# Strukture podataka i algoritmi

## Projektni zadatak 007\_GOOGLE\_PLAY

Vaš zadatak je napisati program (konzolnu aplikaciju u C++) koji će korisniku omogućiti rad s podacima o Google Play aplikacijama. Program mora biti smisleno strukturiran i intuitivan. U nastavku su opisane funkcionalnosti koje treba program imati. Za izradu projekta se koristi datoteka **SPA\_PROJ\_007\_GOOGLE\_PLAY\_data.csv**. Opis stupaca:

* App: Application name
* Category: Category the app belongs to
* Rating: Overall user rating of the app (as when scraped)
* Reviews: Number of user reviews for the app (as when scraped)
* Size: Size of the app (as when scraped)
* Installs: Number of user downloads/installs for the app (as when scraped)
* Type: Paid or Free
* Price: Price of the app (as when scraped)
* Content Rating: Age group the app is targeted at - Children / Mature 21+ / Adult
* Genres: An app can belong to multiple genres (apart from its main category). For eg, a musical family game will belong to Music, Game, Family genres.
* Last Updated: Date when the app was last updated on Play Store (as when scraped)
* Current Ver: Current version of the app available on Play Store (as when scraped)
* Android Ver: Min required Android version (as when scraped)

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

Pretraga po kategoriji

Prikaz po ocjeni

Kopiranje u novi vektor

Prikaz po kategoriji

1. (Bodova: **2**) Dizajnirajte i implementirajte tip podataka GooglePlay koji može čuvati sve podatke o aplikaciji iz datoteke **SPA\_PROJ\_007\_GOOGLE\_PLAY\_data.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 aplikaciji na ekran (naziv, kategorija, prosječna ocjena korisnika, tip).
2. (Bodova: **2**) Kad korisnik odabere opciju „Pretraga po kategoriji“, prvo prekopirajte sve aplikacije iz vektora u odgovarajući kontejner. Zatim pitajte korisnika da upiše kategoriju koju želi tražiti pa mu nakon toga ispišite podatke o aplikacijama u toj kategoriji ili napišite da ne postoje. 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 ocjeni“, koristeći prioritetni red ispišite aplikacije rastuće prema ocjeni (svojstvo „Rating“).
5. (Bodova: **3**) Kad korisnik odabere „Kopiranje u novi vektor“, kreirajte novi vektor i napravite sljedeće:

* Prekopirajte sve aplikacije u novi kontejner koji omogućuje da se pretraživanje po kategoriji odvija u logaritamskoj složenosti (birajte map, multimap, set ili multiset).
* Sve dok to korisnik želi, omogućite mu unos kategorije.
* Kad korisnik unese neku kategoriju, pronađite sve aplikacije iz te kategorije u vašem odabranom kontejneru i kopirajte ih u novi vektor.
* Ponavljajte sve dok to korisnik želi.
* Na kraju ispišite sve aplikacije iz novog vektora.

1. (Bodova: **3**) Kad korisnik odabere „Prikaz po kategoriji“, koristeći prioritetni red ispišite sve aplikaciju padajuće prema kategoriji. Sve aplikacije iz iste kategorije obavezno prikažite tako da prvo idu besplatne, a zatim one koje se plaćaju (svojstvo „Type“).
2. (Bodova: **1**) Dodajte u izbornik sljedeće opcije:

Usporedba sortiranja

Pogodi trajanje

Lets Bogo!

The Merge Dance

1. (Bodova: **3**) Kad korisnik odabere „Usporedba sortiranja“, napravite sljedeće:

* Kreirajte dva pomoćna vektora *v1* i *v2* i iskopirajte u svako od njih sve aplikacije.
* Uzmite s predavanja Selection i Insertion sortiranja te ih promijenite tako da sortiraju aplikaciju, padajuće po imenu.
* 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 pokemona u novi vektor. Isprogramirajte svoju vlastitu determinističku verziju bogo sorta te je iskoristite za slaganje pokemona po imenu 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 ocjeni

Vizualiziraj

Izrada stringa iz stringa

1. (Bodova: **2**) Kad korisnik odabere „Pretraga po ocjeni“, prvo prekopirajte aplikacije iz vektora u odgovarajući kontejner. Zatim pitajte korisnika da upiše minimalnu i maksimalnu ocjenu (svojstvo „Rating“) pa mu nakon toga ispišite podatke o svim aplikacijama čije su ocjene u tom rasponu. 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 aplikacije u unordered\_multimap tako da ključ bude njihov ID. Nakon toga, vizualizirajte *hash* tablicu iz unordered\_multimap tako da u tekstualnu datoteku iscrtate sve njene *buckete* te uz svaki *bucket* prikažete i aplikacije u njemu. Primjer početka ispisa (vrijednosti su izmišljene):

Bucket 0: American Muscle Car Race -- Uber Driver -- Box

Bucket 1: Viber Messenger -- Google Allo -- Learn C++ -- Toy Blast

Bucket 2: [EMPTY]

1. (Bodova: **6**) Kad korisnik odabere „Izrada imena iz imena“, omogućite mu da odabere dvije aplikacije te mu zatim, koristeći unordered\_map, unordered\_multimap, unordered\_set ili unordered\_multiset, ispišite može li se ime druge aplikacije kreirati koristeći raspoloživa slova iz imena prve aplikacije (pri tome mala i velika slova smatrajte jednakima). Primjerice:

* Ako je ime prve aplikacije „Frogger's Adventures: Temple of the Frog“, a druge „Ventures“, onda se ime druge može konstruirati iz imena prve.
* Ako je ime prve aplikacije „Lady Bullseye“, a druge „Lady Bulls Eye“, onda se ime druge ne može konstruirati iz prve jer nedostaje jedan razmak.