Programmering i C++
EDAF30
Strömmar och filer

Strömmar och filer

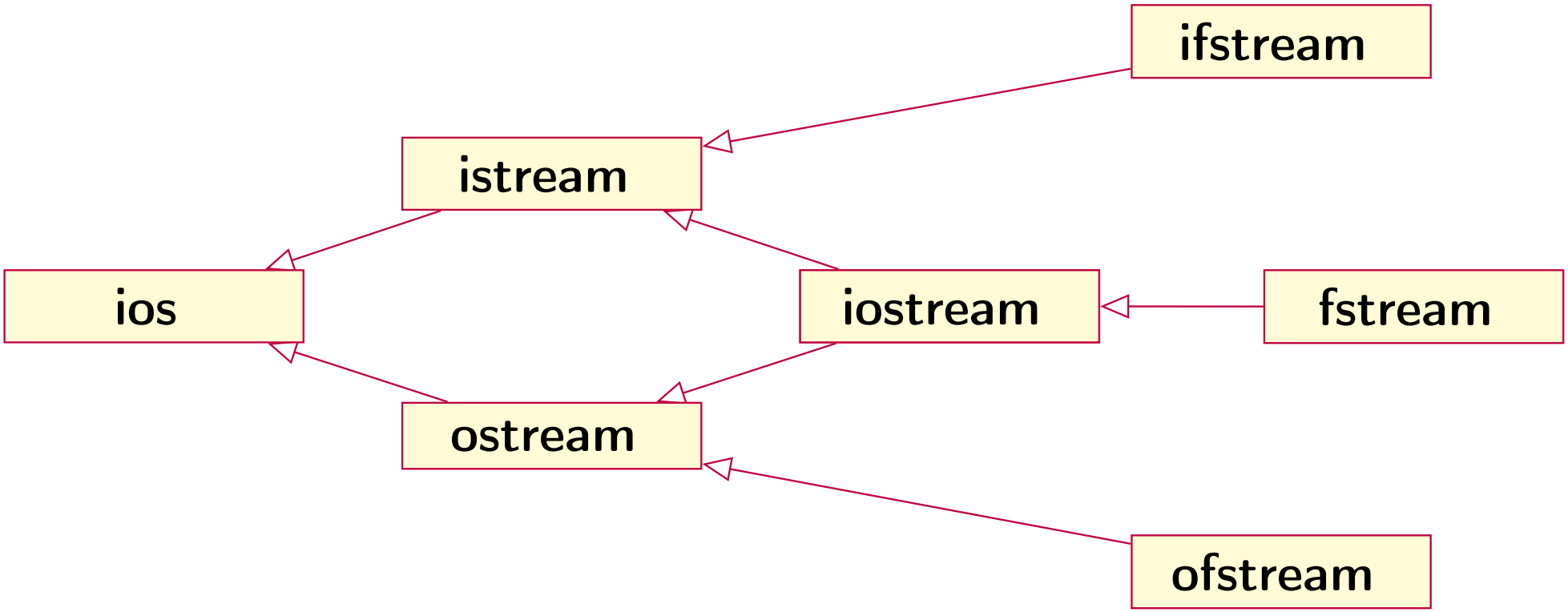
Innehåll

- Klassen `ios`
- Läsning av strömmar
- Utskrift av strömmar
- Koppling av filer till strömmar
- Direktaccess

Klassen `ios`

- Ström (stream) = Följd av tecken (bytes)
- Grundläggande klass för strömmar: `ios`

Klassträd för strömmar



Klassen `ios`

- Fördefinierade strömmar deklarerade i `<iostream>`:
  `cin`, `cout`, `cerr`, och `clog`
- Klasser deklarerade i inkluderingsfilen: `<fstream>`:
  `ifstream`, `ofstream`, `fstream`
- Flaggor definierade i `ios`, vilka beskriver en ströms tillstånd:

`failbit`	Senaste operationen misslyckades
`eofbit`	Ett filslut påträffades vid senaste operationen
`badbit`	Ett allvarligare fel av intern art har inträffat

Klassen `ios`

Medlemsfunktioner i klassen `ios`:

`void clear();`	Slår av alla tillståndsflaggorna
`bool good();`	Ger `true` om alla flaggor `false`
`bool fail();`	Ger `true` om `failbit` el. `badbit` satt
`bool eof();`	Ger `true` om `eofbit` är satt
`bool bad();`	Ger `true` om `badbit` är satt
`bool operator!();`	Ger resultatet `fail()`

Klassen ios

Inläsning tecken för tecken

```
char c;
// Felaktigt sätt:
while (!cin.eof()) {
    cin.get(c);
    //...
} // Läser 2 tecken för mycket (LF och EOF)

// Korrekt sätt:
while (cin.get(c)) {
    //...
} // Läser rätt antal tecken
```

Läsning av strömmar

Strömmar från klassen `istream` eller från subklass till denna

- Oformaterad inmatning: Läser data från strömmen utan att konvertera till annat format
  – Görs via medlemsfunktioner
- Formaterad inmatning: Data från strömmen görs om till annat format
  – Görs med operatorn >>

Läsning av strömmar

Formaterad inmatning med >>

```
// Kodavsnitt
string avr_ort;
int dag, manad;
cout << "Ange avreseort, dag och månad: ";
cin >> avr_ort >> dag >> manad;
cout << "Avresa från " << avr_ort
     << " den " << dag << '/' << manad << endl;


// In- och utmatning
Ange avereseort, dag och månad: Lund 24 10
Avresa från Lund den 24/10
```

