# Strukture podataka i algoritmi

## Projektni zadatak 010\_WORLD\_CUP

Vaš zadatak je napisati program (konzolnu aplikaciju u C++) koji će korisniku omogućiti rad s podacima o svjetskim prvenstvima u nogometu. Program mora biti smisleno strukturiran i intuitivan. U nastavku su opisane funkcionalnosti koje treba program imati. Za izradu projekta se koristi datoteke naziva **SPA\_PROJ\_010\_WORLD\_CUP\_data\_1.csv** (stupce koji nisu ovdje opisani možete ignorirati):

* + Year: The year in which the match was played
  + Datetime: The Date on which the match was played along with a 24 hour format time
  + Stage: The stage at which the match was played
  + Stadium: Stadium name where the match was held
  + City: The city name, where the match was played
  + Home Team Name: Home team country name
  + Home Team Goals: Total goals scored by the home team by the end of the match
  + Away Team Goals: Total goals scored by the away team by the end of the match
  + Away Team Name: Away team country name
  + Attendance: Total crowd present at the stadium
  + Home Team Initials: Home team country's three letter initials
  + Away Team Initials: Away team country's three letter initials

1. (Bodova: **1**) Pokretanjem aplikacije prikažite korisniku sljedeći izbornik:

Pretraga po reprezentaciji

Prikaz po godini

Kopiranje u novu listu

Prikaz po godini i domaćinu

1. (Bodova: **2**) Dizajnirajte i implementirajte tip podataka Match koji može čuvati sve podatke o utakmici iz datoteke **SPA\_PROJ\_010\_WORLD\_CUP\_data\_1.csv**. Pripremite tip podataka tako da se može konstruirati na smislene načine te definirajte gettere, settere i potrebne metode za svakog člana. Omogućite korisniku da pozivanjem metode na objektu može ispisati osnovne podatke o utakmici na ekran (datum i vrijeme u hrvatskom formatu, stadion, sudionike i rezultat). Primjerice, za neku utakmicu treba ispisati: 13.7.1930, Pocitos, France 4:1 Mexico.
2. (Bodova: **3**) Kad korisnik odabere „Pretraga po reprezentaciji“, prvo prekopirajte sve utakmice iz vektora u odgovarajući kontejner. Zatim pitajte korisnika da upiše troslovnu oznaku reprezentacije pa mu nakon toga ispišite podatke o svim utakmicama te reprezentacije. Osigurajte da se pretraživanje odvija u logaritamskoj složenosti koristeći najbolji od sljedećih kontejnera: map, multimap, set, multiset.
3. (Bodova: **2**) Promijenite prethodni zadatak tako da pretraživanje radite prvo po odabranom kontejneru, a zatim i po vektoru i po listi (kopirajte podatke i u listu). Ispišite top ljestvicu brzine kontejnera: prvo prikažite najbrži, onda srednji pa najsporiji.
4. (Bodova: **2**) Kad korisnik odabere „Prikaz po godini“, koristeći prioritetni red ispišite utakmice rastuće prema godini.
5. (Bodova: **3**) Kad korisnik odabere „Kopiranje u novu listu“, kreirajte novu listu i napravite sljedeće:

* Prekopirajte sve utakmice u novi kontejner koji omogućuje da se pretraživanje po ukupnom broju postignutih golova na utakmici odvija u logaritamskoj složenosti (birajte map, multimap, set ili multiset).
* Sve dok to korisnik želi, omogućite mu unos ukupnog broja postignutih golova na utakmici.
* Kad korisnik unese vrijednost, pronađite sve utakmice s toliko golova u vašem odabranom kontejneru i kopirajte ih u novu listu.
* Ponavljajte sve dok to korisnik želi.
* Na kraju ispišite sve utakmice iz nove liste.

1. (Bodova: **3**) Kad korisnik odabere „Prikaz po godini i domaćinu“, koristeći prioritetni red ispišite sve utakmice rastuće prema godini. Sve utakmice u istoj godini obavezno prikažite sortirane rastuće prema punom nazivu domaćina.
2. (Bodova: **1**) Dodajte u izbornik sljedeće opcije:

Selection vs Insertion

Pogodi trajanje

Lets Bogo!

The Merge Dance

1. (Bodova: **3**) Kad korisnik odabere „Selection vs Insertion“, napravite sljedeće:

* Kreirajte dva pomoćna vektora *v1* i *v2* i iskopirajte u svako od njih sve utakmice.
* Uzmite s predavanja Selection i Insertion sortiranja te ih promijenite tako da sortiraju utakmice, rastuće po troslovnoj oznaci reprezentacije.
* Sortirajte vektor *v1* koristeći Selection sort.
* Sortirajte vektor *v2* koristeći Insertion sort.
* Ispišite koliko je trajalo svako sortiranje.

