Ulazno / izlazni tokovi

Ulazno/Izlazni – Tokovi

- Nasledje iz C-a: Ne postoji ugradjena naredba tj. operacija za ulaz i izlaz.
- Sve operacije za ulaz/izlaz su realizovane kao bibliotečke funkcije.
- C – standardne funkcije za ulaz-izlaz (u <stdio.h>):
  - `int printf( char* format, …)`
  - `int scanf(char* format, …)`
- C++ :
  - <stdio.h>
    - `int printf(const char* format, …)`
    - `int scanf(const char* format, …)`
  - biblioteka klasa za objektno orijentisanu realizaciju ulaza i izlaza (efikasniji način)

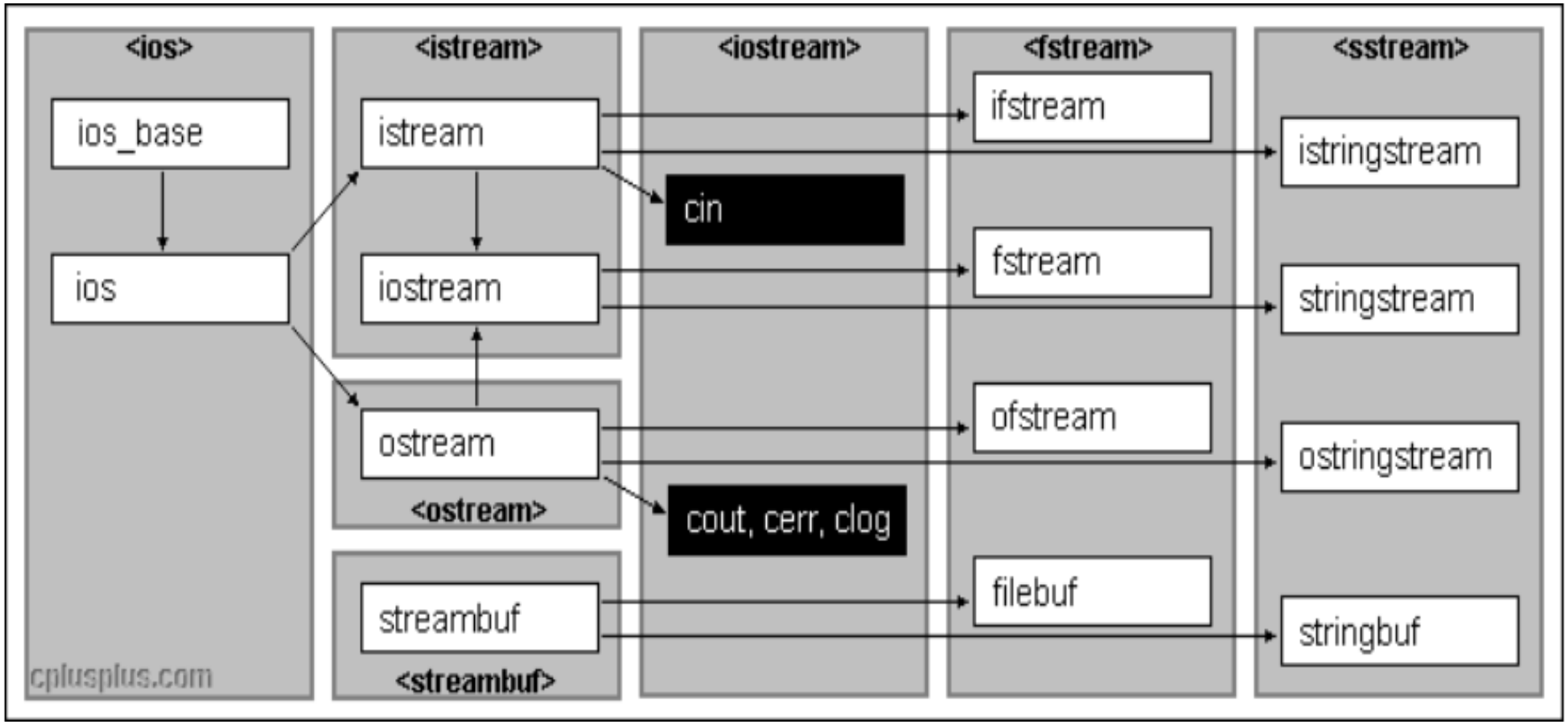
Tokovi

- Tok (stream) – logički koncept koji predstavlja sekvencijalni ulaz ili izlaz znakova (bajtova) na neki uredjaj ili datoteku.
- Datoteke u C++-u su samo dugački nizovi bajtova i nazivaju se tokovima. Ne postoji suštinska razlika između toka na disku (datoteke) i toka u operativnoj memoriji.
- Rad sa tokovima u C++ se realizuje odgovarajućim klasama. Konkretni tokovi su instance (objekti) tih klasa.
- Većina operacija nad tokovima je ista, bez obzira gde su oni smešteni.
- Osnovna klasa za U/I je ios (u hederu <iostream>) i ona je koren hijerarhije klasa za UI

UI klase

- Dve osnovne klase koje se izvode iz ios **(<iostream>)**
  - **istream** - ulaz
  - **ostream – izlaz**
- Ove dve klase su osnovne klase za izvedenu klasu iostream
- Dve najvažnije klase koje se izvode iz klase **iostream** su:
  - **fstream** - za rad sa datotekama
  - **stringstream** - za rad sa nizovima (eng. *strings*, tj. tokovima u operativnoj memoriji)
- Iz **istream** se izvode
  - ifstream – ulaz iz datoteke
  - istringstream - za uzimanje podataka iz tokova
- Iz **ostream** se izvode
  - ofstream – izlaz u datoteku
  - ostringstream - za smeštanje podataka u tokove
- Klase za rad sa datotekama su u **<fstream>**
- Klase za rad sa stringovima (tokovi u OM) su u **<sstream>**

