

Arv og operatoroverlasting

Forelesning 6,
Effektiv kode med C og C++, vår 2015
Alfred Bratterud

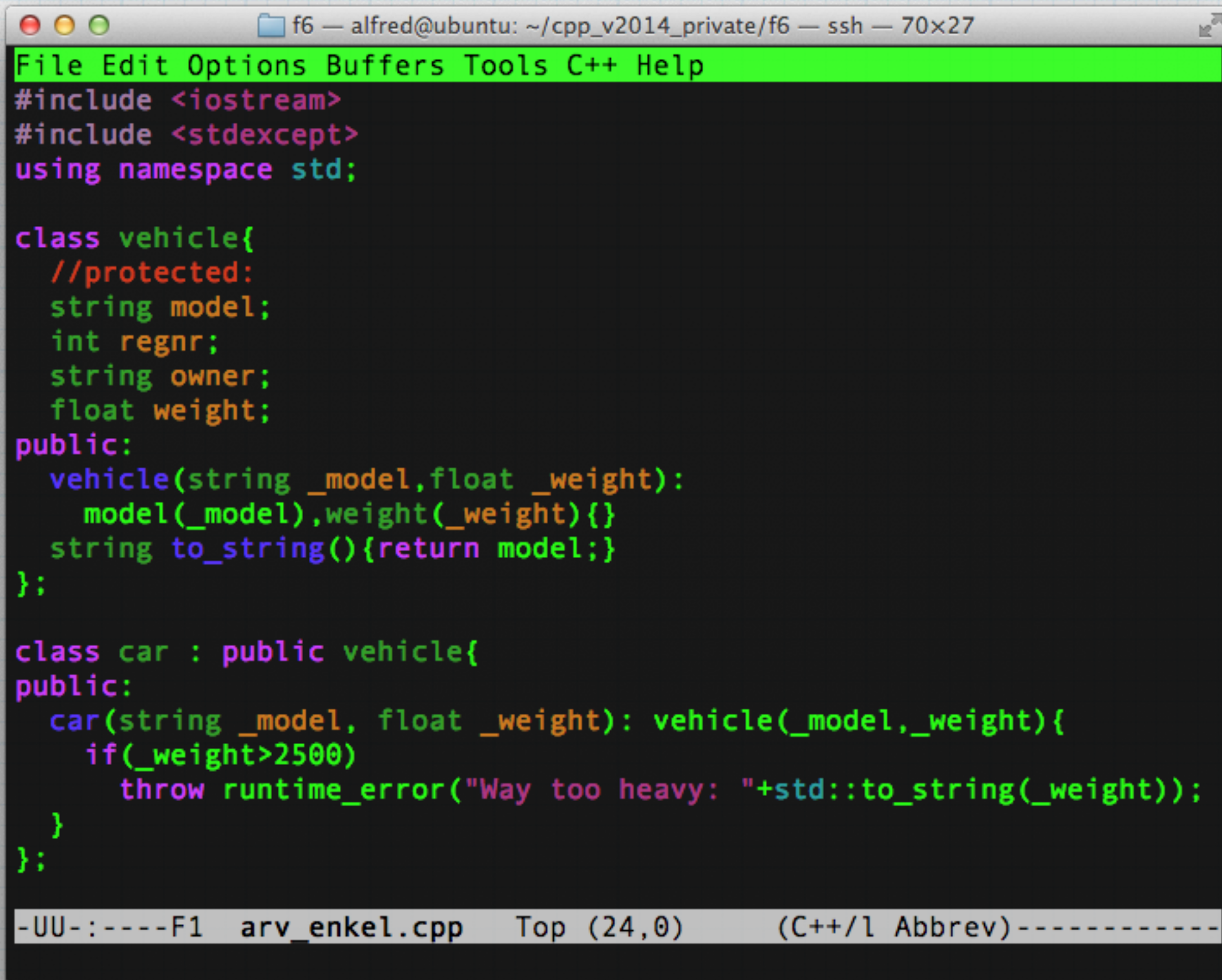
Agenda:

- * Går det greit med oblige?
- * Oppdater linken din!
- * 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! (20% for minibuss.cpp)
 - * Er det så viktig?
 - * Færre duplikater, færre feil.
- * Mest brukt: legge til nye egenskaper i eksisterende klasser:
 - * String med stringsplit?
 - * Invalid_XML_exception?

