**NAME: ZUFRA JAHAN**

**INTERNSHIP TASK 1**

**CODE:**

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

#include <fstream>

using namespace std;

class Location {

private:

string name;

double longitude;

double latitude;

public:

Location() : name("unknown"), longitude(0.0), latitude(0.0) {}

Location(string NM, double LG, double LT) : name(NM), longitude(LG), latitude(LT) {}

void addLocation(string NM, double LG, double LT) {

name = NM;

longitude = LG;

latitude = LT;

}

void listLocation() const {

cout << "Name of location: " << name << endl;

cout << "Longitude of location: " << longitude << endl;

cout << "Latitude of location: " << latitude << endl;

}

string getName() const { return name; }

double getLongitude() const { return longitude; }

double getLatitude() const { return latitude; }

};

class WeatherVariable {

private:

float temperature;

float windSpeed;

public:

WeatherVariable() : temperature(0.0f), windSpeed(0.0f) {}

WeatherVariable(float TP, float WP) : temperature(TP), windSpeed(WP) {}

void addVariables(float TP, float WP) {

temperature = TP;

windSpeed = WP;

}

void displayWeatherVariable() const {

cout << "Temperature: " << temperature << " °C" << endl;

cout << "Wind Speed: " << windSpeed << " m/s" << endl;

}

};

class WeatherForecastingSystem {

public:

static string fetchWeatherData(const Location& loc) {

return "Weather Data for " + loc.getName() + ": 25°C, Wind Speed: 5 m/s";

}

static void displayWeatherData(const Location& loc) {

string weatherData = fetchWeatherData(loc);

cout << weatherData << endl;

}

};

class HistoricalWeatherSystem {

public:

static string fetchHistoricalData(const Location& loc) {

return "Historical Weather Data for " + loc.getName() + ": 20°C, Wind Speed: 3 m/s";

}

static void displayHistoricalData(const Location& loc) {

string historicalData = fetchHistoricalData(loc);

cout << historicalData << endl;

}

};

class AirQualityForecastingSystem {

public:

static string fetchAirQualityData(const Location& loc) {

return "Air Quality Data for " + loc.getName() + ": AQI 42 (Good)";

}

static void displayAirQualityData(const Location& loc) {

string airQualityData = fetchAirQualityData(loc);

cout << airQualityData << endl;

}

};

void exportToCSV(const vector<Location>& locations, const string& filename) {

ofstream file(filename);

if (file.is\_open()) {

file << "Name,Longitude,Latitude\n";

for (const auto& loc : locations) {

file << loc.getName() << "," << loc.getLongitude() << "," << loc.getLatitude() << "\n";

}

file.close();

cout << "Data exported to " << filename << endl;

}

else {

cout << "Unable to open file " << filename << endl;

}

}

int main() {

WeatherVariable wv;

vector<Location> locations;

wv.addVariables(12.6f, 200.0f);

wv.displayWeatherVariable();

wv.addVariables(12.3f, 100.0f);

wv.displayWeatherVariable();

locations.emplace\_back("Paris", 2.3522, 48.8566);

locations.emplace\_back("London", -0.1276, 51.5074);

for (const auto& loc : locations) {

loc.listLocation();

}

string removeName;

cout << "Enter name of location to remove: ";

cin >> removeName;

auto it = remove\_if(locations.begin(), locations.end(),

[&removeName](const Location& loc) { return loc.getName() == removeName; });

if (it != locations.end()) {

locations.erase(it, locations.end());

cout << "Location " << removeName << " successfully deleted." << endl;

}

else {

cout << "Location " << removeName << " not found." << endl;

}

if (!locations.empty()) {

WeatherForecastingSystem::displayWeatherData(locations.front());

HistoricalWeatherSystem::displayHistoricalData(locations.front());

AirQualityForecastingSystem::displayAirQualityData(locations.front());

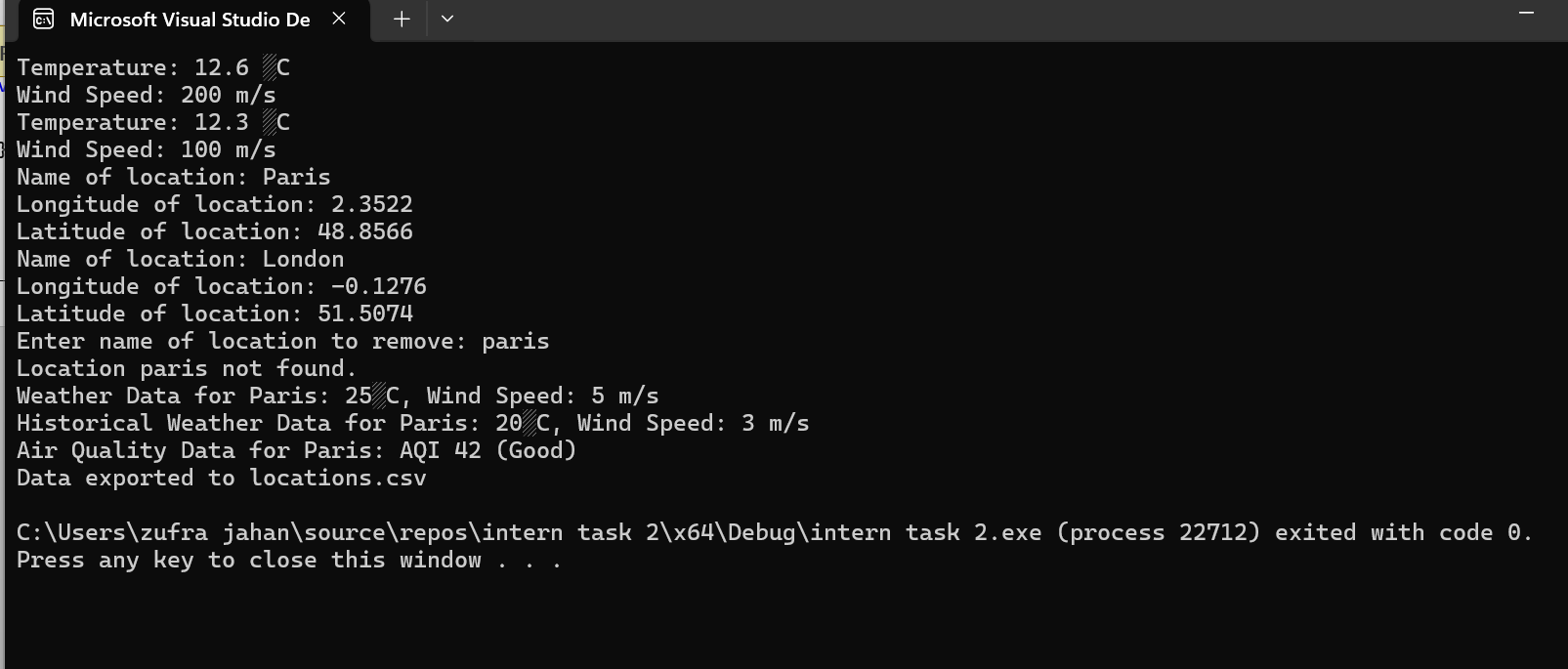
}

exportToCSV(locations, "locations.csv");

return 0;

}

**OUTPUT:**

****