Programiranje 1

Datoteke

- Motivacija...

Datoteke

- Datoteke omogućavaju **trajnu pohranu** podataka
- Pdaci ostaju dostupni i nakon izvođenja programa
- Ulazna datoteka može se koristiti više puta ponovo.
  - Npr., nije potrebno ponovo utipkavati podatke kod testiranja
- Možemo kreirati novu ulaznu datoteku ili za ulaznu datoteku možemo koristiti izlaznu datoteku nekog drugog programa.
- Datoteke nam omogućavaju rad sa većim skupovima podataka (matrice, liste)

Datoteke

- Dva oblika datoteka jesu:
  - **tekstualne** koje sadrže (formatirani) tekst
  - **binarne** koje mogu sadržavati različite tipove podataka kodirane u binarnom obliku.

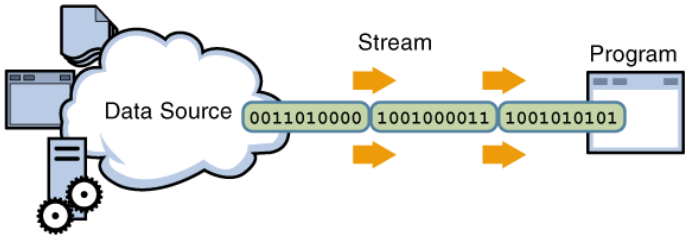
- Ovdje će biti opisan rad s tekstualnim datotekama

Rad s datotekama

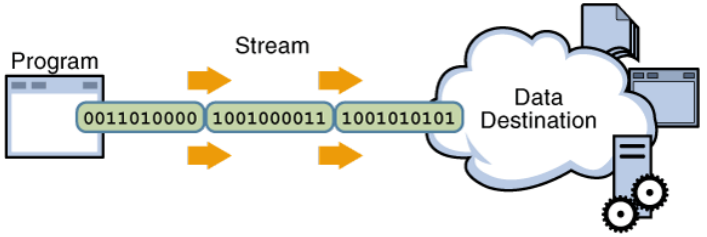
- Osnovne operacije s datotekom:
  - **otvaranje datoteke** – otvara se komunikacija između programa i datoteke
  - **unos i/ili ispis podataka** iz datoteke
  - **zatvaranje datoteke** – ne smije se preskočiti, bitno da bi se oslobodili resursi kako bi se kasnije moglo pristupiti datoteci (ukoliko zaboravimo zatvoriti datoteku, neće se moći adekvatno pristupiti podacima)

Tok podataka (*stream*)

- engl. ***stream*** – tok, tijek podataka
- Ulazni tok (engl. input stream)
  - podaci "teku" u program



- Izlazni tok (engl. output stream)
  - podaci "teku" iz programa



Tok podataka (*stream*)

- Stream je standardni I/O objekt koji pripada nekoj od klasa:
  - **ifstream** – podaci teku iz datoteke u program (*input*)
  - **ofstream** – podaci teku iz programa u datoteku (*output*)
  - **fstream** – podaci mogu biti i ulazni i izlazni (podržava i *input* i *output*)
- Napomena: Postoje i druge klase koje se odnose na input i otuput podataka

Tok podataka

- **`cin`**
  - Ulazni tok povezan s tipkovnicom
- **`cout`**
  - Izlazni tok povezan s ekranom
- **`cin`** i **`cout`** - definirani su u biblioteci iostream
  - Koristimo direktivu include: **`#include <iostream>`**
- Analogno korištenju **`cin`** i **`cout`** tokova koriste se ulazni i izlazni tokovi datoteka, a potrebno je uključiti biblioteku **`fstream`**

Tok podataka

- Ulaz je dodijeljen **programu** preko toka
- Ulaz može biti:
  - **s tipkovnice**
  - **iz datoteke**
- Ako ulazni tok teče s tipkovnice, program će prihvatiti podatke s tipkovnice (cin>>)
- Ako ulazni tok teče iz datoteke, program će prihvatiti podatke iz datoteke (datoteka>>)

