Osnovi računarskih sistema

Milena Vujošević–Janičić

Čas 1

Vežbe — 24 05 2004

1.1 Testovi

Napisati program za podršku izvođenja testova uz pomoć računara. Potrebno je podržati više različitih tipova pitanja:

- tekstualna pitanja — pitanja bez ponuđenih odgovora;
- abcd pitalice — pitanja sa ponuđenim odgovorima;
- abcd redosled — pitanja sa ponuđenim odgovorima koje treba sortirati na odgovarajući način.

Svako pitanje mora imati metod za postavljanje pitanja na standardnom izlazu i prihvatanje odgovora sa standardnog ulaza i metod za davanje izveštaja o odgovoru na pitanje i broju poena koji nosi dati odgovor.

Primer 1

- *Napisati klasu* `Pitalica` *koja predstavlja baznu klasu za sve konkretne vrste pitanja.*
- *Napisati klasu* `TekstualnaPitalica` *za postavljanje pitanja bez ponuđenih odgovora. Izostavljen odgovor se vrednuje sa -1 a naveden odgovor sa 0 poena.*
- *Napisati klasu* `Test` *koja predstavlja kolekciju pitanja. Klasa* `Test` *mora da ima metod za postavljanje svih pitanja u testu kao i metod za štampanje izveštaja o rezultatima čitavog testa.*

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

// Klasa Pitalica predstavlja baznu klasu
// za sve konkretne vrste pitanja
class Pitalica
```

```
{
public:
    virtual ~Pitalica()
        {}
    virtual void Postavi() = 0;
    virtual int Izvestaj( ostream& ) const = 0;

    // Metod Kopija() treba pisati svaki put kad treba obezbediti mogucnost
    // kopiranja objekata neke klase koja kao podatak clan ima pokazivac
    // (ili pokazivace) na objekte drugih klasa. U ovom slucaju to je podatak
    // vector<Pitalica*> _Pitanja u okviru klase Test cije je kopiranje
    // potrebno obezbediti, tako da je neophodno u svakoj od tih klasa
    // (u ovom slucaju u svim klasama koje pripadaju hijararhiji sa baznom
    // klasom Pitalica) definisati metod Kopija() kako se pri kopiranju
    // pokazivaca ne bi desilo da i originalni i iskopirani podaci pokazuju
    // na isti objekat u memoriji.
    virtual Pitalica* Kopija() const = 0;
};

// Tekstualne pitalice odnose se na pitanja bez ponudjenih odgovora.
// Izostavljen odgovor se vrednuje sa -1 a naveden odgovor se vrednuje sa
// 0 poena (neko naknadno mora da pregleda ove odgovore).
// Tekstualna pitalica pamti tekst pitalice i
// odgovor koji je za to pitanje dat.
class TekstualnaPitalica : public Pitalica
{
public:
    TekstualnaPitalica( const string& s )
        : _TekstPitanja(s)
        {}

    // Nakon postavljanja pitanja ucitava se odgovor.
    void Postavi()
        {
        cout << _TekstPitanja << endl
             << "----------------------------------------" << endl
             << "Upisite odgovor u jednom redu ili znak '@' ako ne znate:"
             << endl;
        cin >> ws;
        getline( cin, _Odgovor );
        }

    // Stampanje izvestaja postuje pravila ocenjivanja
    // za ovu vrstu pitanja.
```

```
    int Izvestaj( ostream& ostr ) const
        {
        if( _Odgovor == "@" ){
            ostr << "(-1) Nije odgovoreno.";
            return -1;
            }
        else{
            ostr << "( ? ) Odgovoreno je: " << _Odgovor;
            return 0;
            }
        }

    TekstualnaPitalica* Kopija() const
        { return new TekstualnaPitalica(*this); }

private:
    string _TekstPitanja;
    string _Odgovor;
};

// Klasa Test omogucava postavljanje pitanja i stampanje izvestaja.
// Klasa Test cuva niz pokazivaca na pitalice.
class Test
{
public:
    // U konstruktoru ubacujemo dva tekstualna pitanja u test
    // (da bi smo mogli da testiramo program, ovo je neophodno
    // pre nego sto se obezbedi ucitavanje pitalica iz ulaznog toka)
    Test()
        {
        _Pitanja.push_back( new TekstualnaPitalica(
            "Koliko stonoga ima nogu?"
            ));
        _Pitanja.push_back( new TekstualnaPitalica(
            "Sta je to stolica?"
            ));
        }

