Übungsblatt 3 19.09.2023

Aufgabe 1

Diese Aufgabe soll ohne die Verwendung der Schlüsselwörter `auto` und `decltype` erfolgen, d. h. ihr müsst alle Datentypen explizit nennen. Achtet außerdem auf die Speicherverwaltung beim Umgang mit Pointern auf Objekte. Wo lassen sich Standard-Algorithmen mit Lambda-Funktionen sinnvoll einsetzen?

a) Erweitert die Klasse `Point` aus Aufgabe 3 von Übungsblatt 2 so, dass nach dem erfolgreichen Einlesen ein Objekt der Klasse erzeugt und mit den eingelesenen Zahlen initialisiert wird. Überladet die Stream-I/O-Operatoren `<<` und `>>` für die Klasse `Point` wie hier beschrieben: `https://stackoverflow.com/q/4421706`.

b) Erweitert dann das Programm so, dass diese Operatoren für das Einlesen und die Ausgabe der eingelesenen Daten genutzt werden.

c) Nun sollen in einer Schleife zeilenweise so lange neue mit `new()` angelegte Objekte der Klasse `Point` eingelesen und in einem `std::vector<unique_ptr<Point>>` gespeichert werden, bis beim Einlesen ein Fehler auftritt. Anschließend sollen alle eingelesenen Punkte ausgegeben werden. Wenn weniger als ein Objekt eingelesen wurde, soll eine Fehlermeldung ausgegeben und das Programm beendet werden.

d) Implementiert im Destruktor für `Point` eine Debug-Ausgabe auf `cout`, anhand derer sich nachvollziehen lässt, dass ein `Point`-Objekt zerstört worden ist.

e) Implementiert eine Funktion zur Berechnung der Länge des durch ein Objekt der Klasse `Point` repräsentierten Vektors. Zu jedem eingelesenen Objekt soll zusätzlich die Länge ausgegeben werden.

f) Falls noch nicht erledigt, überladet die Operatoren für die Subtraktion zweier Vektoren. Implementiert dann eine Funktion, die durch Berechnung der Differenz und deren Länge die Distanz zwischen zwei gegebenen Punkten berechnet. Zu jedem eingelesenen Objekt soll nunmehr die Distanz zum insgesamt letzten eingelesenen Objekt ausgegeben werden.

g) Erweitert das Programm so, dass nur der Punkt ausgegeben wird, dessen Distanz zum letzten eingelesenen Punkt am kleinsten ist. Wenn weniger als zwei Objekte eingelesen wurden, soll eine Fehlermeldung ausgegeben und das Programm beendet werden.

Aufgabe 2

In dieser Aufgabe sollt ihr eine Ordnung auf `Point` entwickeln. Dazu definiert im ersten Schritt eine Hilfsklasse `Polar`, deren Konstruktor eine Instanz der Klasse `Point` als Argument erhält.

`Polar` soll die Polarkoordinaten[1] zu dem gegebenen Punkt liefern. (Hinweis: Verwendet dazu die Funktionen `atan2`, `pow` und `sqrt` aus `<cmath>`.)

Anschließend definiert einen `operator<` auf `Point`-Objekten a und b, der genau dann `true` liefert, wenn der Winkel von a kleiner ist als der Winkel von b oder beide Winkel gleich sind und der Radius von a kleiner ist als der Radius von b.

Zum Abschluss könnt ihr mit `min_element` das kleinste Element in dem Point-Vektor ermitteln oder mittels `any_of` herausfinden, ob ein Punkt in einem bestimmten Quadranten liegt oder sortieren und alle doppelten Punkte entfernen usw.

[1] Siehe dazu auch `https://de.wikipedia.org/wiki/Polarkoordinaten#Umrechnung_von_kartesischen_Koordinaten_in_Polarkoordinaten`.