UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE


# Strukture podataka

## Stack/queue

Zadaća 4

Tuzla, maj/svibanj 2023.

# Sadržaj

# Zadatak 1:

Implementirati postfix kalkulator sa cijelim brojevima i sa osnovnim operacijama (+,-,*,/). Postfiksna notacija ili postfiksni sistem oznaka je matematička notacija u kojoj svaki operator slijedi nakon svih svojih operanada, te je poznatija kao obrnuta poljska notacija (ili samo RPN od eng. *Reverse Polish notation*). Objašnjenje i upute za implementaciju ovog kalkulatora možete pronaći na http://hr.wikipedia.org/wiki/Obrnuta_poljska_notacija.

Ovu implementaciju realizovati u klasi *PostfixCalc*. Implementirati metode klase, kao i način korištenja prema svojoj intuiciji.
Unos operanada i operatora se vrši preko standardnog ulaza. Argumenti na ulazu su razmaknuti praznim mjestom. Operandi (brojevi) mogu biti pozitivni i negativni brojevi, dok su osnovne operacije jedan karakter.
Potrebno je implementirati čitanje ulaza na način da svaka linija ulaza predstavlja jedan matematički izraz za koju se ispisuje rezultat izraza ili informacija o grešci, nakon čega korisnik unosi narednu liniju ulaza, koja je novi izraz.

Zadatak možete riješiti korištenjem kontejnera iz standardne biblioteke.

# Zadatak 2:

Haris nastavlja raditi na svom novom programskom jeziku, boltić. Program u boltiću može sadržavati samo zagrade svih vrsta:

- uglaste zagrade < >,
- oble zagrade ( ),
- uglaste zagrade [ ],
- vitičaste zagrade { }.

Program koji sadrži samo zagrade, da bi se uspješno kompajlirao, mora
imati sve zagrade pravilno umetnute. Da bi izraz bio pravilan, svaka otvorena zagrada mora imati svoju uparenu zatvorenu zagradu, te se mora poštovati hijerarhija zagrada.

Na žalost, kompajler koji Haris pravi još uvijek ne može reći koji izraz u program ima pogrešno umetnute zagrade već jednostavno ne prihvata nepravilan program. Pomozite Harisu da napravi program koji će odrediti koji izraz nije pravilan.

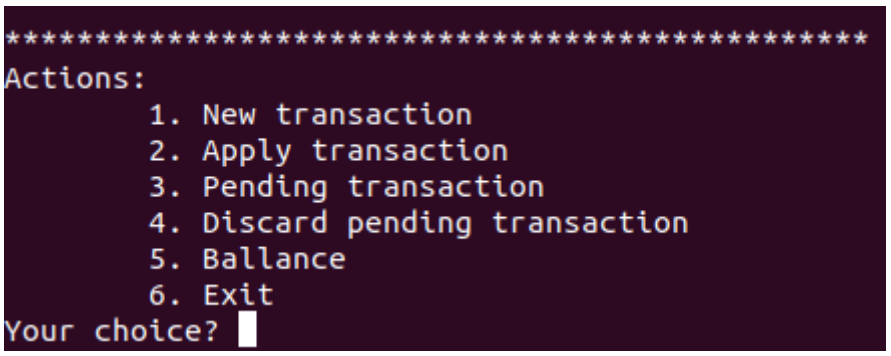
Zagrade poštuju hijerarhiju:
uglaste zagrade **<>** su najviše unutrašnje zagrade, pa onda oble **()**, pa onda uglaste **[]** pa onda vitičaste **{}**. Drugim riječima, vitičasta zagrada ne može biti otvorena unutar nekog drugog tipa zagrada. Uglaste zagrade **[]** ne mogu biti otvorene unutar oblih **()** ili uglastih **<>** zagrada, ali može biti otvorena unutar vitičastih zagrada. Obla zagrada ne može biti otvorena unutar para uglastih zagrada **<>**.

Potrebno je učitavati linije sa standardnog ulaza, gdje svaka linije predstavlja jedan izraz. Karakteri na ulazu su neki od '[ ] ( ) { } < >'. Na izlazu treba ispisati samo „dobar" ili „pogrešan". Program se završava tako što pročitamo sve karaktere sa ulaza, odnosno kad korisnik završi unos slanjem EOF karaktera (ctrl+d na Linuxu ili ctrl+z na Windowsu).

Zadatak možete riješiti korištenjem kontejnera iz standardne biblioteke.

# Zadatak 3:

Napisati program koji predstavlja upravljanje bankovnim računom. Prilikom pokretanja programa, korisniku se ispisuje sljedeći meni:




- Dodavanje nove transakcije ne vrši promjenu stanja računa. Transakcija se sprema za naknadno apliciranje te u sistemu može postojati beskonačno transakcija koje čekaju na apliciranje.
- Transakcije se apliciraju poštujući redoslijed dolaska transakcija.
- Korisnik ima opciju pregleda transakcije koja je spremna za apliciranje, odbacivanje naredne transakcije te ispisa trenutnog stanja računa
- Program mora voditi računa o graničnim slučajevima kao što je odabir opcije za apliciranje transakcije kada u sistemu ne postoje transakcije.
- Jedna transakcija je opisana pomoću vrijednosti koja se dodaje/uklanja sa računa, datum i vrijeme.

Zadatak možete riješiti korištenjem kontejnera iz standardne biblioteke.

# Zadatak 4:

Implementirati generičku data strukturu queue/red, koja je implementirana koristeći pristup cirkularnog buffera koji je rađen na predavanju.

Metode koje kontejner treba podržavati su sljedeće:

- konstruktor koji prima maksimalni broj elemenata koji se može smjestiti u cirkularni buffer,
- standardni skup konstruktora i destruktora,
- metod push koji dodaje element na kraj reda. Potrebno je implementirati ovaj metod tako da zna prepoznati move semantiku. Ukoliko nije moguće dodati element u red, potrebno je baciti iznimku tipa out_of_range.
- metod pop koja vraća prvi element iz reda po vrijednosti, i uklanja ga iz reda,
- metod front i back, zajedno sa const overloadima,
- metode size, capacity i empty.

Zajedno sa data strukturom predajete i main.cpp file koji demonstrira korištenje metoda iz zadatka.