Subtype Relationship in C++

The „*chevron shape*” inheritance

Problem decomposition

- ***Build & combine*** **reusable components**
- **Most problems have several *dimensions***
- **We are good at <u>single</u> dimension decomposition**
- ***<u>Conflicting decompositions</u>***

Design principles

- Generic programming
- Policy/strategy pattern
- Component based design

- Templates
- Inheritance, virtual functions
- CRTP *(static polimorphism, F-bounded)*
- Multiple inheritance, mixins
- Type-erasure, overloading

Why inheritance

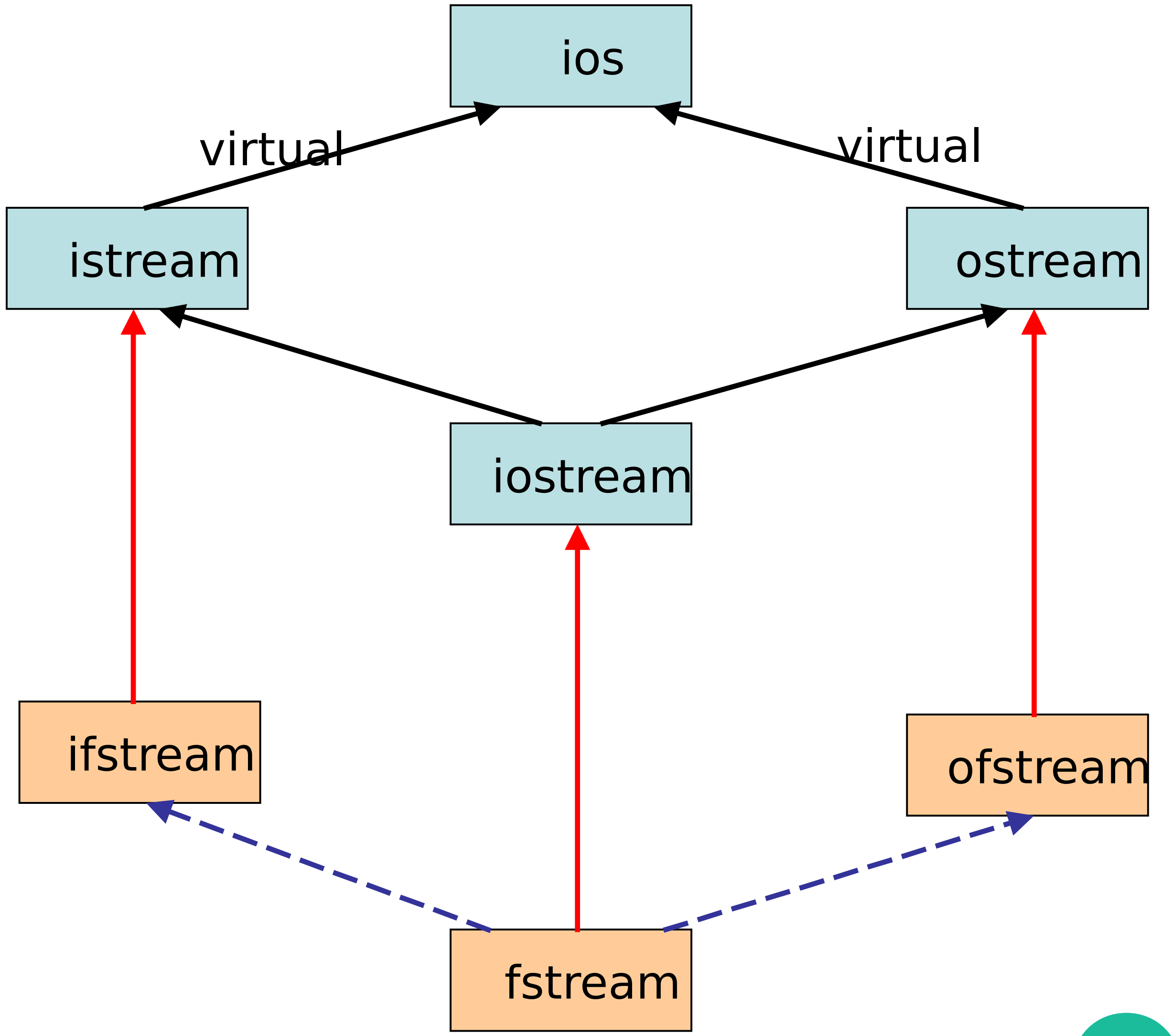
Goal

- **code reuse**
- **loosely coupled code**
- **extensible**
- **readability**
- **performance**
- **value semantics** ***(eg. vector-of-pointers)***

Inheritance vs tagged unions

inheritance	variant/union
Type set: open	Type set: closed
Operation set: closed	Operation set: open

