**SLAGALICA – MOJ BROJ**

**Predmet: Objektno orijentisano programiranje 2**

**Autor: Nemanja Stjepanović SV-75/2021**

Spisak klasa

**Calculator**

Atributi klase: /

Funkcije klase:

* InfixToRPN
* racunajRPN
* prvenstvo
* jeloperacija
* racunanje
* rucniDigit
* jelDobarRPN

Greške:

* Problem sa zagradom
* Unet pogrešan karakter
* Izraz nije dobar

Opis funkcija:

**InfixToRPN**

* Funkcija prima korisnikov uneti izraz i pretvara ga u RPN notaciju i na kraju vraća vector stringova u kojima se nalazi svaki broj i operacija korisnikovog izraza. Prolazi se kroz svaki karakter u stringu i na osnovu toga da li je broj ili operacija smešta se na odgovarajući stek(vector) i kada se dođe do sledeće operacije sa steka se popuje operacija i smešta na izlazni stek na kom su već brojevi. Ako je karakter zatvorena zagrada,pushuju se sve operacije na izlazni stek dok se ne dođe do zatvorene zagrade.U koliko se ne dođe do zatvorene zagrade program baca grešku.Program takođe baca grešku ako neki od karaktera nije broj ili operacija.Na kraju se samo pushuje sve sto je ostalo na stekovima brojeva i operacija,a baca se greška ako se nađe otvorena zagrada.

**racunajRPN**

* Funkcija prima vector stringova(ili stek) u kom je izraz u RPN notaciji, i računa ga tako što skida brojeve sa steka dok ne dođe do operacije,onda izračuna i pushuje nazad na stek sve dok ne dođe do kraja.Na steku bi trebao da ostane samo 1 broj koji predstavlja rezultat,a program baca grešku Greška kod računanja u koliko se na steku nalazi više od 1 vrednosti.

**prvenstvo**

* Funkcija prima char(operaciju) i proverava koja operacija ima prednost,to jest prioritet i vraća 1 ako su \* ili / ,u ostalim slučajevima 0.

**jeloperacija**

* Funkcija prima karakter i proverava da li je on aritmetička operacija

**racunanje**

* Funkcija prima 2 broja i operaciju i vraća njihov rezultat u zavisnosti od operacije.
* Baca grešku Deljenje sa nulom/nije int ako je drugi broj 0 ili ako deljenje 2 broja nije celobrojna vrednost.

**jelDobarRPN**

* Funkcija prima vektor brojeva u RPN notaciji i proverava da li je on ispravan tako što dodaje 2 pa oduzme 1 ako je operacija,ili dodaje 1 ako je broj, ako vrednost na kraju nije 1 vraća false,inače true.

**rucniDigit**

* Funkcija proverava da li su u stringu samo brojevi i vraća true ako jesu, u suprotnom false.

**Broj**

Atributi klase:

* brojevi - čuva vector brojeva za pronalaženje rešenja
* resenje - broj koji se traži
* na\_potezu - koji igrač prvi unosi izraz

Funkcije klase: /

Greske: /

**Runda**

Atributi klase:

* IzrazA - izraz koji je uneo korisnik A
* IzrazB
* ResenjeA - broj koji je dobio korisnik A
* ResenjeB
* pobednik - pobednik runde
* odstupanjeA - koliko broj korisnika A odstupa od rešenja
* odstupanjeB
* racunarIzraz - izraz koji je našao računar
* brojevi - brojevi koji su mogli da se koriste za izraz
* trazeni - broj koji se traži

Funkcije klase:/

Greske: /

Spisak slobodnih funkcija

**unos\_fajla**

* Funkcija proverava da li je korisnik uneo postojeći fajl koji sadrži brojeve za svaku rundu,ako nije ponavlja unos a ako jeste vraća string naziva fajla.

**ucitaj\_fajl**

* Funkcija prima kao parametar string koji sadrži ime fajla iz kog treba da se učitavaju podaci,i kroz ifstream učitava sve brojeve iz fajla koji je korisnik uneo, zatim pravi objekte klase Broj koji se smestaju u vektor koji funkcija na kraju vraća.

**racunaj**

* Funkcija prima dva broja i operaciju, i vrsi određenu aritmetičku operaciju nad ta 2 broja i vraća rezultat te operacije.
* Funkcija baca grešku Deljenje sa nulom u koliko je drugi broj 0.

**s**

* Prima double vrednost koju pretvara u string otklanjajući decimalni deo.

