**Strukture podataka – Lab 2**

**Lančane liste**

**Polinom i sabiranje polinoma**

//class.h

class Lancana\_lista

{

double koef;

int exp;

Lancana\_lista\* sledeci;

public:

Lancana\_lista();

Lancana\_lista(double k,int m);

~Lancana\_lista();

void dodaj(double k,int m);

void povezi(Lancana\_lista& p);

double vrati\_vrednost() {return this->koef;}

int vrati\_eksponent(){return this->exp;}

Lancana\_lista\* ukazi\_na\_sledeci(){return sledeci;};

};

class CPolynom

{

int broj\_clanova;

Lancana\_lista\* start;

public:

CPolynom();

~CPolynom();

void Coef(int exp,double coef);

friend CPolynom& Add(CPolynom& poly1,CPolynom& poly2);

CPolynom& operator=(CPolynom& poly);

void stampaj();

double vrati\_vrednost(int stepen);

int vrati\_stepen();

int vrati\_broj\_clanova(){return this->broj\_clanova;}

};

//class.cpp

#include "class.h"

#include <iostream>

using namespace std;

Lancana\_lista::Lancana\_lista()

{

this->koef=0;

this->exp=0;

this->sledeci=NULL;

}

Lancana\_lista::Lancana\_lista(double k,int m)

{

if(k!=0)

this->koef=1;

this->koef=k;

this->exp=m;

this->sledeci=NULL;

}

Lancana\_lista::~Lancana\_lista()

{

delete this->sledeci;

}

void Lancana\_lista::dodaj(double k,int m)

{

if(k!=0)

this->koef=1;

this->koef=k;

this->exp=m;

}

void Lancana\_lista::povezi(Lancana\_lista& p)

{

this->sledeci=&p;

}

CPolynom::CPolynom()

{

this->broj\_clanova=0;

this->start=new Lancana\_lista();

}

CPolynom::~CPolynom()

{

delete start;

}

void CPolynom::Coef(int exp,double coef)

{

if(this->broj\_clanova==0)

{

Lancana\_lista \*x=new Lancana\_lista(coef,exp);

start->povezi(\*x);

this->broj\_clanova++;

}

else

{

Lancana\_lista \*next;

next=start->ukazi\_na\_sledeci();

Lancana\_lista \*before;

before=start;

int i=0;

int k=this->broj\_clanova;

while(i<=k)

{

if(next->vrati\_eksponent()==exp)

{

next->dodaj(coef,exp);

}

else if(next->ukazi\_na\_sledeci()!=NULL&&next->vrati\_eksponent()>exp)

{

before=next;

next=next->ukazi\_na\_sledeci();

}

else if(next!=NULL&&next->vrati\_eksponent()<exp)

{

Lancana\_lista \*x=new Lancana\_lista(coef,exp);

before->povezi(\*x);

x->povezi(\*next);

next=NULL;

this->broj\_clanova++;

}

else

{

Lancana\_lista \*x=new Lancana\_lista(coef,exp);

next->povezi(\*x);

next=NULL;

this->broj\_clanova++;

}

i++;

if(next==NULL) break;

}

}

}

CPolynom& Add(CPolynom& poly1,CPolynom& poly2)

{

CPolynom \*x=new CPolynom;

for(int i=0;i<=max(poly1.vrati\_stepen(),poly2.vrati\_stepen());i++)

{

x->Coef(i,poly1.vrati\_vrednost(i)+poly2.vrati\_vrednost(i));

}

return \*x;

}

void CPolynom::stampaj()

{

Lancana\_lista \*pom;

pom=start->ukazi\_na\_sledeci();

while(pom!=NULL)

{

double x=pom->vrati\_vrednost();

if(pom==start->ukazi\_na\_sledeci()&&pom->vrati\_eksponent()>1)

cout<<x<<"x^"<<pom->vrati\_eksponent()<<" ";

else

{

if(pom==start->ukazi\_na\_sledeci()&&pom->vrati\_eksponent()==1)

{

cout<<x<<"x";

goto kraj;

}

if(pom==start->ukazi\_na\_sledeci()&&pom->vrati\_eksponent()==0)

{

cout<<x;

goto kraj;

