Projekt iz predmeta *“Strukture podataka i algoritmi”*

Student: Edin Smajić

Smjer: Teorijska kompjuterska nauka (TKN)

Godina: Druga (II)

Status: Redovni-samofinansirajući

Broj indeksa: 5810/M

Tema 1

Uvod:

U prilogu sa ovom dokumentacijom poslat je i folder pod imenom “Projekt 2” u kojem se nalaze: “stablo.cbp”, “stablo.h”, “stablo.cpp” i “main.cpp”.

“stablo.cbp” - fajl koji pri pokretanju automatski otvara ostale navedene fajlove.

“stablo.h” - header fajl, u kojem su navedeni prototipovi funkcija, bez implementacija istih, te su u njemu uključeni neki drugi headeri koji su neophodni za ispravan rad projekta.

“stablo.cpp” - source fajl, u kojem su implementirane sve funkcije čiji su prototipovi navedeni u “stablo.h”.

“main.cpp” - source fajl, u kojem se vrše testiranja tih funkcija, o kojima će ispod biti više govora.

U fajlu “stablo.h” definisana je template klasa Stablo koja ima sljedeće atribute (koji su privatni):

1. struct Cvor - struktura kojom gradimo našu klasu Stablo.
    - Klasa Stablo je ustvari “stablo” instanci strukture Cvor koji su međusobno povezani.
    - Svaki čvor je jedinstven i tačno određen sa svojim “roditeljom”, “lijevim” i “desnim dijetom”.

- Atributi strukture Cvor:

  A) Tip element – vrijednost koju čuvamo u čvoru.

  B) int prioritet – prioritet čvora.

  C) Cvor *rod, *ld, *dd – sve tri pokazivači na čvor, redom sinonimi za "roditelj", "lijevo dijete", te "desno dijete".

- Struktura Cvor sadrži jedan konstruktor sa 4 parametra, od kojih su 3 opcionalna, koristi se za inicijalizaciju instance strukture Cvor.

2. Cvor* korijen - pokazivač na čvor, ono po čemu je specifičan korijen jeste da on nema "roditelja", tj. "roditelj" mu je uvijek nula-pokazivač.

3. int velicina – broj čvorova u stablu.

Privatne funkcije:

1) **Stablo**(Cvor *korijen);

- Jedini privatni konstruktor koji prima čvor i inicijalizira korijen stabla sa tim čvorom, te inicijalizira veličinu na nulu.
- Ukoliko korijen nije nula-pokazivač stavlja mu roditelja na nula-pokazivač (što je odlika korijena) i poziva funkciju SetujVelicinu sa korijenom kao parametrom.
- On je neophodan za funkciju **Razdvajanje** o kojoj će biti govora nešto kasnije.

2) void **SetujVelicinu**(Cvor *korijen);

- Koristi se za setovanje veličine stabla na osnovu korijena koji prima kao parametar.
- Ovu funkciju poziva samo konstruktor naveden iznad tj. 1).

3) void **OcistiStabloRekurzivno**(Cvor *korijen);

- Koristi se, kao što samo ime funkcije govori, za čišćenje stabla, brisanjem svih čvorova od kojih se stablo sastoji i smanjivanjem veličine do nule.
- Ovu funkciju poziva samo destruktor.

4) Cvor* **PomocniKopirac**(Cvor *korijen, Cvor *korijen_roditelj, int &velicina);

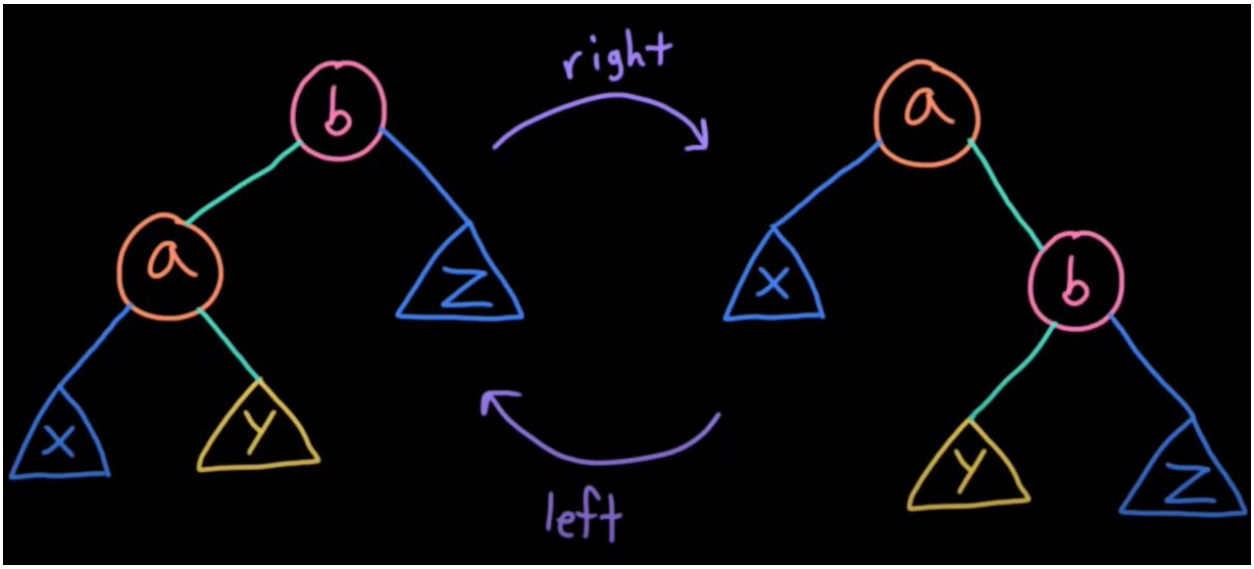
- Rekurzivna funkcija koju pozivaju konstruktor kopije i operator dodjele.
- Na pametan način pravi novo stablo u memoriji tako što dinamički alocira novi čvor i inicijalizira ga sa elementom čvora korijen, te setuje roditelja novog čvora na korijen_roditelj i poistovijeti prioritet novog čvora sa prioritetom čvora korijen, poveća veličinu, a zatim rekurzivno poziva samu sebe za setovanje lijevog i desnog dijeta sa odgvovorajućim parametrima.

