Osnovi računarskih sistema

Milena Vujošević–Janičić

Čas 1

Vežbe 05 04 2005

1.1 Tokovi

Ulazno izlazna funkcionalnost programa u C++-u ostvaruje se preko biblioteke `iostream` (biblioteka **ulaznih i izlaznih tokova**). Ova biblioteka predstavlja objektno orijentisanu hijerarhiju klasa realizovanu putem višestrukog i virtuelnog nasleđivanja. Biblioteka `iostream` je komponenta standardne biblioteke jezika C++.

Ulazne operacije su ugrađene u klasu **istream** (input stream, ulazni tok) a izlazne u klasu **ostream** (output stream, izlazni tok).

1. cin je objekat klase istream vezan za standardni ulaz.

2. cout je objekat klase ostream vezan za standardni izlaz.

Klasa `iostream` nasleđuje obe ove klase i omogućava dvosmernu komunikaciju.

Za izlaz se koristi operator prosleđivanja `<<`. To je binarni operator koji se upotrebljava u infiksnoj notaciji: levi operand je tok kome se prosleđuju podaci a desni operand je objekat koji se prosleđuje. Rezultat je referenca na izlazni tok (objekat klase ostream), čime je omogućeno nadovezivanje više primena ovog operatora. Npr: `cout << x << y;`

Za ulaz se koristi operator izdvajanja `>>`. To je binarni operator koji se upotrebljava u infiksnoj notaciji. Levi operand je tok iz koga se izdvajaju podaci a desni operand je objekat čiji se sadržaj izdvaja iz toka. Rezultat je referenca na ulazni tok (objekat klase istream), čime je omogućeno nadovezivanje više primena ovog operatora. Npr: `cin >> x >> y;`

Primer 1 *Manipulatori.*

```
#include <iostream>

// Manipulator modifikuje interno stanje
// objekta u/i toka.
// Maniplator se primenjuje na objekat u/i toka
// na isti nacin kao da su u pitanju podaci.
// Ukljucujemo datoteku zaglavlja iomanip
```

```
// koja omogucava upotrebu manipulatora
// setw
#include <iomanip>
#include <string>

using namespace std;

main()
{
    int n = 10;
    int* p = &n;
    char c = 'a';
    string s = "ovo je niska";
    char cs[] = "c-ovska niska";

    // iostream biblioteka pruza podrsku
    // za u/i opearcije za ugradjene tipove
    // podataka

    cout << n << endl;
    cout << p << endl;
    cout << (long)p << endl;
    cout << c << endl;
    cout << s << endl;
    cout << cs << endl;

/*
    cin >> n >> c >> s >> cs;

    cout << n << endl;
    cout << c << endl;
    cout << s << endl;
    cout << cs << endl;
*/

    cout << "Ovo se nece videti odmah...";

    // Ovime se data niska smesta u bafer
    // pridruzen objektu cout. Ovaj bafer
    // se prazni iz nekoliko razloga
    // (navescemo dva) tako sto se njegov sadrzaj
    // ispisuje na standardnom izlazu
    // 1. bafer se popunio da kraja pa mora da se isprazni
    // 2. eksplicitno praznjenje, npr pomocu manipulatora
```

```
    // flush, endl ili ends

    // endl je manipulator klase ostream
    // koji u bafer izlaznog toka upisuje
    // znak za novi red, a zatim prepisuje
    // podatke izlaznog bafera klase ostream
    // cout << endl; je ekvivalnetno sa
    // cout << '\n' << flush;

    // ends je manipulator klase ostream
    // koji u bafer izlaznog toka upisuje
    // zavrsni znak, a zatim prepisuje
    // podatke izlaznog bafera klase ostream
    // char ch[2]; ch[0] = 'a'; ch[1] = 'b';
    // cout << ch << ends;
    // je ekvivalentno sa
    // cout << ch << '\0' << flush;

    // ako bi smo napisali sledeci kod
    // cin >> cs;
    // tada bi doslo do greske prilikom izvrsavanja
    // programa ukoliko bi neko uneo rec duzu od
    // velicine niske cs.
    // Ovaj problem se resava upotrebom manipulatora
    // setw( velicina ) koja podesava da se
    // izdvaja iz toka i smesta u cs najvise
    // niska duzine velicina
    cin >> setw(sizeof (cs) ) >> cs;
    cout << cs << endl;

    return 0;
}
/* Izlaz za specifican unos:
10
0012FF70
1245040
a
ovo je niska
c-ovska niska
Ovo se nece videti odmah...
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmm
*/
```

1.1.1 Datotečni tokovi

Ulazno izlazne operacije nad datotekama obezbeđuju klase **ofstream** (izlazni datotečni tok), **ifstream** (ulazni datotečni tok) i **fstream** (datotečni tok za ulaz i izlaz).

