#include <iostream>

using namespace std;

class PolozeniPredmet {

char\* Naziv;

char\* Profesor;

int ocjena;

public:

PolozeniPredmet(char\* prof, char\* n, int ocj) {

Naziv = new char[strlen(n) + 1];

strcpy\_s(Naziv, strlen(n) + 1, n);

Profesor = new char[strlen(prof) + 1];

strcpy\_s(Profesor, strlen(prof) + 1, prof);

ocjena = ocj;

}

~PolozeniPredmet() {

delete[] Profesor; Profesor = nullptr;

delete[] Naziv; Naziv = nullptr;

}

PolozeniPredmet(PolozeniPredmet &orig) {

Naziv = new char[strlen(orig.Naziv) + 1];

strcpy\_s(Naziv, strlen(orig.Naziv) + 1, orig.Naziv);

Profesor = new char[strlen(orig.Profesor) + 1];

strcpy\_s(Profesor, strlen(orig.Profesor) + 1, orig.Profesor);

ocjena = orig.ocjena;

}

void operator =(PolozeniPredmet &orig) {

if (this != &orig) {

delete[]Naziv; delete[]Profesor;

Naziv = new char[strlen(orig.Naziv) + 1];

strcpy\_s(Naziv, strlen(orig.Naziv) + 1, orig.Naziv);

Profesor = new char[strlen(orig.Profesor) + 1];

strcpy\_s(Profesor, strlen(orig.Profesor) + 1, orig.Profesor);

ocjena = orig.ocjena;

}

}

int getOcjena() { return ocjena; }

bool operator ==(PolozeniPredmet &pred) {

return (strcmp(Naziv, pred.Naziv) == 0 && strcmp(Profesor, pred.Profesor) == 0);

}

friend ostream & operator << (ostream &out, PolozeniPredmet &pred) {

out << "Predmet: " << pred.Naziv << endl;

out << "Profesor: " << pred.Profesor << endl;

out << "Ocjena: " << pred.ocjena << endl;

return out;

}

};

ostream & operator << (ostream &out, PolozeniPredmet &pred);

class Clanstvo {

char brojKartice[10];

bool ClanarinaPlacena;

int dug;

public:

Clanstvo(char\* brojK = "---") {

ClanarinaPlacena = false;

strncpy\_s(brojKartice, 10, brojK, \_TRUNCATE);

dug = 1500;

}

Clanstvo(Clanstvo &obj) {

strcpy\_s(brojKartice, 10, obj.brojKartice);

ClanarinaPlacena = obj.ClanarinaPlacena;

dug = obj.dug;

}

int uplataClanarine(int vrijednost) {

if (ClanarinaPlacena == true)

{

throw exception("Clanarina je već plaćena.");

}

if (vrijednost > dug)

{

dug = 0;

cout << "Iznos kusura: " << vrijednost - dug << " KM." << endl;

return 0;

}

dug -= vrijednost;

return dug;

}

void ClanstvoInfo() {

cout << "Broj kartice: " << brojKartice << endl;

if (dug == 0)

cout << "Clan je platio clanarinu! " << endl;

else

cout << "Clan duguje " << dug << endl;

}

};

class Osoba {

char\* imePrezime;

char JMBG[13];

char Pol[2];

public:

Osoba(char\* ip, char\* jm, char\* p)

{

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

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

strncpy\_s(JMBG, 13, jm, \_TRUNCATE);

strncpy\_s(Pol, 1, p, \_TRUNCATE);

}

~Osoba() {

delete[]imePrezime; imePrezime = nullptr;

}

Osoba(Osoba &orig) {

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

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

strncpy\_s(JMBG, 13, orig.JMBG, \_TRUNCATE);

strncpy\_s(Pol, 1, orig.Pol, \_TRUNCATE);

}

virtual void Info()

{

cout << "Ime i prezime: " << imePrezime << endl;

cout << "JMBG: " << JMBG << endl;

cout << "Pol: " << Pol << endl;

}

};

class Student : public Osoba {

int index;

int GodStudija;

PolozeniPredmet\* polozeni[40];

int BrojPolozenih;

public:

Student(int indeks = 0, int godina = 0, char\* ip = "--", char\* jmbg = "--", char\* pol = "-", char\* brojK = "--") :

Osoba(ip, jmbg, pol)

{

for (int i = 0; i<40; i++)

{

polozeni[i] = nullptr;

}

index = indeks;

GodStudija = godina;

BrojPolozenih = 0;

}

void dodajPolozeni(char\* prof, char\* n, int ocj)

{

if (BrojPolozenih == 40)

throw exception("Nemoguce dodati vise od 40 polozenih predmeta\n");

PolozeniPredmet temp(prof, n, ocj);

for (int i = 0; i < BrojPolozenih; i++) {

if (\*polozeni[i] == temp)

throw exception("Nemoguce dodavati iste predmete\n");

}

polozeni[BrojPolozenih] = new PolozeniPredmet(temp);

BrojPolozenih++;

}

void ukloniPolozeni(int lokacija)

{

if (lokacija < 0 || lokacija >= BrojPolozenih)

throw exception("Lokacija nije validna\n");

for (int i = lokacija; i < BrojPolozenih - 1; i++) {

delete polozeni[i];

polozeni[i] = new PolozeniPredmet(\*polozeni[i + 1]);

}

delete polozeni[BrojPolozenih - 1];

polozeni[BrojPolozenih - 1] = nullptr;

BrojPolozenih--;

}

PolozeniPredmet\* operator[](int index)

{

if (index < 0 || index >= BrojPolozenih)

throw exception("Indeks nije validan!");

return polozeni[index];

}

void Info() {

Osoba::Info();

cout << "Indeks: " << index << endl;

cout << "Godina studija: " << GodStudija << endl;

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

cout<<\*polozeni[i]<<endl;

}

int pretraga(int ocjena)

{

int brojac = 0;

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

{

if (polozeni[i]->getOcjena() == ocjena)

{

brojac++;

cout << \*polozeni[i] << endl;

}

}

return brojac;

}

};

void main()

{

Student\* s = new Student(1489, 2, "Test Test", "1921680244", "M", "12345");

try {

s->dodajPolozeni("Denis Music", "PR3", 10);

s->dodajPolozeni("Jasmin Azemovic", "BPII", 10);

s->dodajPolozeni("Denis Music", "PR3", 10);

}

catch (exception &err) {

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

}

s->pretraga(10);

s->Info();

s->pretraga(8);

PolozeniPredmet\* p = (\*s)[0];

if (p != nullptr)

cout << \*p << endl;

try {

s->ukloniPolozeni(1);

s->ukloniPolozeni(6);

}

catch (exception &err) {

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

}

s->Info();

system("PAUSE");

}