UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET

Zadaća 5

Tehnike programiranja

Student: **Daris Mujkić**

Grupa: RI1-2b

Broj indeksa: 19413

Ocjena: 2.25

Potpis:

Potpis tutora:

Zadatak . 1.cpp

```
//TP 2022/2023: Zada?a 5, Zadatak 1
#include <iostream>
#include <cmath>
#include <complex>
#include <tuple>
#include <stdexcept>
const double PI=4*std::atan(1);

class Sinusoida{
private:
    double amplituda;
    double frekvencija;
    double faza;
public:
    Sinusoida(double A, double omega, double fi) : amplituda(A),
 frekvencija(omega), faza(fi)
    {
        Normiraj();
    }

    void Normiraj()
    {
        if (amplituda<0)
        {
            amplituda=-amplituda;
            faza+=PI;
        }
        if (frekvencija<0)
        {
            frekvencija=-frekvencija;
            faza=-faza+PI;
        }
        while (faza>PI) faza-=2*PI;
        while (faza<-PI) faza+=2*PI;
    }

    double DajAmplitudu() const {return amplituda;}
    double DajFrekvenciju() const {return frekvencija;}
    double DajFazu() const {return faza;}
    std::tuple<double,double,double> DajParametre() const {return std::
make_tuple(amplituda,frekvencija,faza);}

    Sinusoida& PostaviAmplitudu(double A)
    {
        amplituda=A;
        Normiraj();
        return *this;
    }
    Sinusoida& PostaviFrekvenciju(double omega)
    {
        frekvencija=omega;
        Normiraj();
        return *this;
    }
```

```
    Sinusoida& PostaviFazu(double fi)
    {
        faza=fi;
        Normiraj();
        return *this;
    }
    Sinusoida& PostaviParametre(const std::tuple<double,double,double> &parametri)
    {
        std::tie(amplituda,frekvencija,faza)=parametri;
        Normiraj();
        return *this;
    }

    Sinusoida& operator-()
    {
        amplituda=-amplituda;
        Normiraj();
        return *this;
    }
    Sinusoida& operator+=(const Sinusoida &tmp)
    {
        if (frekvencija!=tmp.frekvencija) throw std::domain_error(
"Razlicite frekvencije");
        //double novaAmplituda=amplituda+tmp.amplituda;
        double re1=amplituda*std::cos(faza);
        double im1=amplituda*std::sin(faza);
        double re2=tmp.amplituda*std::cos(tmp.faza);
        double im2=tmp.amplituda*std::sin(tmp.faza);
        double re=re1+re2;
        double im=im1+im2;
        double novaFaza=std::atan(im/re);
        double novaAmplituda=std::sqrt(re*re+im*im);
        amplituda=novaAmplituda;
        faza=novaFaza;
        Normiraj();
        return *this;
    }
    Sinusoida& operator-=(const Sinusoida &tmp)
    {
        if (frekvencija!=tmp.frekvencija) throw std::domain_error(
"Razlicite frekvencije");
        //double novaAmplituda=amplituda-tmp.amplituda;
        double re1=amplituda*std::cos(faza);
        double im1=amplituda*std::sin(faza);
        double re2=tmp.amplituda*std::cos(tmp.faza);
        double im2=tmp.amplituda*std::sin(tmp.faza);
        double re=re1-re2;
        double im=im1-im2;
        double novaFaza=std::atan(im/re);
        double novaAmplituda=std::sqrt(re*re+im*im);
        amplituda=novaAmplituda;