Primer 2 *Rezultat rada programa su dve datoteke u koje je prepisan sadržaj datoteke "tokovi2.cpp".*

```
#include <iostream>

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

using namespace std;

main()
{
    // Kreira se ulazni tok inf i on se vezuje za datoteku
    // po imenu "tokovi2.cpp"
    ifstream inf("tokovi2.cpp");

    // Kreira se izlazni tok outf i on se vezuje
    // za datoteku (koja se po potrebi kreira) "izlaz.txt"
    ofstream outf("izlaz.txt");

    while(1){
        // U karakter c smesta se jedan bajt iz ulaznog
        // datotecnog toka koristeci metod get
        // koja iz toka izdvaja jedan bajt
        char c = inf.get();

        // Operator ! primenjen na objekat toka
        // vraca 1 ukoliko citanje nije uspelo
        // Primetimo da !(!f) nije isto sto i f

        if( !inf )
            break;

        // U izlzni tok upisuje se karakter c koristeci
        // metod put koji u izlazni tok upisuje jedan bajt
        outf.put(c);
    }

    // Zatvara se veza izmedju toka outf i datoteke izlaz.txt
    outf.close();
```

```
// Izlazni tok outf vezuje se za novu datoteku izlaz2.txt
// Ova datoteka se otvara za pisanje
// prlikom cega se uklanja njen prethodni sadrzaj

outf.open("izlaz2.txt");

// ekvivalento sa
// outf.open("izlaz2.txt", ios::out);

// 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


// Brise se bit ulaznog toka koji je
// bio postavljen prilikom dolaska do
// kraja fajla. Na ovaj nacin se ulazni
// tok ponovo dovodi u ispravno stanje
inf.clear( ios::eofbit );

// Ulazni tok se postavlja na pocetak
// Metod seekg pomera citanje na apsolutnu
// poziciju u toku (u ovom slucaju na nultu)
inf.seekg(0);

// Ponovo vrsimo prepisivanje datoteke, samo
// sada u drugu izlaznu datoteku
```

```
    while(1){
        char c;
        inf >> c;
        if( !inf )
            break;
        outf << c;
    }
    // Zatvara se veza izmedju izlaznog toka i datoteke
    // i izmedju ulaznog toka i datoteke
    outf.close();
    inf.close();
    return 0;
}
```

Primer 3 *Izračunavanje dužine datoteke.*

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

using namespace std;

unsigned long velicinaDatoteke( char* s )
{

    // Ulazni tok ifs vezuje se za datoteku cije se ime nalazi
    // u nizu karaktera s
    ifstream ifs(s);

    // 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
    ifs.seekg( 0, ios::end );

    // Funkcija tellg vraca trenutnu poziciju u toku
    return ifs.tellg();
}

main()
{
    cout << "Velicina datoteke 'tokovi3.cpp' je "
         << velicinaDatoteke( "tokovi3.cpp" )
         << " bajtova."
         << endl;

    return 0;
```

```
}
/* Velicina datoteke 'tokovi3.cpp' je 401 bajtova. */
```

Primer 4 *Pisanje i čitanje struktura iz toka.*

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

using namespace std;

// Struktura student sadrzi podatke o studentu
struct student
{
    int indeks;
    int godina;
    char ime[50];
    char prezime[50];
};

// Funkcija pisanje upisuje niz studenata u izlaznu datoteku
void pisanje()
{
    // Niz studenata se inicijalizuje
    student studenti[] = {
        { 25, 2002, "Pera", "Peric" },
        { 31, 2001, "Zika", "Zikic" },
        { 17, 1999, "Persa", "Persic" }
    };

    // Izlazni tok f vezuje se za datoteku
    // "studenti.dat" koja se otvara u
    // binarnom rezimu rada
    ofstream f( "studenti.dat", ios::binary );

    // Prolazimo kroz niz studenata
    for( int i=0; i<sizeof(studenti)/sizeof(student); i++ )
        // Upisujemo svakog studenta u datoteku.
        // 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
```

```
        // Adresu i-tog studenta smo konvertovali
        // u pokazivac na char, a duzinu koju koristimo
        // prilikom upisivanja u tok je velicina
        // strukture student
        f.write( (char*)&(studenti[i]), sizeof(student));

}

void citanje( int i )
{
    student s;

    // Ulazni tok vezujemo za datoteku otvorenu
    // u binarnom rezimu rada
    ifstream f( "studenti.dat", ios::binary );

    // U ulaznoj datoteci postavljamo se na
    // odgovarajuce mesto
    f.seekg( i * sizeof(student));

    // Metod read klase istream ima sledeci potpis
    // read( char* adresa, streamsize size)
    // i funkcionise inverzno u odnosu na funkciju
    // write: ona izdvaja size susednih bajtova iz
    // ulaznog toka i smesta ih u memoriji sa
    // pocetkom na adresi adresa
    f.read( (char*)&s, sizeof(student));

    // Proverava se da li je citanje uspelo
    if( f ){
        cout << s.indeks << ' '
             << s.godina << ' '
             << s.ime << ' '
             << s.prezime << endl;
    }
    // ukoliko nije stampa se odgovarajuca poruka
    else
        cout << "Ne postoji " << i << ". student." << endl;
}

main()
{
    pisanje();
```

```
    citanje(2);
    citanje(1);
    citanje(7);
    return 0;
}

/*
17 1999 Persa Persic
31 2001 Zika Zikic
Ne postoji 7. student.
*/
```

Sadržaj