MULTIPLE INHERITANCE

Plan for Today

- Static Polymorphism
- Multiple Inheritance
- Combining Static and Dynamic Polymorphism
- Mixins
- CRTP

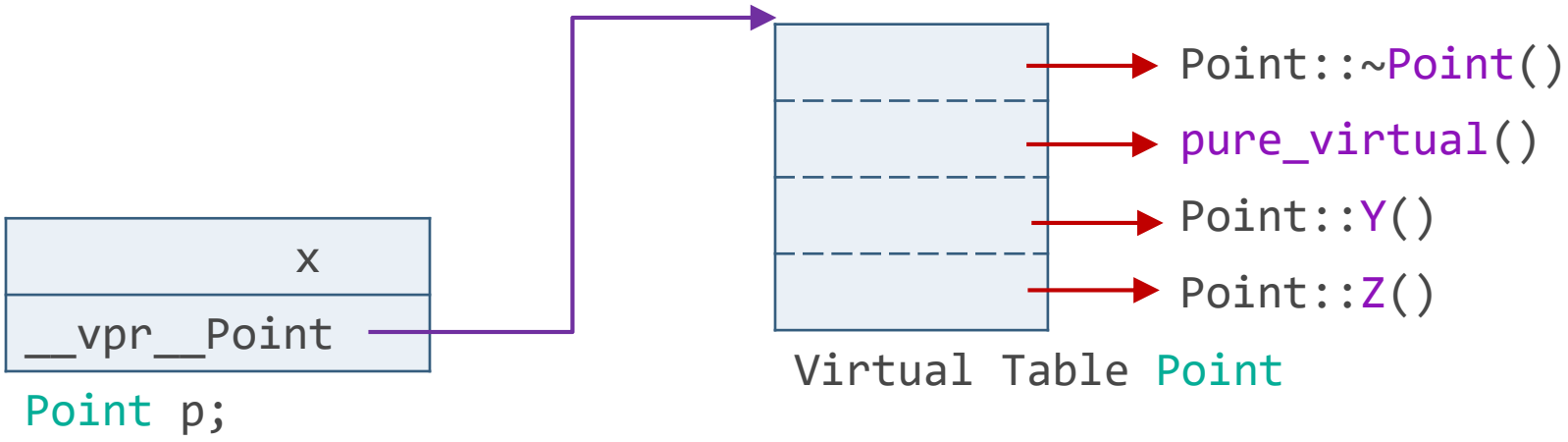
Virtual Functions

- When virtual function is called, code executed must correspond to dynamic type of pointer
- Static (declared) type of pointer doesn't matter
- Since set of virtual functions is known at compile time, C++ uses virtual tables and virtual table pointers

Virtual Functions


```
class Point {
public:
  virtual ~Point();
  virtual Point& mult(double) = 0;
  // other stuff ...
  double X() const { return x; }
  virtual double Y() const { return 0; }
  virtual double Z() const { return 0; }
  // ...
protected:
  Point(double mx = 0.0);
  double x;
};
```

