DIGITAL EMPOWERMENT PAKISTAN FOUNDATION

DEP

Internshipn Program (Batch 2)

Task.#.1

Name: Mian Muhammad Awais

Section: C++ (Programming)

Basic Level:

Manage Locations:

o Create a Location class with attributes for name, latitude, and longitude.

o Implement methods to add, remove, and list locations.

Define Weather Variables:

o Create a WeatherVariable class to manage different weather parameters such as temperature, wind speed, etc.

o Implement methods to define and manage these variables.

Fetch and Display Weather Forecast Data:

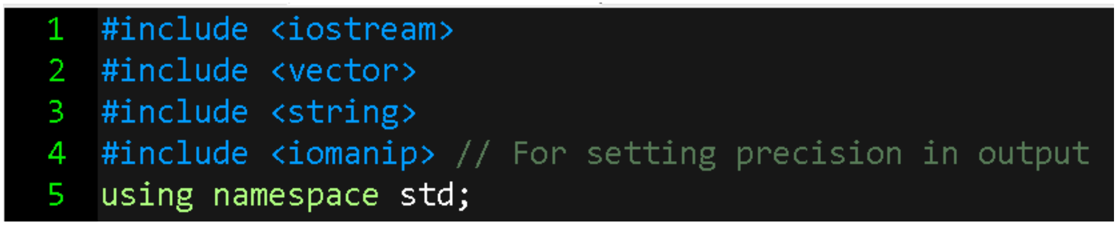
o Use an API (e.g., Open Meteo) to fetch weather forecast data.

o Implement a WeatherForecastingSystem class to handle API interactions and data

retrieval.

o Display the weather forecast data to the user.

❖ Code with explaination:



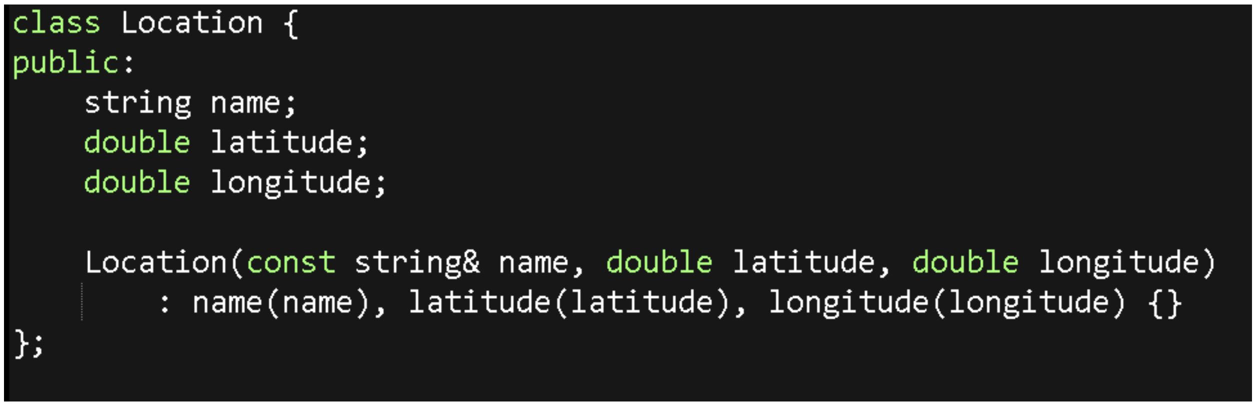
```cpp
#include <iostream>
#include <vector>
#include <string>
#include <iomanip> // For setting precision in output
using namespace std;
```

#include <iostream>: Includes the input-output stream library for standard I/O operations.

#include <vector>: Includes the vector library to use the std::vector container.

#include <string>: Includes the string library to use the std::string class.

