



Univerzitet u Novom Sadu
Fakultet tehničkih nauka



Dokumentacija za individualni projekat

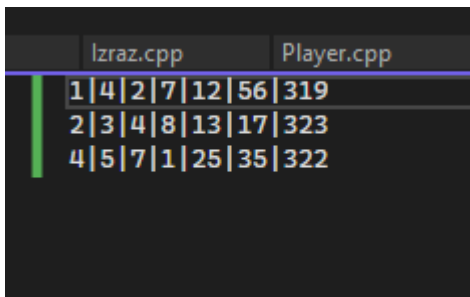
Student: Živanović Marija, SV 19/2021
Predmet: Objektno orijentisano programiranje 2
Tema projektnog zadatka: Kviz “Slagalice”, igra “Moj broj”

SADRŽAJ

- Rad U/I podsistema (učitavanje i ispis/upis)
- Spisak svih klasa, izuzetaka i slobodnih funkcija i njihova objašnjenja
- Objašnjenje najbitnijih atributa, klasa, funkcija članica, slobodnih funkcija i izuzetaka
- Argumenti komandne linije
- Strukture ulazne i izlazne datoteke
- Opis algoritma za pronalaženje tačnog rešenja
- Opis načina testiranja
- Uočeni problemi i ograničenja

Rad U/I podsistema (učitavanje i ispis/upis)

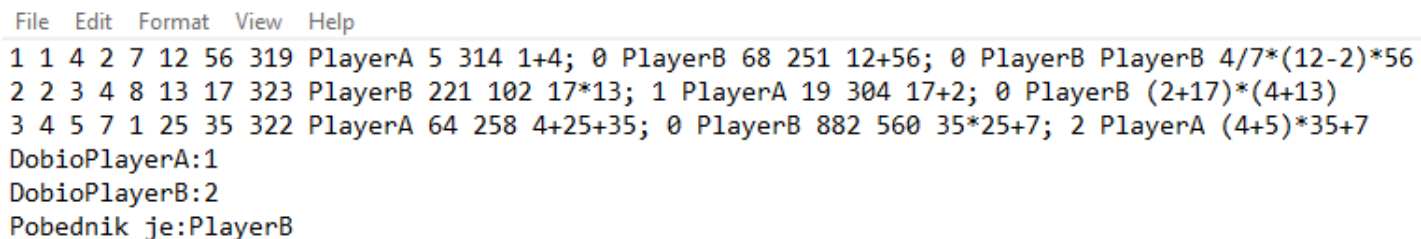
Ulazna datoteka (igra1.txt) je organizovana na način prikazan na slici ispod (u svakom redu postoji 7 brojeva razvojenih uspravnom linijom kao delimiterom, prvih 6 brojeva su brojevi koji su na raspolaganju da se pomoću njih dobije sedmi broj- broj koji je trocifreni).



```
Izraz.cpp  Player.cpp
1|4|2|7|12|56|319
2|3|4|8|13|17|323
4|5|7|1|25|35|322
```

Izlazna datoteka (Rezultat.txt) je struktuirana kao što je prikazano na slici ispod. U jednom redu su dati podaci o jednoj rundi. Prvi broj predstavlja broj runde, sledeci brojevi su brojevi koji su bili na raspolaganju za racunanje i trazeni trocifreni broj, za koliko broj koji je nasao igrac odstupa od trazenog, zatim trocifreni broj koji je dobio 1. igrac, postupak, tj. izraz kojim je dosao do njega. Na kraju se stampa koliko dobijenih rundi ima PlayerA, a koliko PlayerB i ko je konacni pobednik (ili da li je mozda nereseno).

