Classes w/ dynamic memory

- Constructors: are called when a variable is created. E.g.,

```
vector v;
// before any other member functions are
// called, vector::vector will be called
// to set things up in a "sane" way.
// Another way, they establish "class invariant"
```

- copy constructor: used to make copies.

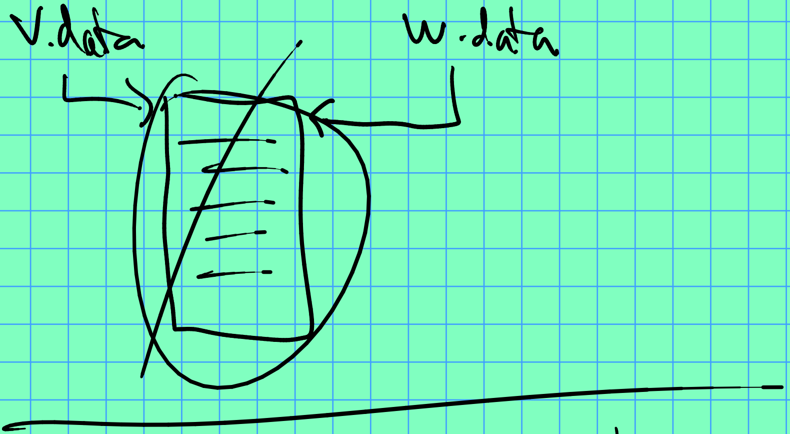
  E.g.
```
vector v;
  // do stuff to v...
vector w(v);
```
  Also used automatically when you
    - call by value
    - return by value
    - temporary objects...

Why do we need a copy constructor??

```
void f(vector w) {
    // stuff...
} ← destructor for w will
    be called.
int main() {
    vector v;
    v.push_back(...);
        :
    f(v);
      :
}
```

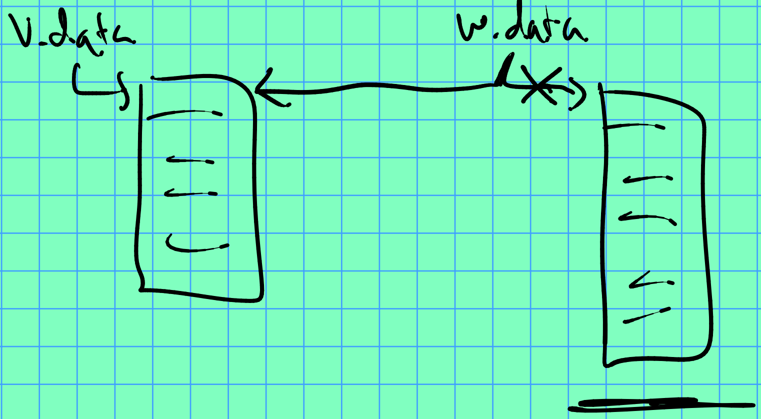
Default copy constructor:

```
vector::vector(vector& v) {
    size = v.size;
    capacity = v.capacity;
    data = v.data;
}
```



Similar issues arise w/ the assignment operator:

W = V ;



Same issue — double free.

Also W.data is "lost"
so you can't free it.
Could also lead to segmentation
fault (put this in a
function call...)

(last thing you always need: destructor.
But we've been assuming that was implemented
all along.)