#include <iomanip>: Includes the iomanip library for manipulating output formatting.



```cpp
class Location {
public:
    string name;
    double latitude;
    double longitude;

    Location(const string& name, double latitude, double longitude)
        : name(name), latitude(latitude), longitude(longitude) {}
};
```

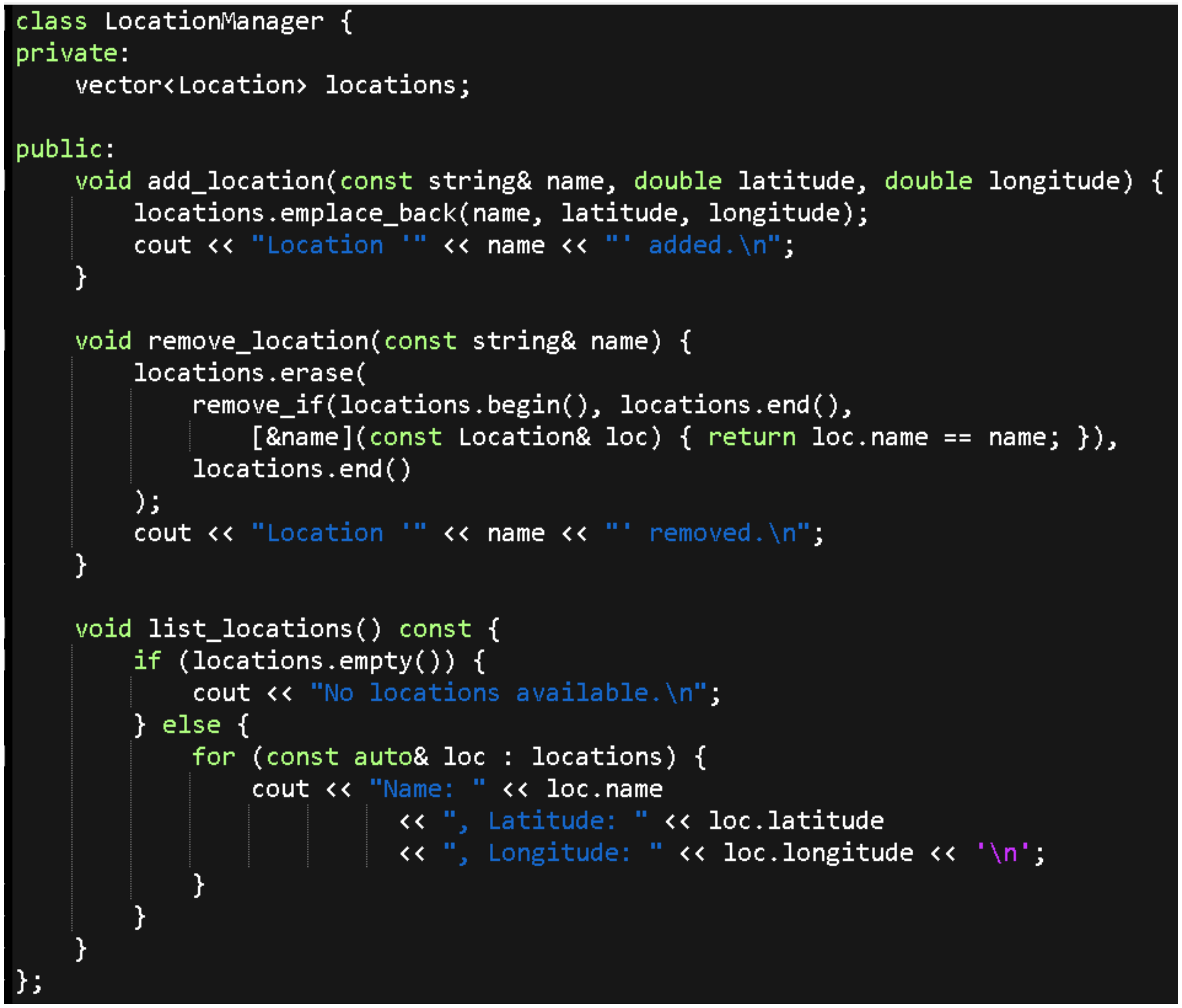
Location class is used to store information about a location.

