# 1.Время выполнения

В бибилиотеке ctime есть функция clock, которая возвращает количество тактов процессора, прошедших с момента запуска программы. (1)

С другой стороны, в библиотеке chrono есть простые функции для подсчёта времени работы того или иного куска кода. (2)

Такт процессора или такт ядра процессора — промежуток между двумя импульсами тактового генератора, который синхронизирует выполнение всех операций процессора.

Гипотеза: поделив (1) на (2) я получил время одного такта, то есть период колебания частотного резонатора. Период и частота – взаимнообратные величины. Таким образом, я смогу вычислить частоту своего процессора.

Мой процессор Intel Core i7 920 использует частоту шины 140 МГц и множитель 20, в результате чего тактовая частота равна 2800 МГц. = 2.8 ГГЦ до 4.9 ГГЦ (Turbo Mode). (\*)

Включу программу в 1 режиме. Там происходит заполнение огромного массива огромными числами, что требует высокой производительности ЦП. При чём я обратил внимание на то, что заполнение одного и того же вектора всегда происходит за РАЗНОЕ время (незначительно, конечно). Основным результатом было то, что я получил значения частоты даже не близкие к искомым (\*).

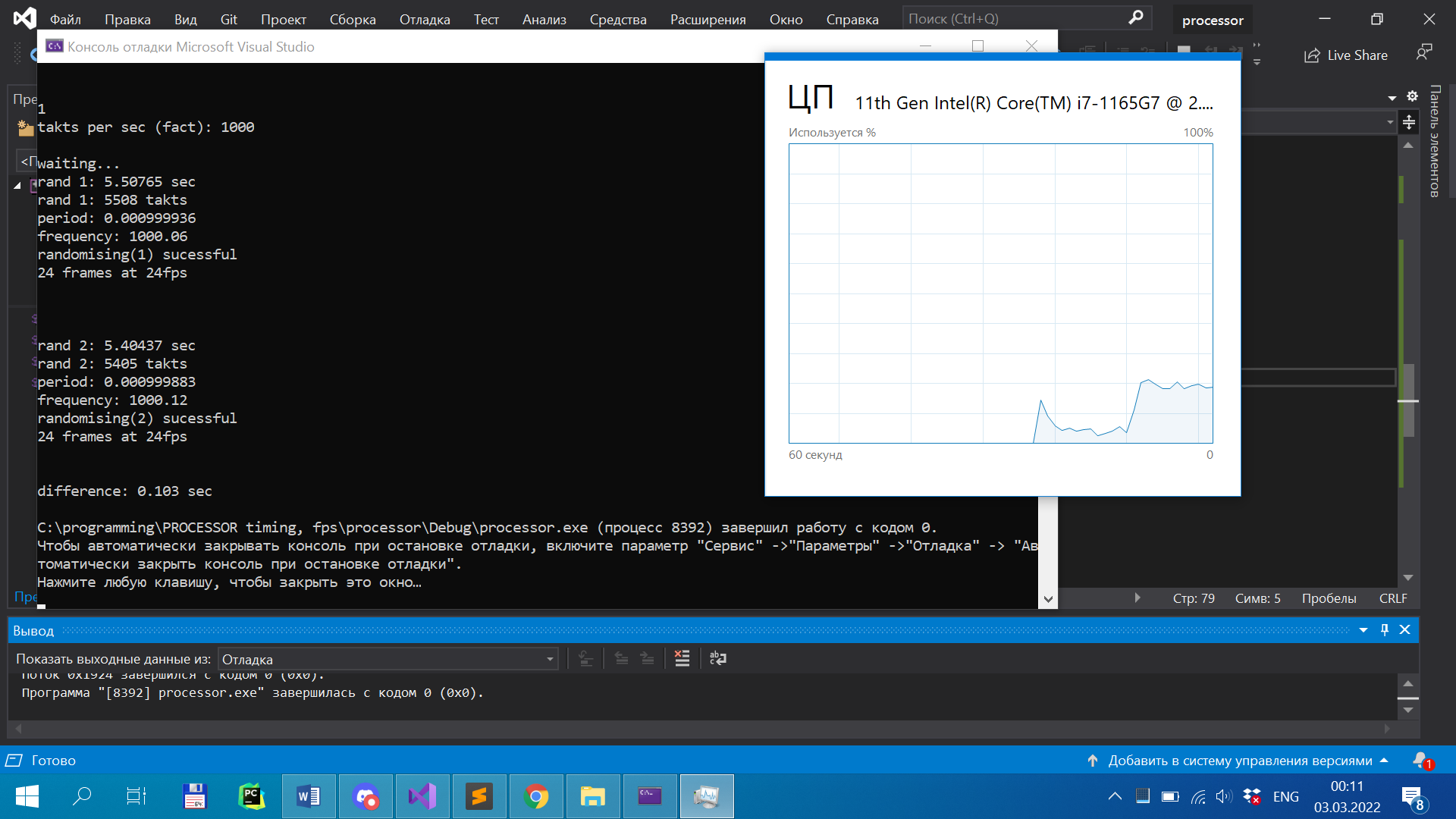
Также при искусственном изменении частоты процессора (в настройках ноута) время выполнения программы глобально не изменялось.

From Taks Manager (наблюдения):

- средняя частота без проги - 1.55 ГГЦ,

- макс при вкл. проге - 3.55 ГГЦ (3550 МГц),

- на с++ выделено 2 ГГЦ = 2 миллиарда циклов в секунду.



