**9η Προγραμματιστική άσκηση για το μάθημα**
**343 Εισαγωγή στον Προγραμματισμό**
**Μιχάλης Μπέκος / Κωνσταντίνα Τζουβάρα**

**Χειμερινό εξάμηνο 2025-2026**
**Πανεπιστήμιο Ιωαννίνων**
**7/12/2025**

# 9η Προγραμματιστική Εργασία

**Στόχοι:** Εξοικείωση με τους δείκτες και τις ροές εισόδου/εξόδου αρχείων.

## Άσκηση 1: **Ανάπτυξη κλάσης με πεδία-δείκτες σε αντικείμενα.**

Στο αρχείο `transform.cpp` που σας δίνεται, θέλουμε να φτιάξουμε ένα πρόγραμμα που να προσομοιώνει ένα μετασχηματισμό στο δισδιάστατο Ευκλείδειο χώρο. Υπενθυμίζεται ότι ένας τέτοιος μετασχηματισμός ορίζεται από δύο συνιστώσες, κλιμάκωση (scale) και περιστροφή (rotation). Για την υλοποίηση σας, σας δίνονται οι υλοποιήσεις των κλάσεων `Scale` και `Rotation`. Η πρώτη ορίζει δύο κατευθύνσεις για την κλιμάκωση (`sx` για την κατεύθυνση του άξονα `x` και `sy` για την κατεύθυνση του άξονα `y`), ενώ η δεύτερη μια γωνία περιστροφής σε `radians` (`angle`).

Καλείστε να υλοποιήσετε την κλάση `Transform2D`, η οποία περιλαμβάνει τα παρακάτω (ιδιωτικά) πεδία:

```
Scale* scale;          // δείκτης σε αντικείμενο Scale
Rotation* rotation;    // δείκτης σε αντικείμενο Rotation
```

Αντικείμενα της κλάσης `Transform2D` υποστηρίζουν τις παρακάτω (δημόσιες) μεθόδους:

| | |
|---|---|
| **`Transform2D(Scale* s, Rotation* r)`** | Δημιουργεί μετασχηματισμό με τα δοθέντα ορίσματα (δείκτες) |
| **`Scale* getScale()`** | Επιστρέφει δείκτη προς το αντικείμενο `scale` για την κλιμάκωση |
| **`Rotation* getRotation()`** | Επιστρέφει δείκτη προς το αντικείμενο `rotation` για την περιστροφή |
| **`void setScale(Scale* s)`** | Αναθέτει τον νέο δείκτη στο αντικείμενο `scale` |
| **`void setRotation(Rotation* r)`** | Αναθέτει τον νέο δείκτη στο αντικείμενο `rotation` |
| **`double determinant()`** | Υπολογίζει την ορίζουσα της κλιμάκωσης, η οποία ορίζεται ως το γινόμενο `sx * sy` |
| **`bool preservesOrientation()`** | Ελέγχει αν ο μετασχηματισμός διατηρεί τον προσανατολισμό (`true` ανν `determinant > 0`) |
| **`bool isIsotropic()`** | Ελέγχει αν ο μετασχηματισμός είναι ισότροπος (`true` ανν `sx == sy`) |

| | |
|---|---|
| **`Transform2D* inverse()`** | Επιστρέφει νέο αντικείμενο `Transform2D`, που αντιστοιχεί στον αντίστροφο μετασχηματισμό:<br>• Αν κάποιο από τα `sx`, `sy` είναι `0`, επιστρέφει `nullptr`<br>• Διαφορετικά, η κλιμάκωση του αντίστροφου μετασχηματισμού αποτελείται από νέο `Scale (1/sx, 1/sy)` και νέο Rotation `(-angle)`<br>• Η μέθοδος επιστρέφει νέο δυναμικά δεσμευμένο αντικείμενο |

Όταν ολοκληρώσετε την υλοποίηση της κλάσης `Transform2D`, μελετήστε την συνάρτηση-οδηγό `main()`, η οποία σας δίνεται (είναι πλήρως υλοποιημένη).

Βεβαιωθείτε ότι έχει γίνει σωστή χρήση των δεικτών και ότι καταλαβαίνετε τις επιπτώσεις που έχει η αλλαγή των αντικειμένων στα οποία δείχνουν οι δείκτες.

Τρέξτε το πρόγραμμά σας, για να ελέγξετε ότι εκτελείται χωρίς σφάλματα μνήμης.

## Άσκηση 2: **Αντιγραφή Μέρους Περιεχομένου Αρχείου (extractLines)**

Να γραφεί η δήλωση και η υλοποίηση της συνάρτησης `bool extractLines(string filename, int start, int end)`, η οποία δέχεται ως παραμέτρους το όνομα ενός αρχείου κειμένου (`filename`) και δύο αριθμούς γραμμών (`start` και `end`).

Η συνάρτηση θα ελέγχει αν το αρχείο υπάρχει στον δίσκο και αν υπάρχει:

- Δημιουργεί νέο αρχείο με όνομα `filename.extract`.
- Αντιγράφει στο νέο αρχείο μόνο τις γραμμές από την `start` έως και την `end`.

Αν το αρχείο δεν υπάρχει τότε δεν δημιουργείται νέο αρχείο. Η συνάρτηση επιστρέφει `true` αν η διαδικασία ολοκληρώνεται επιτυχώς και `false` σε οποιαδήποτε περίπτωση αποτυχίας.

Στη συνέχεια υλοποιήστε τη συνάρτηση–οδηγό `main()`, η οποία ζητά από τον χρήστη το όνομα ενός αρχείου κειμένου, τον αριθμό πρώτης γραμμής (`start`), τον αριθμό τελευταίας γραμμής (`end`), καλεί τη συνάρτηση `extractLines()` και εμφανίζει κατάλληλο μήνυμα ανάλογα με την επιτυχία ή αποτυχία της διαδικασίας.