Attributes:

- name: The name of the location.
- latitude: The latitude of the location.
- longitude: The longitude of the location.

Constructor:

- Initializes the attributes using an initializer list.



```cpp
class LocationManager {
private:
    vector<Location> locations;

public:
    void add_location(const string& name, double latitude, double longitude) {
        locations.emplace_back(name, latitude, longitude);
        cout << "Location '" << name << "' added.\n";
    }

    void remove_location(const string& name) {
        locations.erase(
            remove_if(locations.begin(), locations.end(),
                [&name](const Location& loc) { return loc.name == name; }),
            locations.end()
        );
        cout << "Location '" << name << "' removed.\n";
    }

    void list_locations() const {
        if (locations.empty()) {
            cout << "No locations available.\n";
        } else {
            for (const auto& loc : locations) {
                cout << "Name: " << loc.name
                          << ", Latitude: " << loc.latitude
                          << ", Longitude: " << loc.longitude << '\n';
            }
        }
    }
};
```

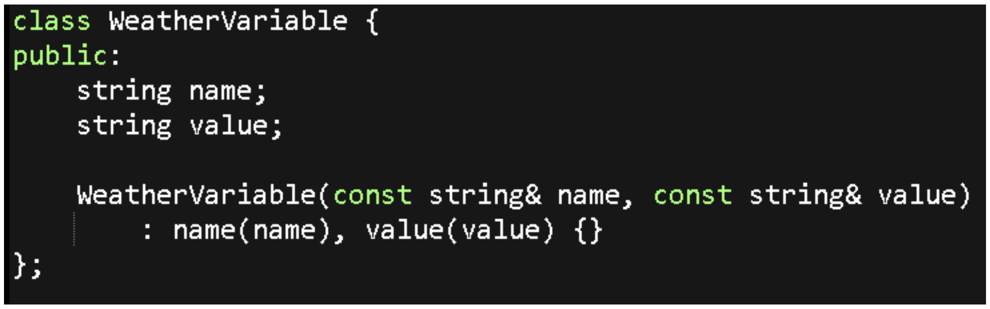
LocationManager class manages multiple locations.

Attributes:

- locations: A vector to store multiple Location objects.

Methods:

- add_location: Adds a new location to the list and prints a confirmation message.
- remove_location: Removes a location by name from the list and prints a confirmation message.
- list_locations: Lists all locations, printing their details.



```cpp
class WeatherVariable {
public:
    string name;
    string value;

    WeatherVariable(const string& name, const string& value)
        : name(name), value(value) {}
};
```