1. (Bodova: **2**) Promijenite prethodni zadatak tako da svaki algoritam pri sortiranju zbraja broj zamjena mjesta (*swap*) koje napravi i vraća to kao izlaznu vrijednost iz funkcije. Sortirajte pa ispišite za oba sortiranja broj zamjena mjesta.
2. (Bodova: **3**) Kad korisnik odabere opciju „Pogodi trajanje“, generirajte jedan slučajni broj *n* između 1 i 100.000.000 te pitajte korisnika da upiše koliko milisekundi misli da će trajati razbacivanje, sortiranje i binarno pretraživanje tog polja u potrazi za brojem 7. Kad korisnik upiše vrijednost, generirajte to polje s vrijednostima od 1 do *n*, razbacajte ga, sortirajte ga te binarnim pretraživanjem i pronađite broj 7. Ispišite korisniku kolika je razlika između njegove pogođene vrijednosti trajanja i konkretnog trajanja.
3. (Bodova: **4**) Kad korisnik odabere „Lets Bogo!“, slučajnim odabirom prekopirajte pet utakmica u novi vektor. Isprogramirajte svoju vlastitu determinističku verziju bogo sorta te je iskoristite za slaganje utakmica po godini u rastućem redoslijedu. Ispišite na ekranu svaku isprobanu permutaciju.
4. (Bodova: **4**) Kad korisnik odabere opciju „The Merge Dance“, pitajte ga da upiše broj *n* koji predstavlja broj elemenata. Nakon što upiše broj, u vektor ubacite slučajnim redoslijedom *n* pokemona. Zatim vektor sortirajte Merge sortom (koristite i promijenite prema potrebi implementaciju s predavanja) te za vrijeme sortiranja radite sljedeće:

* Svaki put kad se napravi poziv rekurzivne funkcije, brojač povećajte za 1
* Svaki put kad se završi poziv rekurzivne funkcije, brojač smanjite za 1.
* Za svaku promjenu brojača iscrtajte u tekstualnu datoteku jedan redak koji sadrži onoliko znakova # kolika je trenutna vrijednost brojača.

1. (Bodova: **1**) Dodajte u izbornik sljedeće opcije:

Pretraga po troslovnoj oznaci

Vizualiziraj

Izrada stringa iz stringa

1. (Bodova: **2**) Kad korisnik odabere „Pretraga po troslovnoj oznaci“, prvo prekopirajte utakmica iz vektora u odgovarajući kontejner. Zatim pitajte korisnika da upiše troslovnu oznaku koje želi tražiti pa mu nakon toga ispišite podatke o svim utakmicama te reprezentacije. Osigurajte da se pretraživanje odvija u najboljoj mogućnoj složenosti koristeći neki od sljedećih kontejnera: unordered\_map, unordered\_multimap, unordered\_set, unordered\_multiset.
2. (Bodova: **2**) Promijenite prethodni zadatak tako da pretraživanje radite prvo po odabranom kontejneru, a zatim i po vektoru i po listi (kopirajte podatke i u listu). Ispišite top ljestvicu brzine kontejnera: prvo prikažite najbrži, onda srednji pa najsporiji.
3. (Bodova: **5**) Kad korisnik odabere „Vizualiziraj“, natrpajte sve utakmica u unordered\_multimap tako da ključ bude stadion odigravanja. Nakon toga, vizualizirajte *hash* tablicu iz unordered\_multimap tako da iscrtate sve njene *buckete* te uz svaki *bucket* prikažete i reprezentacije koje su tu igrale. Primjer početka ispisa (vrijednosti su izmišljene):

Bucket 0: Estadio Sausalito (ESP vs MEX) -- Arena Amazonia (CMR vs CRO)

Bucket 1: El Molinon (FRG vs ALG) -- Estadio Azteca (ARG vs FRG)

Bucket 2: [EMPTY]

1. (Bodova: **6**) Kad korisnik odabere „Izrada imena iz imena“, omogućite mu da odabere jednu utakmicu. Koristeći unordered\_map, unordered\_multimap, unordered\_set ili unordered\_multiset, ispišite može li se kreirati troslovna oznaka gostujuće reprezentacije koristeći raspoloživa slova iz naziva stadiona (pri tome mala i velika slova smatrajte jednakima). Primjerice:

* Ako je ime stadiona „Olita Stadium Big Eye“, a oznaka gostujuće reprezentacije „BEL“, onda se oznaka može konstruirati iz imena stadiona.
* Ako je ime stadiona „Olita Stadium Big Eye“, a oznaka gostujuće reprezentacije „ESP“, onda se oznaka ne može konstruirati iz imena stadiona jer nedostaje slovo p.