Objektno-orijentisano programiranje 2

Individualni projekat

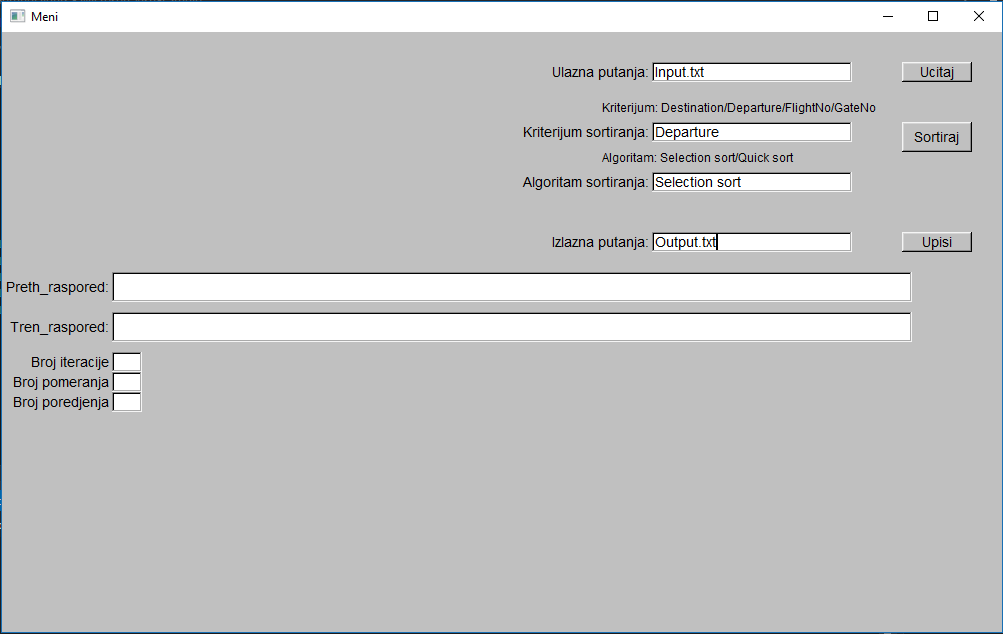
SortingFlights

Bojan Vlasonjić SW76-2016

1. Zadatak projekta i šta je urađeno:

Zadatak projekta je da iz pravilno formirane ulazne datoteke. Potom korisnik bira jedan od dva algoritma sortiranja, kao i jedan od četiri kriterijuma sortiranja. Pri svakoj iteraciji sortiranja se ispisuje trenutno i prethodno stanje elemenata. Sortirani elementi se potom mogu upisati u odgovarajuću izlaznu datoteku u istom formatu kao u ulaznoj datoteci.

1. Rad ulazno-izlaznog podsistema:



**Ulazna putanja** – putanja do ulaznog tekstualnog fajla koji sadrži podatke o letovima. Pre nego što pokrene sortiranje, korisnik mora učitati podatke iz ulazne datoteke.

**Kriterijum sortiranja –** kriterijum po kom ce se izvršiti vizuelizacija sortiranja. Postoje 4 kriterijuma – odrediste, vreme polaska, broj leta i broj izlaza. Korisnik treba da unese pun naziv kriterijuma kako bi pokrenuo sortiranje.

**Algoritam sortiranja-** algoritam po kom ce se izvršiti vizuelizacija sortiranja. Postoje dva algoritma: “Selection sort” i “Quick sort”. Korisnik mora uneti pun naziv algoritma kako bi pokrenuo sortiranje.

**Izlazna putanja** – putanja do izlaznog tekstualnog fajla u kom ce biti upisani letovi(sortirani po odgovarajućem kriterijumu ili nesortirani). Izlazna datoteka je u istom format kao i ulazna.

1. Klase, izuzeci, slobodne funkcije, preklopljeni operatori:

* Klase**:** “Flight”, “MyWindow”, “Sort”, “Selection sort”, “Quick sort”.
* Preklopljeni operatori – “operator>>”, “operator<<”.
* Program ne rukuje izuzecima.
* Slobodne funkcije – void readFile(), void writeToFile(),ostream &operator<<(), int poredjenje(), int poredjenjeVreme(), int proveriKriterijum(), int proveriAlgoritam(), string popuniRaspored()

1. Najbitniji atributi klase i metode, slobodne funkcije - objašnjenja:

* Klasa Flight:

Atributi – odrediste, vreme polaska, broj leta, broj izlaza(uopšteni podaci o letu).

Metoda – compare(int kriterijum, Flight let) – poredi dva leta po zadatom kriterijumu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kriterijum: | 1 | 2 | 3 | 4 |
| Poređenje po: | Odredište | Vreme polaska | Broj leta | Broj izlaza |

* Klasa MyWindow:

Atributi – osnovni elementi grafičke korisničke sprege.