Chevron-shape inheritance



Source: Free Icons Library

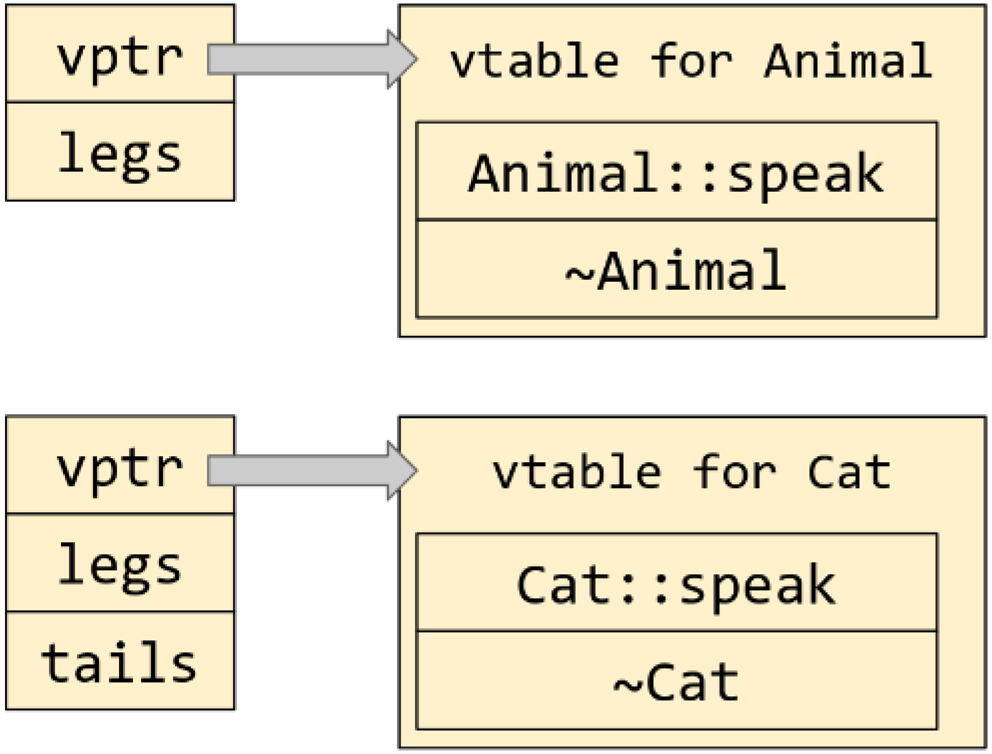
Multiple inheritance

“Multiple inheritance is like a parachute; you don’t need it very often, but when you do it is essential” - (Grady Booch, 1991)

- **Name collisions?**
- **Diamond shape?**
- **Runtime implications?**
- **Partial override of the virtual interface**
- **You don't pay for what you don't use?**

Quick recap: Polimorphism

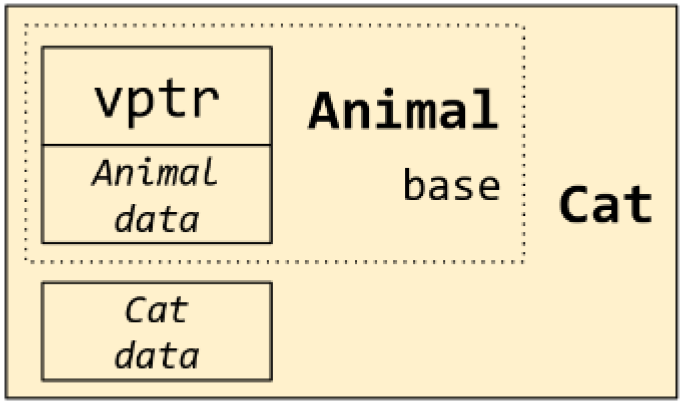
```
class Animal {
public:
  int legs;
  virtual void speak() { puts("hi"); }
  virtual ~Animal();
};

class Cat : public Animal {
public:
  int tails;
  void speak() override {
    printf("Ouch, my %d tails!",
              tails);
  }
};
```



```
# a->speak();
movq (%rdi), %rax
callq *(%rax)
```

