Pismeni deo ispita iz ORS, jun 2005.

Ideja zadatka: Napisati program koji predstavlja "pojednostavljenu" prodavnicu na Internetu, koja korisniku prikazuje podatke o artiklima i omogućava mu da ih naruči. Kompletna komunikacija sa udaljenom prodavnicom obuhvaćena je klasom *UdaljenaProdavnica*. Ona je već implementirana, a nama je na raspolaganju zaglavlje *UdaljenaProdavnica.h*:

```
typedef map<string,string> PodaciOArtiklu;

class UdaljenaProdavnica {
public:
     UdaljenaProdavnica();
     bool PročitajArtikal( const string& katBr, PodaciOArtiklu & x ) const;
     void PronađiArtikle( const string& s, vector<string>& katBrojevi ) const;
     bool NaručiArtikle( const vector<string>& katBrojevi );
};
```

Metod *PročitajArtikal* čita podatke o artiklu sa datim kataloškim brojem i u upisuje ih u *x*. Podaci o artiklu se dobijaju u obliku kolekcije parova (naziv atributa, vrednost atributa). Sve vrednosti atributa su zapisane tekstualno pa se po potrebi moraju prevesti u cele ili realne brojeve. Svaki artikal je opisan bar kataloškim brojem (atribut sa nazivom *katbr*), nazivom (*naziv*), cenom (*cena*), tekstualnim opisom (*opis*) i tipom (*_tip_*). Metod izračunava *false* akko čitanje nije uspelo.

Metod *PronađiArtikle* traži kataloške brojeve artikala koji (po naslovu, klasifikaciji i sl.) odgovaraju traženom uslovu.

Metod *NaručiArtikle* obezbeđuje naručivanje artikala sa datim kataloškim brojevima. Ako se naručuje više primeraka istog artikla, njegov kataloški broj se ponavlja odgovarajući broj puta u datom nizu. Smatra se da konstruktor klase *UdaljenaProdavnica* obuhvata metode za identifikaciju korisnika, pa pri naručivanju nećemo voditi računa o upisivanju adrese i sl. Takođe, podrazumeva se isporuka pouzećem, pa se nećemo starati ni o plaćanju. Metod izračunava *false* akko naručivanje nije uspelo.

1. deo (10 poena): Napisati pomoćne funkcije za čitanje zapisa celog i realnog broja iz niske. Smatrati da se u zapisima brojeva ne koriste znak i eksponent:

```
int ceoBroj( const string& s )            // izračunava zapisan ceo broj ili pravi izuzetak
double realanBroj( const string& s )      // izračunava zapisan realan broj ili pravi izuzetak
```

2. deo (10 poena): Napisati klasu *Artikal* koja predstavlja jedan artikal iz ponude prodavnice. Napisati naredne metode i sve ostalo što je neophodno za ispravno ponašanje klase:

```
Artikal( const PodaciOArtiklu& p ) // konstruktor, inicijalizuje članove objekta na osnovu podataka iz p
string KatBroj() const             // izračunava kataloški broj
string Naziv() const               // izračunava naziv artikla
double Cena() const                // izračunava cenu artikla
string Opis() const                // izračunava tekstualni opis artikla
string Tip() const                 // izračunava tip artikla ("knjiga", "CD", "DVD",...)
```

3. deo (5 poena): Napisati klasu *Knjiga* (artikal tipa "knjiga"). Osim uobičajenih, sadrži i atribute: *autor*, *format* (niska), *brojStrana* (ceo broj).

4. deo (5 poena): Napisati klasu *AudioCD* (artikal tipa "CD"). Osim uobičajenih, sadrži i atribute: *izvođač* (niska), *trajanje* (u minutima, realan broj).

5. deo (5 poena): Napisati klasu *Računar* (artikal tipa "računar"). Osim uobičajenih, sadrži i atribute: *procesor* (niska), *memorija* (u MB, ceo broj), *disk* (u GB, ceo broj).

