Objektno orjentisano programiranje – C++

I/O Sistemi

Teme

- Pojam i vrste toka
- Standardne C++ biblioteke
- Tekstualni i binarni tokovi
- Klase tokova
- Formatiranje ulaza i izlaza
- Zastavice (engl. flags)
- Zaglavlja (engl. headers)
- Korišćenje U/I manipulatora

I/O (ulazno-izlazni sistem)

- Postoje dve verzije I/O biblioteka:
  - starija, zasnovana na originalnoj specifikaciji jezika C++, podržana u zaglavlju <iostream.h>
  - novija, koju definiše standard jezika, podržana u zaglavlju <iostream>
- Iz ugla programera, obe biblioteke izgledaju (skoro) isto, zato što se novija zasniva na starijoj
- Koristiće se nova verzija biblioteke

Termini

- Tok (engl. stream) je tok karaktera
  - **ulazni tok** je protok karaktera u program
  - **izlazni tok** je protok karaktera van programa
  - **cin** je predefinisani ulazni tok definisan u <iostream>
  - **cout** predefinisani izlazni tok definisan u <iostream>
  - predefinisani izlazni **tok greške** (engl. error stream) definisan u <iostream>, u koji se trebaju slati sve poruke grešaka. Izbacuje poruke na terminal kao i cout
- Tokovi su definisani u biblioteci tokova u datoteci zaglavlja (engl. header file) <iostream> i implementirani su u iostream biblioteci

C++ tokovi (engl. streams)

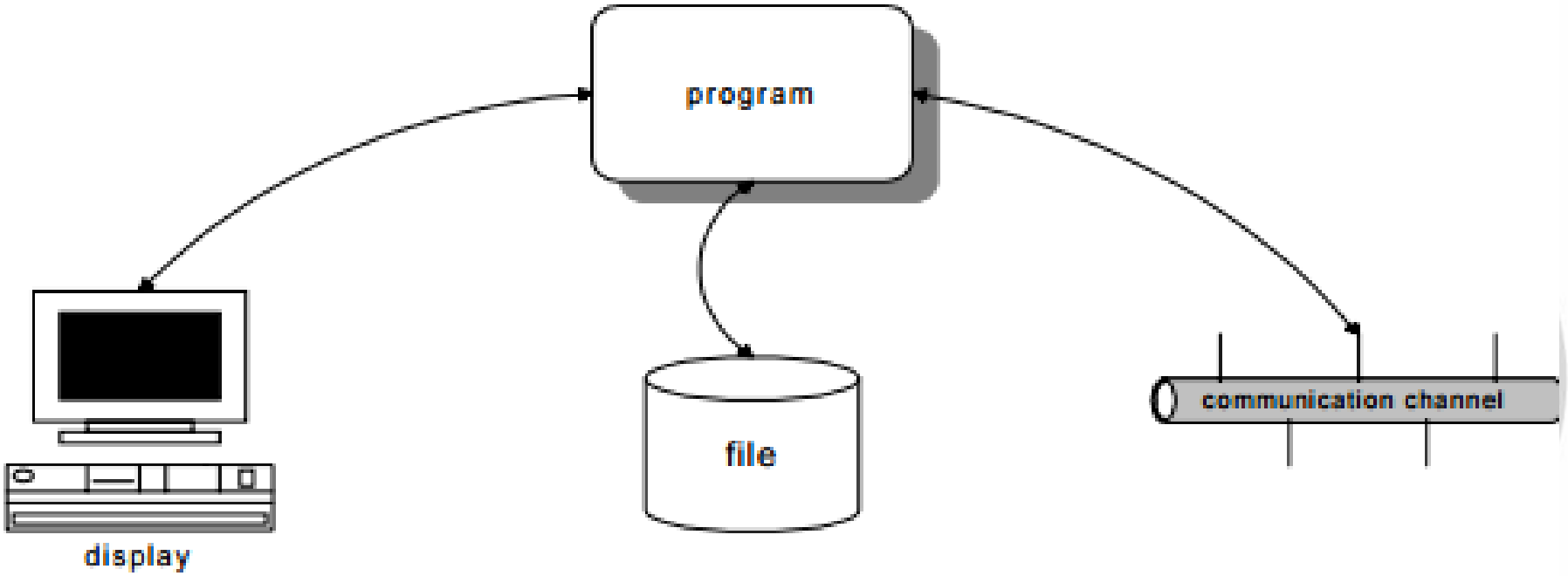
- Ulaz i izlaz podataka nije deo jezika C++ i ne postoje posebne naredbe za izvršavanje tih operacija.
- Postoje, međutim klase iz biblioteke za izvršavanje ovih operacija.
- Prenos podataka se svodi na prenos sekvence bajtova koji se nazivaju **tokovi** (engl. stream).
- Tokovi sa izvora podataka se nazivaju ulazni tokovi, a tokovi odredišta podataka izlazni tokovi.

C++ tokovi (2)

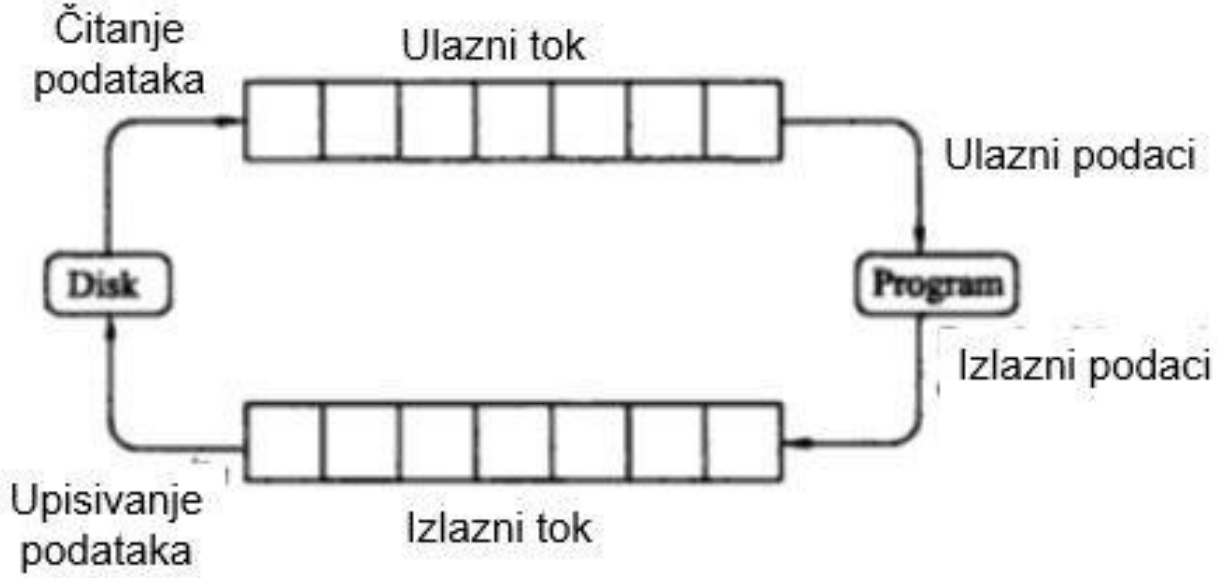
- Tok je apstrakcija koja može da proizvodi ili prima informacije
- I/O sistem jezika povezuje tokove sa stvarnim, fizičkim uređajima (monitorom, štampačem, tastaturom itd.)
- Svi tokovi se ponašaju na isti način čak i kada su povezani sa različitim fizičkim uređajima
  - na primer, isti metod se koristi za ispis na ekran i za snimanje na disk i za štampanje

C++ tokovi (3)

- I/O tokovi podržavaju prenos podataka između programa i vanjskih uređaja.



