1. #include<iostream>

#include "Source.h"

using namespace std;

//U programu implementirati nasljeðivanje izmeðu odgovarajuæih klasa

//Klasu deklarisati kao apstraktnu

class Osoba

{

protected:

char\* \_ime;

char\* \_prezime;

char\* \_adresa;

char \_telefon[15];

public:

//Potrebne konstruktor i destruktor funkcije

Osoba(char\* ime, char\* prezime, char\* adresa, char\* telefon)

{

int vel = strlen(ime) + 1;

\_ime = new char[vel];

strcpy\_s(\_ime, vel, ime);

vel = strlen(prezime) + 1;

\_prezime = new char[vel];

strcpy\_s(\_prezime, vel, prezime);

vel = strlen(adresa) + 1;

\_adresa = new char[vel];

strcpy\_s(\_adresa, vel, adresa);

strcpy\_s(\_telefon, 15, telefon);

}

virtual ~Osoba() = 0

{

delete[] \_ime;

\_ime = NULL;

delete[] \_prezime;

\_prezime = NULL;

delete[] \_adresa;

\_adresa = NULL;

}

//Kreirati funkciju Info za ispis svih vrijednosti atributa klase

virtual void Info() = 0

{

cout << "Ime i prezime: " << \_ime << " " << \_prezime << endl;

cout << "Adresa: " << \_adresa << endl;

cout << "Telefon: " << \_telefon << endl;

}

};

static int pom = 1;

class Klijent : public Osoba

{

protected:

int \_id;

double \_mjesecnaPotrosnja;

char\* \_email;

public:

char \* getIme() { return this->\_ime; }

char \* getPrezime() { return this->\_prezime; }

//Potrebne konstruktor i destruktor funkcije

Klijent(char\* ime = "---", char\* prezime = "---", char\* adresa = "---", char\* telefon = "---",

double potrosnja = 0, char \*mail = "---") : Osoba(ime, prezime, adresa, telefon) {

\_id = pom++;

\_mjesecnaPotrosnja = potrosnja;

\_email = new char[strlen(mail) + 1];

strcpy\_s(\_email, strlen(mail) + 1, mail);

}

~Klijent() {

delete[]\_email; \_email = nullptr;

}

Klijent(Klijent & orig) :Osoba(orig) {

\_id = orig.\_id;

\_mjesecnaPotrosnja = orig.\_mjesecnaPotrosnja;

\_email = new char[strlen(orig.\_email) + 1];

strcpy\_s(\_email, strlen(orig.\_email) + 1, orig.\_email);

}

//Kreirati funkciju za izmjenu mail adrese.

void IzmijeniMail(char \*novi) {

delete[]\_email;

\_email = new char[strlen(novi) + 1];

strcpy\_s(\_email, strlen(novi) + 1, novi);

}

void setMail(char \* Mail) {

int temp = strlen(Mail) + 1;

this->\_email = new char[temp];

strcpy\_s(this->\_email, temp, Mail);

}

char \* GetMail() { return this->\_email; }

double getPotrosnja() { return \_mjesecnaPotrosnja; }

void Info() {

cout << "Ime i prezime: " << \_ime << " " << \_prezime << endl;

cout << "Adresa: " << \_adresa << endl;

cout << "Telefon: " << \_telefon << endl;

cout << "Klijent ID: " << \_id << endl;

cout << "Mjesecna potrosnja: " << \_mjesecnaPotrosnja << endl;

cout << "E-mail: " << \_email << endl;

}

};

class Zaposlenik : public Osoba {

protected:

char \_radnoMjesto[30];

int \_godineStaza;

public:

//Potrebne konstruktor i destruktor funkcije

Zaposlenik(char\* ime = "---", char\* prezime = "---", char\* adresa = "---", char\* telefon = "---", char \*rm = "--", int staz = 0) :

Osoba(ime, prezime, adresa, telefon)

{

strcpy\_s(\_radnoMjesto, 29, rm);

\_godineStaza = staz;

}

~Zaposlenik() {

cout << "Brisem zaposlenika\n";

}

Zaposlenik(Zaposlenik & orig) :Osoba(orig) {

strcpy\_s(\_radnoMjesto, 29, orig.\_radnoMjesto);

\_godineStaza = orig.\_godineStaza;

}

char \* GetRM() {

return \_radnoMjesto;

}

void Info() {

Osoba::Info();

cout << "Radno mjesto: " << \_radnoMjesto << endl;

cout << "Staz: " << \_godineStaza << endl;

