#include <vector>

#include <iostream>

#include <cassert>

#include <functional>

using namespace std;

int MIN\_GODINA = 1, MAX\_GODINA = 10000;

enum Mjesec { Januar = 1, Februar, Mart, April, Maj, Juni, Juli, August, Septembar, Oktobar, Novembar, Decembar };

enum class Spol { Muski = 1, Zenski = 0 };

//Z0.1

char\* Alokator(const char\* tekst);

//Z0.2 :: Funkcija za alokaciju pametnog (shared\_ptr) pokazivaca na niz karaktera

shared\_ptr<char> SharedAlokator(const char\* tekst);

//Z0.3 :: Funkcija za alokaciju pametnog (unique\_ptr) pokazivaca na niz karaktera

unique\_ptr<char> UniqueAlokator(const char\* tekst);

//Z0.4 :: Funkcija vraca max. od dva elementa

template<typename T>

T Max(T el1, T el2);

//Z0.5 :: Funkcija vraca min. od dva elementa

template<typename T>

T Min(T el1, T el2);

class Vrijeme {

protected:

    int\* \_sati;

    int\* \_minute;

    int\* \_sekunde;

public:

    \_\_declspec(property (get = GetSati, put = SetSati)) int Sati;

    \_\_declspec(property (get = GetMinute, put = SetMinute)) int Minute;

    \_\_declspec(property (get = GetSekunde, put = SetSekunde)) int Sekunde;

    //Z1.1

    //Vraca true ako postoji makar jedno obiljezje sa vrijednoscu nullptr

    static bool CheckIfNull(const Vrijeme& obj);

    //Z1.2

    Vrijeme();

    //Z1.3

    Vrijeme(int sati, int minute, int sekunde);

    //Z1.4

    Vrijeme(const Vrijeme& obj);

    //Z1.5

    Vrijeme(Vrijeme&& obj);

    //Z1.6

    Vrijeme& operator =(const Vrijeme& obj);

    //Z1.7

    void SetSati(int sati);

    void SetMinute(int minute);

    void SetSekunde(int sekunde);

    //Z1.8

    int GetSati() const;

    int GetMinute() const;

    int GetSekunde() const;

    //Z1.9 :: koristenjem rand() funkcije generisati slucajne vrijednosti za novi objekat(Vrijeme)

    static Vrijeme GetRandomVrijeme();

    //Z1.10

    bool CheckIfValidnoVrijeme();

    //Z1.11

    ~Vrijeme();

};

//Z1.12

ostream& operator << (ostream& COUT, const Vrijeme& obj);

//Z1.13

bool operator == (const Vrijeme& v1, const Vrijeme& v2);

bool operator != (const Vrijeme& v1, const Vrijeme& v2);

bool operator > (const Vrijeme& v1, const Vrijeme& v2);

bool operator >= (const Vrijeme& v1, const Vrijeme& v2);

bool operator < (const Vrijeme& v1, const Vrijeme& v2);

bool operator <= (const Vrijeme& v1, const Vrijeme& v2);

class DatumVrijeme : public Vrijeme {

private:

    shared\_ptr<int> \_dan;

    shared\_ptr<int> \_mjesec;

    shared\_ptr<int> \_godina;

public:

    \_\_declspec(property (get = GetDan, put = SetDan)) int Dan;

    \_\_declspec(property (get = GetMjesec, put = SetMjesec)) int Mjesec;

    \_\_declspec(property (get = GetGodina, put = SetGodina)) int Godina;

    //Z2.1

    static bool CheckIfNull(const DatumVrijeme& obj);

    //Z2.2

    DatumVrijeme();

    //Z2.3

    DatumVrijeme(int dan, int mjesec, int godina, int sati, int minute, int sekunde);

    //Z2.4

    DatumVrijeme(const DatumVrijeme& obj);

    //Z2.5

    DatumVrijeme(DatumVrijeme&& obj);

    //Z2.6

    DatumVrijeme& operator =(const DatumVrijeme& obj);

    //Z2.7

    void SetDan(int dan);

    void SetMjesec(int mjesec);

    void SetGodina(int godina);

    //Z2.8

    int GetDan() const;

    int GetMjesec() const;

    int GetGodina() const;

    //Z2.9

    bool CheckIfValidnoDatumVrijeme();

    //Z2.10 Obratiti paznju na validnost novokreiranog objekta (DatumVrijeme) sa slucajnim vrijednostima

    static DatumVrijeme GetRandomDatumVrijeme();