Digresija:

Da bi objasnili rotacije koje su neophodne za očuvanje poretka u našem stablu pri umetanju i brisanju elemenata, neophodno je poznavati sljedeće:

- Prilikom umetanja elementa u stablo, taj novi element će uvijek biti u čvoru koji je “list” (čvor koji nema ni lijevo ni desno dijete, tj. njegova djeca su nula-pokazivači). Njemu će biti dodijeljen prioritet na slučajan način, stoga on može narušiti svojstvo našeg stabla da su čvorovi takvi da je prioritet nekog čvora veći od prioriteta njegove djece, ali manji od prioriteta njegovog roditelja. Stoga je neophodno izvršiti te neke rotacije koje će očuvati to svojstvo našeg stabla.
- Lahko je zaključiti da su rotacije koje su neophodne za očuvanje navedenog svojstva ustvari tzv. “AVL” rotacije koje se koriste i u “AVL” stablu (stablo implementirano na drugi način).

- Postoje dvije vrste “AVL” rotacija i to: “desna” i “lijeva” rotacija. One su prikazane na sljedećoj slici (a i b su konkretni elementi, dok su x, y i z podstabla koja sadrže druge elemente):



- Lijevi dio slike: ukoliko element ‘a’ ima veći prioritet od njegovog roditelja tj. elementa ‘b’ onda vršimo “desnu” rotaciju (“right” na slici).
- Desni dio slike: ukoliko element ‘b’ ima veći prioritet od njegovog roditelja tj. elementa ‘a’ onda vršimo “lijevu” rotaciju (“left” na slici).
- Ispod slijedi ništa drugo do implementacija svega navedenog.

Završetak digresije

5) void **Rotacija**(Cvor *cvor);

- Funkcija koja prima čvor i u ovisnosti od njegovog sadržaja ne radi ništa ili, poziva funkciju **DesnaRotacija** ili funkciju **LijevaRotacija** sa odgovarajućim parametrima.

6) void **DesnaRotacija**(Cvor *lijevi, Cvor *gornji);

- "Stavlja" lijevi čvor na mjesto gornjeg čvora, a gornji čvor "spušta" udesno.

7) void **LijevaRotacija**(Cvor *desni, Cvor *gornji);

- "Stavlja" desni čvor na mjesto gornjeg čvora, a gornji čvor "spušta" ulijevo.

8) bool **List(**Cvor *cvor) const;

- Vraća true ukoliko čvor cvor nema djece.

9) Cvor* **FindCvor**(const Tip &element);

- Traži čvor čiji se element poklapa sa parametrom ove funkcije i vraća pokazivač na taj čvor, u suprotnom vraća nula-pokazivač.

10) Cvor* **OdgovarajuceDijete**(Cvor *cvor);

- Vraća nula-pokazivač ukoliko čvor cvor nema djece, ukoliko on ima jedno dijete vrati ga, a inače vraća dijete koje ima veći prioritet od drugog.

11) void **ObradiElementeRekurzivno**(Cvor *korijen, void (f)(Tip)) const;

- Obrađuje elemente sa funkcijom f rekurzivno tj. poziva funkciju f nad svakim elementom u stablu. (i to u rastućem poretku)

12) Tip **OdgovarajuciBroj**(const Stablo<Tip> &s1, const Stablo<Tip> &s2) const;

- Za stabla koja prima kao parametar vrijedi da su svi elementi stabla s1 manji od svih elemenata u stablu s2, ova funkcija pronalazi najveći element u stablu s1, te najmanji element u stablu s2, zatim vraća polovinu zbira ova dva elementa, jer je takav broj veći od svih elemenata stabla s1, a manji od svih elemenata stabla s2, što nam je poretbno za "spajanje" ova dva stabla.
- Poziva se samo u funkciji **PostupakSpajanja** posredstvom funkcije **Spajanje** o kojima će ispod biti govora.