}

};

class VIPKlijent :public Klijent

{

double \_popust;

char\* \_parkingOznaka;

public:

//Potrebne konstruktor i destruktor funkcije

VIPKlijent(char\* ime = "---", char\* prezime = "---", char\* adresa = "---", char\* telefon = "---",

double popust = 0, char \*parking = "--") :

Klijent(ime, prezime, adresa, telefon)

{

\_popust = popust;

\_parkingOznaka = new char[strlen(parking) + 1];

strcpy\_s(\_parkingOznaka, strlen(parking) + 1, parking);

}

~VIPKlijent() {

delete[]\_parkingOznaka;

\_parkingOznaka = nullptr;

}

VIPKlijent(VIPKlijent & orig) :Klijent(orig) {

\_popust = orig.\_popust;

\_parkingOznaka = new char[strlen(orig.\_parkingOznaka) + 1];

strcpy\_s(\_parkingOznaka, strlen(orig.\_parkingOznaka) + 1, orig.\_parkingOznaka);

}

//Funkciju koja treba da izmijeni vrijednost mjeseène potrošnje na naèin da se

//uraèuna popust (prosjeèna potrošnja treba da se umanji)

void SmanjiPotrosnju() {

\_mjesecnaPotrosnja -= \_mjesecnaPotrosnja\*\_popust;

}

//Funkciju za ispis svih vrijednosti atributa klase

void Info() {

Klijent::Info();

cout << "Popust: " << \_popust << endl;

cout << "Parking oznaka: " << \_parkingOznaka << endl;

}

};

//Funkcija treba da pronaðe sve one klijente koji su ostvarili mjeseènu potrošnju veću

//od zadate. Ispisati podatke i vratiti ukupan broj pronaðenih klijenata.

int pretragaPotrosnjeKlijenata(Osoba\* niz[], int max) {

cout << "Pretraga Potrosnje Klijenata\n";

double potrosnja;

int brojac = 0;

cout << "Unesite potrosnju koju trazite: ";

cin >> potrosnja;

Klijent \*pok = nullptr;

for (int i = 0; i<max; i++) {

pok = dynamic\_cast<Klijent \*>(niz[i]);

if (pok != NULL && pok->getPotrosnja()>potrosnja) {

pok->Info();

brojac++;

}

}

return brojac;

}

//Pronaæi sve VIP klijente, te im zaraèunati popust na kupljene artikle i ispisati

//informacije o onima koji su ostvarili potrošnju veæu od 50 KM mjeseèno.

void pretragaVIPKlijenata(Osoba\* niz[], int max) {

cout << "Pretraga VIP Klijenta\n";

VIPKlijent \*pok = nullptr;

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

{

pok = dynamic\_cast<VIPKlijent \*>(niz[i]);

if (pok != nullptr) {

if (pok->getPotrosnja()>50) {

pok->SmanjiPotrosnju();

pok->Info();

}

}

}

}

//Svim klijentima postaviti mail adresu u formatu: ime.prezime@firma.ba.

void generisiMailAdrese(Osoba\* niz[], int max) {

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

{

Klijent \*pok = nullptr;

for (int i = 0; i<max; i++) {

pok = dynamic\_cast<Klijent \*>(niz[i]);

if (pok != NULL) {

char \*novi = new char[100];

strcpy\_s(novi, strlen(pok->getIme()) + 1, pok->getIme());

strcat\_s(novi, 100, ".");

strcat\_s(novi, 100, pok->getPrezime());

strcat\_s(novi, 100, "@firma.ba");

pok->setMail(novi);

}

}

}

}

//Omoguæiti pretragu klijenata po imenu ili prezimenu (ispisati informacije

//o svima koji sadrže uneseni tekst u dijelu imena ili prezimena.

void pretragaKlijenataPoImenu(Osoba\* niz[], int max) {

cout << "\n=========================== PRETRAGA KLIJENATA PO IMENU ================================\n";

cin.ignore();

char ime[20];

cout << "Unesite ime ili prezime: " << endl;

cin.getline(ime, 20);

char \* pom = new char[strlen(ime) + 1];

strcpy\_s(pom, strlen(ime) + 1, ime);

Klijent \*pok = nullptr;

for (int i = 0; i<max; i++) {

pok = dynamic\_cast<Klijent \*>(niz[i]);

if (pok != NULL && (strstr(pok->getIme(), pom) || strstr(pok->getPrezime(), pom))) {

pok->Info();

}

}

}

//Omoguæiti pretragu zaposlenika na osnovu radnog mjesta koje definiše korisnik.

