## Rješenje ovog zadatka se nalazi ispod same postave. Preporuka je da prvo pokušate sami riješiti zadatak.

#include <iostream>

using namespace std;

class Student {

int \_brojIndeksa;

char \* \_imePrezime;

public:

//konstruktore, destruktor

//operator =

//operator == dvije verzije (jedna prima objekat tipa student, druga prima ime i prezime studenta)

// operatore << i >>

};

class PrijaveIspita {

char \* \_nazivPredmeta;

Student \* \_listaPrijavljenih;

int \_trenutno;

int \_max;

public:

//konstruktore, destrukor

/\*

1. konstruktor, destruktor i potrebne funkcije

2. operator+= objekat tipa Student dodaje u listu prijavljenih.

onemoguciti dupliranje prijava (isti student dva puta prijavio jedan

ispit). broj studenata u listi nije ogranicen. To znaci da ako je npr max=3, a pokusa se dodati

4. student, listu treba prosiriti tako da stane i ovaj četvrti student.

3. operator() vraca niz objekata tipa Student koji se u listi prijavljenih

nalaze izmedju lokacija odredjenih vrijednostima parametara(OD-DO).

ukoliko na zahtijevanim lokacijama ne postoje objekti funkcija treba

da vrati NULL

4. operator<< ispisuje vrijednosti svih atributa objekta tipa PrijaveIspita

5. operator[] vraca objekat tipa Student koji posjeduje ime i prezime identicno

vrijednosti parametra operatorske funkcije

\*/

};

void main() {

//na objektu klase PrijaveIspita testirati dostupne funkcionalnosti

//dozvoljeno je mijenjati i prilagođavati main

Student studenti[3];

for (int i = 0; i < 3; i++)

{

cin >> studenti[i];

}

char predmet[100];

int max;

cout << "Unesite naziv predmeta: ";

cin.getline(predmet, 100);

cout << "Koliko se maksimalno prijava moze dodati: ";

cin >> max;

cin.ignore();

PrijaveIspita p(predmet, max);

for (int i = 0; i < 2; i++)

{

p += studenti[i];

}

//pokusaj dodavanja istog studenta

p += studenti[0];

//dodavanje novog, tako da se premasi max po mogucnosti

p += studenti[2];

cout << p << endl;

Student \*pretraga = p(1, 2);

if (pretraga == NULL)

cout << "Nisu pronadjeni studenti u datom opsegu\n";

else {

cout << "Studenti u opsegu su: "<<endl;

for (int i = 0; i < 2; i++)

{

cout << pretraga[i] << endl;

}

}

char temp[100];

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

cin.getline(temp, 100);

Student s = p[temp];

cout << s << endl;

delete pretraga; pretraga = NULL;

system("pause>0");

}

RJEŠENJE

#include <iostream>

using namespace std;

class Student {

int \_brojIndeksa;

char \* \_imePrezime;

public:

//konstruktor, destruktor i potrebne funkcije

Student() {

\_brojIndeksa = 0;

\_imePrezime = new char[4];

strcpy\_s(\_imePrezime, 4, "---");

}

Student(int brojIndeksa, char \*imePrezime) {

\_brojIndeksa = brojIndeksa;

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

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

}

Student(Student &original) {

\_brojIndeksa = original.\_brojIndeksa;

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

strcpy\_s(\_imePrezime, strlen(original.\_imePrezime) + 1, original.\_imePrezime);

}

Student& operator =(Student &original) {

\_brojIndeksa = original.\_brojIndeksa;

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

strcpy\_s(\_imePrezime, strlen(original.\_imePrezime) + 1, original.\_imePrezime);

return \*this;

}

bool operator ==(Student &drugi) {

return (\_brojIndeksa == drugi.\_brojIndeksa);

}

bool operator ==(char \*imePrezime) {

return (strcmp(\_imePrezime,imePrezime)==0);

}

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

COUT << "Student " << s.\_imePrezime << endl;;

COUT << "Broj indeksa: " << s.\_brojIndeksa << endl;

return COUT;

}

friend istream & operator >> (istream &CIN, Student &s) {

char temp[100];

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

CIN.getline(temp, 100);

s.\_imePrezime = new char[strlen(temp) + 1];

strcpy\_s(s.\_imePrezime, strlen(temp) + 1, temp);

cout << "Unesite broj indeksa: ";

CIN >> s.\_brojIndeksa;

CIN.ignore();

return CIN;

}

~Student() {

delete[]\_imePrezime; \_imePrezime = nullptr;

}

};

class PrijaveIspita {

char \* \_nazivPredmeta;

Student \* \_listaPrijavljenih;

int \_trenutno;

int \_max;

public:

PrijaveIspita() {

\_nazivPredmeta = new char[4];

strcpy\_s(\_nazivPredmeta, 4, "---");

\_max = 0;

\_trenutno = 0;

\_listaPrijavljenih = nullptr;

