10.6. Rad s datotekama

Temeljna su dva načina zapisa: binarni i tekstualni (ili formatirani).

Kaže se da je izvršen *zapis* u *binarnu* datoteku kada se informacije na disk zapisuju u istom binarnom obliku kako su kodirane u memoriji računala. U tekstualne datoteke se zapis vrši formatirano pomoću slijeda ASCII znakova, na isti način kako se vrši tekstualni ispis na video monitoru. Sadržaj tekstualnih datoteka se može pregledati bilo kojim editorom teksta, dok sadržaj binarnih datoteka može razumjeti samo program (ili programer) koji ih je formirao.

10.6.1 Otvaranje i zatvaranje datoteka

Za manupuliranje s datotekama koriste se objekti klase `ofstream` za pisanje u datoteku i objekti klase `ifstream` za čitanje iz datoteke. Oni naslijeđuju članske funkcije i manipulatere iz `istream` i `ostream` klasa. Deklaracijom:

```
ifstream ulaz;
ofstream izlaz;
```

formiraju se dva objekta: ulaz i izlaz. Da bi se pomoću njih moglo pristupiti nekoj datoteki, mora se otvoriti pristup do te datoteke. To se postiže članskom funkcijom open(str), kojoj argument mora biti string koji sadrži ime datoteke (uključujući i stazu direktorija);

```
ulaz.open("ime_ulazne_datoteke");
izlaz.open("ime_izlazne_datoteke");
```

Pristup ulaznoj datoteci, bit će otvoren ako postoji datoteka zadanog imena i ako ona nije zauzeta od nekog drugog procesa. Pristup izlaznoj datoteci, bit će otvoren ako se može formirati datoteka zadanog imena, ili ukoliko postoji datoteka istog imena.
Provjera ispravnosti otvaranja datoteke može se provesti članskom funkcijom `is_open()`, koja vraća true, ako je datoteka ispravno otvorena. Također, je poželjno ispitati stanje objekta članskom funkcijom `good()`.

```
int x;
ulaz >> x;            // dobava x iz ulazne datoteke
izlaz << x << endl;   // ispis se x u izlaznu datoteku
ulaz.close(); izlaz.close();
```

Nakon završetka rada s datotekom potrebno je zatvoriti datoteku, kako bi omogućili drugim procesima da koriste tu datoteku. To se postiže članskom funkcijom `close()`.

Program vrši kopiranje datoteke imena "copf1.cpp" u datoteku "copyf1.bak".

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

void error(char*str){    cout << str << endl; exit(1);}

int main()
{
    ifstream ulaz ;    ofstream izlaz;

    ulaz.open("copyf1.cpp") ;
    if (!ulaz.good() || !ulaz.is_open())
        error("Ne moze se otvoriti ulazna datoteka") ;

    izlaz.open("copyf1.bak") ;
    if (!izlaz.good() || !izlaz.is_open())
     error("Ne moze se otvoriti izlazna datoteka") ;

    // kopiraj znak po znak iz ulaza na izlaz
    char c ;
    while (ulaz.get(c))      // dobavi s ulaza
          izlaz.put(c);      // upiši na izlaz
    //zatvori datoteku
    ulaz.close();
    izlaz.close();
    return 0;
}
```

Otvaranje datoteke u konstruktoru

Proces otvaranja datoteke može se provesti i tako da se koristi konstruktor sa argumentom tipa char *, koji sadrži ime datoteke.

```
ifstream ulaz("copyf1.cpp");
if (!ulaz)
        error("Ne moze se otvoriti ulazna datoteka")
```

U ovom slučaju se provjera otvaranja datoteke provodi tako da se ispita stanje objekta.
Kompletan primjer, je dan u programu copyf2.cpp.

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

void error(char*str){    cout << str << endl; exit(1);}

int main()
{
    ifstream ulaz("copyf2.cpp") ;
    if (!ulaz) error("Ne moze se otvoriti ulazna datoteka");

    ofstream izlaz("copyf2.bak") ;
    if (!izlaz) error("Ne moze se otvoriti izlazna datoteka") ;

    while (ulaz.get(c)) izlaz.put(c) ;

    return 0;
}
```

Na kraju ovog programa nisu korištene funkcije za zatvaranje datoteka. Razlog tome je što destruktor ofstream i ifstream klasa sam vrši zatvaranje datoteke.

