UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Strukture podataka

Projekat

Tuzla, juni 2022.

Sadržaj

Sadržaj **2**

Zadatak 1 **3**

Zadatak 2 **3**

Zadatak 3 **4**

Zadatak 1

Implementirati strukturu podataka Heap pomoću niza kao sto je to objašnjeno na predavanju.
Struktura podataka treba da implementira metode insert, min, max, removeMin te removeMax.
Pored toga, struktura mora imati implementirane default, copy i move konstruktor, copy i move assignment operator, destruktor, te operator << (koristiti inorder prolazak).

Napisati kratki main program koji testira gore navedene funkcionalnosti.

Zadatak 2

Implementirati kontejner UnorderedMap koji implementira funkcionalnosti hash mape.
Sve operacije nad kontejnerom moraju biti O(1).
Kontejner mora imati implementirane naredne operacije:

- Operator []: Dodaje novi element sa kljucem ukoliko element ne postoji, u suprotnom mijenja vrijednost elementa pod proslijeđenim ključem.
- Metod erase() koji uklanja element sa proslijeđenim ključem, vraća true ako je element uspješno obrisan, u suprotnom vraća false.
- Metod emplace(k, v) koji dodaje novi par ključ-vrijednost u kontejner (ukoliko element sa tim ključem ne postoji, u suprotnom metod baca iznimku InvalidArgument)
- Kontejner mora podržavati bidirekcione iteratore sa osnovnim operacijama sa iteratorima: inkrement i dekrement (postfix i prefix forma), operator dereferenciranja i operatori poređenja
- Metod find() koji vraca iterator na element sa proslijeđenim ključem ako postoji, u suprotnom vraca end() iterator.

Problem kolizija riješiti korištenjem algoritma otvorenog hashiranja.
Napomena: Za spremanje elemenata nije potrebno implementirati svoj list kontejner. Dozvoljeno je korištenje std::list u tu svrhu.

Kontejner minimalno mora zadovoljiti sljedeci program:

```
#include "UnorderedMap.hpp"
#include <iostream>

int main(int argc, char* argv[]) {
```

```
  UnorderedMap<std::string, std::string> mapa;
  mapa.emplace("kljuc1", "vrijednost1");
  mapa.emplace("kljuc2", "vrijednost2");
  mapa.emplace("kljuc3", "vrijednost3");
  mapa.emplace("kljuc4", "vrijednost4");
  mapa.emplace("kljuc5", "vrijednost5");
  mapa["kljuc6"] = "vrijednost6";

  for (auto it = mapa.begin(); it != mapa.end(); ++it) {
    std::cout << "Key: " << it->first << ", value: " << it->second <<
std::endl;
  }
  mapa["kljuc1"] = "nova vrijednost1";

  auto it = mapa.find("kljuc1");
  if (it != mapa.end()) {
    std::cout << "Element pronadjen: " << it->second << std::endl;
  } else {
    std::cout << "Element ne postoji" << std::endl;
  }

  bool erased = mapa.erase("kljuc1");

  if(erased){
    std::cout << "Element je izbrisan" << std::endl;
  }
  return 0;
}
```

Zadatak 3

Potrebno je kreirati program koji ce voditi evidenciju o studentima, ispitima i njihovim rezultatima. Kada se program pokrene potrebno je korisniku prikazati meni sa opcijama:

1. Dodaj novog studenta
2. Dodaj novi ispit
3. Dodaj novi predmet
4. Dodaj novi rezultat ispita (indeks studenta, ocjena, datum ispita, ime predmeta) (O(1))
5. Dohvati sve ispite za određeni datum (Dohvatanje: O(1), ispisivanje O(n))
6. Dohvati po datumu i imenu predmeta, sve studente koji su položili određeni ispit (Dohvatanje ispita: O(1))

7. Dohvati po indeksu (O(1)) sve ispite koje je položio student hronološki (O(n)), te prosjecnu ocjenu (O(1))

Imena predmeta su unikatna. Indeksi studenata su unikatni. U jednom danu moze biti vise ispita, ali za jedan predmet u jednom danu može biti samo jedan ispit.
Potrebno je sve eksplicitno nenavedene operacije uciniti sto efikasnijim i brzim u vremenskom i memorijskom aspektu, te demonstrirati razumijevanje DRY (Don't repeat yourself) principa, kao i razdvajanje koda u logičke cjeline.
Dopusteno je koristiti standardnu biblioteku za sve potrebne kontejnere.