## 1：使用示例分析

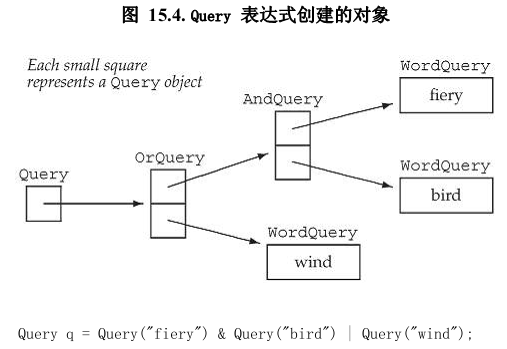
1： **值型句柄**

程序将处理计算查询，而不建立查询，但是，需要能够创建查询以便运行程

序。最简单的办法是编写 C++ 表达式直接创建查询，例如，可以编写这样的代

码：

query q = query("fiery") & query("bird") | query("wind");



* 首先调用query::query(const std::string&)

query("fiery") query("bird") query("wind")分别生产WordQuery基本查询对象

* 然后调用 friend inline query operator&(const query &lhs, const query &rhs);

query("fiery") & query("bird") 使用query("fiery") 和 query("bird")生成AndQuery对象

* 然后调用friend inline query operator|(const query &lhs, const query &rhs);

使用 query("fiery") & query("bird")这个Andquery对象 + 和query("wind")这个Wordquery对象 🡪 OrQuery对象

2：

tq.read\_file(infile);

std::set<TextQuery::line\_no> locs = q.eval(tq);

OrQuery::eval

rhs.eval(file)

lhs.eval(file)

AndQuery::eval

WordQuery::eval

rhs.eval(file)

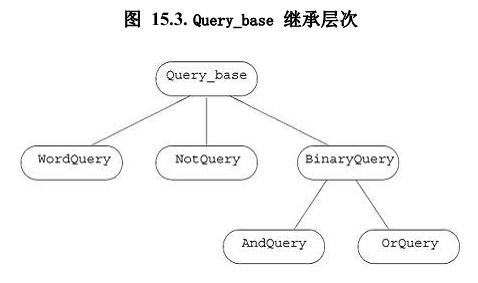
lhs.eval(file)

WordQuery::eval

WordQuery::eval

由于最后返回的是一个OrQuery类型的派生类，即query中query\_base\*中记录的指针是OrQuery派生类的指针，所以q.eval调用的是版本是OrQuery中的版本，OrQuery在使用其记录的两个query对象，其中一个query对象的query\_base\*中记录的指针是AndQuary派生类的指针，同理所以继续调用AndQuary中的版本，最后调用WordQuery中版本返回。其实是一个递归计算的过程。

## 2：继承层次



可以看到程序设计中Query句柄类通过定义抽象基类Query\_base来调用其他的执行具体功能的派生类，这种方式对类的用户来讲是非常友好的。

继承

继承

Query\_base指针

句柄方式

## 3 句柄类与继承

C++ 中面向对象编程的一个颇具讽刺意味的地方是，不能使用对象支持面向

对象编程，相反，必须使用指针或引用。例如，下面的代码段中：

void get\_prices(Item\_base object,

const Item\_base \*pointer,

const Item\_base &reference)

{

// which version of net\_price is called is determined at run time

cout << pointer->net\_price(1) << endl;

cout << reference.net\_price(1) << endl;

// always invokes Item\_base::net\_price

cout << object.net\_price(1) << endl;

}

通过 pointer 和 reference 进行的调用在运行时根据它们所绑定对象的动态类型而确定。

但是，使用指针或引用会加重类用户的负担。在前一节中讨论继承类型对象

与容器的相互作用时，已经碰到了一种这样的负担。

C++ 中一个通用的技术是定义包装（cover）类或句柄类。句柄类存储和管理基类指针。指针所指对象的类型可以变化，它既可以指向基类类型对象又可以指向派生类型对象。用户通过句柄类访问继承层次的操作。因为句柄类使用指针执行操作，虚成员的行为将在运行时根据句柄实际绑定的对象的类型而变化。因此，句柄的用户可以获得动态行为但无须操心指针的管理。

包装了继承层次的句柄有两个重要的设计考虑因素：

• 像对任何保存指针（第 13.5 节）的类一样，必须确定对复制控制做些什

么。包装了继承层次的句柄通常表现得像一个智能指针（第 13.5.1 节）

或者像一个值（第 13.5.2 节）。

• 句柄类决定句柄接口屏蔽还是不屏蔽继承层次，如果不屏蔽继承层次，用

户必须了解和使用基本层次中的对象。

对于这些选项没有正确的选择，决定取决于继承层次的细节，以及类设计者希望程序员如何与那些类相互作用。下面两节将实现两种不同的句柄，用不同的方式解决这些设计问题。

* 指针型句柄 –》c++.mm中大练习：书店购物篮（Sales\_item）可以看到
* 值型句柄-》c++.mm中 大练习：文本查询示例可以看到