# Εργαστήριο 2 - Δημιουργία και Καταστροφή Αντικειμένων: Constructors, Destructors και λίστες αρχικοποίησης

Στο δεύτερο εργαστήριο θα εμβαθύνουμε στη λειτουργία της δημιουργίας και καταστροφής αντικειμένων.

Στην C++, κάθε φορά που δημιουργείται ένα αντικείμενο μιας κλάσης, καλείται αυτόματα μια ειδική συνάρτηση που ονομάζεται **constructor**.

Αντίστοιχα, όταν το αντικείμενο πάψει να υπάρχει, για παράδειγμα όταν τελειώσει το scope μέσα στο οποίο δηλώθηκε, καλείται αυτόματα μια άλλη ειδική συνάρτηση που ονομάζεται **destructor**.

Σε αυτό το εργαστήριο θα επεκτείνετε τις δύο κλάσεις που δημιουργήσατε στο προηγούμενο εργαστήριο, δηλαδή την Protein και τη Gene, προσθέτοντας σε καθεμία έναν constructor και έναν destructor.

Ο constructor θα χρησιμοποιείται για να αρχικοποιεί τα δεδομένα του αντικειμένου με προκαθορισμένες ή δοσμένες τιμές (θα εκτυπώνει και ένα μήνυμα), ενώ ο destructor θα εμφανίζει στην οθόνη μήνυμα που δηλώνει την καταστροφή του αντικειμένου, ώστε να παρακολουθήσετε τη ροή δημιουργίας και τερματισμού.

## *Λίστες Αρχικοποίησης*

Θα παρουσιαστεί η έννοια της **λίστας αρχικοποίησης (initialization list)**, που χρησιμοποιείται για την απευθείας αρχικοποίηση των μελών μιας κλάσης πριν την εκτέλεση του σώματος του constructor.

Η λίστα αρχικοποίησης είναι προτιμότερη από την απλή ανάθεση τιμών, επειδή εξασφαλίζει ότι τα μέλη παίρνουν τις αρχικές τους τιμές τη στιγμή της δημιουργίας του αντικειμένου.

Με αυτόν τον τρόπο κάθε μέλος **αρχικοποιείται σωστά πριν εκτελεστεί το σώμα του constructor**.

Οι λίστες αρχικοποίησης είναι απαραίτητες όταν τα μέλη είναι σταθερά (const) ή

αναφορές (references), αφού αυτά δεν μπορούν να αλλάξουν τιμή μετά τη δημιουργία του αντικειμένου.

Για παράδειγμα, μπορείτε να τροποποιήσετε την κλάση Gene ώστε να περιλαμβάνει ένα σταθερό μέλος που δηλώνει τον οργανισμό προέλευσης του γονιδίου:

```
class Gene {
private:
    const string organism;
    ...
```

## *Κυρίως πρόγραμμα*

Στο κυρίως πρόγραμμα θα δημιουργήσετε μερικά αντικείμενα από τις δύο κλάσεις και θα παρατηρήσετε πότε ακριβώς καλούνται οι constructors, δηλαδή όταν τα αντικείμενα δημιουργούνται, και πότε οι destructors, δηλαδή όταν ολοκληρώνεται η εκτέλεση της συνάρτησης main ή όταν ένα αντικείμενο βγαίνει από το πεδίο ορατότητας του (πχ δημιουργείται μέσα σε συνάρτηση).

Προαιρετικά μπορείτε να προσθέσετε έναν μετρητή που αυξάνεται κάθε φορά που δημιουργείται ένα αντικείμενο και μειώνεται όταν καταστρέφεται. Έτσι θα μπορείτε να βλέπετε πόσα αντικείμενα «ζουν» κάθε στιγμή στη μνήμη.

Στο τέλος του εργαστηρίου θα κατανοήσετε τον κύκλο ζωής ενός αντικειμένου, τη σωστή χρήση των constructors και destructors, τη σημασία της αρχικοποίησης μέσω λίστας και τη βασική χρήση των STL `string` και `vector` για ασφαλή και πιο ευέλικτο προγραμματισμό.

Στο επόμενο εργαστήριο θα οργανώσουμε τις κλάσεις μας έτσι ώστε να αποτελούνται από επιμέρους δομές που συνεργάζονται μεταξύ τους, εισάγοντας την έννοια της σύνθεσης, όπως ακριβώς ένα γονίδιο αποτελείται από μικρότερα λειτουργικά τμήματα του DNA.

## *STL*

Στη συνέχεια θα γίνει εισαγωγή σε δύο από τα πιο συχνά χρησιμοποιούμενα δοχεία της βιβλιοθήκης STL, τις `std::string` και `std::vector`.

Η `std::string` προσφέρει έναν ασφαλή και δυναμικό τρόπο διαχείρισης ακολουθιών χαρακτήρων, χωρίς περιορισμούς μεγέθους και χωρίς ανάγκη για συναρτήσεις όπως `strcpy` ή `strlen`.

Μπορείτε να αντικαταστήσετε τους πίνακες χαρακτήρων που χρησιμοποιήθηκαν στο προηγούμενο εργαστήριο με αντικείμενα `string` και να δείτε πόσο πιο απλός γίνεται ο κώδικας.

Παράλληλα, η `std::vector` παρέχει έναν δυναμικό πίνακα που μπορεί να αλλάζει μέγεθος κατά τη διάρκεια εκτέλεσης του προγράμματος.

Είναι ιδιαίτερα χρήσιμη για περιπτώσεις όπως η αποθήκευση πολλών ισομορφών ενός γονιδίου ή πολλών ακολουθιών πρωτεϊνών.

Μπορείτε να δημιουργήσετε έναν απλό `vector` που θα περιέχει ονόματα πρωτεϊνών ή γονιδίων και να προσθέτετε στοιχεία με την εντολή `push_back()`.