C++ tokovi (4)


Čitanje
podataka
Ulazni tok
Ulazni podaci
Disk
Program
Izlazni podaci
Upisivanje
podataka
Izlazni tok

Predefinisani tokovi

- C++ sadrži nekoliko predefinisanih tokova (globalnih objekata klasa tokova) koji se automatski otvaraju kada C++ program počne da se izvršava
  - **cin** je tok vezan za standardni ulaz
  - **cout** je tok vezan za standardni izlaz
  - **cerr** je tok vezan za standardni izlaz
  - **clog** je takođe vezan za standardni izlaz
- Razlika između tokova cerr i clog (koji služe za debug i ispis informacija o greškama) je u tome što clog koristi bafere, a cerr ne koristi bafere.

Standardni tokovi u C++ jeziku

- **cin** -- je istream_withassign klasa koja se povezuje na standardni ulaz (tipično tastatura)
- **cout** -- je ostream_withassign klasa koja se povezuje standardni izlaz (tipično monitor)
- **cerr** -- je ostream_withassign klasa koja se povezuje na standardni izlaz greške (tipično monitor), obezbeđujući nebaferisani izlaz (engl. unbuffered output)
- **clog** -- je ostream_withassign klasa koja se povezuje na standardni izlaz greške (tipično monitor), obezbeđujući baferisani izlaz (engl. buffered output)

Primer: Standardi tokovi u C++ jeziku

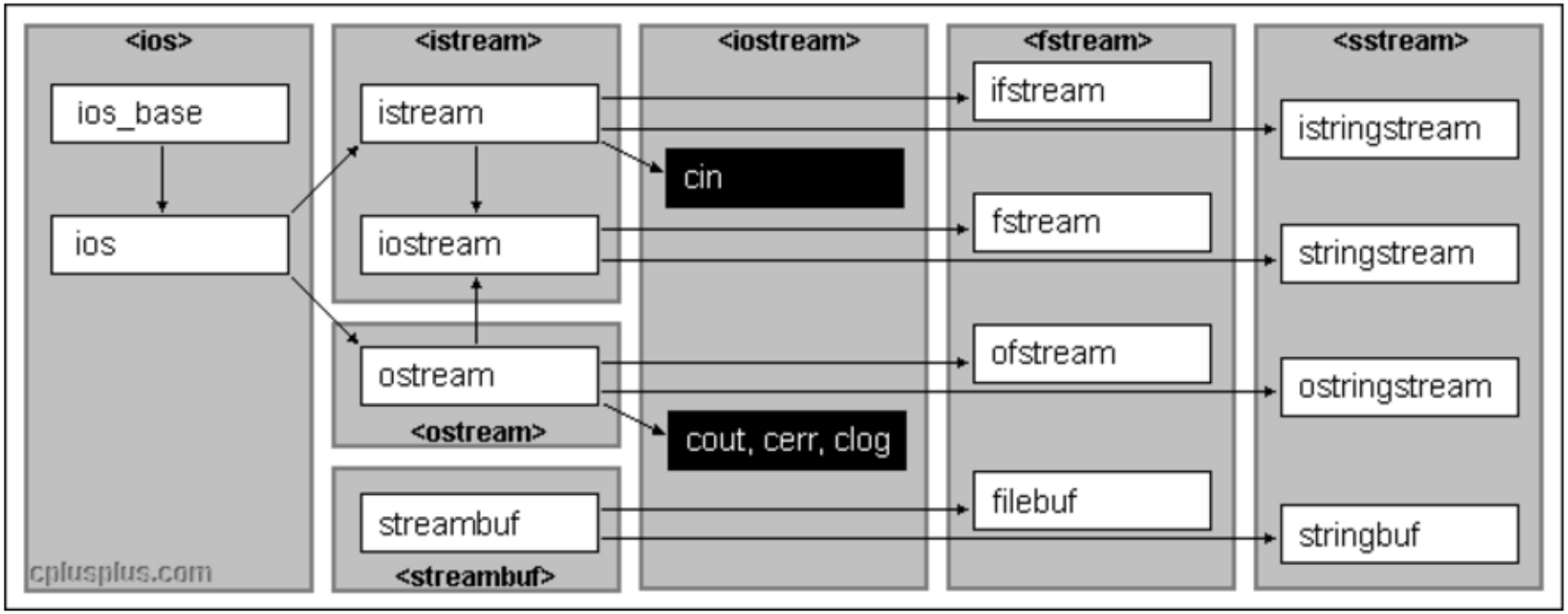
```
#include <iostream>
using namespace std;
int main() {
  cout << "Koliko imate godina: " << endl;
  int nAge;
  cin >> nAge;
  if (nAge <= 0){
      cerr << "Pogrešno unesen podatak!"<<endl;
      exit(1);
  }
  cout << "Imate " << nAge << ", godina" << endl;
  system("pause");
  return 0;
}
```

```
Koliko imate godina:
35
Imate 35, godina
```

Standardne C++ biblioteke

Tip podataka	Opis
ofstream	Ovaj tip podataka predstavlja tok izlazne datoteke i koristi se za kreiranje datoteka i za upis informacija u datoteke • cout je objekat klase ostream
ifstream	Ovaj tip podataka predstavlja tok ulazne datoteke i koristi se za čitanje informacija iz datoteka • cin je objekat klase istream
fstream	Ovaj tip podataka predstavlja opšti tok podataka (engl. the file stream) i ima mogućnosti i ofstream i ifstream (ifstream može da kreira datoteke, upisuje u njih podatke i čita iz njih podatke).