    //Z2.11

    ~DatumVrijeme();

};

//Z2.12

ostream& operator << (ostream& COUT, const DatumVrijeme& obj);

//Z2.13

bool operator == (const DatumVrijeme& d1, const DatumVrijeme& d2);

bool operator !=(const DatumVrijeme& d1, const DatumVrijeme& d2);

//Z2.14

bool operator >(const DatumVrijeme& d1, const DatumVrijeme& d2);

bool operator >= (const DatumVrijeme& d1, const DatumVrijeme& d2);

bool operator < (const DatumVrijeme& d1, const DatumVrijeme& d2);

bool operator <= (const DatumVrijeme& d1, const DatumVrijeme& d2);

template<class T, int max> //max oznacava velicinu niza

class Array {

private:

    int \_trenutno; //trenutno elemenata

    T\* \_elementi[max] = { nullptr }; //staticki niz pokazivaca

    bool \_omoguciDupliranje;

public:

    \_\_declspec(property (get = GetTrenutno)) int Length;

    //Z3.1

    Array(bool omoguciDupliranje = true);

    //Z3.2

    Array(const Array<T, max>& obj);

    //Z3.3

    Array(Array<T, max>&& obj);

    //Z3.4

    Array<T, max>& operator=(const Array<T, max>& obj);

    //Z3.5 :: Osigurati da je index unutar opsega [0,\_trenutno-1]. Koristiti genericke funkcije 'Min' i 'Max'

    T& operator [] (int index) const;

    //Z3.6 :: Dodati (alocirati u niz) novi element

    bool operator +=(T noviElement);

    //Z3.7 :: Izbrisati (dealocirati) zadnji element i umanjiti brojac

    bool UkloniZadnjiElement();

    //Z3.8

    bool CheckIfSadrzi(T element);

    //Z3.9 :: Sortiranje elemenata pomocu algoritma za sortiranje "Bubble Sort"

    void Sortiraj(bool reverse = false);

    //Z3.10

    int GetTrenutno() const;

    int GetMaxElemenata() const;

    bool GetOmoguciDupliranje() const;

    //Z3.11

    ~Array();

};

//Z3.12

template<class T, int max>

ostream& operator << <>(ostream& COUT, const Array<T, max>& obj);

//Z3.13

template<class T, int max>

void Ispis(const Array<T, max>& obj, const char\* delimiter = " ");

//Z3.14

template<class T, int max>

bool operator == (const Array<T, max>& arr1, const Array<T, max>& arr2);

template<class T, int max>

bool operator != (const Array<T, max>& arr1, const Array<T, max>& arr2);

class Osoba {

protected:

    shared\_ptr<char> \_ime;

    shared\_ptr<char> \_prezime;

    unique\_ptr<DatumVrijeme> \_datumRodjenja;

    Spol \_spol;

public:

    \_\_declspec(property (get = GetIme, put = SetIme)) const char\* Ime;

    \_\_declspec(property (get = GetPrezime, put = SetPrezime)) const char\* Prezime;

    \_\_declspec(property (get = GetDatumRodjenja, put = SetDatumRodjenja)) DatumVrijeme DatumRodjenja;

    \_\_declspec(property (get = GetSpol, put = SetSpol)) Spol Spol\_;

    //Z4.1

    Osoba(Spol spol = Spol::Zenski);

    //Z4.2

    Osoba(const char\* ime, const char\* prezime, DatumVrijeme datumRodjenja, Spol spol);

    //Z4.3

    Osoba(const Osoba& obj);

    //Z4.4

    Osoba(Osoba&& obj);

    //Z4.5

    Osoba& operator = (const Osoba& obj);

    //Z4.6

    const char\* GetIme() const;

    const char\* GetPrezime() const;

    DatumVrijeme GetDatumRodjenja() const;

    Spol GetSpol() const;

    //Z4.7

    void SetIme(const char\* ime);

    void SetPrezime(const char\* prezime);

    void SetDatumRodjenja(DatumVrijeme datumRodjenja);

    void SetSpol(Spol spol);

    //4.8

    ~Osoba();

};

//Z4.9

ostream& operator << (ostream& COUT, const Osoba& osoba);

//Z4.10

bool operator == (const Osoba& o1, const Osoba& o2);

bool operator != (const Osoba& o1, const Osoba& o2);

class Pjevac : public Osoba {

    const char\* \_zanr;

    vector<const char\*> \_instrumenti;

    unique\_ptr<char> \_aktivan; // Npr. 1978 - present

public:

    \_\_declspec(property (get = GetZanr, put = SetZanr)) const char\* Zanr;

    \_\_declspec(property (get = GetInstrumenti)) vector<const char\*> Instrumenti;

    \_\_declspec(property (get = GetAktivan, put = SetAktivan)) const char\* Aktivan;

    //Z5.1

    Pjevac();

    //Z5.2

    Pjevac(const char\* ime, const char\* prezime, DatumVrijeme dv, Spol sp, const char\* zanr, const char\* aktivan);

    //Z5.3

    Pjevac(const Pjevac& obj);

    //Z5.4

    Pjevac(Pjevac&& obj);

    //Z5.5

    Pjevac& operator = (const Pjevac& obj);

    //Z5.6

    const char\* GetZanr()const;

    const char\* GetAktivan()const;

    vector<const char\*> GetInstrumenti() const;

    //Z5.7

    void SetZanr(const char\* zanr);

    void SetAktivan(const char\* aktivan);

    //Z5.8

    void DodajInstrument(const char\* instrument);

    //Z5.9

    ~Pjevac();

};

//Z5.10

ostream& operator << (ostream& COUT, const Pjevac& pjevac);

//Z5.11

bool operator == (const Pjevac& p1, const Pjevac& p2);

bool operator != (const Pjevac& p1, const Pjevac& p2);

class Pjesma {

    const char\* \_naziv;

    char\* \_tekst;

    pair<int, int> \_trajanje; // <minute, sekunde> [Po defaultu postaviti na <0,0>]

    shared\_ptr<char> \_tekstopisac;

    int \_trenutnoZanrova;

    char\* \_zanrovi[5] = { nullptr };

public:

    \_\_declspec(property (get = GetNaziv, put = SetNaziv)) const char\* Naziv;

    \_\_declspec(property (get = GetTekst, put = SetTekst)) const char\* Tekst;

    \_\_declspec(property (get = GetTrajanje, put = SetTrajanje)) pair<int, int> Trajanje;

    \_\_declspec(property (get = GetTekstopisac, put = SetTekstopisac)) const char\* Tekstopisac;

    //Z6.1

    Pjesma();

    //Z6.2

    Pjesma(const char\* naziv, const char\* tekst, pair<int, int> trajanje, const char\* tekstopisac);

    //Z6.3

    Pjesma(const Pjesma& obj);

    //Z6.4

    Pjesma(Pjesma&& obj);

    //Z6.5

    Pjesma& operator = (const Pjesma& obj);

    //Z6.6

    const char\* GetNaziv() const;

    const char\* GetTekst() const;

    pair<int, int> GetTrajanje()const;

    const char\* GetTekstopisac()const;

    int GetTrenutno()const;

    const char\* GetZanr(int index) const;

    //Z6.7

    void SetNaziv(const char\* naziv);

    void SetTekst(const char\* tekst);

    void SetTrajanje(pair<int, int> trajanje);

    void SetTekstopisac(const char\* tekstopisac);

    //Z6.8

    bool DodajZanr(const char\* zanr);

    //Z6.9

    ~Pjesma();

};

//Z6.10

ostream& operator << (ostream& COUT, const Pjesma& obj);

//Z6.11

bool operator == (const Pjesma& p1, const Pjesma& p2);

bool operator != (const Pjesma& p1, const Pjesma& p2);

class Album {

    char\* \_naziv;

    Pjevac\* \_pjevac;

    Array<Pjesma, 100> \_pjesme;

    vector<const char\*> \_izdavackeKuce;

public:

    \_\_declspec(property (get = GetNaziv, put = SetNaziv)) const char\* Naziv;

    \_\_declspec(property (get = GetPjevac, put = SetPjevac)) Pjevac Pjevac\_;

    \_\_declspec(property (get = GetPjesme)) Array<Pjesma, 100> Pjesme;

    \_\_declspec(property (get = GetIzdavackeKuce)) vector<const char\*> IzdavackeKuce;

    //Z7.1

    Album();

    //Z7.2

    Album(const char\* naziv, Pjevac pjevac);

    //Z7.3

    Album(const Album& obj);

    //Z7.4

    Album(Album&& obj);

    //Z7.5

    Album& operator =(const Album& obj);

    //Z7.6

    const char\* GetNaziv() const;

    Pjevac GetPjevac()const;

