CSE 333
Section 7

Inheritance & VTable

Logistics

Friday (tomorrow)

Exercise 13 @ 10 am

Tuesday:

HW3 @ 11 pm

Inheritance

- **Derived** class inherits from the **base** class
  - In 333, we always use *public* inheritance
  - Inherits all *non-private* member variables
    - But private data is still present in the "base" part of the derived object
  - Inherits all *non-private* member functions
    except for ctor, cctor, dtor, op=
- Access specifiers revisited:
  - **Private** members cannot be accessed by derived classes
  - **Protected** members are available to base & derived

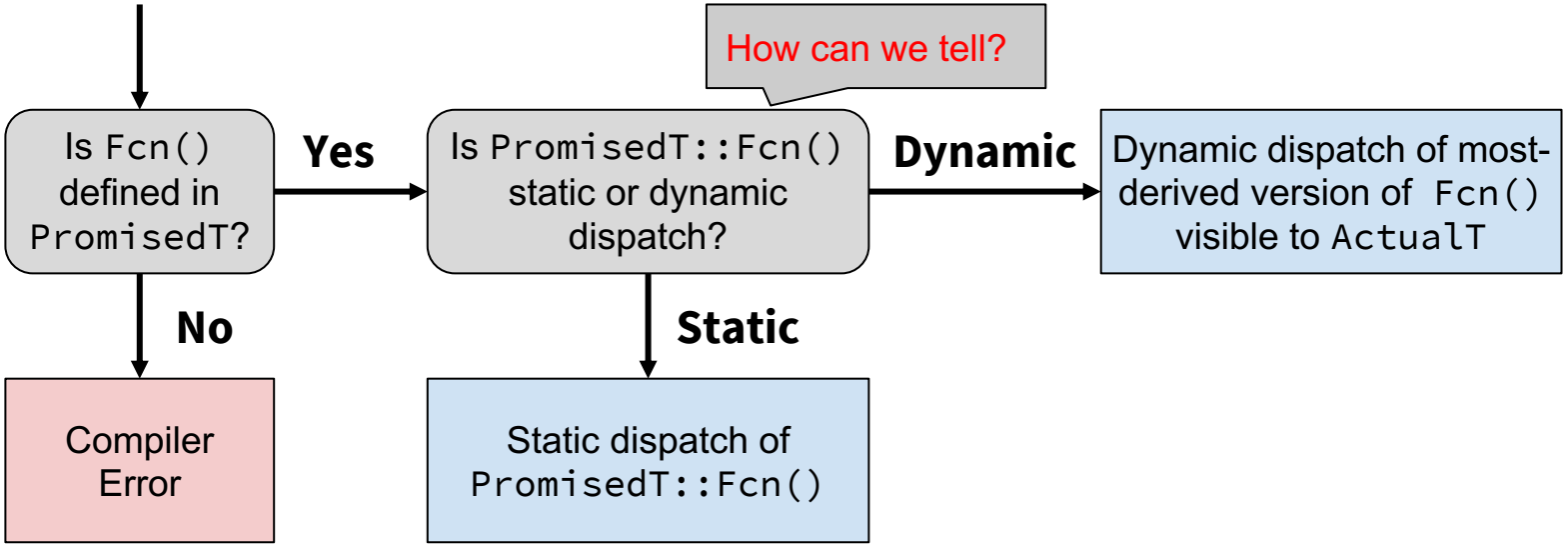
Static vs. Dynamic Dispatch

How to resolve invoking a method via a polymorphic pointer:

1. Static dispatch
    - Default behavior in C++
2. Dynamic dispatch
    - Which implementation is determined *at runtime* via lookup
    - Compiler generates code that accesses function pointers added to the class

Dispatch Decision Tree

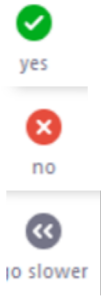
```
PromisedT *ptr = new ActualT();
ptr->Fcn();  // which version is called?
```



Dispatch Keywords

- `virtual` – request dynamic dispatch
  - Is “sticky”: overridden virtual method in derived class is still virtual with or without the keyword
- `override` – ensures that the function is virtual and is overriding a virtual function from a base class (`@override` in Java)
  - Generates a compiler error if conditions are not met
  - Catches overloading vs. overriding bugs at compile time

Practice: static, dynamic, or error?



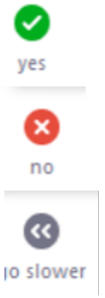
Static dispatch

Dynamic dispatch

Error

```
class Base {
  void Foo();              //
  void Bar();              //
  virtual void Baz();      //
};

class Derived : public Base {
  virtual void Foo();      //
  void Bar() override;     //
  void Baz();              //
};
```

Practice: static, dynamic, or error?



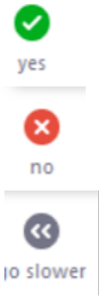
Static dispatch

Dynamic dispatch

Error

```
class Base {
  void Foo();              // static dispatch
  void Bar();              //
  virtual void Baz();      //
};

class Derived : public Base {
  virtual void Foo();      //
  void Bar() override;     //
  void Baz();              //
};
```

Practice: static, dynamic, or error?



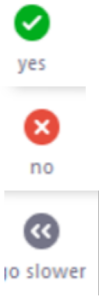
Static dispatch

Dynamic dispatch

Error

```
class Base {
  void Foo();              // static dispatch
  void Bar();              // static dispatch
  virtual void Baz();      //
};

class Derived : public Base {
  virtual void Foo();      //
  void Bar() override;     //
  void Baz();              //
};
```

Practice: static, dynamic, or error?



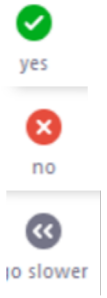
Static dispatch

Dynamic dispatch

Error

```
class Base {
  void Foo();              // static dispatch
  void Bar();              // static dispatch
  virtual void Baz();      // dynamic dispatch
};

class Derived : public Base {
  virtual void Foo();      //
  void Bar() override;     //
  void Baz();              //
};
```

Practice: static, dynamic, or error?



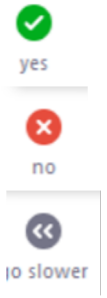
Static dispatch

Dynamic dispatch

Error

```
class Base {
  void Foo();              // static dispatch
  void Bar();              // static dispatch
  virtual void Baz();      // dynamic dispatch
};

class Derived : public Base {
  virtual void Foo();      // now dynamic (for more derived)
  void Bar() override;     //
  void Baz();              //
};
```

Practice: static, dynamic, or error?



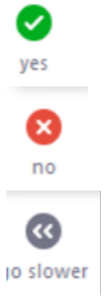
Static dispatch

Dynamic dispatch

Error

```
class Base {
  void Foo();              // static dispatch
  void Bar();              // static dispatch
  virtual void Baz();      // dynamic dispatch
};

class Derived : public Base {
  virtual void Foo();      // now dynamic (for more derived)
  //void Bar() override;// compiler error
  void Bar();              // static dispatch
  void Baz();              //
};
```

Practice: static, dynamic, or error?