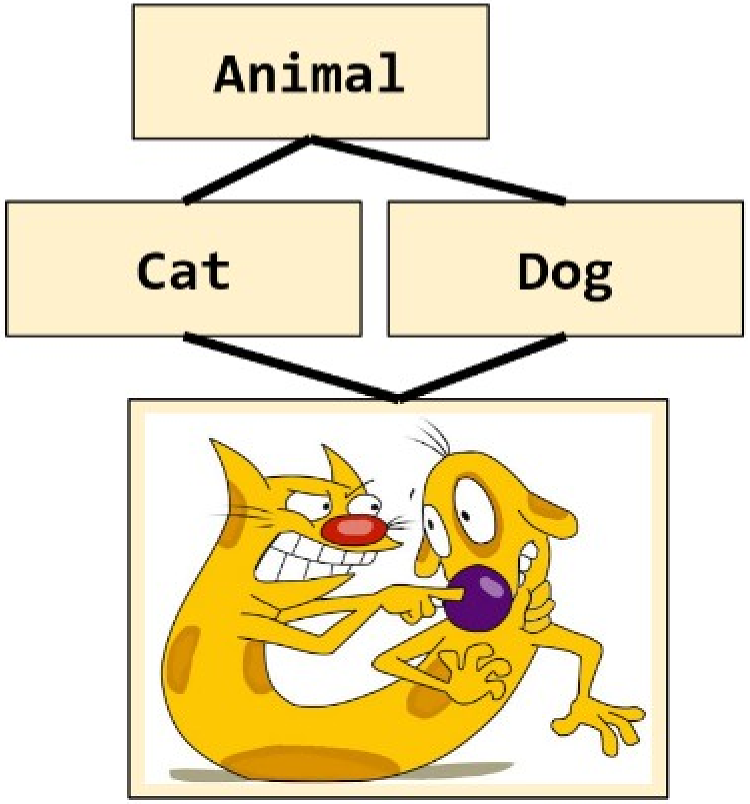
Memory layout

Cat **IS-AN** Animal

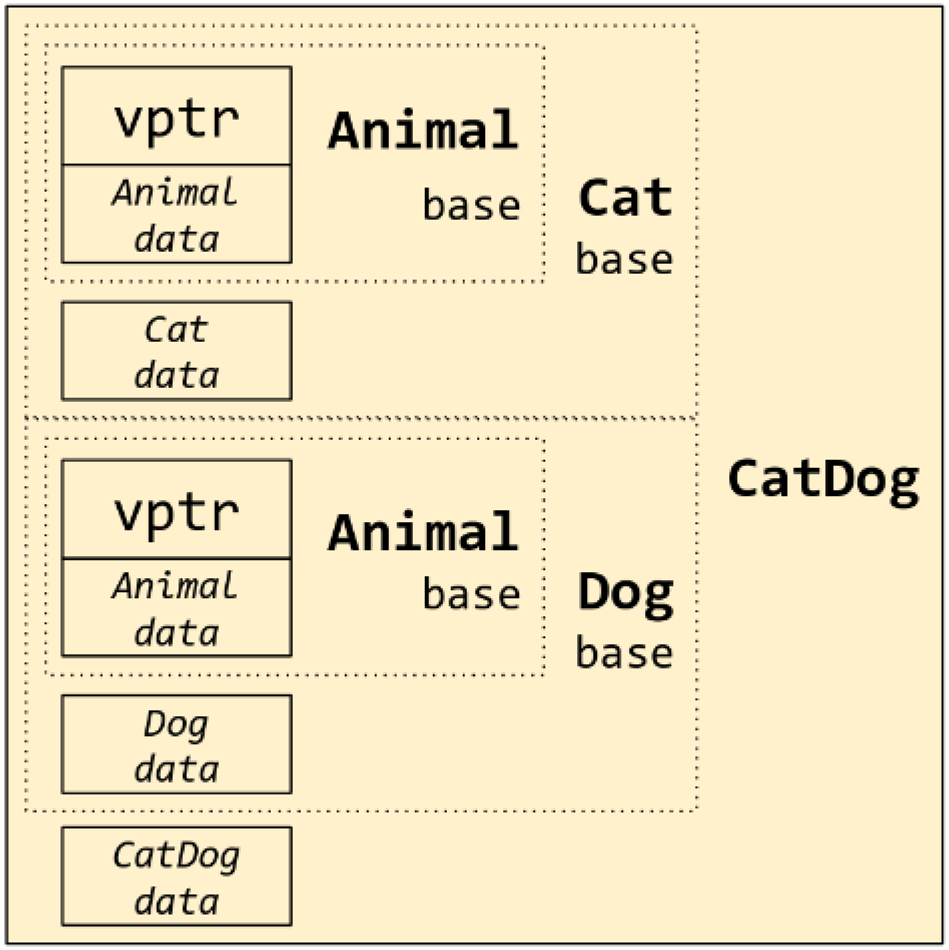


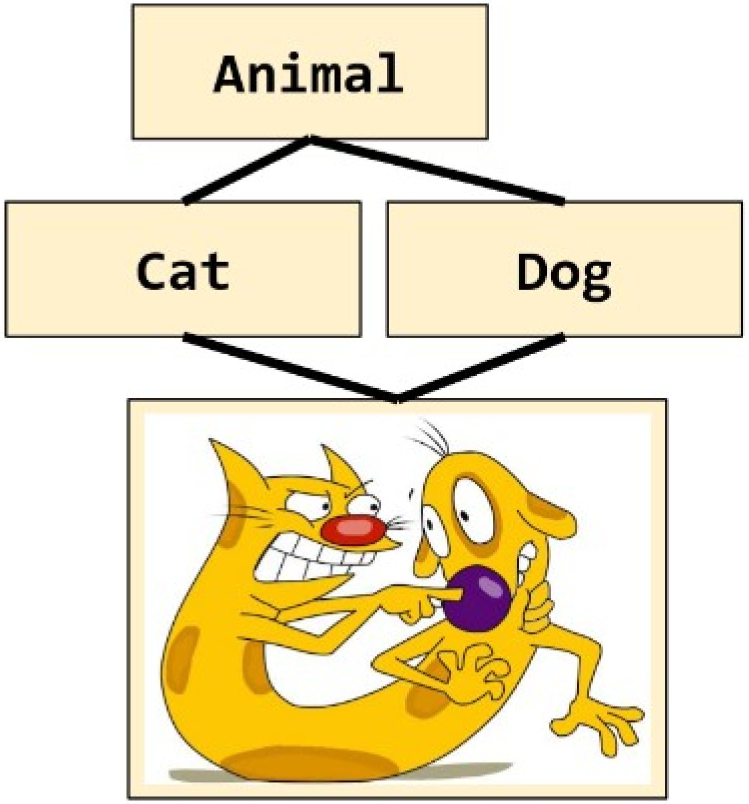
Multiple inheritance: CatDogs

```
class Animal {
  virtual ~Animal();
};

class Cat : public Animal { };

class Dog : public Animal { };

class CatDog :
  public Cat, public Dog { };
```

