Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Программирование

Лабораторная работа №2

Использование внешних библиотек.

Выполнил студент группы № M3109
Гумбатов Владислав Юрьевич
Подпись:

Санкт-Петербург
2021

Задача

Целью лабораторной работы является написание программы, которая раз в 10 секунд получает данные курса с сайта ЦБ РФ и выводит их значения. По окончанию работы выводит среднее и медианное значения.

Для работы была выбрана библиотека C++ REST SDK, под кодовым именем “Casablanca”.

Входные данные:

Входными данными в программе является ссылка на API, где находятся значения курса валют в формате JSON.

Выходные данные:

В качестве выходных данных программа выводит курсы валют каждые 10 секунд, а по окончанию работы среднее и медианное значения.

Код:

```
#include <cpprest/http_client.h>

#include <cpprest/filestream.h>
#include <cpprest/json.h>
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>
#include <vector>
#include <Windows.h>
#include <conio.h>

using namespace utility;
using namespace web;
using namespace web::http;
using namespace web::http::client;
using namespace concurrency::streams;
using namespace web::json;



std::vector<std::vector <double>> vala;



int main(int argc, char* argv[])
{
    std::vector<double> subVec;

    std::vector<utility::string_t> secSubVec;



    bool fl = 1;

    bool anotherFl = 0;

    while (!_kbhit()) {
        try {

            std::cout << "\n";

            http_client client(U("https://www.cbr-xml-daily.ru/latest.js"));

            auto resp = client.request(methods::GET);

            http_response response = resp.get();

            auto a = response.status_code();

            printf("Received response status code:%u\n", a);
```

```
try {
    auto js = response.extract_json().get();

    auto rates = js.at(U("rates"));

    std::wofstream adas("result.txt");

    utility::ostream_t  &dasd = adas;

    js.serialize(dasd);

    auto subArr = rates.as_object();


    auto size = subArr.size();

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

        vala.push_back(subVec);



    }

    int cnt = 0;

    for (auto iter = subArr.begin(); iter != subArr.end(); iter++) {

        auto currencyName = iter->first;

        auto currencyValue = iter->second;

        double temp = 1 / currencyValue.as_double();

        if (anotherFl == 0) {

            secSubVec.push_back(currencyName);

        }


        std::wcout << "" << currencyName << " " << 1 / currencyValue.as_double() << std::endl;

        vala[cnt].push_back(temp);
        cnt++;
    }

    fl = 0;

    try {

        std::wofstream adas("result.txt");
```

```
            try {

                std::wofstream adas("result.txt");

                utility::ostream_t  &dasd = adas;

                js.serialize(dasd);
            }
            catch (const std::exception &e)
            {
                printf("Error exception:%s\n", e.what());
            }
        }

        catch (const std::exception &e)
        {
            printf("Error exception:%s\n", e.what());
        }


        try {

            auto localFile = file_buffer<uint8_t>::open(U("results.txt")).get();

            auto respSec = client.request(methods::GET);

            http_response responseSec = respSec.get();

            responseSec.body().read_to_end(localFile).get();

        }

        catch (const std::exception &e)
        {
            printf("Error exception:%s\n", e.what());
        }



    }

    catch (const std::exception &e)
    {
        printf("Error exception:%s\n", e.what());
    }

    anotherFl = 1;

    Sleep(10000);

}
```

```
    std::cout << std::endl;

    for (int i = 0; i < vala.size(); i++) {
        sort(vala[i].begin(), vala[i].end());
        double sum = 0;
        bool flafForPrint = 0;
        for (int j = 0; j < vala[i].size(); j++) {
            sum = sum + vala[i][j];
        }
            if (flafForPrint == 0) {
                if (vala[i].size() % 2 != 0) {
                    auto temp = vala[i][((vala[i].size() / 2) + 0.5) - 1];
                    std::wcout << secSubVec[i] << "     " << " Average " << "     " << sum / vala[i].size() << "     " << " Median " << "     " << temp
                }
                else {
                    auto taemp = (vala[i][((vala[i].size() / 2)) - 1] + vala[i][((vala[i].size() / 2))]) / 2;
                    std::wcout << secSubVec[i] << "     " << " Average " << "     " << sum / vala[i].size() << "     " << " Median " << "     " << taemp
                }

                //std::wcout << secSubVec[i] << " Average " << sum / vala[i].size() << std::endl;
            }
        }


}
```

Вывод:

Программа успешно работает. Курсы валют получаются с помощью C++ REST SDK “Casablanca”, а затем они выводятся на экран и обновляются каждые 10 секунд. Цель лабораторной работы достигнута.