}

}

if(pom->vrati\_eksponent()>1&&pom!=start->ukazi\_na\_sledeci())

{

if(x>=0)

cout<<"+"<<x<<"x^"<<pom->vrati\_eksponent();

else

cout<<x<<"x^"<<pom->vrati\_eksponent();

}

if(pom->vrati\_eksponent()==1&&pom!=start->ukazi\_na\_sledeci())

{

if(x>=0)

cout<<"+"<<x<<"x";

else

cout<<x<<"x";

}

if(pom->vrati\_eksponent()==0&&pom!=start->ukazi\_na\_sledeci())

{

if(x>=0)

cout<<"+"<<x;

else

cout<<x;

}

kraj:

cout<<" ";

pom=pom->ukazi\_na\_sledeci();

}

cout<<endl;

}

double CPolynom::vrati\_vrednost(int stepen)

{

Lancana\_lista \*x;

x=start;

while(x!=NULL&&x->vrati\_eksponent()!=stepen)

{

x=x->ukazi\_na\_sledeci();

}

if(x==NULL)

return 0;

else

return x->vrati\_vrednost();

}

int CPolynom::vrati\_stepen()

{

return start->ukazi\_na\_sledeci()->vrati\_eksponent();

}

CPolynom& CPolynom::operator=(CPolynom& poly)

{

this->broj\_clanova=poly.broj\_clanova;

this->start=new Lancana\_lista;

Lancana\_lista\* pom;

pom=start;

for(int i=this->broj\_clanova-1;i>=0;i--)

{

if(poly.vrati\_vrednost(i)!=0)

{

Lancana\_lista\* x=new Lancana\_lista(poly.vrati\_vrednost(i),i);

pom->povezi(\*x);

pom=x;

}

}

return \*this;

}

//main.cpp

#include <iostream>

using namespace std;

#include "class.h"

int main()

{

CPolynom x,y;

x.Coef(1,1);

x.Coef(2,3);

x.Coef(3,5);

cout<<"Prvi polinom:"<<endl;

x.stampaj();

y.Coef(2,1);

y.Coef(7,3);

y.Coef(3,4);

cout<<"Drugi polinom:"<<endl;

y.stampaj();

CPolynom z;

z=Add(x,y);

cout<<"Zbir polinoma:"<<endl;

z.stampaj();

system("pause");

}

**Polinom i množenje polinoma**

//class.h

#include <iostream>

using namespace std;

class Monom

{

double vrednost;

int eksponent;

Monom\* sledeci;

public:

Monom()

{

this->vrednost=0;

this->eksponent=0;

this->sledeci=NULL;

}

Monom(int exp,double value)

{

this->eksponent=exp;

this->vrednost=value;

this->sledeci=NULL;

}

void dodeli(int exp,double value)

{

this->eksponent=exp;

this->vrednost=value;

}

int vrati\_exponent()

{

return this->eksponent;

}

double vrati\_vrednost()

{

return this->vrednost;

}

void ukazi\_na(Monom& m)

{

this->sledeci=&m;

}

Monom\* vrati\_sledeci()

{

return this->sledeci;

}

~Monom()

{

delete this->sledeci;

}

};

class Polynom

{

int stepen;

int broj\_elemenata;

Monom\* start;

public:

Polynom()

{

this->stepen=0;

this->broj\_elemenata=0;

start=new Monom();

}

void Coef(int exp,double value)

{

if(this->broj\_elemenata==0)

{

Monom\* x=new Monom(exp,value);

start->ukazi\_na(\*x);

this->broj\_elemenata++;

}

else

{

Monom\* y=start;

Monom\* x=start->vrati\_sledeci();

while(x!=NULL&&x->vrati\_exponent()>exp)

{

y=x;

x=x->vrati\_sledeci();

}

if(x==NULL)

{

Monom\* z=new Monom(exp,value);

y->ukazi\_na(\*z);

this->broj\_elemenata++;

}

else

{

if(x->vrati\_exponent()==exp)

x->dodeli(exp,value);

else

{

Monom\* z=new Monom(exp,value);

y->ukazi\_na(\*z);

z->ukazi\_na(\*x);

this->broj\_elemenata++;

