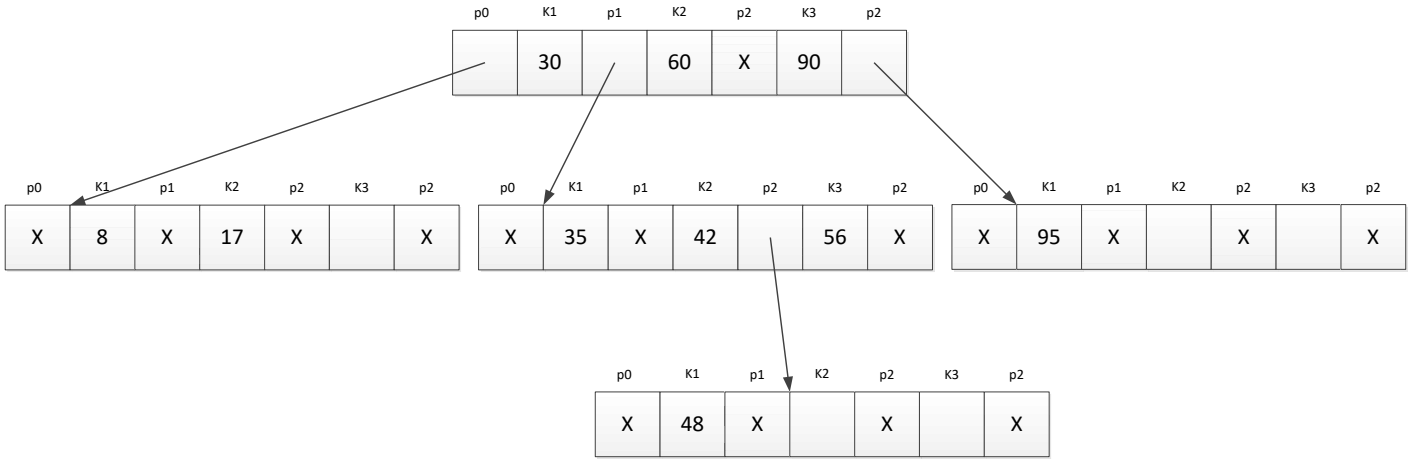
OBJEKTNO ORIJENTISANO PROGRAMIRANJE

- domaći zadatak broj 2 -

Funkcionalna specifikacija

Na programskom jeziku C++ implementirati biblioteku klasa za dinamičku reprezentaciju stabla M-arnog pretraživanja (*MSearchTree*). U nastavku teksta opisane su funkcionalnosti tri klase: `MSearchTree`, `Key` i `TreeNode`.



Slika 1 Primer stabla M-arnog pretraživanja za M=3. Prazni pokazivači su označeni znakom X. Brojevi predstavljaju ključeve za pretraživanje. Prazne kućice označavaju nedostajuće ključeve.

Specifikacija klase MSearchTree

Dinamičko stablo M-arnog pretraživanja je ulančana nelinearna struktura podataka koja se sastoji od povezanih čvorova, pri čemu ne sme da se pojavi ciklus. Svaki čvor (*TreeNode*) sadrži najviše M ključeva za pretragu (*Key*) i M+1 pokazivača na čvorove decu. Klasa stabla ima privatno polje `root` koje predstavlja pokazivač na koreni čvor stabla.

Stablo se stvara prazno sa zadatim parametrom M.

Implementirati konstruktor klase `MSearchTree(int m);` sa parametrom M, podrazumevano 1.

Stablo se može stvoriti kao kopija već postojećeg stabla.

Implementirati obe varijante konstruktora kopije (duboko kopiranje i kopiranje sa premeštanjem) za potrebe kloniranja stabala.

Stablu se može dodeliti drugo stablo.

Preklopiti obe varijante operatora za dodelu vrednosti (operator=), sa kopiranjem i sa premeštanjem.

U stablo se može dodati ključ.

Implementirati metodu `bool insert(const Key& k);`.

Metoda traži odgovarajući čvor list (*TreeNode*) za umetanje ključa. Ako nađeni čvor nije pun, u njega se dodaje ključ. U suprotnom ključ se dodaje u novi čvor list, koji se zatim vezuje na odgovarajuće mesto u trenutnom listu. Ukoliko ključ već postoji u stablu, vraća se false, u suprotnom true.

Može se proveriti da li zadati ključ već postoji u stablu.

Implementirati metodu `bool lookup(const Key&) const;`.