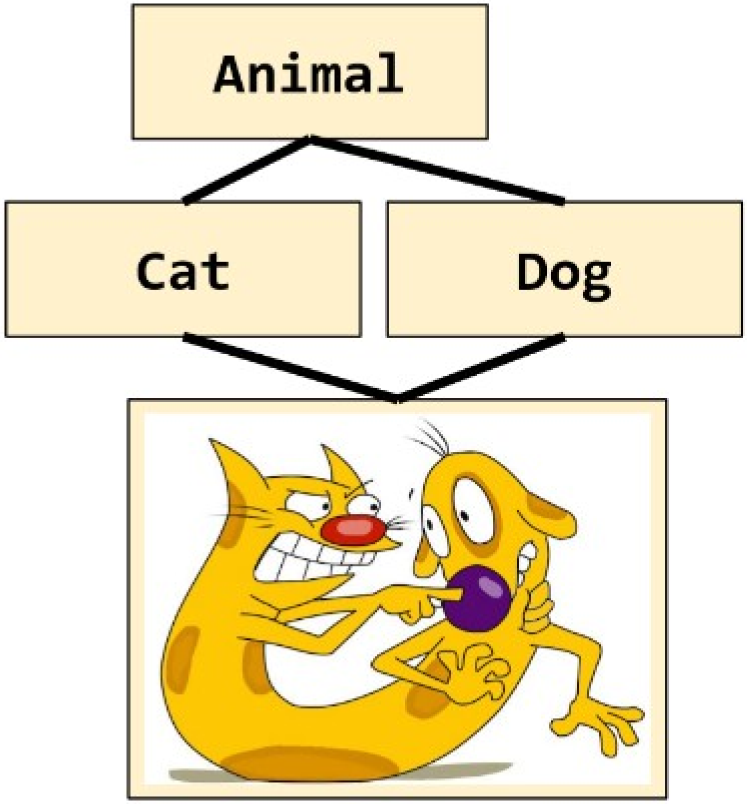


Multiple inheritance: CatDogs



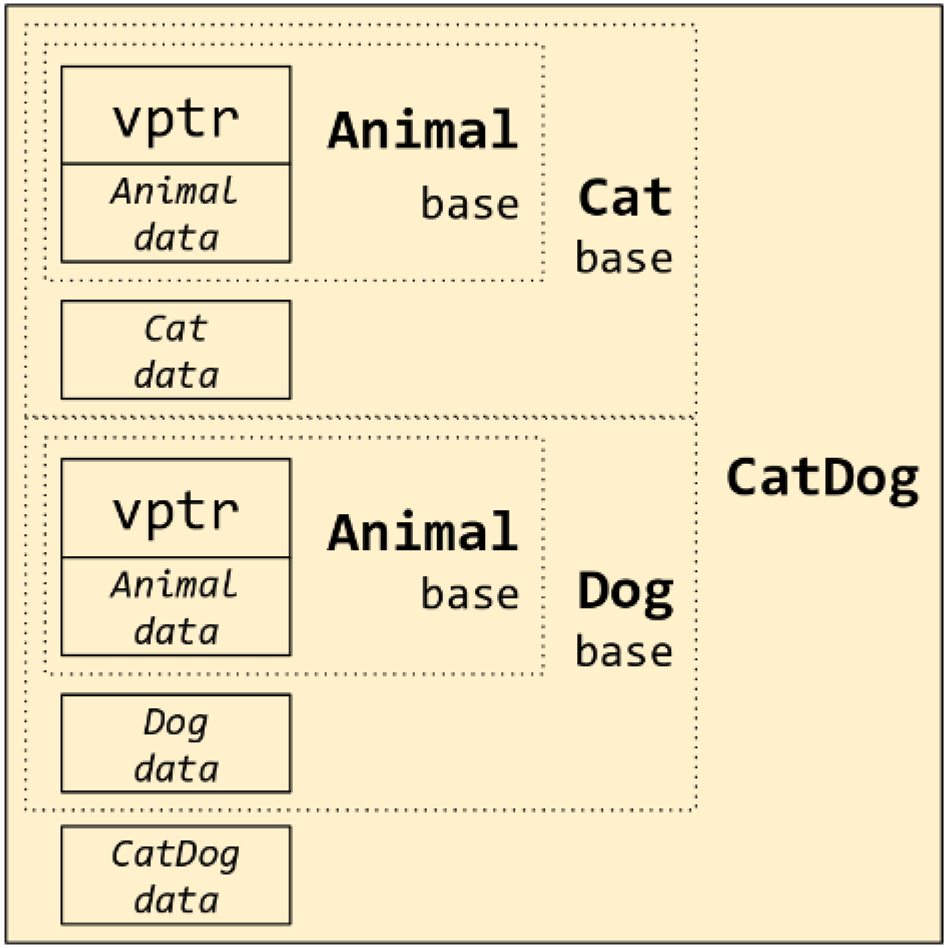


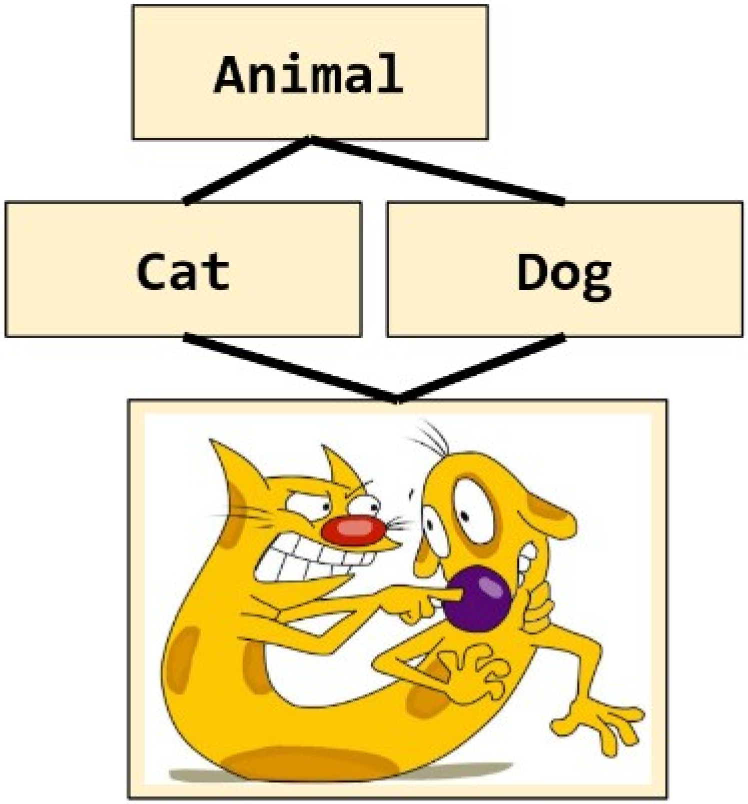
IS-A CatDog an Animal?



