#include <iostream>

#include <string>

#include <sstream>

#include <vector>

#include <list>

#include <exception>

#include <regex>

using namespace std;

//::::::::::::::::::::PRIPREMA ZA ISPIT:::::::::::::::::::::::::::::::::::::::::://

const char\* linija = "\n---------------------------------------------\n";

//Z0.1

char\* Alokator(const char\* str);

//Z0.2

bool CheckIfIsti(const char\* str1, const char\* str2);

//Z0.3

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

//Z0.4

const char\* GetErrorMessage(int linija, const char\* funkcija);

//Z0.5

template<class T>

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

//Z0.6

template<class T>

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

class FizikalnaVelicina abstract {

protected:

    unique\_ptr<float> \_vrijednost;

public:

    \_declspec (property (get = GetVrijednost, put = SetVrijednost)) float Vrijednost;

    //Z0.7

    FizikalnaVelicina(float vrijednost = 0.0);

    //Z0.8

    FizikalnaVelicina(const FizikalnaVelicina& obj);

    //Z0.9

    FizikalnaVelicina(FizikalnaVelicina&& obj);

    //Z0.10

    float GetVrijednost() const;

    //Z0.11

    void SetVrijednost(float vrijednost);

    //Z0.12

    FizikalnaVelicina& operator = (const FizikalnaVelicina& obj);

    //Z0.13

    virtual ~FizikalnaVelicina();

};

enum class JedinicaUdaljenosti { m, km, er, sr, ls, au, ly };

//Z0.14

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

class Udaljenost : public FizikalnaVelicina

{

protected:

    unique\_ptr<JedinicaUdaljenosti> \_jedinica;

public:

    \_declspec (property (get = GetJedinica, put = SetJedinica)) JedinicaUdaljenosti Jedinica;

    //Z1.0

    Udaljenost();

    //Z1.1

    Udaljenost(float vrijednost, JedinicaUdaljenosti jed);

    //Z1.2

    Udaljenost(const Udaljenost& obj);

    //Z1.3

    Udaljenost(Udaljenost&& obj);

    //Z1.4

    JedinicaUdaljenosti GetJedinica() const;

    //Z1.5

    void SetJedinica(JedinicaUdaljenosti jedinica);

    //Z1.6

    Udaljenost& operator = (const Udaljenost& obj);

    //Z1.7

    ~Udaljenost();

};

//Z1.8

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

//Z1.9

bool operator == (const Udaljenost& u1, const Udaljenost& u2);

//Z1.10

bool operator != (const Udaljenost& u1, const Udaljenost& u2);

enum class JedinicaMase { kg, EM, SM }; // EM- Earth Mass, SM - Solar Mass

//Z1.11

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

class Masa : public FizikalnaVelicina

{

protected:

    unique\_ptr<JedinicaMase> \_jedinica;

public:

    \_declspec (property (get = GetJedinica, put = SetJedinica)) JedinicaMase Jedinica;

    //Z1.12

    Masa();

    //Z1.13

    Masa(float vrijednost, JedinicaMase jed);

    //Z1.14

    Masa(const Masa& obj);

    //Z1.15

    Masa(Masa&& obj);

    JedinicaMase GetJedinica() const;

    //Z1.17

    void SetJedinica(JedinicaMase jedinica);

    //Z1.18

    Masa& operator = (const Masa& obj);

    //Z1.19

    ~Masa();

};

//Z1.20

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

//Z1.21

bool operator == (const Masa& m1, const Masa& m2);

//Z1.22

bool operator != (const Masa& m1, const Masa& m2);

//planete, zvijezde, meteori, komete, nebulae, crne rupe, ...

class NebeskoTijelo {

protected:

    string \_naziv;

    unique\_ptr<Udaljenost> \_poluprecnik;

    unique\_ptr<Masa> \_masa;

    string\* \_sastav;

public:

    \_declspec (property (get = GetNaziv, put = SetNaziv)) string Naziv;

    \_declspec (property (get = GetPoluprecnik, put = SetPoluprecnik)) Udaljenost Poluprecnik;

    \_declspec (property (get = GetMasa, put = SetMasa)) Masa Masa\_;

    \_declspec (property (get = GetSastav, put = SetSastav)) string Sastav;

