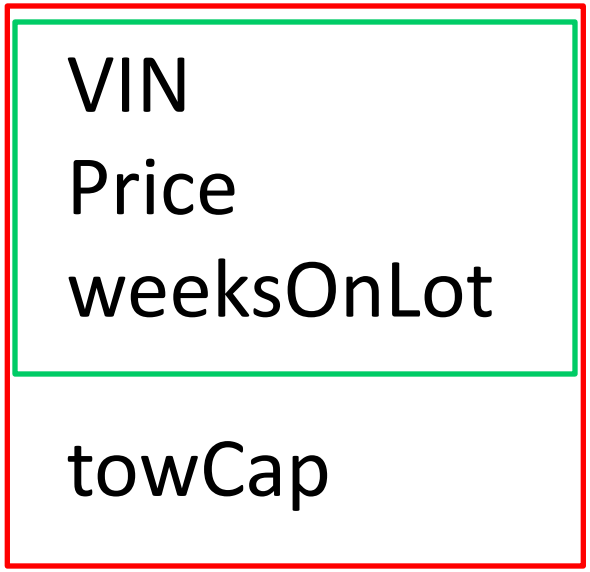
With redefined member functions

At compile time, the compiler generates code for each defined version of display1() function. The ***type*** of a pointer then lets compiler know which version of display1() to link.

```
Vehicle * vPtr = &Truck;
vPtr->display1();
```

Since vPtr is a pointer to Vehicles, the Vehicle version of display1() is called, even though pointed at truck…



Code: Vehicle display1()

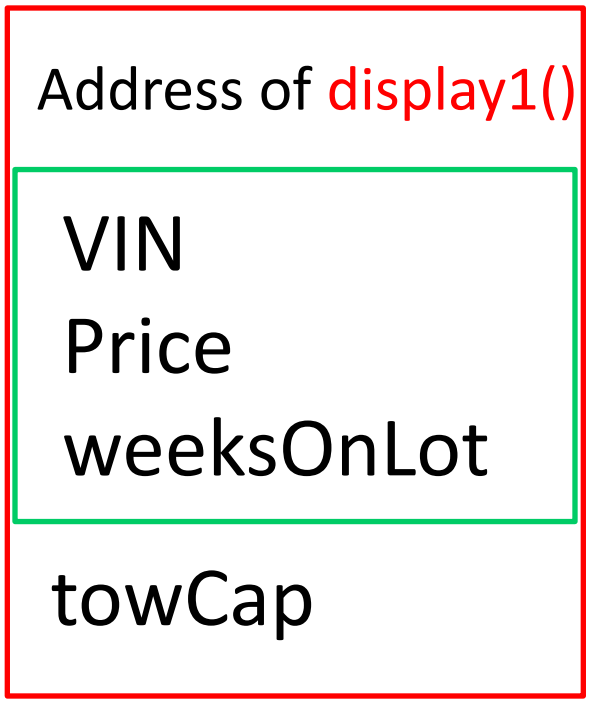
Code: Truck display1()

With Virtual member functions

At compile time, the compiler builds extra info. into each Truck object. That info is then used at ***runtime*** if that ***object is pointed at***.

```
Vehicle * vPtr = &Truck;
vPtr->display1();
```

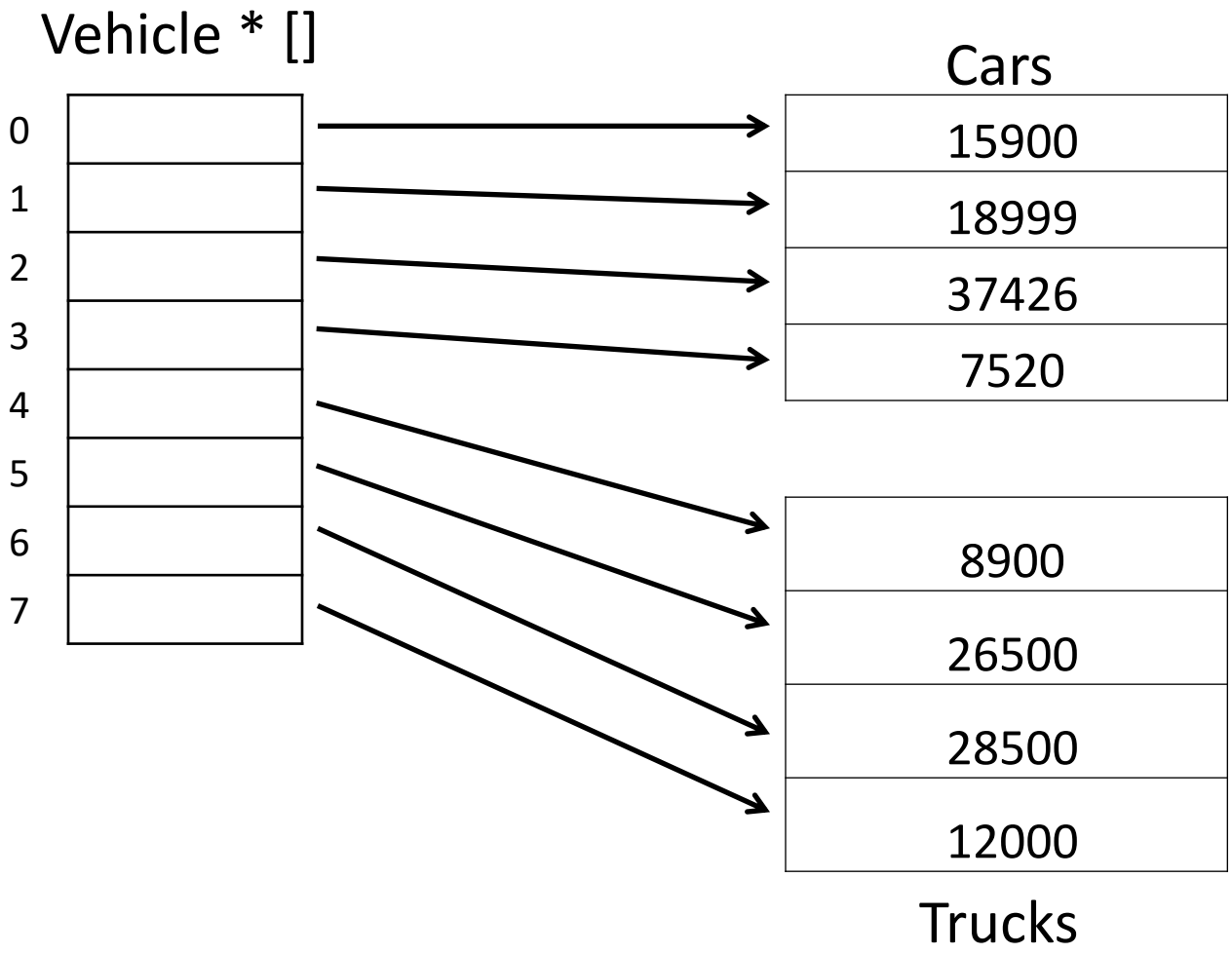
Since vPtr is pointing at a Truck object, the Truck version of display1() is called, making use of the extra info.