Metode – Za svako dugme imamo veze između dugmeta i akcije, kao i akciju ukoliko se dugme pritisne.

* Klase Sort, Selection sort, Quick sort:

Atributi – broj poredjenja, broj iteracije, broj pomeranja.

Metode – sort(vektor letova, int kriterijum) – implementacija sortiranja,

sortiraj(vektor letova, int kriterijum, stek parova int tipa) – implementacija sortiranja kod Quick sort-a.

* Slobodne funkcije:

void readFile(string putanja, vektor letova) – učitava podatke iz ulaznog fajla u vektor letova.

void writeToFile(string putanja, vekor letova) – upisuje podatke u izlazni tekstualni fajl.

int poredjenje(string s1, string s2) – poredi dva stringa i vraća celobrojnu vrednost kao rezultat poredjenja. Svrha - sortiranje.

string popuniRaspored(vektor letova, int kriterijum) – u zavisnosti od odabranog kriterijuma ispise odgovarajuće podatke svih letova u jednu liniju stringa, i vrati taj string. Svrha – vizuelizacija sortiranja.

1. Struktura ulazne i datoteke:

Ulazna datoteka je u formatu: “**Odrediste; Vreme polaska; Broj leta; Broj izlaza;**”. Izmedju svakog podatka treba da stoji “;” kao i bar jedan razmak, Svi podaci trebaju biti zapisani ili isključivo malim ili isključivo velikim slovima. Ukoliko odredište sadrži više reči, reči ne smeju biti odvojene razmakom. Vreme polaska je u formatu “hh:mm” ili “h:mm”. Broj leta u formatu “ssbbb”( s = slovo, b = broj), a broj izlaza u formatu “sb“. U ulaznoj datoteci se ne sme izostaviti nijedan od podataka inače se neće propisno učitati podaci. Izlazna datoteka je u istom formatu kao i ulazna, ali nema zaglavlje.

1. Opis algoritama za sortiranje:

Implementirana su dva algoritma sortiranja. Kriterijum poređenja zadaje korisnik

**“Selection sort”** – Iteriram kroz vector dvema ugnježdenim “for petljama”. U svakoj iteraciji poredim odgovarajuće elemente vektora i po potrebi im menjam raspored. Pamtim svaku iteraciju, i nakon svake iteracije izlazim iz funkcije zbog vizuelizacije sortiranja.

**“Quick sort”** – Iterativni “quick sort” koristeći stek kao pomoćnu strukturu. Stek sadrži parove “<int, int>“. U stek smeštam početni i krajnji indeks vektora koji sortiram. U svakoj iteraciji vršim particiju elemenata vektora, poredim njihove vrednosti i po potrebi im menjam raspored. Iteracija se zaustavlja ako je stek prazan. Nakon svake iteracije izlazim iz funkcije zbog vizuelizacije sortiranja, a stek prosleđujem kao parametar kako bih očuvao konzistentnost podataka.

1. Napredni OO koncepti:

* Nasleđivanje – apstraktnu klasu Sort nasleđuju klase SelectionSort i QuickSort. Od klase sort nasleđuju atribute: “broj iteracije”, “broj poređenja” i “broj premeštanja”, kao i virtuelnu metodu void sort(). Klase naslednice preklapaju ovu metodu.
* Preklopljeni operatori

“operator>>” – koristim ga kako bih lakše učitavao podatke iz ulaznog fajla. Svakom učitanom podatku uklanja poslednji karakter “;”.

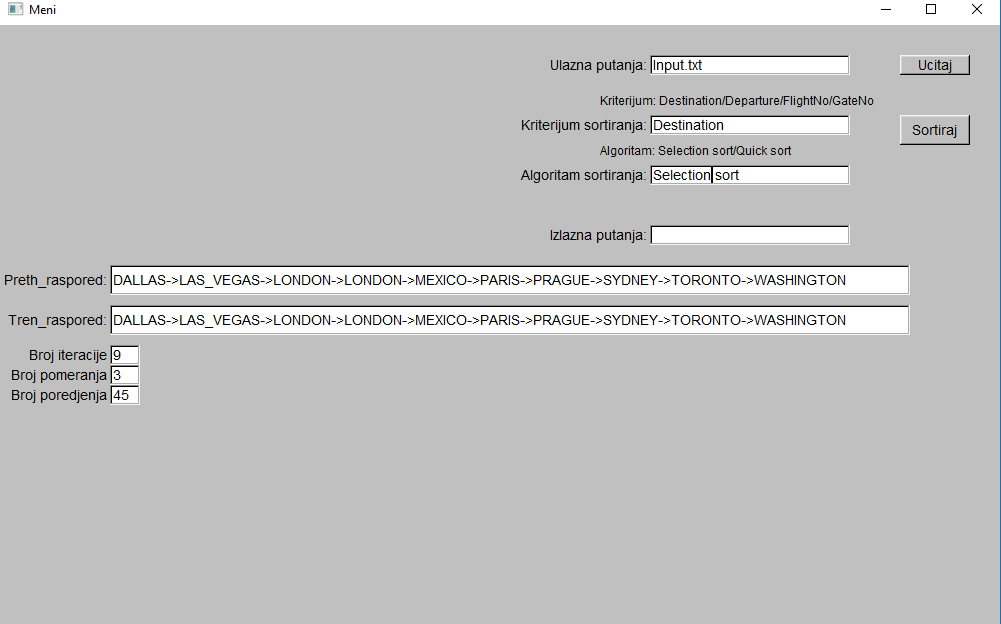
“operator<<” – koristim ga kako bih olakšao upisivanje u datoteku. Upisujemo u istom formatu kao pri učitavanju.

