Итак, приступим. e20 с точностью до 7 знаков после запятой имеет вид:

В сумме имеем 53 знака, то есть при выделении порядка (который будет равен 28) имеем мантиссу:

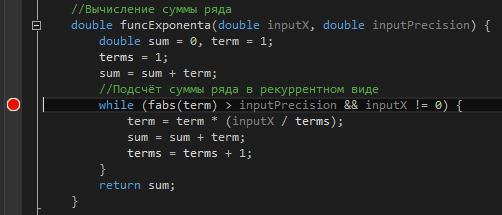
Под порядок отведены 11 бит (со 2-ого по 12-ый). Порядок записывается со сдвигом – 1023, получаем 1023 + 28 = 105110 = 100000110112. Число положительное, поэтому бит знака равен нулю. Итого имеем следующую запись числа с двойной точностью согласно IEEE 754:

Оно однозначно декодируется обратно, только дробная часть будет иметь другой:

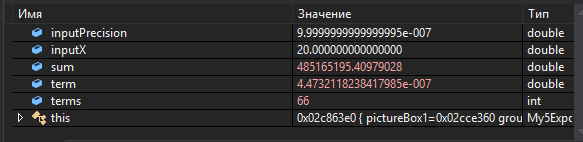
Однако в нём узнаётся число 0.4097903.

Таким образом, проблема заключается не в невозможности отобразить это число – что-то не так с алгоритмом.

Я поставил точку остановки в следующем месте программы:

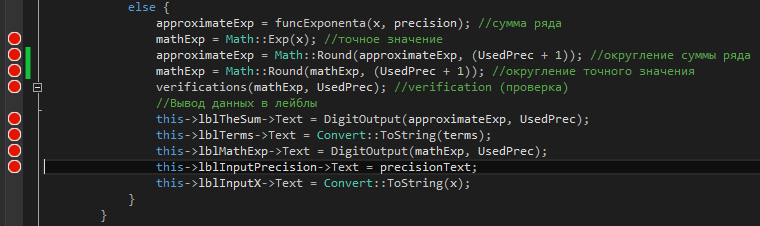


Это написанная мною функция для подсчёта суммы ряда. Если выполнить программу с Х = 20 и максимальной точностью, то узнаем, что при подсчёте используется всего 66 слагаемых. Дойдя до этого этапа, можно увидеть возвращаемое функцией значением переменной с плавающей запятой двойной точности (double) sum. Её значение представлено ниже:

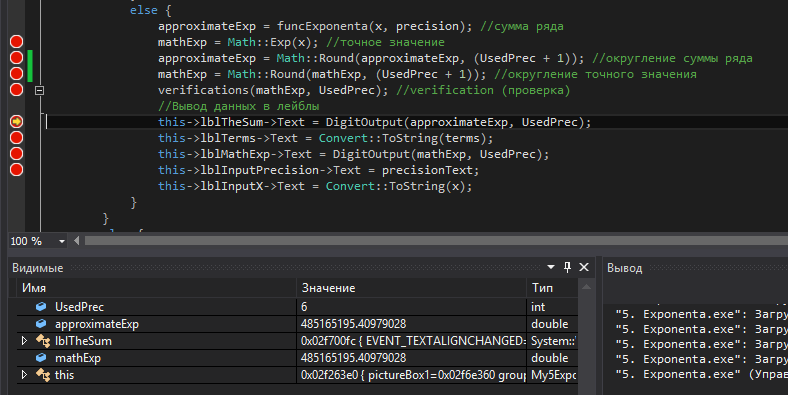


Как видно, в переменной хранятся данные вплоть до 8 знака (большую точность для таких больших цифр переменная содержать, к сожалению, не может).

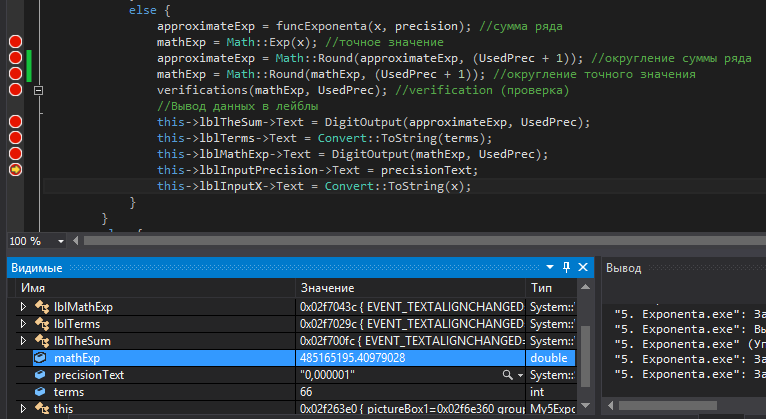
Перейдём к другому участку программы, где переменные округляются:



