|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ |
| **«Национальный исследовательский ядерный университет «МИФИ»**  **(НИЯУ МИФИ)** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (№12) |

Лабораторная работа №4

«Реализация на CUDA»

по дисциплине «Гибридные суперкомпьютерные технологии»

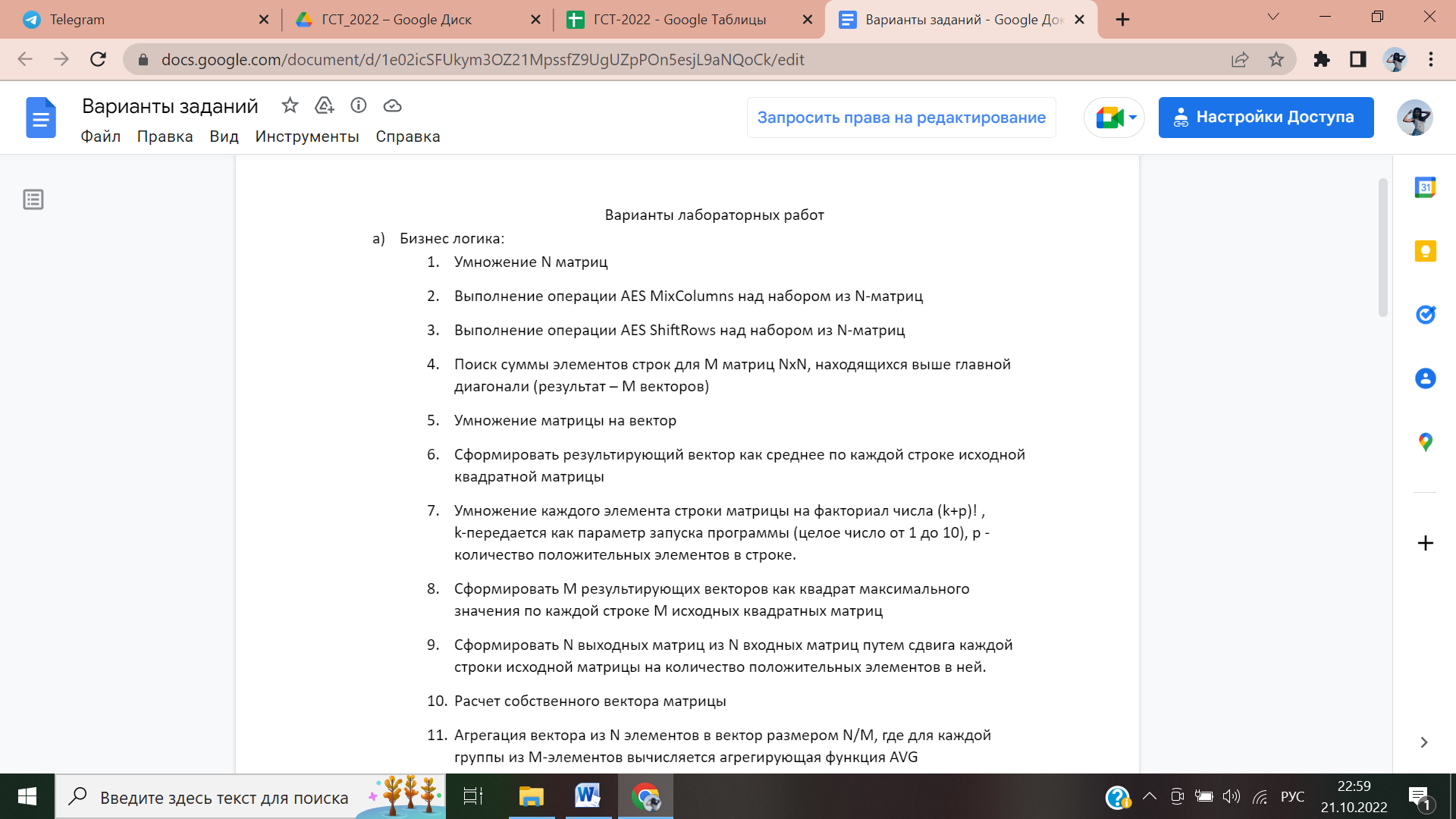
Работу выполнила студентка группы М21-502:

Нургазиева Д.Р.

Проверил: Синельников Д.М.

