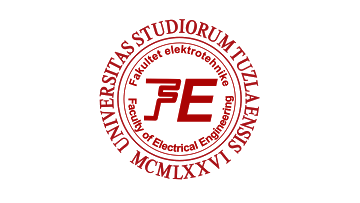
UNIVERZITET U TUZLI

FAKULTET ELEKTROTEHNIKE



Objektno-orijentirano programiranje

Zadaća 3

Tuzla, decembar 2021.

Sadržaj

[**Zadatak 1**](#_6rzze7bdpon4) **2**

[**Zadatak 2**](#_63qnptp0p8ug) **3**

[**Zadatak 3**](#_6i7t1fpwosz1) **3**

[**Zadatak 4**](#_rovf4i8vwf32) **4**

[**Zadatak 5**](#_9w89y9tvh1si) **4**

# 

# 

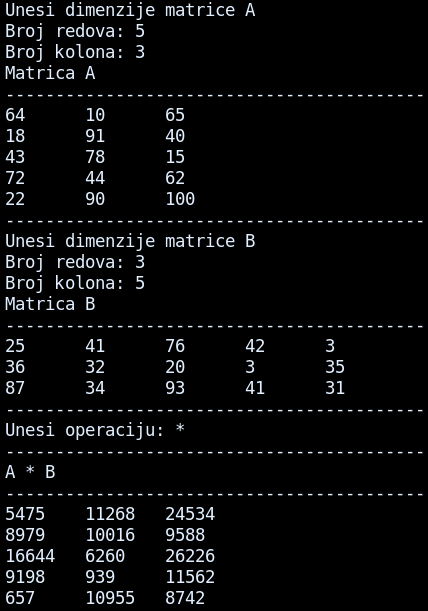
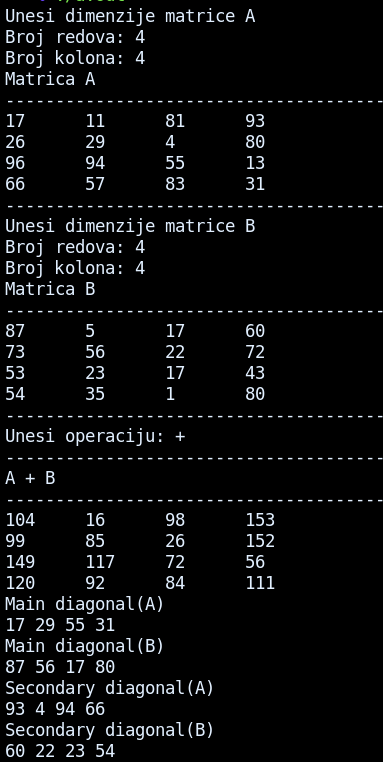
# Zadatak 1

Napisati program koji omogućava rad sa matricama. Program treba da omogući operacije sabiranja i množenja matrica, kao i pronalazak elemenata na glavnoj i sporednoj dijagonali.

Neophodno je da korisnik sa standardnog ulaza unese dimenzije za matrice. Elemente u matrici generisati korištenjem *rand()* funkcije u opsegu [0, 100]. Ukoliko nije moguće izvršiti neku od navedenih operacija (množenje, sabiranje ili pronalazak elemenata na glavnoj i sporednoj dijagonali generisati iznimku tipa std::logic\_error. Neophodno je koristit vise fajlova (deklaracije funkcija izdvojiti u zaglavlje pod imenom *Matrica.hpp*, definicije funkcija u *Matrica.cpp*). U fajlu *main.cpp* smije postojati samo *main()* funkcija u kojoj pozivamo funkcije:

* Generisanje matrica: *kreirajMatricu()*
* Sabiranje matrica: *saberiMatrice()*
* Mnozenje matrica: *pomnoziMatrice()*
* Pronalazak elemenata na glavnoj dijagonali: *pronadjiGlavnuDijagonalu()*
* Pronalazak elemenata na sporednoj dijagonali: *pronadjiSporednuDijagonalu()*
* Ispis matrice: *ispisiMatricu()*

Obratiti pažnju na način prosljeđivanja argumenata u funkciju.