13) void **PostupakSpajanja**(Stablo<Tip> &s1, Stablo<Tip> &s2, const Tip&element, const int &prioritet);

- Ovu funkciju poziva samo funkcija **Spajanje**.
- Funkcija spaja stabla s1 i s2 na način koji je opisan u opisu teme 1, ali ako se poziva samo sa stablima kao argumentima tj. obično spajanje (slučaj prioritet==0).
- Funkcija u stablo nad kojim je pozvana (koje je prazno) umeće element koji mu je proslijeđen kao treći parametar i setuje njegov prioritet na –prioritet koji je četvrti parametar, jer se prioritet uvijek prosljeđuje kao negativan iz funkcije **UnijaRekurzivno**. Te kao lijevo dijete korijena setuje korijen stabla s1, analogno kao desno dijete korijena setuje korijen stabla s2 i to na način da zauzme taj prostor u memoriji, a zatim korijene stabala s1 i s2 setuje na nula-pokazivače, a veličine na nulu, dakle briše ta stabla. Ovaj slučaj je samo za pozive iz funkcije **UnijaRekurzivno**. (slučaj prioritet<0)

14) Stablo<Tip> UnijaRekurzivno(Stablo<Tip> &s1, Stablo<Tip> &s2);

- Ovu funkciju poziva samo funkcija **Unija**.
- Funkcioniše na način koji je opisan u opisu teme 1.

Javne funkcije:

Konstruktori, operatori dodjele i destruktor će biti izostavljeni iz opisa zbog njihove jednostavnosti.

1) int **Size**() const;

- Vraća veličinu stabla.

2) bool **Find**(const Tip &element) const;

- Vraća true ukoliko se element koji je proslijeđen kao parametar nalazi u stablu, a false u suprotnom.

3) void **Insert**(const Tip &element, const bool &jelKljuc=false);

- Vrši obično umetanje elementa koji je prvi parametar u stablo po defaultu, a zatim vrši dodatne rotacije za očuvanje svojstva našeg stabla vezanog za poredak.
- Međutim ako je jelKljuc true, tada funkcija vrši umetanje elementa koji je prvi parametar u stablo i setuje njegov prioritet na za jedan veći prioritet od prioriteta korijena stabla, zatim vrši dodatne rotacije za očuvanje svojstva našeg stabla vezanog za poredak, pri čemu će umetnuti element "izbiti" na vrh I bit će korijen stabla. Ovo koristi funkcija **Razdvajanje** o kojoj će biti govora.

4) void **Delete**(const Tip &element);

- Briše element koji je proslijeđen kao parametar iz stabla, zatim vrši dodatne rotacije za očuvanje svojstva našeg stabla ukoliko je to potrebno.

5) void **ObradiElemente**(void (f)(Tip)) const;

- Poziva funkciju **ObradiElementeRekurzivno** sa funkcijom f kao argumentom ukoliko korijen stabla nije nula-pokazivač.

6) pair<Stablo<T>, Stablo<T>> **Razdvajanje**(Stablo<T> &s, const T &kljuc);

- Razdvaja stablo s na osnovu ključa kljuc na dva stabla i smješta ih u par, tako da prvi par sadrži sve elemente manje od ključa dok drugi par sadrži preostale elemente tj. elemente veće od ključa.
- Konkretno radi sljedeće: insertuje ključ koristeći funkciju Insert kojoj proslijedi drugi parametar kao true, pri čemu će Insert umetnuti ovaj ključ na način da će ključ kljuc "isplivati" na vrh. Pri čemu će lijevo dijete ključa biti korijen podstabla koji je prvi argument para, a desno dijete ključa će biti korijen podstabla koji je drugi argument para koji ćemo vratiti.

7) Stablo<T> **Spajanje**(Stablo<T> &s1, Stablo<T> &s2, const T &element, const int &prioritet);

- Pravi instancu klase Stablo i nad tom instancom poziva funkciju **PostupakSpajanja**, a zatim vrati tu instancu.

8) Stablo<T> **Unija**(Stablo<T> &s1, Stablo<T> &s2);

- Vraća stablo koje predstavlja uniju stabala s1 i s2, pozivajući funkciju **UnijaRekurzivno** sa stablima s1 i s2 kao parametrima.

9) void **ispisi**(Tip e);

- Ispisuje e sa razmakom poslije.

10) ostream& operator<<(ostream &ispis, const Stablo<Tip> &s);

- Ispisuje stablo u sljedećem formatu:
- N: { e1, e2, …, eN }, gdje N predstavlja veličinu stabla, a ei, i=1, N elemente tog stabla.
- To radi pozivajući funkciju **ObradiElemente** sa funkcijom ispisi kao argumentom.