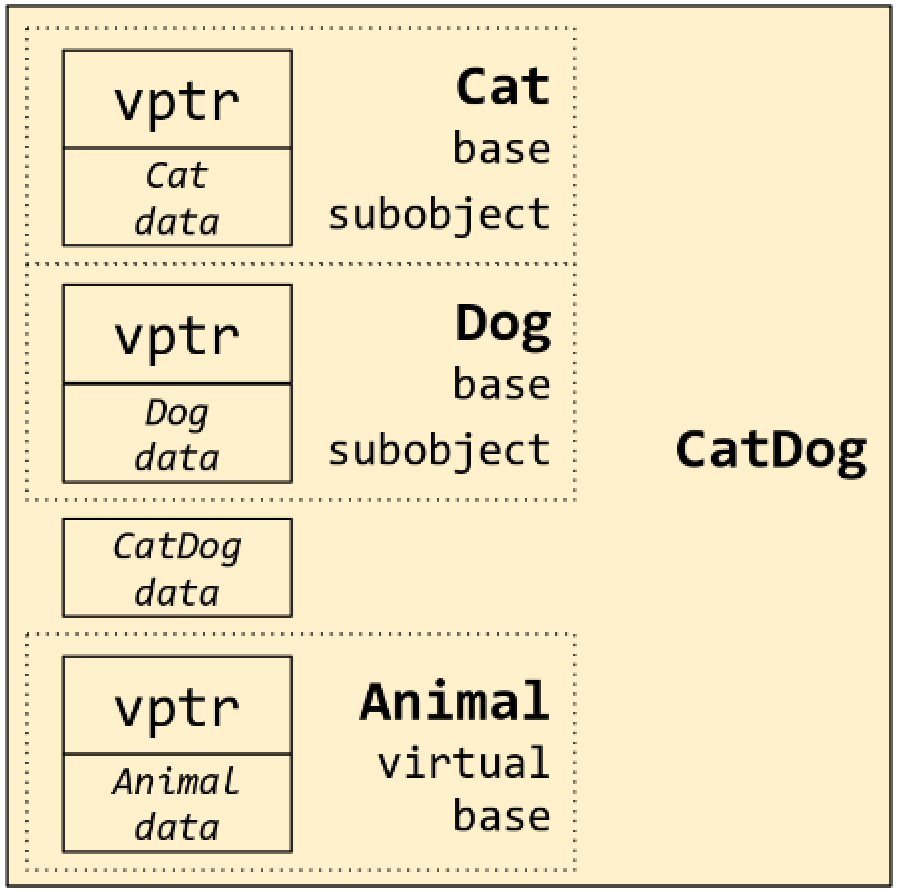
IS-A CatDog an Animal?

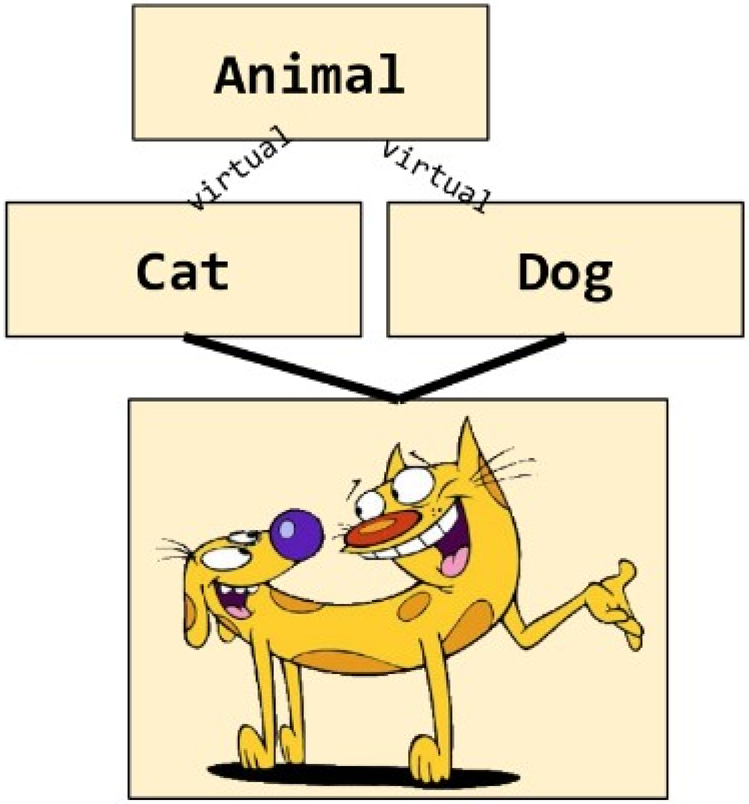
No, it's **two** Animals.