Arv



The image shows a screenshot of a C++ IDE window. The title bar indicates the file is 'f6' and the user is 'alfred@ubuntu'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C++', and 'Help'. The code defines a 'vehicle' class with attributes 'model', 'regnr', 'owner', and 'weight', and a 'car' class that inherits from 'vehicle'.

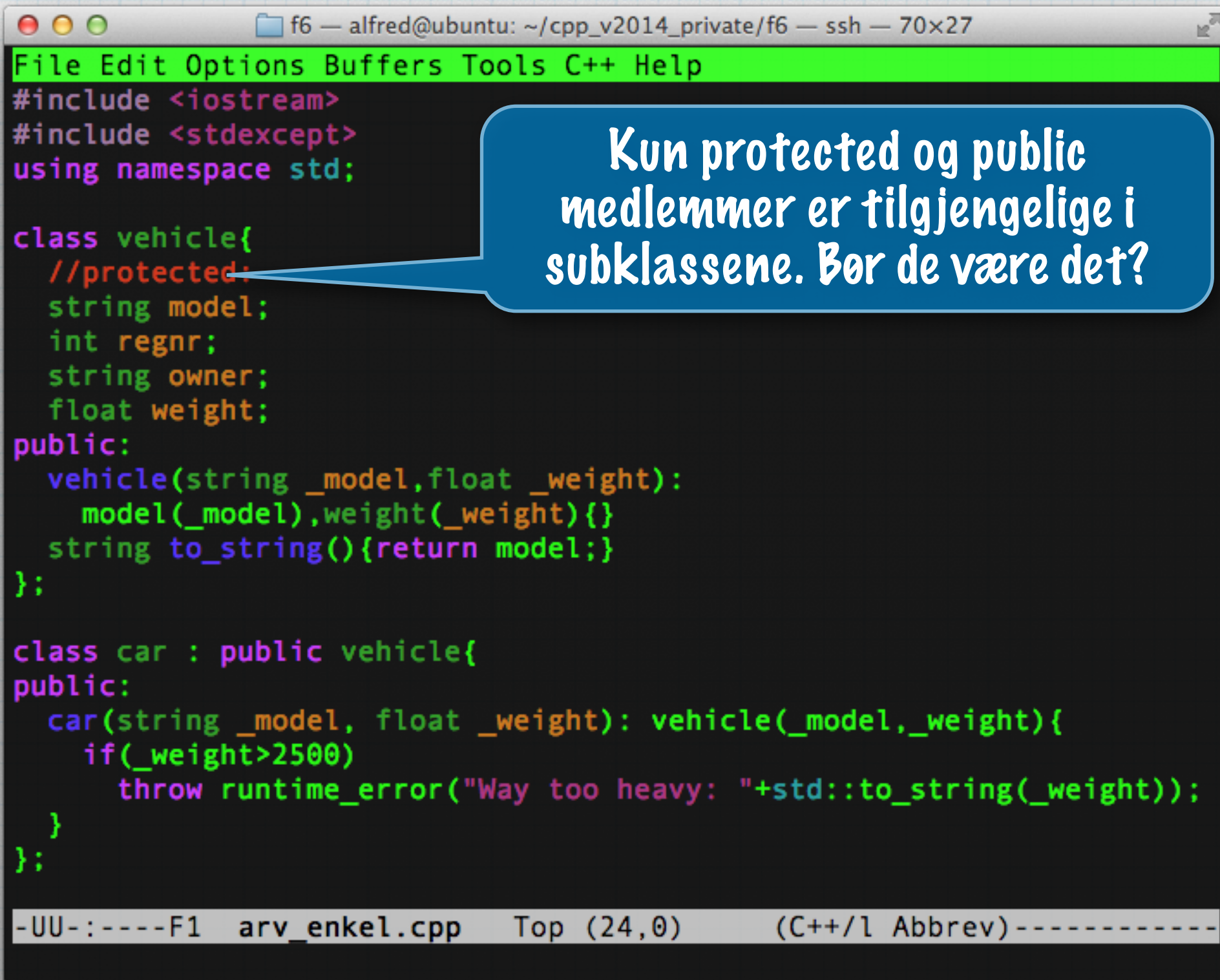
```
f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 70x27
File Edit Options Buffers Tools C++ Help
#include <iostream>
#include <stdexcept>
using namespace std;

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++/1 Abbrev)-----
```


