Functions and Libraries

Programming Abstractions for Geoinformatics with C++

Depicted Candela

December 4, 2024

Exercises

Exercise 1

Given three points $A(x_1, y_1)$, $B(x_2, y_2)$, and $C(x_3, y_3)$ in 2D space, write a C++ program to calculate the area of the triangle formed by these points using the formula:

$$\text{Area} = \frac{1}{2} |x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)|$$

Exercise 2

Write a C++ program to check if three given points $A(x_1, y_1)$, $B(x_2, y_2)$, and $C(x_3, y_3)$ are collinear. Points are collinear if the area of the triangle formed by them is zero.

Exercise 3

Given two lines defined by the equations $a_1x + b_1y = c_1$ and $a_2x + b_2y = c_2$, write a C++ program to find their intersection point. If the lines are parallel, the program should indicate this.

Exercise 4

Write a C++ program to compute the centroid C of a polygon given its vertices (x_1, y_1), (x_2, y_2), ..., (x_n, y_n). The centroid C is given by:

$$C_x = \frac{1}{6A} \sum_{i=1}^{n-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$C_y = \frac{1}{6A} \sum_{i=1}^{n-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

where A is the area of the polygon.

Exercise 5: Geometric Shape Interface for Geodesy

Objective:
Create an interface for geometric shapes commonly used in geodesy, such as circles (representing the Earth), lines (meridians and parallels), and points (specific locations). This interface will help geodesists calculate basic properties, such as distances and areas, which are essential in their fieldwork.

Requirements:

- Define an interface file `geoshapes.h` that exports:
  - An enumerated type `ShapeType` with values `CIRCLE`, `LINE`, and `POINT`.
  - A constant `PI` representing the value of π, to be used in circle calculations.
  - A structure `GeoPoint` that contains the latitude and longitude of a point.
  - Functions `calculateArea(ShapeType type)` and `calculateDistance(GeoPoint a, GeoPoint b)`.

Expected Outcomes:

- The `calculateArea` function should compute the area for a circle (approximating Earth's surface area) when the `ShapeType` is `CIRCLE`. For `LINE` and `POINT`, it should return zero.
- The `calculateDistance` function should return the great-circle distance between two points on the Earth's surface, using the haversine formula.

Exercise 6: Topological Operations Interface

Objective:
Design an interface that allows basic topological operations related to geodesy, such as determining the direction between two points and computing the midpoint on the surface of a sphere.

Requirements:

- Define an interface file `topooperations.h` that exports:
  - The `Direction` enumerated type with values `NORTH`, `EAST`, `SOUTH`, and `WEST`.
  - A function `Direction getDirection(GeoPoint start, GeoPoint end)` that returns the primary compass direction from the start point to the end point.
  - A function `GeoPoint midpoint(GeoPoint a, GeoPoint b)` that returns the midpoint between two geographical points on the Earth's surface.

Expected Outcomes:

- The `getDirection` function should return the primary compass direction based on the bearing calculated between the two points.
- The `midpoint` function should return a new `GeoPoint` that is the geographical midpoint, considering the spherical shape of the Earth.