#include <vector>

#include <iostream>

#include <cassert>

#include <functional>

using namespace std;

//Z0.1

char\* Alokator(const char\* tekst);

//Z0.2

int IzracunajBrojZnamenki(int broj);

//Z0.3 :: Pretvoriti (int) u (char\*).

char\* IntToStr(int broj);

//Z0.4 :: Funkcija za alokaciju pametnog (shared\_ptr) pokazivaca na niz karaktera

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

//Z0.5 :: Funkcija za alokaciju pametnog (unique\_ptr) pokazivaca na niz karaktera

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

//Z0.6:: Provjeriti da li su dva niza karaktera ista (po vrijednosti)

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

//Z0.7 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);

//Z0.8 Provjeriti da li su dva vektora nizova karaktera jednaka (da li imaju iste elemente)

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

//Po mogucnosti postavljati 'assert' tamo gdje se radi dereferenciranje pokazivaca

//Za postojece objekte mozete koristiti property-je za pristup vrijednostima atributa

//Za ispravno koristenje property-ja, uraditi implementaciju odgovarajucih gettera i settera

class Uredjaj {

private:

    const char\* \_id;

protected:

    unique\_ptr<char> \_naziv;

    int \_godinaProizvodnje;

    shared\_ptr<float> \_cijena;

    int\* \_garancija;

    vector<int> \_ocjene;

public:

    \_\_declspec(property (get = GetId, put = SetId)) const char\* Id;

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

    \_\_declspec(property (get = GetGodinaProizvodnje, put = SetGodinaProizvodnje)) int GodinaProizvodnje;

    \_\_declspec(property (get = GetCijena, put = SetCijena)) float Cijena;

    \_\_declspec(property (get = GetGarancija, put = SetGarancija)) int Garancija;

    \_\_declspec(property (get = GetOcjene, put = SetOcjene)) vector<int> Ocjene;

    //Z1.0 :: Staticka metoda koja vraca niz karaktera (brojeva) u opsegu "10000" do "99999"

    static const char\* GetRandomId();

    //Z1.1 :: Postaviti '\_id' na rezultat koji vrati staticka funkcija 'GetRandomId'

    Uredjaj();

    //Z1.2

    Uredjaj(const char\* naziv, int godinaProizvodnje, float cijena, int garancija);

    //Z1.3 :: mozete koristiti property-je drugog (vec postojeceg) objekta za pristup vrijednostima njegovih atributa

    //Kopirati sve atribute, ukljucujuci i polje '\_id'

    Uredjaj(const Uredjaj& obj);

    //Z1.4

    Uredjaj(Uredjaj&& obj);

    //Z1.5

    const char\* GetId() const;

    const char\* GetNaziv() const;

    int GetGodinaProizvodnje() const;

    float GetCijena() const;

    int GetGarancija() const;

    vector<int> GetOcjene() const;

    //Z1.6

    void SetId(const char\* id);

    void SetNaziv(const char\* naziv);

    void SetGodinaProizvodnje(int godinaProizvodnje);

    void SetCijena(float cijena);

    void SetGarancija(int garancija);

    void SetOcjene(vector<int> ocjene);

    //Z1.7 :: mozete koristiti property-je za oba objekta (obzirom da vec postoje)

    Uredjaj& operator =(const Uredjaj& obj);

    //Z1.8

    void AddOcjena(int ocjena);

    //Z1.9 :: Mozete koristiti property-je obzirom da objekt vec postoji

    float GetProsjecnaOcjena(function<float(vector<int>)> tipProsjeka);

    //Z1.10 :: implementacija funkcije 'Ispis' se prepisuje (override-a) sa implementacijom istoimene funkcije  najizvedenije klase

    // Mozete koristiti property-je obzirom da objekt vec postoji

    virtual void Ispis();

    //Z1.11

    virtual ~Uredjaj();

};

//Z1.12

//Iskljuciti polje 'Id' iz uporedbe

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

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

class Mobitel : public Uredjaj {

private:

    char \_operativniSistem[50];

    unique\_ptr<float> \_velicinaEkrana;//u incima Npr. 6.1

    int\* \_memorija; //GB

public:

    \_\_declspec(property (get = GetOperativniSistem, put = SetOperativniSistem)) const char\* OperativniSistem;

