Тестовое задание

Имеется заготовка класса `ResultWriter`, который необходимо реализовать:

```
template<typename Key, typename Value, typename R>
concept KVRange =
        std::ranges::range<R> &&
        requires(R r)
        {
            { std::get<0>(*std::ranges::begin(r)) } ->
std::convertible_to<Key>;
            { std::get<1>(*std::ranges::begin(r)) } ->
std::convertible_to<Value>;
        };

using value_type = std::variant<std::int64_t, double, std::string>;

template<typename T>
concept Dictionary = KVRange<std::string, value_type, T>;

using Connection = /* some sharable mongo connection type */;

class ResultWriter {
public:
    template<Dictionary T>
    explicit ResultWriter(Connection connection, const T &metadata);

    template<Dictionary T>
    void write(const T &result);
};
```

В его публичном интерфейсе имеется шаблонные конструктор и метод `write`. Конструктор принимает `Connection`, соединение с mongodb (реальный тип выбрать самому), и `Dictionary metadata`. При создании объекта должен создаваться документ в mongodb следующего формата:

```
{
    "metadata": {
        // metadata provided in constructor
    },
    "results": [
        { /* some result */ },
        ...
    ]
}
```

То есть данные из `metadata` должны быть записаны в соответствующее поле созданного документа.

Метод `write` принимает `Dictionary result`, и должен на каждый вызов добавлять в поле документа `results` объект, содержащий данные из `result`.

Ключи Dictionary имеют формат `top_key.inner_key1.....inner_keyN` и должны быть сериализованы в виде вложенных объектов.

Пример использования

```
std::unordered_map<std::string, value_type> metadata = {
    {"key1", 1},
    {"key2.inner_key", 2.0},
    {"key3", "value3"}
};

ResultWriter writer(connection, metadata);

std::map<std::string, int> result1 = {
    {"key1", 1},
    {"key2", 2},
    {"key3.boom", 3}
};

writer.write(result1);

std::unordered_map<std::string, double> result2 = {
    {"key1.q.e.r", 234.2},
    {"key2.tttt", 5},
    {"key3", .32332}
};

writer.write(result2);
```

В результате должен быть создан следующий документ в mongodb:

```
{
    "metadata": {
        "key1": 1,
        "key2": { "inner_key": 2.0 },
        "key3": "value3"
    },
    "results": [
        {
            "key1": 1,
            "key2": 2,
            "key3": { "boom": 3 }
        },
        {
            "key1": { "q": { "e": { "r": 234.2 } } },
            "key2": { "tttt": 5 },
            "key3": 0.32332
        }
    ]
}
```

Для работы с mongodb использовать библиотеку [`mongo-cxx-driver`](https://github.com/mongodb/mongo-cxx-driver).

Что ожидается

Как результат ожидается cmake проект с использованием c++23 и gcc версии не менее 13, который включает в себя реализацию класса `ResultWriter`. Проект должен включать readme в формате Markdown с описанием работы и инструкцией по сборке (должна быть возможность собрать на ubuntu 22.04).

Так же должны присутствовать несколько тестов для проверки корректности работы класса (они могут быть написаны в качестве функций и вызываться в main).

Код должен быть чистым, при необходимости можно вводить дополнительные классы, поля и методы, не меняющие публичный интерфейс `ResultWriter`. Код должен быть логически разнесен по файлам.

Mongodb должна подниматься с помощью docker compose. Инструкцию по его поднятию тоже добавить в readme.