PRIRODNO-MATEMATIČKI FAKULTET

UNIVERZITET SARAJEVO

# Tema 2

PREDMET: STRUKTURE PODATAKA I ALGORITMI

SMJER: STRUČNI STUDIJ „INFORMACIONE TEHNOLOGIJE“

STUDENT: IDRIZOVIĆ ERDIN

INDEKS: 118/IT-19

## Uvod

Tema projekta je izrada generičke strukture podataka koja čuva elemente kao binarno stablo pretrage. Potrebno je održavati balansiranost stabla na sljedeći način: kada se na visini barem 3 desi da je broj elemenata u jednom podstablu duplo veći od broja elemenata u drugom podstablu, podstablo tog čvora se mijenja. Balansiranost se izvodi tako što se od elemenata podstabla napravi sortirani niz i onda se napravi podstablo koje je idealno balansirano. Pored balansiranja podstabala, ukoliko je potrebno prilikom unosa elemenata da se balansira čitavo stablo po kriterijumu koji se odnosi i za podstablo, to će se i izvršiti. Dakle, projekat obuhvata balansiranje čitavog stabla ukoliko se ispune kriteriji koji su zadati za ovu temu, te podstabla po istom kriteriju, prilikom unosa elemenata ili brisanja istih.

## Projekat se sastoji od sljedećih fajlova:

- „StabloProjekat.cbp“ – fajl koji pri pokretanju otvara ostale fajlove u projektu
- „stablo.h“ – header fajl, unutar kojeg su navedene funkcije
- „stablo.cpp“ – source fajl, unutar kojeg su implementirane sve funkcije koje su navedene unutar fajla „stablo.h“
- „main.cpp“ – source fajl, unutar kojeg se vrše testiranja implementiranih funkcija

Unutar fajla „stablo.h“ definisana je template klasa Stablo koja ima sljedeće atribute:

- struct Cvor – struktura koja gradi klasu Stablo. Svaki čvor unutar stabla je određen elementom, svojim roditeljem, lijevim i desnim dijetetom, te sa brojem elemenata sa lijeve strane čvora, brojem elemenata sa desne strane čvora i visinom čvora. Dakle, kako je prethodno navedeno, struktura Cvor se sastoji od atributa:
  - **Tip el** – vrijednost čvora
  - **Cvor *rod, *ld, *dd** – pokazivači na čvor, redom na roditelja, lijevo i desno dijete

    - **int brojElemenataLijevo** – vrijednost broja elemenata koji se nalaze sa lijeve strane čvora
    - **int brojElemenataDesno** – vrijednost broja elemenata koji se nalaze sa desne strane čvora
    - **int visinaCvora** – vrijednost koja predstavlja visinu čvora unutar stabla
    - Konstruktor Cvor sa paremetrima: element, roditelj, lijevo dijete, desno dijete, brojElemenataLijevo, brojElemenataDesno, visinaCvora

- **Cvor *korijen** – pokazivač na čvor
- **int broj_elemenata** – broj elemenata unutar stabla

**Privatne funkcije unutar fajla:**

- **void OcistiStablo(Cvor *korijen)** – funkcija za brisanje svih čvorova stabla počevši od proslijeđenog korijena. Dakle, briše i podstabla čiji proslijeđeni čvor predstavlja korijen tog podstabla. Koristi se prilikom poziva destruktora i također prilikom balansiranja stabla i podstabla. Pored brisanja elemenata, setuje broj elemenata na 0, što je i očekivano.
- **Cvor *KopirajStablo(Cvor *korijen, Cvor *rod_korijena, int &broj_elemenata)** – funkcija dinamički alocira novi čvor i inicijalizira ga sa korijenom stabla, potom setuje roditelja novog čvora sa roditeljem postojećeg korijena, povećava broj elemenata i poziva sebe za setovanje lijevog i desnog djeteta. Funkciju pozivaju konstruktor kopije i operator dodjele. Također, funkcija se koristi prilikom balansiranja stabla koje se dešava ili umetanjem ili brisanjem čvorova, da bi zapamtilo nove pozicije, visine, elemente sa lijeve i elemente sa desne strane.
- **void InOrderRek(Cvor *cvor, void (f)(Tip&)) const** – rekurzivna funkcija koja ispisuje elemente stabla u sortiranom poretku, po principu da prvo ispiše sve elemente koji se nalaze sa lijeve strane korijena, pa korijen i onda sve elemente koji se nalaze sa desne strane korijena. Funkcija prima i funkciju kao parametar. Ta funkcija predstavlja funkciju ispisi, koja ispisuje elemente stabla. Ključna riječ const čini funkciju konstantnom. To znači da ne može pozivati nijednu nekonstantnu funkciju članicu, niti može promijeniti bilo koju varijablu članicu.

