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

StatickoStablo.h \\\\\\\\ parni

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

#include <iostream>

using namespace std;

template<class T>

class StatickoStablo

{

private:

T\*\* cvorovi;

int m; // maksimalni broj cvorova;

int t; // trenutni broj cvorova;

public:

StatickoStablo(int brojNivoa)

{

t = 0;

m = 1;

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

m \*= 2;

m--;

cvorovi = new T\*[m];

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

cvorovi[i] = NULL;

}

~StatickoStablo()

{

if(cvorovi!=NULL)

{

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

{

if(cvorovi[i]!=NULL)

delete cvorovi[i];

}

delete[] cvorovi;

}

}

void ubaci(T element)

{

if(m==t)

throw "Stablo je puno!";

int tekuci = 0;

int prethodni;

while(cvorovi[tekuci]!=NULL && tekuci<m)

{

prethodni = tekuci;

if(\*cvorovi[tekuci] > element)

tekuci = 2\*tekuci+1;

else

if(\*cvorovi[tekuci] < element)

tekuci = 2\*tekuci+2;

else

return;

}

if(tekuci>=m)

return;

if(cvorovi[0]==NULL)

{

cvorovi[0] = new T;

\*cvorovi[0] = element;

t++;

}

else

if(\*cvorovi[prethodni] > element)

{

cvorovi[2\*prethodni+1] = new T;

\*cvorovi[2\*prethodni+1] = element;

t++;

}

else

if(\*cvorovi[prethodni] < element)

{

cvorovi[2\*prethodni+2] = new T;

\*cvorovi[2\*prethodni+2] = element;

t++;

}

}

void prikazi()

{

prikazi(0);

cout << endl;

}

void prikazi(int p)

{

if(p<m && cvorovi[p]!=NULL)

{

prikazi(2\*p+1);

cout << \*cvorovi[p] << " ";

prikazi(2\*p+2);

}

}

***4. Projektovati klasu za rad sa statičkim binarnim stablom i implementirati metod za određivanje visine stabla.***

int visinaStabla() // 4. zadatak

{

return visinaStabla(0);

}

int visinaStabla(int p) // 4. zadatak

{

if(p>=m || cvorovi[p]==NULL)

return 0;

if((2\*p+1>=m || cvorovi[2\*p+1]==NULL) && (2\*p+2>=m || cvorovi[2\*p+2]==NULL))

return 1;

return 1 + max(visinaStabla(2\*p+1), visinaStabla(2\*p+2));

}

***6. Projektovati klasu za rad sa statičkim binarnim stablom i implementirati metod za određivanje momenta stabla***.

int momentStabla() // 6. zadatak

{

return t;

}

***8. Projektovati klasu za rad sa statičkim binarnim stablom i implementirati metod za određivanje težine stabla.***

int tezinaStabla() // 8. zadatak

{

int tezina = 0;

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

{

if(cvorovi[i]!=NULL)

{

if(2\*i+1>=m)

tezina++;

else

if(cvorovi[2\*i+1]==NULL && cvorovi[2\*i+2]==NULL)

tezina++;

}

}

return tezina;

}

***10. Projektovati klasu za rad sa statičkim binarnim stablom i implementirati metod za brisanje svih listova.***

void brisanjeListova() // 10. zadatak

{

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

{

if(cvorovi[i]!=NULL)

{

if(2\*i+1>=m)

{

delete cvorovi[i];

cvorovi[i] = NULL;

}

else

if(cvorovi[2\*i+1]==NULL && cvorovi[2\*i+2]==NULL)

{

delete cvorovi[i];

cvorovi[i] = NULL;

}

}

}

}

***12. Projektovati klasu za rad sa statičkim binarnim stablom i implementirati metod za formiranje mirror kopije datog stabla (slika u ogledalu).***

void mirrorKopija() // 12. zadatak

{

T\*\* pom = new T\*[m];

int i = m-1;

int shift = (m+1)/2;

int max = m;

while(shift>=1)

{

for(int k=0; k<shift; k++)

{

pom[k] = cvorovi[i];

i--;

}

for(int k=0; k<shift; k++)

{

pom[max-1-k] = pom[max-k-shift];

}

max = max-shift;

shift /= 2;

}

for(int k=0; k<m; k++)

cvorovi[k] = pom[k];

delete[] pom;

}

***14. Projektovati klasu za rad sa statičkim binarnim stablom i implementirati metod za spajanje dva uređena stabla. Rezultujuće stablo takođe mora biti uređeno.***

void spajanjeStabala(StatickoStablo<T>& ss) // 14. zadatak

{

for(int i=0; i<ss.m; i++)

if(ss.cvorovi[i]!=NULL)

this->ubaci(\*ss.cvorovi[i]);

}

***16. Projektovati klasu za rad sa statičkim binarnim stablom i implementirati metod za određivanje nivoa sa najviše čvorova.***

int nivoSaNajviseCvorova() // 16. zadatak

{

int bmax = 0;

int b = 0;

int k = 2;

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

{

if(((i+1)%k)!=0)

{

if(cvorovi[i]!=NULL)

b++;

}

else

{

if(b>bmax)

bmax = b;

b = 0;

k \*= 2;

if(cvorovi[i]!=NULL)

b++;

}

}

return bmax;

}

};

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

main.cpp \\\\\\\\ parni

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

#include "StatickoStablo.h"

int main()

{

StatickoStablo<int> ss1(4);

try

{

ss1.ubaci(5);

ss1.ubaci(4);

ss1.ubaci(2);

ss1.ubaci(1);

ss1.ubaci(7);

ss1.ubaci(3);

ss1.ubaci(6);

}

catch(const char\* poruka)

{

cout << poruka << endl;

}

cout << "Inorder stabla: ";

ss1.prikazi();

cout << "Visina stabla je: " << ss1.visinaStabla() << endl;

cout << "Moment stabla je: " << ss1.momentStabla() << endl;

cout << "Tezina stabla je: " << ss1.tezinaStabla() << endl;

ss1.brisanjeListova();

cout << "Inorder stabla posle brisanja listova: ";

ss1.prikazi();

ss1.mirrorKopija();

cout << "Inorder mirror kopije stabla : ";

ss1.prikazi();

StatickoStablo<int> prvoStablo(4);

StatickoStablo<int>\* drugoStablo = new StatickoStablo<int>(3);

prvoStablo.ubaci(5);

prvoStablo.ubaci(4);

prvoStablo.ubaci(2);

prvoStablo.ubaci(1);

prvoStablo.ubaci(7);

prvoStablo.ubaci(3);

prvoStablo.ubaci(6);

drugoStablo->ubaci(8);

drugoStablo->ubaci(1);

drugoStablo->ubaci(10);

drugoStablo->ubaci(9);

cout << "Inorder prvog stabla: ";

prvoStablo.prikazi();

cout << "Inorder drugog stabla: ";

drugoStablo->prikazi();

prvoStablo.spajanjeStabala(\*drugoStablo);

delete drugoStablo;

cout << "Inorder spojenih stabala: ";

prvoStablo.prikazi();

StatickoStablo<int> ss2(4);

ss2.ubaci(5);

ss2.ubaci(4);

ss2.ubaci(2);

ss2.ubaci(1);

ss2.ubaci(7);

ss2.ubaci(3);

ss2.ubaci(6);

ss2.ubaci(8);

cout << "Inorder stabla: ";

ss2.prikazi();

cout << "Nivo sa najvise cvorova je: " << ss2.nivoSaNajviseCvorova() << endl;

return 0;

}