Metoda vraća logičku vrednost true ukoliko pronađe zadati ključ, u suprotnom false.

Omogućiti proizvoljan obilazak čvorova stabla.

Implementirati metodu `void traverse (Traversal& t)`.

Klasa `Traversal` treba da bude deklarisana kao prijateljska klasa klasi `MSearchTree`. Implementirati klasu `Traversal`, tako da nema nijedno polje i ima praznu metodu

`virtual void visit(TreeNode* node);`

Metoda `traverse` u klasi `MSearchTree` treba da ima jednu liniju koda: `t.visit(root);`.

Može se izbaciti zadati ključ iz stabla.

Implementirati metodu `bool remove(const Key& key)`, koja izbacuje zadati ključ iz stabla.

Vraća logičku vrednost true ukoliko ključ postoji i izbačen je, u suprotnom false. Metoda prvo pronalazi čvor koji sadrži zadati ključ, a zatim se iz čvora izbaci ključ, pri čemu se upražnjeno mesto mora popuniti nekim drugim ključem iz stabla, ako je to moguće.

Upražnjeno mesto u čvoru nakon izbacivanja ključa mora se nadomestiti na sledeći način: (1) minimalnim ključem iz desnog podstabla ako ono postoji, (2) maksimalnim ključem iz levog podstavla ako postoji, (3) najbližim desnim sledbenikom u istom čvoru, ako postoji (uz pomeranje svih sledbenika ulevo) i (4) ostaje prazno mesto. Ukoliko se desi da bilo koji čvor u preraspodeli ključeva ostane prazan, mora biti obrisan iz stabla. Metoda mora da obezbedi korektan poredak ključeva nakon izbacivanja zadatog ključa.

Ispisivanje stabla.

Preklopiti operator<< koji ispisuje stablo u notaciji sa zagradama. Za primer stabla na slici 1, notacija sa zagradama treba treba da bude: ((8,17),30,(35,42,(48),56),60,90,(95))

Omogućiti propisno uništavanje stabla.

Implementirati destruktor koji treba da oslobodi sve čvorove stabla.

Specifikacija klase Key

Ključ za pretragu (*Key*) sadrži celobrojnu vrednost *value* (ceo broj).

Ključ se pravi na osnovu zadate celobrojne vrednosti.

Implementirati konstruktor `Key(int value)` za inicijalizaciju ključa.

Ključ se može uporediti sa drugim ključem.

Preklopiti operatore <, <=, >, >=, ==, !=. Poređenje se svodi na poređenje celobrojnih vrednosti.

Ključ se može ispisati u izlazni tok.

Preklopiti `operator<<` za ispis objekata klase Key u izlazni tok.

Specifikacija funkcionalnosti klase TreeNode.

Čvor stabla sadrži M ključeva za pretragu, koji se čuvaju u rastućem poretku, i M+1 pokazivača na čvorove podstabla (*TreeNode**). Ključevi i pokazivači su organizovani tako da svaki ključ razdvaja dva pokazivača. Za svaki ključ K_i u čvoru važi da su svi ključevi u podstablu, na koji ukazuje njegov levi pokazivač P_{i-1}, manji od Ki i da su svi ključevi u podstablu na koje ukazuje njegov desni pokazivač P_i veći od K_i.

Čvor se stvara prazan sa zadatim brojem ključeva M.

Implementirati konstruktor klase `TreeNode(int m)` koji ima parametar M.

Čvor se može napraviti kao kopija drugog čvora.

Implementirati konstruktor kopije koji kopira samo ključeve (pokazivači kopije su jednaki nullptr).

Čvoru se može dodati novi ključ.

Implementirati metodu `bool add(const Key& k);`

Ukoliko čvor nije pun, ključ k se stavlja na odgovarajuću poziciju, tako da se održi poredak ključeva, i vraća true. U suprotnom, vraća se false i ne radi ništa.

Može se proveriti da li je čvor pun.

Implementirati metodu `bool isFull() const;` koja vraća true samo ako čvor sadrži M ključeva.

Može se proveriti da li je čvor prazan.

Implementirati metodu `bool isEmpty() const;` koja vraća true ako čvor ne sadrži nijedan ključ.

Dohvatanje ključeva u čvoru.

Implementirati metodu `vector<Key> getKeys() const;` koja vraća ključeve u čvoru.

Dohvatanje levog/desnog podstabla ključa.

Metoda vraća adresu podstabla ako postoji ključ, u suprotnom poziva se errorDetected.