6. deo (15 poena): Dodati svim klasama artikala metode koji opisuju sve raspoložive podatke o artiklu u datom toku:

```
void Prikaži( ostream& tok ) const  // piše SVE podatke o artiklu (uključujući i opis tipa i kat.broj) u tok
```

7. deo (15 poena): Napisati klasu *Prodavnica*. Napisati metode:

```
Artikal* Pročitaj( const string& katBr ) const   // čita artikal sa datim kat.brojem iz udaljene prodavnice
                                                  // i pravi odgovarajući objekat (ako ne postoji, vraća 0)
void Pronadji( const string& s, ostream& ostr ) const
          // traži artikle po datom uslovu i ispisuje (1) koliko je artikala pronađeno i (2) za svaki
          // pronađeni artikal ispisuje sve raspoložive podatke
```

8. deo (20 poena): U klasi *Prodavnica* napisati metode:

```
void DodajArtikalNaSpisak( const string& katBr, int kol )
                           // dodaje artikal na spisak za naručivanje u datoj količini
void IspisiSpisak( ostream& tok ) const
                           // ispisuje (1) broj artikala na spisku, (2) tip, kat.broj, naziv, količinu,
                           // pojedinačnu i ukupnu cenu za svaki artikal, (3) ukupnu vrednost
bool Naruči()      // naručuje sve artikle koji su na spisku, vraća true akko je naručivanje uspelo
```

9. deo (15 poena): Napisati program *Prodavnica* koji prima (sa standardnog ulaza) i izvršava sledeće naredbe:

trazi <niska do kraja reda>	traži artikle koji odgovaraju traženom uslovu i ispisuje sve podatke o njima
dodaj <kat.br.>	dodaje jedan artikal na spisak i ispisuje kompletan spisak i zbrinu cenu; obavezno proveravati da li postoji artikal sa datim kataloškim brojem i ispisivati odgovarajuću poruku ako ne postoji
naruči	naručuje artikle sa spiska i izveštava da li je naručivanje uspešno izvedeno; ako jeste, prazni spisak za naručivanje
kraj	prekida rad programa

Program mora da uhvati sve izuzetke koje izbacuje.

Napomene:

- **U svim klasama svi članovi podaci moraju biti privatni.**
- **<u>Ne implementirati</u> metode klase *UdaljenaProdavnica* !!!**
- **Zadaci se rešavaju 4 sata. Za pozitivnu ocenu je neophodno 50 poena. Delovi zadatka se ne moraju rešavati redom. Pisati čitko i sa potrebnim komentarima. U svim delovima zadatka je dopušteno (i preporučljivo) upotrebljavati standardnu biblioteku.**

