Osnovi računarskih sistema

— prateći materijal za vežbe —

Milena Vujošević–Janičić i Jelena Tomašević

Sadržaj

1

Konverzije

Zadatak 1 *Napisati funkciju* `unsigned citanjeZapisa( const string& s )` *koja izracunava ceo broj na osnovu niske* `s` *koja predstavlja zapis broja u osnovi deset.*

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

using namespace std;

const unsigned osnova = 10;

unsigned vrednostCifre( char c )
{
    return c - '0';
}

unsigned citanjeZapisa( const string& s )
{
    unsigned n = 0;
    for( unsigned i=0; i<s.length(); i++ )
        n = n * osnova + vrednostCifre(s[i]);
    return n;
}

main()
{
    string zapis = "327513";
    unsigned broj = citanjeZapisa(zapis);
    cout << zapis << " = " << broj << endl;

    return 0;
}
```

Zadatak 2 *Napisati funkciju* `unsigned citanjeZapisa( const string& s, unsigned osnova )`

koja izracunava ceo broj na osnovu niske `s` *koja predstavlja zapis broja u proizvoljnoj osnovi manjoj od 36.*

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

using namespace std;

// ustanovljavamo vrednost cifre
// za bilo koju osnovu od 2 do 36
// !!! pretpostavljamo da je cifra ispravna
unsigned vrednostCifre( char c )
{
    if( c >= '0' && c <= '9' )
        return c - '0';
    else if( c>='a' && c <='z' )
        return c - 'a' + 10;
    else //if( c>='A' && c <='Z' )
        return c - 'A' + 10;
}

// citamo zapis celog neoznacenog broja
// za datu osnovu od 2 do 36
// !!! pretpostavljamo da je zapis ispravan
unsigned citanjeZapisa( const string& s, unsigned osnova )
{
    unsigned n = 0;
    for( unsigned i=0; i<s.length(); i++ )
        n = n * osnova + vrednostCifre(s[i]);
    return n;
}

main()
{
    string zapis = "327513";
    unsigned broj = citanjeZapisa(zapis,10);
    cout << zapis << " = " << broj << endl;

    broj = citanjeZapisa(zapis,8);
    cout << zapis << " = " << broj << endl;

    broj = citanjeZapisa(zapis,16);
    cout << zapis << " = " << broj << endl;

    return 0;
```

```
}

izlaz:
327513 = 327513
327513 = 110411
327513 = 3306771
```

Zadatak 3 *Napisati funkciju* `string zapisivanjeBroja( unsigned n )` *koja na osnovu broja* `n` *u osnovi deset formira string koji se sastoji od cifara broja* `n`.

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

using namespace std;

// ustanovljavamo vrednost cifre
// za bilo koju osnovu od 2 do 36
// !!! pretpostavljamo da je cifra ispravna
unsigned vrednostCifre( char c )
{
    if( c >= '0' && c <= '9' )
        return c - '0';
    else if( c>='a' && c <='z' )
        return c - 'a' + 10;
    else //if( c>='A' && c <='Z' )
        return c - 'A' + 10;
}

// citamo zapis celog neoznacenog broja
// za datu osnovu od 2 do 36
// !!! pretpostavljamo da je zapis ispravan
unsigned citanjeZapisa( const string& s, unsigned osnova )
{
    unsigned n = 0;
    for( unsigned i=0; i<s.length(); i++ )
        n = n * osnova + vrednostCifre(s[i]);
    return n;
}

char zapisCifre( unsigned n )
{
    return '0' + n;
}

void obrniNisku( string& s )
{
```

```
    unsigned l = s.length();
    for( unsigned i=0; i<l/2; i++ ){
        char t = s[i];
        s[i] = s[l-1-i];
        s[l-1-i] = t;
        }
}

string zapisivanjeBroja( unsigned n )
{
    const unsigned osnova = 10;
    if( n == 0 )
        return "0";
    else{
        string zapis;
        while( n>0 ){
            zapis = zapis + zapisCifre( n%osnova );
            n = n / osnova;
            }
        obrniNisku( zapis );
        return zapis;
        }
}

main()
{
    string zapis = "327513";
    unsigned broj = citanjeZapisa(zapis,10);
    string z2 = zapisivanjeBroja(broj);
    cout << zapis << " = " << broj << " = " << z2 << endl;

