SecondOrderIn.C

```
// File to deal with second order systems
#include <stdio.h> // Including stdio.h header file

void clear(void);
// Declaring the function to clear the input buffer
int askUserSecond(struct transfer_function_second_order *t2, int i, float
step); // Declaring the function to clear the input buffer
void DisplaySystemSecond(struct transfer_function_second_order *t2, int
i);      // declaring the displayy function for the second order system

struct transfer_function_second_order // Defining the transfer function
structure for the second order system
{
    float a; // Defining the numeratior
    float b; // Defining the first s term denominator
    float c; // Defining the first pole term denominator
    float d; // Defining the second s term denominator
    float e; // Defining the second pole term denominator
};

int askUserSecond(struct transfer_function_second_order *t2, int i, float
step)
{
    float alph;
// Initialising the temporary alpha variable, used to calculate the
numerator of the open loop system transfer function
    printf("\nPlease enter the transfer function of the system you wish
to design the controller for:\n"); // Tells the user this is now the
transfer function values they're entering
    printf("\ta\n ----------------\n (bs + c)(ds + e)\n");
// Shows the user the format of their entry
    printf("Note all values must be between -10000 and 10000\n");
// Lets the userknow the range of valid inputs
    do
// Loops through at least once, as the do while is initiated
    {
        printf("a:\t");
// Asks the user to enter the value of the numerator
        if (scanf("%f", &alph) != 1 || (alph > 10000 || alph < -10000))
// Prompts the user for an answer and checks the answer is numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the input is invalid
        };
        clear(); // Clears the buffer
    } while (alph > 10000 || alph < -10000); // Keeps looping through
until a valid answer is entered
    t2[i].a = (alph * step); // Calculates the value of the numerator
using the step input and alph vairable, and stores it in the transfer
function structure's a value
    do                      // Loops through at least once, as the do
while is initiated
    {
        printf("b:\t");
// Asks the user to enter the value of the first s term in the
denominator
```

```
        if (scanf("%f", &t2[i].b) != 1 || (t2[i].b > 10000 || t2[i].b < -
10000)) // Prompts the user for an answer and checks the answer is
numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the user enters an invalid
input
        };
        clear(); // Clears the buffer
    } while (t2[i].b > 10000 || t2[i].b < -10000); // Keeps looping
through until a valid answer is entered
    do // Loops through at least once, as the do while is initiated
    {
        printf("c:\t");
// Asks the user to enter the first pole
        if (scanf("%f", &t2[i].c) != 1 || (t2[i].c > 10000 || t2[i].c < -
10000)) // Prompts the user for an answer and checks the answer is
numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the user enters an invalid
input
        };
        clear(); // Clears the buffer
    } while (t2[i].c > 10000 || t2[i].c < -10000); // Keeps looping
through until a valid answer is entered
    do // Loops through at least once, as the do while is initiated
    {
        printf("d:\t");
// Asks the user to enter the value of the second s term in the
denominator
        if (scanf("%f", &t2[i].d) != 1 || (t2[i].d > 10000 || t2[i].d < -
10000)) // Prompts the user for an answer and checks the answer is
numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the user enters an invalid
input
        };
        clear(); // Clears the buffer
    } while (t2[i].d > 10000 || t2[i].d < -10000); // Keeps looping
through until a valid answer is entered
    do // Loops through at least once, as the do while is initiated
    {
        printf("e:\t");
// Asks the user to enrer the value of the second pole
        if (scanf("%f", &t2[i].e) != 1 || (t2[i].e > 10000 || t2[i].e < -
10000)) // Prompts the user for an answer and checks the answer is
numerical
        {
            printf("Invalid input, please enter a number between -10000
and 10000\n"); // Error message thrown if the user enters an invalid
input
        };
        clear(); // Clears the buffer
    } while (t2[i].e > 10000 || t2[i].e < -10000); // Keeps looping
through until a valid answer is entered
```

```
    return 0; // Signifies the end of the function
};

void DisplaySystemSecond(struct transfer_function_second_order *t2, int
i) // Definition of the dsiplay function for the second order system
{
    printf("You have entered the folloing transfer function:\n");
// Calls the userInputs function to get the user inputs for the
controller design
    printf("\t\t%f\n ----------------------------------------------\n
(%.5f * S + %.5f)(%.5f * S + %.5f)\n\n", t2[i].a, t2[i].b, t2[i].c,
t2[i].d, t2[i].e); // displays the values
};

void clear(void) // Function definition for the clear function
{
    int c = 0; // Defining the input variable used to clear the buffer
    do         // Runs at least once
    {
        c = getchar(); // Gets the next character in the buffer
    } while (c != '\n'); // loops through until it reaches a new line
}; // End of function definition for clearing
```