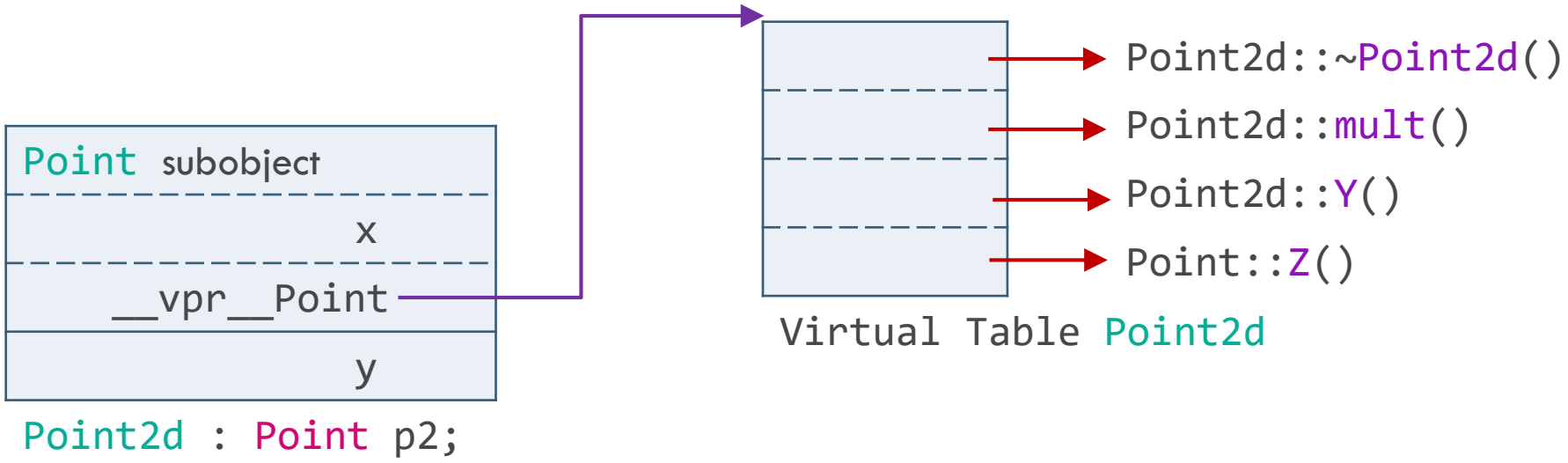
Virtual Functions



Virtual Functions


```
class Point2d : public Point {
public:
  Point2d(double mx=0.0, double my=0.0
  : Point(mx), y{my} {}
  ~Point2d();
  // overridden base class virtual functions
  Point2d& mult(double);
  double Y() const { return y; }
  // other functions ...
protected:
  double y;
};
```