Tok podataka

- Izlaz je dodijeljen **izlaznom uređaju** preko toka
- Izlaz može biti :
  - **na ekran**
  - **u datoteku**
- Ako izlazni tok teče na ekran, program šalje podatke na ekran (cout<<)
- Ako ulazni tok teče u datoteku, program šalje podatke u datoteku (datoteka<<)

Tok podataka

- Kao i druge varijable, varijabla toka:
  - Mora se **deklarirati** prije uporabe
  - Mora se **inicijalizirati** prije nego joj se dodjele vrijednosti
    - Inicijalizirati tok znači povezati tok sa datotekom – pridružiti toku vrijednost koji predstavlja naziv datoteke
    - Vrijednost varijable toka je zapravo datoteka s kojom je povezana
  - Može mijenjati svoju vrijednost
    - Mijenjati varijablu toka znači prekinuti njezinu vezu sa jednom datotekom i povezivanje sa drugom datotekom

Tok podataka

- Tokovi za ulazne datoteke su tipa **ifstream** (a može se koristiti i općenito fstream)

- Tip ifstream je definiran u biblioteci **fstream**
  - Koristiti direktive include i using

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

- Deklariramo tok ulazne datoteke uporabom:

```
ifstream ulaz;
```

Tok podataka

- tokovi za izlazne datoteke su tipa **ofstream** (a može se koristiti i općenito fstream)

- Tip ofstream je definiran u biblioteci **fstream**
  - Koristiti direktive include i using

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

Deklariramo tok izlazne datoteke uporabom:

```
ofstream izlaz;
```

Otvaranje datoteke

- Deklarirana varijabla toka povezuje se sa datotekom na disku
- Ime datoteke na disku koristimo samo u naredbi za otvaranje datoteke, a za ostatak rukovanja koristi se ime toka koji joj je dodijeljen

- Povezivanje toka sa datotekom predstavlja otvaranje datoteke
  - 1. način: koristimo funkciju **open** u klasi (ulaznog/izlaznog) toka, bez navođenja dodatnih parametara:

```
ulaz.open("ulaznaDatoteka.txt");
izlaz.open("izlaznaDatoteka.dat");
```

Primjeri

- Ukoliko navedeno samo ime datoteke, ona se pohranjuje/čita lokalno, u istoj mapi gdje je pohranjen i kod:
  - `ulaz.open("ulaznaDatoteka.dat");`
- Ukoliko želimo datoteku pohraniti/čitati iz neke druge mape, potrebno je navesti cijelu putanju:
  - `ulaz.open("C:\\podaci\\ulaznaDatoteka.dat");`

Otvaranje datoteke, parametar mode

- Osim navedenog načina otvaranja datoteke, postoje i drugi načini otvaranja datoteke
  - Npr., ukoliko se otvara datoteka za pisanje podataka, bitno je navesti ako želimo brisati postojeće podatke ukoliko oni postoje u datoteci ili se podaci nadopisuju u datoteku
- Ukoliko želimo otvoriti datoteku na neki drugi način, osim naziva datoteke, kao drugi parametar **dodaje se parametar mode**;
- Mode je varijabla tipa integer i prethodno joj se mora pridružiti vrijednost

Otvaranje datoteke, parametar mode

TABLICA s popisom načina otvaranja datoteke	
mode	**opis**
in	Otvara za input (default za ifstreams)
out	Otvara za output (default za ofstreams)
app	Dodaje podatke, uvijek piše podatke na kraj datoteke (samo za output)
ate	Traži kraj datoteke
binary	Otvara datoteku u binarnom obliku
trunc	Briše sadržaj datoteke ako ona već postoji
nocreate	Ako datoteka ne postoji, ne može otvoriti datoteku
noreplace	Ako datoteka postoji neće se otvoriti za obradu ukoliko ate ili app nisu postavljeni

Otvaranje datoteke

- Ukoliko 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).
- Parametar mode predstavlja način na koji će se datoteka otvarati (2. način):

```
int mode = ios::app;
izlaz.open("ime_datoteke.dat");
```

Greške kod otvaranja datoteke

- Otvaranje datoteke ne mora uspjeti iz nekoliko 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!