Static dispatch

Dynamic dispatch

Error

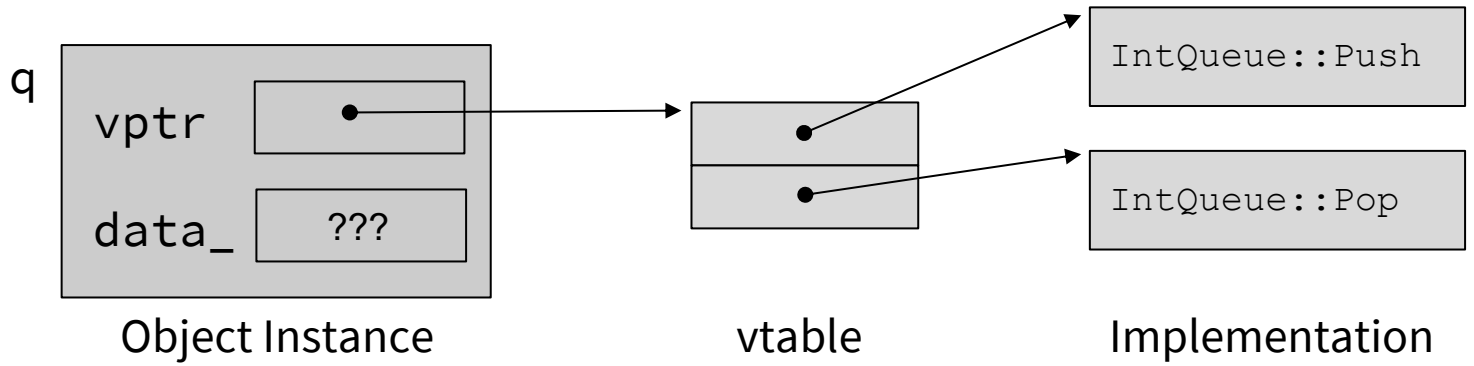
```
class Base {
  void Foo();              // static dispatch
  void Bar();              // static dispatch
  virtual void Baz();      // dynamic dispatch
};

class Derived : public Base {
  virtual void Foo();      // now dynamic (for more derived)
  //void Bar() override;// compiler error
  void Bar();              // static dispatch
  void Baz();              // still dynamic (sticky!)
};
```

Vtable (Virtual Function Table) & Vptr (Vtable pointer)

- vtable: An array of function pointers defined for each class that has at least one virtual method to enable dynamic dispatch
  - One per class
- vptr: Each class object instance has a pointer to that vtable
  - One per object instance

Vtable Diagrams

```
class IntQueue {
 public:
  virtual void Push(int x);
  virtual int Pop();
 private:
  vector<int> data_;
};
IntQueue q;
```

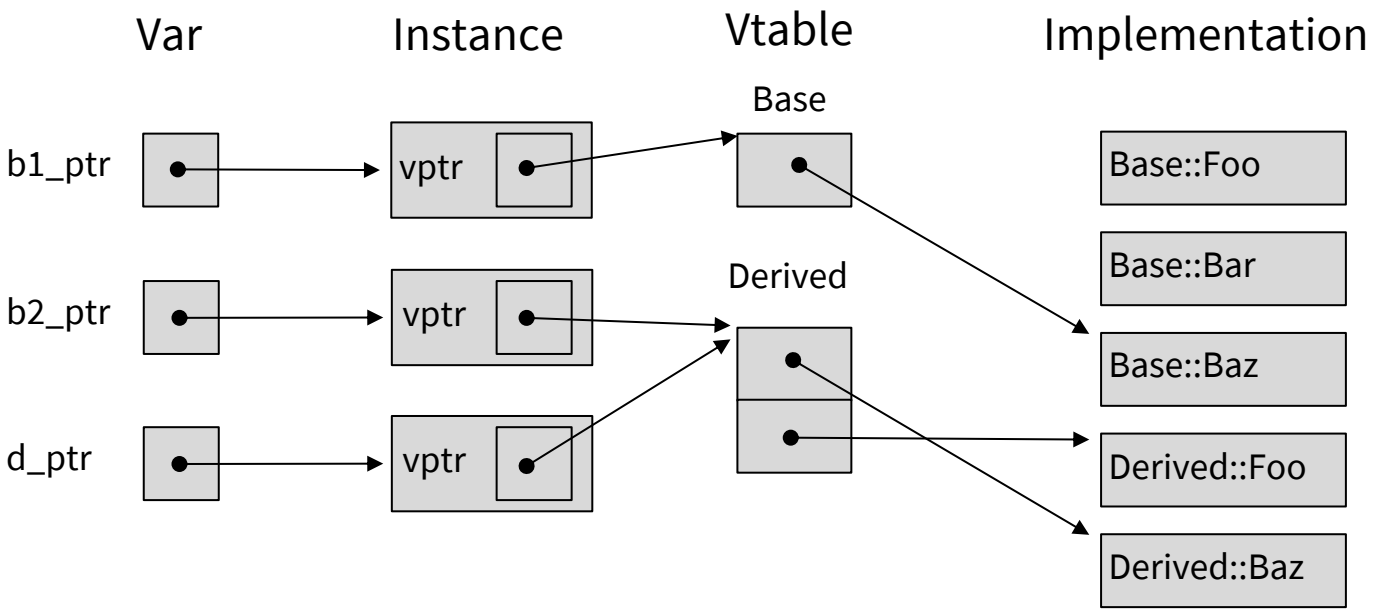


Vtable Diagrams

```
Base *b1_ptr = new Base;
Base *b2_ptr = new Derived;
Derived *d_ptr = new Derived;
```

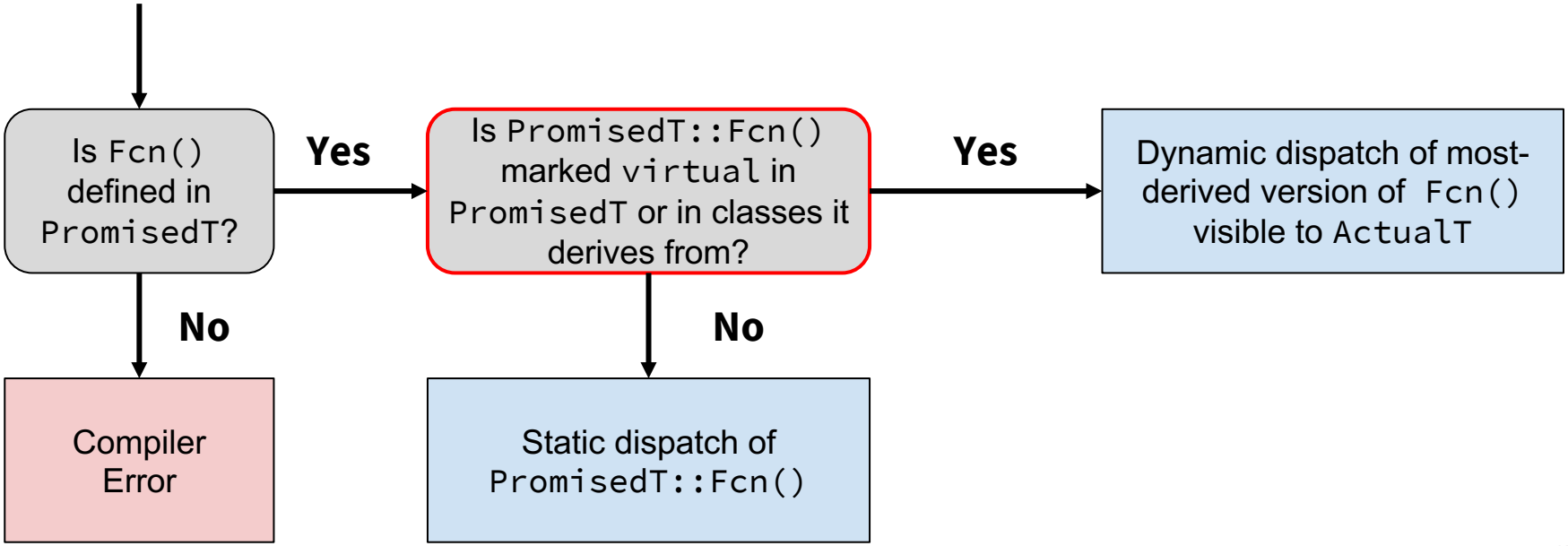
```
class Base {
  void Foo();
  void Bar();
  virtual void Baz();
};
```

```
class Derived :
public Base {
  virtual void Foo();
  void Baz();
};
```