1. Elementi grafičke sprege i funkcionalnosti:

* In\_box ulazPutanja, Button ucitButton
* Nakon klika na dugme uzimamo putanju iz In\_box-a i pozovemo funkciju za učitavanje podataka fajla zadatog ulaznom putanjom.
* Text kriterijumText,Text algoritamText
* Služe kako bi korisnik razumeo koje kriterijume i algoritme sortiranja može da unese.
* Button sortButton, In\_box kriterijum, In\_box algoritam
* Klikom na dugme počinje sortiranje po kriterijumu i algoritmu iz In\_box-a.
* Button upisButton, In\_box izlazPutanja
* Nakon klika na dugme uzimamo putanju iz In\_box-a i pozovemo funkciju za upisivanje podataka u fajl zadat putanjom.
* Button slIterButton, Out\_box prethodniRaspored, Out\_box trenutniRaspored, Out\_box brIter, Out\_box brPoredj, Out\_box brPomer
* Klikom na dugme završava se sledeća iteracija nakon čega se odgovarajući podaci ispišu u svaki Out\_box.

1. Testni slučajevi i izlazni rezultati:

|  |  |  |  |
| --- | --- | --- | --- |
| Odredište | Vreme polaska | Broj leta | Broj izlaza |
| LAS\_VEGAS | 21:15 | AA223 | A3 |
| DALLAS | 21:00 | BA036 | A3 |
| LONDON | 20:30 | AA220 | B4 |
| MEXICO | 20:00 | VI303 | B4 |
| LONDON | 17:45 | BA087 | B4 |
| PARIS | 16:00 | AA342 | A7 |
| PRAGUE | 13:20 | VI309 | F2 |
| TORONTO | 08:30 | QU607 | F2 |
| SYDNEY | 8:20 | AA224 | A7 |
| WASHINGTON | 7:45 | AF342 | A3 |



|  |  |  |
| --- | --- | --- |
| Quick sort | | |
| Broj iteracije | Broj pomeranja | Broj poređenja |
| 1 | 9 | 9 |
| 2 | 7 | 8 |
| 3 | 6 | 6 |
| 4 | 5 | 5 |
| 5 | 2 | 4 |
| 6 | 0 | 1 |
| 7 | 0 | 1 |
| Ukupno: | 29 | 34 |

|  |  |  |
| --- | --- | --- |
| Selection sort | | |
| Broj iteracije | Broj pomeranja | Broj poređenja |
| 1 | 1 | 9 |
| 2 | 0 | 8 |
| 3 | 1 | 7 |
| 4 | 0 | 5 |
| 5 | 2 | 5 |
| 6 | 0 | 4 |
| 7 | 0 | 5 |
| 8 | 1 | 1 |
| 9 | 0 | 1 |
| Ukupno: | 3 | 45 |

**Zaključak** – u ovom primeru Selection sort ima manji broj premeštanja pa se čini da bio brži. Međutim, ima veći broj iteracija. Broj poređenja i pomeranja se kod quick sort-a postepeno smanjuje. U testnom primeru smo imali 10 podataka. Ukoliko bismo imali mnogo veći broj podataka ispostavilo bi se da je quick sort brži zato što bi u tom slučaju imao manji broj iteracija i mnogo manji broj poređenja.

1. Uočeni problemi i ograničenja:

Ulazna datoteka mora ispoštovati navedenu strukturu. U suprotnom sortiranje ne funkcioniše kako bi trebalo. Korisnik ne moze koristiti skraćenice, već mora ukucati pune nazive algoritma sortiranja i kriterijuma sortiranja. Ukoliko u tekstualnom fajlu ima mnogo podataka, neće svi podaci moći da stanu u liniju za prikaz rasporeda.