Hijerarhija klasa za realizaciju ulazno-izlaznih operacija



Standardni tokovi

- Postoje 4 standardna toka (globalni statički objekti) :
  - **cin** – glavni (standardni) ulaz tipa istream. Standardno predstavlja tastaturu, ukoliko se drugačije ne specificira (da se izvrši skretanje glavnog ulaza unutar samog programa ili u komandi operativnog sistema za izvršavanje programa).
  - **cout** – glavni (standardni) izlaz tipa ostream. Predstavlja ekran, koristi se za ispisivanje podataka koji čine rezultate izvršavanja programa.
  - **cerr** – standardni izlaz za poruke tipa ostream. Predstavlja ekran, obično se koristi za ispisivanje poruka o greškama.
  - **clog** – standardni izlaz za zabeleške tipa ostream. Predstavlja ekran, koristi se za “vođenje evidencije” o događajima za vreme izvršenja programa.

UI nasleđe iz C-a

- Mogu da se koriste i UI funkcije iz C-a (nalaze se u <cstdio> odnosno <stdio.h>
- Klase za UI koje nudi C++ su efikasnije i preporučuju se umesto UI funkcija iz C-a.
- Nikako se ne preporučuje da se koriste obe vrste ulazno-izlaznih biblioteka tj. UI funkcije i iz C-a i iz C++

- Klasa istream ima po jednu funkciju članicu **`operator>>`**
za sve ugradjene tipove koja služi za ulaz podataka

```
istream& istream::operator>> (tip &T)
```

`tip` - je ugradjeni tip objekta koji se čita

-Klasa ostream ima po jednu funkciju članicu **`operator<<`**
za sve ugradjene tipove koja služi za izlaz podataka

```
ostream& ostream::operator<< (tip T)
```

`tip` - je ugradjeni tip objekta koji se ispisuje

Klase za ulazne tokove

- Tri najvažnije klase za ulazne tokove su :
  - **istream**,
  - **ifstream**,
  - **istringstream**.
- Klasa **istream** je najbolja za rad sa sekvencijalnim tekstualnim ulazom.
- Klasa **ifstream** podržava ulaz iz datoteke.

