STRUKTURE PODATAKA I ALGORITMI

ZBIRKA ZADATAKA

Ishod učenja 5

2024-2025.

Zadatak 1

Učitajte sve riječi iz datoteke **english-adjectives.txt** u listu. Zatim sortirajte riječi u listi od najkraće prema najdužoj, a sve riječi iste duljine sortirajte abecednim redom. Rezultat treba biti:

```
all
any
apt
bad
big
dim
…
impressionable
quintessential
black-and-white
inconsequential
well-documented
```

Moguće rješenje:

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

bool komparator(const string& a, const string& b) {
	if (a.length() < b.length()) {
		return true;
	}
	else if (a.length() > b.length()) {
		return false;
	}
	else {
		return a < b;
	}
}

int main() {
	ifstream dat("english-adjectives.txt");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	list<string> rijeci;
	string ucitana_rijec;

	while (getline(dat, ucitana_rijec)) {
		rijeci.push_back(ucitana_rijec);
	}

	rijeci.sort(komparator);

	for (auto it = rijeci.begin(); it != rijeci.end(); ++it) {
		cout << *it << endl;
	}
```

```
        dat.close();
}
```

Zadatak 2

Učitajte sve riječi iz datoteke **english-adjectives.txt** u vektor. Zatim optimalno prikažite riječ koja je zadnja po abecedi:

```
Abecedno zadnja rijec je: zigzag
```

Moguće rješenje:

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

int main() {
	ifstream dat("english-adjectives.txt");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<string> rijeci;
	string ucitana_rijec;

	while (getline(dat, ucitana_rijec)) {
		rijeci.push_back(ucitana_rijec);
	}

	auto zadnja_rijec = rijeci.end() - 1;

	nth_element(rijeci.begin(), zadnja_rijec, rijeci.end());

	cout << "Abecedno zadnja rijec je: " << *zadnja_rijec << endl;

	dat.close();
}
```

Zadatak 3

Napišite program koji učitava objekte s imenom (stupac "Name") i promjerom (stupac "Diameter (Earth=1)") svih svemirskih tijela iz datoteke **Solar_System_Reduced.csv** u vektor. Nakon toga na najučinkovitiji način ona tijela koja imaju promjer veći od 1 stavite na početak vektora, a ona koja nemaju na kraj vektora. Na kraju, ispišite sva tijela koja imaju promjer veći od 1.

Moguće rješenje:

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

struct Planet {
	string naziv;
	double promjer;
};

bool veci_promjer_predikat(const Planet& p) {
	return p.promjer > 1;
}

int main() {
	ifstream dat("Solar_System_Reduced.csv");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<Planet> planeti;
	string naziv, promjer, temp;
	getline(dat, temp);

	while (getline(dat, naziv, ',')) {
		getline(dat, promjer);
		planeti.push_back({ naziv, stod(promjer) });
	}

	auto partition_point = partition(planeti.begin(), planeti.end(),
veci_promjer_predikat);

	cout << "Tijela koja imaju promjer veci od 1:" << endl;
	for (auto it = planeti.begin(); it != partition_point; ++it) {
		cout << " - " << it->naziv << " (" << it->promjer << ")" << endl;
	}

	dat.close();
}
```

Zadatak 4

Učitajte sve imenice iz datoteke **nounlist.csv** u vektor. Zatim sortirajte imenice rastuće po broju slova, a sve imenice s jednakim brojem slova dodatno sortirajte rastuće abecedno. Rezultat sortiranja treba biti ovakav:

```
ad
id
ox
act
age
aid
aim
air
alb
…
rehospitalization
revascularisation
revascularization
cross-contamination
```

Moguće rješenje:

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

bool komparator(const string& a, const string& b) {
	if (a.length() < b.length()) {
		return true;
	}
	else if (a.length() > b.length()) {
		return false;
	}
	else {
		return a < b;
	}
}

int main() {
	ifstream dat("nounlist.csv");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<string> rijeci;
	string ucitana_rijec;

	while (getline(dat, ucitana_rijec)) {
		rijeci.push_back(ucitana_rijec);
	}

	sort(rijeci.begin(), rijeci.end(), komparator);

	for (auto it = rijeci.begin(); it != rijeci.end(); ++it) {
```

```
            cout << *it << endl;
        }

        dat.close();
}
```

