#include <vector>

#include <iostream>

#include <cassert>

using namespace std;

int Min(int a, int b) { return (a <= b) ? a : b; }

int Max(int a, int b) { return (a >= b) ? a : b; }

//:::::::::::ZADACI::::::::::::

//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:: Provjeriti da li su dva niza karaktera ista (po vrijednosti)

bool CheckIfIsti(const char\* tekst1, const char\* tekst2);

//Z0.5 Provjeriti da li su dva vektora jednaka (da li imaju iste elemente)

template<class T>

bool operator == (const vector<T>& v1, const vector<T>& v2);

template<class T>

bool operator != (const vector<T>& v1, const vector<T>& v2);

bool operator == (const vector<const char\*>& v1, const vector<const char\*>& v2);

bool operator != (const vector<const char\*>& v1, const vector<const char\*>& v2);

//Simulacija ratne stateske igre

enum class Moral { Nizak = 1, Srednji, Visok };

class Vojnik {

protected:

    unique\_ptr<int> \_zdravlje; //[0-100]

    unique\_ptr<int> \_izdrzljivost; //[0-100]

    Moral\* \_moral;

    vector<const char\*> \_oruzja;

    bool \_statusZiv;

public:

    \_\_declspec(property (get = GetZdravlje, put = SetZdravlje)) int Zdravlje;

    \_\_declspec(property (get = GetIzdrzljivost, put = SetIzdrzljivost)) int Izdrzljivost;

    \_\_declspec(property (get = GetMoral, put = SetMoral)) Moral Moral\_;

    \_\_declspec(property (get = GetOruzja, put = SetOruzja)) vector<const char\*> Oruzja;

    \_\_declspec(property (get = GetStatusZiv, put = SetStatusZiv)) bool StatusZiv;

    //Z1.1

    Vojnik(int zdravlje = 100, int izdrzljivost = 100, Moral moral = Moral::Visok, bool statusZiv = true);

    //Z1.2

    Vojnik(const Vojnik& obj);

    //Z1.3

    Vojnik(Vojnik&& obj);

    //Z1.4 :: Getteri

    int GetZdravlje() const;

    int GetIzdrzljivost() const;

    Moral GetMoral() const;

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

    bool GetStatusZiv() const;

    //Z1.5 :: Setteri

    void SetZdravlje(int zdravlje);

    void SetIzdrzljivost(int izdrzljivost);

    void SetMoral(Moral moral);

    void SetOruzja(vector<const char\*> oruzja);

    void SetStatusZiv(bool statusZiv);

    //Z1.6

    Vojnik& operator = (const Vojnik& obj);

    //Z1.7

    virtual void Ispis();

    //Z1.8

    virtual ~Vojnik();

};

//Z1.9

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

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

//interfejs

class IPonasanje {

public:

    virtual void OglasiSe() = 0;

    virtual void Napadaj(Vojnik\* meta) = 0;

    virtual void OporavljajSe() = 0;

};

enum class Kategorija { Laka = 1, Srednja, Teska };

class Pjesak : public Vojnik, public IPonasanje {

protected:

    Kategorija \_kategorija;

    unique\_ptr<char> \_oklop;

public:

    \_\_declspec(property (get = GetKategorija, put = SetKategorija)) Kategorija Kategorija\_;

    \_\_declspec(property (get = GetOklop, put = SetOklop)) const char\* Oklop;

    //Z2.1

    Pjesak(Kategorija kategorija = Kategorija::Srednja, const char\* oklop = "metalni");

    //Z2.2

    Pjesak(const Pjesak& obj);

    //Z2.3

    Pjesak(Pjesak&& obj);

    //Z2.4 :: Getteri

    Kategorija GetKategorija() const;

    const char\* GetOklop() const;

    //Z2.5 :: Setteri

    void SetKategorija(Kategorija kategorija);

    void SetOklop(const char\* oklop);

    //Z2.6

    Pjesak& operator = (const Pjesak& obj);

    //Z2.7

    void Ispis();

    //Z2.8 :: Ispisati poruku "Ja sam tipa <Tip Podatka>" ...