Konstruisanje objekata ulaznih tokova

- **Konstruisanje objekata ulaznih tokova**
  - Ako se koristi **cin** objekat, ne treba da se napravi ulazni tok.
  - Ulazni tok se mora napraviti ukoliko se koristi:
    - tok iz datoteke (file stream)
    - tok iz niza (string stream)

Konstruktori ulaznih tokova datoteka

- Kreiranje objekta ulaznog toka:
  1. korišćenjem konstruktora bez argumenata (podrazumevani konstruktor **ifstream()**)
  2. konstruktor sa argumentima tj. navođenjem naziva datoteke i odgovarajućih parametara

Konstruktor bez argumenata

- Korišćenjem konstruktora bez argumenata kreira se objekat klase **ifstream**, a zatim se poziva funkcija **open** koja otvara navedenu datoteku:

```
ifstream f;
f.open ( "imedatoteke.txt", iosmod);
```

- ili

```
ifstream* f = new ifstream;
f->open("imedatoteke.txt", iosmode);
```

Konstruktor sa argumentima

- Navođenje naziva datoteke i odgovarajućih parametara (mode flags) se vrši na sledeći način:
- **`ifstream(const char* imedatoteke, int iosmode=ios::in);`**

Primer:

```
ifstream f("imedatoteke.txt", iosmode);
```

Parametar **iosmode** može da uzme sledeće vrednosti:

ios::app	upisivanje na kraj datoteke
ios::ate	pozicioniranje na kraj datoteke posle otvaranja
ios::in	otvara ulaznu datoteku
ios::out	otvara izlaznu datoteku
ios::nocreate	otvara datoteku ukoliko već postoji
ios::noreplace	otvara datoteku ukoliko već ne postoji
ios::trunc	otvara datoteku i briše stari sadržaj
ios::binary	binarna datoteka. Ako se ništa ne kaže, podrazumeva se rad sa tekstualnom datotekom.

Tokovi u memoriji (OM)

- Tokovi u memoriji (OM) omogućavaju i jednostavnu konverziju iz binarnog u tekstualni oblik i obrnuto
- Sadržaj tokova u memoriji se prikazuje pomoću objekta tipa `string`
- Metode za pristup sadržaju
  - **`string& str();`** - vrednost je sadržaj toka za koji je pozvana ova metoda
  - **`void str (const string& s);`** - postavlja vrednost kopije argumenta `s` kao novi sadržaj toka za koji je pozvana
- Primer:

```
nekitok.str("Neki tekst koji ćemo postaviti u tok");
string sadrzaj = ulaznitok.str();
```

Konstruktori tokova u memoriji

- Konstruktori za tokove u memoriji
  - `explicit stringstream (int mode=ios::in|ios::out)`
  - `explicit ostringstream (int mode=ios::out)`
  - `explicit istringstream (int mode=ios::in)`
  - `explicit stringstream (const string& s, int mode=ios::in|ios::out)`
  - `explicit ostringstream (const string& s, int mode=ios::out)`
  - `explicit istringstream (const string& s, int mode=ios::in)`
- Primeri:
  - `stringstream strprazantok;  // prazan objekat`
  - `istringstream ulstr("Ulazni tekst");  //čitanje`
  - `ostringstream izstr(ios::out|ios::app); //dopisivanje`

Konstruktori ulaznih tokova u memoriji

- Konstruktor ulaznog toka niza zahteva adresu prethodno alocirane i inicijalizovane memorije:

```
char s[] = "23.12";
double broj;
istringstream myString( s );
myString >> broj; // broj = 23.12
```

Operacije ulaznog toka

- Operator ekstrakcije (>>) je programiran za sve standardne C++ tipove podataka, i predstavlja najlakši način da se preuzmu bajtovi iz objekta ulaznog toka.

```
char ime[20];
cin >> ime;
```

- Operator >> pamti ulazne podatke samo do prvog unetog blanko znaka.

