Тестовое задание:

Необходимо разработать систему, состоящую из нескольких сущностей: программы на языке C++, скрипта Python, и скрипта Bash.

Описание задачи:

Есть несколько файлов в разных, но известных форматах.
Внутри файлов записаны данные о точках на плоскости.
Каждый файл может содержать несколько групп таких точек, различающихся идентификатором группы.
Необходимо с помощью запуска скрипта bash прочитать все переданные в скрипт файлы с помощью запуска программы на C++, выход программы на C++ передать в скрипт Python, который должен сгенерировать картинку, отображающую точки из каждого файла таким образом, что можно идентифицировать к какому файлу и какой группе в файле относится каждая точка (например с помощью легенды на графике).

Описание форматов файлов:

Координаты: всегда неотрицательные целые числа.
Группа: любое уникальное значение соответствующее формату. Например для бинарного файла это будут значения от 0 до 255.

1. Текстовый файл (`*.txt`):
   на каждой строке файла сожержится одна точка с указанием группы и координат `<group>:<x>,<y>` :

```
t1:1,2
t2:3,4
```

(имя группы не может включать в себя двоеточие и символ переноса строки)

2. Бинарный файл (`*.bin`):
   содержит такие же данные, что и текстовый, но в бинарном формате:

```
struct PointToFile {
  uint32_t group:8;
  uint32_t x:12;
  uint32_t y:12;
};
```

т.е. каждые 32 бита файла- это описание одной точки. У файла нет заголовка и других структур внутри.

3. Файл в формате JSON (`*.json`).
   Содержит те же данные, что и остальные файлы

```
{
  "points": [
      {
          "group": "1",
          "x": 1,
          "y": 1
      },
      {
          "group": "2",
          "x": 2,
          "y": 2
```

```
        }
    ]
}
```

Требования к составляющим системы:

1. Программа на языке C++ должна:
  - принять список файлов в качестве входа
  - извлечь информацию в соответствии с форматом (расширением) каждого файла, вероятно тут будет смысл рассмотреть подход с фабрикой парсеров.
  - передать извлеченную информацию в скрипт Python любым способом
  - в случае возникновения ошибки программа должна вернуть код возврата отличный от 0 и вывести причину ошибки в стандартный поток вывода
2. Скрипт на Python должен:
  - принять информацию, полученную после запуска программы на C++
  - извлечь информацию соответствующим образом
  - построить по извлеченной информации график типа scatter например с помощью matplotlib и сохранить его в файл формата png
  - в случае возникновения ошибки программа должна вернуть код возврата отличный от 0 и вывести причину ошибки в стандартный поток вывода
3. Скрипт bash должен:
  - принять на вход список файлов для анализа (`:> bash thescript.sh file1.bin file2.txt file3.json file4.bin`)
  - запустить последовательно программу на C++ и скрипт на Python
  - вернуть 0 если все прошло хорошо, и не 0 если произошла ошибка
4. В директории с исходными кодами должен быть файл README, содержащий особенности использования, например необходимые пакеты Python.

Требования к системе сборки C++:

В качестве системы сборки должен использоваться CMake.

Требования к сборке и запуску:

Вся сборка и установка должна происходить с помощью инструментов cmake.
Сборку можно объединить в отдельный скрипт типа build_my_project.sh, в который вынести все этапы сборки.
Инструкция по сборке должна быть в файле README.
По результату сборки должна быть сформирована директория в рамках домашней директории пользователя, содержащая все три сущности (запускаемая программа, скрипт запуска, скрипт Python, возможно какие-то дополнительные зависимости в виде библиотек).
Этого можно достичь например с помощью install CMake или каким-то другим образом.
Программы могут использовать дополнительные модули и библиотеки, которые возможно поставить из базовых репозиториев, или поставляемых в виде исходного кода.

Требования к среде сборки и запуска:

Программа должна корректно собираться и запускаться в следующих условиях:
1. ОС Linux Ubuntu 22/04
2. компилятор GCC >=11
3. Python >=3.10
4. стандарт C++ >=11
5. CMake >=3.16

Дополнительное задание:
Парсинг файлов сделать параллельным.