2022

**Цель:** изучить основы реализация программы на CUDA. Выполнить задание по варианту 5:



Особенности исполнения в каждом из вариантов:

* Считывание данных происходит из файла (либо данные передаются по протоколу TCP)
* Данные генерируются утилитой, принимающей в качестве параметров размер данных для обработки в мегабайтах и имя файла (TCP хост-порт) куда будут выгружены данные
* Программа выполняет бизнес-логику и записывает результат в выходной файл (отправляет данные на порт возврата результатов программы-генератора по TCP, сохранение файла с результатами осуществляет программа-генератор)
* В конце файла с результатами сохраняется информация о времени выполнения вычислений и размере обработанных данных

# **Ход работы:**

Программа состоит из двух подпрограмм:

1. Generator.c – отвечает за генерацию данных;
2. Counter.c – выполняет умножение матрицы на вектор, подсчитывает время.

В первую очередь запускается generator.c и вводится размер файла и имя файла. Генерируется матрица с цифрами от 0 до 100. Сгенерированные данные грузятся файл

Далее запускается counter.c и вводится размер файла и имя файла.

Происходит: 1. Выделение памяти под массив, матрицу. 2. перемножение матрицы на вектор, в рамках этого процесса подсчитывается время выполнения. Также происходит реализация через CUDA. Приницип работы представлен на рисунке 1.

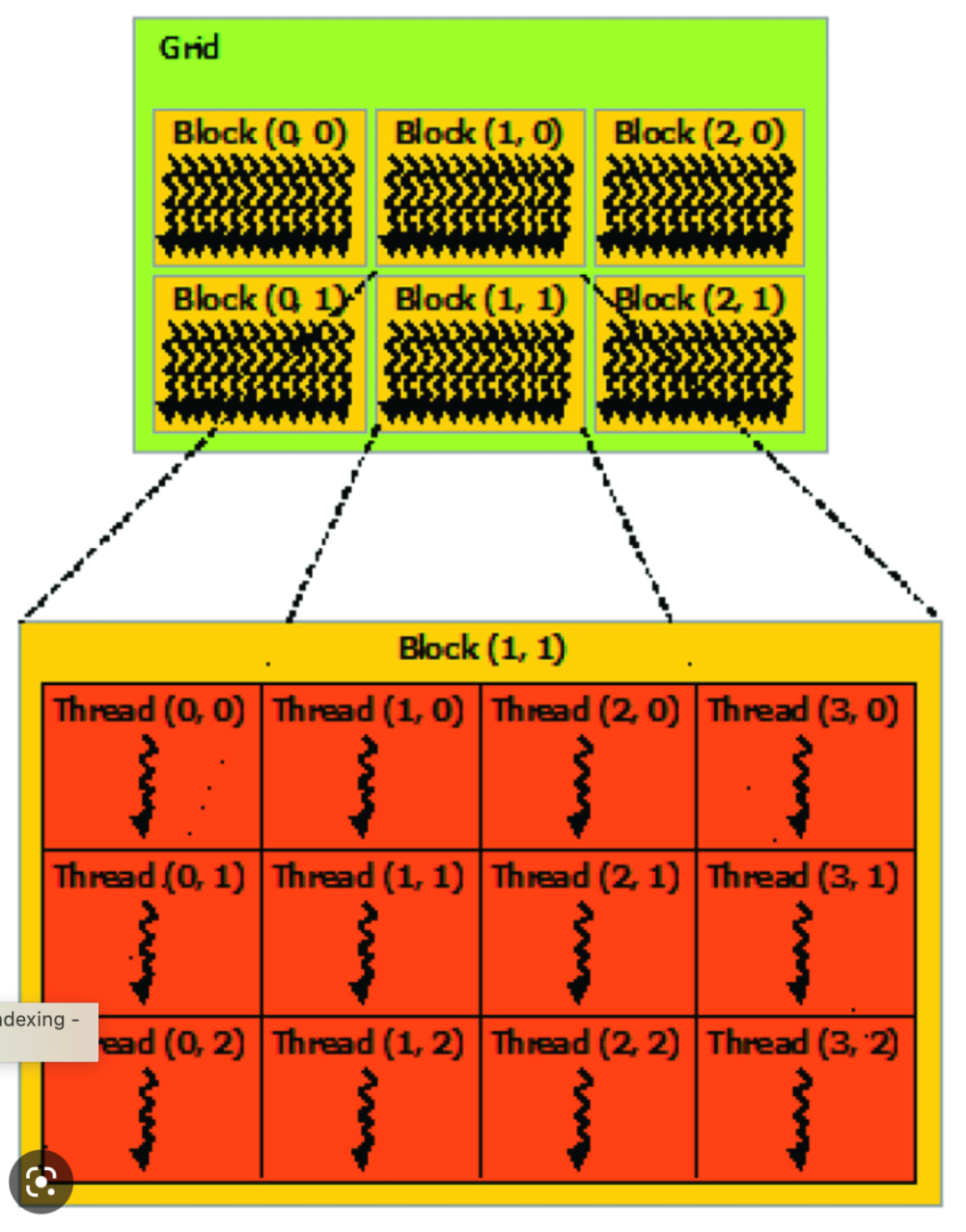


Рисунок 1 - CUDA

### Настраиваем параметры

Запуск программы:

nvcc -arch=sm\_35 -o counter counter.c

\*\*cudaMalloc - выделение памяти в gpu

\*\*cudaMemcpyAsync - копируем данные из программы в область gpu

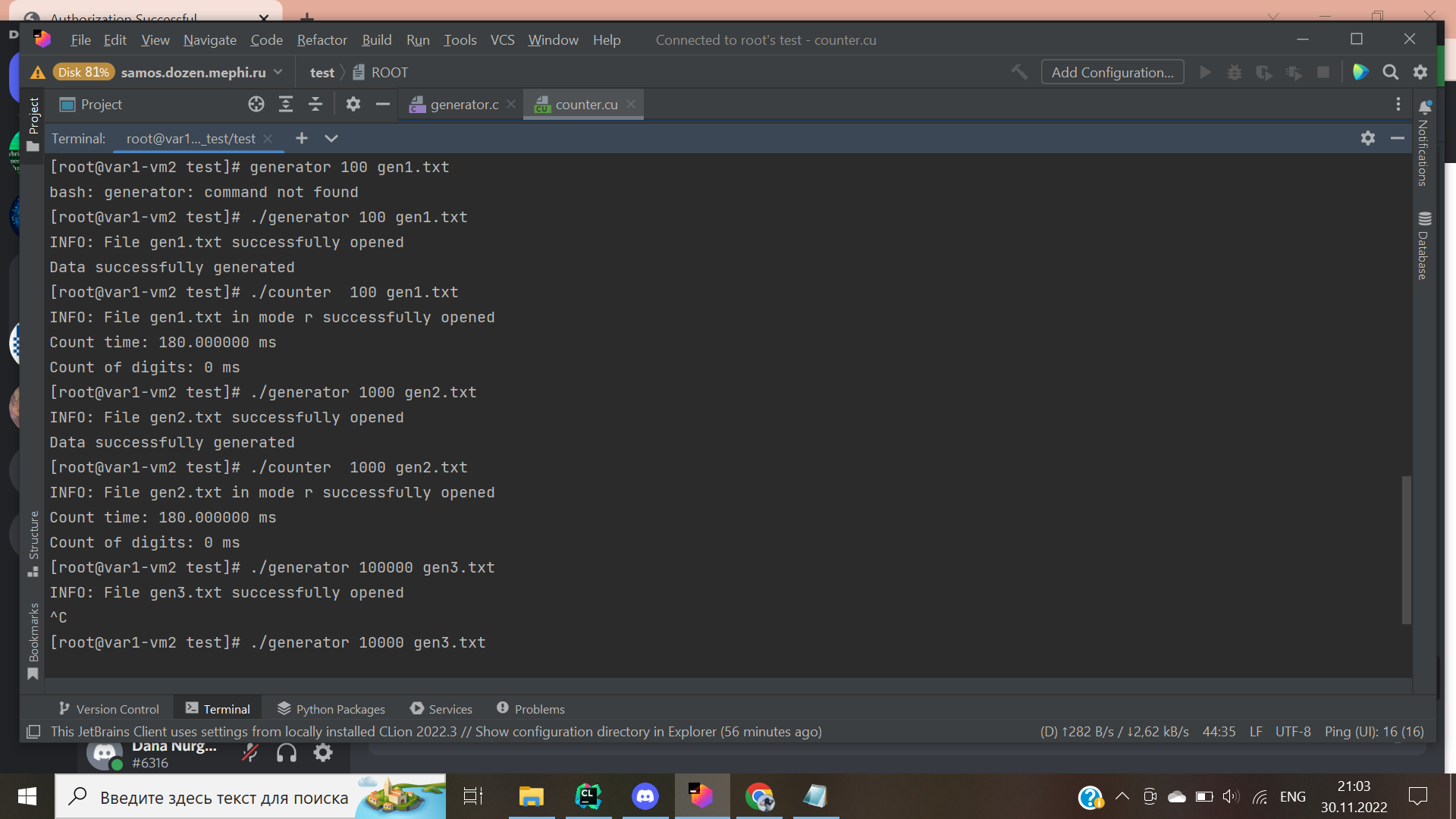
int block\_size = 16;

