

Rad s datotekama

file streams

Dejan Ljubobratović, mag. educ. math. et inf.

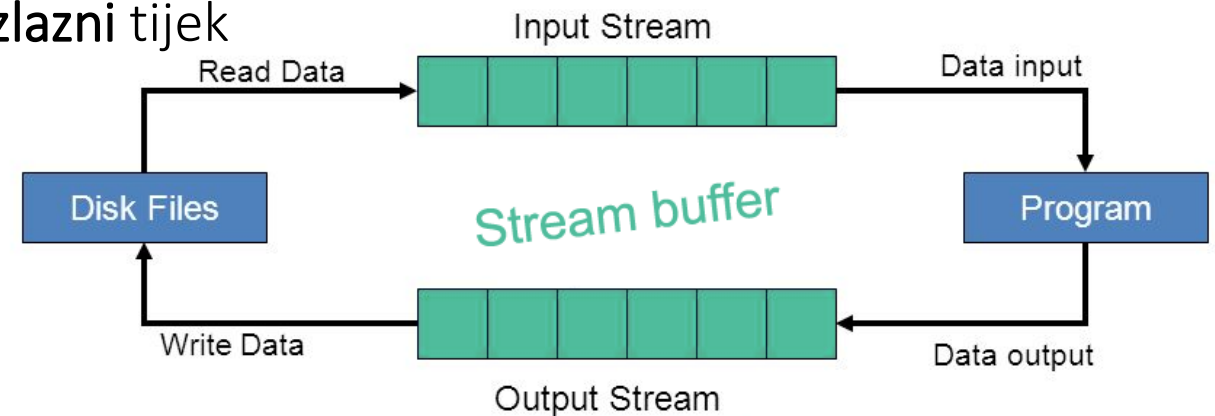
dejan.ljubobratovic@uniri.hr

Fakultet informatike i digitalnih tehnologija, Sveučilište u Rijeci

Što su tijekovi (engl. streams)

C++ program vidi ulaz (ili izlaz) kao **tijek** bajtova.

- Na ulazu, program **ekstrahira (>>)** bajtove iz ulaznog tijeka
- Na izlazu, program **umeće (<<)** bajtove u izlazni tijek



Stream je tijek podataka.

Tijek posreduje između programa i izvora ili odredišta tijeka

- Na taj način, ulaznim i izlaznim podacima rukuje se uporabom **međuspremnik tijeka** (stream buffer)
- Međuspremnik je blok memorije koji se koristi kao posredna, privremena memorija za prijenos informacije između programa i uređaja

... ulazni i izlazni tijekovi (I/O streams)

I/O odnosi se na **ulaz** i **izlaz** programa (input/output)

Ulaz je dodijeljen programu preko objekta tijeka

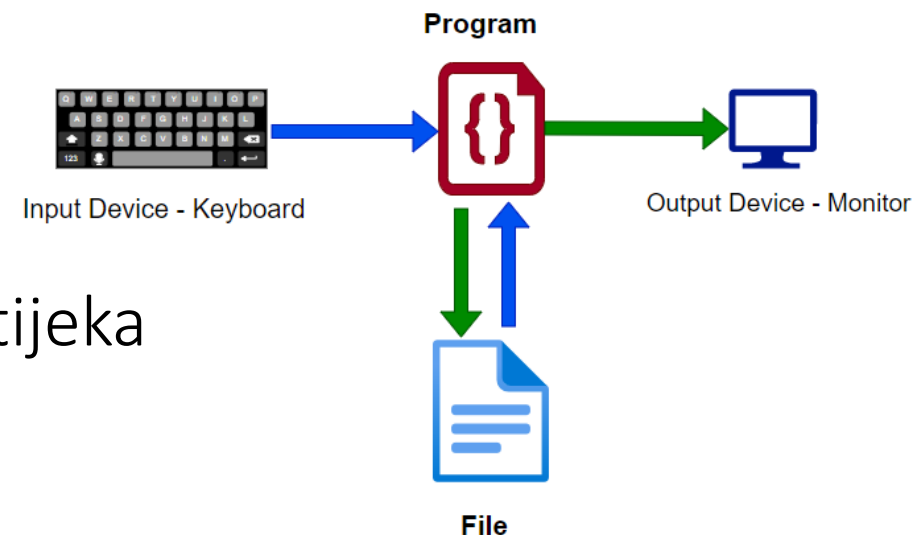
Ulaz može biti sa:

- tipkovnice
- datoteke

Izlaz je dodijeljen izlaznom uređaju preko objekta tijeka

Izlaz može biti:

- na ekran
- u datoteku



Kao i za druge varijable, za **varijablu tijeka** vrijedi:

- mora se **deklarirati** prije uporabe
- mora se **inicijalizirati** prije nego joj se dodjele vrijednosti
 - inicijalizirati tijek znači **povezati** tijek sa datotekom
 - **vrijednost** varijable tijeka **je** zapravo **datoteka** s kojom je povezana
- može mijenjati svoju vrijednost
 - mijenjati varijablu tijeka znači prekinuti njezinu vezu sa jednom datotekom i povezati je sa drugom datotekom

Ulazno-izlaznim tijekovima se rukuje funkcijama biblioteke `iostream`.

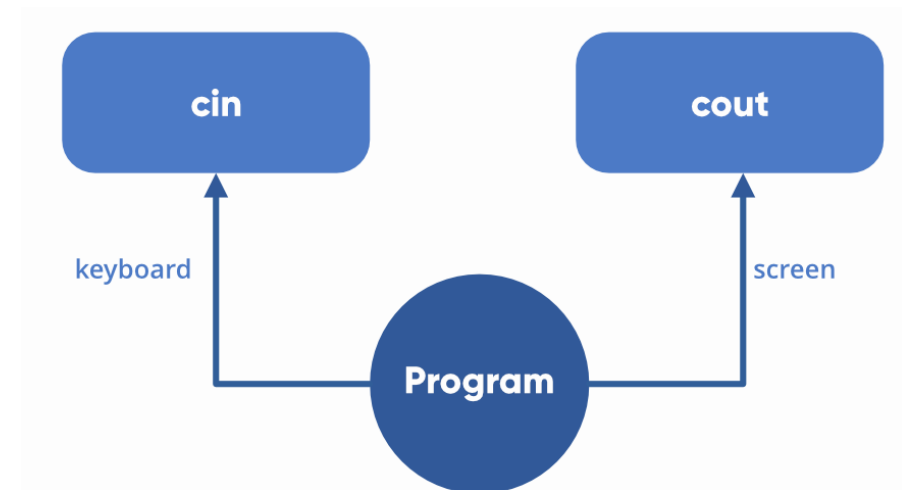
Dvije funkcije koje su najčešće u uporabi su `cin` i `cout`:

- `cin` - ulazni tijek povezan sa tipkovnicom
- `cout` - izlazni tijek povezan sa ekranom

`cin` i `cout` su definirani u biblioteci `iostream`

Koristimo direktivu include: `#include <iostream>`

Možemo deklarirati vlastite tokove za uporabu sa datotekama.



`iostream` biblioteka sadrži niz ulazno-izlaznih funkcija koje omogućavaju rukovanje ulazno-izlaznim tokovima.

Deklariranje varijable ulaznog tijeka

Tijekovi za ulazne datoteke su tipa (klase) `ifstream`

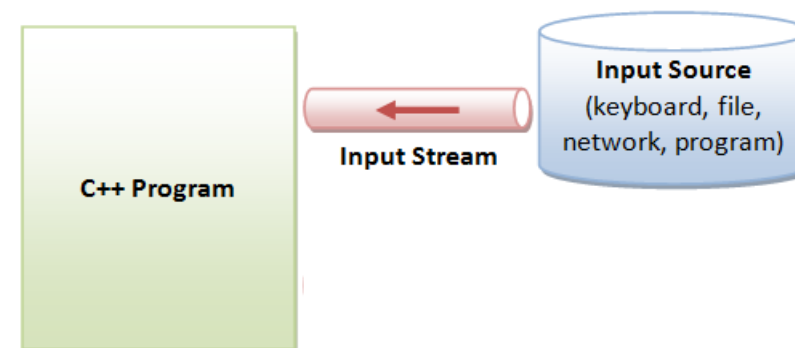
Tip `ifstream` je definiran u biblioteci `fstream`.