- **Cvor *Sljedbenik(Cvor *cvor)** – funkcija koja vraća sljedbenika od čvora koji je proslijeđen funkciji. Ukoliko proslijeđeni čvor ima desno dijete, onda je sljedbenik proslijeđenog čvora, čvor koji je najdublji sa lijeve strane čvora desnog dijeteta, odnosno čvor čije je lijevo dijete na kraju nullptr. Ukoliko proslijeđeni čvor nema desno dijete, onda se sljedbenik traži po principu da sve dok je roditelj čvora različit od nullptr-a i dok desno dijete roditelja čvora je jednak tom čvoru, čvor se inicijalizira na čvor svog roditelja. Kada jedan od ova 2 uslova prestanu da važe, funkcija vraća čvor, čiji je roditelj nullptr ili čije desno dijete roditelja nije jednako tom čvoru. Funkcija se koristi prilikom unosa elemenata u niz po principu da krene od najmanjeg elementa u stablu ili podstablu, pa ide ka sljedećem dok god je uslov zadovoljen, te prilikom brisanja, i to konkretno u situaciji kada korisnik želi obrisati čvor koji ima svoje lijevo i desno dijete. Detaljnije informacije vezane za funkcionisanje prilikom brisanja će biti opisano pod funkcijom Erase().
- **Cvor *Pocetak(Cvor *trenutni)** – funkcija koja pronalazi najmanji element stabla gdje je korijen tog stabla čvor koji je proslijeđen kao parametar funkcije. Dakle, while petljom prolazimo kroz svako lijevo dijete, dok god je taj čvor različit od nullptr. Ova funkcija se koristi za pronalazak najmanjeg elementa stabla koje se treba balansirati, tako da od najmanjeg do najvećeg elementa se unosi u niz da bi niz bio sortiran, kako bi se moglo stablo uz pomoć tih elemenata izbalansirati.
- **Cvor *FindCvor(Tip element)** – funkcija koja pronalazi čvor po proslijeđenom elementu i vraća ga. Funkcija se koristi unutar funkcije Visina, da bi se pronašla visina tog pronađenog čvora u stablu.
- **int VisinaRek(Cvor *cvor)** – rekurzivna funkcija za pronalazak visine proslijeđenog čvora unutar stabla.
- **int BrojElemenataLijevoRek(Cvor *cvor)** – funkcija koja prima kao parametar lijevo dijete od čvora kojem želimo izbrojati broj elemenata, te potom se rekurzivno prolazi kroz svaki lijevi i desni čvor i dodaje se +1 za svaki pronalazak čvora.
- **int BrojElemenataDesnoRek(Cvor *cvor)** – funkcija koja prima kao parametar desno dijete od čvora kojem želimo izbrojati broj elemenata, te

potom se rekurzivno prolazi kroz svaki lijevi i desni čvor i dodaje se +1 za svaki pronalazak čvora.