Zatvaranje datoteke se ipak često eksplicitno navodi. Više je razloga:

1. **zatvaranjem datoteke se sav sadržaj iz bafera sprema na disk,**
2. **zatvaranjem datoteke ona je na raspolaganju drugim procesima u računalu,**
3. **ponekad se ista datoteka koristi za ulaz i izlaz**

U programu fileio.cpp ista se datoteka koristi najprije za izlaz, a zatim za ulaz.

```
int main()
{
    char imedatoteke[80];    char buffer[255];
    cout << "Ime datoteka: ";    cin >> imedatoteke;    cin.ignore(1,'\n');

    ofstream izlaz(imedatoteke);       // formiraj izlaznu datoteku

    cout << "Otkucajte liniju teksta: ";
    cin.getline(buffer,255);           // dobavi tekst s tipkovnice
    izlaz << buffer << "\n";           //  i upiši  u datoteku
    izlaz.close();                     // zatvori datoteku

    ifstream ulaz(imedatoteke); // otvori za čitanje

    cout << "\nU datoteci je upisan tekst: \n";
    // Dobavi i ispiši sadržaj datoteke na standardni izlaz
    char ch;
    while (ulaz.get(ch)) cout << ch;
    cout << "\n***Kraj***\n";
    return 0;
}
```

10.6.2 Modovi otvaranja datoteke

Funkcija open() i konstruktori fstream klasa se mogu koristiti i sa dodatnim cjelobrojnim argumentom:

```
fstream& fstream (char *ime, int mod);
void fstream::open (char *ime, int mod);
```

Tim argumentom se dodatno određuje način otvaranja datoteke, a on može imati slijedeće vrijednosti:

`ios::app`	**Prije svakog upisa trenutna pozicija se postavlja na kraj datoteka, odnosno sadržaj se uvijek dodaje na kraj postojeće datoteke (eng. append).**
`ios::ate`	**Početno se postavlja na kraj datoteke (at end). Ne mijenja se trenutni sadržaj datoteke.**
`ios::binary`	**Datoteka se otvara u binarnom modu (važno samo kod MSDOSa ili Windowsa). U tom slučaju se ne vrši pretvorba '\n' u "\n\r"**
`ios::in`	**otvara se postojeća datoteka za čitanje.**
`ios::out`	**Otvara se datoteka za pisanje. Ukoliko ne postoji datoteka zadanog imena kreira se nova datoteka.**
`ios::trunc`	**Započni rad s praznom datotekom. Postojeći sadržaj se gubi.**

Primjer: U programu append.cpp demonstrirano je kako se dopunjuje unos u neku datoteku.

```
int main()
{
    char imedatoteke[80];    char buffer[255];
    cout << "Otipkaj ime datoteke: ";    cin >> imedatoteke;
    ifstream fin(imedatoteke);
    if (fin)                // datoteka postoji
    {
        cout << "Trenutni sadrzaj datoteke:\n";
        char ch;
        while (fin.get(ch))
            cout << ch;
        cout << "\n*** Kraj***\n";
    }
    fin.close();

    cout << "\nOtvara " << imedatoteke
         << " u append modu...\n";

    ofstream fout(imedatoteke,ios::app);
    if (!fout){ cout << "Ne moze otvoriti " << imedatoteke << endl;
        return(1);
    }

    cout << "\nTekst koji spremas u datoteku: ";   cin.ignore(1,'\n');
    cin.getline(buffer,255);
    fout << buffer << "\n";
    fout.close();
```

```
    fin.open(imedatoteke);
    if (!fin)
    {
      cout << "Ne moze otvoriti " << imedatoteke << endl;
      return(1);
    }
    cout << "\nSadrzaj datoteke je:\n";
    char ch;
    while (fin.get(ch))
          cout << ch;
    cout << "\n***Kraj***\n";
    fin.close();
    return 0;
 }
```

