1. Simulacija kolekcija uz klasu Student

#include <iostream>

using namespace std;

template<class T>

class Kolekcija

{

public:

T\* niz;

int max;

int trenutno;

Kolekcija(int max = 10) {

this->max = max;

trenutno = 0;

niz = new T[max];//zbog ovog mora postojati konstrukor u klasi Student koji ne prima parametre

}

~Kolekcija() {

delete[]niz; niz = NULL;

}

bool ukloni() {

if (!prazan()) {

trenutno--;

return true;

}

return false;

}

void dodaj(T el) {

if (trenutno == max) {

max += 5;

T \*temp = new T[max];

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

{

temp[i] = niz[i];

}

delete[]niz;

niz = temp;

}

niz[trenutno] = el;//moramo preklopiti operator = u klasi Student

trenutno++;

}

bool prazan() {

return trenutno == 0;

}

friend ostream &operator <<(ostream &COUT, Kolekcija<T> &obj) {

for (int i = 0; i < obj.trenutno; i++)

{

COUT << obj.niz[i] << endl;//zbog ovog moramo preklopiti operatror << u klasi Student

}

return COUT;

}

};

class Student

{

private:

char \*imePrezime;

int brojIndeksa;

public:

Student() {

char temp[4] = "---";

imePrezime = new char[strlen(temp) + 1];

strcpy\_s(imePrezime, strlen(temp) + 1, temp);

brojIndeksa = 0;

}

Student(char \*\_imePrezime, int \_brojIndeksa) {

imePrezime = new char[strlen(\_imePrezime) + 1];

strcpy\_s(imePrezime, strlen(\_imePrezime) + 1, \_imePrezime);

brojIndeksa = \_brojIndeksa;

}

~Student() { delete[]imePrezime; imePrezime = nullptr; }

Student(Student &obj)

{

imePrezime = new char[strlen(obj.imePrezime) + 1];

strcpy\_s(imePrezime, strlen(obj.imePrezime) + 1, obj.imePrezime);

brojIndeksa = obj.brojIndeksa;

}

Student &operator =(Student & obj) {

imePrezime = new char[strlen(obj.imePrezime) + 1];

strcpy\_s(imePrezime, strlen(obj.imePrezime) + 1, obj.imePrezime);

brojIndeksa = obj.brojIndeksa;

return \*this;

}

friend ostream & operator <<(ostream&COUT, Student &obj) {

COUT << "Ime prezime: " << obj.imePrezime << "\nBroj indeksa: " << obj.brojIndeksa << endl;

return COUT;

}

friend istream & operator >>(istream&CIN, Student &obj) {//Ovaj operator nije bio obavezan za preklopiti

char temp[100];

cout << "Unesite ime i prezime studenta: ";

CIN.getline(temp, 100);

obj.imePrezime = new char[strlen(temp) + 1];

strcpy\_s(obj.imePrezime, strlen(temp) + 1, temp);

cout << "Unesite broj indeksa studenta: ";

CIN >> obj.brojIndeksa;

CIN.ignore();

return CIN;

}

};

void main() {

Student s1, s2, s3;

cin >> s1 >> s2 >> s3;

Kolekcija<Student> studenti(2);

studenti.dodaj(s1);

studenti.dodaj(s2);

studenti.dodaj(s3);//ovdje treba da se proširi kolekcija

cout << studenti << endl<<endl;

studenti.ukloni();

cout << "Studenti nakon uklanjanja zadnjeg:\n";

cout << studenti << endl;

system("pause>0");

}

2.

#include <iostream>

using namespace std;

template<class T>

class Kolekcija

{

public:

T\* niz;

int max;

int trenutno;

Kolekcija(int max = 10) {

this->max = max;

trenutno = 0;

niz = new T[max];//zbog ovog mora postojati konstrukor u klasi Student koji ne prima parametre

}

~Kolekcija() {

delete[]niz; niz = NULL;

}

bool ukloni() {

//uklanja zadnji element iz niza

if (!prazan()) {

trenutno--;

return true;

}

return false;

}

bool ukloni(T el) {

//Uklanja element koji je jednak proslijeđenom

if (prazan())

return false;

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

{

if (niz[i] == el) {

for (int j = i+1; j < trenutno; j++)

{

niz[i] = niz[j];

}

trenutno--;

return true;

}

}

return false;