Zadatak 5

Napišite program koji u vektor učitava objekte s nazivom (stupac „Name“) i promjerom (stupac „Diameter (Earth=1)“) svih svemirskih tijela iz datoteke **Solar_System_Reduced.csv**. Nakon toga, na najefikasniji način ispišite nazive tri svemirska tijela koja imaju najveće promjere.

Moguće rješenje:

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

struct Planet {
	string naziv;
	double promjer;
};

bool promjer_desc_komparator(const Planet& a, const Planet& b) {
	return a.promjer > b.promjer;
}

int main() {
	ifstream dat("Solar_System_Reduced.csv");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<Planet> planeti;
	string naziv, promjer, temp;
	getline(dat, temp);

	while (getline(dat, naziv, ',')) {
		getline(dat, promjer);
		planeti.push_back({ naziv, stod(promjer) });
	}

	auto granica_sortiranja = planeti.begin() + 3;
	partial_sort(planeti.begin(), granica_sortiranja, planeti.end(),
promjer_desc_komparator);

	cout << "Tijela koja imaju promjer veci od 1:" << endl;
	for (auto it = planeti.begin(); it != granica_sortiranja; ++it) {
		cout << " - " << it->naziv << " (" << it->promjer << ")" << endl;
	}

	dat.close();
}
```

Zadatak 6

Učitajte sve riječi iz datoteke **english-adjectives.txt** u vektor. Zatim, učitajte neku riječ od korisnika i iskoristite algoritam binarnog pretraživanja da ispišete postoji li tražena riječ ili ne u vektoru, primjerice:

```
Rijec overcooked postoji
Rijec trapav ne postoji
```

Moguće rješenje:

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

int main() {
	ifstream dat("english-adjectives.txt");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<string> rijeci;
	string ucitana_rijec;

	while (getline(dat, ucitana_rijec)) {
		rijeci.push_back(ucitana_rijec);
	}

	string trazimo;
	cout << "Upisite rijec za pretragu: ";
	getline(cin, trazimo);

	sort(rijeci.begin(), rijeci.end());

	if (binary_search(rijeci.begin(), rijeci.end(), trazimo)) {
		cout << "Rijec " << trazimo << " postoji" << endl;
	}
	else {
		cout << "Rijec " << trazimo << " ne postoji" << endl;
	}

	dat.close();
}
```

Zadatak 7

Učitajte sve riječi iz datoteke **english-adjectives.txt** u vektor. Zatim, optimalno pronađite i ispišite tri riječi koje abecedno dolaze posljednje:

```
Tri posljednje rijeci su:
zigzag
zesty
zealous
```

Moguće rješenje:

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

bool abecedno_desc(const string& a, const string& b) {
	return a > b;
}

int main() {
	ifstream dat("english-adjectives.txt");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<string> rijeci;
	string ucitana_rijec;

	while (getline(dat, ucitana_rijec)) {
		rijeci.push_back(ucitana_rijec);
	}

	auto zadnja_rijec = rijeci.begin() + 3;

	partial_sort(rijeci.begin(), zadnja_rijec, rijeci.end(), abecedno_desc);

	cout << "Tri posljednje rijeci su:" << endl;
	for (auto it = rijeci.begin(); it != zadnja_rijec; ++it) {
		cout << *it << endl;
	}

