Programabilni uređaji i objektno orjentisano programiranje

Računske vježbe 11

1. Realizovati šablonsku funkciju kojom se od dva uređena neopadajuća niza formira treći, na isti način uređen niz. Realizovati glavni program koji primjenjuje ovu funkciju nad nizovima cijelih brojeva i nad nizovima tačaka, pri čemu koordinate tačaka mogu biti cijeli ili realni brojevi. Klasu tačka realizovati kao šablonsku klasu kako bi koordinate mogle biti traženog tipa.

```
#include <iostream>
#include <math.h>
using namespace std;

template<class T>
void uredi(T *prvi, int n1,T* drugi, int n2,T *treci)
{
    for(int ia=0,ib=0,ic=0;ia<n1||ib<n2;ic++)
    treci[ic]=(ia==n1)?drugi[ib++]:(ib==n2)?prvi[ia++]:(prvi[ia]<drugi[ib])?prvi[ia++]:drugi[ib++];
}


template<class S>
class sablon
{
       S x;
       S y;
       public:
       sablon(S=0,S=0);
       ~sablon(){};
       bool operator<(const sablon&)const;
       void citaj(){cout<<"("<<x<<","<<y<<")"<<endl;}
};


template<class S>
sablon<S>::sablon(S a,S b):x(a),y(b){}

template<class S>
bool sablon<S>::operator<(const sablon& b)const
{
       return (pow(x,2)+pow(y,2))<(pow(b.x,2)+pow(b.y,2));
}


int main()
{
       int d1[]={1,2,5,7,9};
       int d2[]={3,4,7,8};
       int *d3=new int[9];

       uredi(d1,5,d2,4,d3);

       cout<<"Uredjeni niz je:"<<endl;
       for(int i=0;i<9;i++)cout<<" "<<d3[i];
       cout<<endl;

       delete []d3;

       sablon<int> *p;
       cout<<"Koliko ima tacaka prvi niz?"<<endl;
       int i1;
       cin>>i1;
       cout<<"Unesite niz tacaka sa cjelobrojnim koordinatama"<<endl;
```

```
	p=new sablon<int>[i1];

	int a,b;
	for(int i=0;i<i1;i++) {cin>>a>>b; p[i]=sablon<int>(a,b);}
	cout<<"Koliko ima tacaka drugi niz?";

	int i2;
	cin>>i2;
	cout<<"Unesite niz tacaka sa cjelobrojnim koordinatama"<<endl;
	sablon<int> *q;
	q=new sablon<int>[i2];
	for(int i=0;i<i2;i++) {cin>>a>>b; q[i]=sablon<int>(a,b);}

	int i3;
	i3=i2+i1;

	sablon<int> *r;
	r=new sablon<int>[i3];
	uredi(p,i1,q,i2,r);

	cout<<"Uredjen niz je"<<endl;
	for(int i=0;i<i3;i++) {cout<<" "; r[i].citaj();}
	cout<<endl;

	delete []p;
	delete []q;
	delete []r;

	sablon<float> t1(2.3,4.6),t4(2.5,6.8);
	if(t1<t4)
	{
		cout<<"Tacka ima koordinate: ";
		t1.citaj();
	}
	else
	{
		cout<<"Tacka ima koordinate: ";
		t4.citaj();
	}
}
```

2. Projektovati klasu za obradu vektora (niza) realnih brojeva sa zadatim opsezima indeksa. Za razrješavanje konfliktnih situacija koristiti mehanizam obrade izuzetaka. Napisati glavni program za prikazivanje mogućnosti klase.

```
#include <iostream>
#include<string.h>
using namespace std;

class vektor
{
	private:
		float *niz;
		int min_ops;
		int max_ops;
	public:
		enum greska //Kodovi gresaka
		{OK,
		OPSEG, //neispravan opseg indeksiranja
		MEMORIJA, //dodjela memorije nije uspjela,
		PRAZAN, //vektor je prazan
		INDEKS, //indeks je izvan opsega
		DUZINA}; //neusaglasene duzine vektora
		vektor(){niz=0;}
```

```
            vektor(int, int);
            vektor(const vektor &);
            ~vektor(){delete []niz; niz=0;}
            vektor & operator=(const vektor &);
            float & operator[](int)const;
            friend double operator*(const vektor &, const vektor &);
};


vektor::vektor(int a, int b):min_ops(a),max_ops(b)
{
      if(min_ops>max_ops) throw(OPSEG);
      if(!(niz=new float[max_ops-min_ops+1])) throw MEMORIJA;
      for(int i=0; i<max_ops-min_ops+1; i++) niz[i]=0;
}


vektor::vektor(const vektor &a):min_ops(a.min_ops), max_ops(a.max_ops)
{
      if(!(niz=new float[max_ops-min_ops+1])) throw MEMORIJA;
      for(int i=0;i<max_ops-min_ops+1;i++) niz[i]=a.niz[i];
}


vektor & vektor::operator=(const vektor &a)
{
      if(a.niz==0) throw PRAZAN;
      if(&a!=this)
      {
            delete []niz;
            min_ops=a.min_ops;
            max_ops=a.max_ops;
            if(!(niz=new float[max_ops-min_ops+1])) throw MEMORIJA;
            else for(int i=0;i<max_ops-min_ops+1;i++) niz[i]=a.niz[i];
      }
      return *this;
}


float & vektor::operator[](int i)const
{
      if(!niz) throw PRAZAN; // u niz nije nista upisano, samo je izvršen default konstruktor
      else if(i<min_ops || i>max_ops) throw INDEKS;
      else return niz[i-min_ops];
}


double operator*(const vektor &a, const vektor &b)
{
      if(!a.niz || !b.niz) throw vektor::PRAZAN;
      else if((a.max_ops-a.min_ops) != (b.max_ops-b.min_ops)) throw vektor::DUZINA;
      else
      {
            double s=0;
            for(int i=0; i<a.max_ops-a.min_ops+1; i++)
                  s+=a.niz[i]*b.niz[i];
            return s;
      }
}




int main()
{
```

```
	while(1)
	{
		try
		{
			int maks, minm;
			cout<<"Unesi opseg indeksa prvog niza"<<endl;
			cin>>minm>>maks;
			if(cin.eof()) throw 1;
			vektor v1(minm, maks);
			cout<<"Elementi prvog niza"<<endl;
			for(int i=minm; i<=maks; i++) cin>>v1[i];

			cout<<"Unesi opseg indeksa drugog niza"<<endl;
			cin>>minm>>maks;
			vektor v2(minm, maks);
			cout<<"Elementi drugog niza"<<endl;
			for(int i=minm; i<=maks; i++) cin>>v2[i];

			cout<<"Skalarni proizvod dva zadata niza je "<<v1*v2<<endl;
		}
		catch(vektor::greska g)
		{
			char *poruke[]={" ","Neispravan opseg indeksa!",\
			"Neuspjelo dodjeljivanje memorije!","vektor je prazan!",\
			"Indeks je izvan opsega!", "Neusaglasene duzine vektora"};
			cout<<poruke[g]<<endl;
		}
		catch(...)
		{
			cout<<"Kraj unosa"<<endl;
			break;
		}
	}
}
```