    // Konstruktor kopije
    Test( const Test& t )
        { init(t); }

    // Destruktor
    ~Test()
```

```
        { deinit(); }

    // Operator dodele
    Test& operator = ( const Test& t )
        {
        if( this != &t ){
            deinit();
            init(t);
            }
        return *this;
        }

    // Metod koji postavlja pitanja
    void Postavi()
        {
        for( unsigned i=0; i<_Pitanja.size(); i++ ){
            cout << endl
                 << "*** Pitanje br. " << (i+1) << " ***"
                 << endl << endl;
            _Pitanja[i]->Postavi();
            cout << endl;
            }
        }

    // Metod koji daje izvestaj o rezultatima testiranja
    void Izvestaj( ostream& ostr ) const
        {
        int suma = 0;
        ostr << "Rezultat testiranja:" << endl
             << "---------------------------------------" << endl;
        for( unsigned i=0; i<_Pitanja.size(); i++ ){
            ostr << (i+1) << ".  ";
            suma += _Pitanja[i]->Izvestaj( ostr );
            ostr << endl;
            }
        ostr << "---------------------------------------" << endl
             << "Rezultat: " << suma << " poena" << endl;
        }

private:
    // Oslobadja se memorija koju su zauzimele pitalice u pitanjima
    void deinit()
        {
        for( unsigned i=0; i<_Pitanja.size(); i++ )
```

```
            delete _Pitanja[i];
        _Pitanja.clear();
        }

    // Inicijalizuju se pitanja na osnovu testa t
    void init( const Test& t )
        {
        for( unsigned i=0; i<t._Pitanja.size(); i++ )
            _Pitanja.push_back( t._Pitanja[i]->Kopija() );
        }

    // Kad god imamo kolekciju razlicitih objekata koji pripadaju
    // istoj hijararhiji i kad je potrebno izvrsavati iste operacije
    // nad svim objektima, onda treba cuvati pokazivace na te objekte, a
    // te metode treba definisati kao virtuelne odnosno treba obezbediti
    // dinamicko a ne staticko vezivanje.
    vector<Pitalica*> _Pitanja;
};

main()
{
    Test t;
    t.Postavi();
    cout << endl << endl << endl;
    t.Izvestaj(cout);

    return 0;
}
```

Primer 2 *Dopunimo klasu* `Pitalica` *iz prethodnog primera metodom za učitavanje pitalice a klasu* `Test` *metodom za učitavanje testa iz ulaznog toka. U ulaznom toku podaci o testu se navode tako što se na početku navodi broj pitalica a zatim na početku teksta svake od pitalica stoji ime vrste pitalice. Pretpostaviti da se ulazna pitanja nalaze u datoteci* `pitanja.txt`.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

// Klasi Pitalica dodajemo jos metode koje nam omogucavaju
// ucitavanje pitalica iz ulaznog toka
class Pitalica
{
```

```
public:
    virtual ~Pitalica()
        {}
    virtual void Ucitaj( istream& istr ) = 0;
    virtual void Postavi() = 0;
    virtual int Izvestaj( ostream& ) const = 0;
    virtual Pitalica* Kopija() const = 0;

    // Staticki metod omogucava formiranje pokazivaca
    // na odgovarajuci objekat u zavisnosti od vrste
    // pitanja iz ulaznog toka
    static Pitalica* UcitajPitanje( istream& istr );
};

class TekstualnaPitalica : public Pitalica
{
public:
    // Iz ulaznog toka izdvaja se tekst pitanja
    void Ucitaj( istream& istr )
        {
        getline( istr, _TekstPitanja );
        if(!istr)
            throw "Greska pri citanju teksta pitanja!";
        }

    void Postavi()
        {...}

    int Izvestaj( ostream& ostr ) const
        {...}

    TekstualnaPitalica* Kopija() const
        { return new TekstualnaPitalica(*this); }

private:
    string _TekstPitanja;
    string _Odgovor;
};

// Za sada imamo samo jedan tip pitalice, kasnije ce
// ovaj metod biti dopunjen mogucnoscu kreiranja
// pokazivaca na ostale tipove pitanja.
// U datoteci se pre teksta pitanja uvek navodi
// ime vrste pitalice.
```

```
Pitalica* Pitalica::UcitajPitanje( istream& istr )
{
    string tip;
    istr >> tip >> ws;
    if(!istr)
        throw "Greska pri citanju tipa pitanja!";

    Pitalica* p = 0;
    if( !strcmp(tip.c_str(),"TekstualnoPitanje"))
        p = new TekstualnaPitalica();
    else
        throw "Nepoznat tip pitanja!";
    p->Ucitaj( istr );
    return p;
}

// Klasa test sada ima mogucnost formiranja testa na
// osnovu podataka iz ulaznog toka koji postuje odgovarajucu
// formu.
class Test
{
public:
    Test()
        {}

    Test( const Test& t )
        { init(t); }

    ~Test()
        { deinit(); }

    Test& operator = ( const Test& t )
        {...}

    // Na pocetku ulazne datoteke nalazi se broj koji
    // oznacava broj pitanja testa.
    void Ucitaj( istream& istr )
        {
        deinit();
        int n;
        istr >> n >> ws;
        if( !istr )
            throw "Greska pri citanju testa!";
        if( n<=0 )
```

```
            throw "Test je prazan!";
        for( int i=0; i<n; i++ ){
            Pitalica* p = Pitalica::UcitajPitanje( istr );
            _Pitanja.push_back( p );
            }
        }

    void Postavi()
        {...}


    void Izvestaj( ostream& ostr ) const
        {...}

private:
    void deinit()
        {...}

    void init( const Test& t )
        {...}

    vector<Pitalica*> _Pitanja;
};

main()
{
    try {
        Test t;
        ifstream f( "pitanja.txt" );
        if( !f )
            throw "Nije uspelo otvaranje datoteke sa testom!";
        t.Ucitaj( f );
        t.Postavi();
        cout << endl << endl << endl;
        t.Izvestaj(cout);
        }
    catch( const char* s ){
        cerr << "GRESKA: " << s << endl;
        }
    return 0;
}
```