Kada prvi put izvršimo ovaj program dobit će se ispis:

```
Otipkaj ime datoteke: app
Otvara app u append modu...

Tekst koji spremas u datoteku: Prvi tekst!

Sadrzaj datoteke je:
Prvi tekst!

***Kraj***
```

Ako ponovo izvršimo ovaj program dobit će se ispis:

```
Otipkaj ime datoteke: app
Trenutni sadrzaj datoteke:
Prvi tekst!

*** Kraj***

Otvara app u append modu...

Tekst koji spremas u datoteku: Drugi tekst!

Sadrzaj datoteke je:
Prvi tekst!
Drugi tekst !

***Kraj***
```

Primjer rada s binarnim datotekama

U binarne datoteke se zapisuje binarni sadržaj memorijskih objekat. Kod MSDOS i Windows računala tada je potrebno otvoriti datoteke s zastavicom `ios::binary`. Kod drugih operativnih sustava ova zastavica nema nikakovo značenje.

U radu s binarnim datotekama koriste se članske funkcije `get()`, `put()`, `read()` i `write()`.
Princip je sljedeći:

Ako se neka struktura podataka zapisuje s funkcijom `write()`, kasnije se ona očitava s funkcijom `read()`.

Obje ove funkcije primaju dva argumenta; prvi je adresa objekta, a drugi je broj bajta koji se zapisuje u datoteku ili učitava iz datoteka.

```
istream& istream::read(char *buffer, int len);
ostream& ostream::write(char *buffer, int len);
```

Pošto je prvi parametar deklariran kao pokazivač na char, bit će potrebno pri pozivu funkcije primijeniti operator pretvorbe tipova u svim slučajevima osim kod rada sa stringom. Primjerice;
ako imamo objekt koji ima tip Tip, njegov binarni zapis cijelog vršimo sa

```
Tip obj;

izlaz.write((char *) &obj, sizeof(Tip));
```

U programu binfile.cpp demonstriran je rad s binarnom datotekom, u koju se zapisuje i očitava sadržaj objekta.

```
class Tocka
{
    int m_x, m_y;
public:
       Tocka(int x=0, int y=0):m_x(x), m_y(y){}
       int x() const { return m_x; }
       void x(int x) { m_x =x; }
       int y() const { return m_y; }
       void y(int y) { m_y =y; }
};

int main()
    {
       char imedatoteke[80];
       char buffer[255];

       cout << "Upisite ime datoteke: ";
       cin >> imedatoteke;
       ofstream fout(imedatoteke, ios::binary);
       if (!fout)  {
         cout << "Ne moze se otvoriti "
                 << imedatoteke << " za pisanje.\n";
            return(1);
       }

       Tocka T1(50,100);
       fout.write((char*) &T1,sizeof (Tocka));
```

```
        fout.close();

        ifstream fin(imedatoteke,ios::binary);
        if (!fin){
             cout << "Ne moze se otvoriti "
                  << imedatoteke << " za citanje.\n";
             return(1);
        }

        Tocka T2;
        fin.read((char*) &T2, sizeof(Tocka));
        cout << "Tocka x=" << T2.x()
             << " y=" << T2.y() << endl;

        return 0;
}
```

fstream klasa

Kada se neka datoteka koristi za ulaz i izlaz povoljno je koristiti klasu fstream. Njome se mogu deklarirati tokovi koji služe za izlaz, ulaz i za ulaz/izlaz. Promjerice,

```
fstream file("ime", ios::in | ios::out);
```

stvara tok `file` kojim se može vršiti ulazne i izlazne operacije. Primjer će biti dan u sljedećem odjeljku.