Način izvršavanja programa je prikazan na narednoj slici:  


# Zadatak 2

Napisati funkciju saberi koja može primiti argumente proizvoljnih tipova. Za numeričke tipove ova funkcija vraća rezultat generiran korištenjem + operatora, dok za stringove i niz karaktera vrsi spajanje (konkatenacija). Funkcija može primati opcioni (treći argument) koji ako je proslijeđen množi rezultat sabiranja.   
**Primjer**:  
 *suma(1,2,3)* - rezultat ce biti (1+2)\*3.   
Za stringove omogućava višestruko spajanje na sljedeci nacin:  
(“abc”,”d”, 3) = “abcdabcdabcd”.  
Treci defaultni argument ako nije proslijedjen treba da ima vrijednost 1.

# Zadatak 3

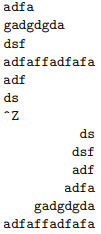
Napisati funkciju **transformiraj** koja uzima dva vektora realnih brojeva, te dodatni parametar, lambda izraz koji definira transformaciju. Ukoliko su vektori iste dužine, funkcija transformira elemente prvog vektora aplicirajući operaciju definiranu lambda izrazom na odgovarajuce elemente oba vektora. U slučaju da se dužine vektora razlikuju, funkcija treba da generira iznimku tipa **domain\_error**. Funkcija ne vraca nista. Izabrati odgovarajuci nacin prosljeđivanja parametara u funkciju. Napisati i odgovarajuci program kojim se testira kreirana funkcija.

# Zadatak 4

Definirati zaglavlja i implementaciju za sljedeće module:

* IOrijeci.hpp Modul za unos i ispis vektora string-ova. Modul treba da ima dvije funkcije jednu za unos drugu za ispis vektora.
* dodavanje.hpp Modul koji sadrži dvije funkcije. Prva verzija funkcije uzima string a, karakter b i cijeli broj c. Funkcija dodaje c karaktera b na lijevoj strani stringa a. Druga verzija funkcije uzima string a i cijeli broj c. Funkcija dodaje c karaktera ’ ’ na lijevoj strani stringa a. Koristiti overloading ulaznih parametara.
* MAXrijec.hpp Modul sadrži jednu funkciju koja nalazi riječ unutar vektora riječi koja ima maksimalnu dužinu. Funkcija vraća pronađenu riječ.

Za sve funkcije u gornjim modulima odabrati najadekvatniji način za prosljeđivanje parametara. Napisati glavni program koji koristeći gore navedene module čita niz riječi a zatim ispisuje učitani niz sortiran po dužini riječi i poravnat na desnu stranu kao u sljedećem primjeru pozivanja programa.



# Zadatak 5

Napisati program koji omogućava kreiranje proizvoljnog broja objekata tipa *Radnik*.

Broj radnika koje treba kreirati je neophodno unijeti sa ulaza.

*Radnik* je struktura koja u sebi ima sljedeće članove:

* *Ime*
* *Prezime*
* *Godina rođenja*
* *Mjesto stanovanja*
* *Iznos plate*

Sve kreirane radnike je neophodno spremiti u *vektor*.  
Napisati funkcije za:

* Unos radnika: *unos()*
* Ispis radnika: *ispis()*

Odabrati pravi način za proslijeđivanje argumenata u funkciju. Nakon završenog unosa radnika, neophodno je izvršiti sortiranje radnika na način opisan u nastavku:

* Po plati: Radnik sa najvećom platom treba biti na početku vektora, radnik sa najmanjom platom na kraju.
* Ukoliko postoji više radnika sa jednakom platom, kao dodatni kriterij za sortiranje treba da bude na osnovu imena tako da imena radnika budu sortirana alfabetski.
* Ukoliko postoji više radnika sa jednakom platom i istim imenom, radnike sortirati po godini rođenja tako da radnik koji je stariji bude ispred.

Strukturu *Radnik* neophodno je kreirati u zaglavlju pod nazivom **Radnik.hpp**. Funkcije za unos i ispis radnika definirati u cpp fajlu pod imenom **Radnik.cpp**. U main.cpp fajlu smije postojati **samo** main() funkcija u kojoj koristimo funkcije za unos i ispis radnika.

# 

# Zadatak 6

Napisati funkciju koja prima std::vector<int> kao prvi argument, te vektor funkcija koje će biti primijenjene nad svakim elementom u vektoru. Ukoliko element ne zadovolji kriterije svih predicate funkcija, u tom slučaju taj element zamijeniti sa defaultnim trećim argumentom kojeg funkcija prima koji ima vrijednost -1. Ukoliko bilo koja predicate funkcija vrati false, taj element zamijeniti sa defaultnim argumentom.

Funkcija ima sljedeci potpis:  