}

}

}

}

void CoefPlus(int exp,double value)

{

if(this->broj\_elemenata==0)

{

Monom\* x=new Monom(exp,value);

start->ukazi\_na(\*x);

this->broj\_elemenata++;

}

else

{

Monom\* y=start;

Monom\* x=start->vrati\_sledeci();

while(x!=NULL&&x->vrati\_exponent()>exp)

{

y=x;

x=x->vrati\_sledeci();

}

if(x==NULL)

{

Monom\* z=new Monom(exp,value);

y->ukazi\_na(\*z);

this->broj\_elemenata++;

}

else

{

if(x->vrati\_exponent()==exp)

x->dodeli(exp,x->vrati\_vrednost()+value);

else

{

Monom\* z=new Monom(exp,value);

y->ukazi\_na(\*z);

z->ukazi\_na(\*x);

this->broj\_elemenata++;

}

}

}

}

void stampaj()

{

if(start->vrati\_sledeci()!=NULL)

{

if(start->vrati\_sledeci()->vrati\_exponent()==0)

cout<<start->vrati\_sledeci()->vrati\_vrednost();

else

{

if(start->vrati\_sledeci()->vrati\_exponent()==1)

cout<<start->vrati\_sledeci()->vrati\_vrednost()<<"x";

else

{

Monom\* pom=start->vrati\_sledeci();

while(pom!=NULL)

{

if(pom==start->vrati\_sledeci())

cout<<pom->vrati\_vrednost()<<"x^"<<pom->vrati\_exponent();

else

{

if(pom->vrati\_vrednost()>0)

cout<<"+"<<pom->vrati\_vrednost()<<"x^"<<pom->vrati\_exponent();

else

cout<<pom->vrati\_vrednost()<<"x^"<<pom->vrati\_exponent();

}

pom=pom->vrati\_sledeci();

}

}

}

cout<<endl;

}

else

cout<<"Prazan polinom"<<endl;

}

~Polynom()

{

delete start;

}

int vrati\_stepen()

{

return start->vrati\_sledeci()->vrati\_exponent();

}

double vrati\_vrednost(int exp)

{

Monom\* x=start->vrati\_sledeci();

}

friend Polynom& Mul(Polynom& poly1, Polynom& poly2)

{

Polynom\* rez=new Polynom; //Uvek inicijalizovati pokazivac

if(poly1.broj\_elemenata!=0&&poly2.broj\_elemenata!=0)

{

rez->stepen=poly1.stepen+poly2.stepen;

Monom\* x1=poly1.start->vrati\_sledeci();

while(x1!=NULL)

{

Monom\* x2=poly2.start->vrati\_sledeci();

while(x2!=NULL)

{

rez->CoefPlus(x1->vrati\_exponent()+x2->vrati\_exponent(),x1->vrati\_vrednost()\*x2->vrati\_vrednost());

x2=x2->vrati\_sledeci();

}

x1=x1->vrati\_sledeci();

}

}

return \*rez;

}

Polynom& operator=(Polynom& poly)

{

this->broj\_elemenata=poly.broj\_elemenata;

this->stepen=poly.stepen;

this->start=poly.start;

return \*this;

}

};

//main.cpp

#include "class.h"

int main()

{

Polynom x;

x.Coef(1,-2);

x.stampaj();

x.Coef(3,4);

x.stampaj();

Polynom y;

y.Coef(2,5);

y.Coef(3,2);

y.stampaj();

Polynom z;

z=Mul(x,y);

z.stampaj();

x.Coef(6,4);

z=Mul(z,x);

z.stampaj();

system("pause");

return 0;

}

**Sortiranje – Bubble sort**

//class.h

#include <iostream>

using namespace std;

class Element\_liste

{

int broj;

Element\_liste\* sledeci;

public:

Element\_liste()

{

broj=0;

this->sledeci=NULL;

}

Element\_liste(int x)

{

broj=x;

sledeci=NULL;

}

void ukazi\_na(Element\_liste& k)

{

this->sledeci=&k;

}

Element\_liste\* vrati\_sledeci() //Obratiti paznju na \*

{

return sledeci; //Obratiti paznju na povratni tip

}

int vrati\_vrednost()