Potrebno koristiti direktive `include` i `using`:

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

Deklariranje tijeka ulazne datoteke:

- `ifstream ulazni_tijek;`



Deklariranje varijable izlaznog tijeka

Tijekovi za izlazne datoteke su tipa (klase) `ofstream`

Tip `ofstream` je definiran u biblioteci `fstream`

Potrebno koristiti direktive `include` i `using`:

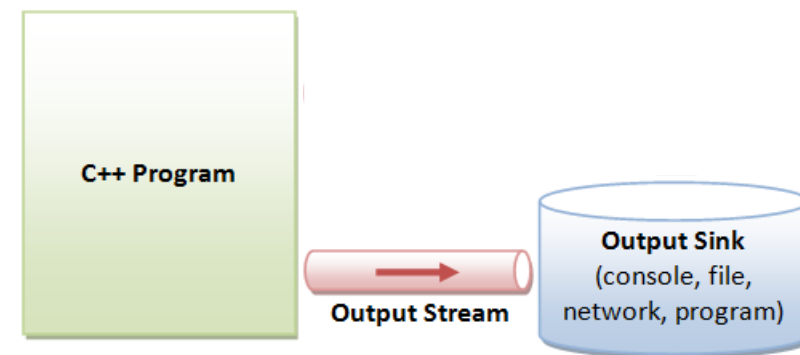
- `#include <fstream>`
- `using namespace std;`

`<fstream>` omogućava dostupnost klasa `ifstream` i `ofstream`

Deklariranje varijable izlaznog tijeka:

- `ofstream izlazni_tijek;`

`izlazni_tijek` je ime varijable tijeka (informacija „izlazi” iz datoteke)



Otvaranje datoteke

(povezivanje s datotekom na disku ili kreiranje ako je nema)

Kada smo deklarirali varijablu tijeka, povezujemo je s datotekom

- **Povezivanje tijeka** s datotekom predstavlja **otvaranje** datoteke
- Koristimo funkciju **open** objekta tijeka:

```
ulazni_tijek.open("ulaznaDatoteka.dat");
```

`ulaznaDatoteka.dat` predstavlja ime datoteke na disku

Za neke kompajlere potrebno je navesti i operacijski mod:

Npr.

```
izlazni_tijek.open("ulaznaDatoteka.dat", ios::out);
```



Oprez!

Kreiranje (otvaranje) datoteke
briše i zamjenjuje datoteku istog
imena na disku ako postoji!

Otvaranje datoteke (2. način)

- Ako želimo otvoriti datoteku na neki drugi način, **osim naziva datoteke**, kao **drugi parametar** dodaje se **parametar mode**.
- Npr. ako se otvara datoteka za pisanje podataka, bitno je navesti želimo li brisati postojeće podatke ako oni postoje u datoteci ili se podaci nadopisuju u datoteku na već postojeće.
- Mode je varijabla tipa integer i prethodno joj se mora pridružiti vrijednost.
- Primjer: ako želimo podatke samo dodati na već postojeće podatke u datoteci (bez brisanja sadržaja), za mode se navodi `ios::app`
- moguće je kombinirati više načina otvaranja datoteke (odvajaju se oznakom `|` npr. `int mode = ios::app | ios::nocreate`)

Načini otvaranja datoteke (mode)

in	Otvora za input (default za ifstreams)
out	Otvora za output (default za ofstreams)
app	Dodaje podatke, uvijek piše podatke na kraj datoteke (samo za output)
ate	Traži kraj datoteke
binary	Otvora datoteku u binarnom obliku
trunc	Briše sadržaj datoteke ako on već postoji
nocreate	Ako datoteka ne postoji, ne može otvoriti datoteku
noreplace	Ako datoteka postoji neće se otvoriti za obradu ako <i>ate</i> ili <i>app</i> nisu postavljeni

Kada je povezana s datotekom, varijabla ulaznog tijeka se može koristiti za preuzimanje ulazne vrijednosti s operatorom ekstrakcije (>>) na isti način kao što to činimo naredbom `cin`.

