

# Arv og operatoroverlastning

---

Forelesning 6,  
Effektiv kode med C og C++, vår 2014  
Alfred Bratterud

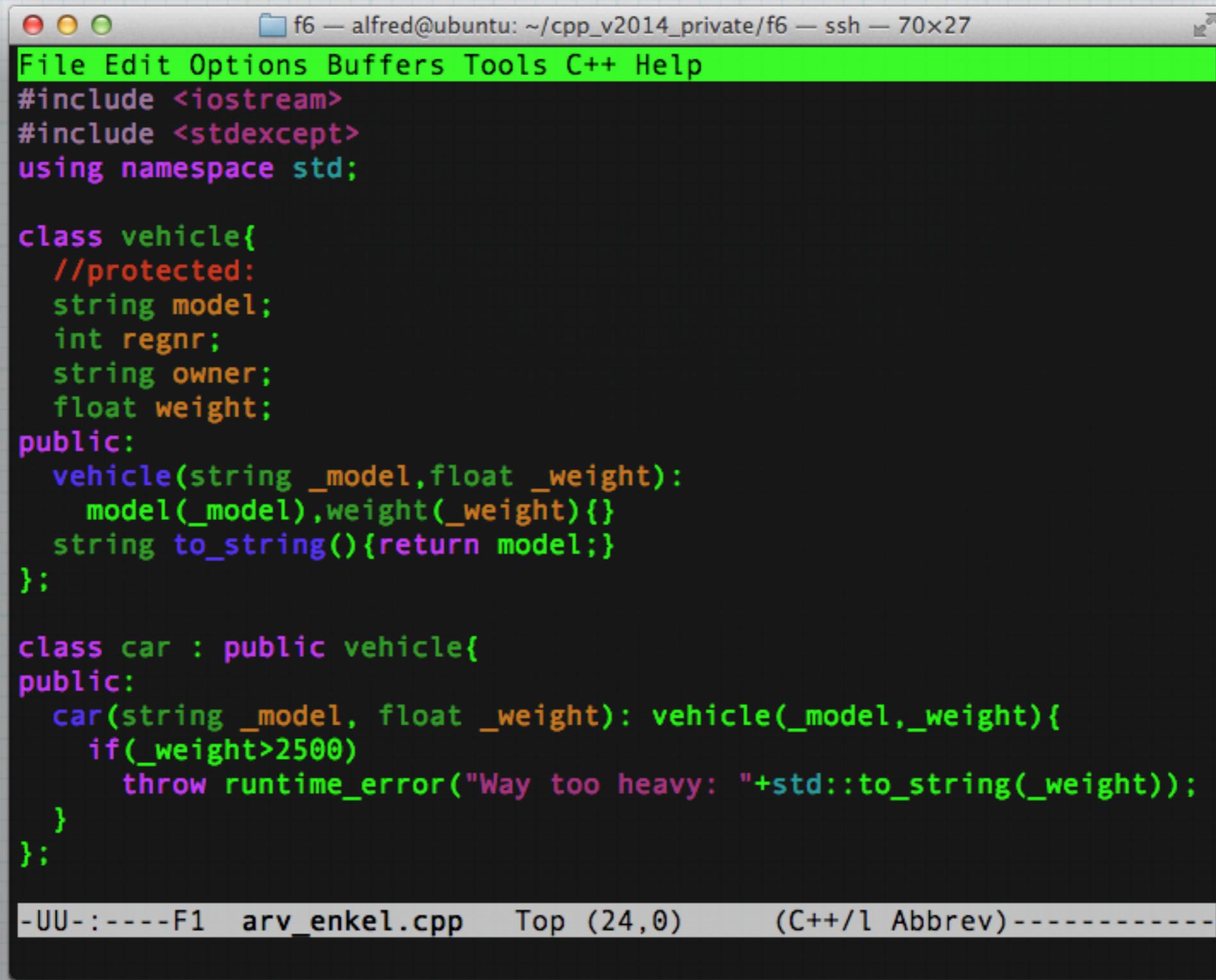
# Agenda:

- \* Går det greit med obligen?
- \* Arv
- \* Operatoroverlasting

# Arv

- \* En klasse kan arve egenskaper fra en annen klasse.
- \* Den som arver er en "sub-class" og den som blir arvet fra er en "base class" (Super-class i java)
- \* Hovedmotiv med arv:
  - \* Mindre kode!
  - \* Er det så viktig?
  - \* Færre duplikater, færre feil.
- \* Mest brukt: legge til nye egenskaper i eksisterende klasser:
  - \* String med stringsplit?
  - \* Invalid\_XML\_exception?