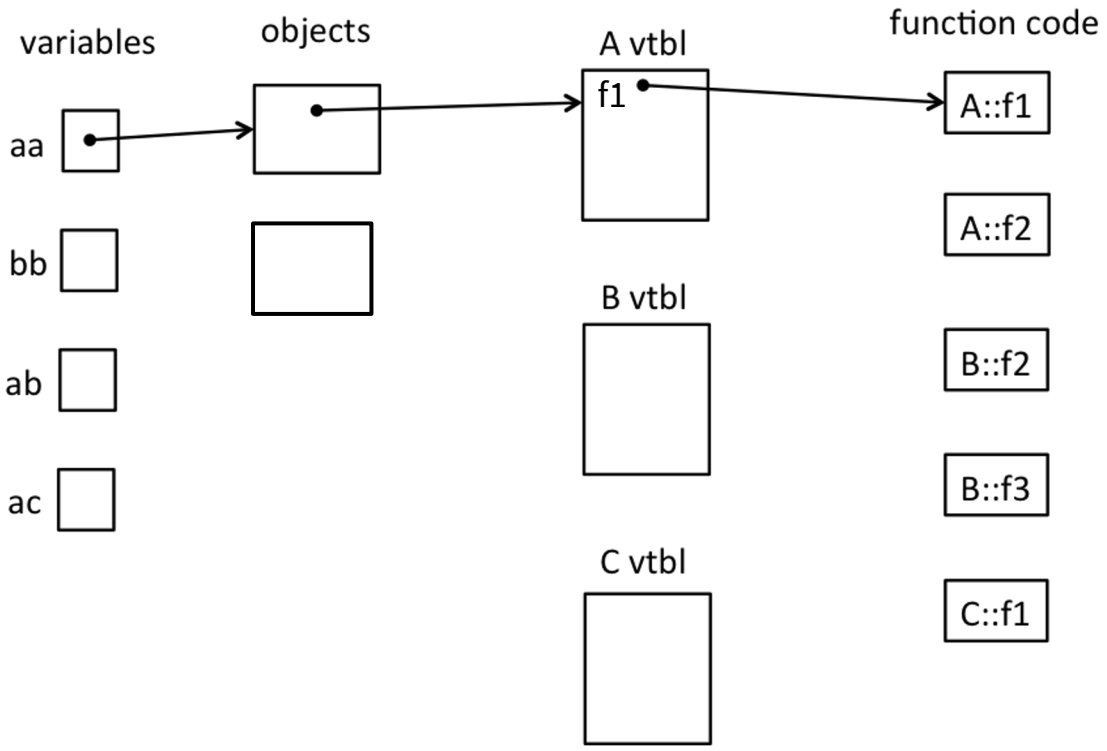
Updated Dispatch Decision Tree

```
PromisedT *ptr = new ActualT();
ptr->Fcn();  // which version is called?
```



Exercise 1!

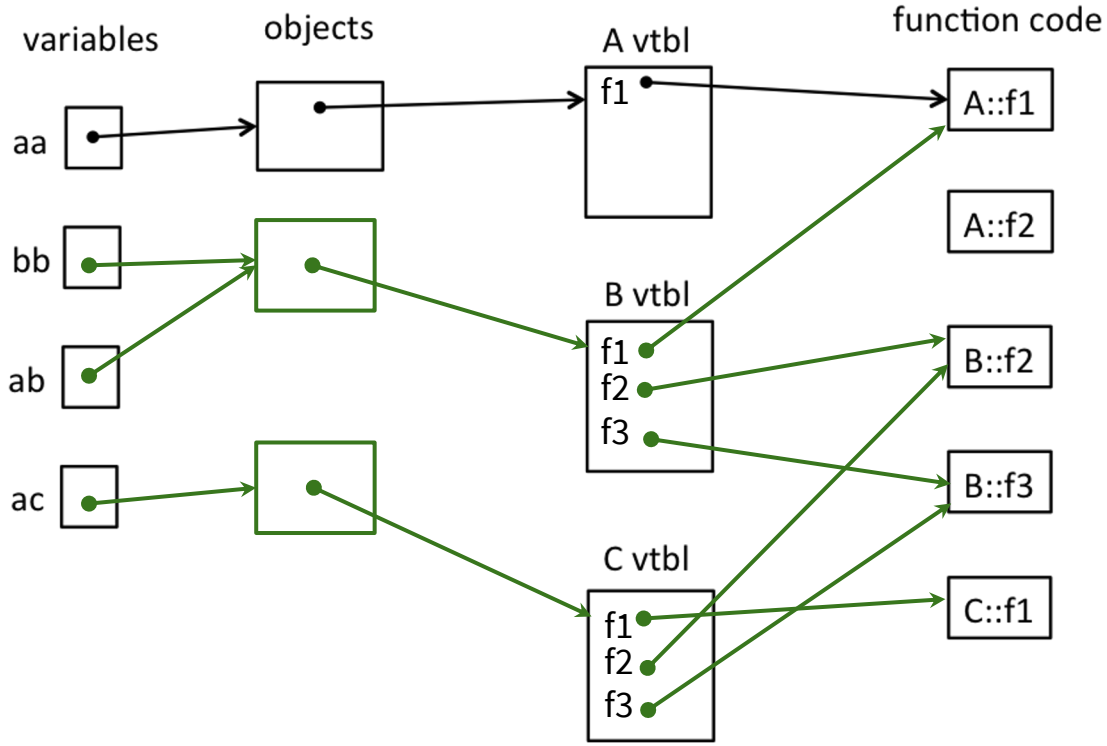
Exercise 1


variables
objects
A vtbl
function code
aa
f1
A::f1
A::f2
bb
B vtbl
ab
B::f2
B::f3
ac
C vtbl
C::f1

Exercise 1 Solution (pointers)

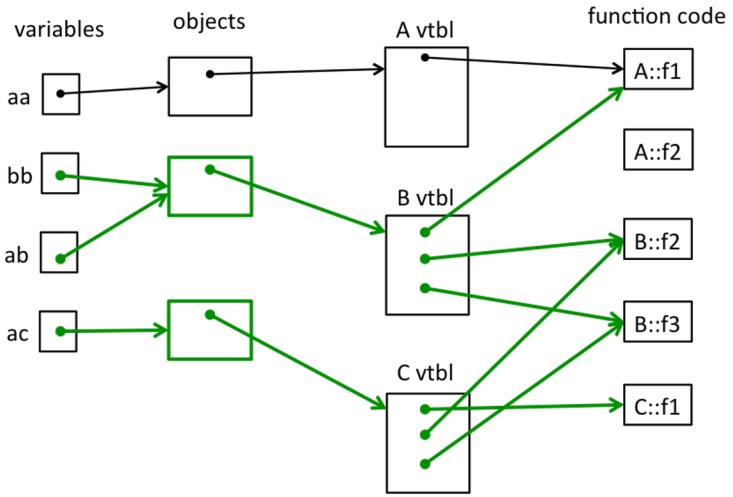
```
class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl; }
  void f2() { cout << "A::f2" << endl; }
};
class B : public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl; }
  virtual void f2() { cout << "B::f2" << endl; }
};
class C : public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

```
int main() {
  A* aa = new A();
  B* bb = new B();
  A* ab = bb;
  A* ac = new C();
```



Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

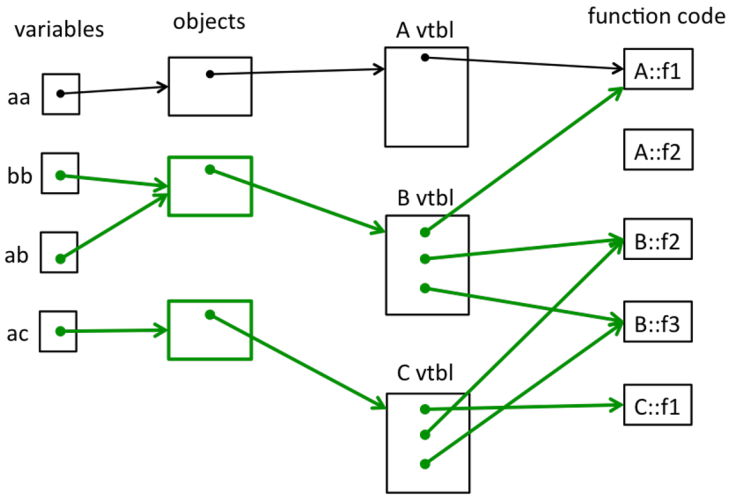



