Лабораторная работа 1.
HTTP-запросы.

Цель проекта: Цель данного проекта - понять и эффективно применить следующие темы и инструменты:

1. JSON – JavaScript Object Notation
2. Инкапсуляция и сокрытие информации в объектно-ориентированном программировании
3. Использование Casablanca для выполнения HTTP-запросов из C++(либо альтернативу)
4. Использование шаблонов Vector/List из C++ STL (Стандартная библиотека шаблонов)

Описание проекта: В рамках данного проекта поставлена задача получить прогноз погоды на 5 дней с сайта openweathermap.com в формате JSON. Необходимо заполнить список объектов DailyWeatherForecast, а затем предоставить пользователю меню, позволяющее выполнить следующие действия:

1. Просмотр прогноза для введенного пользователем почтового индекса
2. Просмотр прогноза на все 5 дней
3. Просмотр прогноза на конкретный день
4. Отображение самой низкой и самой высокой температур из всех прогнозов
5. Реализация еще как минимум одной креативной опции

Задачи:

1. **Создание типа данных на C++, моделирующего источник(и) данных:**
    - Использование правильной инкапсуляции
    - Использование сокрытия информации
2. **Создание программы на C++ для получения JSON-данных:**
    - Получение данных в формате JSON с HTTP-сайта
    - Заполнение списка(ов) STL или пользовательского типа
3. **Реализация пунктов меню:**
    - Прогноз для введенного пользователем почтового индекса
    - Прогноз на все 5 дней
    - Прогноз на конкретный день
    - Отображение самой низкой и самой высокой температур
    - Реализация как минимум еще одной креативной опции

Примечание: Для начала работы над проектом необходимо скачать пакет Casablanca

Полезные ссылки:

- Руководство по Casablanca HttpClient
- Документация Open Weather Map для текущего API погоды
- API-ключ Open Weather Map
- Пример запроса
- JSON Viewer

Зарегистрируйтесь для получения ключа API.

Это может помочь преобразовать строку на C++ в тип строки, используемый в библиотеке:

```
conversions::to_string_t(string variable);
```

Код для чтения из json-ответа должен выглядеть примерно так (но вы должны сохранять значения в объекте, а объект - в списке).

```
#include <iostream>

#include <cpprest/http_client.h>

#include <cpprest/filestream.h>


using namespace std;

using namespace utility; // Общие утилиты, такие как преобразования строк

using namespace web; // Общие функции, такие как URI using

namespace web::http; // Общая функциональность HTTP using

namespace web::http::client; // Функции клиента HTTP using

namespace concurrency::streams;


void main() {

        http_client client(U("http://api.openweathermap.org"));
        uri_builder builder(U("/data/2.5/forecast"));



        builder.append_query(U("zip"), U("72143"));

        builder.append_query(U("appid"),

        U("5ae2ce314387f8118ec73263ed1ff985")); builder.append_query(U("units"),
```

```
        U("imperial"));

        builder.append_query(U("mode"), U("json"));


        http_response response = client.request(methods::GET,
builder.to_string()).get();

        web::json::value forecastJson =

        response.extract_json(true).get(); web::json::value

        forecastListJson = forecastJson.at(U("list"));


        if (forecastListJson.is_array()) {

        for (int i = 0; i < forecastListJson.size(); i++) {

        web::json::value forecastDayJson = forecastListJson[i];

        web::json::value mainJson = forecastDayJson.at(U("main"));

        web::json::value tempJson = mainJson.at(U("temp"));

        cout << tempJson.as_double() << endl;


        web::json::value weatherJson = forecastDayJson.at(U("weather"))[0];

        web::json::value mainWeatherJson = weatherJson.at(U("main")); cout <<

        conversions::to_utf8string(mainWeatherJson.as_string()) << endl; }

        }

}
```

Срок сдачи: Код и отчеты должны быть загружены на GitHub не позднее 30.01.2024.