    Array<Pjesma, 100> GetPjesme() const;

    vector<const char\*> GetIzdavackeKuce() const;

    //Z7.7

    void SetNaziv(const char\* naziv);

    void SetPjevac(Pjevac pjevac);

    //Z7.8

    bool operator+= (Pjesma novaPjesma);

    //Z7.9

    void operator+= (const char\* izdavackaKuca);

    //Z7.10

    ~Album();

};

//Z7.11

ostream& operator <<(ostream& COUT, const Album& obj);

//Z7.12

bool operator == (const Album& a1, const Album& a2);

bool operator !=(const Album& a1, const Album& a2);

auto IspisiPoruku = [](const char\* tipFunkcije, bool success) {

    cout << tipFunkcije << " se izvrsio " << ((success) ? "uspjesno!" : "bezuspjesno!") << endl;

};

void Zadatak1() {

    cout << "Generisanje slucajnog vremena: " << endl;

    Vrijeme vrijeme = Vrijeme::GetRandomVrijeme();

    cout << vrijeme << endl;

    Vrijeme vrijeme2(vrijeme.Sati, vrijeme.Minute, vrijeme.Sekunde);

    IspisiPoruku("User-defined ctor", vrijeme2 == vrijeme);

    Vrijeme vrijeme3(vrijeme);

    IspisiPoruku("Copy ctor", vrijeme3 == vrijeme);

    Vrijeme vrijeme4(move(vrijeme2));

    IspisiPoruku("Move ctor", vrijeme4 == vrijeme);

    Vrijeme vrijeme5;

    vrijeme5 = vrijeme;

    IspisiPoruku("Copy assignment operator", vrijeme5 == vrijeme);

    //Za [27.4]

    cout << endl;

    Vrijeme izlazakSunca(5, 48, 0);

    cout << izlazakSunca << " => " << (izlazakSunca.CheckIfValidnoVrijeme() ? "validno vrijeme" : "nevalidno vrijeme") << endl;

    Vrijeme podne(12, 47, 0);

    cout << podne << " => " << (podne.CheckIfValidnoVrijeme() ? "validno vrijeme" : "nevalidno vrijeme") << endl;

    Vrijeme zalazakSunca(19, 50, 0);

    cout << zalazakSunca << " => " << (zalazakSunca.CheckIfValidnoVrijeme() ? "validno vrijeme" : "nevalidno vrijeme") << endl;

    cout << "\nDealokacija" << endl;

}

void Zadatak2() {

    cout << "Generisanje slucajnog datuma i vremena: " << endl;

    DatumVrijeme dv = DatumVrijeme::GetRandomDatumVrijeme();

    cout << dv << endl;

    DatumVrijeme dv2(dv.Dan, dv.Mjesec, dv.Godina, dv.Sati, dv.Minute, dv.Sekunde);

    IspisiPoruku("User-defined ctor", dv2 == dv);

    DatumVrijeme dv3(dv);

    IspisiPoruku("Copy ctor", dv3 == dv);

    DatumVrijeme dv4(move(dv2));

    IspisiPoruku("Move ctor", dv4 == dv);

    DatumVrijeme dv5;

    dv5 = dv;

    IspisiPoruku("Copy assignment operator", dv5 == dv);

    cout << "\nDealokacija" << endl;

}

void Zadatak3() {

    const int maxElemenata = 100;

    Array<int, maxElemenata> randomNumbers;

    cout << "Dodavanje " << maxElemenata << " elemenata!" << endl;

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

        randomNumbers += rand() % 100;

    int toRemove = 75;

    cout << "Uklanjanje " << toRemove << " elemenata!" << endl;

    for (int i = 0; i < 75; i++)

        randomNumbers.UkloniZadnjiElement();

    cout << "Ostalo nam je " << maxElemenata - toRemove << " elemenata!" << endl;

    Ispis(randomNumbers, ", ");

    //Sortiranje Array<int,max>

    cout << "Sortiranje [ASCENDING]: " << endl;

    randomNumbers.Sortiraj(false);

    Ispis(randomNumbers, ", ");

    cout << "Sortiranje [DESCENDING]: " << endl;

    randomNumbers.Sortiraj(true);

    Ispis(randomNumbers, ", ");

    Array<int, maxElemenata> arr2(randomNumbers);

    IspisiPoruku("Copy ctor", arr2 == randomNumbers);

    Array<int, maxElemenata> arr3(move(arr2));

    IspisiPoruku("Move ctor", arr3 == randomNumbers);

