ASTE 404
HW 4
Nikita Persikov
9/16/2021

Part 1:

Case	Times(s)
C++ Debug Mode, double	0.244027
C++ Release Mode, double	0.113249
C++ Debug Mode, float	0.240614
C++ Release Mode, float	0.109996
MATLAB, for-loop	0.401157
MATLAB, vectors	0.182246
Python, for-loop	26.8813
Python, vectors	0.142062

Part 2:

a) On line 4, r is being set to 2*x +y. On the line before, r is defined as a reference to x, which means **line 4 is actually setting a value to variable x through its reference r.** Therefore, the value of x is 2*1.2+4, which is equal to **6.4.**

b) The function init() just takes the values of a and b and does things with them, but does not change the very variables themselves. Therefore, **c will just be the sum of the default double values, which is 0+0=0**. For c to equal 3, a and b can be declared in the same scope as the functions like this:

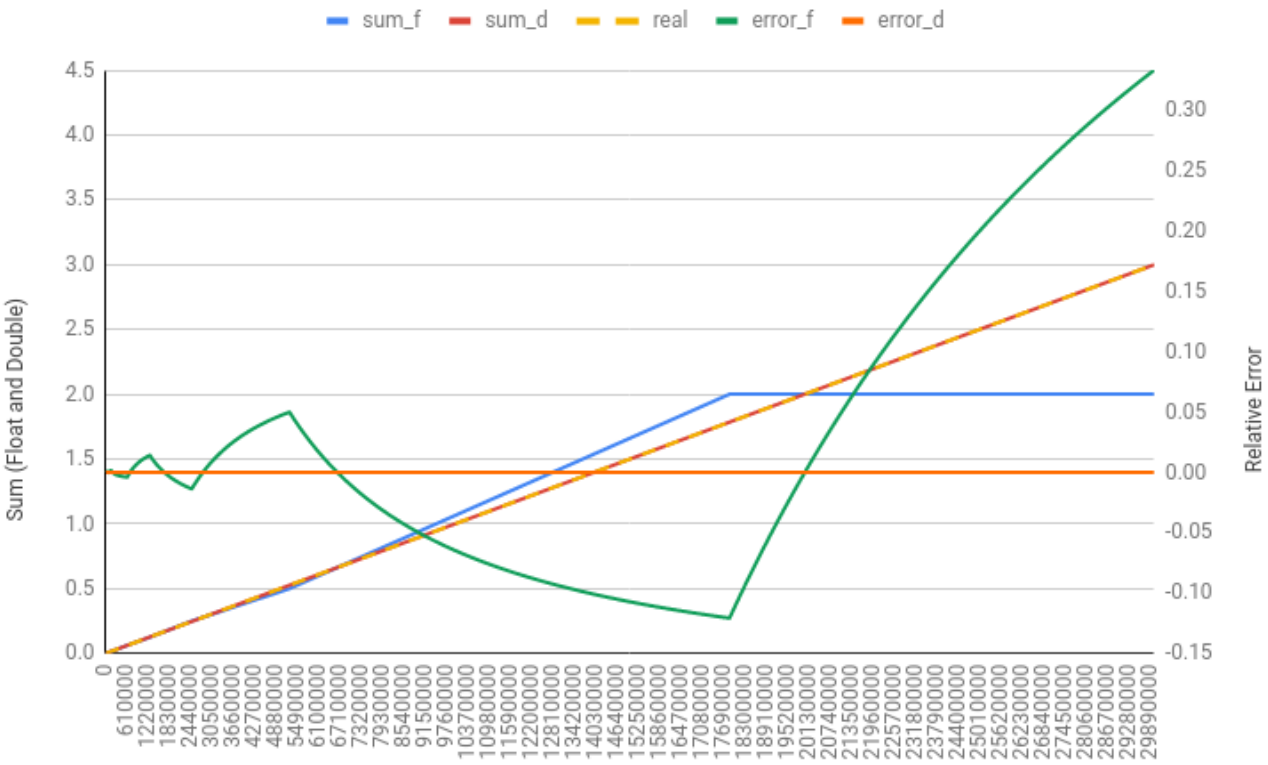
```
double a, b;
void init() {        a = 2; b = 1;      }
```

This way, when init() is run in main() (without arguments), a and b will actually be set and c will become 3.

c) In this case, the addresses of a and b are passed to init, which uses them as pointer variables. When a and b are set in init(), the variables at those memory locations are modified, so when c is calculated, it uses the same values since it pulls them from the same memory address. Therefore, **c is equal to 3.**

d) In this case, init() takes in the variables and treats them not as pointers, but as memory addresses, which the pointers in part c were set equal to. Since the values are applied to the variables at the input memory addresses, the variables updated in init() are in the same memory location as the ones in main(). Therefore, **c will again be 3.**

Part 3:


sum_f
sum_d
real
error_f
error_d
Sum (Float and Double)
Relative Error
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.30
0.25
0.20
0.15
0.10
0.05
0.00
-0.05
-0.10
-0.15
0
610000
1220000
1830000
2440000
3050000
3660000
4270000
4880000
5490000
6100000
6710000
7320000
7930000
8540000
9150000
9760000
10370000
10980000
11590000
12200000
12810000
13420000
14030000
14640000
15250000
15860000
16470000
17080000
17690000
18300000
18910000
19520000
20130000
20740000
21350000
21960000
22570000
23180000
23790000
24400000
25010000
25620000
26230000
26840000
27450000
28060000
28670000
29280000
29890000