Implementations of Parameterization

C templates and Modula-3 generics:

- Similar to macros
- Templates are expanded with parameters
- No code sharing

CLU and ML:

- Everything is an object
- Parameterized types can place restrictions on parameter types
- Code sharing, but extra dispatch

How does this affect separate compilation?

Parameterized/Generic Types

```
stack = class [T:type] exports create, push, pop
   create = proc () returns (stack[T])
   push = proc (s:stack[T], x:T)
   pop = proc (s:stack[T]) returns (T)
end stack
```

How does `stack[int]` relate to `stack[bool]`?

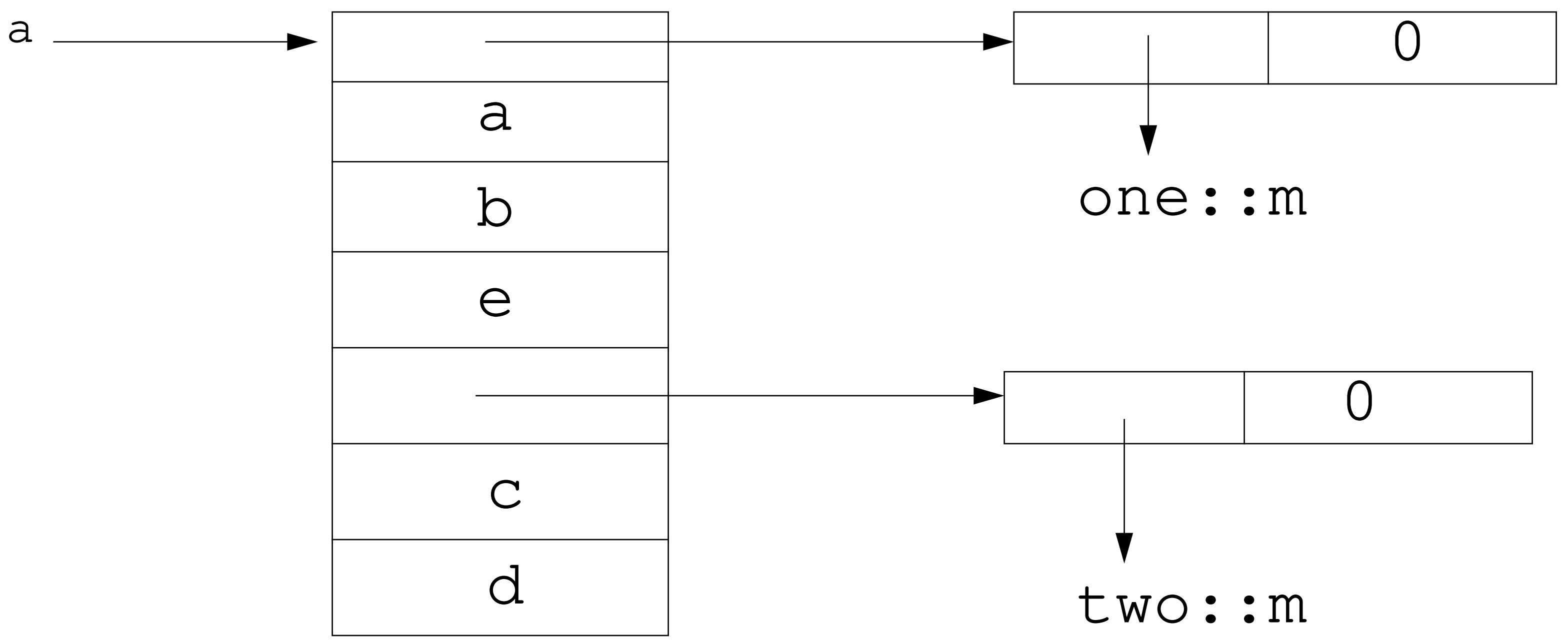
Optimizing Dispatch

Dispatch is expensive!

