# 1.zadatak

#include<iostream>

using namespace std;

template<class T1, class T2, int max>

class Kolekcija {

T1 \* \_elementi1;

T2 \* \_elementi2;

int \_trenutnoElemenata;

public:

//potrebne funkcije

/\*1. AddElement :: omogucava dodavanje novog elementa u kolekciju. Onemoguciti ponavljanje elemenata\*/

/\*2. RemoveElement :: na osnovu parametra tipa T1 uklanja elemente iz kolekcije.

Prilikom uklanjanja elemenata ocuvati redoslijed njihovog dodavanja. Ukoliko elemenat nije pronađen baciti exception.\*/

int getTrenutno() { }

T1 &getPrvi(int i) { }

T2 &getdrugi(int i) { }

//operator <<

};

class Datum {

int \*\_dan, \*\_mjesec, \*\_godina;

public:

//potrebne funkcije

};

class PolozeniPredmet {

char \* \_nazivPredmeta;

char \* \_imePrezimeNastavnika;

Datum \_datumPolaganja;

int ocjena;

//1. potrebne funkcije

public:

char\* getNastavnika() { }

char\* getNazivPredmeta() { }

Datum getDatum() { }

int getOcjena() { }

};

class Student {

int \_brojIndeksa;

char \* \_imePrezime;

//planom i programom je definisano da je student, tokom studija, duzan poloziti 40 predmeta

Kolekcija<PolozeniPredmet, char\*, 40> \_polozeniPredmeti;//string se odnosi na napomenu

public:

//potrebne funkcije

/\*1.AddPolozeniIspit :: dodaje podatke o polozenom predmetu, te napomenu.

Onemoguciti da se dva puta doda istoimeni predmet, te da se kod istog nastavnika polazu vise od tri

predmeta. Ukoliko nije zadovoljen neki od prethodnih uslova, funkcija treba da baci izuzetak. \*/

/\*2.RemovePolozeniIspit :: na osnovu naziva predmeta, uklanja podatke o polozenom predmetu\*/

/\*3.GetProsjekByNastavnik :: vraca prosjecnu ocjenu koju je student ostvario na predmetima kod nastavnika cije ime je odredjeno

vrijednoscu parametra\*/

/\*4.operator() :: vraca sve polozene predmete koji u napomeni posjeduju sadrzaj koji je proslijedjen kao parametar (npr.listu svih

predmeta koji u napomeni posjeduju sadrzaj "prepisivao"). Ukoliko ne postoji niti jedan predmet sa tom napomenom, baciti izuzetak.\*/

};

void main() {

Datum d1(12, 3, 2019);

Datum d2(15, 2, 2019);

Datum d3(21, 1, 2019);

PolozeniPredmet p1("Programiranje 1", "prof. Denis Music", d1);

PolozeniPredmet p2("Računarske mreže", "prof. Denis Music", d2);

PolozeniPredmet p3("Baze podataka", "prof. Jasmin Azemovic", d3);

PolozeniPredmet p4("Programiranje 2", "prof. Denis Music", d1);

PolozeniPredmet p5("Programiranje 3", "prof. Denis Music", d2);

Student s("Test Test");

try {

char napomena[100];

cout << "Unesite napomenu za prvi predmet: ";

cin.getline(napomena, 100);

s.AddPolozeniIspit(p1, napomena);

cout << "Unesite napomenu za drugi predmet: ";

cin.getline(napomena, 100);

s.AddPolozeniIspit(p2, napomena);

cout << "Unesite napomenu za treci predmet: ";

cin.getline(napomena, 100);

s.AddPolozeniIspit(p3, napomena);

cout << "Unesite napomenu za cetvrti predmet: ";

cin.getline(napomena, 100);

s.AddPolozeniIspit(p4, napomena);

cout << "Unesite napomenu za peti predmet: ";

cin.getline(napomena, 100);

s.AddPolozeniIspit(p5, napomena);//Izuzetak - pokušaj dodavanja više od 3 predmeta za istog profesora

cout << "Unesite napomenu za sesti predmet: ";

cin.getline(napomena, 100);

s.AddPolozeniIspit(p2, napomena);//Izuzetak - pokušaj dodavanja istog predmeta

cout << "::::::::::::::::::::::: STUDENT ::::::::::::::::::::::: \n";

cout << s << endl;

s.RemovePolozeniIspit("Računarske mreže");

s.RemovePolozeniIspit("Programiranje 3");//Izuzetak - pokušaj uklanjanja predmeta koji ne postoji

cout << "::::::::::::::::::::::: STUDENT ::::::::::::::::::::::: \n";

cout << s << endl;

char nastavnik[100];

cout << "Unesite ime i prezime nastavnika za kokeg racunate prosjek: ";

cin.getline(nastavnik, 100);

cout << s.GetProsjekByNastavnik(nastavnik);