    //Z2.1

    NebeskoTijelo(const char\* naziv = "");

    //Z2.2

    NebeskoTijelo(const char\* naziv, Udaljenost poluprecnik, Masa masa, const char\* sastav);

    //Z2.3

    NebeskoTijelo(const NebeskoTijelo& obj);

    //Z2.4

    NebeskoTijelo(NebeskoTijelo&& obj);

    //Z2.5

    string GetNaziv() const;

    Udaljenost GetPoluprecnik() const;

    Masa GetMasa() const;

    string GetSastav() const;

    //Z2.6

    void SetNaziv(string naziv);

    void SetPoluprecnik(Udaljenost poluprecnik);

    void SetMasa(Masa masa);

    void SetSastav(string sastav);

    //Z2.7

    NebeskoTijelo& operator = (const NebeskoTijelo& obj);

    //Z2.8

    virtual void Ispis();

    //Z2.9

    virtual ~NebeskoTijelo();

};

//Z2.10

bool operator == (const NebeskoTijelo& n1, const NebeskoTijelo& n2);

//Z2.11

bool operator != (const NebeskoTijelo& n1, const NebeskoTijelo& n2);

class Planeta : public NebeskoTijelo {

protected:

    Udaljenost \_udaljenostOdSunca; // u AJ (astronomska jedinica) // Npr. 1.5 AJ

    vector<string>\* \_sateliti;

public:

    \_declspec (property (get = GetUdaljenostOdSunca, put = SetUdaljenostOdSunca)) Udaljenost UdaljenostOdSunca;

    \_declspec (property (get = GetSateliti, put = SetSateliti)) vector<string> Sateliti;

    //Z3.1

    Planeta();

    //Z3.2

    Planeta(const char\* naziv, Udaljenost radijus, Masa masa, const char\* sastav, Udaljenost doSunca, vector<string> sateliti);

    //Z3.3

    Planeta(const Planeta& obj);

    //Z3.4

    Planeta(Planeta&& obj);

    //Z3.5

    Udaljenost GetUdaljenostOdSunca() const;

    vector<string> GetSateliti() const;

    //Z3.6

    void SetUdaljenostOdSunca(Udaljenost udaljenost);

    void SetSateliti(vector<string> sateliti);

    //Z3.7

    Planeta& operator =(const Planeta& obj);

    //Z3.8

    //Dodavanje novog satelita (mjeseca)

    void operator +=(string satelit);

    //Z3.9

    void Ispis();

    //Z3.10

    ~Planeta();

};

//Z3.11

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

//Z3.12

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

enum class Boja { Crvena, Zuta, Plava, Bijela };

//Z4.0

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

enum class Velicina { Patuljak, Div, Superdiv };

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

class Zvijezda : public NebeskoTijelo {

private:

    unique\_ptr<float> \_absolutnaMagnituda; // Npr. 4.83

    unique\_ptr<int> \_povrsinskaTemperatura; // u kelvinima

    pair<Boja, Velicina> \_klasifikacija;

    list<Planeta> \_planetarniSistem;

public:

    \_declspec (property (get = GetAbsolutnaMagnituda, put = SetAbsolutnaMagnituda)) float AbsolutnaMagnituda;

    \_declspec (property (get = GetTemperatura, put = SetTemperatura)) int PovrsinskaTemperatura;

    \_declspec (property (get = GetBoja, put = SetBoja)) Boja Boja\_;

    \_declspec (property (get = GetVelicina, put = SetVelicina)) Velicina Velicina\_;

    \_declspec (property (get = GetPlanetarniSistem, put = SetPlanetarniSistem)) list<Planeta> Planete;

    //Z4.1

    Zvijezda();

    //Z4.2

    Zvijezda(const char\* nz, Udaljenost pp, Masa masa, const char\* sstv, float absMag, int tmp, pair<Boja, Velicina> klas);

    //Z4.3

    Zvijezda(const Zvijezda& obj);

    //Z4.4

    Zvijezda(Zvijezda&& obj);

    //Z4.5

    float GetAbsolutnaMagnituda() const;

    int GetTemperatura() const;

    Boja GetBoja() const;

    Velicina GetVelicina() const;

    list<Planeta> GetPlanetarniSistem() const;