# Ary



The screenshot shows a terminal window titled "f6 — alfred@ubuntu: ~/cpp\_v2014\_private/f6 — ssh — 70x27". The window contains the following C++ code:

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

class vehicle{
    //protected:
    string model;
    int regnr;
    string owner;
    float weight;
public:
    vehicle(string _model, float _weight):
        model(_model), weight(_weight){}
    string to_string(){return model;}
};

class car : public vehicle{
public:
    car(string _model, float _weight): vehicle(_model,_weight){
        if(_weight>2500)
            throw runtime_error("Way too heavy: "+std::to_string(_weight));
    }
};

-UU-:----F1  arv_enkel.cpp  Top (24,0)  (C++/l Abbrev)-----
```

# Arv

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 70x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <stdexcept>
using namespace std;

class vehicle{
    //protected:
    string model;
    int regnr;
    string owner;
    float weight;
public:
    vehicle(string _model, float _weight):
        model(_model), weight(_weight){}
    string to_string(){return model;}
};

class car : public vehicle{
public:
    car(string _model, float _weight): vehicle(_model,_weight){
        if(_weight>2500)
            throw runtime_error("Way too heavy: "+std::to_string(_weight));
    }
};

-UU-:----F1  arv_enkel.cpp  Top (24,0)  (C++/l Abbrev)-----
```

Kun protected og public  
medlemmer er tilgjengelige i  
subklassene. Bør de være det?

# Arv

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 70x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <stdexcept>
using namespace std;

class vehicle{
    //protected:
    string model;
    int regnr;
    string owner;
    float weight;
public:
    vehicle(string _model, float _weight):
        model(_model), weight(_weight){}
    string to_string(){return model;}
};

class car : public vehicle{
public:
    car(string _model, float _weight): vehicle(_model,_weight){
        if(_weight>2500)
            throw runtime_error("Way too heavy: "+std::to_string(_weight));
    }
};

-UU-:----F1  arv_enkel.cpp  Top (24,0)  (C++/l Abbrev)-----
```

Kun protected og public  
medlemmer er tilgjengelige i  
subklassene. Bør de være det?

I C++11 arves konstruktører!

# Arv

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 70x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <stdexcept>
using namespace std;

class vehicle{
    //protected:
    string model;
    int regnr;
    string owner;
    float weight;
public:
    vehicle(string _model, float _weight):
        model(_model), weight(_weight)
    string to_string(){return model;}
};

class car : public vehicle{
public:
    car(string _model, float _weight): vehicle(_model,_weight){
        if(_weight>2500)
            throw runtime_error("Way too heavy: "+std::to_string(_weight));
    }
};

-UU-:----F1  arv_enkel.cpp  Top (24,0)  (C++/l Abbrev)-----
```

Kun protected og public medlemmer er tilgjengelige i subklassene. Bør de være det?

I C++11 arves konstruktører!

klasse : base - ":" betyr "årver"

# Arv

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 70x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <stdexcept>
using namespace std;

class vehicle{
    //protected:
    string model;
    int regnr;
    string owner;
    float weight;
public:
    vehicle(string _model, float _weight)
        : model(_model), weight(_weight) {}
    string to_string(){return model;}
};

