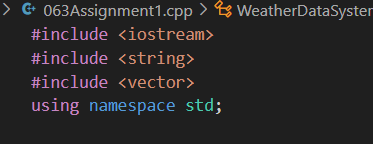
**Assignment-1**

**Weather Data Storage System**

**Includes and Namespace**



**Explanation:**

**#include <iostream>**: For input (cin) and output (cout).

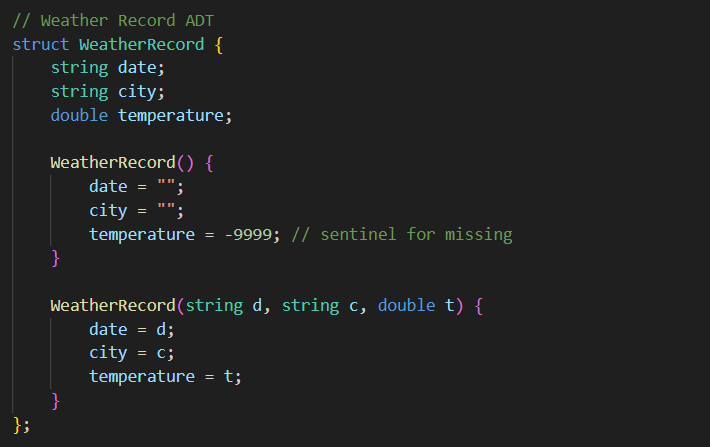
**#include <string>**: For using string data type.

**#include <vector>**: For dynamic arrays (2D arrays are implemented as vectors of vectors).

**using namespace std;** : So, we don’t have to type std:: every time.

**Outcome:** Allows the program to use standard input/output, strings, and vectors easily.

**WeatherRecord ADT**



**Explanation:**

**Purpose:** Stores a single weather record (ADT = Abstract Data Type).

**date →** Stores date of the record

**city →** Stores city name.

**temperature →** Stores temperature value.

**Constructors:**

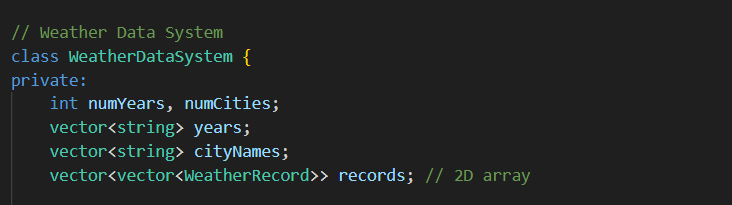
**Default →** Initializes empty date/city and -9999 for temperature (indicates no data).

**Parameterized →** Initializes a record with given date, city, and temperature.

**Outcome:** You can create objects like WeatherRecord w; or WeatherRecord w("01/01/2023", "Delhi", 25.5);.

**Functionality:** Represents each individual weather data entry.

**WeatherDataSystem Class**



**Explanation:**

**numYears, numCities →** Stores number of years and cities.

**years →** Vector storing all year strings.

**cityNames →** Vector storing all city names.

**records →** 2D array (vector of vectors) storing WeatherRecord objects for each year-city combination.

**Outcome:** Sets up the structure to store and manage multiple weather records.

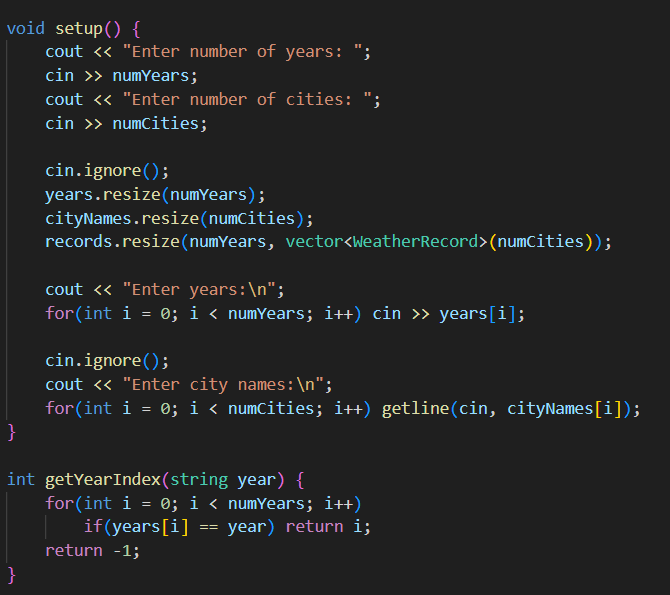
**Constructor**



Explanation: Initializes years and cities to 0.