}

PrijaveIspita(char \*nazivPredmeta, int max) {

\_nazivPredmeta = new char[strlen(nazivPredmeta)+1];

strcpy\_s(\_nazivPredmeta, strlen(nazivPredmeta) + 1, nazivPredmeta);

\_max = max;

\_trenutno = 0;

\_listaPrijavljenih = new Student[\_max];

}

PrijaveIspita(PrijaveIspita &original) {

\_nazivPredmeta = new char[strlen(original.\_nazivPredmeta) + 1];

strcpy\_s(\_nazivPredmeta, strlen(original.\_nazivPredmeta) + 1, original.\_nazivPredmeta);

\_trenutno = original.\_trenutno;

\_max = original.\_max;

\_listaPrijavljenih = new Student[\_max];

for (int i = 0; i < \_trenutno; i++)

{

\_listaPrijavljenih[i] = original.\_listaPrijavljenih[i];

}

}

~PrijaveIspita() {

delete[]\_nazivPredmeta; \_nazivPredmeta = nullptr;

delete[]\_listaPrijavljenih; \_listaPrijavljenih = nullptr;

}

bool operator += (Student novi) {

for (int i = 0; i < \_trenutno; i++)

{

if(\_listaPrijavljenih[i]==novi){

cout << "Student je već prijavio ispit!" << endl;

return false;

}

}

if (\_trenutno >= \_max) {

//proširivanje niza

Student \*novi = new Student[\_trenutno];

for (int i = 0; i < \_trenutno; i++)

{

novi[i] = \_listaPrijavljenih[i];

}

delete[]\_listaPrijavljenih;

\_max++;

\_listaPrijavljenih=new Student[\_max];

for (int i = 0; i < \_trenutno; i++)

{

\_listaPrijavljenih[i] = novi[i];

}

delete[]novi;

}

\_listaPrijavljenih[\_trenutno] = novi;

\_trenutno++;

return true;

}

Student\* operator()(int \_od, int \_do) {

if (\_trenutno == 0 || \_od<0 || \_do>\_trenutno || \_od>\_trenutno || \_do<0 || \_do<\_od)

return NULL;

Student \*niz = new Student[\_do - \_od + 1];

int j = 0;

for (int i = \_od; i <= \_do; i++)

{

niz[j] = \_listaPrijavljenih[i];

}

return niz;

}

friend ostream & operator <<(ostream &COUT, PrijaveIspita &p) {

COUT << "Ispit iz predmeta " << p.\_nazivPredmeta << endl;;

COUT << "PRIJAVE: " << endl;

for (int i = 0; i < p.\_trenutno; i++)

{

COUT << p.\_listaPrijavljenih[i] << endl;

}

return COUT;

}

Student &operator [](char \*imePrezime) {

Student s;

for (int i = 0; i < \_trenutno; i++)

{

if (\_listaPrijavljenih[i] == imePrezime)

return \_listaPrijavljenih[i];

}

return s;

}

/\*

1. konstruktor, destruktor i potrebne funkcije

2. operator+= objekat tipa Student dodaje u listu prijavljenih.

onemoguciti dupliranje prijava (isti student dva puta prijavio jedan

ispit). broj studenata u listi nije ogranicen

3. operator() vraca niz objekata tipa Student koji se u listi prijavljenih

nalaze izmedju lokacija odredjenih vrijednostima parametara(OD-DO).

ukoliko na zahtijevanim lokacijama ne postoje objekti funkcija treba

da vrati NULL

4. operator<< ispisuje vrijednosti svih atributa objekta tipa PrijaveIspita

5. operator[] vraca objekat tipa Student koji posjeduje ime i prezime identicno

vrijednosti parametra operatorske funkcije

\*/

};

void main() {

//na objektu klase PrijaveIspita testirati dostupne funkcionalnosti

//izbjegavati koristenje vlastitog imena i prezimena

Student studenti[3];

for (int i = 0; i < 3; i++)

{

cin >> studenti[i];

}

char predmet[100];

int max;

cout << "Unesite naziv predmeta: ";

cin.getline(predmet, 100);

cout << "Koliko se maksimalno prijava moze dodati: ";

cin >> max;

cin.ignore();

PrijaveIspita p(predmet, max);

for (int i = 0; i < 2; i++)

{

p += studenti[i];

}

//pokusaj dodavanja istog studenta

p += studenti[0];

//dodavanje novog, tako da se premasi max po mogucnosti

p += studenti[2];

cout << p << endl;

Student \*pretraga = p(1, 2);

if (pretraga == NULL)

cout << "Nisu pronadjeni studenti u datom opsegu\n";

else {

cout << "Studenti u opsegu su: "<<endl;

for (int i = 0; i < 2; i++)

{

cout << pretraga[i] << endl;

}

}

char temp[100];

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

cin.getline(temp, 100);

Student s = p[temp];

cout << s << endl;

delete pretraga; pretraga = NULL;

system("pause>0");

}