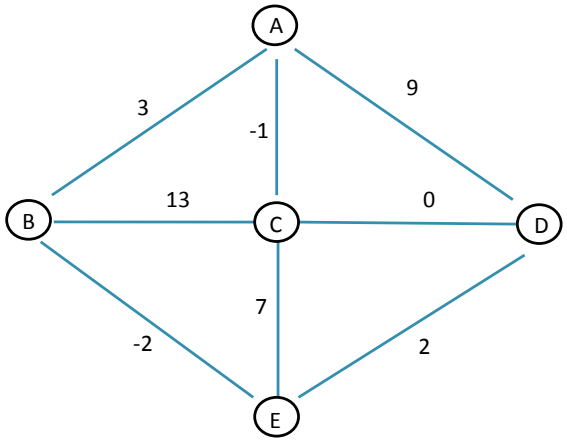
LECTURE 41: GRAPHS/ INHERITANCE

Today

- Reading
  - JS Ch. 16
  - Weiss Ch. 6
- Objectives
  - Prim's algorithm for minimum spanning trees
  - Inheritance in C++

Graph



Inheritance in C++

- Syntax of declaring a subclass

```
class Derived : public Base
```

- Public inheritance implements an “isA” relationship

Inheritance in C++

- Derived class inherits
  - All public and protected members
- Derived class does *not* inherit
  - Base class constructors, destructor, operator=
  - private members
  - (any friends)
- Constructor of derived class must call constructor of base class

Static vs. Dynamic dispatching

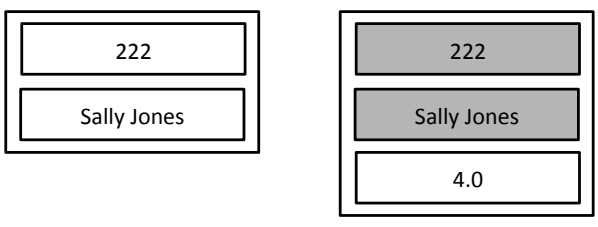
- Static dispatching
  - Determine which member function to call by checking type at *compile* time
- Dynamic dispatching
  - Determine which member function to call by checking type at *runtime*

The virtual keyword

- The `virtual` keyword signals that the function uses dynamic dispatching
- Allows this function to be overwritten in subclasses
- If don't use `virtual`, the function called is based on compile-time type not runtime type

Slicing

- Slicing – when derived copied into base, only fields from base class are preserved
- Slicing occurs whenever objects are copied
  - For example, call-by-value or return-by-value



Slicing

- Assignment of pointers works as expected!
- Bottom line: if want subtyping, use pointers or call by reference. Copying destroys subtyping
- In particular, if you want a vector of Person or subclass,

```
vector<Person*> people;
```

Casting in C++

- Type casts in C++ always succeed!
- Downcasting on pointers always succeeds!
- To get checked conversions, use `dynamic_cast`
  - Returns `NULL` if the cast is incorrect
  - `dynamic_cast` does a compile time *and* runtime check to make sure the cast can work
  - requires that the object you're casting has polymorphic type, i.e. has at least one virtual method