STRUKTURE PODATAKA I ALGORITMI

ZBIRKA ZADATAKA

Ishod učenja 6

2024-2025.

Zadatak 1

Koristeći odgovarajuću *hash* tablicu, kreirajte novi tip podataka i napišite program koji korisniku omogućuje:

a) Spremanje naziva knjige pod ključem koji je ISBN knjige (ISBN je kombinacija brojeva koji jedinstveno identificiraju knjigu). Na primjer, korisnik može spremiti naziv knjige "Dina" s ISBN-om "9789531318341". Ako knjiga postoji, samo zamijenite postojeći naziv novim.

b) Ispis naslova knjige prema ISBN-u kojeg je unio korisnik. Ako ne postoji knjiga s tim ISBN-om, prikažite tu činjenicu.

c) Uklanjanje knjige prema ISBN-u kojeg je unio korisnik.

d) Prikaz svih unesenih knjiga i njihovih ISBN brojeva.

Demonstrirati rad svih operacija.

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <algorithm>
using namespace std;

int main() {
	cout << "1 = nova knjiga" << endl;
	cout << "2 = pretraga" << endl;
	cout << "3 = uklanjanje" << endl;
	cout << "4 = prikaz svih knjiga" << endl;
	cout << "X = nova knjiga" << endl;

	unordered_map<string /*isbn*/, string /*naslov*/> knjige;
	char opcija;
	string isbn;
	string naslov;
	unordered_map<string, string>::iterator it;

	do {
		cout << "> ";
		cin >> opcija;
		cin.ignore();

		switch (opcija) {
		case '1':
			cout << "#ISBN# ";
			getline(cin, isbn);
			cout << "#naslov# ";
			getline(cin, naslov);
			knjige[isbn] = naslov;
			break;
		case '2':
			cout << "#ISBN# ";
			getline(cin, isbn);
			it = knjige.find(isbn);
			if (it != knjige.end()) {
				cout << it->second << endl;
			}
			else {
				cout << "Ne postoji knjiga s trazenim ISBN-om" << endl;
			}
			break;
```

```
            case '3':
                  cout << "#ISBN# ";
                  getline(cin, isbn);
                  knjige.erase(isbn);
                  break;
            case '4':
                  for (it = knjige.begin(); it != knjige.end(); ++it) {
                        cout << it->second << " (ISBN: " << it->first << ")" <<
endl;
                  }
                  break;
            }
      } while (opcija != 'X');
}
```

Zadatak 2

Učitajte sva jedinstvena imena iz datoteke **female_names.txt** u *hash* tablicu. Nakon toga izračunajte i ispišite sljedeće podatke:

- Ukupan broj *bucketa*.
- Broj praznih *bucketa*.
- Koliko imena sadrži najveći *bucket*.

Ispišite rezultate na sljedeći način:

```
Ukupno bucketa: 8192
Praznih bucketa: 4463
Najveci bucket sadrzi: 5 imena
```

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_set>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("female_names.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_set<string> ht;
	string ime;

	while (getline(dat, ime)) {
		ht.insert({ ime });
	}

	dat.close();

	int praznih = 0;
	int max = 0;
	for (int i = 0; i < ht.bucket_count(); i++) {
		if (ht.bucket_size(i) == 0) {
			praznih++;
		}
		if (ht.bucket_size(i) > max) {
			max = ht.bucket_size(i);
		}
	}

	cout << "Ukupno bucketa: " << ht.bucket_count() << endl;
	cout << "Praznih bucketa: " << praznih << endl;
	cout << "Najveci bucket sadrzi: " << max << " imena" << endl;
}
```

Zadatak 3

Napišite program koji sprema sva prezimena iz datoteke **family_names.txt** u *hash* tablicu. Omogućite korisniku da unese prvo slovo prezimena, a zatim optimalno prikaže sva prezimena koja počinju tim slovom.