    //Za utvrdjivanje tipa podatka, koristiti funkciju 'typeid'

    void OglasiSe();

    //Z2.9 :: Ovu funkciju implementirati izvan scope-a

    void Napadaj(Vojnik\* meta);

    //Z2.10 :: Povecati vrijednost atributa Zdravlje za slucajnu vrijednost izmedju [0-25]

    //Pri cemu isti atribut mora ostati u opsegu vrijednosti od 0-100

    void OporavljajSe();

    //Z2.11

    ~Pjesak();

};

//Z2.12

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

//Z2.13

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

class Strijelac : public Vojnik, public IPonasanje {

protected:

    int\* \_municija;

public:

    \_\_declspec(property (get = GetMunicija, put = SetMunicija)) int Municija;

    //Z3.1

    Strijelac(int municija = 50);

    //Z3.2

    Strijelac(const Strijelac& obj);

    //Z3.3

    Strijelac(Strijelac&& obj);

    //Z3.4 :: Getteri

    int GetMunicija() const;

    //Z3.5 :: Setteri

    void SetMunicija(int municija);

    //Z3.6

    Strijelac& operator =(const Strijelac& obj);

    //Z3.7

    void Ispis();

    //Z3.8 :: Ispisati poruku "Ja sam tipa <Tip Podatka>" ...

    //Za utvrdjivanje tipa podatka, koristiti funkciju 'typeid'

    void OglasiSe();

    //Z3.9 :: Ovu funkciju implementirati izvan scope-a

    void Napadaj(Vojnik\* meta);

    //Z3.10  Povecati vrijednost atributa zdravlje za slucajnu vrijednost izmedju [0-35]

    //Pri cemu isti atribut mora ostati u opsegu vrijednosti od 0-100

    void OporavljajSe();

    //Z3.11

    ~Strijelac();

};

//Z3.12

bool operator == (const Strijelac& s1, const Strijelac& s2);

//Z3.13

bool operator != (const Strijelac& s1, const Strijelac& s2);

class Konjanik : public Vojnik, public IPonasanje {

protected:

    unique\_ptr<char> \_oklop;

    const char\* \_vrstaKonja;

public:

    \_\_declspec(property (get = GetOklop, put = SetOklop)) const char\* Oklop;

    \_\_declspec(property (get = GetVrstaKonja, put = SetVrstaKonja)) const char\* VrstaKonja;

    //Z4.1

    Konjanik(const char\* oklop = "metalni", const char\* vrstaKonja = "");

    //Z4.2

    Konjanik(const Konjanik& obj);

    //Z4.3

    Konjanik(Konjanik&& obj);

    //Z4.4 :: Getteri

    const char\* GetOklop() const;

    const char\* GetVrstaKonja() const;

    //Z4.5 :: Setteri

    void SetOklop(const char\* oklop);

    void SetVrstaKonja(const char\* vrstaKonja);

    //Z4.6

    Konjanik& operator = (const Konjanik& obj);

    //Z4.7

    void Ispis();

    //Z4.8 :: Ispisati poruku "Ja sam tipa <Tip Podatka>" ...

    //Za utvrdjivanje tipa podatka, koristiti funkciju 'typeid'

    void OglasiSe();

    //4.9 :: Ovu funkciju implementirati izvan scope-a

    void Napadaj(Vojnik\* meta);

    //Z4.10  Povecati vrijednost atributa zdravlje za slucajnu vrijednost izmedju [0-15]

    //Pri cemu isti atribut mora ostati u opsegu vrijednosti od 0-100

    void OporavljajSe();

    //4.11

    ~Konjanik();

};

//4.12

bool operator == (const Konjanik& k1, const Konjanik& k2);

//4.13

bool operator != (const Konjanik& k1, const Konjanik& k2);

//Z2.9 ::

//Pjesak ima vjerovatnocu od 50% da ce uspjesno sprovesti napad

//\*Utvrditi tip mete koristenjem 'dynamic\_cast'

//U slucaju da je napad uspjesan, steta koja se nanosi je:

// ->30 (ukoliko je meta drugi pjesak)

// ->50 (ukoliko je meta strijelac)

// ->10 (ukoliko je meta konjanik)

//\*Umanjiti zdravlje mete za nacinjenu stetu

//\*Ukoliko je zdravlje mete palo na 0, proglasiti metu nezivom (StatusAlive = false)

void Pjesak::Napadaj(Vojnik\* meta) {

}

//Z3.9 ::

//Strijelac ima vjerovatnocu od 50% da ce uspjesno sprovesti napad

//\*Utvrditi tip mete koristenjem 'dynamic\_cast'

//U slucaju da je napad uspjesan, steta koja se nanosi je:

//->35 (ukoliko je meta pjesak)

//->50 (ukoliko je meta drugi strijelac)