```
A* aa = new A();

aa->f1();
```

B::f2	A::f2	A::f2	B::f2
A::f1	C::f1	A::f1	C::f1
yes	no	go slower	go faster

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

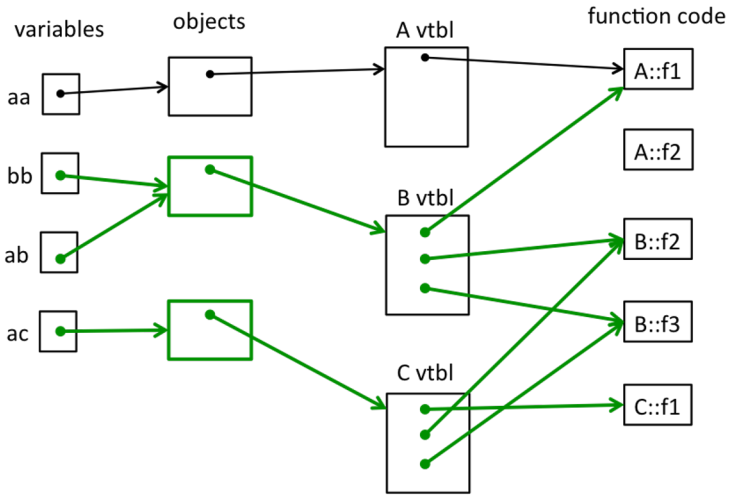



```
A* aa = new A();

aa->f1();
```

```
A::f2
A::f1
```

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

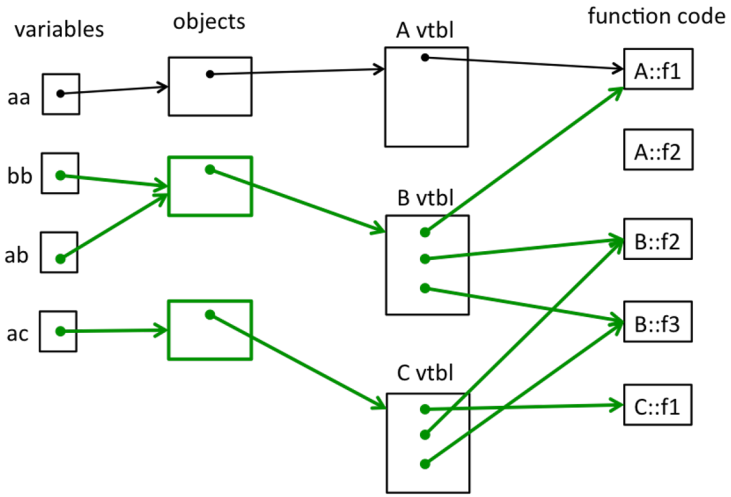



```
B* bb = new B();

bb->f1();
```

B::f2	A::f2	A::f2	B::f2
A::f1	C::f1	A::f1	C::f1
yes	no	go slower	go faster

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

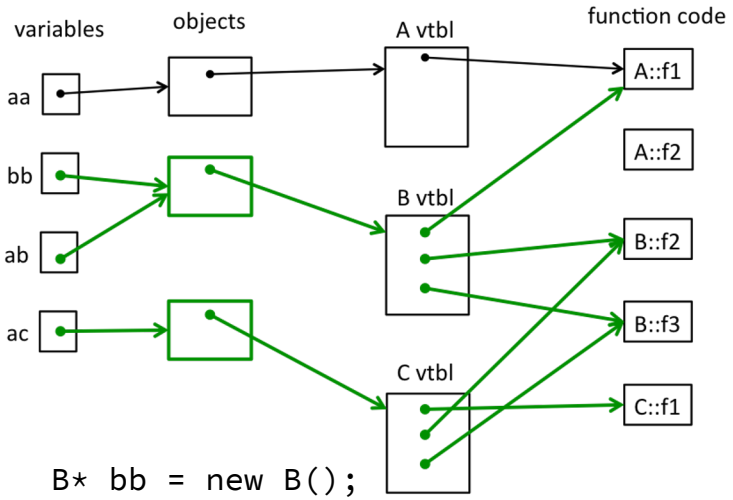



```
B* bb = new B();

bb->f1();
```

```
A::f2
A::f1
```

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```



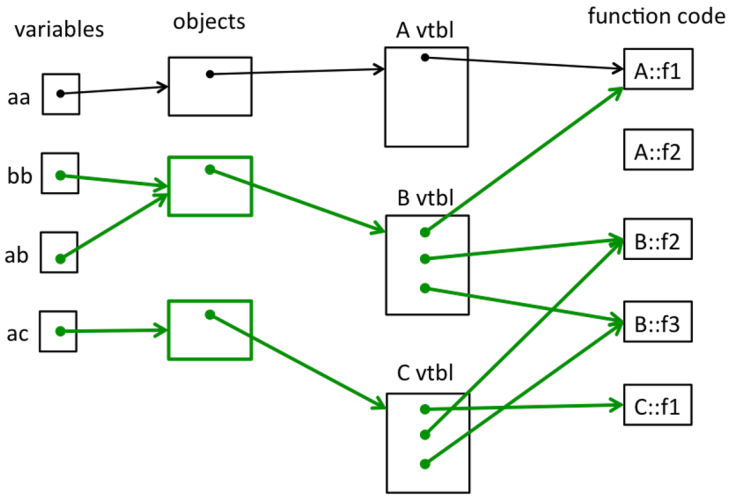

```
B* bb = new B();
A* ab = bb;

bb->f2();
cout << "----" << endl;
ab->f2();
```

```
B::f2   A::f2   B::f2   A::f2
----    ----    ----    ----
B::f2   B::f2   A::f2   A::f2
```

yes no go slower go faster

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

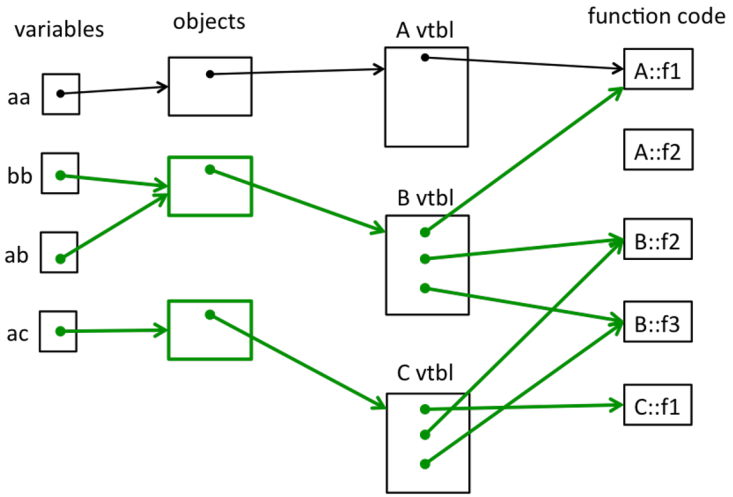



```
B* bb = new B();
A* ab = bb;

bb->f2();
cout << "----" << endl;
ab->f2();
```

```
B::f2
----
A::f2
```

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

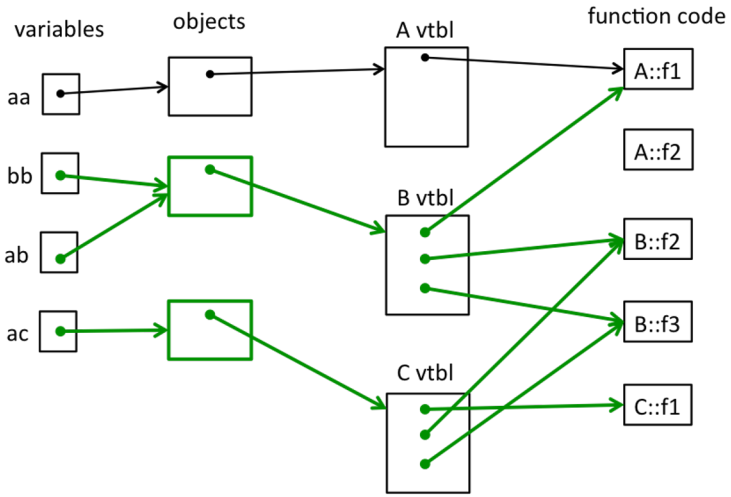