Primer 3 *Dopuniti prethodni primer dodavanjem nove vrste pitalica:* `AbcdPitalica`. *Klasa* `AbcdPitalica` *je opisana brojem ponuđenih odgovora, tačnim odgovorom i listom ponuđenih odgovora. Tekst pitanja i tekstovi ponuđenih odgovora se navode u*

po jednom redu. Prilikom sastavljanja izveštaja, tačan odgovor se boduje 5, netačan -3 a izostavljanje odgovora -1 poen.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

class Pitalica
{
...
};

class TekstualnaPitalica : public Pitalica
{
...
};

// AbcdPitalica pamti tekst pitanja, tekstove ponudjenih
// odgovora na dato pitanje, podatak o tacnom odgovoru i
// podatak o odgovoru koji je dat na ovo pitanje
class AbcdPitalica : public Pitalica
{
public:
    // Na pocetku abcd pitalice nalazi se tekst pitanja a zatim u
    // sledecem redu podaci o broju ponudjenih odgovora i o
    // tacnom odgovoru. Na kraju sledi niz ponudjenih odgovora,
    // pritom se svaki odgovor zadaje u jednom redu.
    void Ucitaj( istream& istr )
        {
        getline( istr, _TekstPitanja );
        if(!istr)
            throw "Greska pri citanju teksta pitanja!";
        int n;
        istr >> n >> ws >> _TacanOdgovor >> ws;
        for( int i=0; i<n; i++ ){
            string s;
            getline( istr, s );
            _PonudjeniOdgovori.push_back( s );
            }
        if(!istr)
            throw "Greska pri citanju odgovora na pitanje!";
        }
```

```
    // Postavljanje pitanja sastoji se pored stampanja teksta
    // pitanja i od stampanja ponudjenih odgovora. Zatim se vrsi
    // ucitavanje odgovora
    void Postavi()
        {
        cout << _TekstPitanja << endl
             << "----------------------------------------" << endl;
        for( int i=0; i<_PonudjeniOdgovori.size(); i++ )
            cout << _PonudjeniOdgovori[i] << endl;
        cout << "----------------------------------------" << endl
             << "Upisite slovo koje stoji ispred tacnog odgovora" << endl
             << "ili znak '@' ako ne znate."
             << endl;
        cin >> _Odgovor;
        }

    // Stampanje izvestaja postuje pravila ocenjivanja
    // za ovu vrstu pitanja.
    int Izvestaj( ostream& ostr ) const
        {
        if( _Odgovor == _TacanOdgovor ){
            ostr << "( 5 ) Tacan odgovor.";
            return 5;
            }
        else if( _Odgovor == "@" ){
            ostr << "(-1) Nije odgovoreno.";
            return -1;
            }
        else{
            ostr << "( -3 ) Netacan odgovor.";
            return -3;
            }
        }