//->65 (ukoliko je meta konjanik)

//\*Umanjiti zdravlje mete za nacinjenu stetu

//\*Ukoliko je zdravlje mete palo na 0, proglasiti metu nezivom (StatusAlive = false)

void Strijelac::Napadaj(Vojnik\* meta) {

}

//Z4.9 ::

//Konjanik ima vjerovatnocu od 50% da ce uspjesno sprovesti napad

//\*Utvrditi tip mete koristenjem 'dynamic\_cast'

//U slucaju da je napad uspjesan, steta koja se nanosi je:

//->50 (ukoliko je meta pjesak)

//->80 (ukoliko je meta strijelac)

//->20 (ukoliko je meta konjanik)

//\*Umanjiti zdravlje mete za nacinjenu stetu

//\*Ukoliko je zdravlje mete palo na 0, proglasiti metu nezivom (StatusAlive = false)

void Konjanik::Napadaj(Vojnik\* meta) {

}

template<class T>

class VojnaJedinica {

protected:

    const char\* \_naziv;

    vector<T> \_pripadnici;

    unique\_ptr<float> \_cijenaObuke;

    unique\_ptr<float> \_cijenaOdrzavanja;

public:

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

    \_\_declspec(property (get = GetPripadnici, put = SetPripadnici)) vector<T> Pripadnici;

    \_\_declspec(property (get = GetCijenaObuke, put = SetCijenaObuke)) float CijenaObuke;

    \_\_declspec(property (get = GetCijenaOdrzavanja, put = SetCijenaOdrzavanja)) float CijenaOdrzavanja;

    //Z5.1

    VojnaJedinica(const char\* naziv = "", float cijenaObuke = 500, float cijenaOdrzavanja = 110);

    //Z5.2

    VojnaJedinica(const VojnaJedinica<T>& obj);

    //Z5.3

    VojnaJedinica(VojnaJedinica<T>&& obj);

    //Z5.4 :: Getteri

    const char\* GetNaziv() const;

    vector<T> GetPripadnici() const;

    float GetCijenaObuke() const;

    float GetCijenaOdrzavanja() const;

    //Z5.5 :: Setteri

    void SetNaziv(const char\* naziv);

    void SetPripadnici(vector<T> pripadnici);

    void SetCijenaObuke(float cijenaObuke);

    void SetCijenaOdrzavanja(float cijenaOdrzavanja);

    //Z5.6

    VojnaJedinica<T>& operator = (const VojnaJedinica<T>& obj);

    //Z5.7

    template<class T>

    void DodajPripadnika(const T pripadnik);

    //Z5.8 :: Ispisati ukupan broj pripadnika umjesto ispisa svakog pojedinacnog pripadnika

    void Ispis();

    //Z5.9

    ~VojnaJedinica();

};

//Z5.11

template<class T>

bool operator == (const VojnaJedinica<T>& obj1, const VojnaJedinica<T>& obj2);

//Z5.12

template<class T>

bool operator != (const VojnaJedinica<T>& obj1, const VojnaJedinica<T>& obj2);

class Armija {

private:

    const char\* \_ime;

    vector<VojnaJedinica<Pjesak>> \_pjesadija;

    vector<VojnaJedinica<Strijelac>> \_strijelci;

    vector<VojnaJedinica<Konjanik>> \_konjica;

public:

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

    \_\_declspec(property (get = GetPjesadija, put = SetPjesadija)) vector<VojnaJedinica<Pjesak>> Pjesadija;

    \_\_declspec(property (get = GetStrijelci, put = SetStrijelci)) vector<VojnaJedinica<Strijelac>> Strijelci;

    \_\_declspec(property (get = GetKonjica, put = SetKonjica)) vector<VojnaJedinica<Konjanik>> Konjica;

    //Z6.1

    Armija(const char\* ime = "");

    //Z6.2

    Armija(const Armija& obj);

    //Z6.3

    Armija(Armija&& obj);

    //Z6.4 :: Getteri

    const char\* GetIme() const;

    vector<VojnaJedinica<Pjesak>> GetPjesadija() const;

    vector<VojnaJedinica<Strijelac>> GetStrijelci() const;

    vector<VojnaJedinica<Konjanik>> GetKonjica() const;

    //Z6.5 :: Setteri

    void SetIme(const char\* ime);

    void SetPjesadija(vector<VojnaJedinica<Pjesak>> pjesadija);

    void SetStrijelci(vector<VojnaJedinica<Strijelac>> strijelci);

    void SetKonjica(vector<VojnaJedinica<Konjanik>> konjica);