}

bool ukloni(int pozicija) {

//uklanja element na određenoj poziciji

if (prazan())

return false;

if (pozicija >= trenutno)

return false;

trenutno--;

for (int i = pozicija; i < trenutno; i++)

{

niz[i] = niz[i+1];

}

return true;

}

void dodaj(T el) {

if (trenutno == max) {

max += 5;

T \*temp = new T[max];

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

{

temp[i] = niz[i];

}

delete[]niz;

niz = temp;

}

niz[trenutno] = el;//moramo preklopiti operator = u klasi Student

trenutno++;

}

bool prazan() {

return trenutno == 0;

}

T\* GetNajmanjiElement() {

if (prazan())

return nullptr;

T \*min = new T;

\*min = niz[0];

for (int i = 1; i < trenutno; i++)

{

if (niz[i] < \*min) {

\*min = niz[i];

}

}

return min;

}

friend ostream &operator <<(ostream &COUT, Kolekcija<T> &obj) {

for (int i = 0; i < obj.trenutno; i++)

{

COUT << obj.niz[i] << endl;//zbog ovog moramo preklopiti operatror << u klasi Student

}

return COUT;

}

};

class Zivotinja

{

private:

char \*vrsta;

int tezina;

public:

Zivotinja() {

char temp[4] = "---";

vrsta = new char[strlen(temp) + 1];

strcpy\_s(vrsta, strlen(temp) + 1, temp);

tezina = 0;

}

Zivotinja(char \*\_vrsta, int \_tezina) {

vrsta = new char[strlen(\_vrsta) + 1];

strcpy\_s(vrsta, strlen(\_vrsta) + 1, \_vrsta);

tezina = \_tezina;

}

~Zivotinja() { delete[]vrsta; vrsta = nullptr; }

Zivotinja(Zivotinja &obj)

{

vrsta = new char[strlen(obj.vrsta) + 1];

strcpy\_s(vrsta, strlen(obj.vrsta) + 1, obj.vrsta);

tezina = obj.tezina;

}

Zivotinja &operator =(Zivotinja & obj) {

vrsta = new char[strlen(obj.vrsta) + 1];

strcpy\_s(vrsta, strlen(obj.vrsta) + 1, obj.vrsta);

tezina = obj.tezina;

return \*this;

}

//operator < koji poredi dvije životinje po težini

//operator =

//operator ==

//operator<< koji ispisuje vrstu i težinu životinje

//operator>> koji učitava vrstu i težinu životinje

bool operator==(Zivotinja &drugi) {

if (strcmp(vrsta, drugi.vrsta) == 0 && tezina == drugi.tezina)

return true;

return false;

}

bool operator < (Zivotinja &druga) {

return this->tezina < druga.tezina;

}

friend ostream & operator <<(ostream&COUT, Zivotinja &obj) {

COUT << "Vrsta životinja: " << obj.vrsta << "\nTezina: " << obj.tezina << endl;

return COUT;

}

friend istream & operator >>(istream&CIN, Zivotinja &obj) {//Ovaj operator nije bio obavezan za preklopiti

char temp[100];

cout << "Unesite vrstu životinje: ";

CIN.getline(temp, 100);

obj.vrsta = new char[strlen(temp) + 1];

strcpy\_s(obj.vrsta, strlen(temp) + 1, temp);

cout << "Unesite težinu životinje: ";

CIN >> obj.tezina;

CIN.ignore();

return CIN;

}

};

void main() {

Zivotinja z1, z2, z3, z4;

cin >> z1 >> z2 >> z3>>z4;

Kolekcija<Zivotinja> zivotinje(2);

zivotinje.dodaj(z1);

zivotinje.dodaj(z2);

zivotinje.dodaj(z3);

zivotinje.dodaj(z4);

cout << zivotinje << endl<<endl;

Zivotinja \*min = zivotinje.GetNajmanjiElement();

if (min != nullptr) {

cout << "Životinja sa najmanjom težinom je: ";

cout << \*min << endl;

}

zivotinje.ukloni();

cout << "Nakon poziva funkcije ukloni(): \n";

cout << zivotinje << endl << endl;

zivotinje.ukloni(1);

cout << "Nakon poziva funkcije ukloni(1): \n";

cout << zivotinje << endl << endl;

zivotinje.ukloni(z1);

cout << "Nakon poziva funkcije ukloni(z1): \n";

cout << zivotinje << endl << endl;

delete min; min = nullptr;

system("pause>0");

}