class car : public vehicle{
public:
    car(string _model, float _weight)
        : vehicle(_model, _weight) {
            if(_weight>2500)
                throw runtime_error("Way too heavy!");
    }
};
```

Kun protected og public medlemmer er tilgjengelige i subklassene. Bør de være det?

I C++11 arves konstruktører!

klasse : base - ":" betyr "årver"

her bestemmer vi tilgangsnivå til baseklassen. Standard er "alt skjult". Public gjør alt "like åpent som i basen"

# Årv

- \* "Protected: " gir tilgang kun til subklasser. Bjarne kaller dette "most likely a design flaw".
  - \* Hvorfor blottlegge de private delene av baseklassen?
  - \* Vi må vansett sette medlemmer via konstruktør
  - \* Det den har som er offentlig er jo tilgjengelig
- \* Vi kan arve "så dypt" vi vil, altså subklaser av subklasser av ... etc.
- \* I C++ kan vi også arve fra flere klasser - "multiple inheritance"
  - \* Gir mulighet for "mixin" system  
<http://en.wikipedia.org/wiki/Mixin>
  - \* `class car_with_color : public vehicle, public with_color{...}`
  - \* Gjør at man må passe på navnekollisjoner (`to_string` i flere?)

f6 — alfred@ubuntu: ~/cpp\_v2014\_private/f6 — ssh — 74x34

```
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <stdexcept>
using namespace std;

class vehicle{
    //protected:
    string model;
    int regnr;
    string owner;
    float weight;
public:
    vehicle(string _model, float _weight):
        model(_model), weight(_weight){}
    string to_string(){return model;}
};

class with_color{
    int _color=0xff0000;
public:
    int color(){return _color;}
    //string to_string(){return std::to_string(_color);} Oups!
};

class car_with_color : public vehicle, public with_color{
public:
    car_with_color(string _model, float _weight): vehicle(_model,_weight){
        if(_weight>2500)
            throw runtime_error("Way too heavy: "+std::to_string(_weight));
    }
};

-UU-----F1  arv_multippel.cpp  Top (31,0)  (C++/l Abbrev)-----
```

Multippel arv:  
"Mixin"-system; litt  
farge, litt sukker...

Nyttig for:  
scrollable\_window  
framed\_picture  
two\_way\_door

...

...

IMO~MSA~qoor  
LSSN6Q~bigJOKS

# Exception-subklasser og “throw specifier”

- \* Det er veldig nyttig å lage egne exceptions.
  - \* Hvorfor ikke bare egne tekster?
  - \* Egne exceptions kan håndteres av egne catch'es.
    - \* Gir mulighet for finmasket feilhåndtering
- \* Før C++11 ble “throw specifiers” brukt for å spesifisere hva en funksjon kunne kaste
- \* Nyttig? Men kompilatoren kan ikke vite det.
- \* Skal du kompilere en exception-subklasse i C++<11 må du ha med destructor som “lover å ikke kaste noe”.
- \* I C++11 er dette tatt bort (noexcept er et alternativ)
- \* Mer her: <http://www.gotw.ca/publications/mill22.htm>

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 63x32
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <exception>
#include <fstream>
using namespace std;

class file_not_found_exception : public exception {
    string reason;
public:
    string what(){
        return reason;
    }

    file_not_found_exception(string _reason): reason(_reason){}
    ~file_not_found_exception() throw() {}
};

int main(){
    string filename="arv_er_kult.cpp";
    try{
        ifstream f(filename.c_str());
        if(!f.is_open())
            throw file_not_found_exception(filename);
    }catch(file_not_found_exception fnfe){
        cout << "File not found: " << fnfe.what() << endl;
    }catch(exception e){
        cout << "Exception: " << e.what() << endl;
    }
}
-UU- :***-F1  arv_exceptions.cpp  All L29      (C++/l Abbrev) ----
```

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 63x32
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <exception>
#include <fstream>
using namespace std;

class file_not_found_exception : public exception {
    string reason;
public:
    string what(){
        return reason;
    }

    file_not_found_exception(string _reason): reason(_reason){}
    ~file_not_found_exception() throw() {}
};

int main(){
    string filename="arv_er_kult.cpp";
    try{
        ifstream f(filename.c_str());
        if(!f.is_open())
            throw file_not_found_exception(filename);
    }catch(file_not_found_exception fnfe){
        cout << "File not found: " << fnfe.what() << endl;
    }catch(exception e){
        cout << "Exception: " << e.what() << endl;
    }
}
-UU- :***-F1  arv_exceptions.cpp  All L29      (C++/l Abbrev) ----
```

Vi arver "exception" som vanlig

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 63x32
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <exception>
#include <fstream>
using namespace std;

class file_not_found_exception : public exception {
    string reason;
public:
    string what(){
        return reason;
    }

    file_not_found_exception(string _reason): reason(_reason){}
    ~file_not_found_exception() throw() {}
};

int main(){
    string filename="arv_er_kult.cpp";
    try{
        ifstream f(filename.c_str());
        if(!f.is_open())
            throw file_not_found_exception(filename);
    }catch(file_not_found_exception fnfe){
        cout << "File not found: " << fnfe.what() << endl;
    }catch(exception e){
        cout << "Exception: " << e.what() << endl;
    }
}
-UU- :***-F1  arv_exceptions.cpp  All L29      (C++/l Abbrev) ----
```