Arv



The screenshot shows a C++ IDE window titled 'f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 70x27'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C++', and 'Help'. The code defines a `vehicle` class with private members `model`, `regnr`, `owner`, and `weight`. It has a public constructor and a `to_string` method. A `car` class inherits from `vehicle` and overrides the constructor to throw an error if the weight is greater than 2500. A blue callout box points to the `//protected:` comment in the `vehicle` class, containing the text: 'Kun protected og public medlemmer er tilgjengelige i subclassene. Bør de være det?'. The status bar at the bottom shows '-UU-:----F1 arv_enkel.cpp Top (24,0) (C++/l Abbrev)-----'.

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

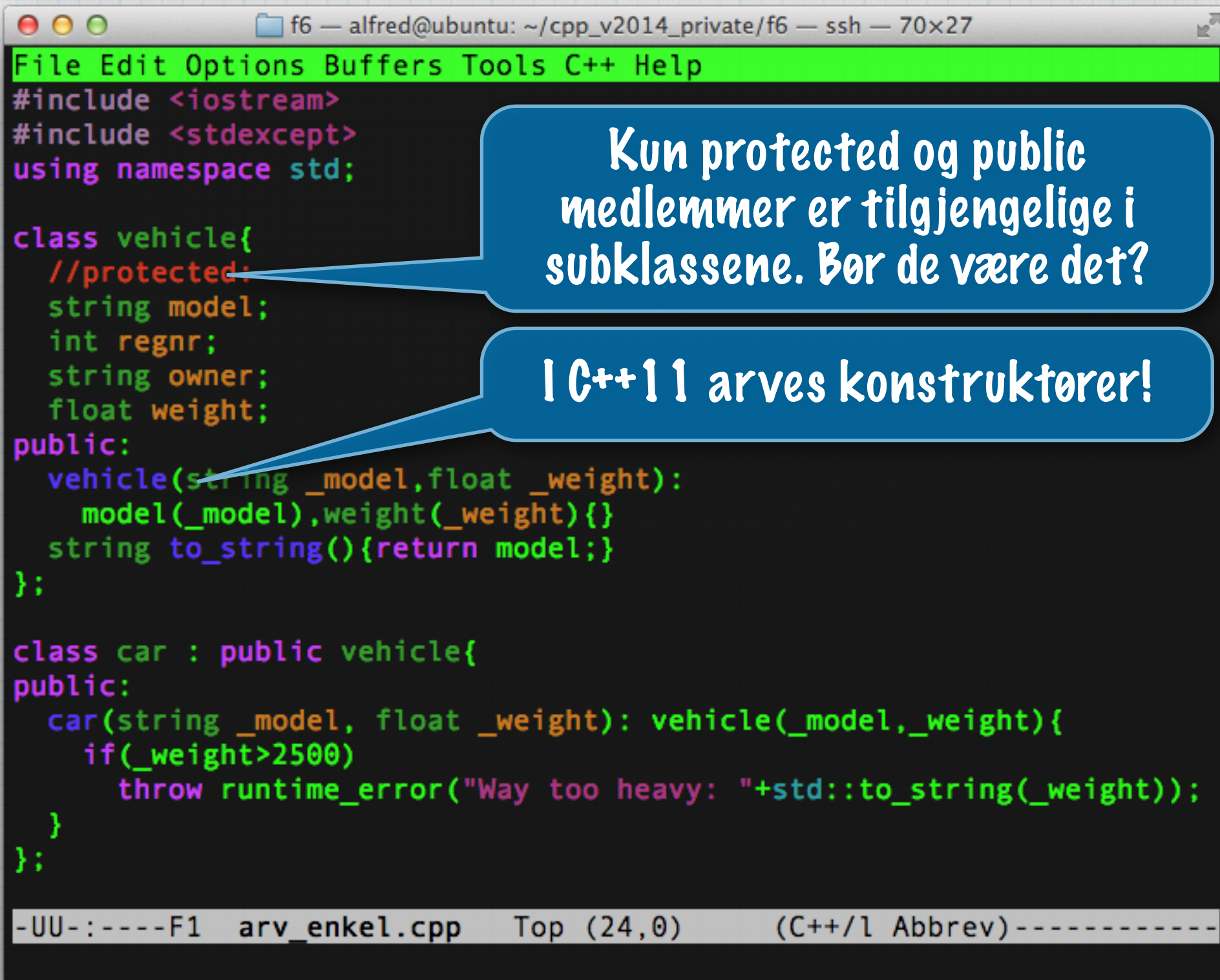
class vehicle{
    //protected:
    string model;
    int regnr;
    string owner;
    float weight;
public:
    vehicle(string _model, float _weight):
        model(_model), weight(_weight){}
    string to_string(){return model;}
};