Mapa C++ I/O biblioteka tokova



Tekstualni i binarni tokovi

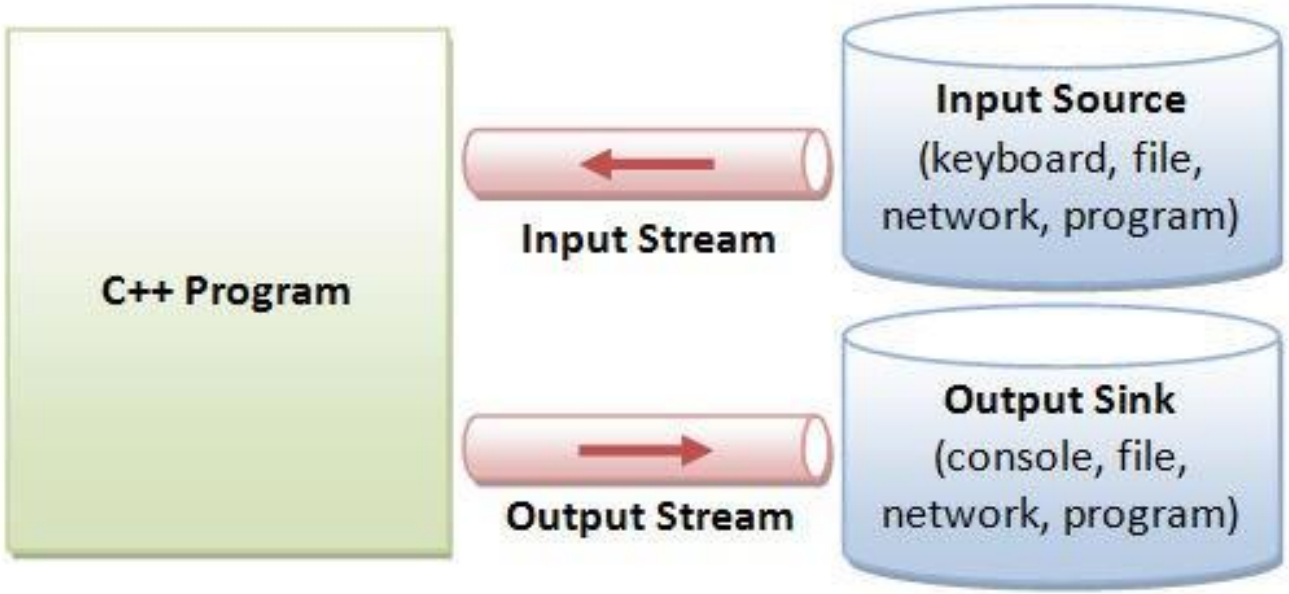
- Najčešće se tok koristi kao logički interfejs ka datoteci
- Tok se sa datotekom povezuje pomoću operacije open, od njega “razdvaja” operacijom close
- Postoje dve vrste tokova:
  - tekstualni (koriste karaktere)
  - binarni (koriste se za bilo koji tip podataka)

Klase tokova

I/O klasa toka

- streambuf: Klasa niskog nivoa koja se retko direktno koristi
- ios: Klasa sa korisnim metodama koje upravljaju ili nadgledaju operacije toka
- istream: Ulazni tok
- ostream: Izlazni tok
- iostream: Ulazno-izlazni tok
- fstream: Tok datoteke
- ifstream: Tok ulazne datoteke
- ofstream: Tok izlazne datoteke

Tokovi: Detalji



Internal Data Formats:

- Text: `char`, `wchar_t`
- `int`, `float`, `double`, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

Preklapanje I/O operatora

- Operator **<<** zove se operator umetanja (engl. insertion), jer "umeće" podatke u tok
- Operator **>>** zove se operator izdvajanja (engl. extraction) zato što odvaja podatke iz toka

- Ovo se ne odnosi na sledeći primer:

```
cout << "Koliko imate godina: " << endl;
int nAge;
cin >> nAge;
```

Preklapanje I/O operatora (2)

- U zaglavlju **<iostream>**, operatori umetanja i izdvajanja preklopljeni su za sve tipove ugrađene u jezik, a mogu se preklapati i za sve korisničke klase
- Da bi preklopljeni operatori mogli da pristupe privatnim članovima klase, treba ih definisati kao **prijateljske funkcije klase**

Preklopljeni operatori “<<“ i “>>”

```
#include <iostream>
using namespace std;i
class A {
public:
   int x,z,y;
    A () {
    }
   A (int a, int b, int c) {
       x=a;
       y=b;
       z=c;
   }
   friend istream& operator >> (istream&, A&);
   friend ostream& operator << (ostream& stream, A& objekat);
};
```

Preklopljeni operatori “<<“ i “>>” (2)

```
istream& operator >> (istream& stream, A& objekat) {
   cout<<"Unesite vrednosti koordinata x,y,z:"<<endl;
   stream>>objekat.x>>objekat.y>>objekat.z;
   return stream;
}
ostream& operator << (ostream& stream, A& objekat){
    stream<<objekat.x << ",";
    stream<<objekat.y << ",";
    stream<<objekat.z << endl;
    return stream;
}
```

Preklopljeni operatori “<<“ i “>>” (3)

```
int main () {
    A a, b, c;
    cin >>a>>b>>c;
    cout <<a<<b<<c;
    system ("pause");
    return 0;
}
```

```
Unesite vrednosti koordinata x,y,z:
1 2 3
Unesite vrednosti koordinata x,y,z:
4 5 6
Unesite vrednosti koordinata x,y,z:
7 8 9
```

Formatiranje ulaza/izlaza

- Formatiranje ulaza/izlaza može se obaviti na dva načina:
  - metodama klase ios
  - pomoću specijalnih, tzv. manipulatorskih funkcija
- Svakom toku pridružen je skup zastavica (engl. flag) za formatiranje koji precizno definišu kako će on formatirati informacije koje prikazuje
  - npr. kada je postavljena zastavica skipws, preskače se prikazivanje praznina na početku
- **Za postavljanje zastavica koristi se funkcija setf() klase ios**

Zastavice (engl. flag)

- ios::in dozvoljava ulaz (read operaciju) iz toka
- ios::out dozvoljava izlaz (write operaciju) u tok
- "|" bitna (engl. bitwise) ILI operacija koja koristi dve ios zastavice, što znači prenos ios::in | ios::out u konstruktor std::fstream omogućavajući i ulaz i izlaz u tok
- Važno:
  - std::ifstream postavlja ios::in zastavicu
  - std::ofstream postavlja ios::out zastavicu.
  - std::fstream ne postavlja ni ios::in ni ios::out automatski