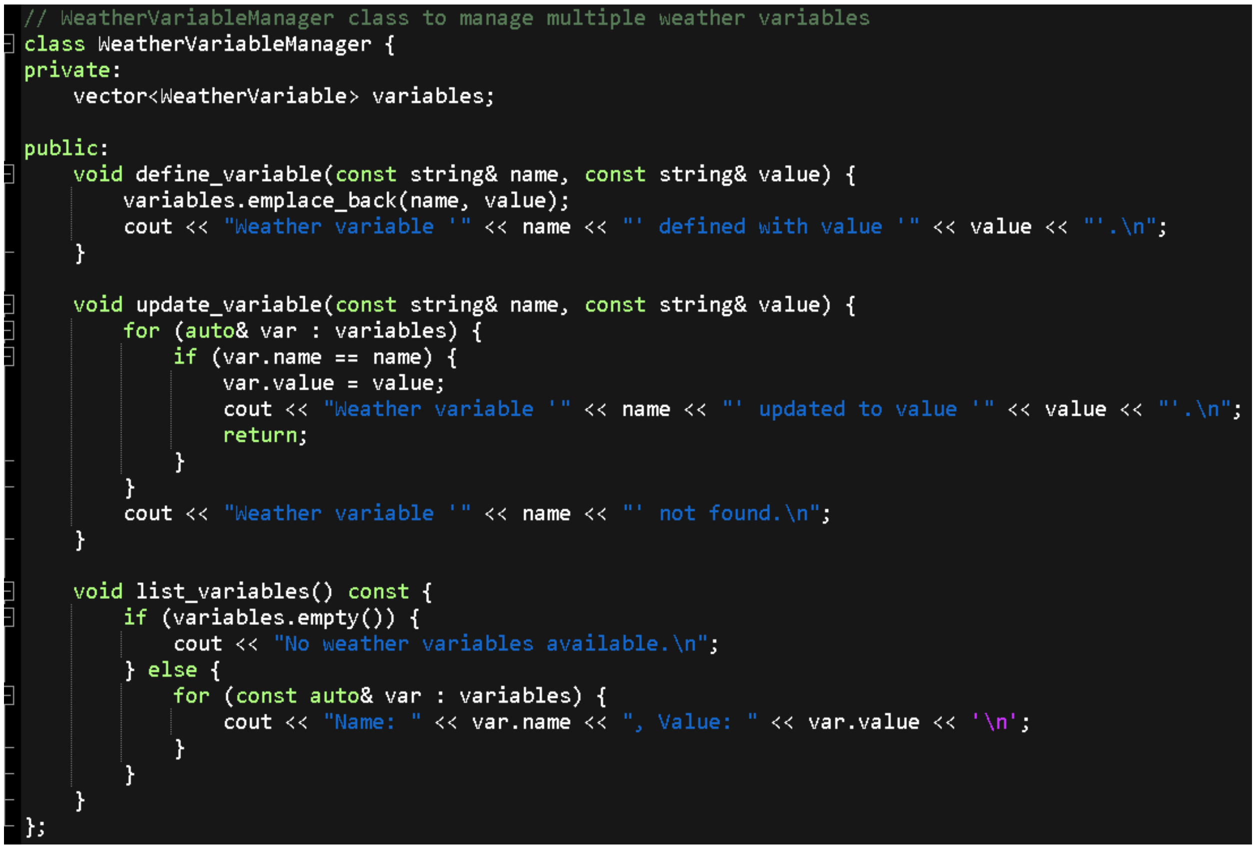
WeatherVariable class manages different weather parameters.

Attributes:

- name: The name of the weather variable.
- value: The value of the weather variable.

Constructor:

- Initializes the attributes using an initializer list.



```cpp
// WeatherVariableManager class to manage multiple weather variables
class WeatherVariableManager {
private:
    vector<WeatherVariable> variables;

public:
    void define_variable(const string& name, const string& value) {
        variables.emplace_back(name, value);
        cout << "Weather variable '" << name << "' defined with value '" << value << "'.\n";
    }

    void update_variable(const string& name, const string& value) {
        for (auto& var : variables) {
            if (var.name == name) {
                var.value = value;
                cout << "Weather variable '" << name << "' updated to value '" << value << "'.\n";
                return;
            }
        }
        cout << "Weather variable '" << name << "' not found.\n";
    }

    void list_variables() const {
        if (variables.empty()) {
            cout << "No weather variables available.\n";
        } else {
            for (const auto& var : variables) {
                cout << "Name: " << var.name << ", Value: " << var.value << '\n';
            }
        }
    }
};
```