Vi arver "exception" som vanlig

Legger til constructor som tar streng

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 63x32
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <exception>
#include <fstream>
using namespace std;

class file_not_found_exception : public exception {
    string reason;
public:
    string what(){
        return reason;
    }

    file_not_found_exception(string _reason): reason(_reason){}
    ~file_not_found_exception() throw() {}
};

int main(){
    string filename="arv_er_kult.cpp";
    try{
        ifstream f(filename.c_str());
        if(!f.is_open())
            throw file_not_found_exception(f);
    }catch(file_not_found_exception fnfe){
        cout << "File not found: " << fnfe.what() << endl;
    }catch(exception e){
        cout << "Exception: " << e.what() << endl;
    }
}
-UU- :***-F1  arv_exceptions.cpp  All L29      (C++/l Abbrev) ----
```

Vi arver "exception" som vanlig

Legger til constructor som tar streng

Vi må spesifisere at ingen ting  
kastes av konstruktoren, fordi  
baseklassen har dette

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 63x32
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <exception>
#include <fstream>
using namespace std;

class file_not_found_exception : public exception {
    string reason;
public:
    string what(){
        return reason;
    }

    file_not_found_exception(string _reason): reason(_reason){}
    //~file_not_found_exception() throw() {}

};

int main(){
    string filename="arv_er_kult.cpp";
    try{
        ifstream f(filename.c_str());
        if(!f.is_open())
            throw file_not_found_exception(filename);
    }catch(file_not_found_exception fnfe){
        cout << "File not found: " << fnfe.what() << endl;
    }catch(exception e){
        cout << "Exception: " << e.what() << endl;
    }
}
-UU- -----F1  arv_exceptions.cpp  All L29      (C++/l Abbrev) -----
```

Men ikke i C++]]

...det står respekt av å gå bort fra  
dårlige idéer, selv de man har  
publisert

# Demo:

---

arv\_enkel.cpp  
arv\_multippel.cpp

# operatoroverlasting

- \* Hva er egentlig en operator?
  - \* Eksempler er `+`, `-`, `==`, `+=`, `=`, `->`, `*` etc.
  - \* En funksjon eller medlemsfunksjon
- \* I mange språk gjelder operatorene kun forhåndsdefinerte typer
- \* I C++ kan de fleste operatorene defineres på nytt, for andre typer. Generell form er:
- \* **type operator**symbol** (parametre) { ... }**

# Operatorer som kan overlastes

Overloadable operators															
+	-	*	/	=	<	>	+=	-=	*=	/=	<<	>>			
<<=	>>=	==	!=	<=	>=	++	--	%	&	^	!				
-	&=	^=	=	&&		%=	[]	()	,	->*	->	new			
delete	new[ ]			delete[ ]											

<http://www.cplusplus.com/doc/tutorial/classes2/>

# Eksempel - []

- \* Anta vi har en klasse, "intPair":

```
class intPair{  
    int left;  
    int right;  
public:  
    intPair(int l,int r):left(l),right(r){};  
    int operator[](unsigned int i){ .. ! .. }
```

- \* type operatorsymbol (parametre) { ... }
- \* Betrakt som int myIntPair[ int i ] { ... }