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("family_names.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_multimap<char, string> mm;
	string prezime;

	while (getline(dat, prezime)) {
		mm.insert({ prezime[0], prezime });
	}

	dat.close();

	cout << "Unesite prvo slovo prezimena: ";
	char slovo;
	cin >> slovo;

	auto rez = mm.equal_range(slovo);
	for (auto it = rez.first; it != rez.second; ++it) {
		cout << it->second << endl;
	}
}
```

Zadatak 4

Datoteka **translation_en_es.txt** u svakom retku sadrži jednu englesku riječ i jedan španjolski prijevod (i engleske i španjolske riječi nemaju razmaka u sebi). Napišite program koji sprema sve riječi iz datoteke u *hash* tablicu te omogućuje korisniku da unese englesku riječ i zatim brzo pronađe sve španjolske prijevode te riječi. Nakon toga učitajte englesku riječ od korisnika i ispišite sve španjolske prijevode. Na primjer, ako korisnik unese "was", trebate ispisati "fue", "estaba" i "era".

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("translation_en_es.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_multimap<string /* engleska riječ */, string /* španjolska riječ
*/> mm;
	string eng_rijec;
	string spa_rijec;

	while (dat >> eng_rijec) {
		dat >> spa_rijec;
		mm.insert({ eng_rijec, spa_rijec });
	}

	dat.close();

	cout << "Unesite englesku rijec: ";
	string trazi;
	cin >> trazi;

	auto rez = mm.equal_range(trazi);
	for (auto it = rez.first; it != rez.second; ++it) {
		cout << it->second << endl;
	}
}
```

Zadatak 5

Napišite program koji koristi optimalnu *hash* tablicu kako bi ponudio sljedeće opcije korisniku, onoliko dugo koliko korisnik to želi:

- Učitajte JMBAG od korisnika i spremite ga u rječnik. Ako već postoji, obavijestite korisnika.
- Učitajte JMBAG od korisnika i izbrišite ga iz rječnika na optimalni način. U slučaju da ne postoji, obavijestite korisnika.
- Prikažite sve JMBAG-ove iz rječnika.

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_set>
#include <fstream>
using namespace std;

int main() {
	unordered_set<string> s;

	cout << "1 = Unos novog" << endl;
	cout << "2 = Brisanje" << endl;
	cout << "3 = Ispis" << endl;
	cout << "4 = Kraj" << endl;

	char opcija;
	string jmbag;
	do {
		cout << "> ";
		cin >> opcija;

		switch (opcija) {
		case '1':
			cout << "JMBAG: ";
			cin >> jmbag;
			if (s.find(jmbag) == s.end()) {
				s.insert(jmbag);
				cout << "JMBAG " << jmbag << " je unesen." << endl;
			}
			else {
				cout << "JMBAG " << jmbag << " vec postoji." << endl;
			}
			break;
		case '2':
			cout << "JMBAG: ";
			cin >> jmbag;
			if (s.erase(jmbag) == 1) {
				cout << "JMBAG " << jmbag << " je obrisan." << endl;
			}
			else {
				cout << "JMBAG " << jmbag << " ne postoji." << endl;
			}
			break;
		case '3':
			for (auto it = s.begin(); it != s.end(); ++it) {
				cout << *it << endl;
			}
```

```
                    break;
                }
        } while (opcija != '4');
}
```

Zadatak 6

Napišite program koji koristi optimalnu *hash* tablicu za brojanje i prikaz koliko se puta svaki nukleotid (C, T, G ili A) pojavljuje u genomu Genome_Escherichia coli (koristite datoteku **Genome_Escherichia coli.txt**). Izlaz bi trebao biti sljedeći:

```
A: 1142228
C: 1179554
G: 1176923
T: 1140970
```

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("Genome_Escherichia coli.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_map<char /* nukleotid */, int /* količina */> m;
	char c;

	while (dat >> c) {
		m[c]++;
	}

	dat.close();

	for (auto it = m.begin(); it != m.end(); ++it) {
		cout << it->first << ": " << it->second << endl;
	}
}
```

Zadatak 7

Napišite program koji sprema sva prezimena iz datoteke **family_names.txt** u *hash* tablicu. Omogućite korisniku da unese broj slova prezimena, a zatim optimalno prikaže sva prezimena koja imaju toliko slova.