void pretragaZaposlenika(Osoba\* niz[], int max) {

cout << "\n=========================== PRETRAGA ZAPOSLENIKA ================================\n";

char ime[20];

cout << " Unesite radno mjesto: " << endl;

cin.getline(ime, 20);

char \* pom = new char[strlen(ime) + 1];

strcpy\_s(pom, strlen(ime) + 1, ime);

Zaposlenik \*pok = nullptr;

for (int i = 0; i<max; i++) {

pok = dynamic\_cast<Zaposlenik \*>(niz[i]);

if (pok != nullptr && !strcmp(pok->GetRM(), pom)) {

pok->Info();

}

}

}

void main()

{

//Testirati sve dostupne funkcionalnosti uz obaveznu provjeru globalnih funkcija

char ime[100], prezime[100], adresa[50], telefon[20], mail[60], rm[70], parking[70];

double potrosnja,popust;

int staz;

Osoba\* niz[4];

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

cout << "Unesite ime: ";

cin.getline(ime, 100);

cout << "Unesite prezime: ";

cin.getline(prezime, 100);

cout << "Unesite email: ";

cin.getline(mail, 60);

cout << "Unesite adresu: ";

cin.getline(adresa, 50);

cout << "Unesite broj telefona: ";

cin.getline(telefon, 20);

cout << "Unesite potrosnju: ";

cin >> potrosnja;

cin.ignore();

niz[0] = new Klijent(ime,prezime,adresa,telefon,potrosnja,mail);

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

cout << "Unesite ime: ";

cin.getline(ime, 100);

cout << "Unesite prezime: ";

cin.getline(prezime, 100);

cout << "Unesite email: ";

cin.getline(mail, 60);

cout << "Unesite adresu: ";

cin.getline(adresa, 50);

cout << "Unesite broj telefona: ";

cin.getline(telefon, 20);

cout << "Unesite naziv radnog mjesta: ";

cin.getline(rm, 70);

cout << "Unesite staz: ";

cin >> staz;

cin.ignore();

niz[1] = new Zaposlenik(ime, prezime, adresa, telefon,rm,staz);

cout << "\n============================== VIP KLIJENT ===============================\n\n";

cout << "Unesite ime: ";

cin.getline(ime, 100);

cout << "Unesite prezime: ";

cin.getline(prezime, 100);

cout << "Unesite email: ";

cin.getline(mail, 60);

cout << "Unesite adresu: ";

cin.getline(adresa, 50);

cout << "Unesite broj telefona: ";

cin.getline(telefon, 20);

cout << "Unesite naziv parking mjesta: ";

cin.getline(parking, 70);

cout << "Unesite staz: ";

cin >> popust;

cin.ignore();

niz[2] = new VIPKlijent(ime, prezime, adresa, telefon,popust,parking);

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

cout << "Unesite ime: ";

cin.getline(ime, 100);

cout << "Unesite prezime: ";

cin.getline(prezime, 100);

cout << "Unesite email: ";

cin.getline(mail, 60);

cout << "Unesite adresu: ";

cin.getline(adresa, 50);

cout << "Unesite broj telefona: ";

cin.getline(telefon, 20);

cout << "Unesite potrosnju: ";

cin >> potrosnja;

cin.ignore();

niz[3] = new Klijent(ime, prezime, adresa, telefon, potrosnja, mail);

generisiMailAdrese(niz, 4);

pretragaKlijenataPoImenu(niz, 4);

pretragaZaposlenika(niz, 4);

system("pause>0");

}

2.

#include<iostream>

using namespace std;

class Artikal {

char \*sifra;

char \*naziv;

double cijena;

int kolicinaNaSkladistu;

public:

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

Artikal(const char code[4] = "---", const char \_naziv[] = "---", double \_cijena = 0, int \_kolicina = 0) {

sifra = new char[strlen(code) + 1];

strcpy\_s(sifra, strlen(code) + 1, code);

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

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

cijena = \_cijena;

kolicinaNaSkladistu = \_kolicina;

}

Artikal(const Artikal &original) {

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

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

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

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

this->cijena = original.cijena;

kolicinaNaSkladistu = original.kolicinaNaSkladistu;

}

~Artikal() {

delete[]sifra; sifra = nullptr;

delete[]naziv; naziv = nullptr;

}

Artikal& operator =(Artikal &original) {

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

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

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

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

this->cijena = original.cijena;

kolicinaNaSkladistu = original.kolicinaNaSkladistu;

return \*this;

