

1. Εισαγωγή & Motivation

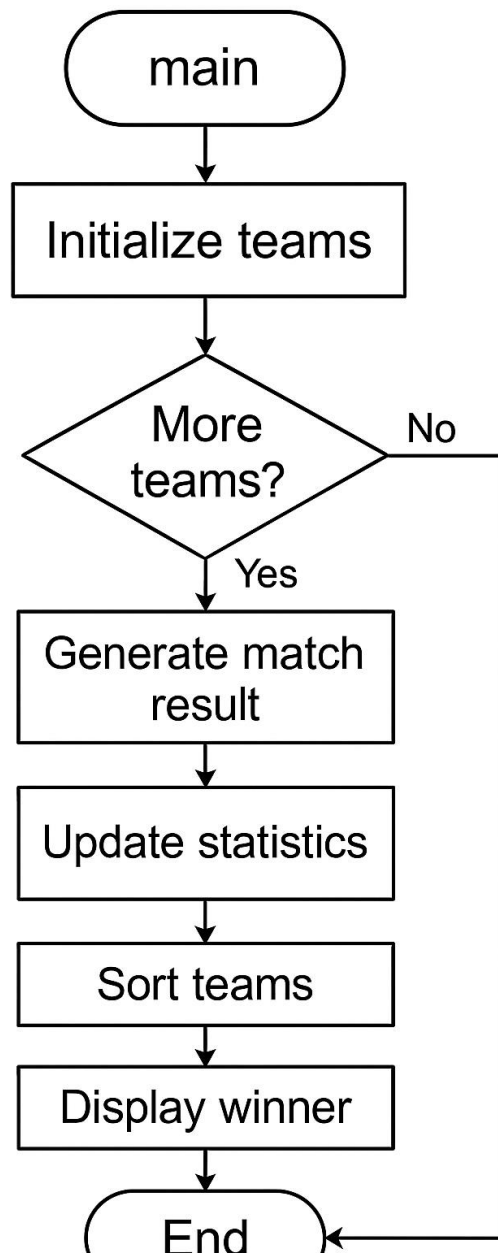
- Το project αφορά τη δημιουργία ενός προσομοιωμένου πρωταθλήματος ποδοσφαίρου (Super League Ελλάδα) σε C++.
- Σκοπός του είναι να παραχθούν όλα τα παιχνίδια μεταξύ των ομάδων, με τυχαία αποτελέσματα, και να δημιουργηθεί η τελική βαθμολογία.
- Δημιουργεί αυτόματα και διαχειρίζεται την βαθμολογία του ελληνικού πρωταθλήματος.
- Επιλέχθηκε επειδή παραμένει κοντά σε ένα ενδιαφέρον και ρεαλιστικό πλαίσιο.

2. Objective & Scope

- **Βασικοί στόχοι:**
 - Δημιουργία όλων των αγώνων (εντός και εκτός έδρας).
 - Εξομοίωση σκορ με τυχαίους αριθμούς.
 - Αυτόματος υπολογισμός και ταξινόμηση της βαθμολογίας.
- **Δυνατότητες:**
 - Υπολογισμός στατιστικών ομάδων (νίκες, ισοπαλίες, ήττες, γκολ υπέρ/κατά, βαθμοί).
 - Εμφάνιση του πρωταθλητή.
 - Εμφάνιση της κατάταξης.

3. System Architecture

- **Λειτουργία:**
 - Οι ομάδες αποθηκεύονται σε πίνακα από `struct Team`.
 - Δημιουργούνται όλα τα δυνατά ζευγάρια αγώνων (26 παιχνίδια ανά ομάδα).
 - Οι αγώνες ανακατεύονται για να σχηματιστούν αγωνιστικές.
 - Για κάθε αγώνα παράγεται τυχαίο σκορ και ενημερώνεται η βαθμολογία.
 - Οι ομάδες ταξινομούνται βάσει βαθμών και εμφανίζεται ο πίνακας.
- **Συνιστώσες:**
 - `struct Team`: αποθηκεύει τα δεδομένα κάθε ομάδας.
 - Πίνακας `Match`: αποθηκεύει τους αγώνες.
 - Συναρτήσεις `updateStats`, `sortTeams`, `printTable`, `generateGoals`.



