Answers to Geometrical Procedures with Spatial Coordinates

Cadastral Engineer and Geodesist Training

August 2, 2024

Answers

Exercise 1

```cpp
// C++ Program to calculate Euclidean distance between two points
#include <iostream>
#include <cmath>

struct Point {
    double x, y, z;
};

double euclideanDistance(Point A, Point B) {
    return sqrt(pow(B.x - A.x, 2) + pow(B.y - A.y, 2) + pow(B.z - A.z,
        2));
}

int main() {
    Point A = {1.0, 2.0, 3.0};
    Point B = {4.0, 5.0, 6.0};

    std::cout << "Distance: " << euclideanDistance(A, B) << std::endl;
    return 0;
}
```

Exercise 2

```cpp
// C++ Program to find the midpoint between two points
#include <iostream>

struct Point {
    double x, y, z;
};

Point midpoint(Point A, Point B) {
    Point M;
    M.x = (A.x + B.x) / 2.0;
    M.y = (A.y + B.y) / 2.0;
    M.z = (A.z + B.z) / 2.0;
    return M;
}

int main() {
    Point A = {1.0, 2.0, 3.0};
    Point B = {4.0, 5.0, 6.0};

```

```
    Point M = midpoint(A, B);
    std::cout << "Midpoint: (" << M.x << ", " << M.y << ", " << M.z <<
        ")" << std::endl;
    return 0;
}
```

Exercise 3

```
// C++ Program to determine if three points form a right triangle
#include <iostream>
#include <cmath>

struct Point {
    double x, y, z;
};

double euclideanDistance(Point A, Point B) {
    return sqrt(pow(B.x - A.x, 2) + pow(B.y - A.y, 2) + pow(B.z - A.z,
        2));
}

bool isRightTriangle(Point A, Point B, Point C) {
    double AB = euclideanDistance(A, B);
    double BC = euclideanDistance(B, C);
    double CA = euclideanDistance(C, A);

    double AB2 = AB * AB;
    double BC2 = BC * BC;
    double CA2 = CA * CA;

    return (std::abs(AB2 + BC2 - CA2) < 1e-9 || std::abs(BC2 + CA2 -
        AB2) < 1e-9 || std::abs(CA2 + AB2 - BC2) < 1e-9);
}

int main() {
    Point A = {0.0, 0.0, 0.0};
    Point B = {3.0, 0.0, 0.0};
    Point C = {0.0, 4.0, 0.0};

    if (isRightTriangle(A, B, C)) {
        std::cout << "The points form a right triangle." << std::endl;
    } else {
        std::cout << "The points do not form a right triangle." << std
            ::endl;
    }

    return 0;
}
```

Exercise 4

```
// C++ Program to calculate the area of a triangle using cross product
#include <iostream>
#include <cmath>

struct Point {
```

```
    double x, y, z;
};

struct Vector {
    double x, y, z;
};

Vector crossProduct(Vector u, Vector v) {
    Vector cross;
    cross.x = u.y * v.z - u.z * v.y;
    cross.y = u.z * v.x - u.x * v.z;
    cross.z = u.x * v.y - u.y * v.x;
    return cross;
}

double magnitude(Vector v) {
    return sqrt(v.x * v.x + v.y * v.y + v.z * v.z);
}

double triangleArea(Point A, Point B, Point C) {
    Vector AB = {B.x - A.x, B.y - A.y, B.z - A.z};
    Vector AC = {C.x - A.x, C.y - A.y, C.z - A.z};

    Vector cross = crossProduct(AB, AC);
    return 0.5 * magnitude(cross);
}

int main() {
    Point A = {0.0, 0.0, 0.0};
    Point B = {3.0, 0.0, 0.0};
    Point C = {0.0, 4.0, 0.0};

    std::cout << "Area of the triangle: " << triangleArea(A, B, C) <<
        std::endl;
    return 0;
}
```