        faza=novaFaza;
        Normiraj();
        return *this;
    }
    Sinusoida operator*(double skalar) const
```

```
{
    double novaAmplituda=amplituda*skalar;
    return Sinusoida(novaAmplituda,frekvencija,faza);
}
Sinusoida operator*(const Sinusoida &tmp) const
{
    double novaAmplituda=amplituda*tmp.amplituda;
    //double novaFaza=faza+tmp.faza;
    return Sinusoida(novaAmplituda,frekvencija,faza);
}
Sinusoida operator/(double skalar) const
{
    double novaAmplituda=amplituda/skalar;
    return Sinusoida(novaAmplituda,frekvencija,faza);
}
Sinusoida operator/(const Sinusoida &tmp) const
{
    double novaAmplituda=amplituda/tmp.amplituda;
    double novaFaza=faza-tmp.faza;
    return Sinusoida(novaAmplituda,frekvencija,novaFaza);
}
Sinusoida& operator*=(double skalar)
{
    *this=*this * skalar;
    return *this;
}
Sinusoida& operator*=(const Sinusoida &tmp)
{
    *this= *this * tmp;
    return *this;
}
Sinusoida& operator/=(double skalar)
{
    *this=*this / skalar;
    return *this;
}
Sinusoida& operator/=(const Sinusoida &tmp)
{
    *this=*this / tmp;
    return *this;
}

double operator()(double t) const
{
    return amplituda*std::sin(frekvencija*t+faza);
}
double& operator[](const std::string &parametar)
{
    if (parametar=="A") return amplituda;
    else if(parametar=="omega" || parametar=="w") return frekvencija;
    else if (parametar=="phi" || parametar=="fi") return faza;
    else throw std::domain_error("Neispravan naziv parametra");
}
double operator [](const std::string &parametar) const
{
    if (parametar=="A") return amplituda;
```

```
        else if (parametar=="omega" || parametar=="w") return frekvencija;
        else if (parametar=="phi" || parametar=="fi") return faza;
        else throw std::domain_error("Neispravan naziv parametra");
    }
    friend Sinusoida operator+(const Sinusoida &s1, const Sinusoida &s2);
    friend Sinusoida operator*(const Sinusoida &s1, int broj);
    friend Sinusoida operator-(const Sinusoida &s1, const Sinusoida &s2);
};

Sinusoida operator+(const Sinusoida &s1, const Sinusoida &s2)
{
    Sinusoida temp(s1.DajAmplitudu(), s1.DajFrekvenciju(), s1.DajFazu());
//pokusao da fixam sesti AT ovako
    temp+=s2;
    temp.Normiraj();
    return temp;
}
Sinusoida operator*(const Sinusoida &s1, int broj)
{
    Sinusoida temp=s1;
    temp*=broj;
    temp.Normiraj();
    return temp;
}
Sinusoida operator-(const Sinusoida &s1, const Sinusoida &s2)
{
    Sinusoida temp=s1;
    temp-=s2;
    temp.Normiraj();
    return temp;
}

int main ()
{
    Sinusoida s(2.5, 1.0, 7.2);
    std::cout<<"Amplituda: "<<s.DajAmplitudu()<<std::endl;
    std::cout<<"Frekvencija: "<<s.DajFrekvenciju()<<std::endl;
    std::cout<<"Faza: "<<s.DajFazu()<<std::endl;
    s.PostaviAmplitudu(3.0);
    s.PostaviFrekvenciju(2.0);
    s.PostaviFazu(PI/6);
    std::cout<<"Novi parametri: "<<std::endl;
    std::cout<<"Amplituda: "<<s.DajAmplitudu()<<std::endl;
    std::cout<<"Frekvencija: "<<s.DajFrekvenciju()<<std::endl;
    std::cout<<"Faza: "<<s.DajFazu();
    return 0;
    //operatori su mi malo kriticni
}
```