    //Z4.6

    void SetAbsolutnaMagnituda(float absolutnaMagnituda);

    void SetTemperatura(int temperatura);

    void SetBoja(Boja boja);

    void SetVelicina(Velicina velicina);

    void SetPlanetarniSistem(list<Planeta> planete);

    //Z4.7

    void DodajPlanetu(Planeta p);

    //Z4.8

    Zvijezda& operator =(const Zvijezda& obj);

    //Z4.9

    void Ispis();

    //Z4.10

    ~Zvijezda();

};

//Z4.11

bool operator == (const Zvijezda& z1, const Zvijezda& z2);

//Z4.12

bool operator != (const Zvijezda& z1, const Zvijezda& z2);

//Z5.0

vector<NebeskoTijelo\*> Kopiraj(vector<NebeskoTijelo\*> izvor);

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

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

class Observatorija {

protected:

    unique\_ptr<char> \_naziv;

    const char\* \_lokacija;

    string \_kontaktTelefon;

    string\* \_email;

    vector<NebeskoTijelo\*> \_katalog;

public:

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

    \_declspec (property (get = GetLokacija, put = SetLokacija)) const char\* Lokacija;

    \_declspec (property (get = GetKontaktTelefon, put = SetKontaktTelefon)) string KontaktTelefon;

    \_declspec (property (get = GetEmail, put = SetEmail)) string Email;

    \_declspec (property (get = GetKatalog, put = SetKatalog)) vector<NebeskoTijelo\*> Katalog;

    //Z5.1

    Observatorija();

    //Z5.2

    Observatorija(const char\* naziv, const char\* lokacija, string kontaktTelefon, string email, vector<NebeskoTijelo\*> katalog);

    //Z5.3

    Observatorija(const Observatorija& obj);

    //Z5.4

    Observatorija(Observatorija&& obj);

    //Z5.5

    const char\* GetNaziv() const;

    const char\* GetLokacija() const;

    string GetKontaktTelefon() const;

    string GetEmail() const;

    vector<NebeskoTijelo\*> GetKatalog() const;

    //Z5.6

    void SetNaziv(const char\* naziv);

    void SetLokacija(const char\* lokacija);

    void SetKontaktTelefon(string telefon);

    void SetEmail(string email);

    void SetKatalog(vector<NebeskoTijelo\*> katalog);

    //Z5.7

    Observatorija& operator= (const Observatorija& obj);

    //Z5.8

    //Neka su dozvoljeni formati -->

    //+38762111222

    //+387 62 111-222

    //+(387)62 111-222

    bool ValidirajTelefon();

    //Z5.9

    //U prvom dijelu email-a, znak '.' ili '\_' smije se pojaviti maksimalno jednom (ostali specijalni znakovi su zabranjeni)

    // Prvo dio emaila ne smije pocinjati niti zavrsavati znakom '.' ili '\_'

    //Mora postojati znak '@'

    //Zavrsetak mora biti validna domena tipa: gmail.com, bih.net, outlook.com, edu.fit.ba

    bool ValidirajEmail();

    //Z5.10

    template<class T>

    void DodajNebeskoTijelo(T nebeskoTijelo);

    //Z5.11

    ~Observatorija();

};

//Z5.12

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

//Z5.13

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

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

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

};

void Zadatak1() {

    Udaljenost sirinaGalaksije; //Proxima Centauri

    sirinaGalaksije.Vrijednost = 130000.0f;

    sirinaGalaksije.Jedinica = JedinicaUdaljenosti::ly;

    cout << "Sirina galaksije Mlijecni put: " << sirinaGalaksije << endl;

    Udaljenost u1(sirinaGalaksije.Vrijednost, sirinaGalaksije.Jedinica);

    IspisiPoruku("User-def. ctor", sirinaGalaksije == u1);

    Udaljenost u2(sirinaGalaksije);

    IspisiPoruku("Copy ctor", sirinaGalaksije == u2);

    Udaljenost u3(move(u1));

    IspisiPoruku("Move ctor", sirinaGalaksije == u3);

    Udaljenost u4;

    u4 = sirinaGalaksije;

    IspisiPoruku("Operator '='", sirinaGalaksije == u4);

    cout << endl;