 Rezultat - Notepad



```
File Edit Format View Help
1 1 4 2 7 12 56 319 PlayerA 5 314 1+4; 0 PlayerB 68 251 12+56; 0 PlayerB PlayerB 4/7*(12-2)*56
2 2 3 4 8 13 17 323 PlayerB 221 102 17*13; 1 PlayerA 19 304 17+2; 0 PlayerB (2+17)*(4+13)
3 4 5 7 1 25 35 322 PlayerA 64 258 4+25+35; 0 PlayerB 882 560 35*25+7; 2 PlayerA (4+5)*35+7
DobioPlayerA:1
DobioPlayerB:2
Pobednik je:PlayerB
```

Učitavanje iz fajla obavlja se u funkciji `read_file()`, a zatim se za svaku pročitanu liniju poziva funkcija `split_string()` koja će pročitanu liniju podeliti po delimitere (u mom slučaju uspravna linija) i iz nje izdvojiti POSTAVKU ZADATAKA: ponuđene brojeve i traženi rezultat (trocifreni broj).

```
// Učitavanje iz ulaznog fajla
vector<Zadatak*> read_file(string file_name) {
    string line;
    ifstream myfile(file_name);
    vector<Zadatak*> zadaci;
    if (myfile.is_open())
    {
        while (getline(myfile, line))
        {
            if (line.empty()) {
                continue;
            }
            Zadatak* zadatak = split_string(line);
            zadaci.push_back(zadatak);
        }
        myfile.close();
    }
    return zadaci;
}
```

Nakon toga, poziva se funkcija `game()` koja igra igru. Tu se korisniku daje input u kom on unosi svoj izraz za traženo rešenje. Neparne runde igra A, parne B. Prvo igra A, ako dobije tačno rešenje, program učitava sledeću postavku zadatka, ukoliko ne dobije tačno rešenje, sansu dobija i Igrac B, nakon toga racunar printa konačno rešenje. Isto vazi i za rundu koju zapocinje PlayerB. Svi podaci o tome šta su unosili igrači A i B kao i kakva rešenja su dobili se upisuju u ispisni fajl `Rezultat.txt` (ali nakon završenih svih rundi) funkcijom `write_file()`.

Na kraju se stampa i konačni pobednik. Ukoliko korisnici budu unosili neka nevalidna rešenja (poput deljenja sa nulom, koriscenja brojeva koji nisu ponudjeni u postavci zadatka i slično → program će im odstampati poruku o neispravnom unosu i pogresnom rešenju; igra se nastavlja dalje).

Spisak svih klasa, izuzetaka i slobodnih funkcija i njihova objašnjenja

Spisak korišćenih klasa:

- Generator
- Player
- Izraz
- Odgovor
- Runda
- Token_stream
- Zadatak

U klasi **Generator** nalaze se polja a, cSlots, cValues, nextInd. Polje a je tipa pokazivač na int i njega koristimo u destrukturu. Polje cSlots je tipa int i predstavlja mesta u vektoru, cValues vrednosti koje će se nalaziti u vektoru, a nextInd ćemo koristiti za prelazak na generisanje sledećeg izraza.

U klasi Generator se nalaze i sledeće funkcije:

```
** doNext  
** generateVector
```

Funkcija doNext će vršiti generisanje izraza kroz koje će računar prolaziti dok bude tražio najadekvatniji izraz za zadato rešenje, a u funkciji generateVector ćemo generisane vrednosti sakupljati u vektore sa odgovarajućim brojem pozicija koji je ograničen vrednošću polja cSlots.

Klasa Odgovor ima polja:

- igrac koje je tipa pokazivač na Player-a
- izraz koje je tipa pokazivač na Izraz tipa int

Klasa Odgovor ima sledeće funkcije:

- getIgrac
- setIgrac
- getIzraz
- setIzraz

Funkcija `getIgrac` ima povratnu vrednost tipa pokazivač na `Player`-a i vrši dobavljanje o podacima igrača koji je dao određeni odgovor. Funkcija `setIgrac` je tipa `void` vrši postavljanje vrednosti pokazivača na `Player`-a. Funkcija `getIzraz` vrši dobavljanje Izraza tipa `int`, a funkcija `setIzraz` vrši postavljanje vrednosti pokazivača na izraz koji je određeni igrač dao kao svoj odgovor.