10.6.3 Sekvencijani i proizvoljni pristup datotekama

Sekvencijalni pristup datoteci označava operacije s datotekama u kojima se čitanje ili pisanje uvijek vrši na kraju datoteke.
Proizvoljni ili slučajni pristup datoteci (random access) označava operacije s datotekama u kojima se čitanje ili pisanje može usmjeriti na proizvoljno mjesto u datoteci.

Svakoj otvorenoj datoteci prideljen je jedan pozicijski indikator koji označava poziciju u bajtima od početka datoteke na kojoj će biti izvršeno čitanje ili pisanje. Vrijednost pozicijskog indikatora se može očitati funkcijom

```
pos_type tellg();
```

koja vraća cjelobrojnu vrijednost tipa pos_type (ekvivalent je tip long).

Proizvoljni pristup datoteci ima smisla samo kod binarnih datoteka, kod kojih čitanje i pisanje se vrši kontrolirano bajt po bajt. On se ostvaruje pomoću funkcije seekg().

```
istream &istream::seekg(off_type pomak, ios::seekdir ref_poz = ios::beg);
```

tako da se trenutna pozicija postavlja na vrijednost koja je jednaka pomaku (`pomak`) od referentne pozicije (`ref_poz`). Vrijednost referentme pozicije (tipa `ios::seekdir`) se zadaje s tri konstante: ios::beg, ios::cur i ios::end.

`ios::beg`	**Referentna pozicija je početak datoteke. Ako se ne zada drugi argument tada se podrazumijeva `ios::beg` is used.**
`ios::cur`	**Referentna pozicija je `trenutna pozicija (koja se dobije s tellg()` funkcijom).**
`ios::end`	**Referentna pozicija je kraj datoteke. Obično se za pomak koristi negativna vrijednost. Ako pomak ima pozitivnu vrijednost pozicijski indikator se postavlja iza kraja datoteke. Čitanje s te pozicije uzrokuje dojavu greške. Pisanje iza EOF je dozvoljeno. Ono se izvršava tako da se od trenutnog kraja datoteke do zadane pozicije upis dopunjuje s nul-znakom. Postavljane pozicijskog indikatora prije početka datoteke nije dozvoljeno.**

Primjer: U programu fseek.cpp generira se datoteka "random.dat" s 50 slučajnih cijelih brojeva. Zatim se po proizvoljnom redoslijedu čita vrijednosti iz te datoteke. Redoslijed bira korisnik tako da unosi indeks elementa. Program završava kada korisnik unese negativnu vrijednost.

```
/* Program fseek.cpp */
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;

#define MAX 50

int main()
{
    int data, i, niz[MAX];
    long indeks;
    // Inicijaliziraj niz od 50 elemenata
    for (i = 0; i < MAX; i++)        niz[i] = rand();
```

```
    // Otvori binarnu datoteku RANDOM.DAT za čitanje i pisanje. */
    fstream file("RANDOM.DAT",  ios::binary|ios::in| ios::out);
    if (!file){
        cerr << "\nGreska pri otvaranje datoteke";
        exit(1);
    }
    /* upiši niz */
    file.write((char *) niz, sizeof(niz));

    /* Pitaj korisnika koji element zeli ucitati, zavrsi za - vrijednost */
    while (1)   {
        cout << "\nIzaberi element datoteke: 0 - " <<MAX-1
             << " ili -1 za kraj: ";
        cin >> indeks;
        if (indeks < 0)   break;
        else if (indeks > MAX-1) continue;

        // Postavi pozicijski indikator datoteke
        file.seekg(indeks*sizeof(int), ios::beg);

         // Zatim učitaj element
        // i prikazi njegovu vrijednost. */
        file.read((char*)&data, sizeof(int));
        cout <<"\nElement " << indeks  <<" ima vrijednost " << data;
    }
    cout << "Kraj" << endl;
    return(0);
}
```