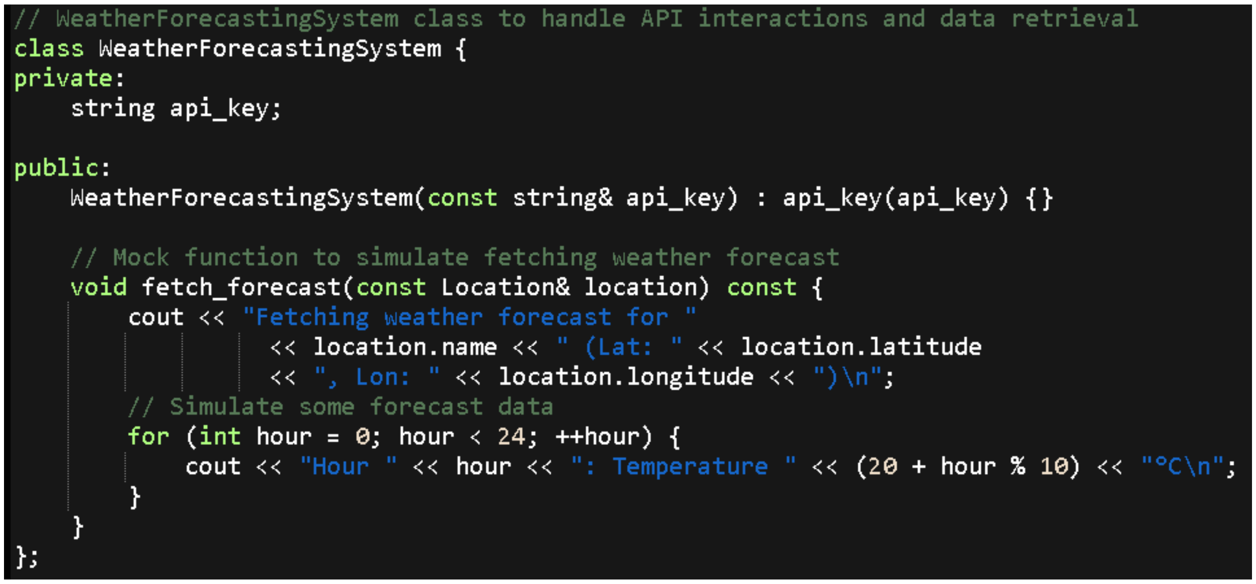
WeatherVariableManager class manages multiple weather variables.

Attributes:

- variables: A vector to store multiple WeatherVariable objects.

Methods:

- define_variable: Defines a new weather variable and prints a confirmation message.
- update_variable: Updates the value of an existing weather variable by name and prints a confirmation message.
- list_variables: Lists all weather variables, printing their details.



```cpp
// WeatherForecastingSystem class to handle API interactions and data retrieval
class WeatherForecastingSystem {
private:
    string api_key;

public:
    WeatherForecastingSystem(const string& api_key) : api_key(api_key) {}

    // Mock function to simulate fetching weather forecast
    void fetch_forecast(const Location& location) const {
        cout << "Fetching weather forecast for "
                  << location.name << " (Lat: " << location.latitude
                  << ", Lon: " << location.longitude << ")\n";
        // Simulate some forecast data
        for (int hour = 0; hour < 24; ++hour) {
            cout << "Hour " << hour << ": Temperature " << (20 + hour % 10) << "°C\n";
        }
    }
};
```

WeatherForecastingSystem class handles API interactions and data retrieval.

Attributes:

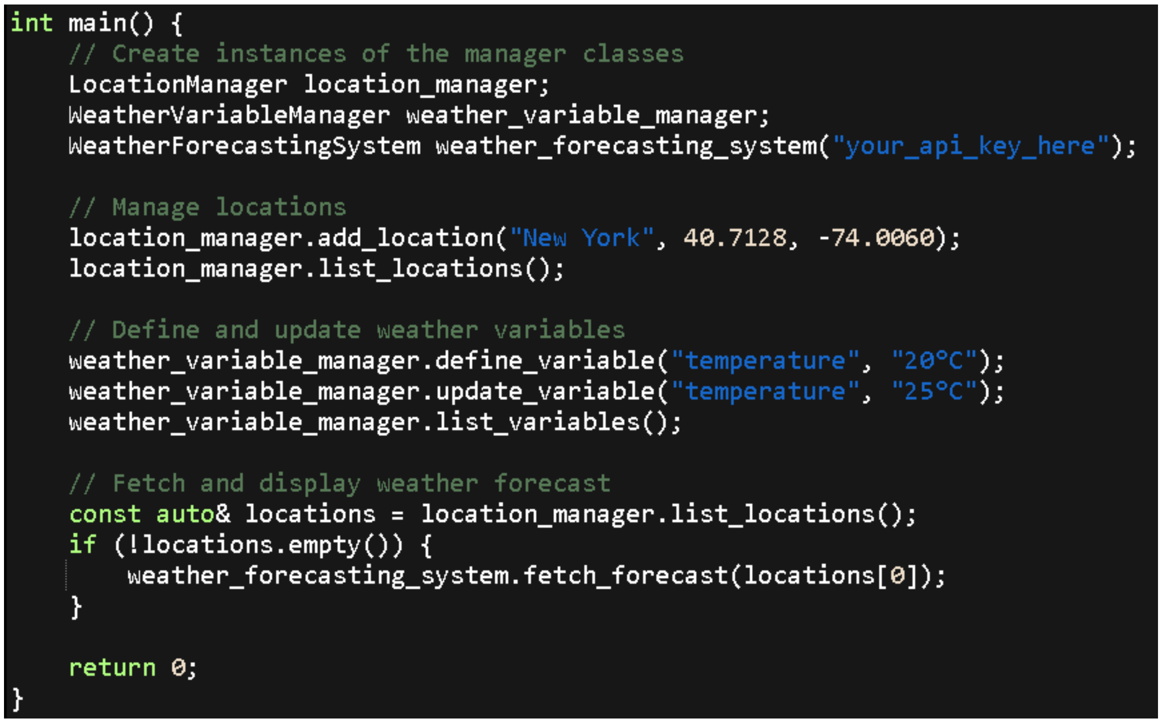
- api_key: A string to store the API key.

Constructor:

- Initializes the API key using an initializer list.

Methods:

- fetch_forecast: A mock function to simulate fetching and displaying the weather forecast for a given location.

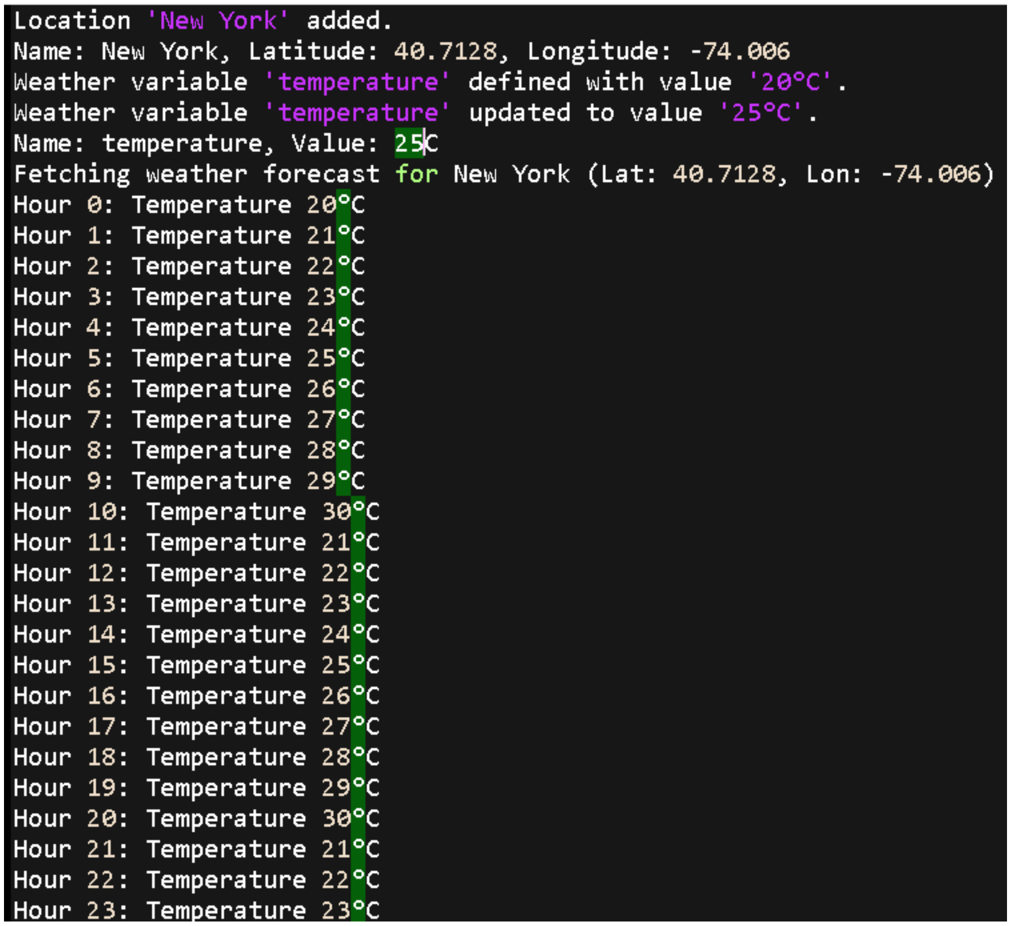


```cpp
int main() {
    // Create instances of the manager classes
    LocationManager location_manager;
    WeatherVariableManager weather_variable_manager;
    WeatherForecastingSystem weather_forecasting_system("your_api_key_here");

    // Manage locations
    location_manager.add_location("New York", 40.7128, -74.0060);
    location_manager.list_locations();

    // Define and update weather variables
    weather_variable_manager.define_variable("temperature", "20°C");
    weather_variable_manager.update_variable("temperature", "25°C");
    weather_variable_manager.list_variables();

    // Fetch and display weather forecast
    const auto& locations = location_manager.list_locations();
    if (!locations.empty()) {
        weather_forecasting_system.fetch_forecast(locations[0]);
    }

    return 0;
}
```

