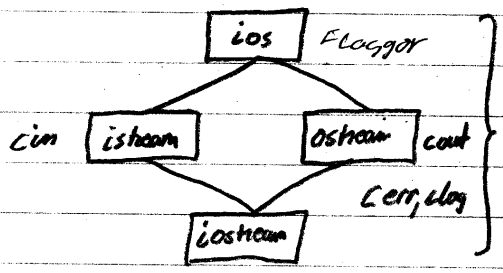
6) In- och utmatning, strömmar.



#include <iostream>
definierar globala objekt
cin, cout ---

Ex

```
!
float x;
!
cin >> x; // 32.5\n   \n62e\n   12.5k\n   k\n
          //  ok!      ok!      ok!      Fel!
!
cout << "Ge reellt x: ";
cin >> x;
while (!cin) // cin mår dåligt!
{
    cin.clear(); // cin mår bra igen!
    cin.ignore(80, '\n'); // Ronsan tom \n dvs
                          // max 80 tecken
    cout << "Ge reellt x: ";
    cin >> x;
}
// Här är x ok!
!
```

Alternativt kan inläsningen stoppas in i while-villkoret enligt `while(!(cin >> x))`

(Ex) !

```
char ch;
!
cout << "Ge ett tecken (avsluta med ENTER): ";
cin.get(ch); // Går ej att använda cin>>ch
             // eftersom man då hoppar över ␣ tab \n
while (ch != '\n')
{
   cout << ch << endl;
   cin.ignore(); // Hoppar över \n
   cout << "Ge ett tecken (avsluta med ENTER): ";
   cin.get(ch);
}
!  Alt! Funktion som returnerar ch=cin.get();
```

(Ex) !

```
char namn[30];
!
cout << "Ge ett namn (avsluta med ENTER): ";
cin.getline(namn, sizeof(namn), '\n'); // OBS! cin>>namn
                                       // läser till blank
while (namn[0] != '\0')
{
    cout << namn << endl;
    cout << "Ge ett namn (avsluta med ENTER): ";
    cin.getline(namn, sizeof(namn), '\n');
}
```

Alt! Funktion som kikar på nästa tecken cin.peek().

Formatflaggor — bestämmer tillståndet för cin, cout, ...

Ex

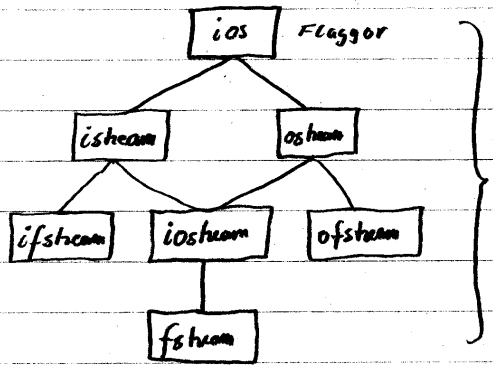
```
cout.precision(3);
cout << 53.2136; // 53.2 (3 värdesiffror)

                          ios::floatfield
cout.setf(ios::fixed, ios::fixed | ios::scientific);
cout << 53.2136; // 53.214 (3 decimaler)

cout.width(10); // Gäller bara närmaste utskrift
cout << 53.2136; //␣␣␣␣53.214

cout.unsetf(ios::fixed);
cout << 53.2136; // 53.2
```

Manipulatorer — manipulerar tillståndet hos strömmar

Ex

```
#include <iomanip>

cout << setprecision(3) << 53.2136 << endl; //53.2

cout << setiosflags(ios::fixed) << 53.2136; //53.214

cout << setw(10) << 53.2136; //␣␣␣␣53.214

cout << resetiosflags(ios::fixed) << 53.2136; //53.2
```

Strömmar – kanaler för in- och utmatning till/från valfria enheter



Textström – ASCII ↷ Binär

(Ex) Beräkna summan av talen i textfilen Retal.txt enligt:

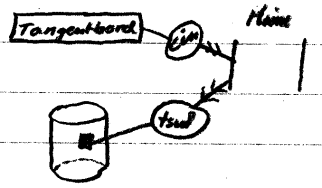
5.26 5.74 --

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

void main()
{
    ifstream tsin("Retal.txt");
    float sum=0.0, x;
    if (!tsin)
        cerr << "Finns ingen sådan fil!" << endl;
    else
    {
```