    broj = citanjeZapisa(zapis,8);
    cout << zapis << " = " << broj << endl;

    broj = citanjeZapisa(zapis,16);
    cout << zapis << " = " << broj << endl;

    return 0;
}

izlaz iz programa:
327513 = 327513 = 327513
327513 = 110411
327513 = 3306771
```

Zadatak 4 *Napisati funkciju* `string zapisivanjeBroja( unsigned n, unsigned osnova )` *koja na osnovu broja* `n` *u osnovi* `osnova` *formira string koji se sastoji od cifara broja* `n`.

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

using namespace std;

// ustanovljavamo vrednost cifre
// za bilo koju osnovu od 2 do 36
// !!! pretpostavljamo da je cifra ispravna
unsigned vrednostCifre( char c )
{
    if( c >= '0' && c <= '9' )
        return c - '0';
    else if( c>='a' && c <='z' )
        return c - 'a' + 10;
    else //if( c>='A' && c <='Z' )
        return c - 'A' + 10;
}

// citamo zapis celog neoznacenog broja
// za datu osnovu od 2 do 36
// !!! pretpostavljamo da je zapis ispravan
unsigned citanjeZapisa( const string& s, unsigned osnova )
{
    unsigned n = 0;
    for( unsigned i=0; i<s.length(); i++ )
        n = n * osnova + vrednostCifre(s[i]);
    return n;
}

// izracunavamo zapis cifre
// za bilo koju osnovu od 2 do 36
// !!! pretpostavljamo da je vrednost cifra ispravna
char zapisCifre( unsigned n )
{
    if( n<10 )
        return '0' + n;
    else
        return 'a' + n - 10;
    /*
        moze i ovako:
        char* cifre = "0123456789abcdefghijklmnopqrstuvwxyz";
```

```
        return cifre[n];
    */
}

// pomocna funkcija koja izvrce nisku
void obrniNisku( string& s )
{
    unsigned l = s.length();
    for( unsigned i=0; i<l/2; i++ ){
        char t = s[i];
        s[i] = s[l-1-i];
        s[l-1-i] = t;
        }
}

// izracunavamo zapis broja
// za bilo koju osnovu od 2 do 36
string zapisivanjeBroja( unsigned n, unsigned osnova )
{
    if( n == 0 )
        return "0";
    else{
        string zapis;
        while( n>0 ){
            zapis = zapis + zapisCifre( n%osnova );
            n = n / osnova;
            }
        obrniNisku( zapis );
        return zapis;
        }
}

// pomocna funkcija za ilustrovanje provera
void provera( unsigned n, unsigned osnova )
{
    string zapis = zapisivanjeBroja( n, osnova );
    unsigned m = citanjeZapisa( zapis, osnova );
    cout << n << " = (" << zapis << ")" << osnova << " = " << m << endl;
}

main()
{
    for( unsigned i=2; i<21; i++ )
        provera( 327513, i );
```

```
    return 0;
}

izlaz:
327513 = (1001111111101011001)2 = 327513
327513 = (121122021010)3 = 327513
327513 = (1033331121)4 = 327513
327513 = (40440023)5 = 327513
327513 = (11004133)6 = 327513
327513 = (2532564)7 = 327513
327513 = (1177531)8 = 327513
327513 = (548233)9 = 327513
327513 = (327513)10 = 327513
327513 = (20407a)11 = 327513
327513 = (139649)12 = 327513
327513 = (b60c4)13 = 327513
327513 = (874db)14 = 327513
327513 = (67093)15 = 327513
327513 = (4ff59)16 = 327513
327513 = (3fb48)17 = 327513
327513 = (322f3)18 = 327513
327513 = (29e4a)19 = 327513
327513 = (20ifd)20 = 327513
```

1.1 Osnovno o datotekama

Zadatak 5 *Čitanje sadržaja datoteke*

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

// Ukljucujemo biblioteku za rad sa datotekama
#include <fstream>

using namespace std;

void citanjeTekstualneDat( char* naziv )
{
    // prvi nacin da otvorimo datoteku
    // ifstream f;
    // f.open("citanjedatoteke.cpp");

    // drugi nacin
    // Kreira se ulazni tok f i on se vezuje za datoteku
```

```
    // po imenu naziv
    ifstream f( naziv );