```
B* bb = new B();

bb->f3();
```

B::f2	A::f2	A::f2	B::f2
A::f1	A::f1	C::f1	C::f1
B::f3	B::f3	B::f3	B::f3
yes	no	go slower	go faster

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

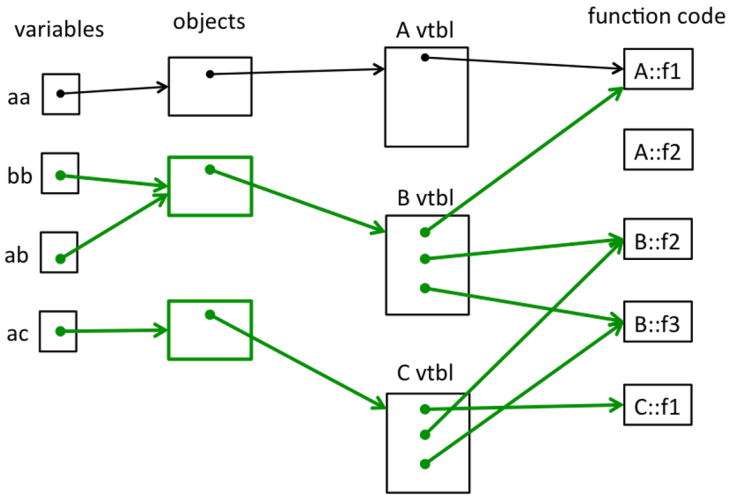



```
B* bb = new B();

bb->f3();
```

```
A::f2
A::f1
B::f3
```

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```



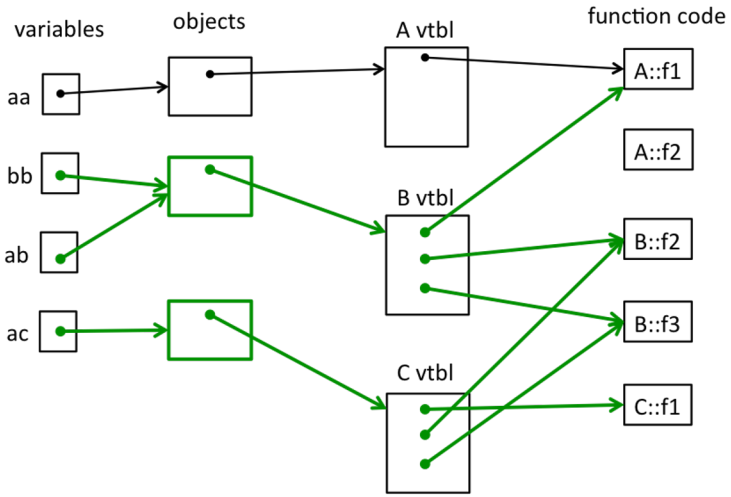

```
A* ac = new C();

ac->f1();
```

B::f2	A::f2	A::f2	B::f2
A::f1	C::f1	A::f1	C::f1

yes no go slower go faster

Exercise 1 Solution (output)

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

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl;
}
  void f2() { cout << "A::f2" << endl; }
};

class B: public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl;
}
  virtual void f2() { cout << "B::f2" << endl; }
};

class C: public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```

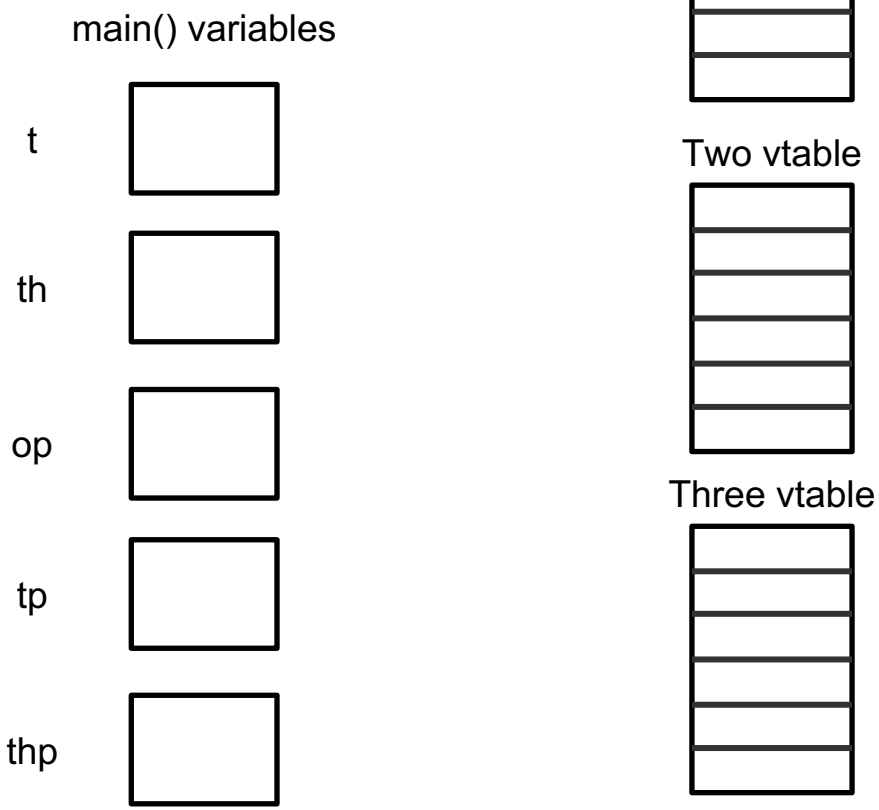



```
A* ac = new C();