    \_\_declspec(property (get = GetVelicinaEkrana, put = SetVelicinaEkrana)) float VelicinaEkrana;

    \_\_declspec(property (get = GetMemorija, put = SetMemorija)) int Memorija;

    //Z3.1

    Mobitel();

    //Z2.2

    Mobitel(const char\* naziv, int godinaP, float cijena, int garancija, const char\* operativniS, float velicinaE, int memorija);

    //Z2.3

    Mobitel(const Mobitel& mobitel);

    //Z2.4

    Mobitel(Mobitel&& obj);

    const char\* GetOperativniSistem()const;

    float GetVelicinaEkrana()const;

    int GetMemorija()const;

    void SetOperativniSistem(const char\* operativniSistem);

    void SetVelicinaEkrana(float velicinaEkrana);

    void SetMemorija(int memorija);

    //Z2.5

    Mobitel& operator = (const Mobitel& obj);

    //Z2.6

    //Ispisati id, naziv mobitela, velicinu ekrana, operativni sistem, kolicinu memorije, cijenu te broj godina garancije

    void Ispis();

    //Z2.7

    ~Mobitel();

};

//Z2.8

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

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

class Televizor : public Uredjaj {

    unique\_ptr<float> \_velicinaEkrana;

    unique\_ptr<char> \_tipEkrana;

    pair<int, int> \_rezolucija;

    vector<const char\*> \_prikljucci;

public:

    \_\_declspec(property (get = GetVelicinaEkrana, put = SetVelicinaEkrana)) float VelicinaEkrana;

    \_\_declspec(property (get = GetTipEkrana, put = SetTipEkrana)) const char\* TipEkrana;

    \_\_declspec(property (get = GetRezolucija, put = SetRezolucija)) pair<int, int> Rezolucija;

    \_\_declspec(property (get = GetPrikljucci, put = SetPrikljucci)) vector<const char\*> Prikljucci;

    //Z3.1

    Televizor();

    //Z3.2

    Televizor(const char\* naziv, int godinaP, float cijena, int garancija,

        float velicinaEkrana, const char\* tipEkrana, pair<int, int> rezolucija);

    //Z3.3

    Televizor(const Televizor& obj);

    //Z3.4

    Televizor(Televizor&& obj);

    float GetVelicinaEkrana()const;

    const char\* GetTipEkrana()const;

    pair<int, int> GetRezolucija()const;

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

    void SetVelicinaEkrana(float velicinaEkrana);

    void SetTipEkrana(const char\* tipEkrana);

    void SetRezolucija(pair<int, int> rezolucija);

    void SetPrikljucci(vector<const char\*>prikljucci);

    //Z3.5

    Televizor& operator = (const Televizor& obj);

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

    //Z3.6

    //Ispisati id, naziv TV-a, velicinu ekrana, tip ekrana, rezoluciju, cijenu, broj godina garancije, te sve prikljucke

    void Ispis();

    //Z3.7

    ~Televizor();

};

bool operator == (const Televizor& t1, const Televizor& t2);

bool operator != (const Televizor& t1, const Televizor& t2);

class Printer : public Uredjaj {

    int\* \_brzinaPrintanja; // br. str. po minuti

public:

    \_\_declspec(property (get = GetBrzinaPrintanja, put = SetBrzinaPrintanja)) int BrzinaPrintanja;

    //Z4.1

    Printer();

    //Z4.2

    Printer(const char\* naziv, int godinaP, float cijena, int garancija, int brzinaPrintanja);

    //Z4.3

    Printer(const Printer& obj);

    //Z4.4

    Printer(Printer&& obj);

    int GetBrzinaPrintanja() const;

    void SetBrzinaPrintanja(int brzinaPrintanja);

    //Z4.5

    Printer& operator= (const Printer& obj);

    //Z4.6

    //Ispisati id, naziv, brzinu printanja, cijenu te broj godina garancije

    void Ispis();

    //Z4.7

    ~Printer();

};

//Z4.8

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

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

class WebShop {

    char\* \_naziv;

    char \_emailAdresa[100];

    int \_trenutnoUredjaja;

    Uredjaj\* \_uredjaji[100] = { nullptr };

public:

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

    \_\_declspec(property (get = GetEmailAdresa, put = SetEmailAdresa)) const char\* EmailAdresa;

