

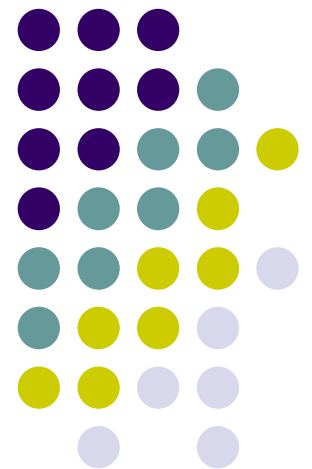
Advanced C++

June 25, 2009

Mike Spertus

mike_spertus@symantec.com

YIM: spertus





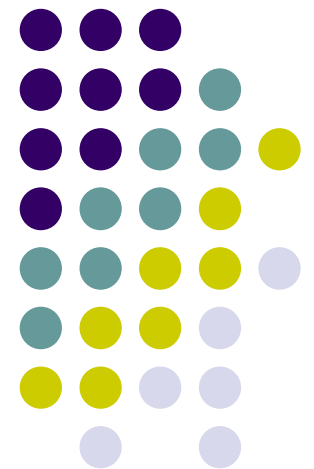
Static vs. Runtime-Types

- Static (“Compile-time”) type of an object is the type of the expression used to reference it in the source
- Dynamic (“Run-time”) type of an object is the type used to create the object when the program runs
- If a method call on an object is non-virtual, then the static type is use. If the method call is virtual, then the run-time type is used
- ```
class D : public B {...};
B *b = new D;
```
- Static type of `*b` is `B`, Run-time type of `*b` is `D`
- `dynamic_cast` uses RTTI to change the static type of a pointer as long as the run-time type is compatible
  - but only works if the classes have virtual functions

# More Best Practices

---

As Numbered in Effective C++



# Item 1: View C++ as a federation of languages



- C
- Object-Oriented C++
- Template C++
- The STL

# Item 5: Know what functions C++ silently writes and calls



- `class Empty {};` is the same as
- ```
class Empty {  
    public:  
        Empty() {} // Because no  
                // constructors defined  
        Empty(const Empty &rhs) {}  
        ~Empty() {};  
        Empty &operator=(const Empty&) {}  
};
```



Item 5 (Cont)

- Method hiding saves the day
- ```
class Base { .. };
class Derived : public Base { ...};
// Below legal without method hiding!
Derived d = b;
```

## Item 6: Explicitly disallow the use of compiler-generated functions you do not want



- ```
class Singleton {  
private:  
    // Don't define anywhere  
    Singleton(const Singleton &);  
    Singleton &  
        operator=(const Singleton &);  
};
```

Item 8: Prevent exceptions from leaving destructors



- Because unwinding the stack during exceptions calls destructors (this is why RAII works), there may be two simultaneously active exceptions, in which case the program either terminates or is undefined

Item 9: Never call virtual functions during construction and destruction



- We spent time on this last quarter



Assignment best practices

- Item 10: Have assignment operators return a reference to `*this`.
 - So `a = b = c;` works
- Item 11: Handle assignment to self in `operator=`.

```
struct Widget {  
    Bitmap *pb;  
}  
Widget &  
Widget::operator=(const Widget &r) {  
    delete pb; // Fails on self-assignment  
    pb = new Bitmap(*rhs.pb);  
    return *this; // Good. Item 10  
};
```
- Item 12: Copy all parts of an object

Item 13: Use objects to manage resources



- RAII
- Use `boost::scoped_array` and `boost::shared_array` instead of `auto_ptr` and `shared_ptr` for arrays to make sure proper delete statement is called

How to pass

- See Items 20 and 21



Item 46: Define non-member functions inside templates when type conversions are desired



- The following code correctly resolves `Complex(3,1) + 2`

```
template <typename T>
class Complex {
public:
    Complex ();
    Complex (typename call_traits<T>::param_type R,
            typename call_traits<T>::param_type I = 0)
        : Real(R), Imag(I) {}
    T Real;
    T Imag;
    friend const Complex<T>
operator+(const Complex<T> &l, const Complex<T> &r) {
    return Complex(l.Real+r.Real, l.Imag+r.Imag);
}
};
```



Class and HW information

- Please email HW to mike_spertus@symantec.com before start of class
- You may need to install Boost libraries from www.boost.org to do HW problems
 - Note that Boost may already be installed on Cluster machines
- Class grade will be 2/3 HW, 1/3 Final or Final project

Exercise 11.1 (Extra Credit)



- From Herlihy and Shavit, *The Art of Multiprocessor Programming*.
- You are one of P recently arrested prisoners. The prison has a tower, with a switch that can be switched up or down by the occupant of the tower. The switch is initially in the down position.
- Periodically, the Warden sends a prisoner into the tower for a brief while. The Warden announces that all the prisoners will be sent free if a prisoner can correctly assert that all prisoners have been in the tower. An incorrect assertion results in everyone being executed.
- Before being sent into isolated cells, the prisoners are allowed to confer on a strategy. Afterwards, they never see each other again.
- What strategy should the prisoners follow?

Exercise 11.2 (Extra Credit)



- From Herlihy and Shavit, *The Art of Multiprocessor Programming*.
- You are one of P recently arrested prisoners. The warden, a deranged computer scientist, orders the prisoners to stand in line, and places red and blue hats on each of their heads. No prisoner knows the color of his own hat, or the color of any hat behind him, but he can see the hats of the prisoners in front. The warden starts at the back of the line and asks each prisoner to guess the color of his own hat. The prisoner can answer only "red" or "blue." If he gives the wrong answer, he is fed to the crocodiles. If he answers correctly, he is freed. Each prisoner can hear the answer of the prisoners behind him, but cannot tell whether that prisoner was correct.
- The prisoners allowed to consult and agree on a strategy beforehand (while the warden listens in) but after being lined up, they cannot communicate any other way besides their answer of "red" or "blue."
- Devise a strategy that allows at least $P - 1$ of P prisoners to be freed.