Zadatak .

2.cpp

```
//TP 2022/2023: Zada?a 5, Zadatak 2
#include <iostream>
#include <cmath>
#include <vector>
#include <functional>
#include <stdexcept>
#include <algorithm>

class Padavine{
private:
    std::vector<int> kolicine;
    int maksimalnaKolicina;
public:
    Padavine (int maksimalnaKolicina) : maksimalnaKolicina(maksimalnaKolicina)
    {
        if (maksimalnaKolicina<=0) throw std::range_error(
"Ilegalna maksimalna kolicina");
    }

    void RegistrirajPadavine(int kolicina)
    {
        if (kolicina<0 || kolicina>maksimalnaKolicina) throw std::
range_error("Ilegalna kolicina padavina");
        kolicine.push_back(kolicina);
    }

    int DajBrojRegistriranihPadavina() const {return kolicine.size();}
    void BrisiSve() {kolicine.clear();}

    int DajMinimalnuKolicinuPadavina() const
    {
        if (kolicine.empty()) throw std::range_error("Nema registriranih padavina");
        return *std::min_element(kolicine.begin(), kolicine.end());
    }

    int DajMaksimalnuKolicinuPadavina() const
    {
        if (kolicine.empty()) throw std::range_error("Nema registriranih padavina");
        return *std::max_element(kolicine.begin(), kolicine.end());
    }

    int DajBrojDanaSaPadavinamaVecimOd(int vrijednost) const
    {
        if (kolicine.empty()) throw std::range_error("Nema registriranih padavina");
        return std::count_if(kolicine.begin(), kolicine.end(), std::bind(std::
greater<int>(), std::placeholders::_1, vrijednost));
    }

    void Ispisi() const
    {
        if (kolicine.empty()) throw std::range_error("Nema registriranih padavina");
        std::vector<int> sortiraneKolicine = kolicine;
        std::sort(sortiraneKolicine.begin(), sortiraneKolicine.end(), std::
greater<int>());
```

```
        for (int kolicina : sortiraneKolicine) std::cout<<kolicina<<std::endl;
    }

    int operator[](int index) const
    {
        if (index<1 || index>kolicine.size()) throw std::range_error(
"Neispravan indeks");
        return kolicine[index-1];
    }

    //MOZE LI OVAKO != i == PROVJERITI
    friend bool operator==(const Padavine &p1, const Padavine &p2)
    {
        if(p1.DajBrojRegistriranihPadavina()!=p2.
DajBrojRegistriranihPadavina()) return false;
        if (std::equal(p1.kolicine.begin(), p1.kolicine.end(), p2.kolicine.begin()))
  return true;
    }

    friend bool operator!=(const Padavine &p1, const Padavine &p2)
    {
        if(!(p1==p2)) return true;
        return false;
    }

    Padavine& operator++()
    {
        std::transform(kolicine.begin(), kolicine.end(), kolicine.begin(), std::
bind(std::plus<int>(), std::placeholders::_1,  1));
        maksimalnaKolicina++;
        return *this;
    }

    Padavine operator++(int)
    {
        Padavine kopija(*this);
        ++(*this);
        return kopija;
    }

    Padavine& operator+=(int broj)
    {