Rešenje

```
#include <iostream>
#include <fstream>
#include <exceptions>
using namespace std;

#include "UdaljenaProdavnica.h"

//-----------------------------------------------------------------------------
//    1. deo
//-----------------------------------------------------------------------------
int ceoBroj( const string& s )
{
    int n=0;
    const char* c = s.c_str();
    if( c ){
        while( *c==' ' )
            c++;
        while( *c>='0' && *c<='9' ){
            n = n*10 + *c - '0';
            c++;
            }
        if( !*c )
            return n;
        }
    throw runtime_error( "Neispravan zapis celog broja!" );
}

double realanBroj( const string& s )
{
    double d=0;
    const char* c = s.c_str();
    if( c ){
        while( *c==' ' )
            c++;
        while( *c>='0' && *c<='9' ){
            d = d*10 + *c - '0';
            c++;
            }
        if( *c == '.' ){
            c++;
            double f = 1;
            while( *c>='0' && *c<='9' )
                d += (f/=10) * (*(c++) - '0');
            }
        if( !*c )
            return d;
        }
    throw runtime_error( "Neispravan zapis realnog broja!" );
}

//-----------------------------------------------------------------------------
//    2. deo
//-----------------------------------------------------------------------------
class Artikal
{
public:
    Artikal( const PodaciOArtiklu& p )
        {
        _KatBroj = atribut( p, "katbr" );
        _Naziv = atribut( p, "naziv" );
        _Cena = realanBroj( atribut( p, "cena" ));
        _Opis = atribut( p, "opis" );
        }
    virtual ~Artikal()
        {}
    virtual string Tip() const = 0;

    string KatBroj() const
        { return _KatBroj; }
    string Naziv() const
        { return _Naziv; }
    double Cena() const
        { return _Cena; }
    string Opis() const
        { return _Opis; }

    // Metod Prikazi je resenje 6. dela zadatka.
    virtual void Prikazi( ostream& tok ) const = 0;

protected:
    string atribut( const PodaciOArtiklu& p, const string& s )
        {
        PodaciOArtiklu::const_iterator i = p.find(s);
        if( i == p.end() )
            throw runtime_error( "Podaci o artiklu ne sadrze neophodne atribute!");
        return i->second;
        }
```

```
private:
    string _KatBroj;
    string _Naziv;
    double _Cena;
    string _Opis;
    string _Tip;
};

//---------------------------------------------------------------------------
//    3. deo
//---------------------------------------------------------------------------
class Knjiga : public Artikal
{
public:
    Knjiga( const PodaciOArtiklu& p )
        : Artikal( p )
        {
        _Autor = atribut( p, "autor" );
        _Format = atribut( p, "format" );
        _BrojStrana = ceoBroj( atribut( p, "brojStrana" ));
        }

    string Tip() const
        { return "Knjiga"; }

    // Metod Prikazi je resenje 6. dela zadatka.
    void Prikazi( ostream& tok ) const
        {
        tok    << Tip() << " " << KatBroj() << endl
            << Naziv() << endl
            << "Autor: " << _Autor << endl
            << Opis()  << endl
            << "Format: " << _Format
            << "    Broj strana: " << _BrojStrana << endl
            << "Cena: " << Cena() << " dinara" << endl;
        }

private:
    string _Autor;
    string _Format;
    int _BrojStrana;
};

//---------------------------------------------------------------------------
//    4. deo
//---------------------------------------------------------------------------
class AudioCD : public Artikal
{
public:
    AudioCD( const PodaciOArtiklu& p )
        : Artikal( p )
        {
        _Izvodjac = atribut( p, "izvodjac" );
        _Trajanje = realanBroj( atribut( p, "trajanje" ));
        }

    string Tip() const
        { return "Audio CD"; }

    // Metod Prikazi je resenje 6. dela zadatka.
    void Prikazi( ostream& tok ) const
        {
        tok    << Tip() << " " << KatBroj() << endl
            << Naziv() << endl
            << "Izvodjac: " << _Izvodjac << endl
            << Opis()  << endl
            << "Trajanje: " << _Trajanje << " minuta" << endl
            << "Cena: " << Cena() << " dinara" << endl;
        }

private:
    string _Izvodjac;
    double _Trajanje;
};

//---------------------------------------------------------------------------
//    5. deo
//---------------------------------------------------------------------------
class Racunar : public Artikal
{
public:
    Racunar( const PodaciOArtiklu& p )
        : Artikal( p )
        {
        _Procesor = atribut( p, "procesor" );
        _Memorija = ceoBroj( atribut( p, "memorija" ));
        _Disk = ceoBroj( atribut( p, "disk" ));
        }
```

```
    string Tip() const
        { return "Racunar"; }

    // Metod Prikazi je resenje 6. dela zadatka.
    void Prikazi( ostream& tok ) const
        {
        tok    << Tip() << " " << KatBroj() << endl
            << Naziv() << endl
            << "Procesor: " << _Procesor
            << "     Memorija: " << _Memorija
            << "     Disk: " << _Disk << endl