- **bool DaLiJePotrebnoBalansiranje(Cvor *korijen, int visinaElementa, int brojElemenataLijevo, int brojElemenataDesno, Cvor *trenutni, Tip element) –** funkcija prima parametre prikazane unutar zagrada funkcije, koje su neophodne za provjeru po kriterijumu koji je zadan u tekstu zadatka (teme). Unutar funkcije su tri uslova, prvi koji provjerava situaciju kada korisnik unosi elemente čiji se čvorovi nadovezuju na desnu stranu korijena, te ukoliko ispunjava uslov kriterijuma, potrebno je balansirati stablo. Drugi uslov provjerava situaciju kada korisnik unosi elemente čiji se čvorovi nadovezuju na lijevu stranu korijena, te ukoliko ispunjava uslov kriterijuma, potrebno je balansirati stablo. Treći uslov se odnosi ukoliko dodje do nebalansiranosti jednog od podstabala unutar stabla po zadatom kriterijumu. Ukoliko niti jedan uslov nije zadovoljen, funkcija vraća false.
- **bool DaLiJePotrebnoBalansiranjeNakonBrisanja(int visinaElementa, int brojElemenataLijevo, int brojElemenataDesno, Cvor *trenutni, Tip element) –** funkcija sa jednim uslovom, koji nakon brisanja određenog elementa vrši provjeru, da li je stablo balansirano po kriterijumu ili ne. Ukoliko nije balansirano, funkcija vraća true, u suprotnom vraća false.
- **Cvor *Kreiraj(Tip el)** – funkcija koja dinamički alocira čvor, te trenutno postavlja lijevo i desno dijete na nullptr, broj elemenata lijevo, broj elemenata desno i visinu čvora na 0, koji će se naknadno (ne u ovoj funkciji, već u funkciji KreirajBalansiranoStablo) setati na prave vrijednosti. Funkcija vraća kreirani čvor.
- **Cvor *KreirajBalansiranoStablo(Tip niz[], int pocetak, int kraj)** – rekurzivna funkcija pomoću koje se kreira balansirano stablo. Princip na kojem funkcioniše ova funkcija, jeste da prolazi kroz sortirani niz koji je proslijeđen, po principu da krene od srednjeg elementa niza, obuhvati lijevu stranu niza, pa potom desnu stranu niza. Dakle, srednji element predstavlja korijen stabla, potom idu lijevi elementi niza, i tako se respektivno veže čvor za čvor po lijevoj strani od korijena. Nakon toga, idu desni elementi niza, i vežu se čvorovi po desnoj strani od korijena. Funkcija vraća pokazivač na korijen tog stabla, koji se potom koristi za setanje ili pokazivača korijena

stabla koji je prvobitno bio van balansa ili pokazivača korijena podstabla koji je isto tako bio van balansa.

- **void UbaciUStablo(Cvor *korijen1, Tip element)** – funkcija koja ubacuje element proslijeđen funkciji, na određeno mjesto unutar stabla. Petljom se kreće od korijena stabla, te se vrši provjera po veličini trenutnog elementa stabla i proslijeđenog elementa, te se pronalazi njegova pozicija. Kada se pronađe njegova pozicija, dinamički se alocira novi čvor, dodijeljuje se stablo i povećava se broj elemenata stabla za 1. Funkcija se koristi unutar funkcije DodajBalansiranoPodstablo, o kojem će biti par riječi u nastavku.
- **void DodajBalansiranoPodstablo(Cvor *korijen, Tip niz[], int pocetak, int kraj)** – u situaciji kada je određeno podstablo van balansa, tj. nije balansirano, uzimaju se elementi i stavljaju u niz, te se niz prosljeđuje ovoj funkciji. Niz je sortiran, te se elementi izvlače tako, da se krene od srednjeg elementa, pa se izvuku svi elementi sa lijeve strane, pa potom svi elementi sa desne strane niza. Pošto je funkcija rekurzivna, a neophodno je obuhvatiti lijevu i desnu stranu niza, vrše se dva poziva, koja obuhvataju obje strane niza.
- **void ResetujPodatkeCvora(Cvor *korijen)** – funkcija koja nakon balansiranja, prolazi kroz čitavo stablo, i resetuje podatke o broju elemenata sa lijeve i desne strane, te visine čvora. Razlog tome je, što nakon balansiranosti, određeni čvorovi mijenjaju svoje mjesto, te potencijalno dolazi do promjene broja elemenata sa njihove lijeve i desne strane, te promjene visine unutar stabla. Pošto se ti podaci čuvaju unutar čvora, neophodno je proći kroz stablo, uzeti nove podatke čvora i setati ih, da bi daljna balansiranja bila ispravna.