        if (!kolicine.empty()) std::transform(kolicine.begin(), kolicine.end(),
 kolicine.begin(), std::bind(std::plus<int>(), std::placeholders::_1, broj));
        return *this;
    }

    Padavine& operator+=(const Padavine &p)
    {
        if (kolicine.size()==p.kolicine.size()) std::transform(kolicine.begin(),
 kolicine.end(), p.kolicine.begin(), kolicine.begin(), std::plus<int>());
        return *this;
    }

    Padavine& operator-=(int broj)
    {
```

```
        if (!kolicine.empty()) std::transform(kolicine.begin(), kolicine.end(),
 kolicine.begin(), std::bind(std::minus<int>(), std::placeholders::_1, broj));
        if(std::count_if(kolicine.begin(), kolicine.end(), std::bind(std::less<int>()
, std::placeholders::_1, 0))) throw std::domain_error(
"Nekorektan rezultat operacije");
        return *this;
    }

    Padavine& operator-=(const Padavine &p)
    {
        if (kolicine.size()==p.kolicine.size()) std::transform(kolicine.begin(),
 kolicine.end(), p.kolicine.begin(), kolicine.begin(), std::minus<int>());
        return *this;
    }

    Padavine& operator-()
    {
        std::vector<int> temp(kolicine.size());
        std::transform(kolicine.begin(), kolicine.end(), temp.begin(), std::bind(
std::minus<int>(), maksimalnaKolicina, std::placeholders::_1));
        kolicine=temp;
        if (std::count_if(kolicine.begin(), kolicine.end(), std::bind(std::less<int>(
), std::placeholders::_1, 0))) throw std::domain_error(
"Nekorektan rezultat operacije");
        return *this;
    }

    friend std::ostream& operator<<(std::ostream& os, const Padavine& padavine)
    {
        if (padavine.kolicine.empty()) os<<"Nema registriranih padavina";
        else for (int kolicina : padavine.kolicine) os<<kolicina<<" ";
        return os;
    }

    friend Padavine operator+(const Padavine &p1, const Padavine &p2)
    {
        Padavine temp=p1;
        temp+=p2;
        return temp;
    }

    friend Padavine operator-(const Padavine &p1, const Padavine &p2)
    {
        Padavine temp=p1;
        temp-=p2;
        return temp;
    }
};


int main ()
{
    try
    {
        Padavine t(10);
        t.RegistrirajPadavine(5);
```

```
        t.RegistrirajPadavine(7);
        t.RegistrirajPadavine(10);
        std::cout<<"Broj registriranih padavina: "<<t.DajBrojRegistriranihPadavina();
        std::cout<<"Minimalna kolicina padavina: "<<t.DajMinimalnuKolicinuPadavina();
        std::cout<<
"Broj dana sa padavinama vecim od 6: "<<t.DajBrojDanaSaPadavinamaVecimOd(6);
        t.Ispisi();
        std::cout<<"Druge padavine: "<<t[2]<<std::endl;
        Padavine T(12);
        T.RegistrirajPadavine(1);
        T.RegistrirajPadavine(11);
        std::cout<<"Maksimalna kolicina padavina: "<<T.
DajMaksimalnuKolicinuPadavina()<<std::endl;
    }
    catch(std::exception e)
    {
        std::cout<<"BELAJ MOJ BEŽE!!! "<<e.what()<<std::endl;
    }


        return 0;
}
```