PolozeniPredmet \*predmeti = s("prepisivao");

if (predmeti != nullptr)

cout << "Predmeti pronađeni." << endl;

}

catch (exception &ex) {

cout << ex.what() << endl;

}

system("pause>0");

}

2.zadatak  
#include <iostream>

using namespace std;

template <class T1, class T2, int max>

class Kolekcija {

T1\*\_elementi1[max];

T2\*\_elementi2[max];

int \_trenutno;

//potrebne funkcije

public:

void AddElement(T1 prvi, T2 drugi) {

//Onemogućiti dodavanje istih elemenata

}

void RemoveElement(int lokacija) {

}

T1 getPrvi(int i) { }

T2 getDrugi(int i) { }

int getTrenutno() { }

//operator <<

};

class Datum {

int \*\_dan;

int \*\_mjesec;

int \*\_godina;

public:

//potrebne funkcije

};

class Igrac {

char\*\_imePrezime;

Datum \_datumRodjenja;

char\*\_pozicija;

Datum \*\_vazenjeUgovora;

int \_plata;

public:

//potrebne funkcije

char \*getPozicija() {

}

Datum getDatVazenjaUgovora() { }

int getPlata() { }

void setDatumVazenjaUgovora(Datum novi) {

}

};

int br = 0;

class Ekipa {

char \*\_nazivEkipe;

char \*\_drzava;

Kolekcija<Igrac\*, int, 30> \_igraciTima;//igrac i broj koji nosi

public:

/\*konstruktor, destruktor i ostale potrebne funkcije.

AddIgrac::dodaje novog igraca u tim. Zabranjeno dodavanje igraca sa brojem koji nosi vec neki drugi igrac

operator()::vraca igrace koji igraju na poziciji koja je proslijedjena kao parametar

operator-=::na osnovu primljenog parametra(Datum)ukloniti sve igrace kojima je ugovor istekao

GetPlate::vraca mjesecne troskove ekipe(plate igraca)

ProduziUgovorIgracu::ima za cilj produziti ugovor igracu cije je ime proslijedjeno kao parametar do datuma koji

je takodjer proslijedjen kao parametar. Funkcija vraca true/false

operator<<::ispisuje sve vrijednosti objekta Ekipa

\*/

};

void main()

{

Ekipa test("Barcelona", "Spanija");

Datum d1(16, 2, 1990);

Datum d2(1, 1, 2020);

Datum d3(16, 2, 1985);

Datum d4(1, 1, 2019);

Datum d5(16, 2, 1987);

Datum d6(1, 1, 2025);

Datum d7(16, 2, 1984);

Datum d8(1, 1, 2017);

Igrac test1("Neymar", "Napadac", 1000000, d1,d2);

Igrac test2("Suarez", "Napadac", 5000000, d3,d4);

Igrac test3("Messi", "Napadac", 11000000,d5,d6);

Igrac test4("Iniesta", "Veznjak", 7000000,d7,d8);

test.AddIgrac(&test1, 20);

test.AddIgrac(&test2, 11);

test.AddIgrac(&test3, 10);

test.AddIgrac(&test4, 8);

cout << test << endl;

char temp[100];

cout << "Unesite ime i prezime igraca kojem želite produziti ugovor: ";

cin.getline(temp, 100);

test.ProduziUgovorIgracu(temp, d4);

Datum d(1, 1, 2020);

test -= d;

cout << "\n\nNakon operatora -= \n";

cout << test << endl;

cout << "Unesite poziciju koju pretražujete: ";

cin.getline(temp, 100);

Igrac \*pok = test(temp);

cout << "::TEST::BY POZICIJA::" << endl;

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

{

cout << pok[i] << endl;

}

br = 0;

delete[]pok;

cout << "::UKUPNO::PLATA::" << test.getPlate() << endl;

system("pause>0");

}