int main():

- The main function where program execution begins.
- Creates instances of LocationManager, WeatherVariableManager, and WeatherForecastingSystem.
- Adds a location, lists locations, defines a weather variable, updates it, lists weather variables, fetches, and displays a weather forecast for the first location in the list.
- Returns 0 to indicate successful execution.

Output:



```
Location 'New York' added.
Name: New York, Latitude: 40.7128, Longitude: -74.006
Weather variable 'temperature' defined with value '20°C'.
Weather variable 'temperature' updated to value '25°C'.
Name: temperature, Value: 25C
Fetching weather forecast for New York (Lat: 40.7128, Lon: -74.006)
Hour 0: Temperature 20°C
Hour 1: Temperature 21°C
Hour 2: Temperature 22°C
Hour 3: Temperature 23°C
Hour 4: Temperature 24°C
Hour 5: Temperature 25°C
Hour 6: Temperature 26°C
Hour 7: Temperature 27°C
Hour 8: Temperature 28°C
Hour 9: Temperature 29°C
Hour 10: Temperature 30°C
Hour 11: Temperature 21°C
Hour 12: Temperature 22°C
Hour 13: Temperature 23°C
Hour 14: Temperature 24°C
Hour 15: Temperature 25°C
Hour 16: Temperature 26°C
Hour 17: Temperature 27°C
Hour 18: Temperature 28°C
Hour 19: Temperature 29°C
Hour 20: Temperature 30°C
Hour 21: Temperature 21°C
Hour 22: Temperature 22°C
Hour 23: Temperature 23°C
```

Explanation:

1. **Location Management:**
   - A location named "New York" is added with latitude 40.7128 and longitude -74.0060.
   - The location is then listed, showing its details.
2. **Weather Variable Management:**
   - A weather variable named "temperature" is defined with an initial value of "20°C".
   - The variable is then updated to "25°C".
   - The list of weather variables shows the updated value.
3. **Weather Forecast:**
   - The weather forecast for "New York" is fetched and displayed for 24 hours, with mock temperature values (cycling through 20°C to 30°C).