C++ Craft: #4

Классы

Тестирование

Язык Программирования C++

- Бьерн Страуструп
  - глава «Классы».

Code Review

1. Повторное использование кода

```
class Complex
{
public:
        Complex( double real, double imaginary = 0 )
…
```

2. explicit

```
explicit complex( double real, double imaginary = 0 )
        : real_( real )
        , imaginary_( imaginary )
{
}
```

3. const &

```
void operator+( const complex& other )
{
        _real = _real + other._real;
        _imaginary = _imaginary + other._imaginary;
}
```

4. a op= b

```
T& T::operator+=( const T& other )
{
        //...
        return *this;
}
const T operator+( const T& a, const T& b )
{
        T temp( a );
        temp += b;
        return temp;
}
```

5.Умело выбирайте члены класса

Члены класса:

- operator =
- operator ()
- operator []
- operator ->
- operator new
- operator new[]
- operator delete
- operator delete[]
- если нельзя сделать **не** членом класса

Не члены класса:

- operator>> и operator<<
- требует приведения левого аргумента (a = 1.0 + b)
- может быть реализован через публичный интерфейс

6.friend

```
void my_method( int,
                const double,
                const std::string& );

class Boy
{
       // …
};
class Girl
{
      friend void my_method( int,
                             const double, const
                             std::string& );
      friend class Boy;
private:
      void boy_could_run_this_method() const;
};
```

7. operator+ через operator+=

```
T const complex operator+( const complex& lhs,
                           const complex& rhs )
{
        complex ret( lhs );
        ret += rhs;
        return ret;
}
```

8. return out(in)

```
std::ostream& print( std::ostream& os ) const
{
        os << "(" << real_ << ","
           << imaginary_ << ")";
        return os;
}
```

9. operator++()

```
complex& operator++()
{
        ++real_;
        return *this;
}
```

10. operator++(int)

```
const complex operator++( int ) // a++++++;
{
        complex temp( *this );
        ++*this; // preincrement
        return temp;
}
```

11. Не используйте зарезервированные имена

```
double real_;
double imaginary_;
```

12. Константные методы

```
class complex
{
      // …
      const double real() const
      {
            return real_;
      }
};
```

13. mutable

```
class test_class
{
     mutable boost::mutex my_protector_;
     mutable int b_;
     const int b() const
     {
            boost::mutex::scoped_lock lock( my_protector_ );
            return ++b_;
     }
};
```

