fullCont.C

```
// This is a program desinged to enable the user to design  a series of
controllers to fit their requirements
// The design will be based off of a vaiety of user information about the
system and desired charicteristics of the controller.
// Ideally the user will use this to save costs and time, to decide which
controllers to test further
// This is not a replacement for MatLab
// Written by Bailey Raven

#include <stdio.h>  // Including stdio.h header file
#include <math.h>   // Including math.h header file
#include <tgmath.h> // Including tgmath.h header file

#include "Displays.C" // Includes display.C file

// Declaring the clearing function to clear the input buffer
void clearing(void);

int main() // The main program to add full functionality
{
    struct transfer_function_first_order t1[20];  // Initialising 20 t1
structs
    struct transfer_function_second_order t2[20]; // Initiallising 20 t2
structs
    struct systemOuts s2[20];                     // Initialising 20 s2
structs
    struct parameters p2[20];                     // Initialising 20 p2
structs
    struct controllers c2[20];                    // Initialising 20 c2
structs

    int i = 1;
// Initiating the counting variable to number each iteration
    int choice;
// Variable to allow the user to loop through
    printf("Welcome, This program is designed to compare controller types
for different continuous time systems.\n"); // Welocome message briefly
descibing the system
    printf("You can design up to 20 systems, that are either 1st or 2nd
order. \n");                                  // Detailing the systems'
constraints
    do
// Runs at least once
    {
        int ans = 1; // Defining the ans variable used to allow the user
to choose between designing a new controller or viewing a previous one
        if (i > 1)   // Allows the user to choose between view a previous
controller or choose to make a new one if there is already at least one
previous controller
        {

            do // Loops through at least once, as the do while is
initiated
            {
                printf("Would you like to design a new set of controllers
or view a preious systems' set? (new = 1/previous = 2)\n"); // Asks the
```

```
user if they would like to design a new controller or view a previous
one, and gives the relevant numerical inputs
                if (scanf("%d", &ans) != 1 || (ans != 1 && ans != 2))
// Prompts the user for an answer and checks the answer is numerical
                {
                    printf("Invalid input, please enter a number between
1 and 2\n"); // Error message thrown if the user enters an invalid input
                };
                clearing(); // Clears the buffer
            } while (ans != 1 && ans != 2); // Loops through if the user
enters any value that isnt 1 or 2
        }
        if (ans == 2) // If the user chooses to view a previous
controller
        {
            int range;
// Range variable for the order of systems the user wants to see
            printf("Would you like to view a specific system [0] or a set
of systems by order [1] - All 1st order or All 2nd order.\n"); // Giving
the user the option
            do
// Runs at least once
            {
                if (scanf("%d", &range) != 1 || (range != 0 && range !=
1)) // Decides wether to throw the error and propts the user
                {
                    printf("Invalid input, please enter either 1 or 0
depending on your answer.\n"); // Error message
                };
            } while (range != 0 && range != 1); // Loops through if the
user enters a value out of range
            if (range == 0) // Run if the user wants to view a specific
system
            {

                int old; // Defining the variable to allow the user to
specify which controller
                do       // Loops through at least once, as the do while
is initiated
                {
                    printf("Which previous system do you want to view?