}

//UmanjiKolicinuNaSkladistu na način da se stanje na skladištu artikla promijeni za proslijeđenu vrijednost

//funkcija treba da bude tipa bool, tj vraća informaciju o tome da li je uspješno promjenjeno stanje na skladištu

bool UmanjiKolicinuNaSkladistu(int kolicina) {

if (kolicina > kolicinaNaSkladistu) {

cout << "Nedovoljno kolicine na skladistu.\n";

return false;

}

kolicinaNaSkladistu -= kolicina;

return true;

}

//Operator UvecajKolicinuNaSkladistu na način da se stanje na skladištu artikla promijeni za proslijeđenu vrijednost

void UvecajKolicinuNaSkladistu(int kolicina) {

kolicinaNaSkladistu += kolicina;

}

//UvecajCijenu na način da se cijena artikla uveća za 1

void UvecajCijenu() {

cijena++;

}

//Operator UmanjiCijenu na način da se cijena artikla umanji za 1

void UmanjiCijenu() {

cijena--;

}

//AreEqual koji poredi artikle na osnovu sifre

bool AreEqual(char \*sifra) {

return strcmp(this->sifra, sifra) == 0;

}

//Info

void Info() {

cout << "Artikal: " << naziv << endl;

cout << "Šifra: " << sifra << endl;

cout << "Cijena: " << cijena << " KM" << endl;

cout << "Količina na skladištu: " << kolicinaNaSkladistu << endl;

}

void Unos() {

char temp[100];

cout << "Unesite naziv artikla: ";

cin.getline(temp, 100);

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

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

cout << "Unesite šifru artikla: ";

cin.getline(temp, 100);

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

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

cout << "Unesite cijenu artikla: ";

cin >> cijena;

cout << "Unesite količinu na skladištu: ";

cin >> kolicinaNaSkladistu;

cin.ignore();

}

int GetKolicinaNaSkladistu() { return kolicinaNaSkladistu; }

double GetCijena() { return cijena; }

char \*GetSifra() { return sifra; }

};

class StavkaRacuna {

Artikal artikal;

int \*kolicina;

public:

StavkaRacuna(int kolicina = 0) {

this->kolicina = new int(kolicina);

}

StavkaRacuna(int kolicina , Artikal artikal) {

this->kolicina = new int(kolicina);

this->artikal = artikal;

}

StavkaRacuna(StavkaRacuna &original) {

this->artikal = original.artikal;

kolicina = new int(\*original.kolicina);

}

StavkaRacuna& operator = (StavkaRacuna &original) {

this->artikal = original.artikal;

delete kolicina;

kolicina = new int;

\*kolicina = \*original.kolicina;

return \*this;

}

~StavkaRacuna()

{

delete kolicina;

}

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

//operator dodjele =

//UvecajStanjeNaSkladistu koji uvećava stanje na skladištu artikla

void UvecajStanjeNaSkladistu(int kolicina) {

artikal.UvecajKolicinuNaSkladistu(kolicina);

}

//UmanjiStanjeNaSkladistu koji umanjuje stanje na skladištu artikla

void UmanjiStanjeNaSkladistu(int kolicina) {

artikal.UmanjiKolicinuNaSkladistu(kolicina);

}

//AreEqual koji poredi stavke po sifri artikla

bool AreEqual(char\* sifra) {

return artikal.AreEqual(sifra);

}

// UvecajKolicinuStavke koji uvećava količinu stavke racuna

void UvecajKolicinuStavke() {

(\*kolicina)++;

}

//Info

void Info() {

cout << "Artikal: ";

artikal.Info();

cout << "Količina: " << \*kolicina << endl;

}

double GetCijena() {//vraća cijenu stavke racuna (kolicina \* cijena artikla)

return (\*kolicina) \* artikal.GetCijena();

}

int GetKolicina() { //vraća količinu stavke računa

return \*kolicina;

}

};

int pom = 1000;

class Racun {

int brojRacuna;

StavkaRacuna stavke[100];

int brojStavki;

double iznosBezPDV;

double iznosSaPDV;

bool naplaceno;

public:

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

Racun() {

brojRacuna = pom++;

brojStavki = 0;

iznosBezPDV = 0;

iznosSaPDV = 0;

naplaceno = false;

}

Racun(Racun &original) {

brojRacuna = original.brojRacuna;

brojStavki = original.brojStavki;

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

{

stavke[i] = original.stavke[i];//da nemamo preklopljen operator = za stavke ovo ne bi radilo

}

iznosSaPDV = original.iznosSaPDV;

iznosBezPDV = original.iznosBezPDV;

naplaceno = original.naplaceno;