    Array<int, maxElemenata> arr4;

    arr4 = randomNumbers;

    IspisiPoruku("Copy assignment operator", arr4 == randomNumbers);

    cout << "\nTestiranje Array<DatumVrijeme,int> \n";

    Array<DatumVrijeme, 10> datumi;

    for (int i = 0; i < 10; i++)

        datumi += DatumVrijeme::GetRandomDatumVrijeme();

    Ispis(datumi, "\n");

    //Sortiranje Array<Datum,max>

    cout << "Sortiranje [ASCENDING]: " << endl;

    datumi.Sortiraj(false);

    Ispis(datumi, "\n");

    cout << "Sortiranje [DESCENDING]: " << endl;

    datumi.Sortiraj(true);

    Ispis(datumi, "\n");

    cout << endl;

    cout << "\nDealokacija" << endl;

}

void Zadatak4() {

    Osoba realDonaldTrump;

    realDonaldTrump.Ime = "Donald J.";

    realDonaldTrump.Prezime = "Trump";

    realDonaldTrump.DatumRodjenja = DatumVrijeme(13, 5, 1945, 17, 30, 0);

    realDonaldTrump.Spol\_ = Spol::Muski;

    cout << realDonaldTrump << endl;

    Osoba jobsPresident(realDonaldTrump.Ime, realDonaldTrump.Prezime, realDonaldTrump.DatumRodjenja, realDonaldTrump.Spol\_);

    IspisiPoruku("User-defined ctor", jobsPresident == realDonaldTrump);

    Osoba borisJohnson(realDonaldTrump);

    IspisiPoruku("Copy ctor", borisJohnson == realDonaldTrump);

    Osoba magaPresident(move(jobsPresident));

    IspisiPoruku("Move ctor", magaPresident == realDonaldTrump);

    Osoba o4;

    o4 = realDonaldTrump;

    IspisiPoruku("Copy assignment", o4 == realDonaldTrump);

    cout << endl;

    cout << "\nDealokacija" << endl;

}

void Zadatak5() {

    Pjevac theRocketMan;

    theRocketMan.Ime = "Kim";

    theRocketMan.Prezime = "Jong-un";

    theRocketMan.DatumRodjenja = DatumVrijeme(32, 12, 1940, 01, 00, 00);

    theRocketMan.Spol\_ = Spol::Muski;

    theRocketMan.Zanr = "K-pop";

    theRocketMan.Aktivan = "2011-present";

    theRocketMan.DodajInstrument("bubnjevi");

    theRocketMan.DodajInstrument("bongo");

    theRocketMan.DodajInstrument("klavir");

    cout << theRocketMan << endl;

    Pjevac supremeFrontman(theRocketMan.Ime, theRocketMan.Prezime,

        theRocketMan.DatumRodjenja, theRocketMan.Spol\_, theRocketMan.Zanr, theRocketMan.Aktivan);

    supremeFrontman.DodajInstrument("bubnjevi");

    supremeFrontman.DodajInstrument("bongo");

    supremeFrontman.DodajInstrument("klavir");

    IspisiPoruku("User-defined ctor", supremeFrontman == theRocketMan);

    Pjevac kPopPrince(theRocketMan);

    IspisiPoruku("Copy ctor", kPopPrince == theRocketMan);

    Pjevac sendMeNukes(move(supremeFrontman));

    IspisiPoruku("Move ctor", sendMeNukes == theRocketMan);

    Pjevac p4;

    p4 = theRocketMan;

    IspisiPoruku("Copy assignment", p4 == theRocketMan);

    cout << endl;

    cout << "Dealokacija" << endl;

}

void Zadatak6() {

    Pjesma horseWithNoName;

    horseWithNoName.Naziv = "America - A Horse With No Name";

    horseWithNoName.Tekst = "On the first part of the journey I was looking at all the life";

    horseWithNoName.Trajanje = make\_pair<int, int>(4, 7);

    horseWithNoName.Tekstopisac = "Dewey Bunnell";

    horseWithNoName.DodajZanr("folk rock");

    horseWithNoName.DodajZanr("soft rock");

    horseWithNoName.DodajZanr("country rock");

    cout << horseWithNoName;

    Pjesma gtaSaSong(horseWithNoName.Naziv, horseWithNoName.Tekst, horseWithNoName.Trajanje, horseWithNoName.Tekstopisac);

    gtaSaSong.DodajZanr("folk rock");

    gtaSaSong.DodajZanr("soft rock");

    gtaSaSong.DodajZanr("country rock");

