make_shared()_任家_新浪博客

shared_ptr很好地消除了显式的delete调用，如果读者掌握了它的用法，可以肯定delete将会在你的编程字典中彻底消失 。

但这还不够，因为shared_ptr的构造还需要new调用，这导致了代码中的某种不对称性。虽然shared_ptr很好地包装了new表达式，但过多的显式new操作符也是个问题，它应该使用工厂模式来解决。

因此，shared_ptr在头文件<boost/make_shared.hpp> 中提供了一个自由工厂函数（位于boost名字空间）make_shared<T>()，来消除显式的new调用，它的名字模仿了标准库的 make_pair()，声明如下：

```
1. template<class T, class... Args>
2.     shared_ptr<T> make_shared( Args && ... args );
```

make_shared()函数可以接受最多10个参数，然后把它们传递给类型T的构造函数，创建一个shared_ptr<T>的对 象并返回。make_shared()函数要比直接创建shared_ptr对象的方式快且高效，因为它内部仅分配一次内存，消除了shared_ptr 构造时的开销。

下面的代码示范了make_shared()函数的用法：

```
1. #include <boost/make_shared.hpp>
2. int main()
3. {
4.     shared_ptr<string> sp =
5.             make_shared<string>("make_shared");
   //创建string的共享指针
6.     shared_ptr<vector<int> > spv =
7.             make_shared<vector<int> >(10, 2);
   //创建vector的共享指针
8.     assert(spv->size() == 10);
9. }
```

make_shared()不能接受任意多数量的参数构造对象，一般情况下这不会成为问题。实际上，很少有如此多的参数的函数接口，即使有，那也会是一个设计的不够好的接口，应该被重构。

除了make_shared()，smart_ptr库还提供一个allocate_shared()，它比make_ shared()多接受一个定制的内存分配器类型参数，其他方面都相同。

分享：