ac->f1();
```

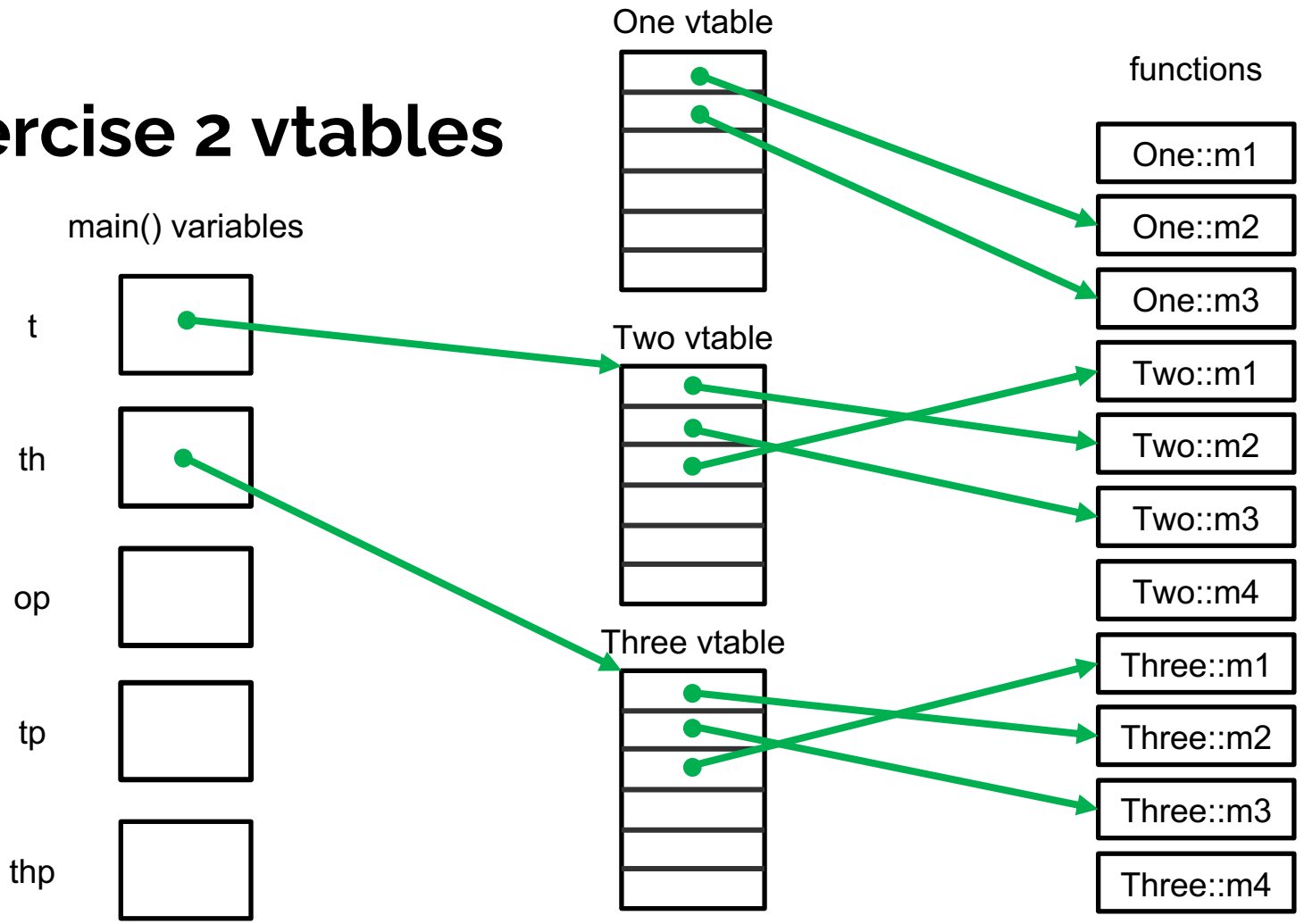
```
B::f2
C::f1
```

Exercise 2!

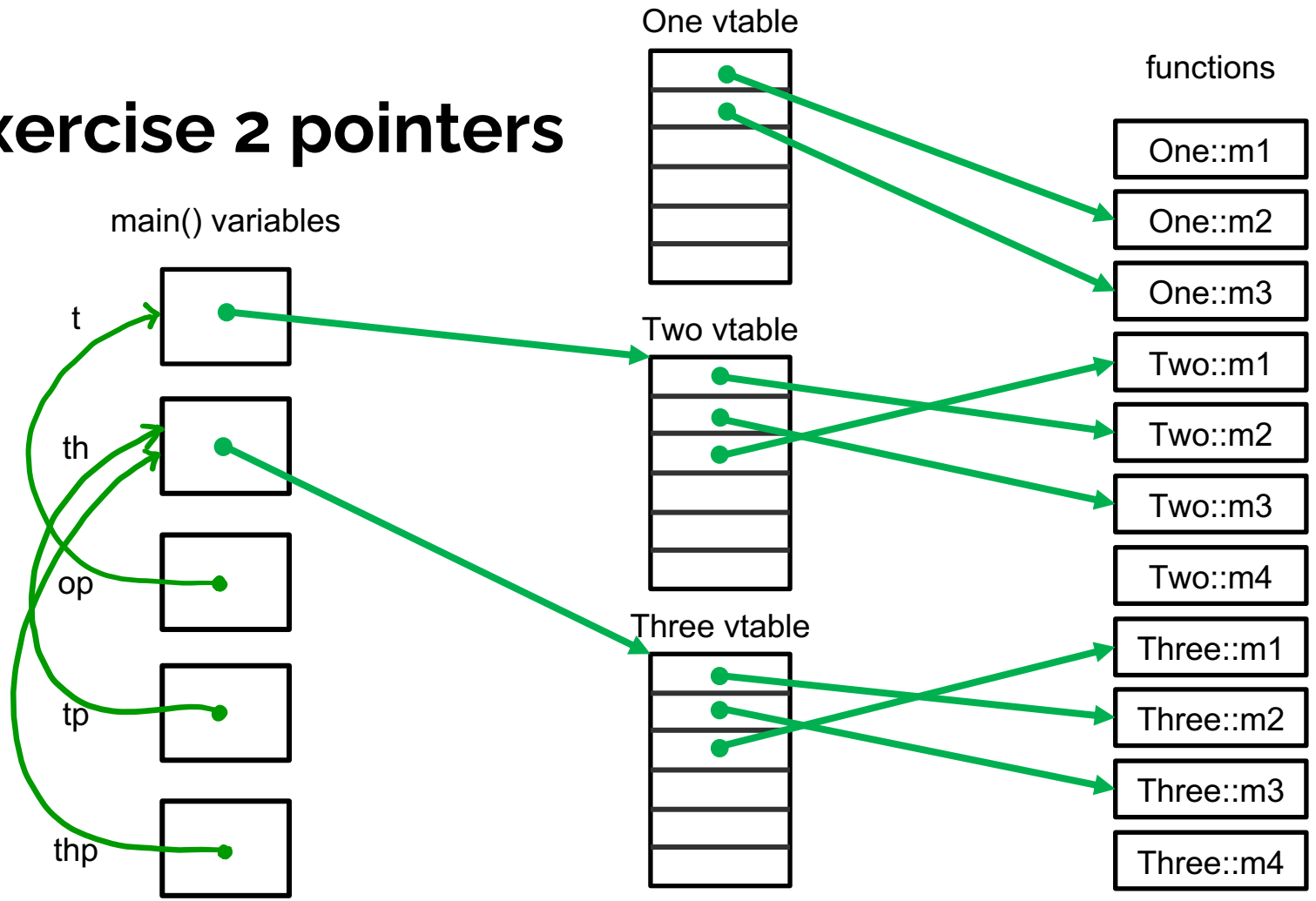
Exercise 2


main() variables
t
th
op
tp
thp
One vtable
Two vtable
Three vtable
functions
One::m1
One::m2
One::m3
Two::m1
Two::m2
Two::m3
Two::m4
Three::m1
Three::m2
Three::m3
Three::m4

Exercise 2 vtables

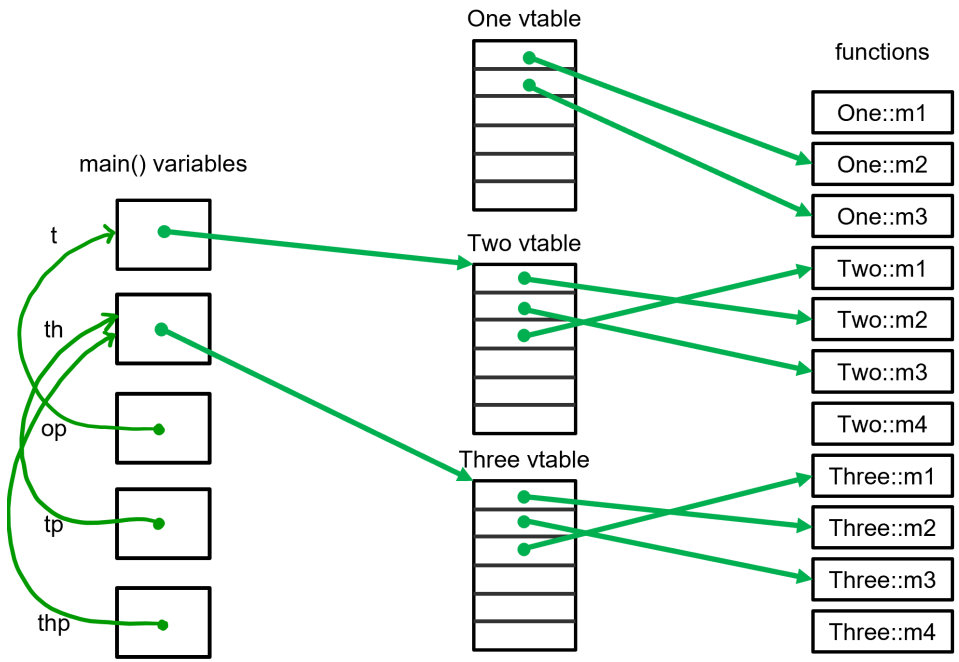


Exercise 2 pointers



Exercise 2 Output