    // Operator ! primenjen na objekat toka
    // vraca 1 ukoliko citanje nije uspelo
    // Primetimo da !(!f) nije isto sto i f
    if( !f ){
        cout << "Nije uspelo otvaranje datoteke!" << endl;
        return;
        }

    unsigned duzina = 0;
    while(1){
        char c;

        // U karakter c smesta se jedan bajt iz ulaznog
        // datotecnog toka koristeci metod get
        // koja iz toka izdvaja jedan bajt
        f.get(c);

        if( !f )
            break;
        cout << c;
        duzina ++;
        }
    cout << "!!! Procitano je " << duzina << " znakova." << endl;
}

void citanjeBinarneDat( char* naziv )
{

    // Drugi argument prilikom otvaranja ulazne
    // ili izlazne datoteke moze biti kombinacija
    // (disjunkcija na nivou bitova)
    // nekih od narednih konstanti
    // definisanih u klasi ios:
    // ios::in  otvaranje za citanje
    // ios::out otvaranje za pisanje
    // ios::app pri pisanju se vrsi dopisivanje
    //          na postojeci sadrzaj datoteke
    // ios::trunc ako datoteka postoji brise se
    //          postojeci sadrzaj
    // ios::ate otvaranje bez brisanja sadrzaja
    //          pozicioniranje na kraj datoteke
```

```
// ios::nocreate ako datoteka ne postoji
//          otvaranje ne uspeva
// ios::noreplace ako datoteka postoji
//          otvaranje ne uspeva
// ios::binary otvaranje u binarnom rezimu
//          umesto u znakovnom

ifstream f( naziv, ios::in | ios::binary );
if( !f ){
    cout << "Nije uspelo otvaranje datoteke!" << endl;
    return;
    }

// izracunavanje duzine
// Metod seekg postavlja poziciju ulaznog toka (na kraj). Prvi argument
// je relativna pozicija u toku u odnosu na drugi argument koji moze biti
// pocetak ios::beg, trenutna pozicija ios::cur ili kraj ios::end
f.seekg( 0, ios::end );

// Funkcija tellg vraca trenutnu poziciju u toku
unsigned duzina = f.tellg();

// Vracamo se na pocetak datoteke
f.seekg( 0, ios::beg );

// alociramo prostor
char* sadrzajDatoteke = new char[duzina+1];

// citamo:
// Metod read klase istream ima sledeci potpis
// read( char* adresa, streamsize size)
// ona izdvaja size susednih bajtova iz
// ulaznog toka i smesta ih u memoriju sa
// pocetkom na adresi adresa
f.read( sadrzajDatoteke, duzina );
if( !f ){
    cout << "Nije uspelo citanje datoteke!" << endl;
    return;
    }

// ispisemo sadrzaj
sadrzajDatoteke[duzina] = 0;
cout << sadrzajDatoteke << endl;
```

```
    delete [] sadrzajDatoteke;

    cout << "!!! Procitano je " << duzina << " bajtova." << endl;
}

main()
{
    cout << "-----------------------------------------------" << endl;
    citanjeTekstualneDat( "citanjedatoteke.cpp" );
    cout << "-----------------------------------------------" << endl;
    citanjeBinarneDat( "citanjedatoteke.cpp" );
    return 0;
}
/*
izlaz iz programa:
.......
!!! Procitano je 3318 znakova.
.......
!!! Procitano je 3435 bajtova.
*/
```

Zadatak 6 *Upisivanje sadržaja u datoteku*

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

using namespace std;

char* txt =
    "Ovo je primer teksta koji zelimo zapisati\r\n"
    "u datoteci.\r\n"
    "Ima nekoliko redova\r\n";

void pisanjeTekstualneDat( char* naziv )
{
    ofstream f( naziv );
    if( !f ){
        cout << "Nije uspelo otvaranje binarne datoteke!" << endl;
        return;
        }

    f << txt;
    if( !f )
        cout << "Nije uspelo pisanje tekstualne datoteke!" << endl;
```

```
}

void pisanjeBinarneDat( char* naziv )
{
    ofstream f( naziv, ios::out | ios::binary );
    if( !f ){
        cout << "Nije uspelo otvaranje binarne datoteke!" << endl;
        return;
        }

    // Koristimo metodu write klase ostream
    // koja omogucava da se u izlazni tok
    // stavlja odredjen broj znakova
    // ukljucujuci i zavrsne znake ako se
    // oni nalaze u nizu.
    // Ona prima dva argumenta:
    // write(const char* s, streamsize duzina)
    // pokazivac na nisku znakova i duzinu tj
    // broj znakova za izlazni tok
    f.write( txt, strlen(txt) );
    if( !f )
        cout << "Nije uspelo pisanje binarne datoteke!" << endl;
}

main()
{
    pisanjeTekstualneDat( "primerTxt1.txt" );
    pisanjeBinarneDat( "primerBin1.txt" );
    return 0;
}
```

