1.

#include<iostream>

using namespace std;

class Osoba {

protected: char \*imePrezime;

protected: char \*JMBG;

public:

//defualtni konstruktor, koji puni samo 'testnim' vrijednostima atribute

//konstruktor koji prima imePrezime, JMBG

//konstruktor kopije

//destruktor

Osoba() {

char temp[] = "---";

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

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

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

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

}

Osoba(const char \*imePrezime, const char \*JMBG) {

this->imePrezime = new char[strlen(imePrezime) + 1];

strcpy\_s(this->imePrezime, strlen(imePrezime) + 1, imePrezime);

this->JMBG = new char[strlen(JMBG) + 1];

strcpy\_s(this->JMBG, strlen(JMBG) + 1, JMBG);

}

Osoba(const Osoba &original) {

this->imePrezime = new char[strlen(original.imePrezime) + 1];

strcpy\_s(this->imePrezime, strlen(original.imePrezime) + 1, original.imePrezime);

this->JMBG = new char[strlen(original.JMBG) + 1];

strcpy\_s(this->JMBG, strlen(original.JMBG) + 1, original.JMBG);

}

char \*GetIme() { return imePrezime; }

char \*GetJMBG() { return JMBG; };

void Info() {

cout << imePrezime << endl;

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

cout << "=========================================\n";

}

~Osoba() {

delete[]imePrezime; imePrezime = nullptr;

delete[]JMBG; JMBG = nullptr;

}

};

class Zaposlenik : public Osoba{//neka Zaposlenik bude izvedena klasa od klase Osoba

protected: int godineRadnogStaza;

public:

//defualtni konstruktor, koji puni samo 'testnim' vrijednostima atribute

//konstruktor koji prima imePrezime, JMBG

//konstruktor kopije

Zaposlenik():Osoba(){

godineRadnogStaza = 0;

}

Zaposlenik(const char \*imePrezime, const char \*JMBG, int godineRadnogStaza):Osoba(imePrezime,JMBG) {

godineRadnogStaza = godineRadnogStaza;

}

Zaposlenik(const Zaposlenik &original):Osoba(original) {

godineRadnogStaza = original.godineRadnogStaza;

}

void Info(){

cout << imePrezime << endl;

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

cout << "Radni staz: " << godineRadnogStaza << endl;

cout << "=========================================\n";

}

};

void main() {

Osoba o;

o.Info();

Zaposlenik z("Niko Nikić","123456",4);

z.Info();

Osoba \*pok = new Osoba(o);

pok->Info();

pok = &z;

pok->Info();

system("pause>0");

}

2.

#include<iostream>

using namespace std;

class Osoba {

protected:

char \* \_imePrezime;

char \_JMBG[14];

public:

//Kreirati potrebne konstruktore i destruktor

Osoba(const char name[]="---", const char JMBG[]="---") {

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

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

strncpy\_s(\_JMBG, 14, name,\_TRUNCATE);

}

Osoba(Osoba &original) {

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

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

strncpy\_s(\_JMBG, 14, original.\_JMBG, \_TRUNCATE);

}

~Osoba() {

delete[]\_imePrezime;

\_imePrezime = NULL;

}

};

class Predmet

{

char \* \_naziv;

int \_ocjena;

public:

//Kreirati:

//1. Potrebne konstruktore i destruktor

//2. Funckiju Kopiraj koja prima paramrtar tipa Predmet, te

//kopira sve podatke iz primljenog objekta u this objekat

//3. Funkciju AreEqual koji poredi predmete po nazivu i vraća true ili false u odnosu na rezultat poređenja

//4. Funkciju Info za ispis podataka o predmetu

Predmet(const char naziv[]="---", int ocjena=0) {

\_naziv = new char[strlen(naziv) + 1];

strcpy\_s(\_naziv, strlen(naziv) + 1, naziv);

\_ocjena = ocjena;

}

Predmet(Predmet &original) {

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

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

\_ocjena = original.\_ocjena;