{

return broj;

}

};

class SortList

{

Element\_liste\* start;

int broj\_elemenata;

public:

SortList()

{

start=new Element\_liste;

broj\_elemenata=0;

}

void dodaj\_na\_pocetak(int k)

{

Element\_liste\* p=new Element\_liste(k);

if(this->broj\_elemenata>0)

{

p->ukazi\_na(\*start->vrati\_sledeci()); //Obratiti paznju na \*

}

start->ukazi\_na(\*p);

this->broj\_elemenata++;

}

void sort(bool rastuci)

{

if(this->broj\_elemenata>1)

{

Element\_liste\* x=start->vrati\_sledeci();

Element\_liste\* zadnji=start->vrati\_sledeci();

Element\_liste\* pom=start;

Element\_liste\* pom2=start;

while(zadnji->vrati\_sledeci()!=NULL&&zadnji->vrati\_sledeci()!=pom2) //Da bismo sprecili da pom napravi problem, koristimo dupli uslov

{

zadnji=zadnji->vrati\_sledeci();

}

for(int i=0;i<this->broj\_elemenata-1;i++)

{

pom=start;

x=start->vrati\_sledeci();

while(x!=zadnji&&x->vrati\_sledeci()!=NULL)

{

if(rastuci==true)

{

if(x->vrati\_vrednost()>x->vrati\_sledeci()->vrati\_vrednost())

{

pom->ukazi\_na(\*x->vrati\_sledeci());

x->ukazi\_na(\*x->vrati\_sledeci()->vrati\_sledeci());

pom->vrati\_sledeci()->ukazi\_na(\*x);

x=pom->vrati\_sledeci();

}

}

else

{

if(x->vrati\_vrednost()<x->vrati\_sledeci()->vrati\_vrednost())

{

pom->ukazi\_na(\*x->vrati\_sledeci());

x->ukazi\_na(\*x->vrati\_sledeci()->vrati\_sledeci());

pom->vrati\_sledeci()->ukazi\_na(\*x);

x=pom->vrati\_sledeci();

}

}

pom=pom->vrati\_sledeci();

x=x->vrati\_sledeci();

if(x->vrati\_sledeci()==zadnji)

pom2=x;

}

zadnji=pom2;

}

}

else

cout<<"Sortiranje nema smisla"<<endl;

}

void stampaj()

{

if(start->vrati\_sledeci()!=NULL)

{

Element\_liste\* x=new Element\_liste;

x=start->vrati\_sledeci();

while(x!=NULL)

{

cout<<x->vrati\_vrednost()<<" ";

x=x->vrati\_sledeci();

}

cout<<endl;

}

else

cout<<"Prazan polinom"<<endl;

}

};

//main.cpp

#include "class.h"

int main()

{

SortList x;

x.dodaj\_na\_pocetak(3);

x.sort(true);

x.dodaj\_na\_pocetak(6);

x.dodaj\_na\_pocetak(1);

x.dodaj\_na\_pocetak(8);

x.dodaj\_na\_pocetak(11);

x.dodaj\_na\_pocetak(5);

x.dodaj\_na\_pocetak(9);

x.stampaj();

x.sort(true);

x.stampaj();

x.sort(false);

x.stampaj();

system("pause");

return 0;

}

**Sortiranje – Insertion sort**

//class.h

#include <iostream>

using namespace std;

class Element

{

int broj;

Element\* sledeci;

public:

Element()

{

broj=0;

sledeci=NULL;

}

Element(int k)

{

broj=k;

sledeci=NULL;

}

Element(int k,Element& m)

{

broj=k;

sledeci=&m;

}

void postavi(int k)

{

broj=k;

}

int vrati\_vrednost()

{

return broj;

}

Element\* vrati\_sledeci()

{

return sledeci;

}

void ukazi\_na(Element& k)

{

sledeci=&k;

}

};

class SortList

{

int broj\_elemenata;

Element\* start;

public:

SortList()

{

broj\_elemenata=0;

start=new Element(0);

}

void dodaj\_na\_pocetak(int k)

{

Element\* x=new Element(k);

if(this->broj\_elemenata==0)

start->ukazi\_na(\*x);

else

{

x->ukazi\_na(\*start->vrati\_sledeci());

start->ukazi\_na(\*x);