Klasa `Player` ima sledeća polja:

- `id` koje je tipa `int` i predstavlja ID igrača;
- `username` koje je tipa `string` koje predstavlja korisničko ime igrača;
- `br_dobijenih_rundi` tipa `int` koje predstavlja broj rundi koje je igrač osvojio.

Klasa `Player` ima sledeće funkcije:

- `getId` sa povratnom vrednošću tipa `int` koja vrši dobavljanje ID-ja igrača.
- `setId` koja je tipa `void` i vrši postavljanje ID-ja igrača.
- `getUsername` koja vrši dobavljanje korisničkog imena igrača i ima povratnu vrednost

tipa `string`.

- `setUsername` koja vrši postavljanje korisničkog imena i koja je tipa `void`.
- `getBrDobijenih` koja vrši dobavljanje broja dobijenih rundi određenog igrača
- `setBrDobijenih` koja vrši postavljanje broja dobijenih rundi određenog igrača.

Klasa `Runda` ima sledeća polja:

- `zadatak` koje je tipa pokazivač na `zadatak` i predstavlja pokazivač na `zadatak` koji se

rešava u određenoj rundi.

- `odgovor1` koje je tipa pokazivač na odgovor i predstavlja pokazivač na odgovor koji je

dao prvi igrač u ovoj rundi.

- `odgovor2` koje je tipa pokazivač na odgovor i predstavlja pokazivač na odgovor koji je

dao drugi igrač u ovoj rundi.

Klasa `Runda` ima i sledeće funkcije:

- `getZadatak()`
- `setZadatak()`
- `getOdgovor1()`
- `setOdgovor1()`
- `getOdgovor2()`
- `setOdgovor2()`

Klasa Token_stream ima funkciju get() kojom dobavlja tokene koji predstavljaju ili karakter u vidu nekog operanda ((,), *, /, +, -) ili broj koji se upisuje kao tip double. Klasa Token_stream ima polja kind tipa char i polje value tipa double u koje čuva vrednost broja.

Klasa Zadatak ima polja:

- tacan tipa int
- numbersPermutations (broj permutacija) koji je vektor koji čuva brojeve tipa int
- generatedExpression (generisani niz) tipa string

Klasa Zadatak ima funkcije:

- getTacan()
- setTacan()
- printNumbers() - koja štampa POSTAVKU ZADATAKA
- funkciju izracunaj() kojom racunar dobija svoj izraz koristeći permutacije sa ponavljanjem
- funkciju izracunajPriblizno() ---> koja se poziva ukoliko racunar nije uspeo da nađe tačno rešenje, pa se traži najpribližnije moguće
- funkciju pronadjenaVrednost() --> u kojima je opisan rad sa operandima većeg prioriteta & i ^

U template klasi Izraz nalaze se polja izraz, vrednost i ts. Tip polja izraz je string i u njemu će biti smešten zapis izraza, polje vrednost je tipa T, pošto je klasa Izraz template klasa, ovom polju se mogu prosledivati vrednosti nekog od tipova brojčanih vrednosti - int, float ili double. Polje ts je tipa Token_stream i ono će nam omogućiti lakše prihvatanje unosa korisničkog izraza sa tastature.