- Static optimization
- Dynamic optimization

Different layouts

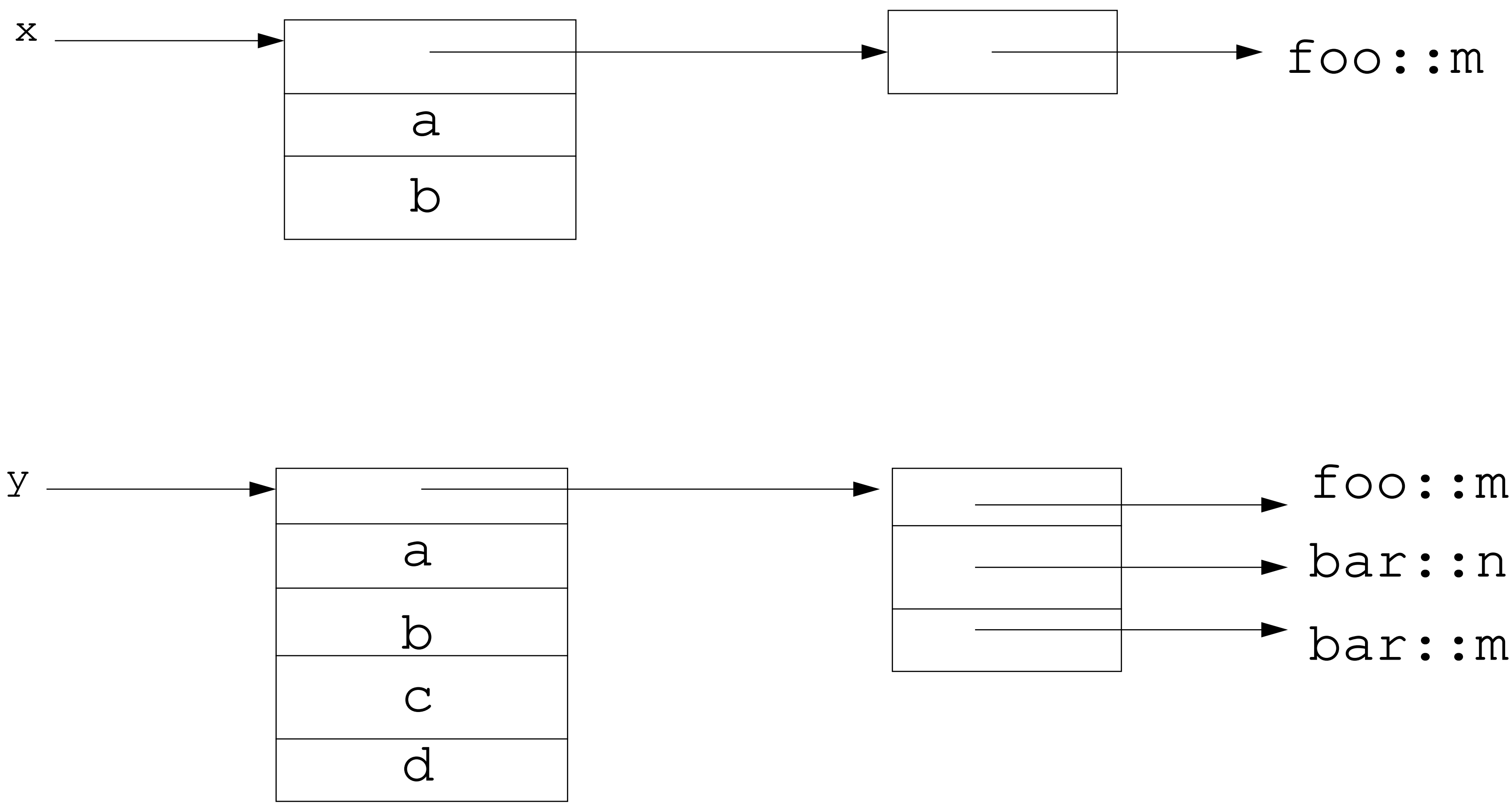
Multiple Inheritance Layout



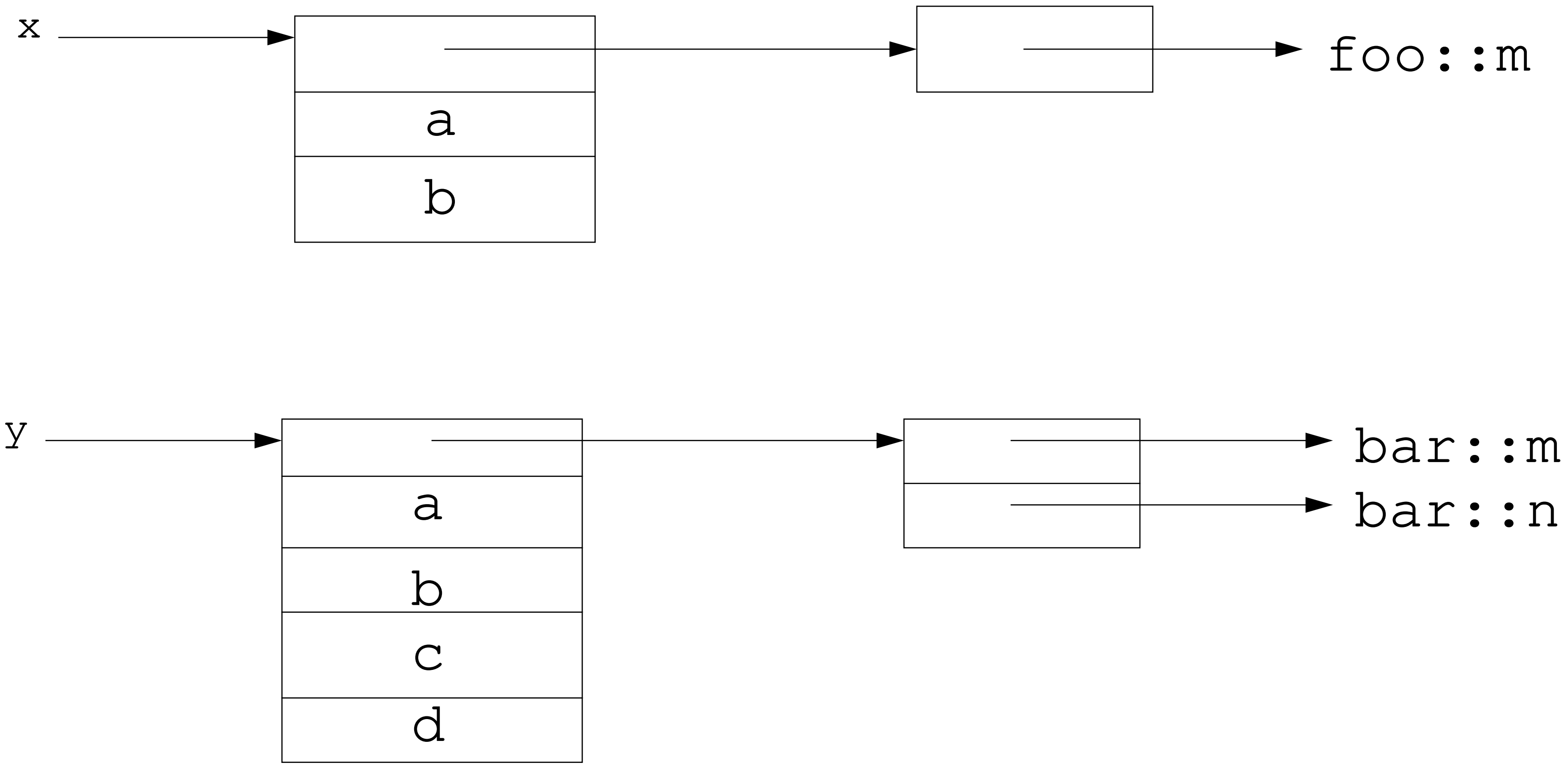
Multiple Inheritance

```
class one {
   public: virtual void m();
   protected: int a; int b;
};

class two {
   public: virtual void m();
   protected: int c; int d;
};

class three : public one, two {
   protected: int e;
};

three *a = new three();
```

Non-Virtual Methods



Runtime Layout



Single Inheritance

```
class foo {
   public: virtual void m();
   protected: int a; int b;
};

class bar : public foo {
   public:
      virtual void m();
      virtual void n();
   protected: int c; int d;
};

foo *x = new foo();
bar *y = new bar();
```

Static vs. Dynamic Typing

Similar to lexical vs. dynamic scoping

When do we know what methods an object supports?

Static typing:

- less expensive
- less flexible
- C++, Modula-3

Dynamic typing:

- more expensive
- more flexible
- Smalltalk

What Is An Object?

Closure : procedure(s) along with environment

environment : bindings for data reference in procedure

procedure : code to execute

Objects respond to *messages:*

map from method names to procedures

Subtyping vs. Inheritance

Subtyping: ability to create a T' that can be used as a T

Inheritance: ability to create a T' that uses code from T

C++: public versus private inheritance

Most languages tie the two together

Object-Oriented Languages

Why?

- Holy grail: software reuse

What is it?

- Subtyping vs. inheritance

How does it work?

- Static vs. dynamic typing
- Single vs. multiple inheritance

Other Optimizations

Redundant array bounds checks

Pointer alias analysis

Tiling: optimize use of cache

Tail recursion: transform recursion to iteration

Parallelization: sequential program -> parallel program

Link-time optimization: whole program optimization