Zadatak 7

```
/*
    Napisati program koji cita tekstualnu datoteku
    ciji je sadrzaj niz cifara 0 i 1 (ne pojavljuje se
    nijedan drugi znak). Pretpostavka je da u toj ulaznoj
    datoteci ima 8n cifara, tj. da se sastoji od n podnizova
    duzine 8.
    Napraviti binarnu datoteku ciji svaki bajt ima,
    redom, vrednost jednog podniza ulazne datoteke od 8
    binarnih cifara.
    Svaki podniz duzine 8 je potrebno procitati
    kao neoznacen ceo binarni broj i njegovu vrednost
    zapisati (kao vrednost bajta) u binarnu datoteku.
*/
```

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

using namespace std;

main()
{
    ifstream txt("nuleijedinice.txt");
    ofstream bajtovi( "bajtovi.dat", ios::binary );

    char cifra;
    unsigned char bajt = 0;
    unsigned brojac=0;

    while(1){
        txt.get(cifra);
        if(!txt)
            break;

        bajt <<= 1;
        if( cifra=='1' )
            bajt ++;

        if( ++brojac == 8 ){
            bajtovi.write( &bajt, 1 );
            bajt = 0;
            brojac = 0;
            }
        }

/* Drugo resenje:

    ifstream txt("nuleijedinice.txt");
    ofstream bajtovi( "bajtovi.dat", ios::binary );

    while(1){
        unsigned char bajt = 0;
        int i;
        for( i=0; i<8; i++ ){
            char cifra;
            txt.get(cifra);
            if(!txt)
```

```
                break;
            bajt = bajt * 2 + cifra-'0';
            }
        if( i<8 )
            break;

        bajtovi.write( &bajt, 1 );
        }
*/
    return 0;
}
```

Zadatak 8 *Napisati program koji cita binarnu datoteku i za svaki procitan bajt na standardnom izlazu ispisuje po dve cifre koje predstavljaju heksadekadni zapis vrednosti tog bajta.*

```
#include <iostream>
#include <fstream>

using namespace std;

const unsigned duzinaBafera = 1000;

string hexZapis( unsigned char c )
{
    string s="00";
    char* cifre="0123456789abcdef";
    s[0] = cifre[c/16];
    s[1] = cifre[c%16];
    return s;
}

main()
{
    unsigned char bafer[duzinaBafera];
    ifstream f( "zadatak.cpp", ios::binary );
    if(!f){
        cout << "Nije uspelo otvaranje datoteke!" << endl;
        return 1;
        }

    f.seekg( 0, ios::end );
    unsigned duzina = f.tellg();
    f.seekg( 0, ios::beg );

    unsigned procitano = 0;
```

```
    while( procitano < duzina ){
        unsigned citamo = duzina - f.tellg();
        if( citamo > duzinaBafera )
            citamo = duzinaBafera;
        f.read( bafer, citamo );
        if(!f){
            cout << "Nije uspelo citanje datoteke!" << endl;
            return 2;
            }
        procitano += citamo;
        for( unsigned i=0; i<citamo; i++ )
            cout << hexZapis(bafer[i]);
        }

    return 0;
}
```

1.2 Klasa CeoBroj

Zadatak 9 *Klasa* `CeoBroj` *treba da obezbedi rad sa proizvoljno velikim celim brojevima pri čemu osnova celog broja može da bude broj između* 2 *i* 36.