    IspisiPoruku("User-defined ctor", gtaSaSong == horseWithNoName);

    Pjesma kdistFTW(horseWithNoName);

    IspisiPoruku("Copy ctor", kdistFTW == horseWithNoName);

    Pjesma breakingBadSong(move(gtaSaSong));

    IspisiPoruku("Move ctor", breakingBadSong == horseWithNoName);

    Pjesma joeysFailedAudition;

    joeysFailedAudition = horseWithNoName;

    IspisiPoruku("Copy assignment", joeysFailedAudition == horseWithNoName);

    cout << endl;

    cout << "Dealokacija" << endl;

}

void Zadatak7() {

    Album finalHour;

    finalHour.Naziv = "Face to face";

    finalHour.Pjevac\_ = Pjevac("Phil", "Collins", DatumVrijeme(13, 5, 1951, 14, 30, 00), Spol::Muski, "Pop, soft rock", "1963-2011");

    Pjesma inTheAirTonight("In the air tonight", "I can feel it coming in the air tonight...", make\_pair(5, 34), "Phil Collins");

    inTheAirTonight.DodajZanr("Experimental pop");

    inTheAirTonight.DodajZanr("Soft rock");

    Pjesma anotherDay("Another day in paradise", "She calls out to the man on the street...", make\_pair(4, 40), "Phil Collins");

    anotherDay.DodajZanr("Soft rock");

    Pjesma rainItDown("I Wish It Would Rain Down", "You know I never meant to see you again...", make\_pair(4, 11), "Phil Collins");

    rainItDown.DodajZanr("Pop");

    finalHour += inTheAirTonight;

    finalHour += anotherDay;

    finalHour += rainItDown;

    finalHour += "Atlas Music Publishing";

    finalHour += "Concord Music Publising";

    finalHour += "Big Deal Music";

    finalHour += "Bluewater Music";

    cout << finalHour << endl;

    Album ritual(finalHour.Naziv, finalHour.Pjevac\_);

    ritual += inTheAirTonight;

    ritual += anotherDay;

    ritual += rainItDown;

    ritual += "Atlas Music Publishing";

    ritual += "Concord Music Publising";

    ritual += "Big Deal Music";

    ritual += "Bluewater Music";

    IspisiPoruku("User-defined ctor", ritual == finalHour);

    Album copycat(finalHour);

    IspisiPoruku("Copy ctor", copycat == finalHour);

    Album stealCredit(move(ritual));

    IspisiPoruku("Move ctor", stealCredit == finalHour);

    Album imaginationLevelZero;

    imaginationLevelZero = finalHour;

    IspisiPoruku("Copy assignment", imaginationLevelZero == finalHour);

    cout << endl;

    cout << "Dealokacija" << endl;

}

void Menu() {

    int nastaviDalje = 1;

    while (nastaviDalje == 1) {

        int izbor = 0;

        do {

            system("cls");

            cout << "::Zadaci::" << endl;

            cout << "(1) Zadatak 1" << endl;

            cout << "(2) Zadatak 2" << endl;

            cout << "(3) Zadatak 3" << endl;

            cout << "(4) Zadatak 4" << endl;

            cout << "(5) Zadatak 5" << endl;

            cout << "(6) Zadatak 6" << endl;

            cout << "(7) Zadatak 7" << endl;

            cout << "Unesite odgovarajuci broj zadatka za testiranje: -->: ";

            cin >> izbor;

            cout << endl;

        } while (izbor < 1 || izbor > 7);

        switch (izbor) {

        case 1: Zadatak1(); cout << "Zadatak 1. Done." << endl; break;

        case 2: Zadatak2(); cout << "Zadatak 2. Done." << endl; break;

        case 3: Zadatak3(); cout << "Zadatak 3. Done." << endl; break;

        case 4: Zadatak4(); cout << "Zadatak 4. Done." << endl; break;

        case 5: Zadatak5(); cout << "Zadatak 5. Done." << endl; break;

        case 6: Zadatak6(); cout << "Zadatak 6. Done." << endl; break;

        case 7: Zadatak7(); cout << "Zadatak 7. Done." << endl; break;

        default:break;

        }

        do {

            cout << "DA LI ZELITE NASTAVITI DALJE? (1/0): ";

            cin >> nastaviDalje;

        } while (nastaviDalje != 0 && nastaviDalje != 1);

    }

}

int main() {

    Menu();

    return 0;

}