U klasi Izraz nalaze se i sledeće funkcije:

- getIzraz
- setIzraz
- getVrednost
- setVrednost
- primary
- term
- expression

Funkcija `getIzraz` ima povratnu vrednost tipa `string` i vrši dobavljanje zapisa izraza. Funkcija `setIzraz` je tipa `void`, odnosno nema povratnu vrednost i vrši postavljanje zapisa odgovarajućeg izraza u polje `izraz`. Funkcija `getVrednost` ima povratnu vrednost tipa `T` zbog toga što je klasa `Izraz` template klasa i vrši dobavljanje vrednosti tipa `T` (što može biti `double`, `int` ili `float`) iz polja `vrednost`. Funkcija `setVrednost` je tipa `void` i vrši postavljanje vrednosti tipa `T` u polje `vrednost`. Funkcije `primary`, `term` i `expression` zajedno obavljaju dobavljanje izraza iz korisničkog unosa. Funkcija `primary` započinje dobavljanjem tokena iz `Token_stream`-a. U zavisnosti od vrste tokena, da li je u pitanju `(`, `)`, `-`, broj ili neki neodgovarajući unos, funkcija će dati određenu reakciju. U slučaju da je naišla na `(`, funkcija `primary` će rekurzivno pozvati funkciju `expression` i tu vrednost će smestiti u promenljivu `dtipa T`. Ako je u pitanju `-`, izvršiće se negiranje povratne vrednosti funkcije `primary` i funkcija će to vratiti. U slučaju da je naišla na broj, funkcija će vratiti njegovu vrednost. U slučaju da je naišla na neki neodgovarajući izraz (npr. slova), funkcija će baciti grešku i ispisati odgovarajuću poruku. Funkcija `term` će započeti rekurzivnim pozivom funkcije `primary` i tu povratnu vrednost će smestiti u promenljivu `left`. Zatim vršimo dobavljanje tokena i u zavisnosti od njegove vrste, funkcija će imati određenu reakciju. Ako je u pitanju `*`, izvršiće se množenje vrednosti promenljive tipa `left`, u slučaju da je `/`, izvršiće se deljenje vrednosti promenljive `left`.

U funkciji `expression` imamo rekurzivni poziv funkcije `term` i vršimo sabiranje i oduzimanje vrednosti `left` u zavisnosti od toga da li je token vrste `+` ili `-`.

Argumenti komandne linije

Parametar koji se prosleđuje preko argumenta komandne linije je zaiv fajla (imeFajla u mom slucaju je to igra1.txt).

General	Local Windows Debugger	
Advanced		
Debugging		
VC++ Directories	Command	\$(TargetPath)
▸ C/C++	Command Arguments	igra1.txt
▸ Linker	Working Directory	\$(ProjectDir)

Opis algoritma za pronalaženje tačnog rešenja

Računar pomoću funkcija `izracunaj()` i `izracunajPriblizno` dobija svoj rezultat koji se pamti u promenljivu `calculated`. `Izracunaj` i `izracunajPriblizno` radi tako što prolazi kroz sve permutacije sa ponavljanjem i operande sa prioritetoj menja sa `+` ili `-` pri čemu oko brojeva postavlja zagrade—> odvajaja tačno odredjen broj mesta ukoliko su brojevi dvocifreni ili jednocifreni.

Zatim se u mainu kada korisnik unese svoj izraz, dati sracunati izraz upoređuje sa traženom vrednošću. Ukoliko je ona tačna tu rundu je dobio igrač koji je odigrao tu kombinaciju brojeva, ukoliko nije igra sledeći igrač i tek posto oba igrača odigraju se štampa izraz koji je pronašao računara. U mainu se pozivaju i validacije za proveru korisničkog unosa koji svaki operator zamenjuju praznim stringom i prolaze kroz brojeve koje je korisnik uneo koji se čuvaju u vektoru tražeci ih u brojevima postavke zadatke pri tom računajući i njihov broj ponavljanja takodje radi validacije.

Uočeni problemi i ograničenja

Uglavnom nisu uočeni nikakvi problemi i ograničenja. U sprezi sa korisnikom prilikom neipravnih unosa program javlja grešku i igra se nastavlja. Jedino ograničenje može biti vreme koje je potrebno računaru da iščita fajl i sam izračuna rešenje pre nego što korisniku u konzoli isprinta postavku zadatka, kao i određeno vreme koje je potrebno da se svi podaci o igračima i rundi upišu u izlazni fajl nakon čega se program završava.