**Javne funkcije unutar fajla:**

- **Stablo()** – konstruktor klase koji inicijalizira broj elemenata na 0, te postavlja pokazivač korijena na nullptr. Izvršava se automatski kad god se kreira objekat klase.
- **~Stablo()** – funkcija koja se automatski poziva kada objekat izađe izvan opsega ili se eksplicitno uništi pozivom za brisanje. Unutar destruktora se poziva funkcija OcistiStablo(), o kojem je rečeno par riječi.

- **Stablo(const Stablo<Tip> &s)** – konstruktor kopije, koji stvara objekt tako da ga inicijalizira s objektom iste klase, koji je prethodno kreiran. Koristi se za inicijaliziranje jednog objekta iz drugog istog tipa.
- **Stablo(Stablo<Tip> &&s)** – pomjerajući konstruktor, koji pomjera resurse u gomilu, tj. za razliku od konstruktora kopije koji kopiraju podatke postojećeg objekta i dodjeljuju ih novom objektu, pomjerajući konstruktor samo čini da pokazivač deklariranog objekta pokazuje na podatke privremenog objekta i poništava podatke privremenog objekta i pokazivača privremenih objekata.
- **Stablo<Tip>& operator=(const Stablo<Tip> &rhs)** – operator dodjele ima imperativno dejstvo, kojim se objektu sa lijeve strane operatora dodjele dodjeljuje vrijednost sa desne strane.
- **Stablo<Tip>& operator=(Stablo<Tip> &&rhs)** – pomjerajući operator dodjele koji prima kao parametar privremeni objekat tj. r-value (r-vrijednost).
- **void Insert(Tip element)** – funkcija koja umeće element u stablo. U slučaju da nema elemenata, dinamički se alocira korijen stabla i povećava broj elemenata za 1. U slučaju da već postoji element ili elementi u stablu, prolazi se kroz stablo da se pronađe odgovarajuće mjesto za novo uneseni element, te se povećava broj elemenata lijevo i broj elemenata desno u zavisnosti od nailaska čvora. Također, prolaskom kroz svaki čvor stabla, vrši se provjera da li će unošenjem elementa doći do narušavanja balansiranosti stabla. Ukoliko se ne narušava balansiranost, element se unosi na poziciju koja je određena već ranijim prolazom kroz svaki čvor stabla i povećava broj elemenata za 1. Određivanje visine unutar stabla se radi tako što se kreira novi čvor sa tim elementom koji je proslijeđen funkciji i postavlja se kao lijevo ili desno dijete prethodnog čvora (koji čini zapravo čvor koji će postati roditelj novo unesenom čvoru) u zavisnosti od ispunjenja uslova, te se roditelj novog čvora postavlja na prethodni. Nakon toga, while petljom se prolazi kroz elemente i setaju se nove visine za čvorove unutar stabla. U slučaju da se narušava balansiranost, počevši od korijena stabla ili podstabla gdje je narušena balansiranost, elementi se unose u niz. Niz je sortiran. Kada se unesu svi potrebni elementi u niz, pozivom funkcije OcistiStablo, brišemo sve elemente koji narušavaju balansiranost i respektivno smanjujemo broj elemenata. Potom, postoje dvije situacije. Prva situacija je da je čitavo stablo van balansa. Tada se pozivom funkcije KreirajBalansiranoStablo(), kreira balansirano stablo, te se poziva konstruktor

kopije kako bi sačuvali promjene tokom balansiranja. Pored navedenih funkcija, poziva se i funkcija ResetujPodatkeCvora kako bi se setovale nove vrijednosti unutar čvorova. Druga situacija je kada je određeno podstablo, unutar stabla van balansa. Tada su svi čvorovi uključujući i „korijen“ tog podstabla izbrisani iz stabla, tako da sad se stablo samo sastoji od elemeneta koji nisu bili van balansa. Pozivom funkcije DodajBalansiranoPodstablo(), se dodaju elementi, po principu da to podstablo bude balansirano. Nakon unosa i posljednjeg elementa iz sortiranog niza, dobijamo balansirano podstablo i samim tim i balansirano stablo. Pozivom funkcije ResetujPodatkeCvora, setuju se nove vrijednosti unutar čvorova.

