Krav

- Tre krav måste vara uppfyllda:
  - Abstrakt datatyp
  - Arv
  - Polymorfism och dynamisk bindning
- Den abstrakta datatypen kallas ofta klass
  - Data kallas **attribut**, funktionaliteten finns i **metoder**
  - Anrop av en metod kallas att skicka ett **meddelande** till ett objekt

Tre sätt ...

- Objektmodell i botten men med 'imperativ stil' för primitiva datatyper
  - + snabbt för heltal och andra primitiva typer
  - - Komplikationer när metoder bara kan ta objekt som parametrar -> Wrapper Classes
- C++ hör till den senare typen
- Allt är objekt utom de primitiva datatyperna som int, double, ...
- String är objekt

Egenskaper

- Hårdare typkontroll → ~Pascal
- Referenstyp → blir mer lättläst
- Egendef typer, klasser → bygger ut språket
- Streams → säkrare I/O
- Arv → beskriver skillnader
- Dynamisk bindning → underlättar utbyggnad
- Typparametrisering → generella mallar
- Undantagshantering → bra felhantering

Viktiga skillnader mot C

- Funktioner måste deklareras innan anrop
- Tidigare global `const` är lokal i filen
- Typen på en teckenkonstant 'a' är char, ej int
- Kommentarer skrivs också med //
- Variabler kan definieras inne i koden
- Nya streams ersätter stdio
- Dynamiskt minne - new, free
- Referensvariabler

Klasser

- Är en egendefinierad typ
- En mall för ett objekt
- Ungefär som en struct men även med funktioner
- Kan skydda åtkomsten för andra, inkapsling

Konstruktor

- Kan användas för initiering av värden
- Anropas automatiskt vid deklaration av objekt (variabler)
- Ingen returtyp eller returvärde
- Har alltid samma namn som klassen
- Kan finnas flera (med olika parametrar)
- Det finns även en destruktor

Funktioner

- Prototyper krävs
- Överlagring (samma namn, olika parametertyper)
- Inline expansion (ej loopar eller rekursiva funktioner)
- Namnmangling ger typsäker länkning
- Defaultvärden (från höger)
- Kan ge operatornamn

Vänner, friends

- Används när vanliga funktioner behöver nå klassmedlemmar
- Klassen talar om vem man litar på

```
Class Bankkonto
{
public:
friend ostream& operator<<(ostream& os,
                           Bankkonto& bk);
private:
int saldo;
};
```

Strömmar

- Typsäkert
- Snabbt
- Utökbart
- Objekt-orienterat
- Även för filer och med strängar

Operationer på strömmar

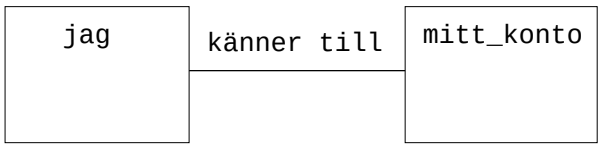
- #include <iostream.h>
- stream << arg; stream >> arg;
- stream.read(char *buffer, int length);
- stream.write(char *buffer, int length);
- stream.getline(char *line, int len, char delim = '\n');
- stream.get(char& ch);
- stream.put(char ch);
- char stream.peek();

Målen med objekt-orientering

- Modellering - nära verkligheten; lättförstått
- Modularisering - kring verkliga begrepp; stabilt
- Förändring - enkel lokalisering
- Utbyggnad - nya klasser,operationer el. arv
- Återanvändbarhet - mha arv
- Produktivitet - klassbibliotek
- Terminologi - samma i alla faser;spårbarhet

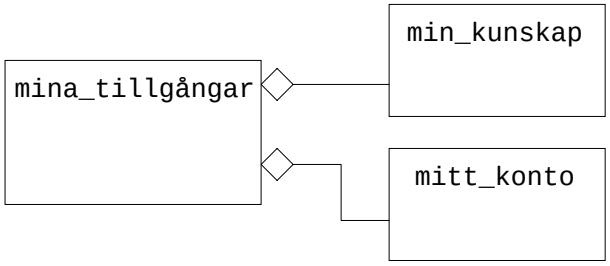
Association "Har koppling till"

- Kan vara enkel- eller dubbelriktad
- 1-1, 1-n, n-1
- statisk (alltid) eller dynamisk (ibland)
- "känner till", "pekar på", "har en"



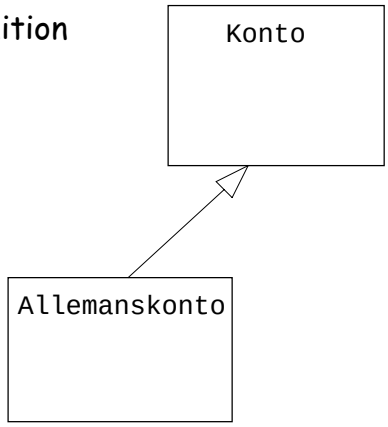
Aggregat "Består av"

- uppbyggt av andra objekt
- alltid statisk
- "innehåller", "ingår i", "är del av"



Arv "Är en/ett"

- Mellan klasser
- Tillägg/omdefinition
- Ej borttag



Abstrakt basklass

- Fångar gemensamma egenskaper
- Påtvingar ett gränssnitt
- Med äkta virtuell funktion:

```
class Figur
{
public:
virtual double getArea() = 0;
};
```

- Eller med `protected` konstruktor

Överlagring av operatorer

- För uttryck med egna typer
  - Kan ej omdefinieras för inbyggda typer
- De flesta (dock ej :: . .* ? :)
- Inga nya operatorer t ex < >, **
- Samma precedens & associativitet
- Samma antal operander (unär/binär)
- 2 sätt att överlagra

Polymorfism och dynamisk bindning