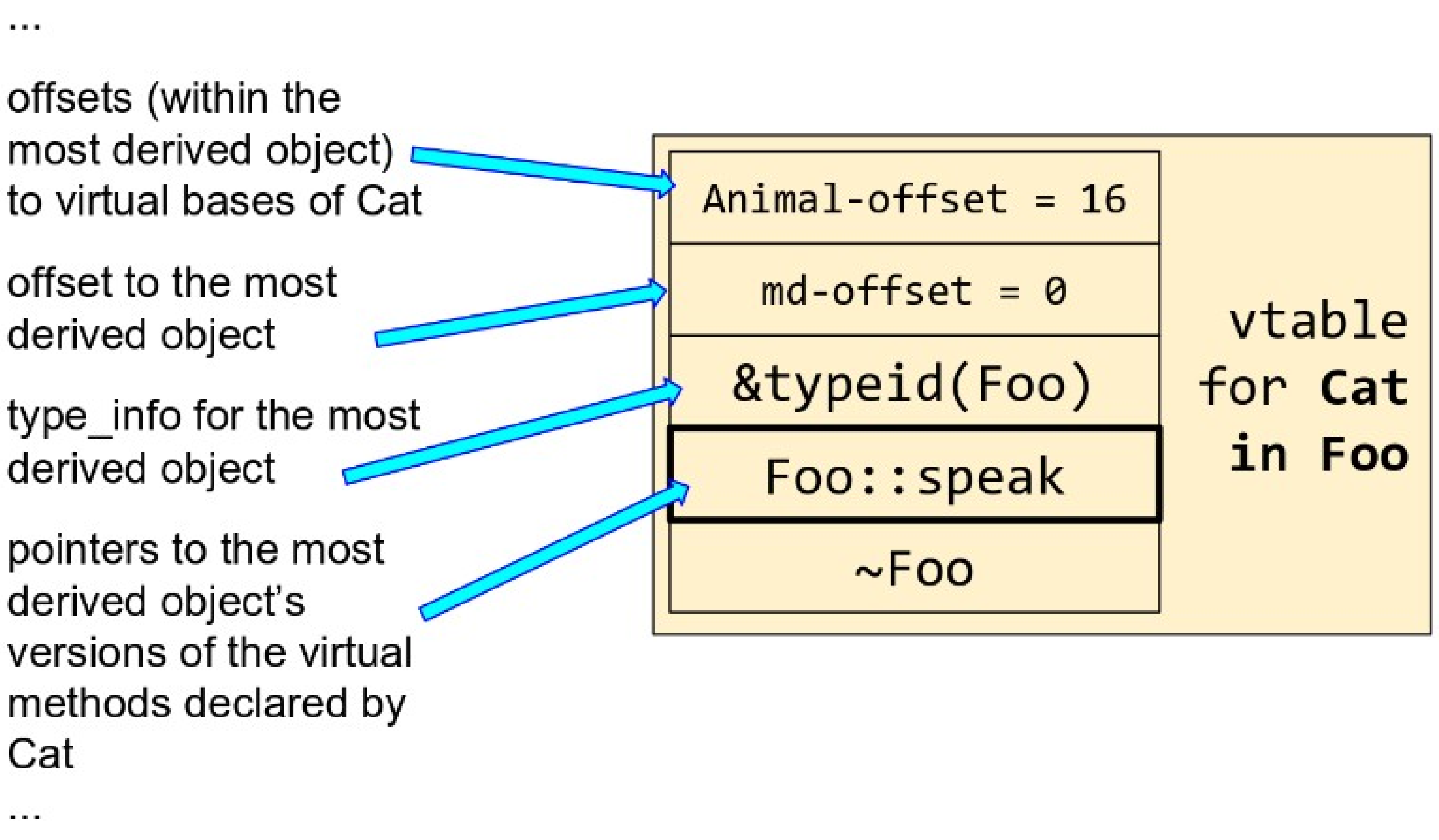


Diamond-shape inheritance





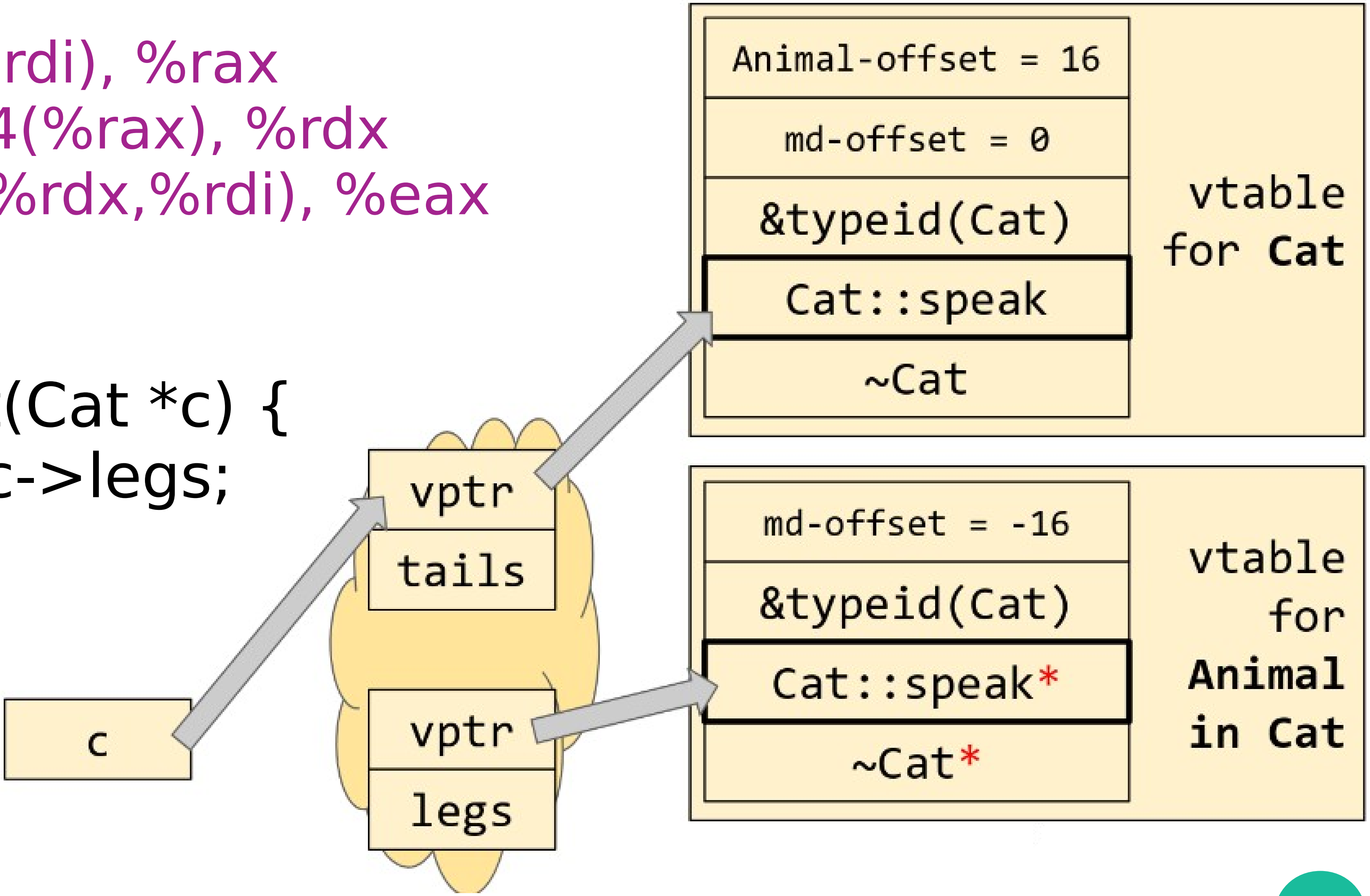
Vtable layout recap *(Itanium ABI)*


...
offsets (within the most derived object) to virtual bases of Cat
offset to the most derived object
type_info for the most derived object