Zadatak .

3.cpp

```
//TP 2022/2023: Zada?a 5, Zadatak 3
#include <iostream>
#include <cmath>
#include <fstream>
#include <vector>
#include <algorithm>

class ApstraktnoVozilo{
protected:
    int tezina;
public:
    ApstraktnoVozilo(int _tezina) : tezina(_tezina){}
    virtual ~ApstraktnoVozilo(){}
    int DajTezinu() const {return tezina;}
    virtual int DajUkupnuTezinu() const = 0;
    virtual ApstraktnoVozilo* DajKopiju() const = 0;
    virtual void IspisiPodatke() const = 0;
};

class Automobil : public ApstraktnoVozilo{
private:
    std::vector<int> tezinePutnika;
public:
    Automobil(int _tezina, const std::vector<int> &_tezinePutnika) :
 ApstraktnoVozilo(_tezina), tezinePutnika(_tezinePutnika) {}
    int DajUkupnuTezinu() const override
    {
        int ukupnaTezina=tezina;
        for (const auto &tezinaPutnika : tezinePutnika) ukupnaTezina+=tezinaPutnika;
        return ukupnaTezina;
    }
    Automobil* DajKopiju() const override {return new Automobil(*this);}
    void IspisiPodatke() const override
    {
        std::cout<<"Vrsta vozila: Automobil"<<std::endl;
        std::cout<<"Vlastita tezina: "<<tezina<<" kg"<<std::endl;
        std::cout<<"Tezine putnika: ";

//for (const auto &tezinaPutnika : tezinePutnika) std::cout<<tezinaPutnika<<" kg, ";
        for (auto it=tezinePutnika.begin(); it!=tezinePutnika.end(); it++)
        {
            std::cout<<*it<<" kg";
            if (it!=tezinePutnika.end()-1) std::cout<<", ";
        }
        std::cout<<std::endl;
        std::cout<<"Ukupna tezina: "<<DajUkupnuTezinu()<<" kg"<<std::endl;
    }
};

class Kamion : public ApstraktnoVozilo{
private:
    int tezinaTereta;
public:
    Kamion(int _tezina, int _tezinaTereta) : ApstraktnoVozilo(_tezina),
```

```
 tezinaTereta(_tezinaTereta){}
    int DajUkupnuTezinu() const override {return tezina+tezinaTereta;}
    Kamion* DajKopiju() const override {return new Kamion(*this);}
    void IspisiPodatke() const override
    {
        std::cout<<"Vrsta vozila: Kamion"<<std::endl;
        std::cout<<"Vlastita tezina: "<<tezina<<" kg"<<std::endl;
        std::cout<<"Tezina tereta: "<<tezinaTereta<<" kg"<<std::endl;
        std::cout<<"Ukupna tezina: "<<DajUkupnuTezinu()<<" kg"<<std::endl;
    }
};

class Autobus : public ApstraktnoVozilo{
private:
    int brojPutnika;
    int prosjecnaTezinaPutnika;
public:
    Autobus(int _tezina, int _brojPutnika, int
 _prosjecnaTezinaPutnika) : ApstraktnoVozilo(_tezina), brojPutnika(
_brojPutnika), prosjecnaTezinaPutnika(_prosjecnaTezinaPutnika) {}
    int DajUkupnuTezinu() const override {return tezina+brojPutnika*
prosjecnaTezinaPutnika;}
    Autobus* DajKopiju() const override {return new Autobus(*this);}
    void IspisiPodatke() const override
    {
        std::cout<<"Vrsta vozila: Autobus"<<std::endl;
        std::cout<<"Vlastita tezina: "<<tezina<<" kg"<<std::endl;
        std::cout<<"Broj putnika: "<<brojPutnika<<std::endl;
        std::cout<<"Prosjecna tezina putnika: "<<
prosjecnaTezinaPutnika<<" kg"<<std::endl;
        std::cout<<"Ukupna tezina: "<<DajUkupnuTezinu()<<" kg"<<std::endl;
    }
};

/*class Vozilo : public ApstraktnoVozilo{ //BELAJ KOPIRAJUCI KONSTRUKTOR NEGDJE
private:
    ApstraktnoVozilo* vozilo;
    //const ApstraktnoVozilo &vozilo;
public:
    Vozilo(ApstraktnoVozilo* _vozilo) : ApstraktnoVozilo(_vozilo->DajTezinu()), vozilo
(_vozilo) {}
    int DajUkupnuTezinu() const override {return vozilo->DajUkupnuTezinu();}
    Vozilo* DajKopiju() const override {return new Vozilo(*this);}
    void IspisiPodatke() const override {vozilo->IspisiPodatke();}
    //Vozilo(const ApstraktnoVozilo &_vozilo) : ApstraktnoVozilo(_vozilo.DajTezinu()),
 vozilo(_vozilo) {}
    //void IspisiPodatke() const override {vozilo.IspisiPodatke();}
};*/
class Vozilo{
private:
    ApstraktnoVozilo* vozilo;
public:
    Vozilo() : vozilo(nullptr){}
    ~Vozilo() {delete vozilo;}
    Vozilo(const ApstraktnoVozilo &v) : vozilo(v.DajKopiju()) {}
    Vozilo(const Vozilo &v)
```

```
    {
        if (!v.vozilo) vozilo=nullptr;
        else vozilo=v.vozilo->DajKopiju();
    }
    Vozilo(Vozilo &&v) {vozilo=v.vozilo; v.vozilo=nullptr;}
    Vozilo &operator =(const Vozilo &v)
    {
        ApstraktnoVozilo *novi=nullptr;
        if (v.vozilo != nullptr) novi=v.vozilo->DajKopiju();
        delete vozilo;
        vozilo=novi;
        return *this;
    }
    int DajUkupnuTezinu() const {return vozilo->DajUkupnuTezinu();}
    Vozilo* DajKopiju() const {return new Vozilo(*this);}
    void IspisiPodatke() const {vozilo->IspisiPodatke();}
};

int main ()
{
    std::vector<ApstraktnoVozilo*> vozila;
    std::ifstream datoteka("VOZILA.TXT");
    if (!datoteka)
    {
        std::cout<<"Greska pri otvaranju datoteke: VOZILA.TXT"<<std::endl;
        return 1;
    }

    char vrstaVozila;
    while (datoteka>>vrstaVozila)
    {
        int tezinaVozila;
        datoteka>>tezinaVozila;
        if (vrstaVozila=='A')
        {
            int brojPutnika;
            datoteka>>brojPutnika;
            std::vector<int> tezinePutnika(brojPutnika);
            for (auto &tezina: tezinePutnika) datoteka>>tezina;
            vozila.push_back(new Automobil(tezinaVozila, tezinePutnika));
        }
        else if (vrstaVozila=='K')
        {
            int tezinaTereta;
            datoteka>>tezinaTereta;
            vozila.push_back(new Kamion(tezinaVozila, tezinaTereta));
        }
        else if (vrstaVozila=='B')
        {
            int brojPutnika;
            datoteka>>brojPutnika;
            int prosjecnaTezinaPutnika;
            datoteka>>prosjecnaTezinaPutnika;
            vozila.push_back(new Autobus(tezinaVozila,brojPutnika,
prosjecnaTezinaPutnika));
        }
```

```
        else std::cout<<"Neispravan format podataka: "<<vrstaVozila<<
tezinaVozila<<std::endl;
    }
    datoteka.close();

    //Sortiranje po ukupnoj tezini
    std::sort(vozila.begin(), vozila.end(), [](const ApstraktnoVozilo* v1, const
 ApstraktnoVozilo* v2){
        return v1->DajUkupnuTezinu() < v2->DajUkupnuTezinu();
    });

    //Ispis ukupnih tezina nakon sortiranja
    //std::cout<<"Ukupne tezine vozila nakon sortiranja: "<<std::endl;
    for (const auto &vozilo : vozila)
    {
        //vozilo->IspisiPodatke();
        //std::cout<<std::endl;
        std::cout<<vozilo->DajUkupnuTezinu()<<std::endl;
    }
    for (const auto &vozilo : vozila) delete vozilo;

    return 0;

}
```