Metode **get i getline**

- Metoda **get** se ponaša kao i operator >>, osim što pamti i blanko znakove. Postoji više prototipova ove funkcije. Najčešće korišćeni su:

- `istream &get(char& znak);` - uzima sledeći znak iz ulaznog toka i smešta ga u promenljivu znak (npr. **`cin.get(c)`**).

- `istream &get(char* niz, int max);` - čita max broj znakova (ukoliko postoji toliko znakova) i smešta ih u niz (npr. **`cin.get(ime, 20)`**).

- `istream &get(char* niz, int max, char kraj='\n');` - čita sve znakove do prvog pojavljivanja znaka kraj. Pored toga, može da pročita najviše max broj znakova (npr. **`cin.get(ime, 20, '\n')`**).

- Metoda **getline** je veoma slična metodi **get**, jedino što uklanja znak kraj (npr. '\n') iz ulaznog toka, dok ga metoda **get** ne uklanja.

```
istream &getline(char* niz, int max, char kraj='\n');
```

Operacija **read**

- Operacija **read** čita bajtove iz toka u određeni deo memorije. Mora se navesti lokacija gde se upisuju pročitani podaci, kao i broj pročitanih bajtova. `istream& read(char* niz, int broj);`
- Za čitanje sloga

```
struct Radnik
{
   char ime[20];
   double plata;
};
```

iz datoteke “datoteka.dat":

Kreiramo tok is:

```
ifstream is(“datoteka.dat", ios::binary | ios::nocreate );
```

- Ukoliko je datoteka uspešno otvorena, vrši se čitanje podataka iz toka koji se upisuju u strukturu r, i odmah zatim i štampanje strukture r:

```
if( is )
{
   Radnik r;
   is.read( (char *) &r, sizeof(r) );
   cout << r.ime << ' ' << r.plata << endl;
}
```

- Na kraju otvoreni tok treba zatvoriti:

```
is.close();
```

- Ukoliko se ne navede broj bajtova koje treba pročitati, čitanje prestaje kada se dođe do kraja datoteke.

- **Operacije:**
  - **get i getline se koriste za ulazne tokove tekstualnih datoteka,**
  - **read i za ulazne tokove binarnih datoteka.**

- Tokovi datoteka pamte pokazivač na poziciju u datoteci koja će biti pročitana sledeća. Vrednost tog pokazivača se može odrediti/postaviti pomoću metode ulaznog toka

```
istream& seekg(long pozicija); ili
istream& seekg(long pozicija, seek_dir poc);
```

  - poc – tačka odakle se računa pomeraj:
    - ios::beg – od početka toka
    - ios::cur – od trenutne pozicije
    - ios::end – od kraja toka

```
ifstream is( "plata.dat", ios::binary | ios::nocreate );
is.seekg (8);      // 8 bajtova
```

- Vrednost pokazivača se može dobiti pomoću metode

```
long tellg();
```

(npr. `is.tellg()`).

Preklapanje operatora ekstrakcije

- Preklapanjem (overloading) operatora >> za neku klasu, promeniće se funkcionalnost tog operatora u slučaju upotrebe te klase.

```
class Datum
{
public:
   int dan, mesec, godina;
   friend istream& operator>> ( istream& is, Datum& dt );

};

istream& operator>> ( istream& is, Datum& dt )
{
   is >> dt.dan >> dt.mesec >> dt.godina;
   return is;
}
```

```
Datum dt;
cin >> dt;
```

Izlazni tokovi

- Tri najvažnije klase za izlazne tokove su **ostream**, **ofstream** i **ostringstream**.
  - Veoma retko se konstruišu objekti klase **ostream**, već se koriste predefinisani objekti (**cout**, **cerr**, **clog**).
  - Klasa **ofstream** podržava rad sa datotekama.
  - Za kreiranje niza u memoriji treba koristiti klasu **ostringstream**.

- **Konstruisanje objekata izlaznih tokova**
  - Konstruisanje objekata izlaznih tokova se vrši potpuno isto kao i kod ulaznih tokova, jedino što se koristi klasa **ofstream**.