```
class One {
public:
  void m1() { cout << "H"; }
  virtual void m2() { cout << "l"; }
  virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
  void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
  void m4() { cout << "p";}
};

class Three: public Two {
public:
  void m1() { cout << "o"; }
  void m2() { cout << "i"; }
  void m3() { cout << "s"; }
  void m4() { cout << "!"; }
};
```



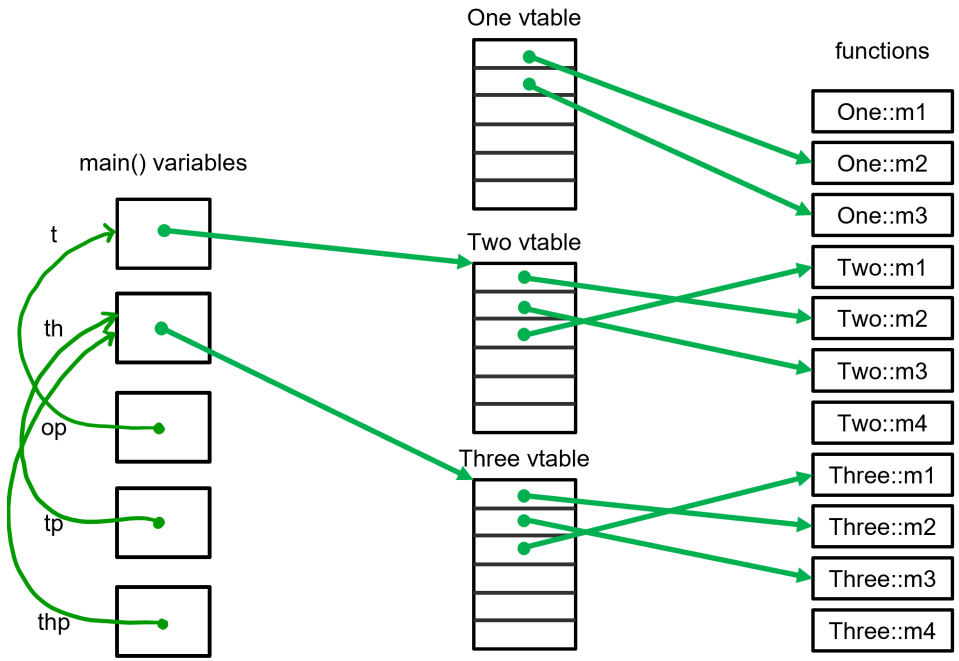

```
Two t;
Three th;
One *op = &t;
Two *tp = &th;
Three *thp = &th;
```

```
op->m1();
tp->m1();
op->m3();
```

Hoy

Exercise 2 Output

```
class One {
public:
  void m1() { cout << "H"; }
  virtual void m2() { cout << "l"; }
  virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
  void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
  void m4() { cout << "p";}
};

class Three: public Two {
public:
  void m1() { cout << "o"; }
  void m2() { cout << "i"; }
  void m3() { cout << "s"; }
  void m4() { cout << "!"; }
};
```



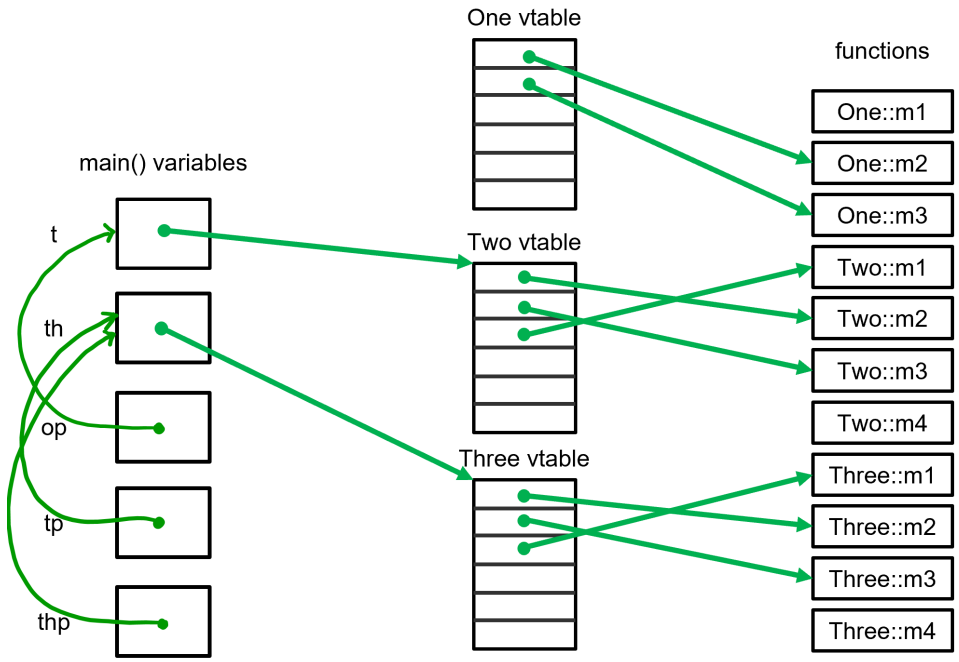

```
Two t;
Three th;
One *op = &t;
Two *tp = &th;
Three *thp = &th;
```

```
op->m3();
tp->m3();
op->m1();
```

HoyysH

Exercise 2 Output

```
class One {
public:
  void m1() { cout << "H"; }
  virtual void m2() { cout << "l"; }
  virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
  void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
  void m4() { cout << "p";}
};

class Three: public Two {
public:
  void m1() { cout << "o"; }
  void m2() { cout << "i"; }
  void m3() { cout << "s"; }
  void m4() { cout << "!"; }
};
```



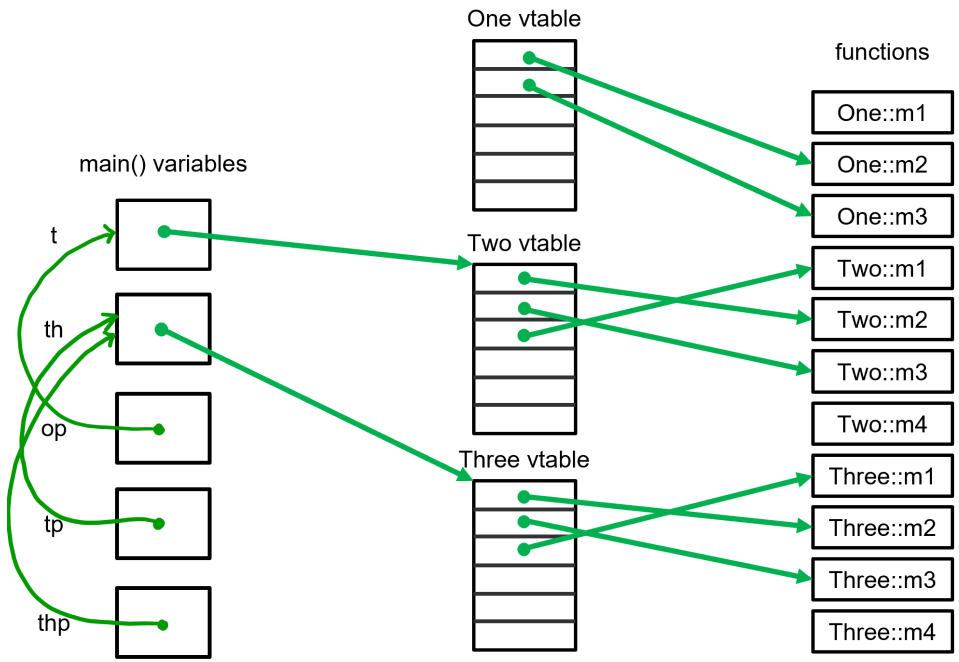