```
#include <iostream>
#include <math.h>
#include <vector>

using namespace std;

class CeoBroj
{
public:
    // Konstruktor na osnovu broja n u osnovi 10
    // formira niz cifara u osnovi "osnova"
    CeoBroj( int n, unsigned osnova=10 )
        : _Osnova( osnova ),
          _Znak( n>=0 ? 1 : -1 )
        {

        // Metod abs definisan je u math.h,
        // vraca apsolutnu vrednost broja
        InicijalizacijaCifara( abs(n) );

        }

    // Niz cifara datog broja n interno
```

```
    // pamtimo u obrnutom redosledu, zato
    // ispis pocinje od poslednje cifre
    // u vektoru. Za svaku cifru pamtimo njenu
    // vrednost, prilikom ispisa u zavisnosti
    // od vrednosti stampamo odgovarajuci znak
    void Ispis( ostream& ostr ) const
        {
        if( _Znak<0 )
            ostr << '-';
        for( int i=_Cifre.size()-1; i>=0; i-- )
            ostr << ZapisCifre( _Cifre[i] );
        }

/*
    CeoBroj operator+ ( const CeoBroj& c ) const
        ...
    CeoBroj operator- ( const CeoBroj& c ) const
        ...
    CeoBroj operator* ( const CeoBroj& c ) const
        ...
    CeoBroj operator/ ( const CeoBroj& c ) const
        ...
    unsigned Osnova() const
        { return _Osnova; }
    int Vrednost() const
        { ... }
*/
    int Znak() const
        { return _Znak; }

private:
    //----------------------------
    // Pomocni metodi

    void InicijalizacijaCifara( unsigned n )
        {
        // Metod clear prazni sadrzaj vektora
        _Cifre.clear();
        while( n>0 ){
            _Cifre.push_back( n % _Osnova );
            n /= _Osnova;
            }

        if( _Cifre.size()==0 )
```

```
            _Cifre.push_back( 0 );
        }

    // ovaj metod moze da bude staticki
    char ZapisCifre( int c ) const
        {
        char* cifre = "0123456789abcdefghijklmnopqrstuwvxyz";
        return cifre[c];
        }

    //----------------------------
    // Clanovi podaci
    unsigned            _Osnova;
    int                 _Znak;
    vector<unsigned>    _Cifre;
};

ostream& operator << ( ostream& ostr, const CeoBroj& c )
{
    c.Ispis( ostr );
    return ostr;
}

main()
{
    cout << CeoBroj( 12345678 ) << endl;
    cout << CeoBroj( -87654321 ) << endl;

    cout << CeoBroj( 12345678, 16 ) << endl;
    cout << CeoBroj( -87654321, 16 ) << endl;

    return 0;
}
/*
Izlaz iz programa:
12345678
-87654321
bc614e
-5397fb1
*/
```

Zadatak 10 *Klasi* `CeoBroj` *dodajemo operatore sabiranja i oduzimanja, kao i pomoćne metode koje su za to neophodne.*

```
#include <iostream>
#include <math.h>
```

```
#include <vector>

using namespace std;

class CeoBroj
{
public:
    CeoBroj()
        {}

    CeoBroj( int n, unsigned osnova=10 )
        : _Osnova( osnova ),
          _Znak( n>=0 ? 1 : -1 )
        { InicijalizacijaCifara( abs(n) ); }

    void Ispis( ostream& ostr ) const
        {
        if( _Znak<0 )
            ostr << '-';
        for( int i=_Cifre.size()-1; i>=0; i-- )
            ostr << ZapisCifre( _Cifre[i] );
        }

    // kod svih operacija pretpostavljamo
    // i ne proveravamo
    // da su svi argumenti u istim osnovama
    CeoBroj operator+ ( const CeoBroj& c ) const
        {
        CeoBroj r;
        // 5+3=5+3, (-5)+(-3)=-(5+3)
        // 3+5=5+3, (-3)+(-5)=-(5+3)
        // (-5)+3=-(5-3), 5+(-3)=5-3
        // 3+(-5)=-(5-3), (-3)+5=5-3
        if( Znak() == c.Znak() ){
            if( AbsVeci( *this, c ))
                AbsSabiranje( *this, c, r );
            else
                AbsSabiranje( c, *this, r );
            r._Znak = Znak();
            }
        else if( AbsVeci( *this, c )){
            AbsOduzimanje( *this, c, r );
            r._Znak = Znak();
            }
```

