John Seredick
10/30/2020

Project Outline

Void handle_binary(double *input1, int count)

- Prompt user for another number using step_1 again
- Simple list of if statements to check what operation we are doing
- Check if the operation with the 2 numbers is not possible such as 1/0 and return error
- Print output to terminal and print to file.

Void handle_unary(char *input1)

- Simple list of if statements to check what operation we are doing
- Check if operation is possible or not and return error if it is not
- Print output to terminal and print to file

Void clear_all(double *input)

- Make use of a for loop to change all values of input to 0

Void recall(double *input)

- Not sure.. This isn't explained in the project

Void store(double *input)

- Not sure.. This isn't explained in the project

Double Step_1(double *inpu, int count) function:

```
Int subtract == 1;
char buf[256];
printf("Please input a number, or 'pi', 'e', 'c', 'ca', 'q' or
'b':  ");
fgets(buf, 255, stdin);
// remove_crlf(buf);
printf("%s", buf);
sscanf(buf, "%lf", input[count]);
printf("%lf", input);
printf("%s\n", buf);
```

- Prompt for a number, assign that to a buf. Check the buf if there is an %lf, if not, assign to the string.
- If string = pi, input=pi value
- If string = e, input=e value

- If string = q break;
- If string = b
    - If count = 0; retry input because no previous numbers
    - Else print type b again or hit enter to use input[count-subtract]
    - If enter is hit; input[count] = input[count-subtract]
    - If b check if count-subtract < 0 if it is not continue previous steps
    - We need some form of loop for this, possibly a while loop, while input is b and exit when input is enter.

- If string = c
    - Print - and return to step 1 again

- If string = ca
    - Clear storage variables 0-9 and return to step one
    - Use a for loop to wipe all variables stored

- Return the input value

Step 1:
Inside main:

Int count = 0;
Double input1[256]
Input[count] = step_1(input[], count)

do{
Step 2
Inside main:
- Set char operation[256]; This will be for the input
- Set an int bool = 0; for if the operation is binary or not. Will check this later for what function to call.
- Prompt for an operation,

- If input = + or addition || input = - or subtract || input = x or multiplication multiply || input = / or division || input = ^ or exponent || input = sto or store
- bool = 1;

- Else If input = sin or sine || input = cos or cosine || input = tan or tangent || input = arcsin or inverse sine || input = arccos or inverse cosine || input = root or square root || input = abs or remove sign || input = inv or 1/x || input = log or log10x || input = log2 or log2x
- bool = 0

- Else If input = rcl or recall call recall function
- Else If input = c print 0 return to step 1
- Else If input = ca clear all call a clear all function and return to step 1
- Else If input = q or quit break loop

- Else return to step 1
- If bool = 1 call handle_binary(input[count],)
- Else call handle_unary(input)

while(!strcasecmp(input, “q”) == 0 && !strcasecmp(input, “quit”) == 0)

I may change around and take things outside of main into a step 2 function. I will be using a while loop of course for the program to run repeatedly until it is prompted to quit. I will use for loops to clear all storage and for the b input situation. I will probably space out some of what is inside main into their own functions.

I know how to do pretty much all of this. I am a bit confused on some of the aspects what like “c” does and “sto” since sto is not explained in the project. I do not know necessarily how I will align the arrays or if I need to even. Because if a function is binary, that adds another number to either that array or a new one. So I need some sort of logic to tell if a number was used along with another number or not. Maybe an array of 0’s as fillers and a number at the same element and compare if the element of input1 has a 0 or a number at the element of the other array.