Zadatak .

4.cpp

```
//TP 2022/2023: Zada?a 5, Zadatak 4
#include <iostream>
#include <cmath>


int main ()
{


        return 0;
}
```

Zadatak .

5.cpp

```
//TP 2022/2023: Zada?a 5, Zadatak 5
#include <iostream>
#include <cmath>
#include <fstream>
#include <algorithm>
#include <vector>
#include <functional>

template<typename TipElemenata>
class DatotecniKontejner{
private:
    std::fstream tok;
public:
    DatotecniKontejner(const std::string &ime_datoteke);
    void DodajNoviElement(const TipElemenata &element);
    int DajBrojElemenata();
    TipElemenata DajElement(int pozicija);
    void IzmijeniElement(int pozicija, const TipElemenata &element);
    void Sortiraj(std::function<bool(const TipElemenata&, const
 TipElemenata&)>kriterij=std::less<TipElemenata>());
};

template <typename TipElemenata>
DatotecniKontejner<TipElemenata>::DatotecniKontejner(const std::string &ime_datoteke)
{
    tok.open(ime_datoteke, std::ios::in | std::ios::out | std::ios::binary);
    if (!tok)
    {
        std::ofstream tmp(ime_datoteke);
        tmp.close();
        tok.open(ime_datoteke, std::ios::in | std::ios::out | std::ios::binary);
        if (!tok) throw std::logic_error(
"Problemi prilikom otvaranja ili kreiranja datoteke");
    }
}

template <typename TipElemenata>
void DatotecniKontejner<TipElemenata>::DodajNoviElement(const TipElemenata &element)
{
    tok.seekp(0, std::ios::end);
    tok.write(reinterpret_cast<const char*>(&element), sizeof(TipElemenata));
    tok.flush();
    tok.seekp(0, std::ios::beg); //pokazivac na pocetak (begin)
}

template <typename TipElemenata>
int DatotecniKontejner<TipElemenata>::DajBrojElemenata()
{
    tok.seekg(0, std::ios::end);
    int velicina=tok.tellg();
    tok.seekg(0, std::ios::beg); //pokazivac na pocetak fajla opet
    //tok.flush();
    return velicina/sizeof(TipElemenata);
}
```

```
template<typename TipElemenata>
TipElemenata DatotecniKontejner<TipElemenata>::DajElement(int pozicija)
{
    int broj_elemenata=DajBrojElemenata();
    if (pozicija<0 || pozicija>=broj_elemenata) throw std::range_error(
"Neispravna pozicija");
    TipElemenata element;
    tok.seekg(pozicija*sizeof(TipElemenata), std::ios::beg);
    tok.read(reinterpret_cast<char*>(&element), sizeof(TipElemenata));
    return element;
}

template<typename TipElemenata>
void DatotecniKontejner<TipElemenata>::IzmijeniElement(int pozicija, const
 TipElemenata &element)
{
    int broj_elemenata=DajBrojElemenata();
    if (pozicija<0 || pozicija>=broj_elemenata) throw std::range_error(
"Neispravna pozicija");
    tok.seekp(pozicija*sizeof(TipElemenata), std::ios::beg);
    tok.write(reinterpret_cast<const char*>(&element), sizeof(TipElemenata));
    tok.flush();
}

template<typename TipElemenata>
void DatotecniKontejner<TipElemenata>::Sortiraj(std::function<bool(const
 TipElemenata&, const TipElemenata&)>kriterij)
{
    int broj_elemenata=DajBrojElemenata();
    if (broj_elemenata<=1) return;

    std::vector<TipElemenata> elementi(broj_elemenata);
//ucitavanje el. iz dat. u vector
    for (int i=0;i<broj_elemenata;i++)
    {
        tok.seekg(i*sizeof(TipElemenata), std::ios::beg);
        tok.read(reinterpret_cast<char*>(&elementi[i]), sizeof(TipElemenata));
    }

    std::sort(elementi.begin(), elementi.end(), kriterij); //sortiranje vektora

    tok.seekp(0, std::ios::beg);
    for (int i=0;i<broj_elemenata;i++) tok.write(reinterpret_cast<const char*>(&
elementi[i]), sizeof(TipElemenata));
}

int main ()
{
    DatotecniKontejner<double> dk("ocjene.bin");
    dk.DodajNoviElement(6);
    dk.DodajNoviElement(10);
    dk.DodajNoviElement(6);
    dk.DodajNoviElement(6);
    dk.DodajNoviElement(7);
    std::cout<<"Broj ocjena: "<<dk.DajBrojElemenata()<<std::endl;
```

```
    std::cout<<"Ocjene: ";
    for (int i=0;i<dk.DajBrojElemenata(); i++) std::cout<<dk.DajElement(i)<<" ";
    dk.Sortiraj();
    std::cout<<"\nSortirane ocjene: ";
    for(int i=0;i<dk.DajBrojElemenata(); i++) std::cout<<dk.DajElement(i)<<" ";
    return 0;
}
```