- **bool Empty()** – provjerava da li je stablo prazno ili ne. U zavisnosti od toga da li je prazan, vraća true, u suprotnom vraća false.
- **void InOrder(void (f) (Tip&)) const** – funkcija koja poziva InOrderRek(), za ispisivanje elemenata stabla u sortiranom poretku, kao i njihovo lijevo i desno dijete.
- **int Size()**– vraća broj elemenata unutar stabla.
- **bool Find(Tip element)** – funkcija koja provjerava da li se proslijeđeni element nalazi unutar stabla ili ne. Ukoliko se nalazi, vraća vrijednost true, u suprotnom vraća vrijednost false.
- **void Erase(Tip element)** – funkcija za brisanje elemenata iz stabla. Funkcioniše po principu da izbriše čvor iz stabla i izvrši provjeru da li je stablo van balansa nakon brisanja. Ukoliko je stablo nakon brisanja tog čvora van balansa, onda se vrši balansiranje. Također pored stabla, funkcioniše i za balansiranje podstabla ukoliko dođe do takve situacije. Erase funkcija se sastoji iz 4 uslova. Prvi uslov je ukoliko korisnik želi obrisati element koji je list u stablu, tj. element koji nema ni lijevo ni desno dijete. U toj situaciji se prosto obriše taj element i smanji broj elemenata za 1. Drugi uslov je ukoliko korisnik želi obrisati element čiji čvor nema desnog djeteta. Tada postavljamo pokazivače tako da roditelj čvora kojeg brišemo postaje roditelj njegovom lijevom djetetu, a lijevo dijete postaje lijevo dijete roditelja čvora. Nakon toga ciljani čvor se briše i smanjuje broj elemenata za 1. Treći uslov je ukoliko korisnik želi obrisati element čiji čvor nema lijevog djeteta. Tada postavljamo pokazivače tako da roditelj čvora kojeg brišemo postaje roditelj njegovom desnom djetetu, a desno dijete postaje desno dijete roditelja čvora. Nakon toga

ciljani čvor se briše i smanjuje broj elemenata za 1. Četvrti uslov je ukoliko korisnik želi obrisati element čiji čvor ima i desno i lijevo dijete. Pronalazimo sljedbenika tog čvora i vršimo provjeru da li je sljedbenik jednak lijevom dijetetu roditelja sljedbenika. Ukoliko jeste, lijevo dijete roditelja sljedbenika postaje desno dijete sljedbenika, u suprotnom, desno dijete roditelja sljedbenika postaje desno dijete sljedbenika. Potom se provjerava da li je desno dijete sljedbenika različito od nullptr. Ukoliko jeste, roditelj desnog djeteta od sljedbenika postaje roditelj sljedbenika. Setuju se lijevo i desno dijete, roditelj, roditelj lijevog djeteta trenutnog elementa i roditelj desnog djeteta trenutnog elementa. Nakon setovanja, vrši se provjera da li je trenutni element jednak lijevom djetetu roditelja od sljedbenika. Ukoliko jeste, lijevo dijete roditelja od sljedbenika se setuje na sljedbenika, u suprotnom, desno dijete roditelja od sljedbenika se setuje na sljedbenika. Balansiranje nakon brisanja se izvršava po istom principu kao i kod Insert funkcije.

- **int Visina(Tip element)** – funkcija koja provjerava na kojoj se visini unutar stabla nalazi element koji je proslijeđen kao parametar funkcije. Trenutni čvor se nalazi tako što se pomoću funkcije FindCvor pronađe čvor proslijeđenog elementa, te se potom taj čvor proslijedi drugoj funkciji VisinaRek(), koja pronalazi visinu tog elementa unutar stabla, te je vraća.

Pored ovih funkcija, kreiran je i **ostream& operator<<(ostream &ispis, const Stablo<Tip> &s)** za ispis elemenata stabla preko cout << u main.cpp. Operator prima ostream i stablo, te uz pomoć funkcija InOrder(), InOrderRek() i ispisi(), ispisuje elemente stabla kao i njihovo lijevo i desno dijete, u sljedećem formatu:

element

LD: lijevo dijete elementa ili prazno ako nema lijevo dijete

DD: desno dijete elementa ili prazno ako nema desno dijete