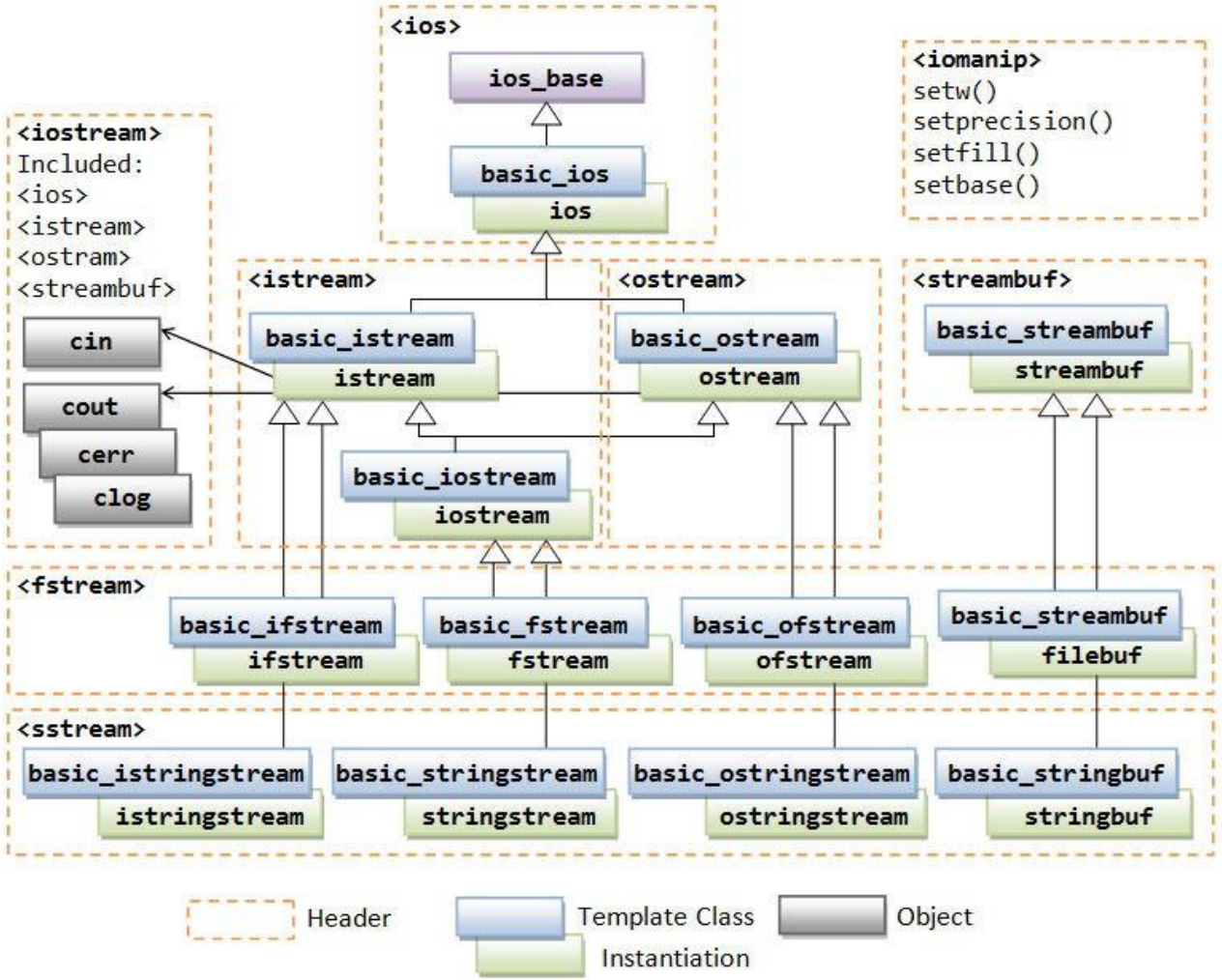
iostream zastavice (engl. flags)

Flag	*Purpose*
`showpoint`	Displays a decimal point and trailing zeros as required by precision
`showpos`	Turns on the plus sign (+) before positive numbers
`left`	Aligns output to the left
`right`	Aligns output to the right
`internal`	Aligns the sign for a number to the left and aligns the value to the right
`showpoint`	Adds trailing zeros as required by the precision
`showpos`	Displays a plus sign (+) if the number is positive
`scientific`	Shows floating-point values in scientific notation
`fixed`	Shows floating-point numbers in decimal notation
`showbase`	Adds "0x" in front of hexadecimal numbers to indicate that it is a hexadecimal value
`Uppercase`	Shows hexadecimal and scientific numbers in uppercase
`dec`	Sets the base of the numbers for display to decimal
`oct`	Sets the base of the numbers for display to octal—base 8
`hex`	Sets the base of the numbers for display to hexadecimal—base 16

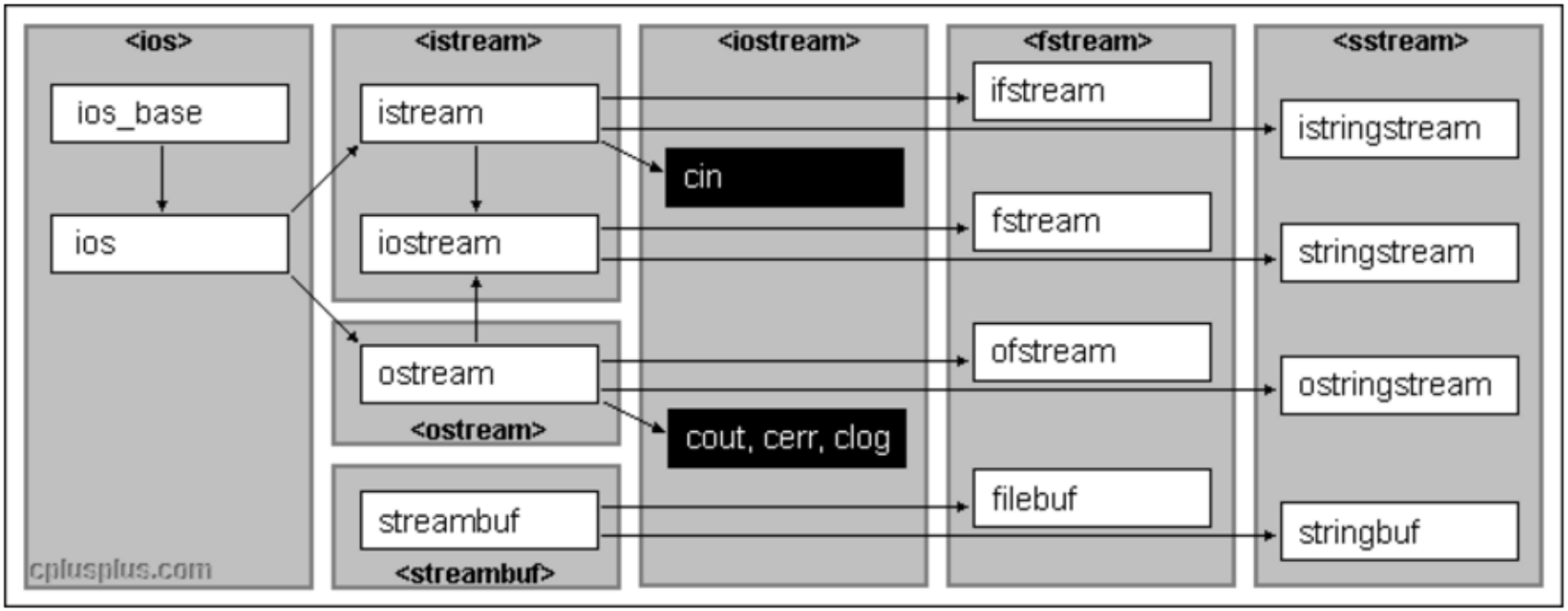
Zaglavlja (engl. headers)

