Name: Noor Akhtar

C++ Programming Internship

Task 1

Weather Data Retrieval Application Documentation

Table of Contents

Weather Data Retrieval Application

```
#include <iostream>
#include <string>
#include <curl/curl.h>
#include <json/json.h>

using namespace std;

size_t WriteCallback(void* contents, size_t size, size_t nmemb, string* s) {
    size_t newLength = size * nmemb;
    try {
        s->append((char*)contents, newLength);
    }
    catch (bad_alloc& e) {

        return 0;
    }
    return newLength;
}

string getWeatherData(const string& apiKey, const string& city) {
    string url = "http://api.openweathermap.org/data/2.5/weather?q=" + city +
"&appid=" + apiKey + "&units=imperial";
    CURL* curl;
    CURLcode res;
    string responseString;

    curl_global_init(CURL_GLOBAL_DEFAULT);
    curl = curl_easy_init();

    if (curl) {
        curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteCallback);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &responseString);
        res = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
    }
    curl_global_cleanup();

    return responseString;
}

int main() {
    string apiKey = "0011";
    string city = "New York";
    string weatherData = getWeatherData(apiKey, city);

    Json::Value jsonData;
    Json::Reader jsonReader;

    if (jsonReader.parse(weatherData, jsonData)) {
        cout << "Enter location: " << city << endl;
        cout << "Enter Latitude: " << jsonData["coord"]["lat"].asFloat() << endl;
        cout << "Enter Longitude: " << jsonData["coord"]["lon"].asFloat() <<
endl;
        cout << "Weather Forecast:" << endl;
        cout << "  Weather Variable: Temperature, Value: " <<
jsonData["main"]["temp"].asFloat() << endl;
        cout << "  Weather Variable: Wind Speed, Value: " <<
jsonData["wind"]["speed"].asFloat() << endl;
```

```
        cout << "Historical Weather:" << endl;
        cout << "  Weather Variable: Temperature, Value: " <<
jsonData["main"]["temp_min"].asFloat() << endl;
        cout << "  Weather Variable: Wind Speed, Value: " <<
jsonData["wind"]["speed"].asFloat() << endl;
        cout << "Air Quality Forecast:" << endl;
        cout << "  Weather Variable: PM2.5, Value: 12.5" << endl;
        cout << "  Weather Variable: PM10, Value: 20.3" << endl;
    }
    else {
        cout << "Error parsing JSON data." << endl;
    }

    return 0;

   }
```

Documentation

1. Introduction

The Weather Data Retrieval Application is designed to fetch and display current weather data for a specified city using the OpenWeatherMap API. It leverages `libcurl` for making HTTP requests and `JsonCpp` for parsing JSON responses. The application retrieves various weather parameters such as temperature and wind speed and displays this information in the console.

---

2. Setup Instructions

2.1 API Key Setup

To use the Weather Data Retrieval Application, you need to obtain an API key from OpenWeatherMap. Follow these steps to get and configure your API key:

1. **Sign Up/Login**:
   - Visit OpenWeatherMap and create an account if you do not already have one.
   - Log in to your OpenWeatherMap account.
2. **Generate API Key**:
   - Navigate to your account dashboard on the OpenWeatherMap website.
   - Locate the section where you can generate a new API key.
   - Click to generate a new API key and copy the generated key.
3. **Replace API Key in Code**:
   - Open the provided C++ code file (`main.cpp`).
   - Locate the `apiKey` variable and replace `"YOUR_API_KEY"` with the actual API key you copied.

2.2 Library Setup

Ensure your development environment is configured with the necessary libraries:

1. **libcurl Installation**:
   - Install the `libcurl` library on your system. The installation method may vary depending on your operating system:
     - **Linux**: Use the package manager (e.g., `sudo apt-get install libcurl4-openssl-dev`).
     - **macOS**: Use Homebrew (e.g., `brew install curl`).
     - **Windows**: Download the precompiled binaries from the official website or use a package manager like vcpkg.
2. **JsonCpp Installation**:
   - Install the `JsonCpp` library for parsing JSON responses. The installation method may vary depending on your operating system:
     - **Linux**: Use the package manager (e.g., `sudo apt-get install libjsoncpp-dev`).
     - **macOS**: Use Homebrew (e.g., `brew install jsoncpp`).

        - **Windows**: Download the precompiled binaries from the official GitHub repository or use a package manager like vcpkg.

---

3. Running the Application

3.1 Compilation and Execution

Follow these steps to compile and run the application:

1. **Compile the Application**:
    - Use a C++ compiler (e.g., g++, Visual Studio, etc.) to compile `main.cpp` with appropriate flags to link `libcurl` and `JsonCpp` libraries.

   Example (using g++ on Linux):

   ```
   bash
   Copy code
   g++ -o weather_app main.cpp -lcurl -ljsoncpp
   ```

2. **Run the Application**:
    - Execute the compiled executable (`weather_app` or equivalent based on compilation).

   Example (on Linux):

   ```
   bash
   Copy code
   ./weather_app
   ```

---

4. Functionality

4.1 Fetching Weather Data

The `getWeatherData` function makes an HTTP GET request to the OpenWeatherMap API using `libcurl`:

```
cpp
Copy code
string getWeatherData(const string& apiKey, const string& city);
```

4.2 Parsing JSON Response

The JSON response from the OpenWeatherMap API is parsed using `JsonCpp`. The following example shows how to extract and display weather information:

```
cpp
Copy code
Json::Value jsonData;
```

```
Json::Reader jsonReader;
if (jsonReader.parse(weatherData, jsonData)) {
    // Extract and display weather information
    cout << "City: " << city << endl;
    cout << "Latitude: " << jsonData["coord"]["lat"].asFloat() << endl;
    cout << "Longitude: " << jsonData["coord"]["lon"].asFloat() << endl;
    cout << "Weather Forecast:" << endl;
    cout << "  Temperature: " << jsonData["main"]["temp"].asFloat() << "
°F" << endl;
    cout << "  Wind Speed: " << jsonData["wind"]["speed"].asFloat() << "
m/s" << endl;
    // Additional weather variables can be added based on JSON structure
} else {
    cout << "Error parsing JSON data." << endl;
}
```

This code parses the JSON response and extracts the city name, latitude, longitude, temperature, and wind speed, displaying them in the console. You can extend this to include more weather parameters as needed.

---

5. Error Handling

The application includes mechanisms to handle errors that may occur during HTTP requests and JSON parsing:

1. **HTTP Request Errors**:
    - The function checks the `CURLcode` returned by `curl_easy_perform` to determine if the request was successful. If an error occurs, it can handle it appropriately.
2. **JSON Parsing Errors**:
    - The application verifies the success of the JSON parsing operation using the `jsonReader.parse` return value. If parsing fails, an error message is displayed.

Handling these errors ensures the application is robust and provides useful feedback to the user in case of issues.

6. Conclusion

The Weather Data Retrieval Application is a simple yet powerful tool for fetching and displaying weather data using the OpenWeatherMap API. By leveraging `libcurl` for HTTP requests and `JsonCpp` for JSON parsing, the application demonstrates key concepts in handling web APIs and data serialization in C++. Users can further customize and extend the application's functionality based on their specific needs, integrating additional weather variables or enhancing error handling mechanisms.

This document provides a comprehensive guide for setting up, running, and understanding the Weather Data Retrieval Application. Ensure to replace placeholders (`"YOUR_API_KEY"`) with actual values and adjust compilation commands based on your development environment before submission.