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));
    }
};
```

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

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

Arv



```
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));
    }
};
```

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

I C++11 arves konstruktører!

-UU-:----F1 arv_enkel.cpp Top (24,0) (C++/1 Abbrev)-----

Arv

```
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));
    }
};
```

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

I C++11 arves konstruktører!

klasse : base - ":" betyr "arver"

-UU-:----F1 arv_enkel.cpp Top (24,0) (C++/1 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: "+string(_weight));
    }
};
```

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

I C++11 arves konstruktører!

klasse : base - ":" betyr "arver"

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

-UU-:----F1 arv_enkel.cpp Top (24,0) (C++/1 Abbrev)-----

Arv

- * “**Protected:**” gir tilgang kun til subklasser. Bjarne kaller dette “most likely a design flaw”.
 - * Hvorfor blottlegge de private delene av baseklassen?
 - * Vi må uansett 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);} Oops!
};

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

...

...


```
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 weight;
    string _model;
    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);} Oops!
};

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)-----
```

OBS: navnekollisjon
kompilerer ikke

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

Nyttig for:
scrollable_window
framed_picture
two_way_door

...

...

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 ... det var ikke mulig å garantere, slik «man skulle tro»*
 - * Herb Sutter forklarer: <http://www.gotw.ca/publications/mill22.htm>
- * Skal du compilere en exception-subklasse i C++11 må du ha med destructor som “lover å ikke kaste noe”.
- * I C++11 er dette tatt bort - men «noexcept» er et alternativ
 - * ...Men dette er ikke for at kompilatoren skal hjelpe deg, men for at du skal hjelpe kompilatoren.
- * Generelt: Bruk C++11 og arv av exception fungerer som forventet :-)


```
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++/1 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++/1 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++/1 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(filename);
    }catch(file_not_found_exception fne){
        cout << "File not found: " << fne.what() << endl;
    }catch(exception e){
        cout << "Exception: " << e.what() << endl;
    }
}
```

-UU-: **--F1 arv_exceptions.cpp All L29 (C++/1 Abbrev)----

Vi arver "exception" som vanlig

Legger til constructor som tar streng

C++03: 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++/1 Abbrev)----
```

Men som forventet
i C++11


```
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++/1 Abbrev)----
```

Men som forventet
i C++11

Enklest?