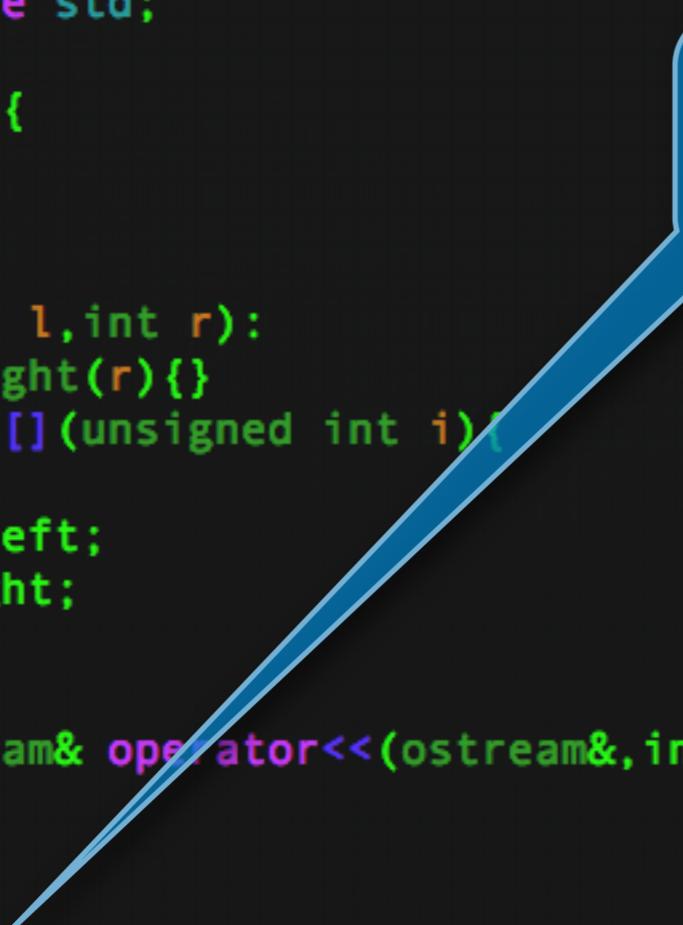
# Ikke alltid rett frem...

- \* Vi så en operator, overlastet som en medlemsfunksjon. Ikke alltid rett frem...
- \* For "standardverjsoner" av `[]` og `==` er det forholdsvis greit, men verre med:
  - \* `c++`, `++c`
  - \* Hva skal returneres? Rekkefølgen teller...
  - \* Og hva med dette:
    - \* `myCash +100 , 500+myCash?`
  - \* Her er det greit for det første, men ikke for det andre.

# Argumenttrekkefølge teller

- \* Vi kan da bruke en frittstående funksjon
- \* **type operatorsymbol (param1, param2) { ... }**
- \* Her kan vi tenke at **param1** er implisitt referanse til **\*this** i en medlemsfunksjon
- \* Mens vi kan eksplisitt oppgi begge når vi lager en ekstern funksjon
- \* Problem: Hva hvis feks. operator+ trenger tilgang til private medlemmer?
- \* Klassen din kan få en venn! (enten en funksjon eller en klasse)
- \* **friend type operator symbol (param1, param2) { ... }**
- \* Lurt? I C++ er du sjefen. ...sjefen har alt ansvaret.

# operator<< v.1



```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }

    friend ostream& operator<<(ostream&,int_pair);
};

ostream& operator<<(ostream& s,int_pair p){
    s << "(" << p.left << "," << p.right << ")"; //<< endl?
    return s;
}

-UU-:----F1  operator_overloading_stream1.cpp  Top L24  (C++/l Abb
```

Stromoperatoren kan ikke  
overlastes som medlem... hvorfor?

# operator<< v.1

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }

    friend ostream& operator<<(ostream&,int_pair);
};

ostream& operator<<(ostream& s,int_pair p){
    s << "(" << p.left << "," << p.right << ")"; //<< endl?
    return s;
}

-UU-----F1  operator_overloading_stream1.cpp  Top L24  (C++/l Abb)
```

Stromoperatoren kan ikke  
overlastes som medlem... hvorfor?

Fordi det vi VIL er å legge til et  
medlem i ostream, for vår klasse

# operator<< v.1

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }
    friend ostream& operator<<(ostream&,int_pair);
};

ostream& operator<<(ostream& s,int_pair p){
    s << "(" << p.left << "," << p.right << ")"; //<< endl?
    return s;
}
```

Stromoperatoren kan ikke  
overlastes som medlem... hvorfor?