}

this->broj\_elemenata++;

}

void stampaj()

{

Element\* x=start->vrati\_sledeci();

while(x!=NULL)

{

cout<<x->vrati\_vrednost()<<" ";

x=x->vrati\_sledeci();

}

cout<<endl;

}

void sort(bool rastuci)

{

if(this->broj\_elemenata>1)

{

Element\* zadnji=start->vrati\_sledeci();

Element\* tekuci=start->vrati\_sledeci()->vrati\_sledeci();

Element\* x=new Element();

Element\* y=new Element();

while(tekuci!=NULL)

{

x=start->vrati\_sledeci();

if(rastuci==true)

{

while(x->vrati\_vrednost()<tekuci->vrati\_vrednost())

{

y=x;

x=x->vrati\_sledeci();

}

if(x!=start->vrati\_sledeci()&&x->vrati\_vrednost()>tekuci->vrati\_vrednost())

{

Element\* pom=new Element();

pom->ukazi\_na(\*x);

y->ukazi\_na(\*tekuci);

zadnji->ukazi\_na(\*tekuci->vrati\_sledeci());

tekuci->ukazi\_na(\*pom->vrati\_sledeci());

tekuci=zadnji->vrati\_sledeci();

}

else

{

if(x==start->vrati\_sledeci()&&start->vrati\_sledeci()->vrati\_vrednost()>=tekuci->vrati\_vrednost())

{

Element\* pom=new Element;

pom->ukazi\_na(\*start->vrati\_sledeci());

start->ukazi\_na(\*tekuci);

zadnji->ukazi\_na(\*tekuci->vrati\_sledeci());

tekuci->ukazi\_na(\*pom->vrati\_sledeci());

tekuci=zadnji->vrati\_sledeci();

}

else

{

zadnji=zadnji->vrati\_sledeci();

tekuci=tekuci->vrati\_sledeci();

}

}

}

else

{

while(x->vrati\_vrednost()>tekuci->vrati\_vrednost())

{

y=x;

x=x->vrati\_sledeci();

}

if(x!=start->vrati\_sledeci()&&x->vrati\_vrednost()<tekuci->vrati\_vrednost())

{

Element\* pom=new Element();

pom->ukazi\_na(\*x);

y->ukazi\_na(\*tekuci);

zadnji->ukazi\_na(\*tekuci->vrati\_sledeci());

tekuci->ukazi\_na(\*pom->vrati\_sledeci());

tekuci=zadnji->vrati\_sledeci(); }

else

{

if(x==start->vrati\_sledeci()&&start->vrati\_sledeci()->vrati\_vrednost()<=tekuci->vrati\_vrednost())

{

Element\* pom=new Element;

pom->ukazi\_na(\*start->vrati\_sledeci());

start->ukazi\_na(\*tekuci);

zadnji->ukazi\_na(\*tekuci->vrati\_sledeci());

tekuci->ukazi\_na(\*pom->vrati\_sledeci());

tekuci=zadnji->vrati\_sledeci();

}

else

{

zadnji=zadnji->vrati\_sledeci();

tekuci=tekuci->vrati\_sledeci();

}

}

}

}

}

}

};

//main.cpp

#include <iostream>

using namespace std;

#include "class.h"

int main()

{

SortList x;

x.dodaj\_na\_pocetak(3);

x.dodaj\_na\_pocetak(6);

x.dodaj\_na\_pocetak(1);

x.stampaj();

x.dodaj\_na\_pocetak(7);

x.dodaj\_na\_pocetak(11);

x.dodaj\_na\_pocetak(5);

x.stampaj();

x.sort(true);

x.stampaj();

x.sort(false);

x.stampaj();

x.dodaj\_na\_pocetak(5);

x.dodaj\_na\_pocetak(9);

x.stampaj();

x.sort(true);

x.stampaj();

x.sort(false);

x.stampaj();

system("pause");

return 0;

}

**Sortiranje – Selection sort**

//class.h

#include <iostream>

using namespace std;

class Element

{

int broj;

Element\* sledeci;

public:

Element()

{

broj=0;

sledeci=NULL;