    \_\_declspec(property (get = GetTrenutnoUredjaja)) int TrenutnoUredjaja;

    //Z5.1

    WebShop();

    //Z5.2

    WebShop(const char\* naziv, const char\* emailAdresa);

    //Z5.3

    WebShop(const WebShop& obj);

    //Z5.4

    WebShop(WebShop&& obj);

    const char\* GetNaziv() const;

    const char\* GetEmailAdresa() const;

    int GetTrenutnoUredjaja() const;

    Uredjaj\* GetElementAt(int index) const;

    void SetNaziv(const char\* naziv);

    void SetEmailAdresa(const char\* emailAdresa);

    //Z5.5

    WebShop& operator = (const WebShop& obj);

    //Z5.6 :: Dodavanje novog elementa u niz pokazivaca

    template <class T>

    bool DodajUredjaj(T uredjaj);

    //Z5.7

    ~WebShop();

    friend ostream& operator << (ostream& COUT, const WebShop& webShop);

};

//Z5.8

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

//Z5.9

bool operator == (const WebShop& shop1, const WebShop& shop2);

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

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

};

void Zadatak1() {

    //Generator ocjena

    const int max = 20;

    vector<int> ocjene;

    for (int i = 0; i < max; i++)

        ocjene.push\_back(rand() % 5 + 1);

    //Testiranje klase 'Uredjaj'

    Uredjaj monitor;

    monitor.Naziv = "LED monitor";

    monitor.GodinaProizvodnje = 2017;

    monitor.Cijena = 250;

    monitor.Garancija = 2;

    monitor.Ocjene = ocjene;

    monitor.Ispis();

    cout << endl;

    Uredjaj monitor2("LED monitor", 2017, 250, 2);

    monitor2.Ocjene = monitor.Ocjene;

    IspisiPoruku("User-def. ctor", monitor == monitor2);

    Uredjaj monitor3(monitor);

    IspisiPoruku("Copy ctor", monitor == monitor3);

    Uredjaj monitor4(move(monitor2));

    IspisiPoruku("Move ctor", monitor == monitor4);

    Uredjaj monitor5;

    monitor5 = monitor;

    IspisiPoruku("Operator '='", monitor == monitor5);

    auto aritmetickaSredina = [](vector<int> brojevi) {

        float suma = 0.0f;

        if (brojevi.size() == 0)

            return suma;

        for (size\_t i = 0; i < brojevi.size(); i++)

            suma += brojevi[i];

        return suma / brojevi.size();

    };

    cout << "Prosjecna ocjena: " << monitor.GetProsjecnaOcjena(aritmetickaSredina) << endl;

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

}

void Zadatak2() {

    Mobitel ironPhone;

    ironPhone.Naziv = "Iron Phone S Ultra";

    ironPhone.GodinaProizvodnje = 2021;

    ironPhone.Cijena = 1200;

    ironPhone.Garancija = 2;

    ironPhone.OperativniSistem = "Onyx";

    ironPhone.VelicinaEkrana = 6.1f;

    ironPhone.Memorija = 16;

    ironPhone.Ispis();

    cout << endl;

    Mobitel ironPhone2("Iron Phone S Ultra", 2021, 1200, 2, "Onyx", 6.1f, 16);

    IspisiPoruku("User-def. ctor", ironPhone == ironPhone2);

    Mobitel ironPhone3(ironPhone);

    IspisiPoruku("Copy ctor", ironPhone == ironPhone3);

    Mobitel ironPhone4(move(ironPhone2));

    IspisiPoruku("Move ctor", ironPhone == ironPhone4);

    Mobitel ironPhone5;

    ironPhone5 = ironPhone;

    IspisiPoruku("Operator '='", ironPhone == ironPhone5);

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

}

void Zadatak3() {

    vector<const char\*> prikljucci;

    prikljucci.push\_back("USB-C");

    prikljucci.push\_back("HDMI");

    prikljucci.push\_back("S-Video");

    Televizor chromethius;

    chromethius.Naziv = "Chromethius";

    chromethius.GodinaProizvodnje = 2020;

    chromethius.Cijena = 700;

    chromethius.Garancija = 2;

    chromethius.VelicinaEkrana = 43.5;

    chromethius.TipEkrana = "LED";

    chromethius.Rezolucija = pair<int, int>(3840, 2160);

    chromethius.Prikljucci = prikljucci;

    chromethius.Ispis();

    cout << endl;