pointers to the most derived object's derived object's versions of the virtual methods declared by Cat
...
Animal-offset = 16
md-offset = 0
&typeid(Foo)
Foo::speak
~Foo
vtable for Cat in Foo

Access a member of a virtual base

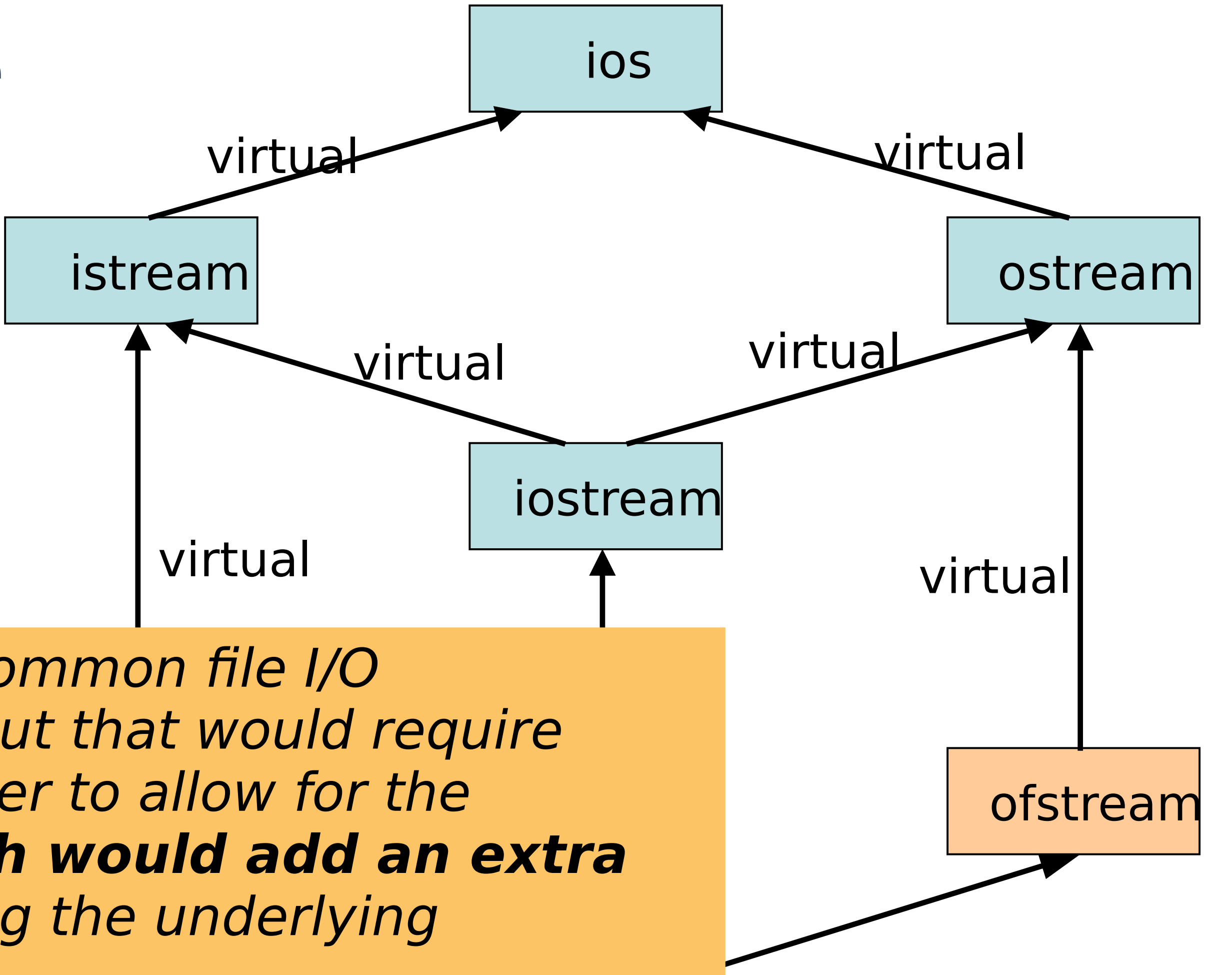
```
movq (%rdi), %rax
movq -24(%rax), %rdx
movq 8(%rdx,%rdi), %eax
```

```
void test(Cat *c) {
  return c->legs;
}
```