- C++ I/O je obezbeđen u zaglavljima
  - <iostream> (uključuje <ios>, <istream>, <ostream> and <streambuf>),
  - <fstream> (IO za datoteke)
  - <sstream> (IO za stringove).
- Zaglavlje <iomanip> obezbeđuje manipulatore kao što su setw(), setprecision(), setfill() i setbase() za formatiranje
- setw() manipulator (in <iomanip> header) - dužina polja
- setfill() manipulator (in <iomanip> header) - popunjavanja karakterima
- left|right|internal manipulator (in <iostream> header) - poravnanje teksta

Zaglavlja (2)


<ios>
ios_base
basic_ios
ios
<iomanip>
setw()
setprecision()
setfill()
setbase()
<iostream>
Included:
<ios>
<istream>
<ostram>
<streambuf>
cin
cout
cerr
clog
<istream>
basic_istream
istream
<ostream>
basic_ostream
ostream
basic_iostream
iostream
<streambuf>
basic_streambuf
streambuf
<fstream>
basic_ifstream
ifstream
basic_fstream
fstream
basic_ofstream
ofstream
basic_streambuf
filebuf
<sstream>
basic_istringstream
istringstream
basic_stringstream
stringstream
basic_ostringstream
ostringstream
basic_stringbuf
stringbuf
Header
Template Class
Instantiation
Object

<ios> I/O osnovne klase



Funkcija
<ios> <iostream> std::showpos

ios_base& showpos (ios_base& str);

- Prikazuje plus (+) simbol
- Postavlja showpos format zastavicu za str stream.
- Kada je showpos zastavica format postavljena, plus simbol (+) predhodi svakoj ne-negativnoj numeričkoj vrednosti koja se ubacuje u stream (uključujući i nule)

Funkcija

<ios> <iostream> std::showpos (2)

```
#include <iostream>     // std::cout, std::showpos,
                        //std::noshowpos
using namespace std;
int main () {
  int p = 1;
  int z = 0;
  int n = -1;
  std::cout << std::showpos    << p << '\t' << z << '\t' << n
            <<endl;
  std::cout << std::noshowpos << p << '\t' << z << '\t' << n
            <<endl;
  cin.get();
  return 0;
}
```

```
+1    +0    -1
1     0     -1
```

Funkcija

<ios> <iostream> std::showpos (3)

```
#include <iostream>
using namespace std;
int main () {
  cout.setf (ios::showpos);
  cout.setf (ios::scientific);
  cout<< 123<< endl <<123.3<<endl;
  cin.get();
  return 0;
}
```

```
+123
+1.233000e+002
```

Funkcija
<ios> <iostream> std::showpos (4)

```
#include <fstream>
#include <iostream>
#include <cstdlib> // for exit()
using namespace std;
int main(){
  // aktivni ios::showpos i ios::uppercase flag
  cout.setf (ios::showpos | ios::uppercase);
  cout << 27 << endl;
  cout.unsetf(ios::dec); //aktivan decimalni izlaz
  cout.setf (ios::hex, ios::basefield); //aktivan hexizlaz
  cout << 27 << endl;
  cin.get();
  return 0;
}
```

```
+27
1B
```

Funkcija
<ios> <iostream>std::scientific

ios_base& scientific (ios_base& str);

- Koristiti „scientific“ notaciju za brojeve sa pomičnom tačkom

Funkcija <ios> <iostream>std::ios_base::setf

- Postavljanje i brisanje zastavica za formatiranje
- Da bi se zastavica postavila, koristi se funkcija setf () koja je članica klase ios.
- Na primer, da bi se postavio fleg showpos, koristi se naredba:

```
cout.setf (ios::showpos);
```

Funkcija
<ios> <iostream>std::scientific (2)

```
#include <iostream>
using namespace std;
int main () {
  cout.setf (ios::showpos);
  cout.setf (ios::scientific);
  cout<< 123<< endl <<123.3<<endl;
  cout.precision (2);
  cout.width (10);
  cout.fill ('#');
  cout<<123<<endl;
  cout.width (10);
  cout << 123.23456;
  cin.get();   return 0;
}
```

```
+123
+1.233000e+002
#######+123
+1.23e+002
```

Funkcija
<ios> <iostream>std::scientific (3)

```
#include <iostream>
int main () {
   double a = 3.1415926534;
   double b = 2006.0;
   double c = 1.0e-10;
   std::cout.precision(5);
   std::cout << "default: " << endl;
   std::cout << a << '\n' << b << '\n' << c << endl;
   std::cout << << endl;;
   std::cout << "scientific:\n" << std::scientific;
   std::cout << a << '\n' << b << '\n' << c << << endl;
   cin.get();
   return 0;
}
```

default:
3.1416
2006
1e-010

scientific:
3.14159e+000
2.00600e+003
1.00000e-010

Funkcija
<ios> <iostream> std::left

```
#include <iostream>  // std::cout, std::internal, std::left, std::right
int main () {
   int n = -77;
   std::cout.width(6);
   std::cout << std::internal << n <<endl;
   std::cout.width(6);
   std::cout << std::left << n << endl;
   std::cout.width(6);
   std::cout << std::right << n << endl;
   cin.get();
  return 0;
}
```

```
-   77
-77
   -77
```

Funkcija
<iomanip> std::showbase

```
#include <iostream>     // std::cout, std::showbase,
                        //std::noshowbase
using namespace std;
int main () {
  int n = 20;
  std::cout << std::hex << std::showbase << n << endl;
  std::cout << std::hex << std::noshowbase << n << endl;
  cin.get();
  return 0;
}
```

```
0x14
14
```

Korišćenje I/O manipulatora

- I/O sistem jezika C++ omogućuje formatiranje parametara toka i pomoću specijalnih funkcija koje se zovu manipulatori
- Da bi se manipulatori sa argumentima koristili u programu, potrebno je uključiti zaglavlje <iomanip>

Zaglavlje iomanip

- Dio I/O biblioteke koja pripada C++Standard Library.
- Definiše manipulatorske funkcije resetiosflags(), setiosflags(), setbase(), setfill(), setprecision() i setw().
- Ove funkcije menjaju stanje iostream objekata.