`TreeNode* getLeftSubtree(const Key&) const;` i

`TreeNode* getRightSubtree(const Key&) const;`.

Može se dohvatiti najveći ključ u čvoru.

Implementirati metodu `Key getMaxKey() const;` koja vraća najveći ključ u čvoru. Ukoliko je čvor prazan, pozvati funkciju errorDetected.

Može se dohvatiti najmanji ključ u čvoru.

Impelementirati metodu `Key getMinKey() const;` koja vraća najmanji ključ u čvoru. Ukoliko je čvor prazan, pozvati funkciju errorDetected.

Može se proveriti da li čvor sadrži zadati ključ.

Implementirati metodu `bool contains(const Key& k) const;` koja vraća true ukoliko pronađe zadati ključ u čvoru. U suprotnom, vraća false.

Može se dohvatiti čvor podstabla kojem pripada zadati ključ.

Implementirati metodu `TreeNode* getSubtree(const Key& k) const;` koja vraća vrednost pokazivača gde bi po pravilu formiranja stabla trebalo da se nalazi ključ k. Ukoliko ključ pripada čvoru poziva se errorDetected i prosleđuje poruka o neadekvatnom protokolu korišćenja date metode.

Može se vezati čvor podstabla na osnovu zadatog ključa.

Implemenirati metodu `bool setSubtree(const Key& k, TreeNode* subnode);` Metoda uvezuje čvor podstabla i vraća true, akko je odgovarajući pokazivač podstabla jednak nullptr i čvor ne sadrži ključ k. U suprotnom, vraća se false.

Može se ukloniti zadati ključ iz čvora.

Implementirati metodu `bool remove(const Key& k);` koja izbacuje ključ iz čvora stabla i vraća true samo ako k postoji. U suprotnom, vraća se vrednost false. Ako izbačeni ključ nije imao ni levo ni desno podstablo, vrši se kompakcija čvora pomeranjem ključeva i pokazivača sa desne strane ključa.

Ispis čvora stabla.

Preklopiti operator<< koji ispisuje ključeve u čvoru kao niz u zagradama: (2, 5, 7).

Obezbediti propisno brisanje čvora.

Implementirati destruktor koji prilikom brisanja uklanja sve ključeve (i pokazivače) iz čvora, ali ne propagira brisanje čvorovima podstabala.

Test i error funkcija

U projektu glavnog programa treba da postoji funkcija `void test();` koja testira rad sa datim klasama. Studenti treba da implementiraju datu funkciju i uslovno je prevode ako makro PROF_TEST nije definisan. Funkcija main treba samo da pozove test funkciju. U svim klasama, funkcija **`test`** treba da bude deklarisana kao prijateljska. Predvideti i uslovno prevoditi funkciju `void errorDetected(string msg)`. Studentska implementacija treba samo da ispiše grešku i prekine izvršavanje programa.

Tehnički zahtevi i smernice za izradu rešenja

Sve klase i metode moraju biti imenovane prema zahtevima iz domaćeg zadatka. Poljima klase, zaštićenim od direktnog pristupa, se pristupa isključivo pomoću predviđenih metoda za čitanje i pisanje vrednosti polja. Nije dozvoljeno dodavanje nijedne javne funkcije/metode koja nije predviđena specifikacijom (privatne uslužne funkcije mogu). Svaka klasa koja koristi dinamičku memoriju mora imati korektno napisan destruktor. Sve metode smestiti u odgovarajuće klase. Programski kod klasa rasporediti u odgovarajuće **.h** i **.cpp** fajlove, pri čemu jedno zaglavlje mora sadržati najviše jednu klasu. Nije dozvoljeno korišćenje globalnih promenljivih za razmenu podataka. U slučaju bilo kakve greške u toku izvršavanja programa, pozvati funkciju `errorDetected`.

VAŽNE NAPOMENE

Za uspešno odbranjen domaći zadatak potrebno je na odbrani pokazati kod podeljen na odgovarajuće projekte, .h i .cpp fajlove.

- Klase kojima su implementirani osnovni koncepti treba da budu smeštene u poseban projekat rešenja koji se prevodi kao statička biblioteka (**`tree.lib`**).
- Glavni program napisati u posebnom projektu koji se prevodi kao Win32 Console Application (**`tree_test.exe`**) i koji treba povezati sa statičkom bibliotekom. Glavni program treba implementirati tako da pozove globalnu funkciju `void test();`.
- NIJE DOZVOLJENO SMESTITI CEO KOD U JEDAN PROJEKAT ILI CPP fajl!