

C++ idioms/general concepts

- **Genericity**
 - Templates, STL, containers, algorithms
- **Copy/Assignment/Memory**
 - Deep copy model, memory management “required”
- **Low-level structures**
 - C-style arrays and strings compared to STL, Tapestry
- **const**
 - Good for clients, bad for designers/coders?
- **From C to C++ to Java**
 - function pointers, function objects, inheritance

C++ idioms

- **What happens with the statement `myDay = d; ?`**
 - assignment is memberwise unless operator = overloaded
 - copy constructor used in passing parameters by value
- **If you need one of: destructor, assignment operator, copy constructor, you need all of them**
 - heuristic only: managing resources other than memory
 - preventing objects from being copied
 - what about non-copyable state, e.g., stream
- **In assignment operator, watch for self-assignment**
- **Study implementation of string/vector**

copy constructor

- Used for “first-time” creation

```
Date d(1,1,2000);  
Date copy(d);
```

- Used for pass-by-value

```
DoStuff(Date d);  
//...  
Date first(1,1,2000);  
DoStuff(first);
```

- what about use of myLength in code as opposed to length()?

```
template <class Item>  
tvector(const tvector<Item> & vec)  
// precondition: Item supports assignment  
// postcondition: return copy of vec  
{  
    // allocate storage  
    myList = new  
        Item[myLength=vec.myLength];  
    assert(myList != 0);  
    // copy elements  
    for(int k = 0; k < vec.myLength; k++)  
    {  
        myList[k] = vec.myList[k];  
    }  
}
```

Assignment operator

- We want to have deep copy when assigning as well as when we copy

```
Object x(23,4);  
Object y;  
y = x;           // assignment operator  
Object z = x;    // copy constructor!!!  
z = y = x;       // how does this work?
```

- Assignment operator returns `*this`
 - Won't be const reference return, will be reference
- Assignment operator checks for not assigning to self
 - Can assign to self via aliasing, e.g., pass-by-reference
- Assign to every data member (deep copy as needed)
- See `tvector` for details

Destructor

- If you need copy constructor, you need assignment operator, and you need destructor
- What is purpose of destructor?
 - Free resources
 - What's a resource: memory, files, network connections
- When is the destructor called?
 - Automatically when a stack object goes out of scope
 - When you call delete on a heap object
- What's the problem with this "automatic destruction"?
 - It's not automatic, it's fraught with problems getting it right
- What about yahoo and the rolling reboot?