    Televizor chromethius2("Chromethius", 2020, 700, 2, 43.5, "LED", pair<int, int>(3840, 2160));

    chromethius2.Prikljucci = chromethius.Prikljucci;

    IspisiPoruku("User-def. ctor", chromethius == chromethius2);

    Televizor chromethius3(chromethius);

    IspisiPoruku("Copy ctor", chromethius == chromethius3);

    Televizor chromethius4(move(chromethius2));

    IspisiPoruku("Move ctor", chromethius == chromethius4);

    Televizor chromethius5;

    chromethius5 = chromethius;

    IspisiPoruku("Operator '='", chromethius == chromethius5);

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

}

void Zadatak4() {

    Printer optomec;

    optomec.Naziv = "Optomec X-40";

    optomec.GodinaProizvodnje = 2021;

    optomec.Cijena = 550;

    optomec.Garancija = 2;

    optomec.BrzinaPrintanja = 30;

    optomec.Ispis();

    //Testiranje user-def. ctora

    Printer optomec2("Optomec X-40", 2021, 550, 2, 30);

    IspisiPoruku("User-def. ctor", optomec == optomec2);

    //Testiranje copy ctora

    Printer optomec3(optomec);

    IspisiPoruku("Copy ctor", optomec == optomec3);

    //Testiranje move ctora

    Printer optomec4(move(optomec2));

    IspisiPoruku("Move ctor", optomec == optomec4);

    //Testiranje operatora '='

    Printer optomec5;

    optomec5 = optomec;

    IspisiPoruku("Operator '='", optomec == optomec5);

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

}

void Zadatak5() {

    //WebShop

    WebShop bosnianVortex;

    bosnianVortex.Naziv = "Bosnian Vortex";

    bosnianVortex.EmailAdresa = "bosnian.vortex@gmail.com";

    //Uredjaji

    Uredjaj napojna("Corsair Napojna jedinica", 2018, 159.0, 2);

    Mobitel wikkoJerry("WIKO JERRY 3 ANTHRACITE", 2019, 149.0, 2, "Android 8.1", 5.45f, 16);

    Televizor phillipsTV("PHILIPS LED TV AMBILIGHT", 2019, 899.0, 3, 43.0, "LED", pair<int, int>(3840, 2160));

    Printer officeJet("B-HP OFFICEJET 202", 2019, 549.0, 2, 20);

    //Dodavanje razlicitih tipova uredjaja

    bosnianVortex.DodajUredjaj<Uredjaj>(napojna); //Mozete izostaviti naziv generickog tipa tj: bosnianVortex.DodajUredjaj(napojna);

    bosnianVortex.DodajUredjaj<Mobitel>(wikkoJerry);

    bosnianVortex.DodajUredjaj<Televizor>(phillipsTV);

    bosnianVortex.DodajUredjaj<Printer>(officeJet);

    //Testiranje user-def. ctora

    WebShop bosnianVortex2("Bosnian Vortex", "bosnian.vortex@gmail.com");

    bosnianVortex2.DodajUredjaj(napojna);

    bosnianVortex2.DodajUredjaj(wikkoJerry);

    bosnianVortex2.DodajUredjaj(phillipsTV);

    bosnianVortex2.DodajUredjaj(officeJet);

    cout << bosnianVortex << endl;

    IspisiPoruku("User-def. ctor", bosnianVortex2 == bosnianVortex);

    //Testiranje copy ctora

    WebShop bosnianVortex3(bosnianVortex);

    IspisiPoruku("Copy ctor", bosnianVortex3 == bosnianVortex);

    //Testiranje move ctora

    WebShop bosnianVortex4(move(bosnianVortex2));

    IspisiPoruku("Move ctor", bosnianVortex4 == bosnianVortex);

    //Testiranje operatora '='

    WebShop bosnianVortex5;

    bosnianVortex5 = bosnianVortex;

    IspisiPoruku("Operator '='", bosnianVortex5 == bosnianVortex);

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

            cin >> izbor;

            cout << endl;

        } while (izbor < 1 || izbor > 5);

        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;

        default:break;

        }

        do {

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

            cin >> nastaviDalje;

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

    }

}

int main() {

    Menu();

    return 0;

}