```
        tsin >> x;
        while (!tsin.eof())
        {
            sum += x;
            tsin >> x;
        }
        cout << setiosflags(ios::fixed) << setprecision(2)
            << "Summa = " << sum << endl;
    }
}
```

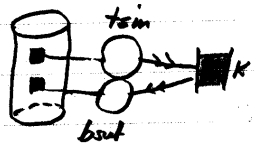
(Ex) Läs komplexa tal från tangentbordet till textfilen Komplex.txt. Avsluta med eof (ctrl-z)

```
#include "Komplex.h"  //Här finns bl.a << och >>
#include <fstream>
using namespace std;
void main()
{
    KOMPLEX k;
    ofstream tsut("Komplex.txt");

    cout << "Inmatningen avslutas med ctrl-z!" << endl;
    while (!(cin >> k).eof())
    {
        tsut << k;          //OBS! Överlagrad << gäller
    }                       Även för textfiler
}
```



Binärström – skyffla bytes

(Ex) Läs alla komplexa tal från textfilen Komplex.txt till binärfilen Komplex.dat




```
#include <fstream>
using namespace std;
void main()
{
  KOMPLEX k;
  ifstream tsin("Komplex.txt");
  ofstream bsut("Komplex.dat", ios::out | ios::binary);

  while(!(tsin>>k).eof())
  {
      bsut.write((char*)&k, sizeof(KOMPLEX));
  }
}
```

(Ex) Läs all komplexa tal från binärfilen Komplex.dat och summera.

```
!
ifstream bsin("Komplex.dat", ios::in | ios::binary);
KOMPLEX k, sum=0;

while(!bsin.read((char*)&k, sizeof(KOMPLEX)).eof());
{
    sum = sum + k;
}
cout << "Summan = " << sum << endl;
!
```

(Ex) I binärfilen Stud.dat finns ett antal studenter

a) Program som skriver ut alla studenter som har mindre än 25 studiepoäng.

```
:
#include "Student.h"
:
void main()
{
    STUDENT a;
    ifstream bsin("Stud.dat", ios::in | ios::binary);

    while(!bsin.read((char *)&a, sizeof(STUDENT)).eof())
    {
        if (a.get_poang() < 25)
        {
            a.skriv();
        }
    }
}
```

b) Program som uppdaterar åldern för alla studenter som fyller år om namnen på dessa finns i textfilen gratis.txt

```
:
void main()
{
    STUDENT a;
```

```
char namn[30];
ifstream tsin("Gratis.txt");
fstream bsinut("Stud.dat", ios::in | ios::out | ios::binary);

while( ! tsin.getline(namn, sizeof(namn)).eof())
{
 bsinut.seekg(0);
 while(! bsinut.read((char *)&a,sizeof(STUDENT)).eof()
       && stricmp(namn, a.get_namn()) != 0)
 {
 }
 if (! bsinut.eof())
 {
    a.set_alder(a.get_alder() + 1);
    bsinut.seekp(-sizeof(STUDENT), ios::cur);
    bsinut.write((char *)&a, sizeof(STUDENT));
 }
 else
 {
    cout << "Studenten finns ej i registret!" << endl;
    bsinut.clear();
 }
}
}
```

7) Templates och Exceptions

Template – mall för olika typer (klasser)

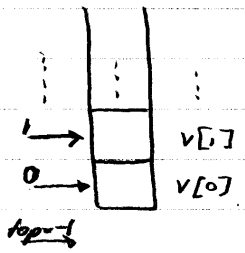
(Ex) Generell sorteringsfunktion

```
//Sort.h

template <class T>
void byt(T &a, T &b)
{
    T temp;

    temp=a;
    a = b;
    b = temp;
}

template <class T>
void unsort(T v[], int nr)
{
    for (int i=0; i<nr-1; i++)
    {
        for(int k=i+1; k<nr; k++)
        {
            if (v[k] < v[i])        //< märk funn!
                byt(v[i], v[k]);
        }
    }
}
```

```
// sortmain.cpp

#include "sort.h"

void main()
{
    float fv[4] = {3.2, 4.6, 1.8, 2.7};
    RTAL  rv[3] = {RTAL(3,4), RTAL(5,6), RTAL(1,2)};

    usort(fv, 4);   //Kompilatorn skapar usort för float!
    usort(rv, 3);   //  ————————,,————————  rtal!
}
```

(Ex) En generell STACK-klass