    //Z6.6

    Armija& operator =(const Armija& obj);

    //Z6.7

    void DodajJedinicu(VojnaJedinica<Pjesak> jedinica);

    void DodajJedinicu(VojnaJedinica<Strijelac> jedinica);

    void DodajJedinicu(VojnaJedinica<Konjanik> jedinica);

    //Z6.8

    void Ispis();

    //Z6.9

    ~Armija();

};

//Z6.10

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

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

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

};

void Zadatak1() {

    vector<const char\*> oruzja{ "mac", "dugi mac", "ratna sjekira", "buzdovan", "helebarda", "koplje" };

    Vojnik univerzalniVojnik;

    univerzalniVojnik.Zdravlje = 85;

    univerzalniVojnik.Izdrzljivost = 75;

    univerzalniVojnik.Moral\_ = Moral::Srednji;

    univerzalniVojnik.Oruzja = oruzja;

    univerzalniVojnik.StatusZiv = true;

    univerzalniVojnik.Ispis();

    Vojnik u1(univerzalniVojnik);

    IspisiPoruku("Copy ctor", univerzalniVojnik == u1);

    Vojnik u2(move(u1));

    IspisiPoruku("Move ctor", univerzalniVojnik == u2);

    Vojnik u3;

    u3 = univerzalniVojnik;

    IspisiPoruku("Operator '='", univerzalniVojnik == u3);

    cout << endl << "Dealokacija..." << endl;

}

void Zadatak2() {

    vector<const char\*> oruzja{ "velika ratna sjekira", "mac s dvije ruke" };

    Pjesak gimly;

    gimly.Oruzja = oruzja;

    gimly.Oklop = "teski celicni oklop";

    gimly.Kategorija\_ = Kategorija::Teska;

    gimly.Ispis();

    Pjesak anotherDwarf(gimly);

    IspisiPoruku("Copy ctor", anotherDwarf == gimly);

    Pjesak anotherDwarf2(move(anotherDwarf));

    IspisiPoruku("Move ctor", gimly == anotherDwarf2);

    Pjesak p3;

    p3 = gimly;

    IspisiPoruku("Operator '='", gimly == p3);

    cout << endl << "Dealokacija..." << endl;

}

void Zadatak3() {

    vector<const char\*> oruzja{ "velski dugacki luk", "samostrijel", "kratki mac" };

    Strijelac legolas;

    legolas.Oruzja = oruzja;

    legolas.Municija = 150;

    legolas.Ispis();

    Strijelac anotherElf(legolas);

    IspisiPoruku("Copy ctor", legolas == anotherElf);

    Strijelac anotherElf2(move(anotherElf));

    IspisiPoruku("Move ctor", legolas == anotherElf2);

    Strijelac s3;

    s3 = legolas;

    IspisiPoruku("Operator '='", legolas == s3);

    cout << endl << "Dealokacija..." << endl;

}

void Zadatak4() {

    vector<const char\*> oruzja{ "dugacko koplje", "buzdovan" };

    Konjanik eomer;

    eomer.Oruzja = oruzja;

    eomer.Oklop = "celicni oklop";

    eomer.VrstaKonja = "frizijski konj";

    eomer.Ispis();

    Konjanik anotherHorserider(eomer);

    IspisiPoruku("Copy ctor", anotherHorserider == eomer);

    Konjanik anotherHorserider2(move(anotherHorserider));

    IspisiPoruku("Move ctor", eomer == anotherHorserider2);

    Konjanik ghostRider;

    ghostRider = eomer;

    IspisiPoruku("Operator '='", eomer == ghostRider);

    Pjesak p1, p2, p3, p4, p5;

    Strijelac s1, s2, s3, s4, s5;

    Konjanik k1, k2, k3, k4, k5;

    cout << "Bitka pocinje: -->" << endl;

    p1.Napadaj(&s1); //Pjesak napada strijelca (50% vjerovatnoca da uspije)

    p2.Napadaj(&p1); //Pjesak napada pjesaka (50% vjerovatnoca da uspije)

    p1.Napadaj(&k1); //Pjesak napada konjanika (50% vjerovatnoca da uspije)

    //

    s1.Napadaj(&s2); // Strijelac napada strijelca (50% vjerovatnoca da uspije)

    s2.Napadaj(&k5); // Strijelac napada konjanika (50% vjerovatnoca da uspije)

    s4.Napadaj(&p5); // Strijelac napada pjesaka (50% vjerovatnoca da uspije)