Primjer:

```
int n1, n2;  
ulazni_tijek >> n1 >> n2;
```

Podaci u tekstualnoj datoteci moraju biti razdvojeni razmakom, kao i kad ih unosimo s tipkovnice

Način za provjeru ako je dostignut kraj datoteke:



Logički izraz (`ulaz >> sljedeci`)

- Čita vrijednost iz *ulaz* i pohranjuje ga u *sljedeci*
- **True** ako se vrijednost može pročitati i spremiti u *sljedeci*
- **False** ako nema vrijednosti koja se može pročitati (**kraj datoteke**)

Pisanje u tekstualnu datoteku

Izlazni tijek radi slično kao ulazni tijek

```
ofstream izlazni_tijek;  
izlazni_tijek.open("izlaznaDatoteka.dat");  
izlazni_tijek << "n1= " << n1 << "n2= " << n2;
```

- Nakon uporabe datoteke, moramo je zatvoriti
 - Time se prekida veza datoteke i varijable tijeka
 - Zatvaramo datoteke da bi reducirali mogućnost oštećivanja datoteke ako se izvođenje programa ne završi regularno
- `izlazni_tijek.close();`
- Važno je zatvoriti izlaznu datoteku ako vaš program kasnije treba čitati ulaz iz te izlazne datoteke
- Sustav će automatski zatvoriti datoteke ako to zaboravite učiniti, ako se izvođenje vašeg programa regularno završava



Datoteka je slična ladici:
kada uzimamo nešto iz
nje ili spremamo nešto u
nju moramo je **otvoriti**, a
kad obavimo moramo je
zatvoriti

Primjer

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    string rijec;

    //Upisujemo riječ u tekstualnu datoteku
    ofstream izlazni_tijek;
    izlazni_tijek.open("datoteka.txt");
    izlazni_tijek << "Pozdrav";
    izlazni_tijek.close();

    //Čitamo riječ iz tekstualne datoteke
    ifstream ulazni_tijek;
    ulazni_tijek.open("datoteka.txt");
    ulazni_tijek >> rijec;
    cout << rijec;
    ulazni_tijek.close();
    return 0;
}
```

Otvaranje datoteke ne mora uspjeti iz više razloga

Najčešći razlozi uključuju:

- Nepostojanje datoteke na disku
- Pogrešno navedeno ime datoteke na disku

Kada dođe do ove greške program se često nastavlja izvršavati bez poruke o greški!

Pripadna funkcija `fail`, može se koristiti za provjeru uspjeha operacija tijekom:

- `fail` vraća logički tip (`true` ili `false`)
- `fail` vraća `true` ako operacija tijekom nije uspjela



Prekid izvođenja programa (exit)

Kada funkcija tijekom `open` ne uspije, možemo zaustaviti program

Funkcija `exit` prekida rad programa

- `exit` vraća parametar operacijskom sustavu
- `exit` uzrokuje prekid programa
- `exit` **NIJE** pripadna funkcija

`exit` zahtijeva sljedeće direktive `include` i `using`:

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

Umjesto `exit`, moguće je koristiti i funkciju `return()` za istu namjenu.



Na nekim kompajlerima nije potrebno navoditi biblioteku `cstdlib`, pošto `iostream` već sadrži naredbu `exit`, no ne smeta ju navesti radi bolje kompatibilnosti i izbjegavanja mogućih grešaka.

Primjer fail()

```
#include <iostream>
#include <cstdlib>
#include <fstream>
using namespace std;
int main()
{
    ifstream ulaz;
    ulaz.open("podaci.dat");
    if(ulaz.fail())
    {
        cout << "Otvaranje datoteke nije uspjelo";
        exit(1);
    }
    ulaz.close();
}
```

Otvaranje datoteke nije uspjelo!

Process returned 0 (0x0) execution time : 0.013 s
Press any key to continue.

Možemo koristiti funkciju `assert()` za otkrivanje problema s otvaranjem datoteke. U tom slučaju program prekida s radom i ispiše se na ekranu poruka o grešci: `Assertion failed!`

`Assert()` je funkcija koja testira istinitost nekog izraza. Ako je izraz istinit, program nastavlja normalno, ali ako je izraz neistinit, program se prekida i prikazuje se poruka o pogrešci.

`assert()` zahtijeva sljedeće direktive `include` i `using`:

```
#include <assert.h>
using namespace std;
```

Primjer `assert()`

```
#include <iostream>
#include <fstream>
#include <assert.h>
using namespace std;
int main()
{
    ifstream ulaz;
    ulaz.open("podaci.dat");
    assert(!ulaz.fail());
    ulaz.close();
}
```

Assertion failed!

Program: c:\Zadatak\bin\Debug\Zadatak.exe

File: c:\Zadatak\test.cpp, Line 9

Expression: !ulaz.fail()

Process returned 3 (0x3) execution time : 3.1 s

Press any key to continue.

Primjer `assert()`

```
#include <iostream>
#include <assert.h>
using namespace std;
```

```
int main()
{
    int a=5;
    assert(a<0);
    return 0;
```

Assertion failed!

Program: c:\Zadatak.exe

File: c:\Zadatak\test.cpp, Line 7

Expression: a<0

Process returned 3 (0x3) execution time: 2.8 s

Press any key to continue

Dodavanje podataka ios::app

U navedenim primjerima u kojima se koristila izlazna datoteka, koristila se **nova datoteka** za izlaz

Ako je izlazna datoteka već sadržavala podatke, podaci su se **gubili**.

Da bi **dodali** novi izlaz na kraj **postojeće** datoteke koristimo parametar **ios::app** definiranu u biblioteci **iostream**:

```
izlaz.open("vazni_podaci.txt", ios::app);
```



Kada koristimo parametar **ios::app**, ako datoteka ne postoji, biti će kreirana nova

Unos imena datoteke

Korisnik programa može unijeti proizvoljno ime datoteke za ulaz ili izlaz

Ime se unosi kao znakovni niz

Deklariranje varijable za prihvatanje imena:

```
char ime_datoteke[16];
```



Primjer

Unesite ime datoteke: datoteka.dat
Otvaranje datoteke nije uspjelo.

Process returned 1 (0x1) execution time : 4.9 s
Press any key to continue.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    char ime_datoteke[16];
    cout << "Unesite ime datoteke: ";
    cin >> ime_datoteke;
    ifstream ulazni_tijek;
    ulazni_tijek.open(ime_datoteke);
    if (ulazni_tijek.fail())
    {
        cout<<"Otvaranje datoteke nije uspjelo.";
        cout<<endl;
        exit(1);
    }
}
```

Ulazne datoteke koje koristi program mogu biti različite duljine

- Program obično ne zna broj podataka u datoteci

Način za provjeru da li je dostignut kraj datoteke:

- Logički izraz (`ulazni_tijek >> sljedeci`):
 - Čita vrijednost iz `ulazni_tijek` i pohranjuje ga u `sljedeci`
 - **True** ako se vrijednost može pročitati i spremiti u `sljedeci`
 - **False** ako nema vrijednosti koja se može pročitati (kraj datoteke)

Primjer kraja datoteke

Računamo prosjek brojeva u zadanoj datoteci.

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

int main()
{
    ifstream ulaz;
    ulaz.open("brojevi.txt");
    double sljedeci, suma=0;
    int brojac=0;

    while(ulaz>>sljedeci)
    {
        suma=suma+sljedeci;
        brojac++;
        cout << sljedeci << endl;
    }

    cout << "Prosjek brojeva u datoteci je: ";
    cout << suma/brojac;

    ulaz.close();
}
```

Otkrivanje kraja datoteke eof()

Pripadna funkcija eof() provjerava kraj datoteke

- Pripadna funkcija svakog ulaznog tijeka
- eof je kratica za *"end of file"*
- eof() vraća logičku vrijednost
 - **true** ako je dostignut kraj datoteke
 - **false** kada ima još podataka za čitanje

Obično se koristi za provjeru da li smo stigli do kraja datoteke

Primjer: `if(!ulazni_tijek.eof())`

Korištenje funkcije eof()

Sljedeća petlja čita svaki znak i ispisuje ga na ekran