	dat.close();
}
```

Zadatak 8

Napišite program koji u vektor učitava objekte s nazivom (stupac „Name“) i promjerom (stupac „Diameter (Earth=1)“) svih svemirskih tijela iz datoteke **Solar_System_Reduced.csv**. Nakon toga, na najefikasniji način ispišite naziv svemirskog tijela koje ima najmanji promjer.

Moguće rješenje:

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

struct Planet {
	string naziv;
	double promjer;
};

bool promjer_asc_komparator(const Planet& a, const Planet& b) {
	return a.promjer < b.promjer;
}

int main() {
	ifstream dat("Solar_System_Reduced.csv");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<Planet> planeti;
	string naziv, promjer, temp;
	getline(dat, temp);

	while (getline(dat, naziv, ',')) {
		getline(dat, promjer);
		planeti.push_back({ naziv, stod(promjer) });
	}

	nth_element(planeti.begin(), planeti.begin() + 1, planeti.end(),
promjer_asc_komparator);

	cout << "Tijelo koja ima najmanji promjer: " << planeti.front().naziv << " ("
<< planeti.front().promjer << ")" << endl;

	dat.close();
}
```

Zadatak 9

Učitajte sve brojeve iz datoteke **brojevi_10k.txt** u vektor. Zatim napravite sve potrebno tako da korisnik može unijeti broj i binarnim pretraživanjem utvrditi postoji li broj u vektoru ili ne.

Moguće rješenje:

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

int main() {
	ifstream dat("brojevi_10k.txt");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<int> brojevi;
	int ucitani_broj;

	while (dat >> ucitani_broj) {
		brojevi.push_back(ucitani_broj);
	}

	int trazimo;
	cout << "Upisite broj za pretragu: ";
	cin >> trazimo;

	sort(brojevi.begin(), brojevi.end());

	if (binary_search(brojevi.begin(), brojevi.end(), trazimo)) {
		cout << "Broj " << trazimo << " postoji" << endl;
	}
	else {
		cout << "Broj " << trazimo << " ne postoji" << endl;
	}

	dat.close();
}
```

Zadatak 10

Učitajte sve brojeve iz datoteke **brojevi_10k.txt** u vektor. Nakon toga na najučinkovitiji način proste brojeve stavite na početak vektora, a ostale na kraj vektora. Na kraju, ispišite prvo sve proste brojeve, a zatim sve ostale.

Moguće rješenje:

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

bool prost_predikat(const int& broj) {
	for (int i = 2; i < broj; i++) {
		if (broj % i == 0) {
			return false;
		}
	}
	return true;
}

int main() {
	ifstream dat("brojevi_10k.txt");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<int> brojevi;
	int ucitani_broj;

	while (dat >> ucitani_broj) {
		brojevi.push_back(ucitani_broj);
	}

	auto partition_point = partition(brojevi.begin(), brojevi.end(),
prost_predikat);

	cout << "Prosti brojevi: ";
	for (auto it = brojevi.begin(); it != partition_point; ++it) {
		cout << *it << " ";
	}
	cout << endl << endl;

	cout << "Brojevi koji nisu prosti: ";
	for (auto it = partition_point; it != brojevi.end(); ++it) {
		cout << *it << " ";
	}
	cout << endl;

	dat.close();
}
```

Zadatak 11

Učitajte sve riječi iz datoteke **english-adjectives.txt** u vektor. Zatim optimalno prikažite riječ koja je najdulja. Ako ima više riječi iste duljine, prikažite samo onu koja je zadnja po abecedi:

```
Najdulja i abecedno zadnja rijec je: well-documented
```

Moguće rješenje:

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

bool komparator(const string& a, const string& b) {
	if (a.length() < b.length()) {
		return true;
	}
	else if (a.length() > b.length()) {
		return false;
	}
	else {
		return a < b;
	}
}

int main() {
	ifstream dat("english-adjectives.txt");
	if (!dat) {
		cout << "Greska pri otvaranju datoteke" << endl;
		return 1;
	}

	vector<string> rijeci;
	string ucitana_rijec;

	while (getline(dat, ucitana_rijec)) {
		rijeci.push_back(ucitana_rijec);
	}

	auto cilj = rijeci.end() - 1;

	nth_element(rijeci.begin(), cilj, rijeci.end(), komparator);

	cout << "Najdulja i abecedno zadnja rijec je: " << *cilj << endl;

	dat.close();
}
```