Outcome: Ensures safe default initialization before user inputs.

**Setup Function**



**Explanation:**

Takes input from the user: number of years and cities.

Resizes vectors to hold records.

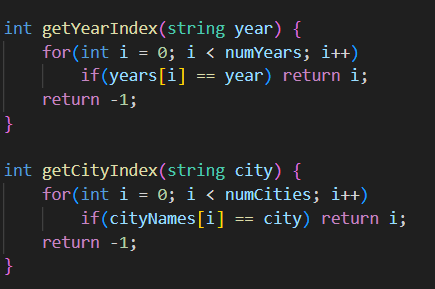
Inputs years and city names.

Initializes the records 2D array.

**Outcome:** Prepares the system to store weather data.

**Functionality:** Initializes the grid (2D array) to hold weather records.

**Helper Functions**



Explanation:

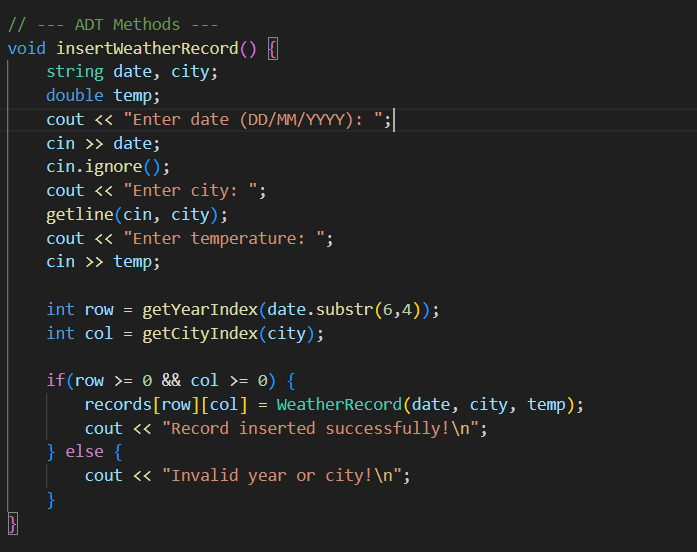
Maps a year or city name to the corresponding row/column index in the 2D array.

Returns -1 if the year/city is invalid.

Outcome: Lets program know where to insert, delete, or retrieve records in the 2D array.

ADT Operations

**Insert Weather Record**



**Explanation:**

Inputs date, city, temperature.

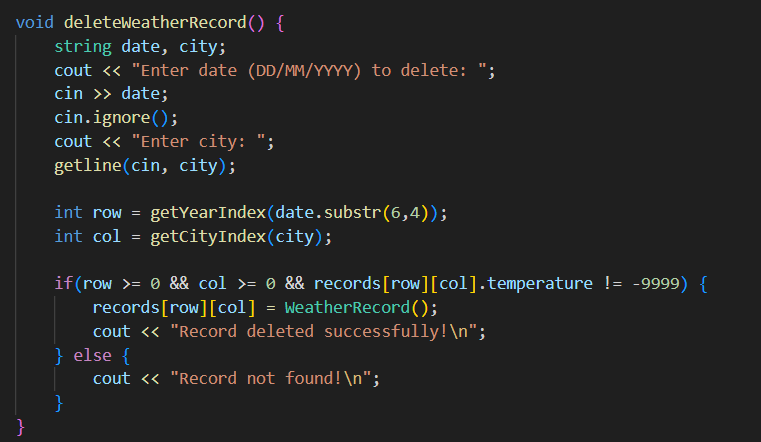
Finds row = year index, col = city index.

Inserts record in 2D array if valid.

**Outcome:** Successfully adds a weather record.

**Functionality:** Uses ADT object to store structured data.

**Delete Weather Record**



**Explanation:**

Inputs date, city, temperature.

Finds row = year index, col = city index.

Inserts record in 2D array if valid.

**Outcome:** Successfully adds a weather record.

**Functionality:** Uses ADT object to store structured data.

**Retrieve Weather Record**



**Explanation:**

Inputs date, city, temperature.

Finds row = year index, col = city index.