Hvatanje grešaka toka
Funkcija fail

- Pripadna funkcija fail, može se koristiti za provjeru uspjeha operacija toka
  - fail vraća logički tip (true or false)
  - fail vraća true ako operacija toka nije uspjela
- Kada funkcija toka open ne uspije, najbolje je zaustaviti program → koristi se funkcija exit

Funkcija exit

- Funkcija exit prekida rad programa
  - exit vraća parametar operacijskom sustavu
  - exit uzrokuje prekid programa
  - exit NIJE pripadna funkcija

- exit zahtijeva direktive include i using

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

Funkcije fail i exit - primjer

- Odmah nakon poziva funkcije open, provjerimo je li operacija bila uspješna:

```
ulaz.open(“podaci.dat");
if ( ulaz.fail( ) )
{
  cout<<“Otvaranje datoteke nije uspjelo.\n";
  exit(1) ;
}
```

Ulazni tok

- Kada je povezana s datotekom, varijabla ulaznog toka se može koristiti za preuzimanje ulazne vrijednosti s operatorom ekstrakcije (>>):

```
int n1, n2;
 ifstream ulaz;
 ulaz.open("ulaznaDatoteka.dat");
 ulaz >> n1 >> n2;
```

- Analogija sa čitanjem sa tipkovnice:

```
cin >> n1 >> n2;
```

- **Napomena:** ulazna datoteka mora sadržavati točno očekivane podatke; u navedenom primjeru očekuje se čitanje cijelih brojeva iz datoteke!

Izlazni tok

- Izlazni tok radi slično kao ulazni tok

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

- Analogija sa pisanjem na ekran:

```
cout<<"n1=" << n1<< "n2=" << n2;
```

- U datoteci ili na ekranu ispisuju se isti podaci, vrijednosti varijabli n1 i n2

Provjeravanje kraja datoteke

- 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 (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)

Provjeravanje kraja datoteke

- Računamo prosjek brojeva u datoteci

```
double sljedeci, suma = 0;
    int brojac = 0;
    while(ulaz >> sljedeci)
{
        suma = suma + sljedeci;
        brojac++;
    }
double prosjek = suma / brojac;
```

Provjeravanje kraja datoteke

- Pripadna funkcija eof provjerava kraj datoteke
  - Pripadna funkcija svakog ulaznog toka
  - 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 još nismo došli do kraja datoteke
    - Primjer: `if ( ! ulaz.eof( ) )`

Provjeravanje kraja datoteke

- Sljedeća petlja čita svaki znak i ispisuje ga na ekran

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

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

Provjeravanje kraja datoteke

- Kraj datoteke se označava posebnim znakom
- ulaz.eof( ) je još uvijek true nakon što je pročitan zadnji znak u datoteci
- ulaz.eof( ) postaje false kada je pročitana posebna oznaka kraja

Provjeravanje kraja datoteke

- Vidjeli smo dvije metode
  - `while ( ulaz >> sljedeci)`
  - `while ( ! ulaz.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

Zatvaranje datoteke

- Nakon uporabe datoteke, moramo je zatvoriti
  - Time se prekida veza datoteke i varijable toka
  - Zatvaramo datoteke da bi reducirali mogućnost oštećivanja datoteke ako se izvođenje programa ne završi regularno
- Važno je zatvoriti izlaznu datoteku ako vaš program kasnije treba čitati ulaz iz te izlazne datoteke

```
ulaz.close();
izlaz.close();
```

- Sustav će automatski zatvoriti datoteke ako to zaboravite učiniti, ako se izvođenje vašeg programa regularno završava