    Masa masaSjevernjace;

    masaSjevernjace.Jedinica = JedinicaMase::SM;

    masaSjevernjace.Vrijednost = 5.4f;

    cout << "Masa zvijezde Polaris: " << masaSjevernjace << endl;

    Masa m1(masaSjevernjace.Vrijednost, masaSjevernjace.Jedinica);

    IspisiPoruku("User-def. ctor", masaSjevernjace == m1);

    Masa m2(masaSjevernjace);

    IspisiPoruku("Copy ctor", masaSjevernjace == m2);

    Masa m3(move(m1));

    IspisiPoruku("Move ctor", masaSjevernjace == m3);

    Masa m4;

    m4 = masaSjevernjace;

    IspisiPoruku("Operator '='", masaSjevernjace == m4);

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

}

void Zadatak2() {

    NebeskoTijelo sagittariusA;

    sagittariusA.Naziv = "Sagittarius A\* - Supermasivna crna rupa";

    sagittariusA.Poluprecnik = Udaljenost(22000000.0f, JedinicaUdaljenosti::km);

    sagittariusA.Masa\_ = Masa(4.15e6, JedinicaMase::SM);

    sagittariusA.Sastav = "vodik, helij";

    sagittariusA.Ispis();

    NebeskoTijelo t1(sagittariusA.Naziv.c\_str(), sagittariusA.Poluprecnik, sagittariusA.Masa\_, sagittariusA.Sastav.c\_str());

    IspisiPoruku("User-def. ctor", sagittariusA == t1);

    NebeskoTijelo t2(sagittariusA);

    IspisiPoruku("Copy ctor", sagittariusA == t2);

    NebeskoTijelo t3(move(t1));

    IspisiPoruku("Move ctor", sagittariusA == t3);

    NebeskoTijelo t4;

    t4 = sagittariusA;

    IspisiPoruku("Operator '='", sagittariusA == t4);

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

}

void Zadatak3() {

    Planeta uran;

    uran.Naziv = "Uran";

    uran.Poluprecnik = Udaljenost(4.007f, JedinicaUdaljenosti::er);

    uran.Masa\_ = Masa(15.54f, JedinicaMase::EM);

    uran.Sastav = "vodik, helij, voda, metan, amonijak";

    uran.UdaljenostOdSunca = Udaljenost(19.21f, JedinicaUdaljenosti::au);

    uran += "Miranda";

    uran += "Umbriel";

    uran += "Titania";

    uran += "Oberon";

    uran.Ispis();

    cout << endl;

    Planeta p1(uran.Naziv.c\_str(), uran.Poluprecnik, uran.Masa\_, uran.Sastav.c\_str(), uran.UdaljenostOdSunca, uran.Sateliti);

    IspisiPoruku("User-def. ctor", uran == p1);

    Planeta p2(uran);

    IspisiPoruku("Copy ctor", uran == p2);

    Planeta p3(move(p1));

    IspisiPoruku("Move ctor", uran == p3);

    Planeta p4;

    p4 = uran;

    IspisiPoruku("Operator '='", uran == p4);

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

}

void Zadatak4() {

    Zvijezda vyCma;

    vyCma.Naziv = "VY Canis Majoris";

    vyCma.Poluprecnik = Udaljenost(2069.0f, JedinicaUdaljenosti::sr);

    vyCma.Masa\_ = Masa(17.5, JedinicaMase::SM);

    vyCma.Sastav = "vodik, helij, litij, ...";

    vyCma.AbsolutnaMagnituda = -0.72f;

    vyCma.PovrsinskaTemperatura = 3490;

    vyCma.Boja\_ = Boja::Crvena;

    vyCma.Velicina\_ = Velicina::Superdiv;

    Planeta p1("CMa-1", Udaljenost(0.5f, JedinicaUdaljenosti::er), Masa(1.3f, JedinicaMase::EM), "zeljezo", Udaljenost(0.7f, JedinicaUdaljenosti::au), {});

    Planeta p2("CMa-2", Udaljenost(0.7f, JedinicaUdaljenosti::er), Masa(2.0f, JedinicaMase::EM), "zeljezo", Udaljenost(5.3f, JedinicaUdaljenosti::au), {});

    vyCma.DodajPlanetu(p1);

    vyCma.DodajPlanetu(p2);

    vyCma.Ispis();