```
        else {
            AbsOduzimanje( c, *this, r );
            r._Znak = c.Znak();
            }
        return r;
        }

    CeoBroj operator- ( const CeoBroj& c ) const
        {
        CeoBroj r;
        if( Znak() != c.Znak() ){
            if( AbsVeci( *this, c ))
                AbsSabiranje( *this, c, r );
            else
                AbsSabiranje( c, *this, r );
            r._Znak = Znak();
            }
        else if( AbsVeci( *this, c )){
            AbsOduzimanje( *this, c, r );
            r._Znak = Znak();
            }
        else {
            AbsOduzimanje( c, *this, r );
            r._Znak = -Znak();
            }
        return r;
        }

/*
    CeoBroj operator* ( const CeoBroj& c ) const
        ...
    CeoBroj operator/ ( const CeoBroj& c ) const
        ...
    unsigned Osnova() const
        { return _Osnova; }
    int Vrednost() const
        { ... }
*/
    int Znak() const
        { return _Znak; }

private:
    //----------------------------
    // Pomocni metodi
```

```
    void InicijalizacijaCifara( unsigned n )
        {
        _Cifre.clear();
        while( n>0 ){
            _Cifre.push_back( n % _Osnova );
            n /= _Osnova;
            }

        if( _Cifre.size()==0 )
            _Cifre.push_back( 0 );
        }

    // ovaj metod moze da bude staticki (!?!)
    char ZapisCifre( int c ) const
        {
        char* cifre = "0123456789abcdefghijklmnopqrstuwvxyz";
        return cifre[c];
        }

    void AbsSabiranje( const CeoBroj& veci, const CeoBroj& manji, CeoBroj& r ) const
        {
        unsigned osn = r._Osnova = veci._Osnova;
        unsigned prenos = 0;
        unsigned brCifaraManjeg = manji._Cifre.size();
        unsigned brCifaraVeceg = veci._Cifre.size();

        // prvi nacin
        for( unsigned i=0; i<brCifaraVeceg; i++ ){
            unsigned c =
                veci._Cifre[i]
                + ( brCifaraManjeg>i ? manji._Cifre[i] : 0 )
                + prenos;
            if( c>=osn ){
                prenos = 1;
                c -= osn;
                }
            else
                prenos = 0;
            r._Cifre.push_back( c );
            }
        if( prenos > 0 )
            r._Cifre.push_back( prenos );

/*
```

```
        // drugi nacin
        unsigned i;
        for( i=0; i<brCifaraManjeg; i++ )
            unsigned c = veci._Cifre[i] + manji._Cifre[i] + prenos;
            if( c>=osn ){
                prenos = 1;
                c -= osn;
                }
            else
                prenos = 0;
            r._Cifre.push_back( c );
            }
        for( ; i<brCifaraVeceg; i++ )
            unsigned c = veci._Cifre[i] + prenos;
            if( c>=osn ){
                prenos = 1;
                c -= osn;
                }
            else
                prenos = 0;
            r._Cifre.push_back( c );
            }
        if( prenos > 0 )
            r._Cifre.push_back( prenos );
*/
        }

    void AbsOduzimanje( const CeoBroj& veci, const CeoBroj& manji, CeoBroj& r ) const
        {
        unsigned osn = r._Osnova = veci._Osnova;
        int pozajmica = 0;
        unsigned brCifaraManjeg = manji._Cifre.size();
        unsigned brCifaraVeceg = veci._Cifre.size();

        for( unsigned i=0; i<brCifaraVeceg; i++ ){
            int c =
                (int)veci._Cifre[i]
                - (int)( brCifaraManjeg>i ? manji._Cifre[i] : 0 )
                - (int)pozajmica;
            if( c<0 ){
                pozajmica = 1;
                c += osn;
                }
            else
```

```
                pozajmica = 0;
            r._Cifre.push_back( c );
            }

        // brisemo vodece nule, ako ih ima
        while( r._Cifre.size()>1 && r._Cifre.back() == 0 )
            r._Cifre.pop_back();
        }