```
class VehicleException : public runtime_error {  
public: VehicleException(string s) : runtime_error{s}{}  
};
```


operatoroverlasting

- * Vi sa at datatyper innebærer «semantikk» - og regler for hvordan operatorene skal tolkes... (a+b ??)
- * Hva er egentlig en operator?
 - * Eksempler er +, -, ==, +=, =, ->, * etc.
 - * ...Akkurat som en funksjon eller medlemsfunksjon, men gjerne med «infix» syntaks
 - * **a+b** vs. **+(a,b)** ... eller **add(a,b)** - eller **a.add(b)**
 - * **a<<b** vs. **<<(a,b)** ...eller **a.PushToStream(b)**
 - * Hva er returtypen og «semantikken» til de over?
- * I mange språk gjelder operatorene kun forhåndsdefinerte typer. Men hvis vi får definere egne typer - bør vi ikke få definere semantikken?
- * I C++ kan de fleste operatorene defineres på nytt, for nye typer. Generell form:
- * **type** operator**symbol** (**parametre**) { ... }

Operatorer som kan overlastes

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

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

Operatorer som kan overlastes

Funksjonsobjekter!

```
class MyFunction {  
    int operator()(int x) { ... };  
}  
MyFunction f;  
f(10);
```

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

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

* **type** operator**symbol** (**parametre**) { ... }

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 operator symbol (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.

Argumentrekkefølge teller

- * Vi kan da bruke en frittstående funksjon
- * `type operator symbol (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++/1 Abb

Strømoperatoren kan ikke overlastes som medlem...

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++/1 Abb

Strømoperatoren kan ikke overlastes som medlem...

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;
}

-UU-:----F1  operator_overloading_stream1.cpp  Top L24  (C++/1 Abb
```

Strømoperatoren kan ikke overlastes som medlem...

