UNIVERZITET U SARAJEVU
PRIRODNO-MATEMATIČKI FAKULTET

Dokumentacija za projekat iz predmeta "Strukture podataka i algoritmi"

TEMA 1

Student:

Nejra Šabanović

5780/M

Sadržaj

Uvod

Tema **1,** prvog projekta iz struktura podataka i algoritama se sastoji od nekoliko stvari koje je trebalo implementirati u klasu Stablo.

Prva od njih je iterator i reverse iterator koji će omogućiti kretanje kroz stablo, zatim funkciju koja provjerava da li je jedno stablo podskup drugog i na kraju omogućiti indeksiranje u stablu, gdje Stablo[k] vraća k-ti element po veličini elemenata u stablu.

Implementacija projekta je podijeljena u tri fajla: main.cpp, stablo.cpp i stablo.h .

U nastavku je objašnjena logika iza svake napravljene funckije i dati odgovori na pitanja zašto? i kako? je nešto implementirano.

Iterator

Klasa Iterator u projektu nazvana TreeIterator napravljena je unutar klase Stablo i public je.

Vodeći se načinom na koji iterator radi u strukturi set, tako ovaj iterator ima neke od osnovnih funkcija koje će omogućiti isto.

Dakle, postoji jedan atribut, a to je *pok tipa Cvor, zatim

- Konstruktor

```
TreeIterator(Cvor *cvor);
```

- Preklopljeni operatori :

```
Tip& operator*();
TreeIterator& operator++();
TreeIterator& operator++(int);
friend bool operator==(TreeIterator it1,TreeIterator it2);
friend bool operator==(TreeIterator it1,TreeIterator it2);
```

Također, dodana je i funkcija NadjiSljedbenikaIterator, kao i pomoćna funkcija NadjiNajmanjiIterator koje služe za implementaciju operatora++ koji i omogućava ključnu ulogu iteratora. Kako? Tako što nam operator++ svakim pozivom daje sljedbenika u stablu od traženog čvora, na taj način iterator će vršiti kretanje kroz stablo, što potvrdi i sami ispis elemenata u in order-u.

Reverse Iterator

Reverse Iterator nazvan Reverse_Iterator nasljeđuje klasu TreeIterator , samim tim posjeduje iste funkcije.

Za razliku od pronalaženja sljedbenika kao u običnom iteratoru, u reverse iteratoru se pronalazi prethodnik. Zato je dodana funkcija NadjiPrethodnikaIterator i pomoćna funkcija NadjiNajveci, koja vraća pokazivac na cvor s najvećim elementom. Uz pomoć prethodnika

reverse_iterator omogućava kretanje unazad, gdje sada operator++ pomjera iterator na čvor unazad.

Da bi naredne petlje mogle raditi,

```
for(auto it = S.Begin(); it != S.End(); it++)
for(auto rev_it = T.rBegin(); rev_it != T.rEnd(); rev_it++)
```

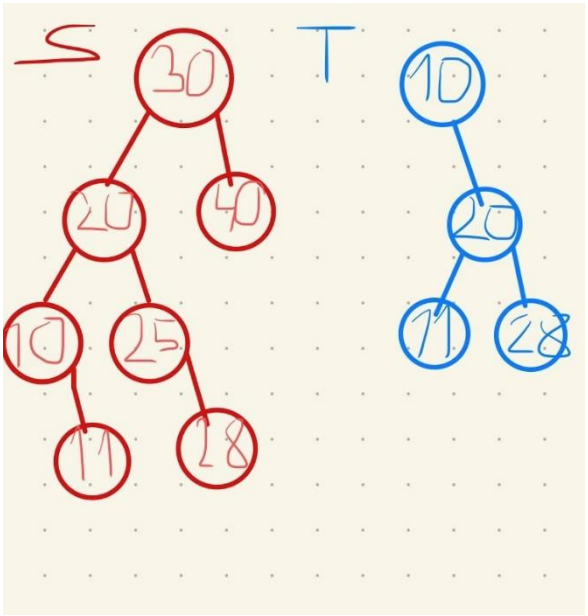
potrebno je bilo napraviti funkcije u stablu:

- `TreeIterator Begin()` – funkcija koja vraća iterator na početak stabla
- `TreeIterator End()` - funkcija koja vraća iterator na iza – posljednjeg – elementa stabla
- `TreeIterator rBegin()` - funkcija koja vraća početak za reverse iterator
- `TreeIterator rEnd()` - funkcija koja vraća kraj za reverse iterator

DaLiJePodskup

Naredna napravljena funkcija je friend funkcija koja prima dva Stabla i provjerava da li je drugo stablo podskup drugog.

```
friend bool daLiJePodskup(Stablo<Tip> s1,Stablo<Tip> s2);
```



Ilustracija 1 Podskup

Za stablo kažemo da je podskup drugog ukoliko posjeduje čvorove sa istim elementima kao prvo stablo pri čemu redoslijed čvorova nije bitan.

Prilikom implementacije korišten je prethodno napravljeni iterator koji omogućava da se vrši kretanje kroz prvo stablo i u isto vrijeme provjerava da li drugo stablo posjeduje posjećene čvorova. Ukoliko se završi obilazak prvog stabla, pri čemu čvorovi u drugom nisu svi posjećeni to znači da ne postoji podskup te funkcija vraća false, u suprotnom funkcija vraća true.

Indeksiranje

Posljednji zahtjev ovog projekta bio je omogućiti indeksiranje u stablu, što bi značilo preklopiti operator []. Stoga funkcija napisana kao

```
Tip operator[](int k);
```

vraća element koji se nalazi na k – tom mjestu u stablu. Elementi se vraćaju po veličini, dakle S[0] će vratiti najmanji element stabla. Ukoliko element ne postoji na datom indeksu funkcija vraća -1 kao rezultat.