Virtual Functions

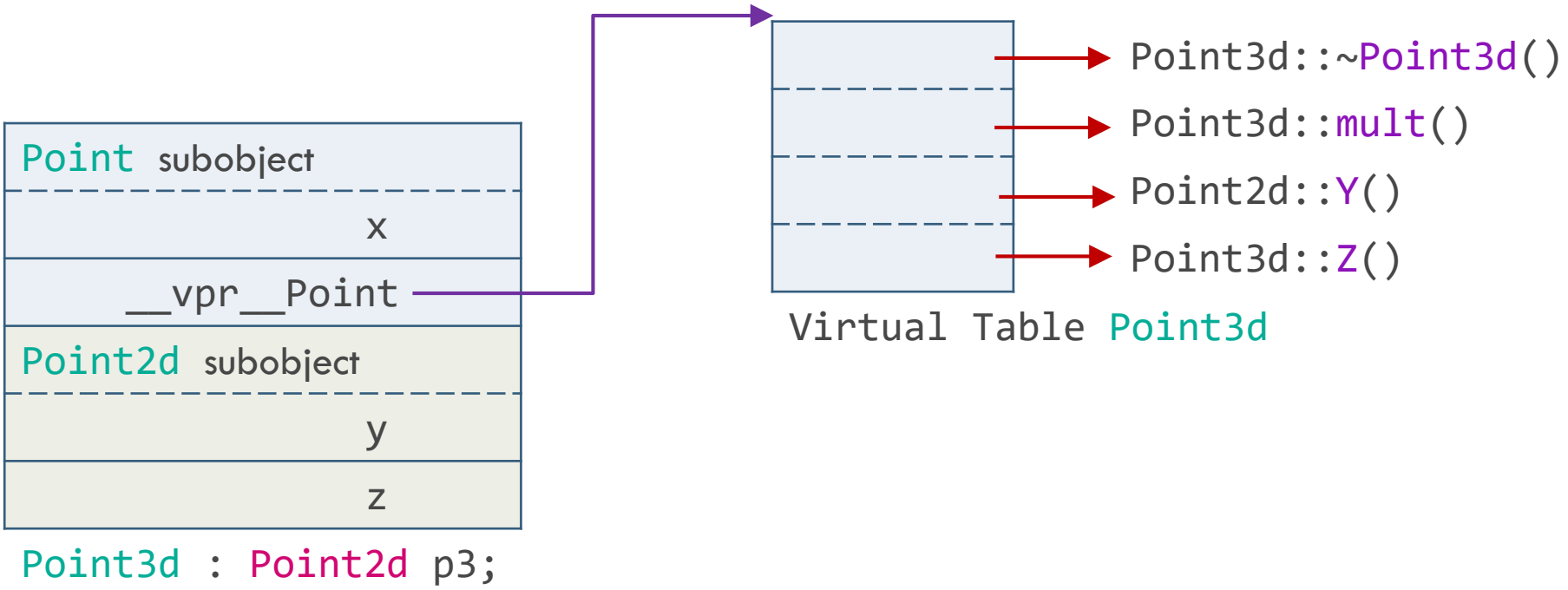


Virtual Functions


```
class Point3d : public Point2d {
public:
  Point3d(double mx=0.0, double my=0.0, double mz=0.0)
         : Point2d(mx, my) : z{mz} {}
  ~Point3d();

  // overridden base class virtual functions
  Point3d& mult(double);
  double Z() const { return z; }
  // other operations ...
protected:
  double z;
};
```

Virtual Functions



Virtual Functions

- Do we know enough at compile time to set up virtual function call `ptr->Z()`
- In general, we don't exact type of object `ptr` addresses
- But, thro' `ptr`, we can access virtual table associated with object's class
- We know instance of `Z()` is contained in slot 2

```
// compiler transforms call
ptr->Z();
// into dereferencing vptr of object pointed to by ptr
(*ptr->vptr[3])(ptr);
```