**rs**

* Prima string izraz koji računa pomoću kalkulatora pretvaranjem u RPN notaciju i daljim računanjem i vraća double vrednost.

**rs\_naj**

* Prima vektor u kom se nalazi rešenje koje je računar pronašao za određeni traženi broj i računa ga,vraća double vrednost.

**pronadji\_rezultat**

* Funkcija je objašnjena u opisu algoritma za traženje

**pronadji\_resenje**

* Funckija koja prima vektor brojeva koje korisnik može da koristi za izraz kao i broj koji se trazi, i referencu na string izraz. U funkciji se poziva prethodna funkcija pronadji rezultat u kojoj računar pronalazi izraz za traženi broj i u zavisnosti od njenog rezultata računa i ispisuje na ekran izraz koji je računar pronašao,ili u koliko nije ispisuje izraz najbližeg broja

**printajBrojeve**

* Funkcija printuje na konzolu brojeve koje korisnik sme da koristi prilikom sastavljanja svog izraza.

**provera**

* Proverava da li se određeni broj nalazi u vektoru i služi kako bi se proverio izraz koji je korisnik uneo.

**ispravni\_brojevi**

* Funkcija prima ponudjene brojeve i korisnikov uneti izraz i proverava da li je korisnik uneo samo brojeve koji su ponudjeni uz pomoc funkcije

provera.

**korisnikIzraz**

* Funkcija proverava izraz koji je korisnik uneo tako što pokušava da ga pretvori u RPN notaciju i izračuna,u koliko je korisnik uneo loš izraz ili uneo brojeve koji nisu ponuđeni catch ce uhvatiti grešku i korisnik mora ponovo da unese novi izraz.Funkcija vraća broj koji je korisnik dobio, kao i izraz koji je uneo.

**koJePobednik**

* Funkcija prima brojeve koje su korisnici dobili i koliko oni odstupaju od rešenja, pa na osnovnu toga se ispisuje na konzoli ko je pobedio rundu.

**upis\_rezultata**

* Funkcija prima broj osvojenih rundi svakog igrača,i vektor koji sadrži sve runde. Pomoću ofstreama upisuje u fajl rezultati.txt informacije o svakoj rundi i ko je na kraju pobedio igru.

Strukture datoteka

Struktura ulazne datoteke

10 9 1 5 4 6 987

1 2 3 4 25 75 755

5 9 7 9 2 4 804

Prvih šest brojeva su brojevi koji se mogu koristiti za stastavljanje izraza,dok je sedmi broj

Struktura izlazne datoteke

Broj runde: 1

Brojevi na raspolaganju: 10 9 1 5 4 6

Trazeni broj: 987

Igrac A:

Dobijeni broj: 987

Odstupanje: 0

Izraz: ((((5+6)\*(10\*9))+1)-4)

Igrac B:

Dobijeni broj: 9999

Odstupanje: 999

Izraz: Nema izraza

Pobednik runde: Igrac A

Racunar: 987 = ((((5+6)\*(10\*9))+1)-4)

Opis algoritma za pronalaženje rešenja

Algoritam koji je korišćen za pronalaćenje rešenja je brute – force algoritam, koji isprobava sve moguće kombinacije brojeva i operacija dok ne dođe do traženog broja.

Za pronalaženje rešenja koristi se funkcija **pronadji\_rezulat** koja prima vector stringova(brojeva),traženi broj, i reference na 2 vektora u kojima će se nalaziti izrazi za traženi broj u koliko postoji i izraz za najbliži broj traženom.

Algoritam radi na principu da se iz vektora brojeva uzimaju po 2 broja i da se svaka operacija primeni na njima. Nakon što se operacija izvrši,pravi se novi vektor iz kog se uklanjaju prve 2 uzete vrednosti i dodaje broj koji se dobio od ta 2 broja primenom neke od operacija.

U koliko je operacija deljenje, a drugi broj 0 ili 1, ili u koliko rezultat deljenja 2 broja nije integer odnostno celobrojna vrednost prelazi se na sledeću operaciju (continue).

Ako je operacija oduzimanje a drugi broj veći od prvog opet se odmah prelazi na sledeću operaciju sa continue.

Takođe i za množenje,ako je neki od brojeva 1 radi se continue.

Kada su zadovoljeni svi uslovi,radi se operacija nad 2 broja, te se sastavlja string od njihovog rešenja i pushuje se na novi vektor stringova,u koliko rešenje nije 0, koji ne sadrži 2 broja nad kojima smo vršili operaciju (npr. pushovace se (5+5) a ne 10).