    AbcdPitalica* Kopija() const
        { return new AbcdPitalica(*this); }

private:
    string _TekstPitanja;
    vector<string> _PonudjeniOdgovori;
    string _TacanOdgovor;
    string _Odgovor;
};
```

```
// Ucitavanje pitanja se dopunjava mogucnoscu ucitavanja
// AbcdPitalice
Pitalica* Pitalica::UcitajPitanje( istream& istr )
{
    string tip;
    istr >> tip >> ws;
    if(!istr)
        throw "Greska pri citanju tipa pitanja!";

    Pitalica* p = 0;
    if( !strcmp(tip.c_str(),"TekstualnoPitanje"))
        p = new TekstualnaPitalica();
    else if( !strcmp(tip.c_str(),"AbcdPitalica"))
        p = new AbcdPitalica();
    else
        throw "Nepoznat tip pitanja!";
    p->Ucitaj( istr );
    return p;
}

class Test
{
...
};

main()
{
...
}
```

Primer 4 *Dopuniti prethodni primer dodavanjem nove vrste pitalica:* `AbcdRedosled`. *Klasa* `AbcdRedosled` *zahteva da se ponuđeni odgovori poređaju u nekom redosledu. Klasa* `AbcdRedosled` *je opisana brojem ponuđenih odgovora, tačnim redosledom odgovora i listom ponuđenih odgovora. Tekst pitanja i tekstovi ponuđenih odgovora se navode u po jednom redu. Prilikom sastavljanja izveštaja, tačan odgovor se boduje 5, netačan -3 a izostavljanje odgovora -1 poen.*

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

class Pitalica
```

```
{
...
};

class TekstualnaPitalica : public Pitalica
...
};

// U klasi AbcdPitalica modifikujemo metod postavljanja
// pitanja uvodjenjem odgovarajuceg uputstva
// tako da klasa AbcdRedosled moze da nasledi ovu klasu
class AbcdPitalica : public Pitalica
{
public:
    void Ucitaj( istream& istr )
        {
        ...
        }

    void Postavi()
        {
        cout << _TekstPitanja << endl
             << "-----------------------------------------" << endl;
        for( int i=0; i<_PonudjeniOdgovori.size(); i++ )
            cout << _PonudjeniOdgovori[i] << endl;
        cout << "-----------------------------------------" << endl
             << Uputstvo()
             << endl;
        cin >> _Odgovor;
        }

    int Izvestaj( ostream& ostr ) const
        {
        ...
        }

    AbcdPitalica* Kopija() const
        { return new AbcdPitalica(*this); }

protected:
    virtual char* Uputstvo() const
        {
        return
            "Upisite slovo koje stoji ispred tacnog odgovora\n"
```

```
            "ili znak '@' ako ne znate.";
        }

private:
    string _TekstPitanja;
    vector<string> _PonudjeniOdgovori;
    string _TacanOdgovor;
    string _Odgovor;
};

// Klasa AbcdRedosled nasledjuje klasu AbcdPitalica i specifikuje
// svoje uputstvo prilikom postavljanja pitanja.
class AbcdRedosled : public AbcdPitalica
{
public:
    AbcdRedosled* Kopija() const
        { return new AbcdRedosled(*this); }

protected:
    char* Uputstvo() const
        {
        return
            "Navedite tacan redosled odgovora upisivanjem slova\n"
            "koja stoje ispred odgovora (npr: acbd)\n"
            "ili znak '@' ako ne znate.";
        }
};

// Modifikujemo ucitavanje pitanja tako da moze
// da prepozna jos jednu vrstu pitanja, tj AbcdRedosled.
Pitalica* Pitalica::UcitajPitanje( istream& istr )
{
    string tip;
    istr >> tip >> ws;
    if(!istr)
        throw "Greska pri citanju tipa pitanja!";

    Pitalica* p = 0;
    if( !strcmp(tip.c_str(),"TekstualnoPitanje"))
        p = new TekstualnaPitalica();
    else if( !strcmp(tip.c_str(),"AbcdPitalica"))
        p = new AbcdPitalica();
    else if( !strcmp(tip.c_str(),"AbcdRedosled"))
        p = new AbcdRedosled();
```

```
    else
        throw "Nepoznat tip pitanja!";
    p->Ucitaj( istr );
    return p;
}

class Test
{
...
};

main()
{
...
}
```

Sadržaj