\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DinamickiHip.h 17 19

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <iostream>

using namespace std;

template<class T>

class Element

{

private:

T info;

Element<T>\* levi;

Element<T>\* desni;

Element<T>\* roditelj;

public:

Element()

{

levi = desni = roditelj = NULL;

}

Element(T info, Element<T>\* roditelj)

{

this->info = info;

this->roditelj = roditelj;

levi = desni = NULL;

}

template<class T> friend class DinamickiHip;

};

template<class T>

class DinamickiHip

{

public:

Element<T>\* koren;

int brojElemenata;

public:

DinamickiHip()

{

koren = NULL;

brojElemenata = 0;

}

~DinamickiHip()

{

obrisiHip(koren);

}

void obrisiHip(Element<T>\* p)

{

if(p!=NULL)

{

obrisiHip(p->levi);

obrisiHip(p->desni);

delete p;

}

}

void ubaci(T element)

{

if(brojElemenata==0)

{

koren = new Element<T>(element, NULL);

brojElemenata++;

return;

}

Element<T>\* p = koren;

Element<T>\* q;

int b = brojElemenata+1;

char put[30];

int i = 0;

while(b>3)

{

if(b%2==0)

put[i++] = 'l';

else

put[i++] = 'd';

b >>= 1;

}

if(b%2==0)

put[i++] = 'l';

else

put[i++] = 'd';

i--;

while(i>0)

{

if(put[i]=='l')

p = p->levi;

else

p = p->desni;

i--;

}

if(put[i]=='l')

{

p->levi = new Element<T>(element, p);

q = p->levi;

}

else

{

p->desni = new Element<T>(element, p);

q = p->desni;

}

brojElemenata++;

while(p!=NULL)

{

if(q->info > p->info)

{

T pom = q->info;

q->info = p->info;

p->info = pom;

}

else

break;

p = p->roditelj;

q = q->roditelj;

}

}

void inorderPrikaz()

{

inorderPrikaz(koren);

cout << endl;

}

void inorderPrikaz(Element<T>\* p)

{

if(p!=NULL)

{

inorderPrikaz(p->levi);

cout << p->info << " ";

inorderPrikaz(p->desni);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

main.cpp 17 19

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include "DinamickiHip.h"

int main()

{

DinamickiHip<int> dh1;

DinamickiHip<int>\* dh2 = new DinamickiHip<int>();

dh1.ubaci(12);

dh1.ubaci(28);

dh1.ubaci(32);

dh1.ubaci(45);

cout << "Inorder prvog hipa: ";

dh1.inorderPrikaz();

dh2->ubaci(35);

dh2->ubaci(14);

dh2->ubaci(17);

dh2->ubaci(50);

cout << "Inorder drugog hipa: ";

dh2->inorderPrikaz();

dh1.spojiHipove(\*dh2);

delete dh2;

cout << "Inorder spojenih hipova: ";

dh1.inorderPrikaz();

cout << "Inorder MinHip konverzije: ";

dh1.prebaciUMinHip();

dh1.inorderPrikaz();

return 0;

}

***17. Projektovati klasu za rad sa dinamičkim binarnim maxheap-om i implementirati metod za spajanje dva maxheap-a u jedan***.

void spojiHipove(DinamickiHip<T>& dh)

{

ubaciElemente(dh.koren);

}

void ubaciElemente(Element<T>\* p)

{

if(p!=NULL)

{

ubaciElemente(p->levi);

this->ubaci(p->info);

ubaciElemente(p->desni);

}

}

***19. Projektovati klasu za rad sa dinamičkim binarnim maxheap-om i implementirati metod za njegovu konverziju u minheap.***

void prebaciUMinHip()

{

int b = brojElemenata;

T\* a = new T[b];

prebaciUPolje(a, koren);

obrisiHip(koren);

brojElemenata = 0;

koren = NULL;

for(int i=0; i<b; i++)

ubaciUMinHip(a[i]);

}

void prebaciUPolje(T\* a, Element<T>\* p)

{

static int i;

if(p!=NULL)

{

prebaciUPolje(a, p->levi);

a[i++] = p->info;

prebaciUPolje(a, p->desni);

}

}

void ubaciUMinHip(T element)

{

if(brojElemenata==0)

{

koren = new Element<T>(element, NULL);

brojElemenata++;

return;

}

Element<T>\* p = koren;

Element<T>\* q;

int b = brojElemenata+1;

char put[30];

int i = 0;

while(b>3)

{

if(b%2==0)

put[i++] = 'l';

else

put[i++] = 'd';

b >>= 1;

}

if(b%2==0)

put[i++] = 'l';

else

put[i++] = 'd';

i--;

while(i>0)

{

if(put[i]=='l')

p = p->levi;

else

p = p->desni;

i--;

}

if(put[i]=='l')

{

p->levi = new Element<T>(element, p);

q = p->levi;

}

else

{

p->desni = new Element<T>(element, p);

q = p->desni;

}

brojElemenata++;

while(p!=NULL)

{

if(q->info < p->info)

{

T pom = q->info;

q->info = p->info;

p->info = pom;

}

else

break;

p = p->roditelj;

q = q->roditelj;

}

}

////////////////////////////////////////////////////////////////////////////

};