
                            state++;
                        }
                        else {
                            strcpy(varBeforeOp, buffer);
                        }
                        break;
                    case 2:
                        strcpy(varAfterOp, buffer);
                        state++;
                        break;
                }
                bufPos = 0;
            }
            continue;
        }

        if (isalnum(ch) || ch == '.') {
            buffer[bufPos++] = ch;
        }
    }

    // Handle any remaining items in the buffer
    if (bufPos > 0) {
        buffer[bufPos] = '\0';
        if(state == 2){
            strcpy(varAfterOp, buffer);
        }
    }

    //printf("leftVar:-%s-, varBeforeOp: -%s-, operator:-%s-, varAfterOp: -%s-\n", leftVar, varBeforeOp, operator, varAfterOp);
}

```