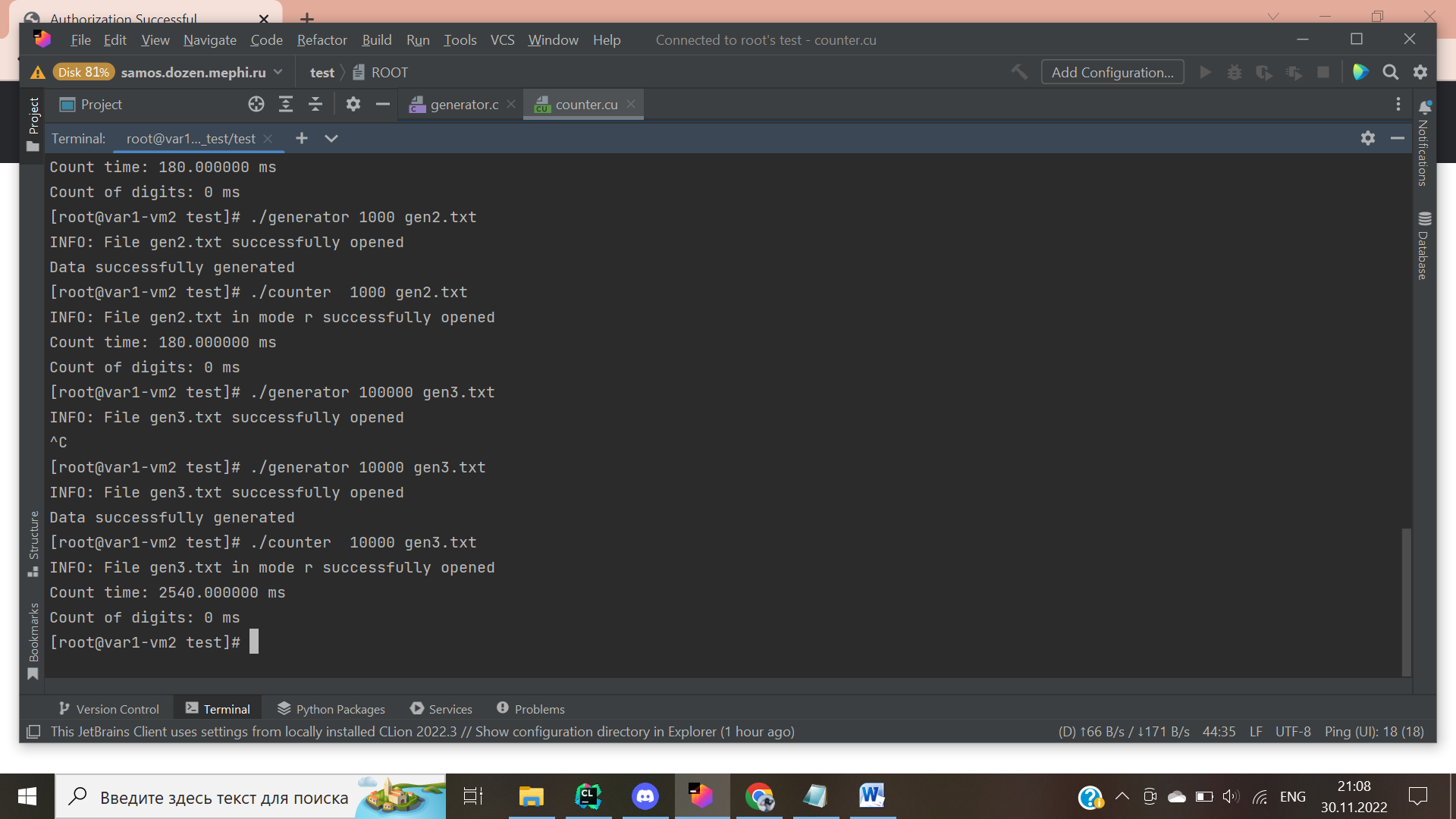
int grid\_size = (res\_buf + block\_size) / block\_size;

dim3 DimGrid(grid\_size, 1,1);

dim3 DimBlock(block\_size, 1,1);

**Выполнение программы:**





**Также результирующая матрица выводится в файл. Файл приложен в папке.**

Сравнительная таблица параллельной реализации, реализации через MPI, CUDA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод | Время (мс)на 1мб | Время (мс)на 32мб | Время (мс)на 64мб | Время (мс)на 128мб | Время (мс)на 512мб |
| Параллельный | 10 | 50 | 120 | 260 | 1080 |
| OpenMP | 0 | 40 | 90 | 190 | 860 |
| MPI | 10 | 110 |  | 310 |  |
| CUDA | При 100  =  180 | При 10000  =  2540 |  |  |  |

**Вывод:** были изучены основы реализации CUDA. Проанализировано время выполнения программы при разных подходах.