Zaglavlje <iomanip>

header
<iomanip>

IO Manipulators

Header providing parametric manipulators:

fx Parametric manipulators

setiosflags	Set format flags (function)
resetiosflags	Reset format flags (function)
setbase	Set basefield flag (function)
setfill	Set fill character (function)
setprecision	Set decimal precision (function)
setw	Set field width (function)
get_money C++11	Get monetary value (function)
put_money C++11	Put monetary value (function)
get_time C++11	Get date and time (function)
put_time C++11	Put date and time (function)

Funkcija
<iomanip> std::setiosflags

```
#include <iostream>       // std::cout, std::hex, std::endl
#include <iomanip>        // std::setiosflags
using namespace std;
int main () {
    std::cout << std::hex;
    std::cout <<
    std::setiosflags (std::ios::showbase | std::ios::uppercase);
    std::cout << 100 << std::endl;
    std::cout << std::oct;
    std::cout << 100 << std::endl;
    cin.get();
    return 0;
}
```

```
0X64
0144
```

Primer manipulatorske funkcije

```
#include <iostream>
#include <iomanip>
using namespace std;
ostream &setup (ostream & stream) {
   stream.setf (ios::left);
   stream <<setw(10)<<setfill ('$');
   return stream;
}
int main () {
   cout<<10<<" "<<setup<<10<<endl;
   cin.get();
   return 0;
}
```

```
10 10$$$$$$$$
```

Neki I/O manipulatori

Manipulator	Svrha	Input/Output
boolalpha	Uključuje fleg boolalpha	Input/output
dec	Uključuje fleg dec	input/output
fixed	Uključuje fleg fixed	output
flush	Prazni tok	output
endl	Prelazi u novi red i prazni tok	output
left	Uključuje fleg left	output
noshowpos	Isključuje fleg showpos	output
right	Uključuje fleg right	output
setbase(int base)	Postavlja brojni sistem na base	input/output
setfill(int ch)	Postavlja znak za popunu na ch	output
setprecision(int p)	Postavlja broj znakova za preciznost	output
setw(int w)	Postavlja širinu polja na w	output

Funkcija
<iomanip> std::setprecision

```
#include <iostream>     // std::cout, std::fixed
#include <iomanip>      // std::setprecision
int main () {
  double f =3.141592345;
  std::cout << std::setprecision(5) << f << endl;
  std::cout << std::setprecision(9) << f << endl;
  std::cout << std::fixed;
  std::cout << std::setprecision (5) << f << endl;
  std::cout << std::setprecision (9) << f << endl;
  std::cout << std::setprecision (3) << f << endl;
  return 0;
}
```

```
3.1416
3.14159234
3.14159
3.141592345
3.142
```

Korišćenje I/O manipulatora

```
#include <iostream>
#include <iomanip>
using namespace std;
int main () {
   cout<<setprecision (2)<<1000.243<<endl;
   cout<< setw (30) << "Hello world!"<<endl;
   cin.get();
   return 0;
}
```

```
1e+003
                    Hello world!
```

Funkcija
<ios> <iostream>:std::skipws

- Izbjeći bijeli prostor

```
#include <iostream>
#include <iomanip>
using namespace std;
int main () {
    //preskoci praznine na početku
    char c[80];
    cin>>skipws >>c;
    cout<<c<<endl;
    system ("pause");
    return 0;
}
```

```
                                    abc
abc
Press any key to continue . . .
```

Rad sa datotekama

- Uključiti zaglavlje <fstream> u kome je definisano nekoliko važnih klasa
  - datoteka se otvara da bi se otvorio ulazni tok, mora se deklarisati kao objekat klase ifstream
  - izlazni tok se deklariše kao objekat klase ofstream
  - ulazno-izlazni tok se deklariše kao objekat klase fstream
- Sve tri klase imaju metodu open kojom se tako napravljen tok povezuje sa fajlom

Binarne ili tekstualne datoteke?

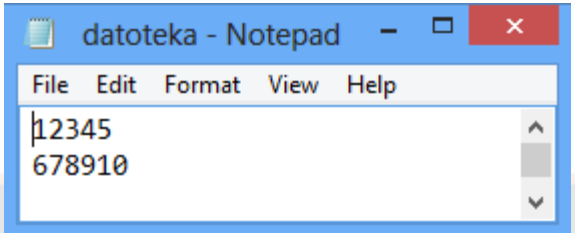
- Svaka datoteka, bez obzira kakva je, može biti otvorena u tekstualnom ili binarnom režimu
  - kada čita brojeve iz tekstualne datoteke, računar prvo mora da konvertuje znakove u binarni kod, što usporava program
  - binarne datoteke ne zahtevaju konverziju, zauzimaju manje mesta od tekstualnih, ali se ne mogu direktno štampati na monitoru ili štampaču

Tekstualne datoteke

- Tekstualne datoteke služe za čuvanje teksta tako da sve vrednosti koje su ulazi ili izlazi od/do datoteka mogu biti formatirane.

Tekstualne datoteke (2)

```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
  ofstream mojaDatoteka ("datoteka.txt");
  if (mojaDatoteka.is_open())   {
  mojaDatoteka << 12345<<endl;;
  mojaDatoteka << 678910<<endl;
  mojaDatoteka.close();
  }
  else cout << "Nije moguce otvoriti datoteku";
  cin.get();
  return 0;
}
```



Čitanje tekstualne datoteke

- Kada čitamo podatke iz datoteke, ona mora već da postoji
- Prilikom čitanja treba proveravati da li se stiglo do kraja (engl. End Of File- EOF)
  - funkcija članica good() klase ifstream vraća false ako smo stigli do kraja fajla
- Za čitanje podataka može se koristiti operator >> (nasleđen iz klase istream) koji prekida čitanje kada naiđe na whitespace, ili funkciju getline() koja učitava datoteku red po red

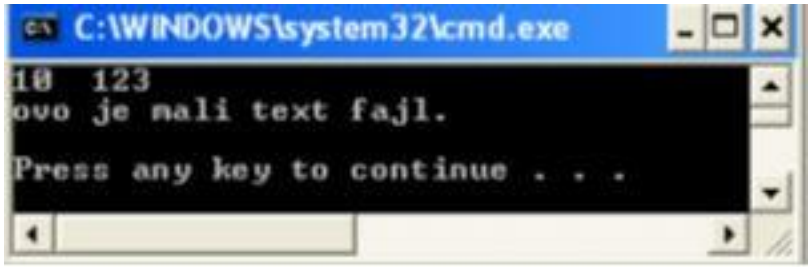
Čitanje tekstualne datoteke (2)

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main () {
  string line;
  ifstream mojaDatoteka ("Datoteka.txt");
  if (mojaDatoteka.is_open())   {
      while (getline (mojaDatoteka,line)) {
          cout << line << '\n';
       }
       mojaDatoteka.close();
  }
  else cout << "Unable to open file";
  cin.get();
  return 0;
}
```

```
12345
678910
```

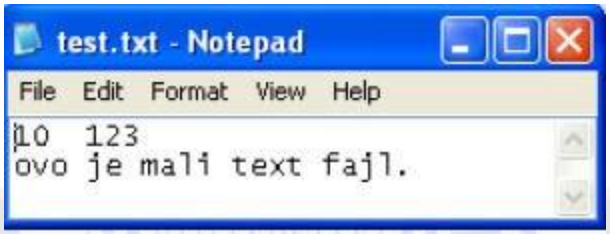
Čitanje tekstualne datoteke (3)