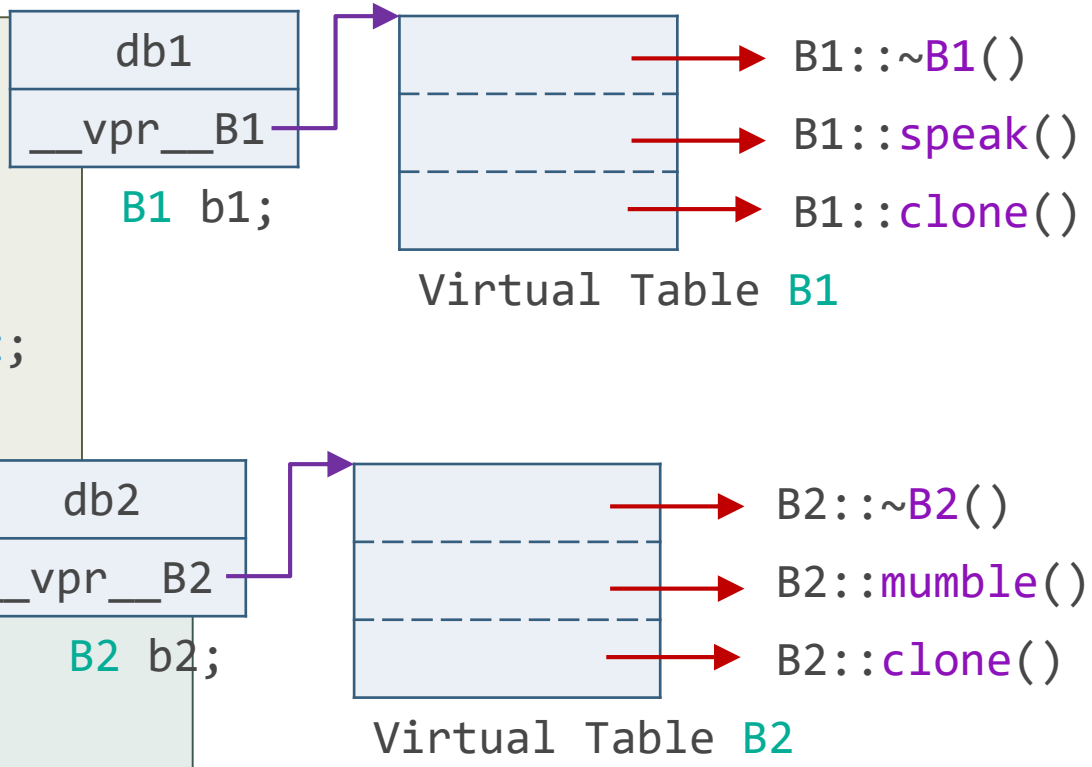
Costs of Virtual Functions

- Compiler has to set aside memory for virtual table for each class that contains virtual functions
- An extra pointer must be embedded inside each object that is of a class containing virtual functions

Virtual Functions Under Multiple Inheritance

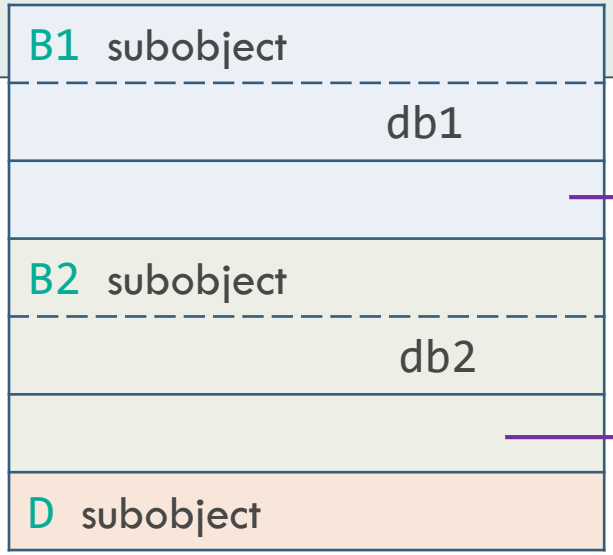

```
class B1 {
public:
  B1();
  virtual ~B1();
  virtual void speak();
  virtual B1* clone() const;
protected:
  double db1;
};
```

```
class B2 {
public:
  B2();
  virtual ~B2();
  virtual void mumble();
  virtual B2* clone() const;
protected:
  double db2;
};
```

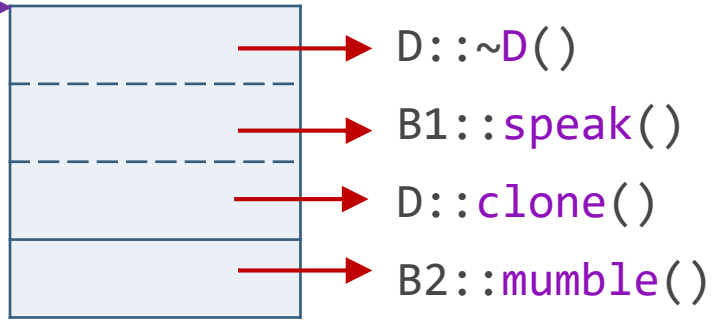


Virtual Functions Under Multiple Inheritance

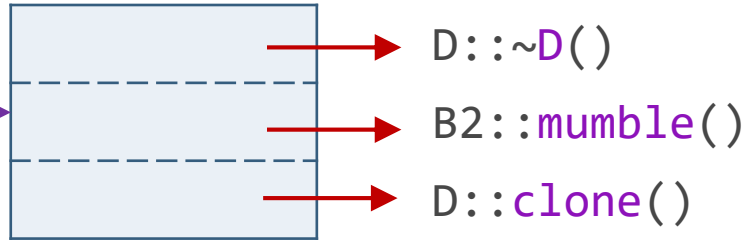
```
class D : public B1, public B2 {
public:
  D();
  virtual ~D();
  virtual D* clone() const;
protected:
  double dd;
};
```



D : B1, B2 d;



Derived virtual table
(shared with B1)



Derived virtual table
(shared with B2)