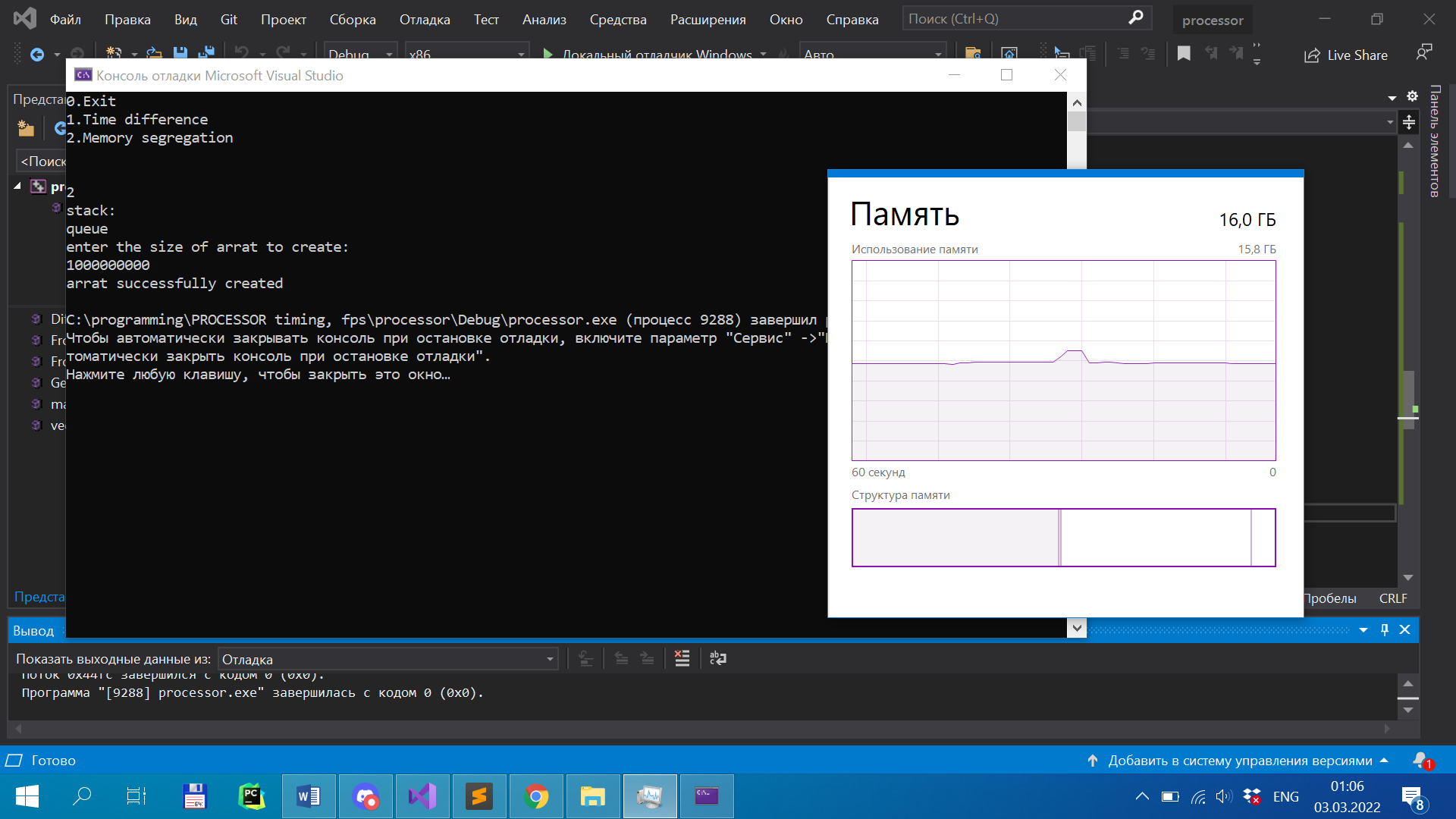
Из этого я сделал вывод, что операционная система (как посредник между моим кодом и железом) не выделяет сразу всю производительность процессора (ровно также как и память) под мои запросы на вычислительные мощности. Даже средние значения не совпадают с макросом CLOCKS\_PER\_SEC от библиотеки <ctime>. Не буду углубляться в API «диалоги» и системные вызовы, но ВЕРОЯТНО операционка изначально «выдаёт» компилятору не больше некоторой определённой вычислительной мощности, на которую я, как программист, повлиять не могу.

# 2. Использование памяти

Второй пункт эксперимента состоял в том, чтобы проверить какой максимальный размер памяти операционная система готова выделить для моей компилируемой программы.

В стеке я не смог выделить массив больше **1.5 Мб** данных (массива char на 1.500.000 ячеек), что ничтожно мало для современных компьютеров. Соизмеримо объёму фото.

В то время как куча дала глобально разнящиеся результаты.



Как видно из скриншота, в динамической памяти моя программа смогла занять примерно **1 Гб** памяти (при чём на моём устройстве почему-то ни на бит больше). Объём существенно увеличился.

Также было измерено время выполнения всех операций при последовательном выделении нескольких массивов из кучи. На удивление, самыми длинно выполняемыми стали операции malloc и delete для перераспределения памяти (65% времени).