```
ulazni_tijek.get(sljedeci);  
while (!ulazni_tijek.eof())  
{  
    cout << sljedeci;  
    ulazni_tijek.get(sljedeci);  
}
```



(!ulazni_tijek.eof())
postaje false kada se
pokuša čitati znak nakon
što su pročitani svi
znakovi u datoteci

- Kraj datoteke se označava posebnim znakom
- `ulazni_tijek.eof()` je još uvijek **true** nakon što je pročitani zadnji znak u datoteci
- `ulazni_tijek.eof()` postaje **false** tek kada se pročita posebna oznaka kraja

Kako testirati kraj datoteke?

Vidjeli smo dvije metode

- `while(ulazni_tijek>>sljedeći)`
- `while(!ulazni_tijek.eof())`

Koji je bolji?

- Općenito, koristimo `eof()` kada se ulaz tretira kao tekst i koristimo pripadnu funkciju `get()` za čitanje ulaza
- Općenito, koristimo metodu s operatorom ekstrakcije (`>>`) kada čitamo numeričke podatke

Kreirajte mapu ***vjezba_1*** i u njoj tekstualnu datoteku ***ulaznaDatoteka.txt*** te .cpp file

- a) Kreirajte **ulazni tok** naziva *ulaz* te ga povežite s datotekom *ulaznaDatoteka.txt* smještenom u istom folderu (*vjezba_1*) u kojem se nalazi vaša .cpp datoteka.
- b) Kreirajte **izlazni tok** naziva *izlaz* te ga povežite s datotekom *izlaznaDatoteka.txt* smještenom na Desktopu vašeg računala.

- a) U ulaznu datoteku koju ste prethodno kreirali (*ulaznaDatoteka.txt*) ručno upisati 3 cijela broja, svaki u svom retku.
- b) Nadopuniti kod vašeg programa tako da se u program učitavaju 3 broja iz datoteke *ulaznaDatoteka.txt*
- c) Zbrojiti učitane brojeve i njihov zbroj zapisati u datoteku *izlaznaDatoteka.txt*

- Modificirajte prethodni program tako što ćete dodati provjeru uspješnosti otvaranja datoteke. Ako otvaranje nije uspješno ispisuje se adekvatna poruka. Testirajte program tako što ćete onemogućiti otvaranje datoteke (npr. pogrešno ime datoteke)

Zadatak 4

- a) Napišite program za evidenciju prodaje koja je u ovom zadatku potpuno pojednostavljena: prati se broj prodanih računala u periodu od 12 mjeseci. Program treba omogućiti korisniku unos broja prodanih računala po mjesecima. Unesene vrijednosti pohranjuju se i u datoteku *prodaja.txt*
- b) Nakon unosa potrebno je izračunati koliki je prosjek prodaje te u kojim mjesecima je prodaja natprosječna. Ispisati sve podatke u datoteku *obracun.txt*.

Zadatak 4 a

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int main() {
    ofstream izlaz;
    float p[12];
    int i=0;
    izlaz.open("prodaja.txt");
    if(izlaz.fail()) {
        cout<<"Greska kod otvaranja datoteke prodaja.txt"<<endl;
        exit(1);
    }
    cout<<"Unos vrijednosti prodaje za 12 mjeseci: "<<endl;
    for(int i=0; i<12; i++) {
        cout<<i+1<<". mjesec: ";
        cin>>p[i];
        izlaz<<p[i]<<endl;
    }
    izlaz.close();
    return 0;
}
```

```

#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int main() {
    ofstream izlaz;
    float p[12];
    int i=0;

    float prosjek = 0;

    izlaz.open("prodaja.txt");
    if(izlaz.fail()) {
        cout<<"Greska kod otvaranja datoteke prodaja.txt"<<endl;
        exit(1);
    }
    cout<<"Unos vrijednosti prodaje za 12 mjeseci: "<<endl;
    for(int i=0; i<12; i++) {
        cout<<i+1<<" mjesec: ";
        cin>>p[i];
        izlaz<<p[i]<<endl;
        prosjek += p[i];
    }

    prosjek /= 12;
    izlaz.close();
    izlaz.open("obracun.txt");
    if(izlaz.fail()) {
        cout<<"Greska kod otvaranja datoteke obracun.txt"<<endl;
        exit(1);
    }
    cout<<"Ispis natprosječnih mjeseci prodaje u datoteku"<<endl;
    izlaz<<"Godisnji prosjek iznosi: "<<prosjek<<endl;
    izlaz<<"Natprosječni mjeseci prodaje"<<endl;
    for(int i=0; i<12; i++) {
        if(p[i]>prosjek) {
            izlaz<<i+1<<" mjesec: "<<p[i]<<endl;
        }
    }
    izlaz.close();
    return 0;
}

```

Zadatak 4 b