Läsning av strömmar

Manipulatorer vid inmatning:

`setw(n)`	Max-antalet tecken i inläsningssträngen
`ws`	Hoppa fram till nästa icke-vita tecken
`skipws`	Hoppa över inl. vita tecken vid anv av >>
`noskipws`	Hoppa ej över inl. vita tecken vid anv av >>
`dec`	Tolka följande heltal som decimalt
`oct`	Tolka följande heltal som oktalt
`hex`	Tolka följande heltal som hexadecimalt
`boolalpha`	Indata för `bool` på formen `false` / `true`
`noboolalpha`	Indata för `bool` på formen `0` / `1`

Läsning av strömmar

Formaterad inmatning med >> och manipulatorer

```
// Kodavsnitt
#include <iomanip>

int i, j, k;
cout "Ange tre heltal: "
cin >> oct >> i >> hex >> j >> k;
cout << i << " " << j << " " << k << endl;


// In- och utmatning
Ange tre heltal:
12 34 56
10 52 56
```

Läsning av strömmar

Oformaterad inmatning med medlemsfunktioner

`gcount()`	Anger antalet tecken vid senaste inläsn.
`get()`	Läser in och returnerar nästa tecken
`get(c)`	Läser in tecken till `c`
`getline(s, n, t)`	Läser `n` tecken till `s` med `t` som radseparator
`get(s, n, t)`	Som getline men sep. `t` läses ej
`read(s, n)`	Läser in `n` st tecken till `s`
`ignore(s, t)`	Hoppar över `n` st tecken vid inl till `s`
`peek()`	Returnerar nästa tecken (som förblir oläst)
`putback(c)`	Lägger tillbaka `c` i strömmen
`unget()`	Lägger tillbaka senast lästa tecken

Utskrift till strömmar

Formaterad utmatning med <<

```
cout << uppercase << "hej svejs" << endl;
// ger utskriften

hej svejs



cout << uppercase << scientific << 123456789.0 << endl;
// ger utskriften

1234568E+008
```

Utskrift till strömmar

Manipulatorer vid utmatning:

`uppercase`	Ger stort `E` vid flyttalsutskrift
`fixed`	Utskrift med fixnotation
`scientific`	Utskrift med flytnotation
`hex`	Utskrift som hexadecimalt tal
`oct`	Utskrift som oktalt tal
`dec`	Utskrift som decimalt tal
`setw(n)`	Sätter minimalt antal positioner
`setfill(c)`	Anger tecken för utfyllnad (padding)
`setprecision(n)`	Sätter antalet decimaler (om `fixed`) eller antalet sign. siffror (om `scientific`)
`setbase(n)`	`setbase(16)`, `setbase(8)` osv
`flush`	Tömmer utskriftsbufferten
`endl`	Lägger in radslut i strömmen

Utskrift till strömmar

Oformaterad utmatning med medlemsfunktioner

`put(c)`	Skriv ut tecknet `c` till strömmen
`write(s, n)`	Skriv ut `n` tecken från fältet `s`
`flush()`	Töm utskriftsbufferten

Koppling av filer till strömmar

Öppnande av fil för läsning

```
ifstream infil("infilen.txt");

// Alt.
ifstream infil;
infil.open("infilen.txt");
```

Öppnande av fil för skrivning

```
ofstream utfil("utfilen.txt");

// Alt.
ofstream utfil;
utfil.open("utfilen.txt");
```

Koppling av filer till strömmar

Stängning av fil

```
infil.close();

utfil.close();
```

Ofta behövs inte `close` användas eftersom destruktorerna för `ifstream` och `ofstream` automatiskt anropas då den associerade strömmen termineras

Koppling av filer till strömmar

Kopiering av fil

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

main (int argc, char* argv[]) {
    if (argc != 3) {
        cout << "Syntax: " << argv[0]
             << " from_file to_file" << endl;
    }
    char c;
    ifstream f1(argv[1], ios::binary);//Binärfil
    ofstream f2(argv[2], ios::binary);

    while (f1.get(c)) // Snabbare sätt (via filbuffert):
        f2.put(c);    // f2 << f1.rdbuf();
}
```

Koppling av filer till strömmar

Filflaggor i klassen `ios`

`in`	Filen skall existera och vara läsbar.
`out`	Om filen existerar skall den skrivas över.
	Om filen inte finns skall en ny skrivbar fil skapas.
`app`	Om filen existerar skall skrivning läggas till i slutet.
	Om filen inte finns skall en ny skrivbar fil skapas.
`trunc`	Om filen redan finns skall den skrivas över
`ate`	Efter öppning flyttas filpekaren till slutet av filen
`binary`	Filen skall hanteras som en binärfil

Kombination av flaggor med `operator|` (bitvis eller)

```
ofstream filen("fil.dat",ios::trunc|ios::binary);
```

Direktaccess

Direkt indexering av filposition där index är av typen `streampos` (heltalstyp)

`is.tellg()`	Ger aktuell position i inströmmen `is`
`os.tellp()`	Ger aktuell position i utströmmen `os`
`is.seekg(pos)`	Sätter aktuell position i `is` resp. `os`
`os.seekp(pos)`	till pos (av typen streampos)
`is.seekg(off, dir)`	Sätter positionen till `off+dir` där
`os.seekp(off, dir)`	`off` är en (ev neg.) offset och dir är `ios::beg` (start), `ios::end` (slut) eller `ios::cur` (akt. pos.)