    // da li je prvi >= drugi
    bool AbsVeci( const CeoBroj& x, const CeoBroj& y ) const
        {
        if( x._Cifre.size() > y._Cifre.size() )
            return true;
        if( x._Cifre.size() < y._Cifre.size() )
            return false;
        for( int i=x._Cifre.size(); i>=0; i-- ){
            if( x._Cifre[i] > y._Cifre[i] )
                return true;
            if( x._Cifre[i] < y._Cifre[i] )
                return false;
            }
        // slucaj x==y
        return true;
        }

    //----------------------------
    // Clanovi podaci
    unsigned            _Osnova;
    int                 _Znak;
    vector<unsigned>    _Cifre;
};

ostream& operator << ( ostream& ostr, const CeoBroj& c )
{
    c.Ispis( ostr );
    return ostr;
}

main()
{
    cout << (CeoBroj(123) + CeoBroj(24)) << endl;
    cout << (CeoBroj(24) + CeoBroj(123)) << endl;
    cout << (CeoBroj(123) + CeoBroj(-24)) << endl;
```

```
    cout << (CeoBroj(-123) + CeoBroj(-24)) << endl;
    cout << (CeoBroj(-123) + CeoBroj(24)) << endl;

    cout << (CeoBroj(123) - CeoBroj(24)) << endl;
    cout << (CeoBroj(123) - CeoBroj(-24)) << endl;
    cout << (CeoBroj(-123) - CeoBroj(-24)) << endl;
    cout << (CeoBroj(-123) - CeoBroj(24)) << endl;

    return 0;
}

/*
147
147
99
-147
-99
99
147
-99
-147
*/
```

Zadatak 11 *Dodajemo operator množenja* `*`*, odgovarajuće metode postaju statičke.*

```
#include <iostream>
#include <math.h>
#include <vector>

using namespace std;

class CeoBroj
{
public:
    CeoBroj()
        {}

    CeoBroj( int n, unsigned osnova=10 )
        : _Osnova( osnova ),
          _Znak( n>=0 ? 1 : -1 )
        { InicijalizacijaCifara( abs(n) ); }

    void Ispis( ostream& ostr ) const
        {
        if( _Znak<0 )
            ostr << '-';
```

```
    for( int i=_Cifre.size()-1; i>=0; i-- )
        ostr << ZapisCifre( _Cifre[i] );
    }

// kod svih operacija pretpostavljamo
// i ne proveravamo
// da su svi argumenti u istim osnovama
CeoBroj operator+ ( const CeoBroj& c ) const
    {
    CeoBroj r;
    if( Znak() == c.Znak() ){
        if( AbsVeci( *this, c ))
            AbsSabiranje( *this, c, r );
        else
            AbsSabiranje( c, *this, r );
        r._Znak = Znak();
        }
    else if( AbsVeci( *this, c )){
        AbsOduzimanje( *this, c, r );
        r._Znak = Znak();
        }
    else {
        AbsOduzimanje( c, *this, r );
        r._Znak = c.Znak();
        }
    return r;
    }

CeoBroj operator- ( const CeoBroj& c ) const
    {
    CeoBroj r;
    if( Znak() != c.Znak() ){
        if( AbsVeci( *this, c ))
            AbsSabiranje( *this, c, r );
        else
            AbsSabiranje( c, *this, r );
        r._Znak = Znak();
        }
    else if( AbsVeci( *this, c )){
        AbsOduzimanje( *this, c, r );
        r._Znak = Znak();
        }
    else {
        AbsOduzimanje( c, *this, r );
```

```
            r._Znak = -Znak();
            }
        return r;
        }

    CeoBroj operator* ( const CeoBroj& c ) const
        {
        CeoBroj r;
        r._Osnova = _Osnova;
        r._Znak = 1;

        for( int i=_Cifre.size()-1; i>=0; i-- ){
            // mnozimo medjurezultat osnovom
            // npr. ako imamo broj 123 on se cuva
            // kao 321. Ako ga pomnozimo sa osnovom,
            // to je 1230, dakle novi broj je 0321,
            // znaci dodajemo nulu na pocetak vektora
            // cifara
            r._Cifre.insert( r._Cifre.begin(), 0 );

            // izracunavamo jedan medjuproizvod
            CeoBroj korak;
            korak._Osnova = _Osnova;
            korak._Znak = 1;
            AbsMnozenjeCifrom( c, _Cifre[i], korak );