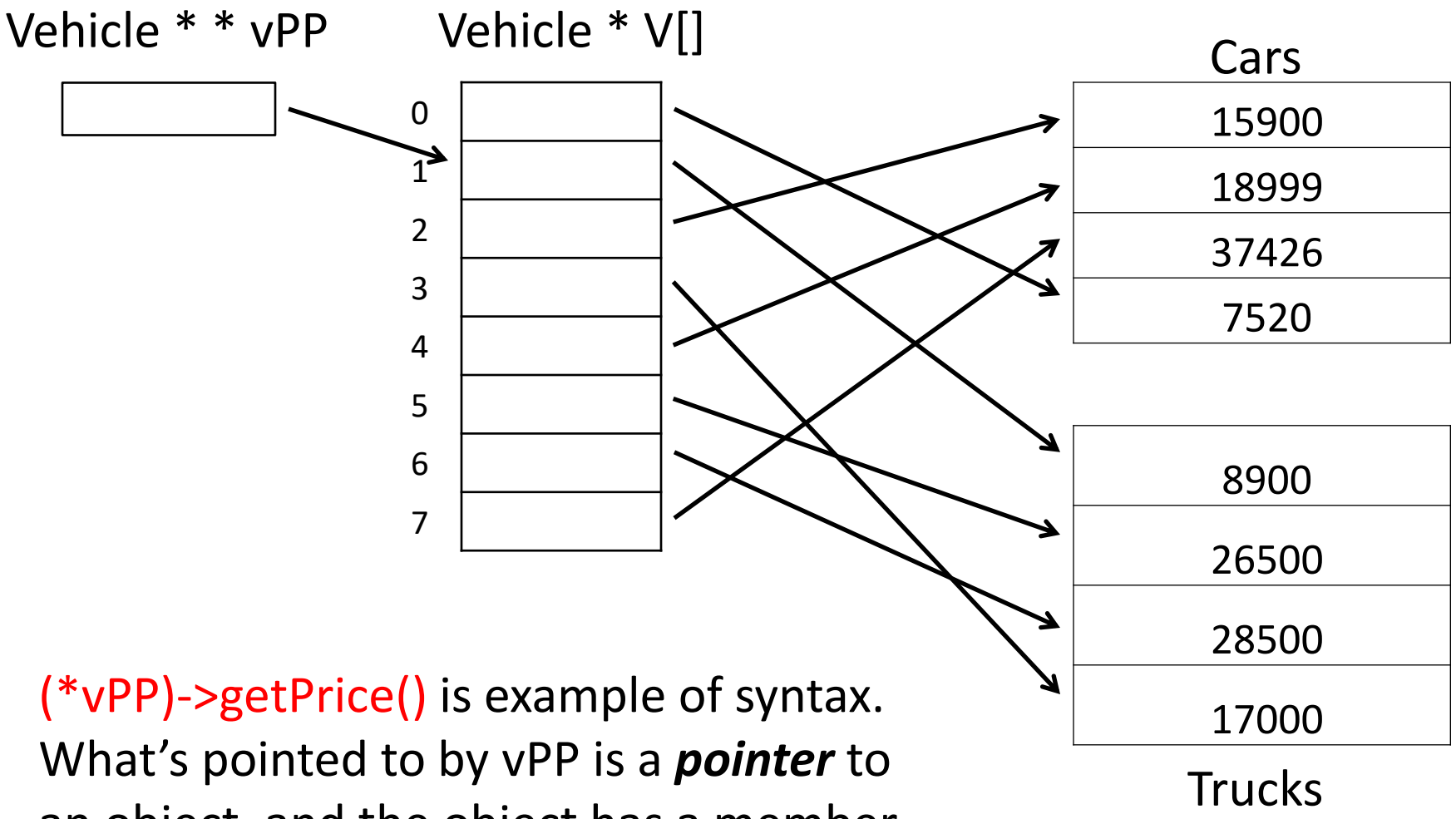
Code: Vehicle display1()

Code: Truck display1()

Using array of POINTERS to objects, to support “polymorphism”



Sorting the POINTERS, rather than sorting the objects



(*vPP)->getPrice() is example of syntax. What's pointed to by vPP is a ***pointer*** to an object, and the object has a member function getPrice()