Inserts record in 2D array if valid.

**Outcome:** Successfully adds a weather record.

**Functionality:** Uses ADT object to store structured data.

2D Array Specific Functions

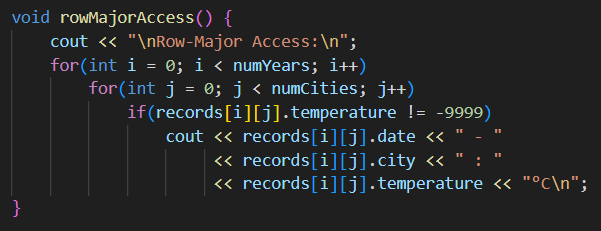
**Insert Temperature**



**Explanation:** Similar to insertWeatherRecord(), but focuses on year + city, not full date.

**Outcome:** Demonstrates 2D array manipulation directly.

**Row-Major Access**



**Explanation:** Traverses 2D array row by row (year by year).

**Outcome:** Displays all records in row-major order.

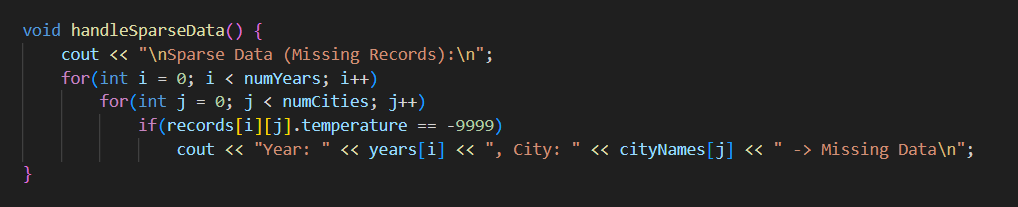
**Column-Major Access**



**Explanation:** Traverses column by column (city by city).

**Outcome:** Shows how data can be accessed differently in 2D arrays.

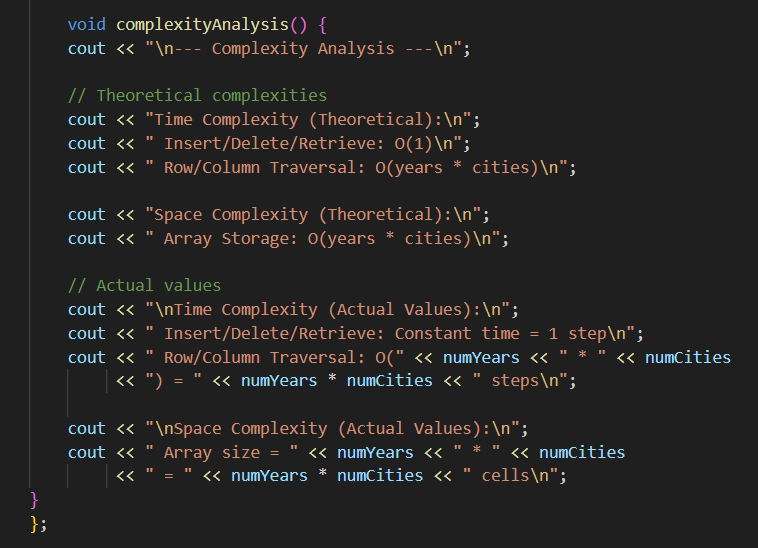
**Handle Sparse Data**



**Explanation:** Traverses column by column (city by city).

**Outcome:** Shows how data can be accessed differently in 2D arrays.

**Complexity Analysis**



**Explanation:**

Shows theoretical and actual time/space complexity.

O(1) for single record operations, O(years \* cities) for traversals.

**Outcome:** Helps understand performance characteristics of the system.

**Main Program**



**Explanation:**

Menu-driven program.