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("family_names.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_multimap<int, string> mm;
	string prezime;

	while (getline(dat, prezime)) {
		mm.insert({ prezime.size(), prezime });
	}

	dat.close();

	cout << "Unesite broj slova prezimena: ";
	int broj_slova;
	cin >> broj_slova;

	auto rez = mm.equal_range(broj_slova);
	for (auto it = rez.first; it != rez.second; ++it) {
		cout << it->second << endl;
	}
}
```

Zadatak 8

Napišite program koji od korisnika učitava rečenicu, a zatim na optimalni način izračunava koliko puta se koje malo slovo pojavljuje u rečenici (ignorirajte znakove koji nisu mala slova). Na kraju ispišite slova i broj pojavljivanja. Primjerice, ako korisnik unese rečenicu: „Zagreb: glavni grad!“, onda program treba ispisati:

```
v: 1
r: 2
n: 1
l: 1
i: 1
g: 3
e: 1
d: 1
b: 1
a: 3
```

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <fstream>
using namespace std;

bool slovo(char c) {
	return (c >= 'a' and c <= 'z');
}

int main() {
	string recenica;
	cout << "Upisite recenicu: ";
	getline(cin, recenica);

	recenica = "Zagreb: glavni grad!";

	unordered_map<char, int> m;

	for (int i = 0; i < recenica.size(); i++) {
		if (slovo(recenica[i])) {
			m[recenica[i]]++;
		}
	}

	for (auto it = m.begin(); it != m.end(); ++it) {
		cout << it->first << ": " << it->second << endl;
	}
}
```

Zadatak 9

Napišite program koji sprema sve brojeve između 1 i 100 u *hash* tablicu. Nakon toga dopustite korisniku da unese novi broj u rječnik na sljedeći način:

- Ako korisnikov broj već postoji u rječniku, obavijestite korisnika o tome i ne dopustite novi unos.
- Ukoliko korisnički broj ne postoji u rječniku, unesite ga i o tome obavijestite korisnika.

Provjeru postojanja i umetanje napravite na optimalan način i ponavljajte onoliko dugo koliko korisnik želi.

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_set>
#include <fstream>
using namespace std;

int main() {
	unordered_set<int> s;
	for (int i = 1; i <= 100; i++) {
		s.insert(i);
	}

	int broj;
	bool dalje;
	do {
		cout << "Unesite broj: ";
		cin >> broj;

		auto rez = s.insert(broj);
		if (rez.second) {
			cout << "Broj " << broj << " je unesen" << endl;
		}
		else {
			cout << "Broj " << broj << " vec postoji" << endl;
		}

		cout << "Dalje (1/0): ";
		cin >> dalje;
	} while (dalje);
}
```

Zadatak 10

Napišite program koji učitava podatke iz datoteke **students_modules.txt**. U datoteci prvi red sadrži ime i prezime studenta, a drugi red naziv kolegija koji je upisao. Nakon učitavanja, omogućite korisniku da optimalno može pretraživati podatke prema imenu i prezimenu (pa dobiti ispisano sve upisane kolegije) ili prema kolegiju (pa dobiti ispisano sve upisane studente). Primjer rada programa:

```
Zelite li traziti po studentu (1) ili kolegiju (2): 1
> Emily Carter
Upisani kolegiji:
Introduction to Programming
Software Engineering
```

Drugi primjer rada programa:

```
Zelite li traziti po studentu (1) ili kolegiju (2): 2
> Computer Networks
Upisani studenti:
Mason Hughes
Matthew Stewart
Alexander Bryant
```

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("students_modules.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_multimap<string, string> students_to_modules;
	unordered_multimap<string, string> modules_to_students;
	string student;
	string module;

	while (getline(dat, student)) {
		getline(dat, module);
		students_to_modules.insert({ student, module });
		modules_to_students.insert({ module, student });
	}

	dat.close();

	int opcija;
	string upit;

	cout << "Zelite li traziti po studentu (1) ili kolegiju (2): ";
	cin >> opcija;
	cin.ignore();
	cout << "> ";
	getline(cin, upit);
```

```
	pair<unordered_multimap<string, string>::iterator, unordered_multimap<string,
string>::iterator> rezultat;
	if (opcija == 1) {
		rezultat = students_to_modules.equal_range(upit);
		cout << "Upisani kolegiji: " << endl;
	}
	else if (opcija == 2) {
		rezultat = modules_to_students.equal_range(upit);
		cout << "Upisani studenti: " << endl;
	}
	else {
		cout << "Nepoznata opcija" << endl;
	}

	for (auto it = rezultat.first; it != rezultat.second; ++it) {
		cout << it->second << endl;
	}
}
```