Fordi det vi VIL er å legge til et  
medlem i ostream, for vår klasse

I stedet kan vi definere  
den eksternt

-UU-----F1 operator\_overloading\_stream1.cpp Top L24 (C++/l Abb)

# operator<< v.1

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }
    friend ostream& operator<<(ostream&,int_pair);
};

ostream& operator<<(ostream& s,int_pair p){
    s << "(" << p.left << "," << p.right << ")";
    return s;
}
```

Stromoperatoren kan ikke  
overlastes som medlem... hvorfor?

Fordi det vi VIL er å legge til et  
medlem i ostream, for vår klasse

I stedet kan vi definere  
den eksternt

Og gi den tilgang til  
klassen vår

-UU-----F1 operator\_overloading\_stream1.cpp Top L24 (C++/l Abb)

# operator<< v.1



```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }

    friend ostream& operator<<(ostream&,int_pair);
};

ostream& operator<<(ostream& s,int_pair p){
    s << "(" << p.left << "," << p.right << ")"; //<< endl?
    return s;
}

-UU-:----F1  operator_overloading_stream1.cpp  Top L24  (C++/l Abb
```

# operator<< v.1

A screenshot of a terminal window titled "f6" showing C++ code for operator overloading. The code defines a class `int_pair` with two integer members `left` and `right`. It includes a constructor, a member function `operator[]` to access either member, and a friend function `operator<<` to output the pair to an ostream. The terminal window has a green header bar and a blue status bar at the bottom.

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

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }
    friend ostream& operator<<(ostream&,int_pair);
};

ostream& operator<<(ostream& s,int_pair p){
    s << "(" << p.left << "," << p.right << ")"; //<< end1?
    return s;
}
```

The terminal window shows the following status bar text:

f6 — alfred@ubuntu: ~/cpp\_v2014\_private/f6 — ssh — 69x27  
File Edit Options Buffers Tools C++ Help  
-UU-:----F1 operator\_overloading\_stream1.cpp Top L24 (C++/l Abb)

Ulempet?

Hvis du skriver  
klassen - og  
“vennen” har du  
kontrollen.

...og ansvaret

FØRSTESVNSEN GO...

# operator<< v.2

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x29
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }

    string to_string(){
        return "("+std::to_string(left)+","+
               std::to_string(right)+")";
    }
};

ostream& operator<<(ostream& s,int_pair p){
    s << p.to_string();
    return s;
}

-UU-----F1  operator_overloading_stream2.cpp  Top L26  (C++/l Abb)
```

Alternativ:  
Lag noe public, som eksponerer det du vil, dropp "vennen" og la operator<< bruke dette

# operator<< v.2

```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x29
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;

class int_pair{
    int left;
    int right;
public:
    int_pair(int l,int r):
        left(l),right(r){}
    int operator[](unsigned int i){
        if(i==0)
            return left;
        return right;
    }

    string to_string(){
        return "("+std::to_string(left)+","+
               std::to_string(right)+")";
    }
};

ostream& operator<<(ostream& s,int_pair p){
    s << p.to_string();
    return s;
}

-UU-----F1  operator_overloading_stream2.cpp  Top L26  (C++/1 Abb)
```

Alternativ:  
Lag noe public, som eksponerer det du vil, dropp "vennen" og la operator<< bruke dette

Tryggere nå?  
Hmmm...

Finn et eksempel!

FINN ET EKSEMPEL!

# Når er det nyttig å overlaste operatorer?

- \* Vi har sett operator <<
- \* operator[] i en lenket liste? (Hvorfor ikke!)
- \* bigint i = pow( 2, 64 ); i+=pow(2, 32); ... i\*99
- \* myCar < yourCar? (Nødvendig for sortering!)
- \* operator() ...?
  - \* class cleverMath ... cleverMath f; f(x)...
  - \* Funksjonsobjekter!
- \* Flere?
- \* Bruk operatorer!

Operatorer er ulike!  
Slå opp her:

---

[http://en.wikipedia.org/wiki/C%2B%2B\\_operators](http://en.wikipedia.org/wiki/C%2B%2B_operators)

# Demo:

---

`operator_overloading[].cpp`  
`operator_overloading_stream1.cpp`  
`operator_overloading_stream2.cpp`  
`(operator_overloading_increment.cpp)`