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

I stedet kan vi definere den eksternt

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;
}
```

-UU-:----F1 operator_overloading_stream1.cpp Top L24 (C++/1 Abb

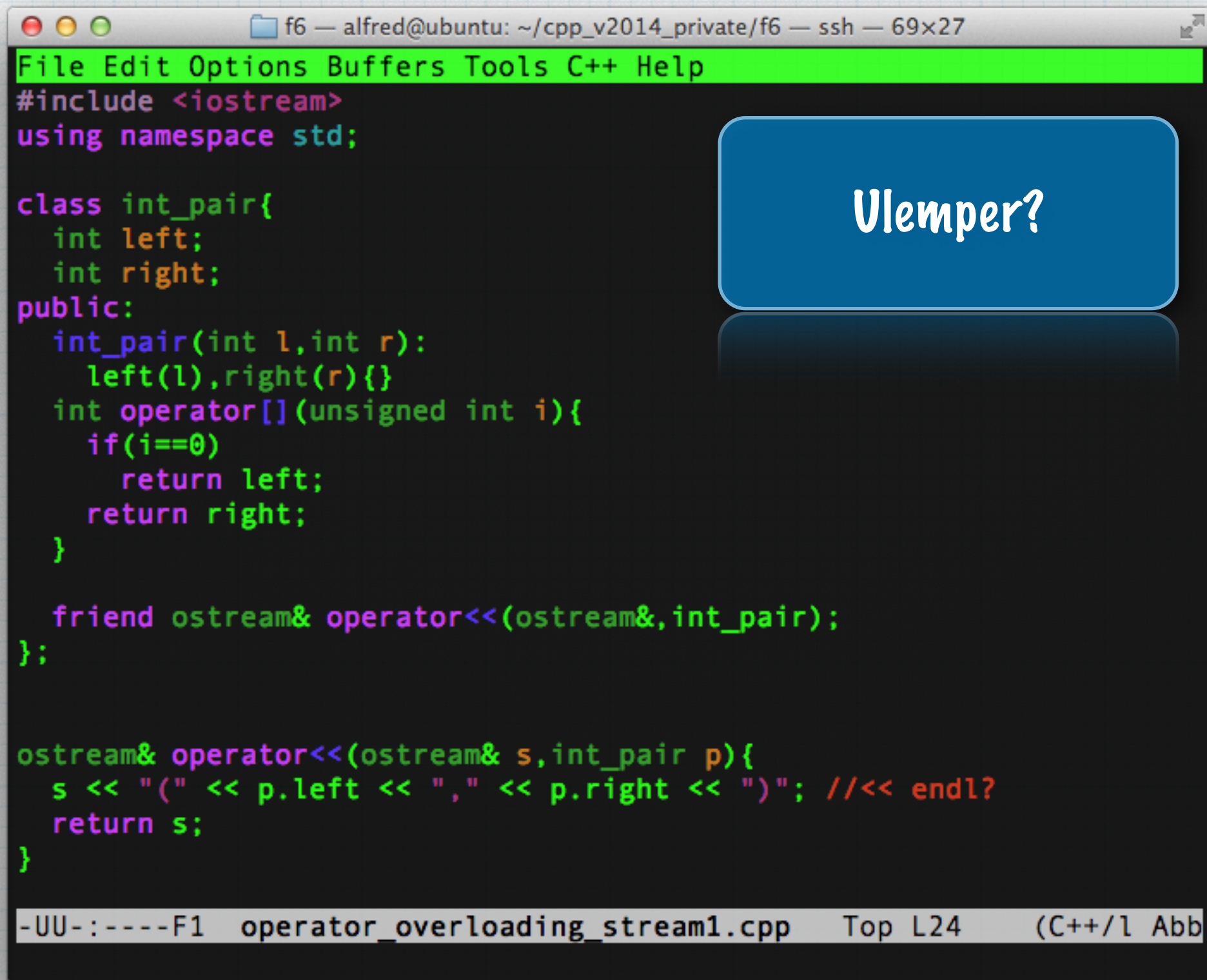
Strømoperatoren kan ikke overlastes som medlem...

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

operator<< v.1



The image shows a screenshot of a C++ IDE window titled "f6 — alfred@ubuntu: ~/cpp_v2014_private/f6 — ssh — 69x27". The window has a menu bar with "File", "Edit", "Options", "Buffers", "Tools", "C++", and "Help". The code is written in a dark-themed editor with syntax highlighting. A blue rounded rectangle with the text "Ulemper?" is overlaid on the right side of the code. The code defines a class `int_pair` with `left` and `right` attributes, a constructor, and an `operator[]` function. It also defines a `friend ostream& operator<<(ostream&, int_pair);` and a function `ostream& operator<<(ostream& s, int_pair p)` that formats the pair as `"(left , right)"`. The status bar at the bottom shows `-UU-:----F1 operator_overloading_stream1.cpp Top L24 (C++/1 Abb`.

```
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++/1 Abb
```


operator<< v.1



```
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);
};