4. Τεχνολογίες που Χρησιμοποιήθηκαν

Βιβλιοθήκες:

- `<iostream>` για είσοδο/έξοδο.
- `<iomanip>` για μορφοποίηση της εξόδου (στοίχιση πινάκων).
- `<cstdlib>` και `<ctime>` για παραγωγή τυχαίων αριθμών.
- `<algorithm>` για ταξινόμηση της βαθμολογίας (`sort()`).

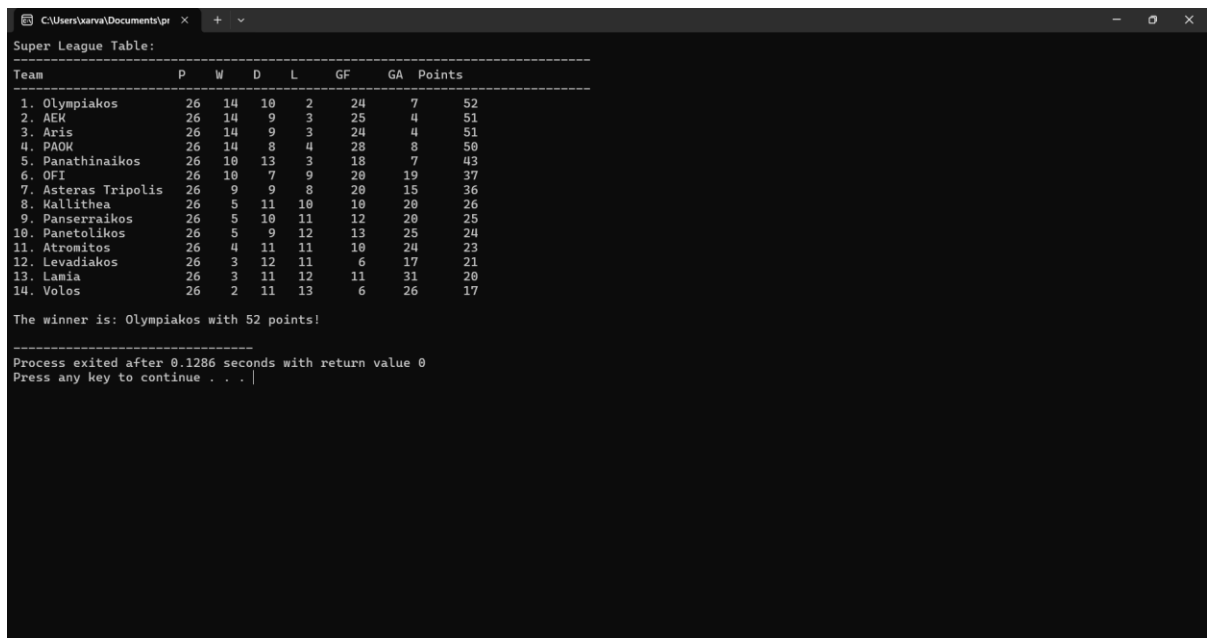
5. Κώδικας & Υλοποίηση

- Ο βασικός κώδικας βασίζεται σε δομές (struct) για τις ομάδες και αγώνες.
- Η ταξινόμηση γίνεται με `std::sort` και custom σύγκριση βάσει βαθμών.
- Χρήση συναρτησης `generateGoals` για ρεαλιστικά αποτελματα.