(enter a value between 1 and %d)\n", i - 1); // Asks the user which
controller they would like to view from the first to the current
                    if (scanf("%d", &old) != 1 || (old < 1 || old > i))
// Prompts the user for an answer and checks the answer is numerical
                    {
                        printf("Invalid input, please enter a number
between 1 and %d\n", i - 1); // Error message thrown if the user enters
an invalid input
                    }
                    clearing(); // Clears the buffer
                } while (old < 1 || old > i); // Loops through until the
user enters a valid controller number from 0 to the curretn value of i
                if (s2[old].order == 1) // Runs if the system is a 1st
order
                {
                    DisplaySystemFirst(&t1[old], old);
// Displays the corresponding first order system
```

```
                    controller_display(&s2[old], &c2[old], &p2[old],
old); // Displays the related controllers
                }
                else // Runs if the system is a 2nd order
                {
                    DisplaySystemSecond(&t2[old], old);
// Displays the corresponding second order system
                    controller_display(&s2[old], &c2[old], &p2[old],
old); // Displays related controllers
                };
            }
            else // Runs if the user wants to view by order of system
            {
                int systemOrder;
// Initialises the system order variable
                printf("Which previous order would you like to view? (1
or 2)\n"); // Asks the user which system order they'd like to view
                do
// Loops through at least once
                {
                    if (scanf("%d", &systemOrder) != 1 || (systemOrder !=
1 && systemOrder != 2)) // Runs the error message if answer out of range,
and propts for an answer
                    {
                        printf("Invalid input, please enter a 1st or 2nd
order system\n"); // error message
                    };
                } while (systemOrder != 1 && systemOrder != 2); // keeps
looping through until the answer is valid
                if (systemOrder == 1) // Runs if the system is a first
order
                {
                    for (int j = 1; j <= i; j++) // Loops through all
systems previouly created
                    {
                        if (s2[j].order == 1) // Runs if the system order
is 1
                        {
                            DisplaySystemFirst(&t1[j], j);
// displays the system
                            controller_display(&s2[j], &c2[j], &p2[j],
j); // displays the controllers
                        };
                    };
                }
                else // Runs if the system is a second order
                {
                    for (int j = 1; j <= i; j++) // Loops through all
previously created systems
                    {
                        if (s2[j].order == 2) // Runs if the system is a
second order
                        {
                            DisplaySystemSecond(&t2[j], j);
// displays the system
                            controller_display(&s2[j], &c2[j], &p2[j],
j); // controller display
                        };
```

```
                    };
                };
            };
        }
        else // Runs if the user wants to design a new set of controllers
        {
            printf("Please enter the order of the system you wish to
design.\n"); // Asks the user to enter the order of the system
            do
// Runs at least once
            {
                if (scanf("%d", &s2[i].order) != 1 || (s2[i].order != 1
&& s2[i].order != 2)) // Throws the error if answer not 1 or 2
                {
                    printf("Invalid system type, this program only works
for 1st and 2nd order sytems.\n"); // error message
                    printf("Please enter a valid system type.\n");
// error message
                };
            } while (s2[i].order != 1 && s2[i].order != 2); // loops
through until it gets a valid answer
            if (s2[i].order == 1) // Runs if a 1st order system
            {
                printf("First, please can you enter the desireed output
parameters for the compensated system.\n"); // Tells the user what to
enter
                float step = outputParams(&p2[i], i);
// assigns the step value to step, and gets relevant parameters from the
user
                askUserFirst(&t1[i], i, step);
// User inputs system info
                DisplaySystemFirst(&t1[i], i);
// Dispalys system info back
                printf("Now enter the types of controllers you want to
design for this system.\n");                 // Tells the user what to
enter
                controllerSelection(&c2[i], i);
// User selects controllers they want to design
                PIDLLcalcs1st(&t1[i], &p2[i], &c2[i], &s2[i], i);
// Calculates the values for the required controllers
                controller_display(&s2[i], &c2[i], &p2[i], i);
// Displays the calculated controllers
                i++;
// Adds 1 to the iteration variable
            }
            else // Runs if a 2nd order system
            {
                printf("First, please can you enter the desireed output
parameters for the compensated system.\n"); // Tells the user what to
enter
                float step = outputParams(&p2[i], i);
// assigns the step value to step, and gets relevant parameters from the
user
                askUserSecond(&t2[i], i, step);
// User inputs system info
                DisplaySystemSecond(&t2[i], i);
// Dispalys system info back
```

```
                printf("Now enter the types of controllers you want to
design for this system.\n");                 // Tells the user what to
enter
                controllerSelection(&c2[i], i);
// User selects controllers they want to design
                PIDLLcalcs2nd(&t2[i], &p2[i], &c2[i], &s2[i], i);
// Calculates the values for the required controllers
                controller_display(&s2[i], &c2[i], &p2[i], i);
// Displays the calculated controllers
                i++;
// Adds 1 to the iteration variable
            };
        };

        if (i < 20) // only asks the user if they are able to go again,
if not the program is ended
        {
            do // Loops through at least once, as the do while is
initiated
            {
                printf("\nWould you like to analyse another system or
view previous one again? (yes = 1/no = 0)\n"); // Asks the user if they
would like to go through again
                if (scanf("%d", &choice) != 1)
// Prompts the user for an answer and checks the answer is numerical
                {
                    printf("Invalid input, please enter either 1 and
2\n"); // Error message thrown if the user enters an invalid input
                };
                clearing(); // Clears the buffer
            } while (choice != 1 && choice != 0); // Keeps the user
looping till they enter a valid choice
        };
    } while (choice == 1); // Keeps iterating through if there is room
for more controllers and the user choses to continue

    return 0; // Signifies the end of the program
};

void clearing(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
```