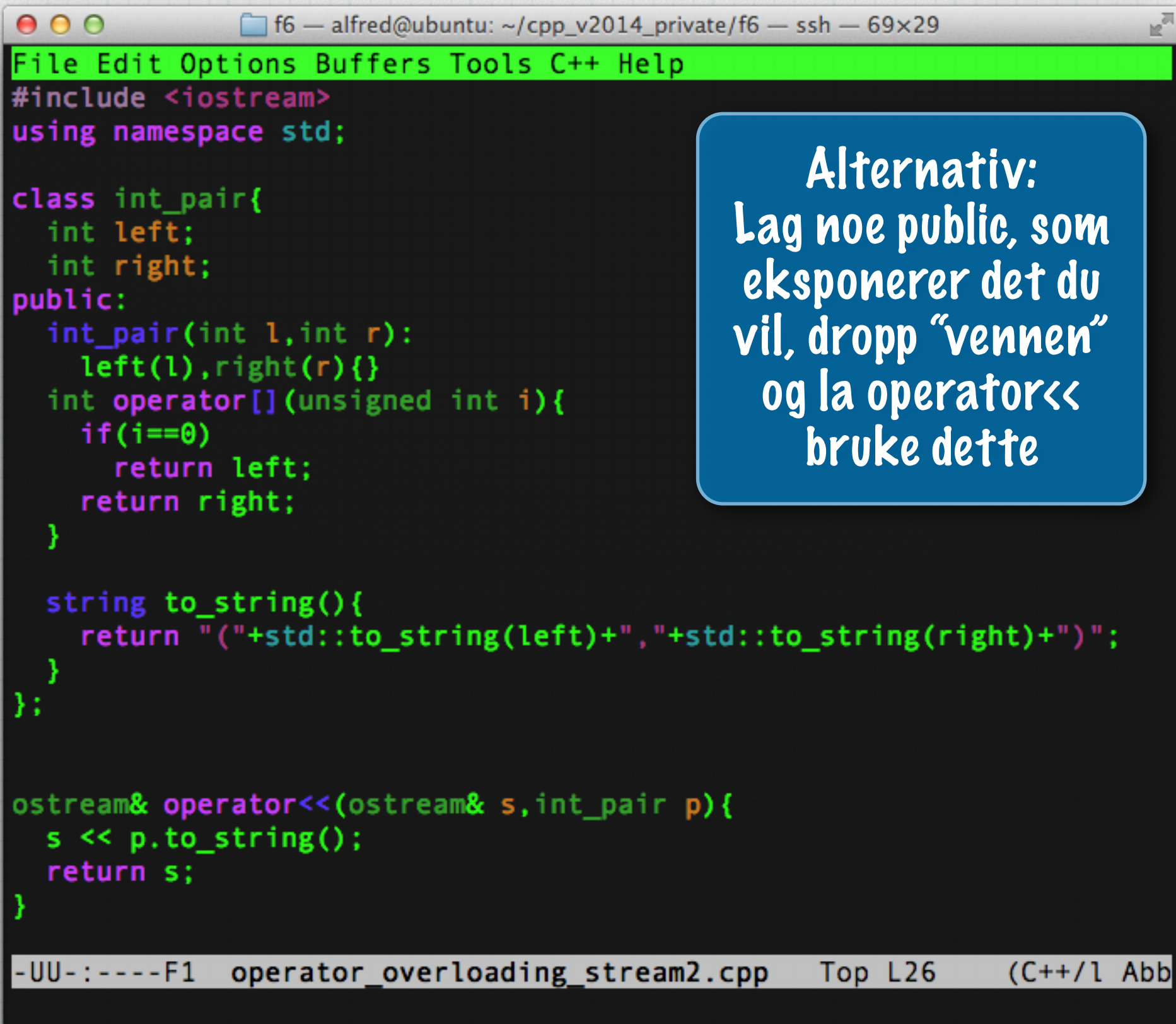
ostream& operator<<(ostream& s,int_pair p){
    s << "(" << p.left << "," << p.right << ")"; //<< endl?
    return s;
}
```

Ulemper?

Hvis du skriver klassen - og "vennen" har du kontrollen.
...og ansvaret

-UU-:----F1 operator_overloading_stream1.cpp Top L24 (C++/1 Abb

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;
}
```

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

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


operator<< v.2

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;
}
```

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

Tryggere nå?
Hmmm...

Finn et eksempel!

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

Når er det nyttig å overlaste operatører?

- * Vi har set 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 operatører!

Operatorer er ulike!
Slå opp her:

http://en.wikipedia.org/wiki/C%2B%2B_operators

Demo på nett:

minibuss.cpp
protected.cpp
(operator_overloading[1].cpp)
(operator_overloading_increment.cpp)