    Zvijezda z1(vyCma.Naziv.c\_str(), vyCma.Poluprecnik, vyCma.Masa\_, vyCma.Sastav.c\_str(),

        vyCma.AbsolutnaMagnituda, vyCma.PovrsinskaTemperatura, pair<Boja, Velicina>(vyCma.Boja\_, vyCma.Velicina\_));

    z1.Planete = vyCma.Planete;

    IspisiPoruku("User-def. ctor", vyCma == z1);

    Zvijezda z2(vyCma);

    IspisiPoruku("Copy ctor", vyCma == z2);

    Zvijezda z3(move(z1));

    IspisiPoruku("Move ctor", vyCma == z3);

    Zvijezda z4;

    z4 = vyCma;

    IspisiPoruku("Operator '='", vyCma == z4);

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

}

void Zadatak5() {

    Observatorija zvjezdarnica;

    zvjezdarnica.Naziv = "Zvjezdarnica";

    zvjezdarnica.Lokacija = "Sarajevo, BiH";

    zvjezdarnica.Email = "zvjezdarnica\_ba@gmail.com";

    zvjezdarnica.KontaktTelefon = "+(387)62 546-241";

    NebeskoTijelo messier61;

    messier61.Naziv = "Messier 61 - Supermasivna crna rupa";

    messier61.Poluprecnik = Udaljenost(133000000.0f, JedinicaUdaljenosti::km);

    messier61.Masa\_ = Masa(70.15e6, JedinicaMase::SM);

    messier61.Sastav = "vodik, helij";

    Planeta nibiru;

    nibiru.Naziv = "Nibiru - planeta Anunnakija";

    nibiru.Poluprecnik = Udaljenost(2.607f, JedinicaUdaljenosti::er);

    nibiru.Masa\_ = Masa(21.3f, JedinicaMase::EM);

    nibiru.Sastav = "voda, zeljezo, amonijak";

    nibiru.UdaljenostOdSunca = Udaljenost(10.3f, JedinicaUdaljenosti::au);

    nibiru.Sateliti = vector<string>();

    Zvijezda uyScuti;

    uyScuti.Naziv = "UY Scuti";

    uyScuti.Poluprecnik = Udaljenost(3500.0f, JedinicaUdaljenosti::sr);

    uyScuti.Masa\_ = Masa(19.45f, JedinicaMase::SM);

    uyScuti.Sastav = "vodik, helij";

    uyScuti.AbsolutnaMagnituda = -0.69f;

    uyScuti.PovrsinskaTemperatura = 4150;

    uyScuti.Boja\_ = Boja::Crvena;

    uyScuti.Velicina\_ = Velicina::Superdiv;

    zvjezdarnica.DodajNebeskoTijelo<NebeskoTijelo>(messier61);

    zvjezdarnica.DodajNebeskoTijelo<Planeta>(nibiru);

    zvjezdarnica.DodajNebeskoTijelo<Zvijezda>(uyScuti);

    cout << zvjezdarnica << endl;

    cout << "Telefon je u " << (zvjezdarnica.ValidirajTelefon() ? "validnom" : "nevalidnom") << " formatu" << endl;

    cout << "Email je u " << (zvjezdarnica.ValidirajEmail() ? "validnom" : "nevalidnom") << " formatu" << endl;

    Observatorija o1(zvjezdarnica.Naziv, zvjezdarnica.Lokacija, zvjezdarnica.KontaktTelefon.c\_str(), zvjezdarnica.Email.c\_str(), zvjezdarnica.Katalog);

    IspisiPoruku("User-def. ctor", zvjezdarnica == o1);

    Observatorija o2(zvjezdarnica);

    IspisiPoruku("Copy ctor", zvjezdarnica == o2);

    Observatorija o3(move(o1));

    IspisiPoruku("Move ctor", zvjezdarnica == o3);

    Observatorija o4;

    o4 = zvjezdarnica;

    IspisiPoruku("Operator '='", zvjezdarnica == o4);

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

}

int main() {

    int nastaviDalje = 1;

    while (nastaviDalje) {

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

            cin >> izbor;

            cout << endl;

        } while (izbor <= 0 || izbor > 5);

        switch (izbor) {

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

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

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

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

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

        default:

            break;

        }

        do {

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

            cin >> nastaviDalje;

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

    }

    return 0;

}