Na primer u koliko imamo brojeve 1 2 3 4 5 6, novi vektor će nakon prve operacije izgledati (1+2) 3 4 5 6.

Nakon svake operacije vrši se provera da li je dobijeni rezultat traženi broj,a ako jeste funkcija će vratiti true. Takođe se i vrši provera u koliko je rezultat bliži od prethodnog najbližeg broja i on se menja.

Posle pushovanja novog broja u vektor, funkcija se zove rekurzivno sa istim parametrima, samo će se sada zvati nad novim vektorom brojeva koji smo napravili, sve dok njegova veličina nije 1 (ako je veličina 1 znači da je ostao samo 1 broj i on može biti naše rešenje). Ako je taj broj naše rešenje funkcija će vratiti true i prekinuće rekurzije, a ako nije nastavlja se dalje isprobavanje kombinacija sve dok se ne isprobaju sve kombinacije ili se ne dođe do tačnog rešenja. Ako funkcija ne nađe tačno rešenje vraća false.

Opis načina testiranja

Testiranje kalkulatora

Ispravnost kalkulatora testira se pomoću funkcije **kalkulator\_test** u kojoj se nalaze 3 unapred zadata stringa sa izrazima za koje već znamo rešenje.

Testiranje se vrši tako što proveravamo rešenje koje nam je kalkulator vratio sa našim unapred poznatim rešenjem i ako su oni jednaki test je uspešan,odnosno naš kalkulator je ispravan.

Testiranje rešenja računara

Nakon što smo utvrdili da je kalkulator ispravan, testiramo i izraze koje nam je računar pronašao. Testiramo pomoću funkcije **racunar\_resenje\_test** u kojoj se uz pomoć funkcije **vrati\_brojeve** nalazi vektor koji sadrzi vektore brojeva za 3 runde. Za svaki proveravamo da li je naš traženi broj jednak sa brojem koji je naš kalkulator vratio računajući izraz koji je računar našao. Ako su brojevi jednaki, znači da je računar pronašao dobar izraz i da algoritam za pronalaženje najboljeg rešenja radi.

Rad U/I podsistema

Učitavanje fajla

Nakon što je prosleđen argument za ulazni fajl, pomoću funkcije **ucitaj\_fajl** iz ulaznog fajla učitavaju se brojevi.

Učitavanje se vrši pomoću ifstreama, tako što će se kroz while petlju čitati linije sve dok se ne dođe do kraja fajla. Parsiranjem linije u kojoj su podaci razdvojeni sa razmakom, svi brojevi smeštaju se u double vrednosti, a zatim pushuju u vektor brojeva.

Za svaku liniju formira se objekat klase **Broj** koji će čuvati naše brojeve,rešenje koje se traži kao I koji igrač je na potezu. Koji igrač je na potezu određujemo tako što za svaku neparnu liniju(svaki neparan unos) prvi na potezu je igrač A,a za svaki paran igrač B. Funkcija će na kraju vratiti vector u kom se nalaze podaci za početak svake runde.

Upisivanje u fajl

Kada se završe sve runde, vrši se upisivanje podataka o svakoj rundi u izlazni fajl rezultati.txt funkcijom **upis\_rezultata** koja prima broj osvojenih rundi za svakog igrača i vector svih rundi. Na kraju svake runde formira se objekat klase **Runda** koji sadrži sve informacije I dodaje se u vector rundi koji se prosleđuje u ovu funkciju.

.Upisivanje u fajl vrši se kroz ofstream , gde se prolazi kroz vector rundi I za svaku rundu se upisuju njene informacije u fajl.

Struktura argumenata komandne linije

Struktura

* int main(int argc, char\*\* argv) {}

Komandna linija prima naziv fajla iz kog će se učitati podaci,proverava se ako je broj unetih argumenata(argc) 2 i ako jeste uzima se drugi argument(argv[1]) koji predstavlja naziv tog fajla i prosleđuje se u funkciju za učitavanje **ucitaj\_fajl.**

Problemi i ograničenja

Jedan od problema je korišćenje brute – force algoritma za pronalaženje rešenja. U koliko nam nije bitno vreme rešavanja problema onda je dobro koristiti ovaj algoritam jer on prolazi kroz sve moguće slučajeve. Mana ovog algoritma je ta što pronalaženje rešenja može da potraje dosta dugo ako se algoritam ne optimizuje kako treba. Za razliku od drugih algoritama ovaj algoritam ima najveću šansu da pronađe odgovarajuće rešenje, ali u koliko nam je vreme bitan faktor, bolje je koristiti neki drugi algoritam.