}

Element(int k)

{

broj=k;

sledeci=NULL;

}

void ukazi\_na(Element& p)

{

sledeci=&p;

}

int vrati\_vrednost()

{

return broj;

}

Element\* vrati\_sledeci()

{

return sledeci;

}

};

class SortList

{

int broj\_elemenata;

Element\* start;

public:

SortList()

{

start=new Element();

broj\_elemenata=0;

}

~SortList()

{

delete start;

}

void dodaj\_na\_pocetak(int k)

{

Element\* pom=new Element(k);

if(this->broj\_elemenata>0)

{

pom->ukazi\_na(\*start->vrati\_sledeci());

}

start->ukazi\_na(\*pom);

this->broj\_elemenata++;

}

void stampaj()

{

if(this->broj\_elemenata>0)

{

Element\* x=start->vrati\_sledeci();

while(x!=NULL)

{

cout<<x->vrati\_vrednost()<<" ";

x=x->vrati\_sledeci();

}

cout<<endl;

}

}

void sort(bool rastuci)

{

if(this->broj\_elemenata>1)

{

Element\* pre\_kraja=start;

Element\* kraj=start->vrati\_sledeci();

Element\* zamena=NULL; //Obavezno inicijalizovati

Element\* pre\_zamene=NULL;

Element\* pom=NULL;

while(kraj->vrati\_sledeci()!=NULL)

{

if(rastuci==true)

{

int min=kraj->vrati\_vrednost();

Element\* tekuci=kraj;

while(tekuci->vrati\_sledeci()!=NULL)

{

if(tekuci->vrati\_sledeci()->vrati\_vrednost()<min)

{

zamena=tekuci->vrati\_sledeci();

pre\_zamene=tekuci;

min=tekuci->vrati\_sledeci()->vrati\_vrednost();

}

tekuci=tekuci->vrati\_sledeci();

}

if(zamena!=NULL)

{

pom=kraj;

pre\_kraja->ukazi\_na(\*zamena);

pre\_zamene->ukazi\_na(\*zamena->vrati\_sledeci());

zamena->ukazi\_na(\*pom);

pre\_kraja=zamena;

kraj=pom;

}

else

{

pre\_kraja=pre\_kraja->vrati\_sledeci();

kraj=kraj->vrati\_sledeci();

}

}

else

{

int max=kraj->vrati\_vrednost();

Element\* tekuci=kraj;

while(tekuci->vrati\_sledeci()!=NULL)

{

if(tekuci->vrati\_sledeci()->vrati\_vrednost()>max)

{

zamena=tekuci->vrati\_sledeci();

pre\_zamene=tekuci;

max=tekuci->vrati\_sledeci()->vrati\_vrednost();

}

tekuci=tekuci->vrati\_sledeci();

}

if(zamena!=NULL)

{

pom=kraj;

pre\_kraja->ukazi\_na(\*zamena);

pre\_zamene->ukazi\_na(\*zamena->vrati\_sledeci());

zamena->ukazi\_na(\*pom);

pre\_kraja=zamena;

kraj=pom;

}

else

{

pre\_kraja=pre\_kraja->vrati\_sledeci();

kraj=kraj->vrati\_sledeci();

}

}

}

}

else

{

cout<<"Sortiranje nema smisla"<<endl;

}

}

};

//main.cpp

#include "class.h"

int main()

{

SortList x;

x.dodaj\_na\_pocetak(4);

x.dodaj\_na\_pocetak(6);

x.stampaj();

x.dodaj\_na\_pocetak(7);

x.stampaj();

x.dodaj\_na\_pocetak(6);

x.dodaj\_na\_pocetak(11);

x.dodaj\_na\_pocetak(2);

x.stampaj();

x.sort(true);

x.stampaj();

x.sort(false);

x.stampaj();

system("pause");

return 0;

}

**Statička lančana lista**

//class.h

#include <iostream>

using namespace std;

class StaticSortList

{

int velicina;

int\* info;

int\* link;

public:

void konstruktor(int k)

{

velicina=k+2;

link=new int[velicina];

for(int i=2;i<velicina-1;i++)

link[i]=i+1;

link[1]=-1;

link[0]=2;

link[velicina-1]=-1;

info=new int[velicina];