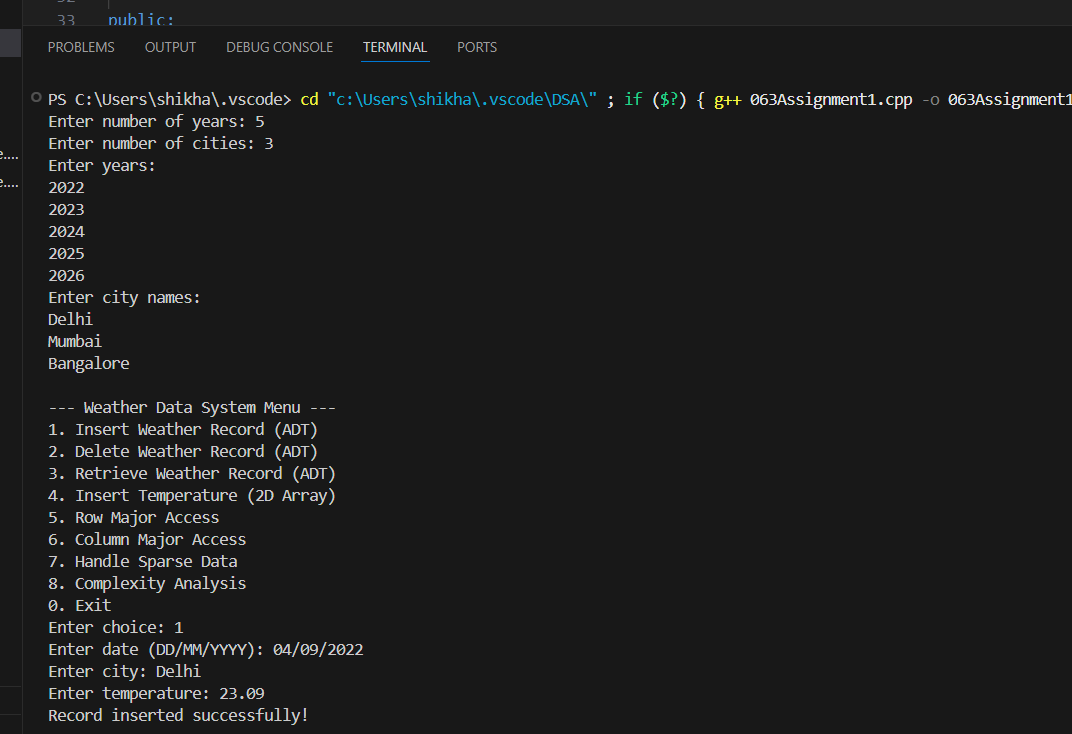
Uses do-while loop to continuously take user input until 0 (exit).

Each menu option calls the corresponding class function.

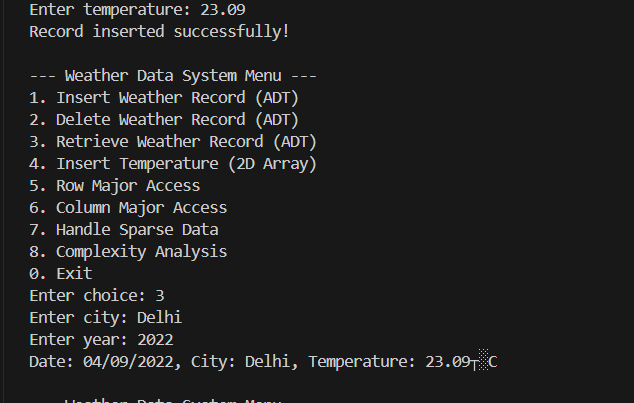
**Outcome:** User can insert, delete, retrieve, traverse, and analyse weather data interactively.