CODE: <https://github.com/arva63/SuperLeagueCPP/blob/main/README.md>

6. Αποτελέσματα & Demo

- Τα αποτελέσματα κάθε φορά αλλάζουν αλλά μπορω να στείλω ενα ενδεικτικό demo



```
C:\Users\arva\Documents\pr > x + v
Super League Table:
-----
Team      P   W   D   L   GF  GA  Points
-----
1. Olympiakos    26  14  10   2   24   7    52
2. AEK           26  14   9   3   25   4    51
3. Aris          26  14   9   3   24   4    51
4. PAOK          26  14   8   4   28   8    50
5. Panathinaikos 26  10  13   3   18   7    43
6. OFI           26  10   7   9   20  19    37
7. Asteras Tripolis 26   9   9   8   20  15    36
8. Kallithea     26   5  11  10   10  20    26
9. Panserraikos  26   5  10  11   12  20    25
10. Panetolikos  26   5   9  12   13  25    24
11. Atromitos    26   4  11  11   10  24    23
12. Levadiakos   26   3  12  11   6   17    21
13. Lamia        26   3  11  12   11  31    20
14. Volos        26   2  11  13   6   26    17

The winner is: Olympiakos with 52 points!

-----
Process exited after 0.1286 seconds with return value 0
Press any key to continue . . . |
```

7. Comparison with AI generated code

Κριτήριο	1ος Κώδικας (Δικός μας)	2ος Κώδικας (AI)
Δομή Δεδομένων	<code>struct Team</code>	<code>class Team</code>
Επεκτασιμότητα / modularity	Μέτρια	Καλή

Χρήση τυχαίων παραμέτρων	Με βάση strength (float), πιο ρεαλιστικό	Απλή isStrong boolean, πιο απλοϊκό
Αναγνωσιμότητα	Καθαρός	Πιο καθαρή δομή (μέσω κλάσης)
Αποδοτικότητα	Περισσότεροι υπολογισμοί (π.χ. strength, goal modifiers)	Πιο απλός και ελαφρύς
Στατιστικά ανά αγώνα	Πολύ αναλυτικά (W/D/L, GF/GA)	Επίσης αναλυτικά
Προσομοίωση	Σωστός διπλός γύρος	Τεσσερις γυροι
Ταξινόμηση ομάδων	Χειροκίνητο bubble-sort	STL sort() με compareTeams
Εμφάνιση αποτελεσμάτων	Καλή, λίγο πυκνή	Πιο απλή και με προβλήματα στα κενά
Ακρίβεια προσομοίωσης	Υψηλότερη (χρήση δύναμης και modifier)	Απλοποιημένη

Τι Κάνει Καλύτερα ο 1ος Κώδικας (Δικός μας):

- Κάνει **πιο ρεαλιστική προσομοίωση** σκορ με χρήση attackStrength - defenseStrength, και προσθέτει modifier, κάνοντάς τον πιο πραγματική.
- Διατηρεί **όλα τα στατιστικά** αναλυτικά (και τα draws, κάτι που λείπει από τον 2ο).
- Υπάρχει **σαφής διαχωρισμός συναρτήσεων** (generateGoals, updateStats, sortTeams).

Τι Κάνει Καλύτερα ο 2ος Κώδικας (AI):

- Καλύτερη **χρήση αντικειμενοστραφούς προγραμματισμού** . Ορίζεται class Team με μεθόδους (goalDifference, printStats).
- Χρησιμοποιεί **std::vector** και **std::sort**, που είναι πιο σύγχρονη και αποδοτική προσέγγιση από τους πίνακες και bubble sort.
- Πιο **ευανάγνωστος** για κάποιον που θέλει να καταλάβει γρήγορα τι κάνει το πρόγραμμα.
- Λιγότερος κώδικας — πιο **lightweight**, αλλά χάνει σε ρεαλισμό.

Αποδοτικότητα (Efficiency)

- Ο **δικός μας κώδικας** κάνει πιο πολλούς υπολογισμούς λόγω της λεπτομερούς προσομοίωσης (modifier, rand 2 φορές ανά ομάδα, κ.λπ.).
- Ο **έτοιμος** είναι πιο γρήγορος σε runtime, λόγω της απλότητας. Επίσης, η STL `sort()` είναι ταχύτερη από bubble sort.

AI code: <https://github.com/arva63/SuperLeagueCPP/blob/main/AI%20code>

8. Conclusions & Lessons learned

- ✓ Πώς να σχεδιάζεις και υλοποιείς πλήρες πρόγραμμα
- ✓ Πώς να χειρίζεσαι structs, functions, τυχαίους αριθμούς
- ✓ Πώς να διαχειρίζεσαι δεδομένα ομάδων και ταξινόμηση