Upotreba operatora umetanja

- Operator umetanja (<<) je programiran za sve standardne C++ tipove podataka, i služi za slanje bajtova objektu izlaznog toka.

```
cout << "Neki text";
```

- On takođe radi sa predefinisanim manipulatorima (elementima koji menjaju formatiranje, npr. endl);

```
cout << "Neki text" << endl;
```

- U ovom primeru manipulator endl predstavlja znak za prelazak u novi red.

Formatiranje izlaza

- Za određivanje jednake širine izlaza, koristi se manipulator **setw** ili metoda **width**.

```
double values[] = { 1.23, 35.36, 653.7, 4358.24 };
for( int i = 0; i < 4; i++ )
{
   cout.width(10);
   cout << values[i] << '\n';
}
```

- U gornjem primeru je pri ispisivanju vrednosti svaki red dopunjen blanko znacima do 10 znakova (desno poravnanje). Ukoliko želimo da se vrši popuna nekim drugim znakom, onda se koristi operacija **fill** (ispred dopuna).

```
for( int i = 0; i < 4; i++ )
{
   cout.width( 10 );
   cout.fill( '*' );
   cout << values[i] << endl
}
```

  - **std::left i std::right - za poravnanje**

- Ukoliko se koristi **setw** sa argumentima, vrednosti se štampaju u poljima iste dužine. U tom slučaju se mora uključiti i IOMANIP.H datoteka.

```
double values[] = { 1.23, 35.36, 653.7, 4358.24 };
char *names[] = { "Zoot", "Jimmy", "Al", "Stan" };
for( int i = 0; i < 4; i++ )
cout << setw( 6 )  << names[i]<< setw( 10 ) <<
   values[i] << endl;
```

Ovim se dobijaju dve kolone, u jednoj su ispisana imena, a u drugoj vrednosti.

Operacije izlaznog toka

- Kao i kod ulaznog toka, i ovde se koriste iste metode za otvaranje i zatvaranje toka (**open** i **close**).

- Za rad sa karakterima koristi se metoda **put**. Ona smešta zadati znak u izlazni tok. Na primer:

```
cout.put ('A');
```

- daje isti rezultat kao i

```
cout << 'A';
```

Funkcija write

- Funkcija **write** se koristi za upis bloka memorije u izlazni tok. Ona ima dva argumenta: prvi, **char** pokazivač i drugi, broj bajtova za upis. Treba napomenuti da je obavezna konverzija u **char*** pre adrese strukture ili objekta. Funkcija write se koristi i za binarne datoteke.
- Primer za upis strukture **Radnik** u binarnu datoteku:

```
Radnik r;
ofstream os("plata.dat", ios::binary);
os.write((char*) &r, sizeof(r));
os.close();
```

- Funkcije **seekp** i **tellp** su gotovo identične funkcijama **seekg** i **tellg**, jedino što se koriste za izlazne tokove.

Preklapanje operatora umetanja za korisničke tipove

- Preklapanje operatora umetanja (<<) za neku klasu dovodi do drugačijeg formata ispisivanja u izlazni tok. Može se koristiti kao sredstvo za formatiranje izlaza. Koristi se zajedno sa manipulatorima.

```
#include <iostream.h>

class complex {
      double real, imag;
      friend ostream& operator<<(ostream&, complex);
…
}

ostream& operator<< (ostream &os, complex c){
      return os << “(“ <<c.real<< “,“<<c.imag<< “)“;
}

void main() {
   complex c(1.2, 2.3);
   cout << "c=" << c;
}
```

Primer

```
void main()
{
    Datum dt(4,11,2004);
    cout << dt;
}
```

```
class Datum
{
public:
    int dan, mesec, godina;
    Datum (int d, int m, int g){
        dan = d; mesec = m; godina = g;
    }
    friend ostream& operator<< ( ostream& os, Datum& dt );
};

ostream& operator<< ( ostream& os, Datum& dt )
{
    os << dt.dan << "." << dt.mesec << "." << dt.godina;
    return os;
}
```