            << Opis()  << endl
            << "Cena: " << Cena() << " dinara" << endl;
        }

private:
    string _Procesor;
    int _Memorija;
    int _Disk;
};


//---------------------------------------------------------------------------
//    7. deo
//---------------------------------------------------------------------------
class Prodavnica
{
public:
    Artikal* Procitaj( const string& katBr ) const
        {
        PodaciOArtiklu x;
        Artikal* a = 0;
        if( udaljenaProdavnica.ProcitajArtikal( katBr, x )){
            string tip = x["_tip_"];
            if( tip == "knjiga" )
                a = new Knjiga(x);
            else if( tip == "CD" )
                a = new AudioCD(x);
            else if( tip == "racunar" )
                a = new Racunar(x);
            }
        return a;
        }

    void Pronadji( const string& s, ostream& ostr ) const
        {
        vector<string> katBrojevi;
        udaljenaProdavnica.PronadjiArtikle( s, katBrojevi );
        if( katBrojevi.size() > 0 ){
            ostr << "Pronadjeno je " << katBrojevi.size()
                 << " artikala." << endl << endl;
            for( unsigned i=0; i<katBrojevi.size(); i++ ){
                Artikal* a = Procitaj( katBrojevi[i] );
                if( a ){
                    a->Prikazi( ostr );
                    ostr << endl;
                    delete a;
                    }
                else
                    throw runtime_error("Pronadjen nepostojeci artikal!");
                }
            }
        else
            ostr << "Nije pronadjen nijedan artikal." << endl;
        }

    //---------------------------------------------------------------------------
    //    8. deo
    //---------------------------------------------------------------------------
    bool DodajArtikalNaSpisak( const string& katBr, int kol=1 )
        {
        Artikal* a = Procitaj( katBr );
        if( a ){
            spisak[katBr] += kol;
            delete a;
            return true;
            }
        else
            return false;
        }
```

```
    bool Naruci()
        {
        vector<string> katBrojevi;
        map<string,int>::const_iterator
            i = spisak.begin(),
            e = spisak.end();
        for( ; i!=e; i++ ){
            for( int j=0; j<i->second; j++ )
                katBrojevi.push_back( i->first );
            }
        if( udaljenaProdavnica.NaruciArtikle( katBrojevi )){
            spisak.clear();
            return true;
            }
        return false;
        }

    void IspisiSpisak( ostream& ostr ) const
        {
        ostr << "Naruceno je " << spisak.size()
             << " artikala." << endl;
        double cena = 0;
        map<string,int>::const_iterator
            i = spisak.begin(),
            e = spisak.end();
        for( ; i!=e; i++ ){
            Artikal* a = Procitaj( i->first );
            if( !a )
                throw runtime_error("Narucen je nepostojeci artikal!");
            ostr << " - " << a->Tip() << " " << a->KatBroj()
                 << " " << a->Naziv() << endl
                 << "     " << i->second << " x " << a->Cena()
                 << " = " << i->second * a->Cena() << " dinara" << endl;
            cena += i->second * a->Cena();
            delete a;
            }
        ostr << "Ukupno: " << cena << " dinara." << endl;
        }

private:
    UdaljenaProdavnica udaljenaProdavnica;
    map<string,int> spisak;
};

//-----------------------------------------------------------------------------
//    8.deo
//-----------------------------------------------------------------------------
main( int argc, char** argv )
{
    try {
        Prodavnica p;
        while(1){
            cout << endl << "Upisi komandu: ";
            string komanda;
            getline( cin, komanda );
            if( komanda.substr(0,6) == "trazi " )
                p.Pronadji( komanda.substr(6,komanda.size()-6) , cout );
            else if( komanda.substr(0,6) == "dodaj " )
                if( p.DodajArtikalNaSpisak( komanda.substr(6,komanda.size()-6) ))
                    p.IspisiSpisak( cout );
                else
                    cout << "Ne postoji takav artikal." << endl;
            else if( komanda == "naruci" )
                cout << ( p.Naruci()
                            ? "Artikli su naruceni. Cekaj na vratima."
                            : "Narucivanje nije uspelo." ) << endl;
            else if( komanda == "kraj" )
                break;
            else{
                cout
                    << "Nepoznata komanda!" << endl
                    << "Moguce komande su:" << endl
                    << "   trazi <niska do kraja reda>" << endl
                    << "   dodaj <kat.br.>" << endl
                    << "   naruci" << endl
                    << "   kraj" << endl;
                }
            }
        }
    catch( exception& e ){
        cerr << "*** " << e.what() << " ***" << endl;
        }
    catch(...){
        cerr << "Nepoznat izuzetak!" << endl;
        }
}
```