Этап 1

1. Реализованные компоненты

1.1 Класс RandomStreamGen

Генерация потока случайных строк для тестирования алгоритма HyperLogLog.

- Используется генератор Mersenne Twister (mt19937_64) для высокого качества случайных чисел
- Длина строк: равномерное распределение от 1 до 30 символов
- Набор символов: 63 символа (26 + 26 + 10 + 1 = A-Z, a-z, 0-9, -)
- Возможность воспроизведения результатов через seed

Реализованные методы:

```
// Генерация всего потока
void generateStream();

// Получение части потока по проценту (0-100%)
std::vector<std::string> getStreamPart(double percentage);

// Получение части потока по индексу
std::vector<std::string> getStreamPartByIndex(size_t end_index);

// Получение всего потока
const std::vector<std::string>& getFullStream();
```

1.2 Класс HashFuncGen

Хеширование строк с равномерным распределением по диапазону $[0, 2^{32})$

алгоритм MurmurHash3 (32-bit)

1. Один из самых быстрых нехриптографических хеш-функций
2. Низкая вероятность коллизий
3. Компактный код, легко оптимизируется компилятором

Характеристики:

- Размер выхода: 32 бита (2^32 возможных значений)
- Операции: битовые сдвиги, XOR, умножение на константы
- Финализация: дополнительное перемешивание для улучшения распределения

2. Результаты тестирования

2.1 Генерация потока данных

Параметры теста:

- Размер потока: 100,000 элементов
- Seed: 12345

Результаты:

- Все элементы успешно сгенерированы
- Распределение длин строк: практически равномерное (~3.3% на каждую длину)
- Разнообразие символов

Распределение длин:

```
Минимальный процент: 3.16% (длина 25)
Максимальный процент: 3.46% (длина 9)
Ожидаемый процент: 3.33% (100% / 30)
Отклонение: ~0.13%
```

Генератор длин работает корректно, распределение близко к равномерному

2.2 Разбиение потока

Тест: Получение 10%, 25%, 50%, 75%, 100% потока

Результаты:

```
10%:  10,000 элементов
25%:  25,000 элементов
50%:  50,000 элементов
75%:  75,000 элементов
100%: 100,000 элементов
```

Вывод: Функция разбиения работает точно.

2.3 Хеш-функция: Качество распределения

Тест коллизий:

- Всего элементов: 100,000
- Уникальных хешей: 95,672
- Коллизий: 4,328
- Процент коллизий: 4.33%

Теоретическая оценка коллизий по парадоксу дней рождения:

```
P(коллизия) ≈ 1 - e^(-n²/2m)
где n = 100,000 (количество элементов)
    m = 2^32 ≈ 4.3 × 10^9 (пространство хешей)

P(коллизия) ≈ 1 - e^(-100000²/(2×2^32)) ≈ 0.116%
```

4.33% коллизий из-за того, что в потоке есть дубликаты строк.

Реальные уникальные строки: 95,672 Сам поток содержит ~4.3% повторяющихся строк, нормально для случайной генерации.

2.4 Равномерность распределения

Тест: Распределение 100,000 хешей по 1,000 бакетам

Результаты:

```
Ожидаемое значение на бакет: 100.00
Минимум:                     67
Максимум:                    177
```

```
Стандартное отклонение:      16.89
Коэффициент вариации:        16.89%
Chi-square статистика:       2853.78
```

Для теста хи-квадрат при 999 степенях свободы (1000 бакетов - 1):

- Критическое значение при α=0.05: ~1073.64
- Наше значение: 2853.78

Наша статистика выше критического значения, но это ожидаемо при большом количестве бакетов и связано с тем, что:

1. Часть элементов в потоке - дубликаты
2. При малых ожидаемых значениях в бакетах (100) отклонения более заметны

Коэффициент вариации 16.89% указывает на разумную степень равномерности.

2.5 Динамика уникальных элементов

Наблюдения:

```
Процент потока | Уникальные элементы | Процент уникальных
---------------|---------------------|-------------------
10%            | 9,727               | 97.27%
20%            | 19,365              | 96.83%
50%            | 48,096              | 96.19%
100%           | 95,672              | 95.67%
```

По мере роста размера потока процент уникальных элементов немного снижается (с 97.27% до 95.67%), соответствует парадоксу дней рождения

3. Выводы по этапу 1

3.1 RandomStreamGen

Генерирует строки с правильным распределением длин

Поддерживает разбиение на части

Воспроизводимые результаты через seed

3.2 HashFuncGen

MurmurHash3 показывает хорошее качество распределения

Коллизии происходят только из-за дубликатов в данных

Коэффициент вариации 16.89%

Подходит для использования в HyperLogLog

5. Файлы

```
├── RandomStreamGen.h       - Заголовок генератора потоков
├── RandomStreamGen.cpp     - Реализация генератора
├── HashFuncGen.h          - Заголовок хеш-функции
```

```
├── HashFuncGen.cpp        - Реализация MurmurHash3
├── test_stage1.cpp        - Тесты этапа 1
├── Makefile              - Сборка проекта
└── 1.docx                - Документация
```