}

StaticSortList()

{

konstruktor(20);

}

StaticSortList(int k)

{

konstruktor(k);

}

~StaticSortList()

{

delete [] info;

delete [] link;

}

void dodaj\_na\_pocetak(int k)

{

if(this->link[0]==-1)

cout<<"Nema mesta"<<endl;

else

{

int temp=link[0];

info[temp]=k;

link[0]=link[temp];

link[temp]=link[1];

link[1]=temp;

}

}

void stampaj()

{

for(int i=link[1];i!=-1;i=link[i])

{

cout<<info[i]<<" ";

}

cout<<endl;

}

void sort(bool rastuci)

{

int i=link[1];

while(link[i]!=-1)

{

int j=link[1];

int k=-1;

while(link[link[j]]!=k)

{

if(rastuci==true)

{

if(info[j]>info[link[j]])

{

int pom=info[j];

info[j]=info[link[j]];

info[link[j]]=pom;

}

}

else

{

if(info[j]<info[link[j]])

{

int pom=info[j];

info[j]=info[link[j]];

info[link[j]]=pom;

}

}

if(link[j]==k)

k=j;

j=link[j];

}

i=link[i];

}

}

};

//main.cpp

#include "class.h"

int main()

{

StaticSortList x;

x.dodaj\_na\_pocetak(5);

x.dodaj\_na\_pocetak(7);

x.dodaj\_na\_pocetak(2);

x.stampaj();

x.dodaj\_na\_pocetak(8);

x.dodaj\_na\_pocetak(22);

x.stampaj();

x.sort(true);

x.stampaj();

x.sort(false);

x.stampaj();

system("pause");

return 0;

}

**Statička dvostrana lančana lista**

//class.h

#include <iostream>

using namespace std;

class StaticSortDDList

{

int velicina;

int\* before;

int\* next;

int\* info;

public:

void konstruktor(int k)

{

velicina=k+2;

before=new int[velicina];

next=new int[velicina];

info=new int[velicina];

for(int i=2;i<velicina-1;i++)

next[i]=i+1;

next[velicina-1]=-1;

for(int i=2;i<velicina;i++)

before[i]=i-1;

next[0]=2;

before[0]=-1;

before[2]=0;

next[1]=-1;

before[1]=-1;

next[velicina-1]=-1;

}

StaticSortDDList()

{

konstruktor(20);

}

StaticSortDDList(int k)

{

konstruktor(k);

}

void dodaj\_na\_pocetak(int k)

{

if(next[0]==-1)

cout<<"Ne moze se dodati novi element"<<endl;

else

{

int temp=next[0];

info[temp]=k;

next[1]=temp;

next[0]=next[temp];

if(temp==2)

{

before[2]=1;

next[2]=-1;

}

else

{

before[temp]=1;

next[temp]=temp-1;

before[temp-1]=temp;

}

}

}

void stampaj()

{

int i=1;

while(next[i]!=-1)

{

i=next[i];

cout<<info[i]<<" ";

}

cout<<endl;

}

void sort(bool rastuci)

{

int i=1;

int k=-1;

while(next[i]!=-1)

{

int j=next[1];

while(next[j]!=k)

{

if(rastuci==true)

{

if(info[j]>info[next[j]])

{

int pom=info[j];

info[j]=info[next[j]];

info[next[j]]=pom;

}

}

else

{

if(info[j]<info[next[j]])

{

int pom=info[j];

info[j]=info[next[j]];

info[next[j]]=pom;

}

}

if(next[j]==k)

k=j;

j=next[j];

}

i=next[i];

}

}

};

//main.cpp

#include "class.h"

int main()

{

StaticSortDDList x(5);

x.dodaj\_na\_pocetak(15);

x.dodaj\_na\_pocetak(6);

x.dodaj\_na\_pocetak(11);

x.stampaj();

x.dodaj\_na\_pocetak(32);

x.dodaj\_na\_pocetak(24);

x.stampaj();

x.dodaj\_na\_pocetak(5);

x.stampaj();

x.sort(true);

x.stampaj();

x.sort(false);

x.stampaj();

system("pause");

return 0;

}