Zadatak 11

Napišite program koji učitava podatke iz datoteke **chess_players.txt**. U datoteci prvi red sadrži ime i prezime studenta, drugi red datum, a treći red njegov broj bodova na taj datum (ELO). Za svakog igrača čuvajte samo podatke o najvećem broju bodova i datumu kad je ostvaren, na optimalni način. Omogućite korisniku da upiše ime i prezime igrača, a vi mu ispišite broj bodova i datum. Primjer rada programa:

```
Upisite ime i prezime: Melissa King
Datuma 27.01.2023 imao/la je bodova: 2365
```

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_map>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("chess_players.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_map<string /* ime i prezime */, pair<string /* datum */, int /*
broj bodova */>> data;
	string ime_prezime;
	string datum;
	string broj_bodova;

	while (getline(dat, ime_prezime)) {
		getline(dat, datum);
		getline(dat, broj_bodova);

		auto existing = data.find(ime_prezime);
		if (existing != data.end()) {
			if (existing->second.second < stoi(broj_bodova)) {
				existing->second.first = datum;
				existing->second.second = stoi(broj_bodova);
			}
		}
		else {
			data[ime_prezime] = { datum, stoi(broj_bodova) };
		}
	}

	dat.close();

	string upit;
	cout << "Upisite ime i prezime: ";
	getline(cin, upit);

	auto rez = data.find(upit);
	if (rez != data.end()) {
		cout << "Datuma " << rez->second.first << " imao/la je bodova: " <<
rez->second.second << endl;
```

```
	}
	else {
		cout << "Ne postoji trazeni igrac" << endl;
	}
}
```

Zadatak 12

Napišite program koji učitava podatke iz datoteke **ue5_spawn_log.txt** koja sadrži podatke o *spawnanim* objektima na mapi. U datoteci prvi red sadrži klasu objekta, drugi red jedinstveni identifikator objekta, a treći red trenutak *spawnanja*. Potrebni su vam samo podaci o klasi, ostale zanemarite. Podatke o klasi čuvajte na optimalni način. Omogućite korisniku da upiše naziv klase, a vi mu ispišite koliko *spawnanih* objekata te klase postoji. Primjer rada programa:

```
Upisite klasu: Power_Up
Trazena klasa je instancirana: 9 puta
```

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_set>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("ue5_spawn_log.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_multiset<string> data;
	string klasa;
	string temp;

	while (getline(dat, klasa)) {
		getline(dat, temp);
		getline(dat, temp);
		data.insert(klasa);
	}

	dat.close();

	string upit;
	cout << "Upisite klasu: ";
	getline(cin, upit);

	cout << "Trazena klasa je instancirana: " << data.count(upit) << " puta" <<
endl;
}
```

Zadatak 13

Učitajte sva jedinstvena imena iz datoteke **female_names.txt** u *hash* tablicu. Nakon toga ispišite sve *buckete* i u svakom *bucketu* ispišite imena koja se u njemu nalaze. Ispis treba izgledati ovako:

```
Bucket 0: Timi Janie
Bucket 1:
Bucket 2:
…
Bucket 8187: Arielle
Bucket 8188: Willy Gredel Dorella
Bucket 8189:
Bucket 8190:
Bucket 8191: Lucinda
```

Moguće rješenje:

```
#include <iostream>
#include <string>
#include <unordered_set>
#include <fstream>
using namespace std;

int main() {
	ifstream dat("female_names.txt");
	if (!dat) {
		cout << "Ne mogu otvoriti datoteku" << endl;
		return 1;
	}

	unordered_set<string> ht;
	string ime;

	while (getline(dat, ime)) {
		ht.insert({ ime });
	}

	dat.close();

	for (int i = 0; i < ht.bucket_count(); i++) {
		cout << "Bucket " << i << ": ";
		for (auto it = ht.begin(i); it != ht.end(i); ++it) {
			cout << *it << " ";
		}
		cout << endl;
	}
}
```