```
//Stack.h

template <class T, int nr>
class STACK
{
    private:
        int top;
        T v[nr];
    public:
        STACK();
        void push();
        T pop();
};
```



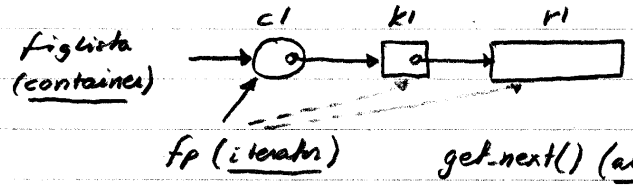
```
template <class T, int nr)
STACK<T,nr> :: STACK()
{
    top = -1;
}

template <class T, int nr>
void STACK<T,nr> ::push(T a)
{
    V[++top] = a;
}

template <class T, int nr>
T STACK<T,nr> :: pop()
{
    return V[top--];
}

// Stackmain.cpp

#include "Stack.h"
|
void main()
{
    STACK <KOMPLEX, 10> ks;
    |
    ks.push(KOMPLEX(5.0, 3.0));
    |
```

Klassbibliotek

Återanvändning
- Arv ex GUI
- Templat ex container (listor, träd

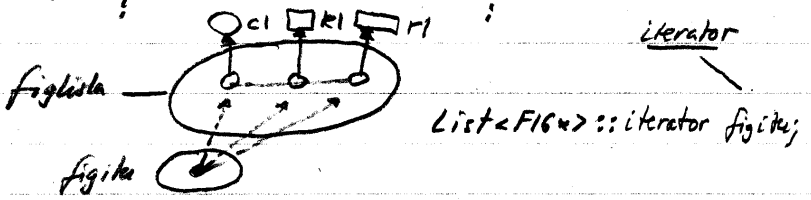
(Ex) Lista av figurer!

a) egentillverkad – nackdel måste ändra i egna klass



b) C++ standardbibliotek (gamla STL-bibliotekt)

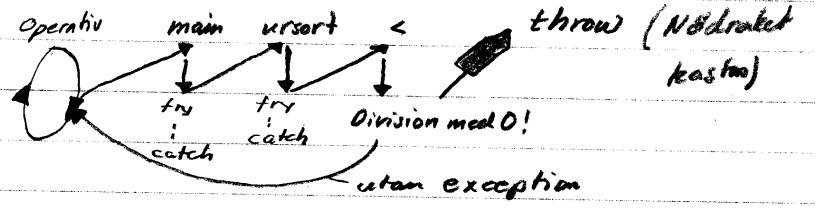
```
#include <list>
using namespace std;

List<FIG *> figlista;        // container

figlista.insert(figlista.begin(), &r1);   // algoritmer
figlista.insert(figlista.begin(), &k1);
```




```
for(figiter = figlista.begin(); figiter != figlista.end(); figiter++)
    asum += (*figiter)->area();
```

Exception — undantag, oväntade fel

(Ex) Sortera bråk



Med exceptions kan felet fångas upp och åtgärdas.

```
bool RTAL::operator<(RTAL r) throw(char *)
{
   if (n*r.n == 0)
        throw("Division med 0 i <-funktion!");
   return float(t)/n < float(r.t)/r.n;
}

void ursort(RTAL v[], int nr) throw(int)
{
  ...
  try
  {
     if (v[k] < v[i]) byt(v[i], v[k]);
  }
  catch (char *mess)
  {
     cerr << mess << endl;
     throw k;
  }
  ...
```

```
void main()
{
    RTAL rv[3] = { RTAL(3,4), RTAL(5,0), ----
    bool done = false;

    while (!done)
    {
        try
        {
            ursort(rv, 3);
            done = true;
        }
        catch (int k)
        {
            if (k==1)  // Måste kolla om rv[0] felaktig!
            {
              cout << "rv[0] = " << rv[0] << endl;
              cout << "Ge nytt bråk : ";
              cin >> rv[0];
            }
            cout << "rv[" << k << "] = " << rv[k] << endl;
            cout << "Ge nytt bråk : ";
            cin >> rv[k];
        }
    }
}
```