1. The operation would take longer as that would through a divide by zero trap and the program would need to go through a series of other functions to handle the trap error. In this case it returns a value that would represent infinity. The effect of this is relatively small, but aft a lot of iterations of this will slow down the process significantly.
2. Alternatively we could use a switch case, or a probably better solution is to use a try catch statement, which would allow you to handle any run time errors the system throws like divide by zero. Fortunately I believe in the assembly layer of C++ and C# there is a trap statement that already handles this error without requiring the programmer to code for this issue. (Note: I only tested it for C# but I am assuming the same is with C++)
3. Assuming the giving starting values, the denominator only has 1 instance at the last iteration where it hits zero so the extra steps that it needs to take only happens once which is insignificant in the total iteration and for human time standards. Now the reason why the doWith function always seems to take a little bit longer is due to that additional check that it does at every iteration. This extract check takes a few CPU cycles for the program to process the if statement. This is what we see in that extra few milliseconds.