```
Two t;
Three th;
One *op = &t;
Two *tp = &th;
Three *thp = &th;
```

```
thp->m1();
op->m2();
thp->m2();
```

HoyysHodi

Exercise 2 Output

```
class One {
public:
  void m1() { cout << "H"; }
  virtual void m2() { cout << "l"; }
  virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
  void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
  void m4() { cout << "p";}
};

class Three: public Two {
public:
  void m1() { cout << "o"; }
  void m2() { cout << "i"; }
  void m3() { cout << "s"; }
  void m4() { cout << "!"; }
};
```



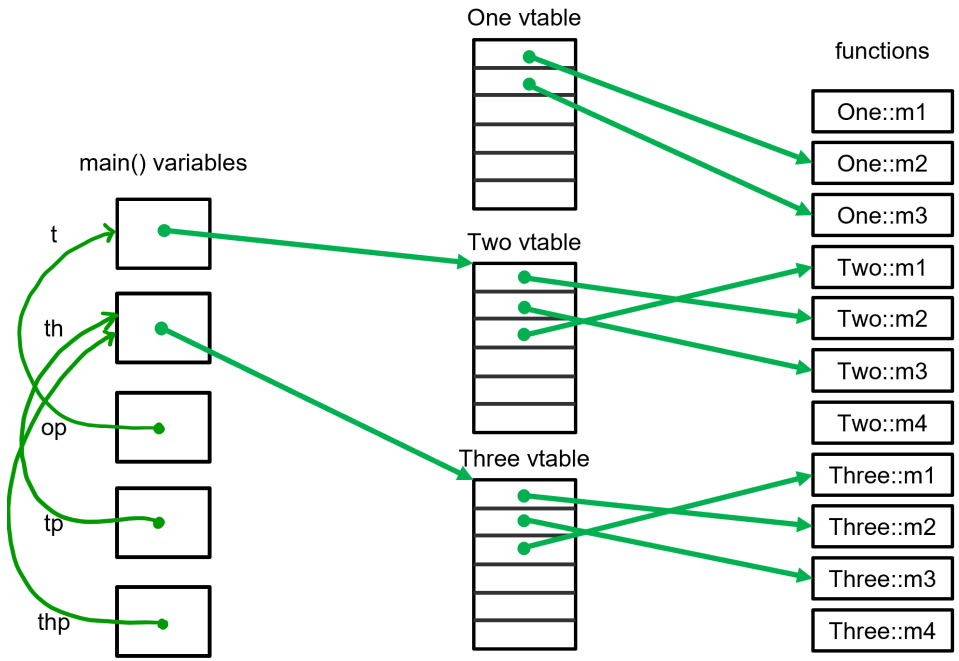

```
Two t;
Three th;
One *op = &t;
Two *tp = &th;
Three *thp = &th;
```

```
tp->m2();
tp->m1();
tp->m3();
```

HoyysHodiios

Exercise 2 Output

```
class One {
public:
  void m1() { cout << "H"; }
  virtual void m2() { cout << "l"; }
  virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
  void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
  void m4() { cout << "p";}
};

class Three: public Two {
public:
  void m1() { cout << "o"; }
  void m2() { cout << "i"; }
  void m3() { cout << "s"; }
  void m4() { cout << "!"; }
};
```



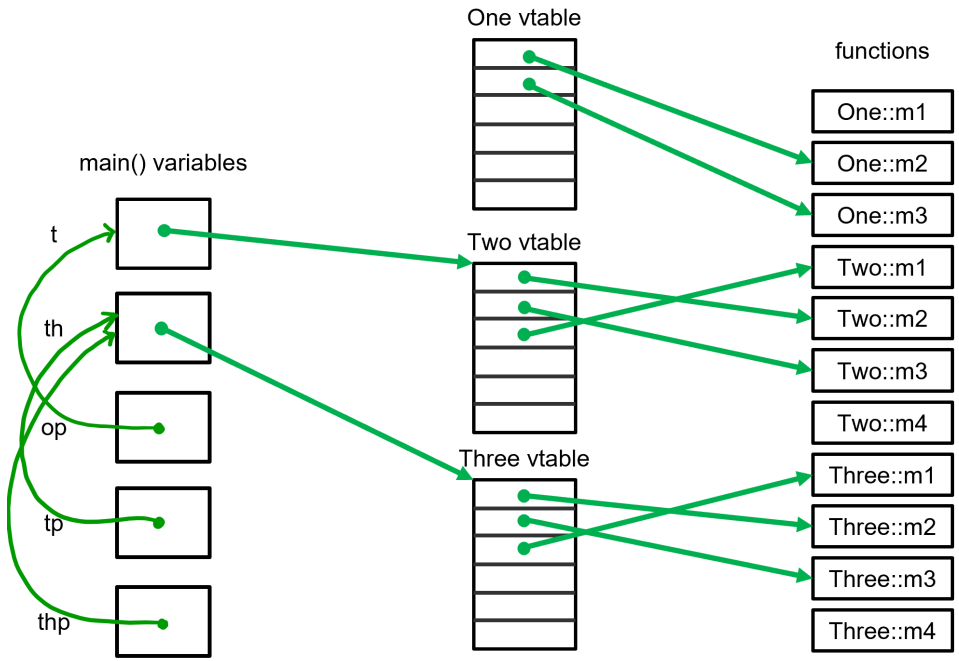

```
Two t;
Three th;
One *op = &t;
Two *tp = &th;
Three *thp = &th;
```

```
thp->m3();
tp->m4();
cout << endl;
```

HoyysHodiiossp(\n)

Exercise 2 Output

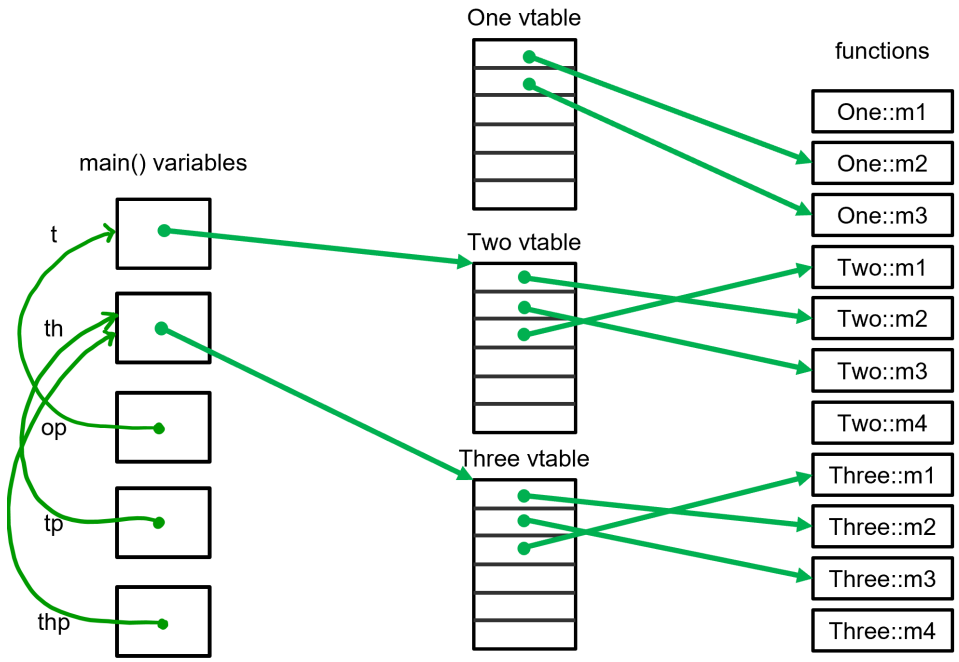
```
class One {
public:
  void m1() { cout << "H"; }
  virtual void m2() { cout << "l"; }
  virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
  void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
  void m4() { cout << "p";}
};

class Three: public Two {
public:
  void m1() { cout << "o"; }
  void m2() { cout << "i"; }
  void m3() { cout << "s"; }
  void m4() { cout << "!"; }
};
```



How do we modify this to print HappyHolidays! (with the !)?

Exercise 2 Output

```
class One {
public:
  void m1() { cout << "H"; }
  virtual void m2() { cout << "l"; }
  virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
  void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
  virtual m4() { cout << "p";}
};

class Three: public Two {
public:
  void m1() { cout << "o"; }
  void m2() { cout << "i"; }
  void m3() { cout << "s"; }
  void m4() { cout << "!"; }
};
```



How do we modify this to print HappyHolidays! (with the !)?

Any Questions on Inheritance & VTable?