**Цель:**  
Реализовать потокобезопасный класс FileCollector, который собирает файлы, передаваемые по сети чанками в произвольном порядке.

**Условия:**

* Сетевое взаимодействие реализовывать **не требуется**.
* Файл передаётся чанками, содержащими данные и позицию в файле.
* Чанки могут приходить **в произвольном порядке**.
* **Один файл может собираться несколькими потоками одновременно**.
* **Гарантируется, что содержимое файлов не изменяется в процессе передачи**, но **отдельные чанки могут пересекаться**
* Класс должен быть **потокобезопасным** и обеспечивать **эффективное восстановление файла** из полученных чанков.

**Пример пересекающихся чанков**

Допустим, передаётся файл размером **10 байт**, и приходят следующие чанки:

1. OnNewChunk(fileId, 0, {A, B, C, D}) – первые 4 байта (pos = 0).
2. OnNewChunk(fileId, 2, {C, D, E, F}) – пересекается с предыдущим (начиная с pos = 2).
3. OnNewChunk(fileId, 6, {G, H, I, J}) – не пересекается с предыдущими.

В памяти файл должен собираться так:

[A, B, C, D, E, F, G, H, I, J]

Заготовка класса:

Включить перенос текстаКопировать как текст

**class** FileCollector

{

**public**:

**void** CollectFile(uint32\_t fileId, size\_t fileSize);

**void** OnNewChunk(

uint32\_t fileId,

size\_t pos, *// позиция в файле*

std::vector<uint8\_t>&& chunk);

*// Метод получения собранного файла.*

*// Требуется предложить и реализовать оптимальный интерфейс.*

*// std::vector<uint8\_t> GetFile(uint32\_t fileId); // Как передавать данные?*

};

**Что нужно сделать?**

1. Оптимальный способ хранения чанков:
   * Использовать структуру данных, позволяющую эффективно вставлять чанки и восстанавливать файл.
   * Учитывать возможные разрывы (чанки приходят не по порядку).
   * Обрабатывать пересекающиеся чанки и дубликаты
2. Потокобезопасность:
   * Обеспечить синхронизацию потоков, работающих с разными файлами.
   * Один файл должен обновляться без гонок данных
3. Реализация методов:
   * CollectFile(fileId, fileSize): Создаёт структуру для хранения чанков файла.
   * OnNewChunk(fileId, pos, chunk): Добавляет чанк к файлу. Должен быть потокобезопасным.
   * Разработать оптимальный интерфейс для получения собранного файла (GetFile), учитывая:
     + Возможность частичного или полного получения данных.
     + Минимизацию копирования больших файлов.
     + Потокобезопасность при доступе к файлу.
4. Производительность и минимизация копирования данных:
   * Использование move-семантики (std::vector<uint8\_t>&&).
   * Минимизация блокировок при работе с чанками.
5. Тесты
   * Желательно использовать GTEST
6. Желательно сборочный скрипт на cmake

**Открытый вопрос:**

Какой интерфейс метода GetFile будет наиболее эффективным для передачи собранного файла?

*Предложите и обоснуйте свой вариант!* 🚀

**ЗАГРУЗКА РЕШЕНИЯ**

Все файлы решения (**вместе с исходниками**) загрузите **одним** архивом с названием ИМЯ\_ФАМИЛИЯ\_С++ в наше хранилище: [тут была ссылка]

В Skillaz подгрузите текстовый файл со словом «Выполнено» внутри