Digital Empowerment Pakistan

Name: Muhammad Ahsan Ali

C++ Programming Internship

Task 1

Weather Data Retrieval Application

Contents

Weather Data Retrieval Application Documentation

Overview

The Weather Data Retrieval Application is a C++ program designed to fetch and display current weather data for a specified city using the OpenWeatherMap API. The program utilizes the cURL library for HTTP requests and the JSONcpp library for parsing JSON responses.

Features

- **Fetch Weather Data**: Retrieves current weather information for a given city.
- **Parse JSON**: Processes the JSON response from the OpenWeatherMap API.
- **Display Information**: Outputs weather data such as temperature, wind speed, and coordinates to the console.

Dependencies

- **cURL**: A free and easy-to-use client-side URL transfer library that supports multiple protocols. It is used for making HTTP requests.
- **JSONcpp**: A C++ library for parsing and manipulating JSON data.

Installation

cURL Installation

Ensure that cURL is installed on your system. Installation commands vary by operating system:

- **Ubuntu/Debian**: `sudo apt-get install libcurl4-openssl-dev`
- **Red Hat/CentOS**: `sudo yum install libcurl-devel`

JSONcpp Installation

You can install JSONcpp using package managers or build it from source. For Ubuntu/Debian:

```
bash
Copy code
sudo apt-get install libjsoncpp-dev
```

Compilation

To compile the program, use the following command:

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

If cURL or JSONcpp headers are located in non-standard directories, include their paths as follows:

```
bash
Copy code
g++ -o weatherApp main.cpp -I/path/to/curl/include -
I/path/to/jsoncpp/include -L/path/to/curl/lib -L/path/to/jsoncpp/lib -lcurl
-ljsoncpp
```

Code Description

Header Inclusions

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

- **`<iostream>`**: Provides input and output stream functionality.
- **`<string>`**: Includes the `std::string` class for string manipulation.
- **`<curl/curl.h>`**: Includes cURL functions for HTTP requests.
- **`<json/json.h>`**: Includes JSONcpp classes for JSON parsing.

WriteCallback Function

```
cpp
Copy code
size_t WriteCallback(void* contents, size_t size, size_t nmemb,
std::string* s) {
    size_t newLength = size * nmemb;
    try {
        s->append((char*)contents, newLength);
    }
    catch (std::bad_alloc& e) {
        return 0;
    }
    return newLength;
}
```

- **Purpose**: Appends data received from cURL to a `std::string`.
- **Parameters**: `contents` is the data buffer, `size` and `nmemb` are the size and number of elements in the buffer, and `s` is the string to append data to.

getWeatherData Function

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

```
    CURLcode res;
    std::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;
}
```

- **Purpose**: Constructs a URL with the API key and city, performs an HTTP GET request, and returns the response as a string.
- **Parameters**: `apiKey` is the OpenWeatherMap API key, and `city` is the name of the city.
- **Returns**: The response string containing weather data in JSON format.

main Function

```
cpp
Copy code
int main() {
    std::string apiKey = "0011";
    std::string city = "New York";
    std::string weatherData = getWeatherData(apiKey, city);
    Json::Value jsonData;
    Json::Reader jsonReader;
    if (jsonReader.parse(weatherData, jsonData)) {
        std::cout << "Location: " << city << std::endl;
        std::cout << "Latitude: " << jsonData["coord"]["lat"].asFloat() <<
std::endl;
        std::cout << "Longitude: " << jsonData["coord"]["lon"].asFloat() <<
std::endl;
        std::cout << "Weather Forecast:" << std::endl;
        std::cout << "Temperature: " << jsonData["main"]["temp"].asFloat()
<< std::endl;
        std::cout << "Wind Speed: " << jsonData["wind"]["speed"].asFloat()
<< std::endl;
        std::cout << "Historical Weather:" << std::endl;
        std::cout << "Temperature (Min): " <<
jsonData["main"]["temp_min"].asFloat() << std::endl;
        std::cout << "Wind Speed: " << jsonData["wind"]["speed"].asFloat()
<< std::endl;
        std::cout << "Air Quality Forecast:" << std::endl;
        std::cout << "PM2.5: 12.5" << std::endl;
        std::cout << "PM10: 20.3" << std::endl;
    }
    else {
        std::cout << "Error parsing JSON data." << std::endl;
    }
    return 0;
}
```

- **Purpose**: Initializes the API key and city, retrieves weather data, parses the JSON response, and displays relevant weather information.

Usage

1. **Compile the Code**: Follow the compilation instructions provided above.
2. **Run the Executable**: Execute the compiled binary. The program will fetch and display weather data for New York City.

```
bash
Copy code
./weatherApp
```

3. **API Key**: Replace `"0011"` with your actual OpenWeatherMap API key for real data.

Error Handling

- **cURL Errors**: If the cURL functions fail, appropriate error messages are printed to the standard error.
- **JSON Parsing Errors**: If the JSON parsing fails, an error message is printed to the standard output.

Future Enhancements

- **Dynamic API Key**: Allow users to input or configure the API key dynamically.
- **Error Handling Improvement**: Enhance error handling for HTTP and JSON parsing errors.
- **Extended Weather Information**: Integrate additional weather parameters and more detailed data