# Задание по программированию: Статистика web-сервера

### 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:

#pragma once

#include "http\_request.h"

#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 соответствующих 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](https://d3c33hcgiwev3.cloudfront.net/w-Fv-nrEEeiqbAqxMNypOA_c43905107ac411e8bcff7d515fb5592b_http_request.h?Expires=1617062400&Signature=LQmoUiixqR~i9rQf0fa1HELVMzwO3wfFNZx1ruS2LgPQkxtx75XEgbsibUjg9OR-vK3B1ld6hkJ38INlhsviis2F6mQUySl9I~jleUsxpCoTXi1fhgg-77Jr9yTPjtj-8ELVvG0nQXZihluxiXAp5bX9hRDjA7RgQgDnl6qhj4c_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

[stats.h](https://d3c33hcgiwev3.cloudfront.net/Bx1E9WTKEei5FgrpHNEYyg_0790c66064ca11e899b2712e81ece485_stats.h?Expires=1617062400&Signature=LqD0xVEV~ZIBZlFUapsmgklzPbEiRyrxNm4zpv-0rVboL9DTi2cv6Rro9rHVm3Nn8E069dN7TrgeJVktdg2Ermu8~Kx6pS8hDD7RERXiBlmeI0AcmQf1QdF38yqBHFzBpGTsJqsfRPJ2K811IruJD2rFvMBjN5o~RMbtqJ01zMY_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

[server\_stats.cpp](https://d3c33hcgiwev3.cloudfront.net/IEsKM2TKEei5FgrpHNEYyg_2074d86064ca11e884860b203c60291a_server_stats.cpp?Expires=1617062400&Signature=UbhpaQYj7fut9PFsIpq6Mh9h2wf46NPaRjz1Yml0iANH2mR8mKSLQT1Rw3fDH84slnewYUfoa~q49t4Yaszhgn9zlJyr5BQ198AhoBcZJLyn8~7YrJeD3uCC3y8DCXjUh~zD98RzjCxDr7570SzvwUKXaCWR14-KN9u9wEKD87A_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

### 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.