            // dodajemo na medjurezultat
            r = r + korak;
            }

        r._Znak = Znak() * c.Znak();
        return r;
        }

/*
    CeoBroj operator/ ( const CeoBroj& c ) const
        ...
    unsigned Osnova() const
        { return _Osnova; }
    int Vrednost() const
        { ... }
*/

    int Znak() const
```

```
        { return _Znak; }

private:
    //----------------------------
    // Pomocni metodi
    void InicijalizacijaCifara( unsigned n )
        {
        _Cifre.clear();
        while( n>0 ){
            _Cifre.push_back( n % _Osnova );
            n /= _Osnova;
            }
        // ovo mozda i nije neophodno
        if( _Cifre.size()==0 )
            _Cifre.push_back( 0 );
        }

    static char ZapisCifre( int c )
        {
        static char* cifre = "0123456789abcdefghijklmnopqrstuwvxyz";
        return cifre[c];
        }

    static void AbsMnozenjeCifrom( const CeoBroj& x, unsigned cifra, CeoBroj& r )
        {
        unsigned prenos = 0;
        for( unsigned j=0; j<x._Cifre.size(); j++ ){
            unsigned a = x._Cifre[j] * cifra + prenos;
            r._Cifre.push_back( a % x._Osnova );
            prenos = a / x._Osnova;
            }
        if(prenos>0)
            r._Cifre.push_back( prenos );
        }

    static void AbsSabiranje( const CeoBroj& veci, const CeoBroj& manji, CeoBroj& r )
        {
        unsigned osn = r._Osnova = veci._Osnova;
        unsigned prenos = 0;
        unsigned brCifaraManjeg = manji._Cifre.size();
        unsigned brCifaraVeceg = veci._Cifre.size();

        for( unsigned i=0; i<brCifaraVeceg; i++ ){
            unsigned c =
```

```
            veci._Cifre[i]
            + ( brCifaraManjeg>i ? manji._Cifre[i] : 0 )
            + prenos;
        if( c>=osn ){
            prenos = 1;
            c -= osn;
            }
        else
            prenos = 0;
        r._Cifre.push_back( c );
        }
    if( prenos > 0 )
        r._Cifre.push_back( prenos );
    }

static void AbsOduzimanje( const CeoBroj& veci, const CeoBroj& manji, CeoBroj& r )
    {
    unsigned osn = r._Osnova = veci._Osnova;
    int pozajmica = 0;
    unsigned brCifaraManjeg = manji._Cifre.size();
    unsigned brCifaraVeceg = veci._Cifre.size();

    for( unsigned i=0; i<brCifaraVeceg; i++ ){
        int c =
            (int)veci._Cifre[i]
            - (int)( brCifaraManjeg>i ? manji._Cifre[i] : 0 )
            - (int)pozajmica;
        if( c<0 ){
            pozajmica = 1;
            c += osn;
            }
        else
            pozajmica = 0;
        r._Cifre.push_back( c );
        }

    // brisemo vodece nule
    while( r._Cifre.size()>1 && r._Cifre.back() == 0 )
        r._Cifre.pop_back();
    }

// da li je prvi >= drugi
static bool AbsVeci( const CeoBroj& x, const CeoBroj& y )
    {
```

```
        if( x._Cifre.size() > y._Cifre.size() )
            return true;
        if( x._Cifre.size() < y._Cifre.size() )
            return false;
        for( int i=x._Cifre.size(); i>=0; i-- ){
            if( x._Cifre[i] > y._Cifre[i] )
                return true;
            if( x._Cifre[i] < y._Cifre[i] )
                return false;
            }
        // slucaj x==y
        return true;
        }

    //----------------------------
    // Clanovi podaci
    unsigned            _Osnova;
    int                 _Znak;
    vector<unsigned>    _Cifre;
};

ostream& operator << ( ostream& ostr, const CeoBroj& c )
{
    c.Ispis( ostr );
    return ostr;
}

main()
{
    CeoBroj a( 1 );
    CeoBroj b( 1 );
    for( int i=1; i<=100; i++ ){
        a = a * -2;
        b = b + b;
        cout << "2^" << i << " = " << a << endl;
        cout << "2^" << i << " = " << b << endl;
        }

    cout << ( CeoBroj(111) * (222) ) << endl;

    return 0;
}
```