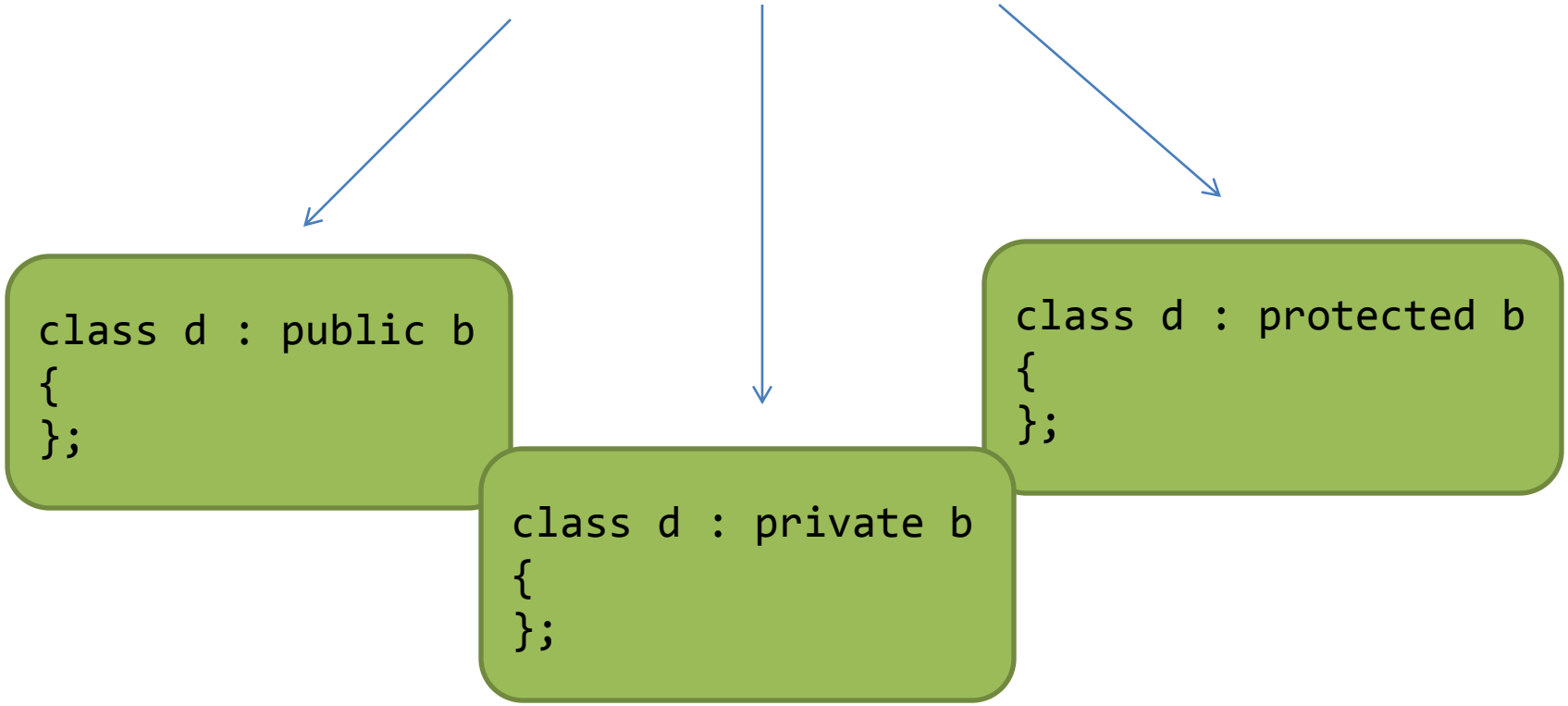
14. #ifndef #define … #endif

```
#ifndef _CLASS_DESIGN_COMPLEX_GOOD_COMPLEX_HPP_
#define _CLASS_DESIGN_COMPLEX_GOOD_COMPLEX_HPP_

#endif // _CLASS_DESIGN_COMPLEX_GOOD_COMPLEX_HPP_
```

наследование: перегрузка и переопределение

inheritance



virtual destructor

не забывать!

virtual

```
class A
{
public:
      std::ifstream inp_;
      void foo() {}
};

class B : public virtual A
{
};
class C : public virtual A
{
};
```

```
class D : public B, public C
{
     D()
          : A()
     {
     }
};
```

http://en.wikipedia.org/wiki/Virtual_inheritance

abstract

```
class abstract_class //interface
{
public:
     void method() = 0;
     virtual ~abstract_class(){}
};

class realisation : public abstract_class
{
public:
     void method()
     {
     }
     virtual ~ realisation (){}
};
```

http://en.wikipedia.org/wiki/Virtual_method_table

Duck quiz

Принципы

- Инкапсулируйте то, что изменяется
- Отдавайте предпочтение композиции перед наследованием
- Программируйте на уровне интерфейсов

Интересно знать

http://en.wikipedia.org/wiki/Argument-dependent_name_lookup

```
namespace NS
{
        class A {};
        void f( A *&, int ) {}
}
int main()
{
        NS::A *a;
        f( a, 0 ); // как он это делает?
}
```

Преобразование типов

- ~~(int)(variable)~~

- static_cast,
- dynamic_cast,
- reinterpret_cast,
- const_cast

?

Тестирование

- BOOST_TEST_FRAMEWORK
- Gtest
- …

TDD

- Test Driven Development

- Когда тесты используются для разработки приложения.

Проверки

- BOOST_CHECK_EQUAL
- BOOST_CHECK_NO_THROW
- BOOST_CHECK_THROW
- BOOST_CHECK_BITWISE_EQUAL
- BOOST_MESSAGE
- BOOST_ERROR

?