}

Racun operator = (Racun &original) {

brojRacuna = original.brojRacuna;

brojStavki = original.brojStavki;

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

{

stavke[i] = original.stavke[i];//da nemamo preklopljen operator = za stavke ovo ne bi radilo

}

iznosSaPDV = original.iznosSaPDV;

iznosBezPDV = original.iznosBezPDV;

naplaceno = original.naplaceno;

return \*this;

}

//operator dodjele =

//DodajStavku - dodaje nove artikle u stavke racuna (ulazni parametri su artikal i kolicina), kolicina nije obavezni parametar

//Voditi računa o tome da li je račun već naplaćen

//Ukoliko nema dovoljno artikala na stanju, ispisati odgovarajuću poruku

//Ukoliko artikal već postoji, potrebno je samo uvećati količinu za odgovarajuću stavku

//Prilikom svakog dodavanja artikla, azurirati vrijednosti iznosBezPDV i iznosSaPDV

bool DodajStavku(Artikal &a, int kolicina) {

if (naplaceno) {

cout << "Racun je vec naplacen, nemoguce dodavati artikle!\n";

return false;

}

if (a.GetKolicinaNaSkladistu() < kolicina) {

cout << "Nema dovoljno artikla na stanju. Nemoguće dodati artikal.\n";

return false;

}

StavkaRacuna novaStavka(kolicina,a);

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

{

if (stavke[i].AreEqual(a.GetSifra())) {//ako ima taj artikal već, samo uvećaj količinu

stavke[i].UvecajKolicinuStavke();

iznosBezPDV += a.GetCijena()\*kolicina;

iznosSaPDV = iznosBezPDV + iznosBezPDV \* 0.17;

//azuriraj stanje na skladistu

stavke[i].UmanjiStanjeNaSkladistu(kolicina);

return true;

}

}

stavke[brojStavki] = novaStavka;

//azuriraj stanje na skladistu

stavke[brojStavki].UmanjiStanjeNaSkladistu(kolicina);

brojStavki++;

iznosBezPDV += a.GetCijena()\*kolicina;

iznosSaPDV = iznosBezPDV + iznosBezPDV \* 0.17;

return true;

}

//UkloniArtikal uklanja artikal iz stavki na osnovu njegove sifre

//azurirati vrijednosti iznosBezPDV i iznosSaPDV

bool UkloniArtikal(char \*sifra) {

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

{

if (stavke[i].AreEqual(sifra)) {

//atzuriranje stanja na skladistu

stavke[i].UvecajStanjeNaSkladistu(stavke[i].GetKolicina());

iznosBezPDV -= stavke[i].GetCijena();

iznosSaPDV = iznosBezPDV + iznosBezPDV \* 0.17;

//uklanjanje stavke iz liste

for (int j = i; j < brojStavki-1; j++)

{

stavke[j] = stavke[j+1];

}

brojStavki--;

return true;

}

}

cout << "Artikal nije pronađen!\n";

return false;

}

double PrintRacun() {

//Ispisuje sve stavke računa zajedno sa iznosom za plaćanje

//Omogućiti da se unese iznos koji je korisnik uplatio, te se

//vraća iznos kusura

naplaceno = true;

cout << "======================== Racun br. " << brojRacuna << " ======================== " << endl;

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

{

stavke[i].Info();

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

}

cout << "\nIznos sa PDV-om: " << iznosSaPDV << " KM\n";

cout << "\nIznos bez PDV-a: " << iznosBezPDV << " KM\n\n";

double uplata;

do {

cout << "Unesite iznos koji uplaćujete: ";

cin >> uplata;

} while (uplata < iznosSaPDV);

cin.ignore();

return uplata - iznosSaPDV;

}

};

void main() {

//main je moguće mijenjati i prilagođavati

//ali je ovo napisano sasvim dovoljno za testiranje funkcionalnosti zadatka

Racun racun;

Artikal artikli[3];

for (int i = 0; i < 3; i++)

{

artikli[i].Unos();

racun.DodajStavku(artikli[i], 2);

}

//pokusaj dodavanja istog artikla na racun

racun.DodajStavku(artikli[0], 2);

char sifra[20];

cout << "Unesite sifru artikla kojeg želite ukloniti: ";

cin.getline(sifra, 20);

racun.UkloniArtikal(sifra);

racun.DodajStavku(artikli[1], 1);

cout << "Kusur: " << racun.PrintRacun() << " KM" << endl;

system("pause>0");

}