```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
  char bafer [100];
  ifstream infile ("test.txt");
  if (!infile) {
     cerr<<endl;
     cerr<<"Greska:datoteka se ne može otvoriti."<<endl;
     return 1;
  }
  while (!infile.eof()) {
     infile.getline (bafer, 100);
     cout << bafer <<endl;
  }
  infile.close();
  cin.get();
  return 0;
}
```



Upisivanje u tekstualnu datoteku

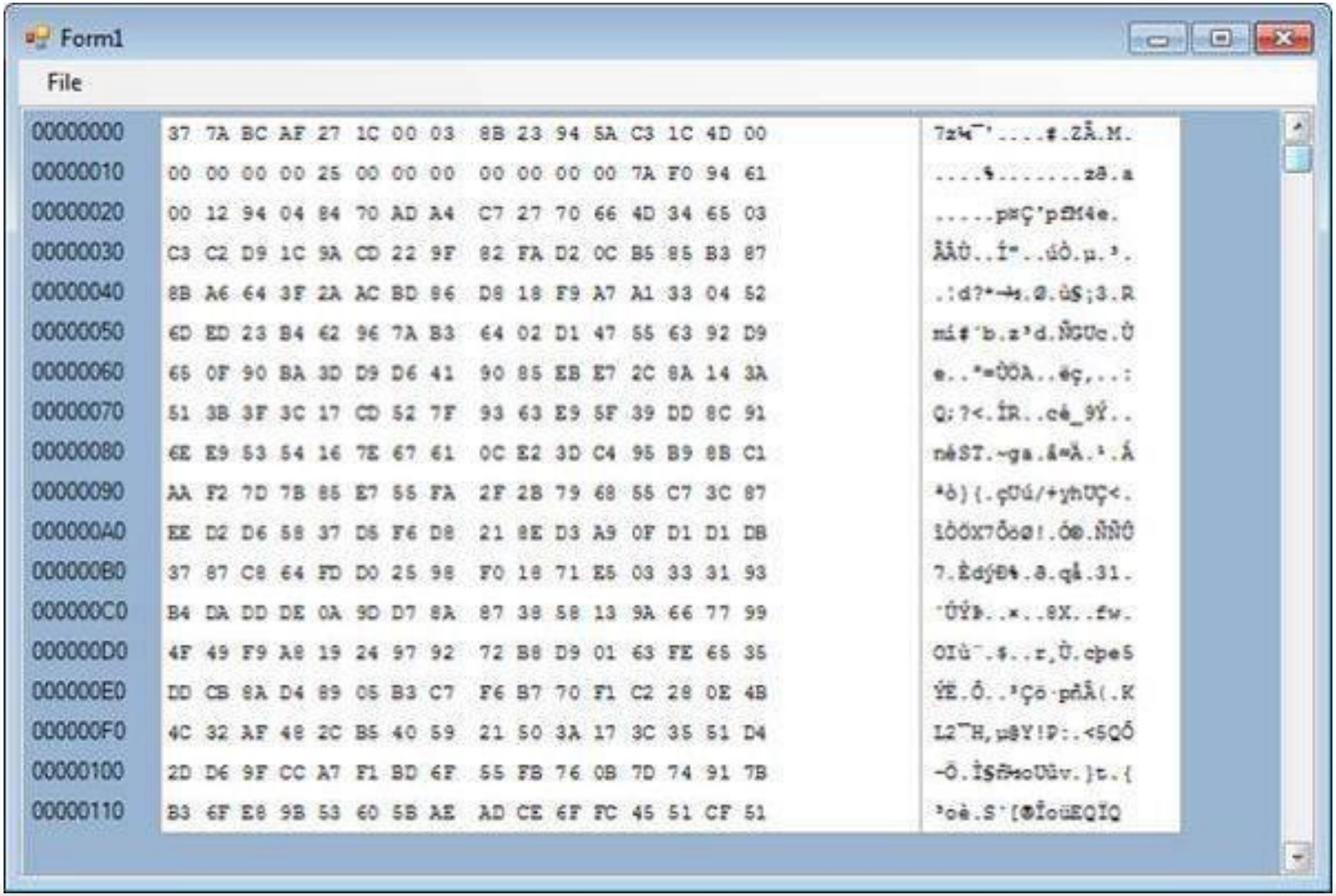
```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
  ofstream out("test.txt");
  if (!out) {
    cout<<"datoteka se ne može kreirati"<<endl;
    return 1;
  }
  out<<10<<"  "<<123<<endl;
  out<<"ovo je mali text fajl."<<endl;
  out.close();
  cin.get();
  return 0;
}
```



Binarne datoteke

- Binarna datoteka je datoteka računara koja nije tekst datoteka.
- Ovde operacije čitanja i pisanja podataka sa operatorima izvlačenja i ubacivanja (<< i >>) i funkcijama kao što su getline () nisu efikasne budući da ne treba formatirati bilo kakve podatke i podaci najverovatnije nisu formatirani u redovima.

Primer binarne datoteke (koristeći ByteViewer)



Funkcije tokova datoteka

- Tokovi datoteka uključuju funkcije članice za posebno kreiranje za čitanje i upisivanje u binarne datoteke: write i read.
- Objekti klase fstream imaju obe funkcije. Njihovi prototipovi su dati kao:

  write (memory_block, size); - Funkcija članica ostream (nasleđeno od ofstream).

  read (memory_block, size); - Funkcija članica istream (nasleđeno od ifstream).

Otvaranje datoteke

- open (ime datoteke, režim rada);

ios::in	Otvaranje za ulazne operacije
ios::out	Otvaranje za izlazne operacije
ios::binary	Otvaranje u binarnom režimu
ios::ate	Postaviti početnu tačku na kraj datoteke. Ako ova zastavica (engl. flag) nije postavljena, postaviti početnu tačku na početak datoteke.
ios::app	Sve izlazne operacije se izvršavaju na kraju datoteke, dodavanjem trenutnog sadržaja datoteci.
ios::trunc	Ako je datoteka otvorena za izlazne operacije i ako već postoji, briše se njen predhodni sadržaj i zamenjuje novim sadržajem.

Metode otvaranja datoteka

- Prvi argument metode open za sve file I/O klase je ime i lokacija datoteke koja treba da se otvori.
- Može se navesti i relativna i apsolutna putanja do datoteke, korišćenje relativne putanje se preporučuje.