    //

    k4.Napadaj(&p3); // Konjanika napada pjesaka (50% vjerovatnoca da uspije)

    k4.Napadaj(&k3); // Konjanik napada konjanika (50% vjerovatnoca da uspije)

    k4.Napadaj(&s4); // Konjanik napada strijelca (50% vjerovatnoca da uspije)

    cout << endl << "Dealokacija..." << endl;

}

void Zadatak5() {

    VojnaJedinica<Pjesak> svicarskaGarda("Svicarska garda", 560.5f, 120.5f);

    svicarskaGarda.Naziv = "Svicarska Garda";

    svicarskaGarda.CijenaObuke = 570.0f;

    svicarskaGarda.CijenaOdrzavanja = 120.0f;

    for (int i = 0; i < 100; i++)

        svicarskaGarda.DodajPripadnika(Pjesak(Kategorija::Srednja, "lagani oklop"));

    svicarskaGarda.Ispis();

    VojnaJedinica<Pjesak> kopija(svicarskaGarda);

    IspisiPoruku("Copy ctor", svicarskaGarda == kopija);

    VojnaJedinica<Pjesak> pjesadija(move(kopija));

    IspisiPoruku("Move ctor", svicarskaGarda == pjesadija);

    VojnaJedinica<Pjesak> p3;

    p3 = svicarskaGarda;

    IspisiPoruku("Operator '='", svicarskaGarda == p3);

    cout << endl << "Dealokacija..." << endl;

}

void Zadatak6() {

    VojnaJedinica<Pjesak> svicarskaGarda("Svicarska garda", 550.0f, 130.0f);

    VojnaJedinica<Pjesak> varangijanskaGarda("Varangijanska garda", 670.0f, 160.0f);

    VojnaJedinica<Pjesak> manAtArms("Man At Arms", 520.0f, 110.0f);

    for (int i = 0; i < 250; i++)

        svicarskaGarda.DodajPripadnika(Pjesak(Kategorija::Srednja, "lagani oklop"));

    for (int i = 0; i < 150; i++)

        varangijanskaGarda.DodajPripadnika(Pjesak(Kategorija::Teska, "celicni oklop"));

    for (int i = 0; i < 100; i++)

        manAtArms.DodajPripadnika(Pjesak(Kategorija::Srednja, "lagani oklop"));

    VojnaJedinica<Strijelac> englishLongbowman("Enlish Longbowman", 350.0f, 110.0f);

    for (int i = 0; i < 250; i++)

        englishLongbowman.DodajPripadnika(Strijelac(100));

    VojnaJedinica<Konjanik> bosanskaElitnaKonjica("Bosanska elitna konjica", 950.0f, 250.0f);

    for (int i = 0; i < 80; i++)

        bosanskaElitnaKonjica.DodajPripadnika(Konjanik("celicni oklop", "bosanski brdski konj"));

    VojnaJedinica<Konjanik> katafrakt("Katafrakt", 850.0f, 230.0f);

    for (int i = 0; i < 100; i++)

        katafrakt.DodajPripadnika(Konjanik("teski celicni oklop", "teski bizantijski konj"));

    Armija sjevernaArmija("Army Of The North");

    sjevernaArmija.DodajJedinicu(svicarskaGarda);

    sjevernaArmija.DodajJedinicu(varangijanskaGarda);

    sjevernaArmija.DodajJedinicu(manAtArms);

    sjevernaArmija.DodajJedinicu(englishLongbowman);

    sjevernaArmija.DodajJedinicu(bosanskaElitnaKonjica);

    sjevernaArmija.DodajJedinicu(katafrakt);

    sjevernaArmija.Ispis();

    Armija juznaArmija(sjevernaArmija);

    IspisiPoruku("Copy ctor", sjevernaArmija == juznaArmija);

    Armija pirane(move(juznaArmija));

    IspisiPoruku("Move ctor", sjevernaArmija == pirane);

    Armija a3;

    a3 = sjevernaArmija;

    IspisiPoruku("Operator '='", sjevernaArmija == a3);

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

            cin >> izbor;

            cout << endl;

        } while (izbor < 1 || izbor > 6);

        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;

        default:break;

        }

        do {

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

            cin >> nastaviDalje;

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

    }

}

int main() {

    Menu();

    return 0;

}