**Test Cases**

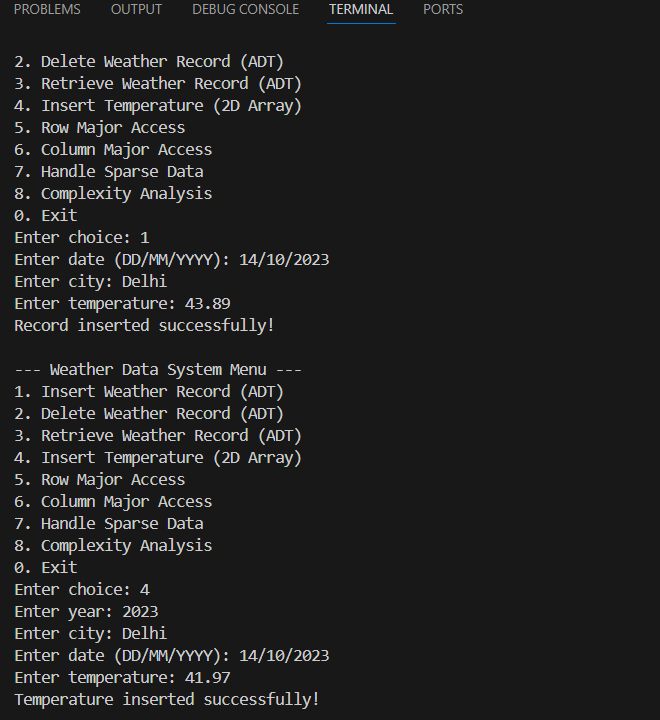
**Test Case: 1 Insert Weather Record**

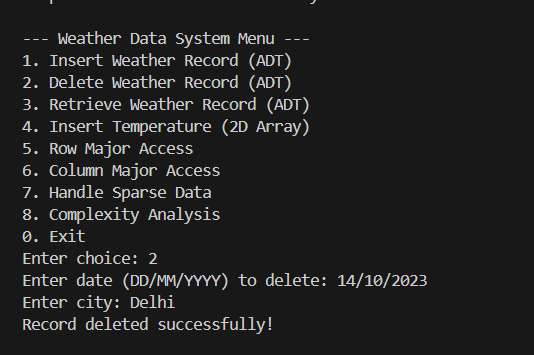


**Test Case: 2 Retrieve Weather Record (ADT)**

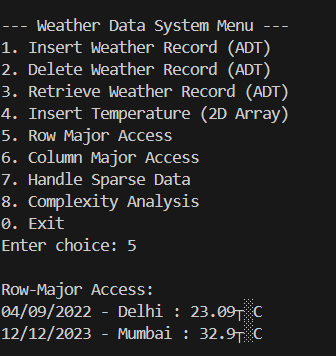
****

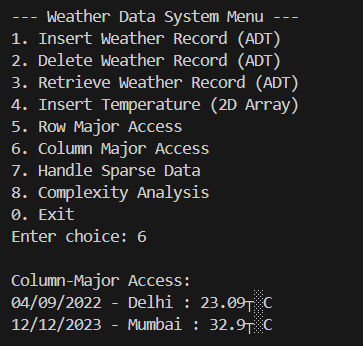
**Test Case: 3 Insert Temperature (2D Array)**

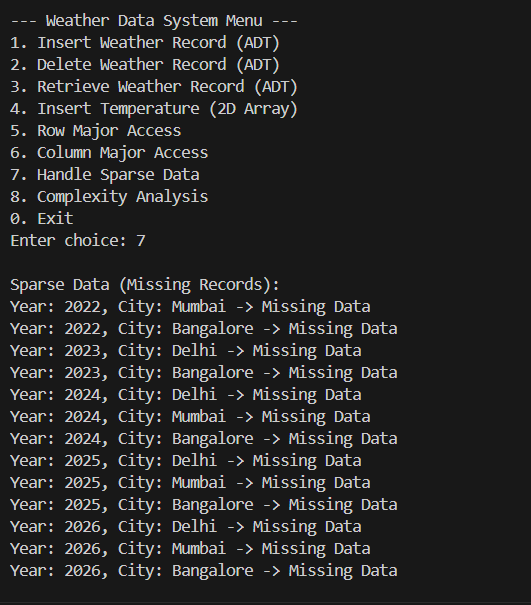
****

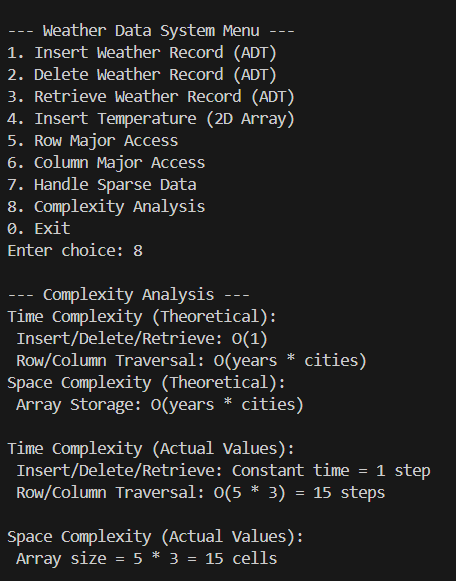
**Test Case: 4 Delete Weather Record**

**Test Case: 5 Row Major Access**

****

**Test Case: 6 Column Major Access**

**Test Case: 7 Handle Sparse Data** ****

**Test Case: 8 Complexity Analysis** ****