}

~Predmet() {

delete[]\_naziv; \_naziv = nullptr;

}

void Kopiraj(Predmet p) {

\_naziv = new char[strlen(p.\_naziv) + 1];

strcpy\_s(\_naziv, strlen(p.\_naziv) + 1, p.\_naziv);

\_ocjena = p.\_ocjena;

}

bool AreEqual(Predmet drugi) {

return strcmp(\_naziv, drugi.\_naziv) == 0;

}

void Info(){

cout << "Predmet: " << \_naziv << endl;

cout << "Ocjena: " << \_ocjena << endl;

}

};

class Student :public Osoba {

protected:

int \_brojIndeksa;

Predmet \* \_predmeti;//\_predmeti predstavljaju pokazivač na niz od 40 predmeta

int \_trenutno;

public:

//Kreirati:

//1. Potrebne konstruktore i destruktor

//2. Funkciju AddPredmet koja dodaje predmete, onemogućiti dodavanje istih predmeta

//3. Funkciju za ispis

Student(const char name[] = "---", const char JMBG[] = "---",int brojIndeksa=0):

Osoba(name,JMBG)

{

\_brojIndeksa = brojIndeksa;

\_trenutno = 0;

\_predmeti = new Predmet[40];

}

Student(Student &original): Osoba(original) {

\_brojIndeksa = original.\_brojIndeksa;

\_predmeti = new Predmet[40];

\_trenutno = original.\_trenutno;

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

{

\_predmeti[i].Kopiraj(original.\_predmeti[i]);

}

}

bool AddPredmet(Predmet &novi) {

if (\_trenutno == 40) {

cout << "Nemoguće dodati više od 40 predmeta.\n";

return false;

}

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

{

if (\_predmeti[i].AreEqual(novi))

{

cout << "Nemoguće dodavati iste predmete.\n";

return false;

}

}

\_predmeti[\_trenutno] = novi;

\_trenutno++;

}

void Info() {

cout << "Student: " << \_imePrezime << " (" << \_brojIndeksa << ")\n";

cout << "JMBG: " << \_JMBG << endl;

cout << "::::::::::::: POLOZENI PREDMETI :::::::::::::\n";

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

{

\_predmeti[i].Info();

}

}

};

class DLStudent :public Student {

private:

char \* \_lokacijaPolaganja;

public:

//Kreirati:

//1. Potrebne konstruktore i destruktor

//2. Funkciju izmijeniLokaciju koja na osnovu primljenog parametra mijenja lokaciju polaganja

//3. Funkciju za ispis

DLStudent(const char name[] = "---", const char JMBG[] = "---", int brojIndeksa = 0,const char lokacija[]="---"):

Student(name,JMBG,brojIndeksa)

{

\_lokacijaPolaganja = new char[strlen(lokacija) + 1];

strcpy\_s(\_lokacijaPolaganja, strlen(lokacija) + 1, lokacija);

}

DLStudent(DLStudent &original) :Student(original){

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

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

}

~DLStudent() {

delete[]\_lokacijaPolaganja; \_lokacijaPolaganja = nullptr;

}

void IzmjeniLokaciju(char novaLokacija[]) {

delete[]\_lokacijaPolaganja;

\_lokacijaPolaganja = new char[strlen(novaLokacija) + 1];

strcpy\_s(\_lokacijaPolaganja, strlen(novaLokacija) + 1, novaLokacija);

}

void Info() {

Student::Info();

cout << "Lokacija polaganja ispita: " << \_lokacijaPolaganja << endl;

}

};

void main() {

Student s("Aida", "123", 100);

Predmet p1("PR1",10);

Predmet p2("PR2",9);

Predmet p3("PR3",8);

s.AddPredmet(p1);

s.AddPredmet(p2);

s.AddPredmet(p1);

s.Info();

DLStudent dl("Test", "456", 400,"Sarajevo");

dl.AddPredmet(p1);

dl.AddPredmet(p2);

dl.AddPredmet(p3);

dl.Info();

system("pause>0");

};