```
#include <fstream>
#include <iostream>
using namespace std;
int main () {
   ofstream outfile;
   outfile.open ("studenti.dat");
   outfile.open ("c:\\fakultet\casovi\studenti.dat"); //aps. putanja
   cin.get();
   return 0;
}
```

Metode otvaranja datoteka (2)

- Drugi argument funkcije članice open je režim (mode) u kome datoteka treba da bude otvorena

Mode	Svrha
ios::app	Append (sadržaj se dodaje na kraj postojeće datoteke)
ios::ate	Ako fajl već postoji, program se pomera direktno na njen kraj. Može se upisivati bilo gde u fajl (ovaj režim se obično koristi sa binarnim fajlovima)
ios::binary	Binary. Sadržaj se upisuje u fajl u binarnom obliku, a ne u tekstualnom (koji je default)
ios::in	Input. Sadržaj se čita iz fajla. Ako fajl ne postoji, neće biti napravljen.
ios::out	Output. Sadržaj se upisuje u fajl, a ako on već ima sadržaj, prepisuje se.
ios::trunc	Truncate. Ako fajl već postoji, njegov sadržaj će biti prepisan (default režim za ios::out)

Kombinovanje načina rada

- Prilikom otvaranja datoteke flegovi se mogu kombinovati
- Za kombinovanje se koristi bitni (engl. bitwise) operator OR (|)
- Na primer, sledeće naredbe otvaraju datoteku u binarnom načinu rada i dopisuju sadržaj u datoteku umesto da ga prepisuju:

```
ofstream outfile;
outfile.open (“studenti.dat“, ios::binary | ios::app)
```

Provera uspješnog otvaranja datoteke

Pre pristupanja fajlu uvek treba proveriti rezultat metode open

```
ofstream outfile;
output.open ("studenti.dat");
if (!outfile) {
  cout<<"ne može se otvoriti datoteka"<<endl;
}
//isto što i
if (!outfile.is_open()){
  cout<<"datoteka nije otvorena!"<<endl;
}
```

Zatvaranje datoteke

- Za rad sa datotekama retko se direktno koristi funkcija open, već se poziva konstruktor klase
- Datoteka se zatvara metodom close() koja nema argumenata, niti povratni tip.

```
ifstream infile (“student.dat“);
infile.close();
```

Primer otvaranja i zatvaranja datoteke

Otvaranje datoteke

```
ofstream mojaDatoteka;
mojaDatoteka.open ("primer.bin", ios::out | ios::app
| ios::binary);
```

klasa	podrazumevani parametar režima rada
ofstream	ios::out
ifstream	ios::in
fstream	ios::in | ios::out

Zatvaranje datoteke

```
mojaDatoteka.close();
```

Primer čitanja i upisivanja u datoteku

```
#include <fstream>
#include <iostream>
using namespace std;
int main (){
   char data[100];
   ofstream outfile;
   outfile.open ("Datoteka.dat");
   cout << "Upisati u datoteku "
        << endl;
   cout  <<  "Uneti  ime:  ";
   cin.getline(data,   100);
   outfile << data << endl;
   cout << "Uneti godinu  rodjenja: ";
   cin >> data;
   cin.ignore();
   outfile << data << endl;
   outfile.close();
   ifstream infile;
   infile.open("Datoteka.dat");
   cout << "Citanje iz datoteke"
        << endl;
   infile >> data;
   cout << data << endl;
   infile >> data;
   cout << data << endl;
   infile.close();
   cin.get();
   return 0;
}
```

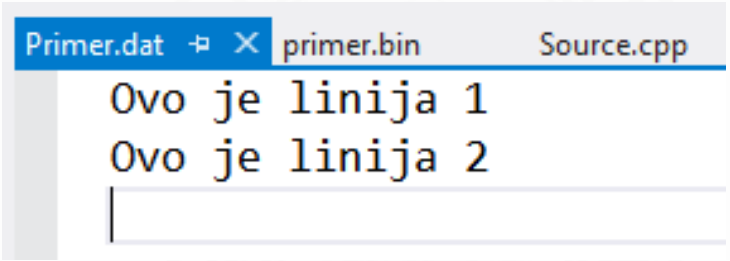
Primer čitanja i upisivanja u datoteku (2)

Rezultat

Upisati u datoteku
Uneti ime: Petar Petrovic
Uneti godinu rodjenja: 1951
Citanje iz datoteke
Petar
Petrovic

Izlaz u datoteku

```
#include <fstream>
#include <iostream>
#include <cstdlib> // for exit()
using namespace std;
int main(){
    using namespace std;
    ofstream outf ("Primer.dat");
    if (!outf)     {
     cerr << "Primer.dat se ne može otvoriti za upisivanje!"<< endl;
        exit(1);
    }
    outf << "Ovo je linija 1" << endl;
    outf << "Ovo je linija 2" << endl;
    cin.get();
    return 0;
}
```



Operator ekstrakcije (engl. extraction operator)

```
char buf[10];
cin >> buf;
```

setw (iomanip zaglavlje) ograničava broj karaktera učitanih iz toka

```
#include <iomanip.h>
char buf[10];
cin >> setw(10) >> buf;
```

Pozicioniranje u toku - get() i put ()

- Svi objekti I/O tokova imaju barem jednu internu poziciju u datoteci:
  - ifstream, na primer istream, sadrži internu get pozicuju lokacije elementa koji treba da se očita u sledećoj read operaciji
  - ofstream, na primer ostream, sadrži internu put pozicuju lokacije elementa gde sledeći element treba da bude upisan
  - fstream, sadrži i get i put pozicije kao na primer iostream.

Funkcije
tellg(), tellp(), seekg() i seekp()

tellg() and tellp()

•Vraćaju tip streampos, odnosno tip reprezentacije tekuće get pozicije (tellg) ili put pozicije (tellp).

seekg() and seekp()

- Dozvoljavaju promenu lokacija get i put pozicija.
- Prototipovi

seekg (position);
seekp (position);

Funkcija seekg()

```
inf.seekg(14, ios::cur);     // pomeranje naprijed 14 bajta
inf.seekg(-18, ios::cur);    // pomeranje nazad 18 bajta
inf.seekg(22, ios::beg);     // pomeranje do 22-og bajta
inf.seekg(24); // pomeranje do 24-og bajta u datoteci
inf.seekg(-28, ios::end); // pomeranje do 28-mog bajta
                          // pre kraja datoteke
inf.seekg(0, ios::beg); // pomeranje na početak datoteke
inf.seekg(0, ios::end); // pomeranje na kraj datoteke
```

Primeri

seekg(0); seekg(0,ios::beg); //get pokazivač na početak

seekg(5,ios::beg); //get pokazivač 5 karaktera napred od početka

tellp(); tellg() //vraća trenutnu vrednost put/get pokazivača

seekp(-10,ios::end); //put pokazivač na 10 karaktera pre kraja

seekp(1,ios::cur); //produžiti do sledećeg karaktera

Zaključak

- I/O stream (tok): cin, cout, cerr, clog
- Standardne C++ biblioteke: ofstream, ifstream, fstream
- I/O biblioteke tokova: ios, istream, ostream, iostream, fstream, sstream
- Vrste tokova: tekstualni (koriste karaktere) i binarni (koriste se za bilo koji tip podataka)
- I/O klase tokova: streambuf, ios, istream, ostream, iostream, fstream, ifstream, ofstream
- Formatiranje ulaza/izlaza:
  - metodama klase ios – setf ()
  - pomoću manipulatorskih funkcija - <iomanip>

Kraj prezentacije

HVALA NA PAŽNJI!