Эффективное использование линейных контейнеров

Видео: Эффективное использование вектора
12 мин

Материал для самостоятельного изучения: Класс StringSet
10 мин

Видео: Инвалидация ссылок
8 мин

Видео: Эффективное использование дека
11 мин

Видео: Инвалидация итераторов
9 мин

Тест: Вектор и дек
3 вопроса

Видео: Контейнер list
8 мин

Видео: Преимущества списка
9 мин

Видео: Неинвалидация указателя списка
7 мин

Тест: Списки
5 вопросов

Тренировочное задание по программированию: Спортсмены
1 ч

Материал для самостоятельного изучения: Решение задачи «Спортсмены»
10 мин

Видео: Контейнер array
12 мин

Видео: Внутреннее устройство массива
8 мин

Тренировочное задание по программированию: StackVector
1 ч 30 м

Материал для самостоятельного изучения: Решение задачи «StackVector»
10 мин

Видео: Класс string_view
8 мин

Видео: Удобное использование string_view
5 мин

Видео: Безопасное использование string_view
5 мин

Тест: Массивы и string_view
8 вопросов

Тренировочное задание по программированию: Translator
2 ч 30 м

Материал для самостоятельного изучения: Решение задачи Translator
10 мин

Задание по программированию: AirportCounter
3 ч

Материал для самостоятельного изучения: Решение задачи AirportCounter
10 мин

Задание по программированию: Текстовый редактор
1 ч 30 м

Материал для самостоятельного изучения: Решение задачи «Текстовый редактор»
10 мин

Задание по программированию: Статистика web-сервера
3 ч 30 м

Материал для самостоятельного изучения: Решение задачи «Статистика web-сервера»
10 мин

Материал для самостоятельного изучения: Контейнер
10 мин

Задание по программированию: Статистика web-сервера

Вы не отправили работу. Для успешной сдачи вам необходимо набрать 1/1 баллов.

Срок сдачи Выполните задание до 25 апр. г., 9:59 MSK

Инструкции | Мои работы | Обсуждения

Условие

Представим, что у нас есть web-сервер, который обслуживает запросы к интернет-магазину. Он поддерживает следующий набор запросов по протоколу HTTP:

- GET / HTTP/1.1 — получить главную страницу магазина
- POST /order HTTP/1.1 — разместить новый заказ
- POST /product HTTP/1.1 — добавить новый товар в магазин (команда админки)
- GET /order HTTP/1.1 — получить детали заказа
- PUT /product HTTP/1.1 — то же самое, что и POST /order HTTP/1.1
- GET /basket HTTP/1.1 — получить состав текущей корзины клиента
- DELETE /product HTTP/1.1 — удалить товар из интернет-магазина (команда админки)
- GET /help HTTP/1.1 — получить страницу о том, как пользоваться интернет-магазином

С точки зрения протокола HTTP, первые части приведённых выше запросов (<GET>, <POST>, <PUT>, <DELETE>) называются **методами**. Вторые части называются *URI (Uniform Resource Identifier)*. Третья часть — это версия протокола. Таким образом, наш web-сервер поддерживает 4 метода: GET, POST, PUT, DELETE и 5 URI: </>, </order>, </product>, </basket>, </help>.

Главный системный администратор нашего сервера озаботился его масштабированием и для начала он решил изучить статистику использования. Он хочет для каждого метода и каждого URI посчитать, сколько раз он встречался в запросах к серверу за последний месяц. Он попросил вас помочь с этим.

У вас уже есть какая-то кодовая база для изучения запросов к серверу, и вы хотите воспользоваться ею, чтобы сэкономить время. У вас есть заголовочный файл http_request.h, содержащий структуру HttpRequest:

```
#pragma once

#include <string_view>

using namespace std;

struct HttpRequest {
  string_view method, uri, protocol;
};
```

Кроме того, есть заголовочный файл stats.h, содержащий объявления класса Stats и функции ParseRequest:

```
#include <string_view>
#include <map>

using namespace std;

class Stats {
public:
  void AddMethod(string_view method);
  void AddUri(string_view uri);
  const map<string_view, int>& GetMethodStats() const;
  const map<string_view, int>& GetUriStats() const;
};

HttpRequest ParseRequest(string_view line);
```

Наконец, у вас есть готовая функция ServeRequests:

```
Stats ServeRequests(istream& input) {
  Stats result;
  for (string line; getline(input, line); ) {
    const HttpRequest req = ParseRequest(line);
    result.AddUri(req.uri);
    result.AddMethod(req.method);
  }
  return result;
}
```

Вам нужно, основываясь на реализации функции ServeRequests, реализовать класс Stats и функцию ParseRequest.

Дополнительные требования к классу Stats:

- метод GetMethodStats возвращает словарь, в котором для каждого метода хранится, сколько раз он встретился в качестве аргумента метода AddMethod;
- метод GetUriStats работает аналогично для URI;
- если метод, переданный в метод AddMethod, не поддерживается нашим сервером (список поддерживаемых методов приведён выше), то нужно на единицу увеличить счётчик для метода "UNKNOWN" (подробнее см. юнит-тесты в заготовке решения);
- если URI, переданный в метод AddUri, не поддерживается нашим сервером, то нужно на единицу увеличить счётчик для URI "unknown".

Дополнительные сведения о функции ParseRequest:

- функция ParseRequest разбивает исходный запрос на три соответствующие части (метод, URI, версия протокола), возвращая результат в полях соответствующей структуры HttpRequest;
- при разборе исходного запроса, каждую часть запроса необходимо выделять и сохранять в результирующую структуру без каких-либо дополнительных изменений. Например, для запроса "UNDEFINED /something HTTP/1.1+=" в поля метода, URI и протокола структуры HttpRequest необходимо записать "UNDEFINED", "/something" и "HTTP/1.1+=", соответственно.

На проверку пришлите архив, состоящий из файлов stats.h и stats.cpp (а также любых других файлов, которые вы посчитаете нужным добавить в свой проект). При этом ваши файлы не должны содержать реализацию функции ServeRequests (если ваша посылка будет содержать функцию ServeRequests, вы получите ошибку компиляции).

Заготовка решения

http_request
H File

stats
H File

server_stats
CPP File

Как будет тестироваться ваша посылка

К проекту из вашего архива будет добавлен cpp-файл, который:

- подключает заголовочный файл stats.h;
- содержит точно такую же реализацию функции ServeRequests, какая приведена в условии;
- содержит функцию main с набором юнит-тестов для функции ServeRequests

Ваш проект будет собран и запущен.

Как отправить
Когда работа будет готова, вы можете загрузить файлы для каждой части задания на вкладке 'Мои работы'.