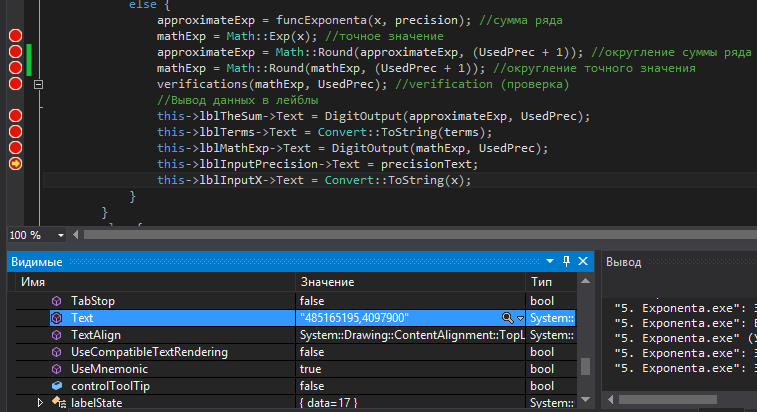
В данном участке кода переменные должны переопределяться, но после выполнения блока команд, в своей среде переменные не переопределились и всё равно продолжают быть равны 485165195.40979028. Это видно на следующем скриншоте:



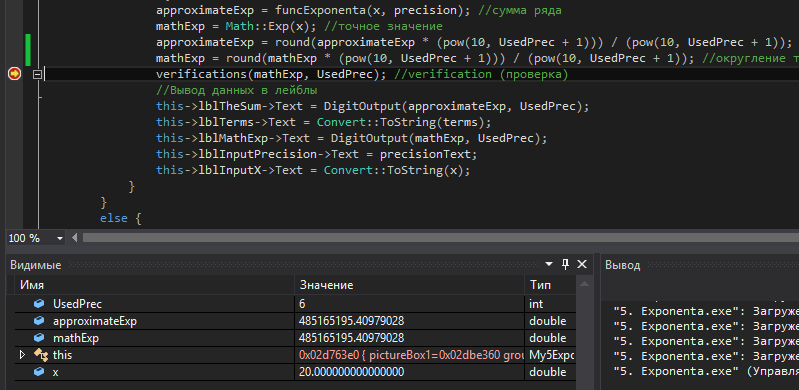
И в конце выполнения всех блоков программы, на которых расставлены точки остановок переменные не изменились:



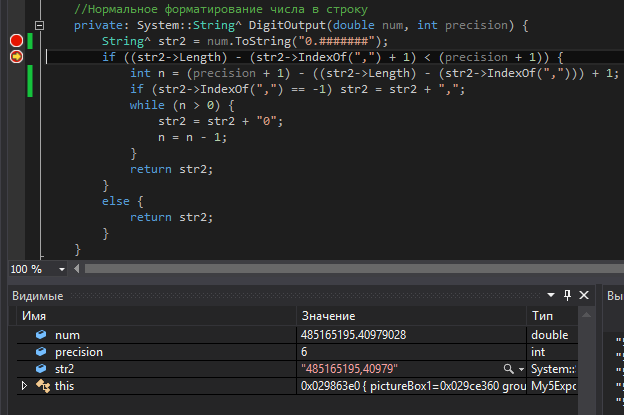
Однако в Text элементов Label попало округлённое значение переменных:



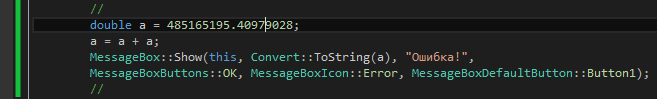
С нулями, после преобразования с помощью написанной мною функции DigitOutput. Вывод: проблема с округлением. Я попробовал заменить её аналогом из библиотеки cmath – round, однако это не сыграло большой роли – переменные всё равно не переопределялись:



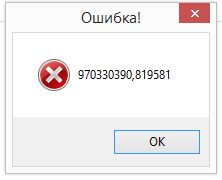
Однако на этапе конвертации в строку они становились округлёнными, хотя своё значение не меняли:



Тогда я понял – дело в двойной точности. Компилятор просто отбрасывает последние числа после выполнения задачи. Простейший пример: умножить то же число на 2. Для этого я написал простой код:

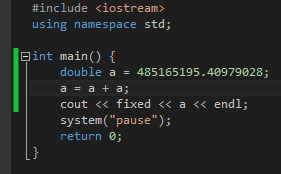


Выполнив который, программа выдала следующий результат:

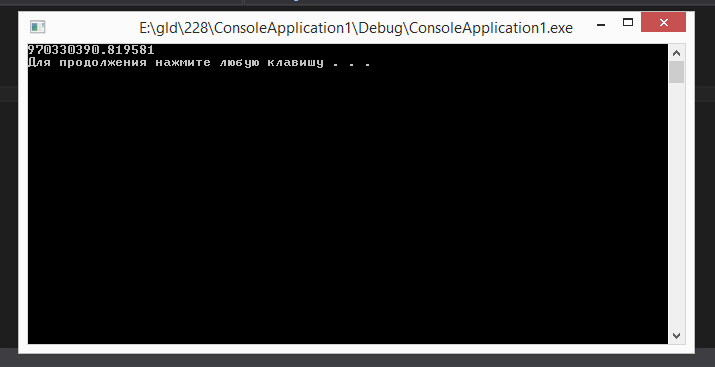


Хотя точное значение будет равняться 970330390,81958056. Выходит, компилятор при конвертации в текст переменной сам же её и округлил.

Изначально я думал, что это связано именно с средой CLR, поэтому набросал простенькую консольную программу:



Результат был тот же:



Таким образом, проблема отбрасывания чисел стоит за невозможностью хранить большое количество знаков после запятой, и сам компилятор округляет их и отбрасывает.