What if **fstream** inherited... *everything*

Probably worse performance?

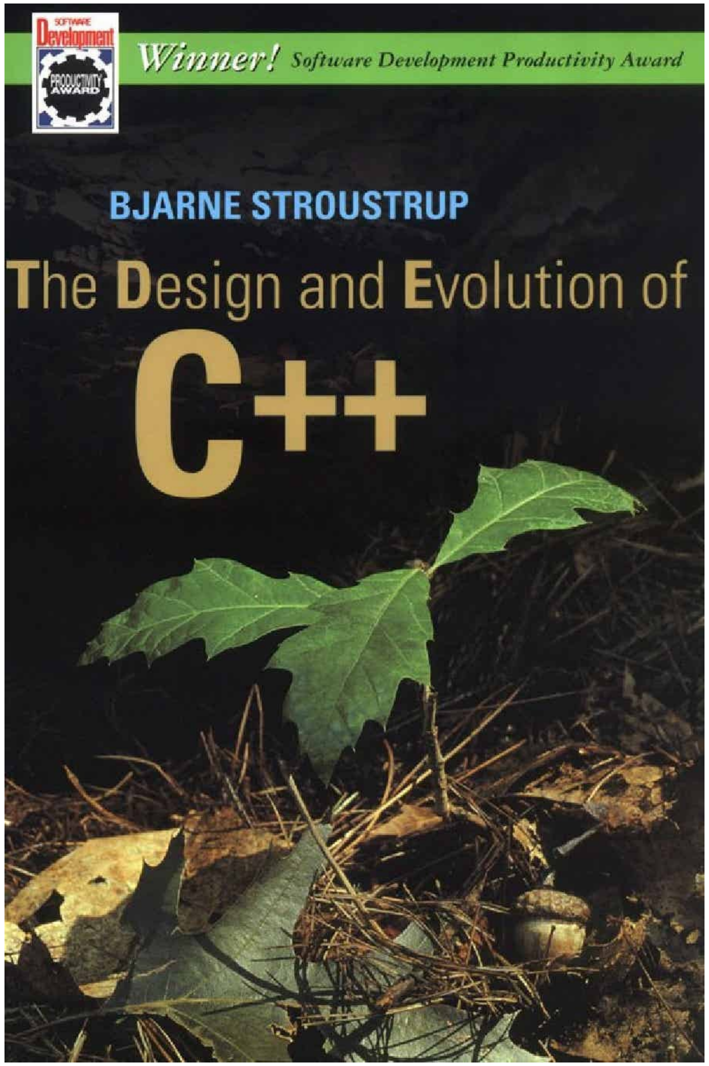


*I guess you could extract common file I/O functionality into a mixin, but that would require **virtual inheritance** in order to allow for the diamond inheritance, **which would add an extra indirection** when accessing the underlying `basic_filebuf`.*
- stackoverflow (rustyx)

Resources

- Zolyomi, Istvan & Porkoláb, Zoltán & Kozsik, Tamás. (2003)
  An Extension to the Subtype Relationship in C++ Implemented with Template Metaprogramming. Lecture Notes in Computer Science. 10.1007/978-3-540-39815-8_13.
- CppCon17 Arthur O'Dwyer "dynamic_cast From Scratch"
- CppCon19 John Bandela "Polymorphism != Virtual: Easy, Flexible Runtime Polymorphism Without Inheritance"
- ACCU18 Louis Dionne Runtime Polymorphism: Back to the Basics
- GoingNative13 Sean Parent Inheritance Is The Base Class of Evil
- Bjarne Stroustrup The Design and Evolution of C++
- J. E. Shopiro An example of multiple inheritance in C++: a model of the iostream library
- Harold Ossher and Peri Tarr Multi-Dimensional Separation of Concerns andThe Hyperspace Approach
- stackoverflow 'Inaccessible direct base' caused by multiple inheritance
- stackoverflow Why fstream is not inherited from ifstream and ofstream in c++?

The Design and Evolution of C++


SOFTWARE Development
PRODUCTIVITY AWARD
Winner! Software Development Productivity Award
BJARNE STROUSTRUP
The Design and Evolution of
C++

- **Published in 1994**
- **Still highly relevant**
- **Even discusses multimethods**
- **And many more...**
- **Must have book**

** I don't get any benefit from advertising this book.*