```
//四：智能指针总结
//（4.1）智能指针背后的设计思想
//智能指针主要目的：帮助我们释放内存，以防止我们忘记释放内存时造成的内存泄漏；
//c++ 98 auto_ptr    == unique_ptr
//myfunc();
```

```
//（4.2）auto_ptr为什么被废弃
//auto_ptr:c++98时代的智能指针，具有unique_ptr一部分特性； unique_ptr,shared_ptr,weak_ptr:
//不能在容器中保存，也不能从函数中返回auto_ptr;
//auto_ptr<string> ps(new string("I Love China"));
//auto_ptr<string> ps2 = ps; //ps2指向字符串，ps变成空了，这可以防止ps和ps2析构一个string两次；
//用ps（你没有意识到ps已经空了），代码就会崩溃；
//这个也是auto_ptr用法上的一个陷阱。
```

```
//shared_ptr<string> ps(new string("I Love China"));
//shared_ptr<string> ps2 = ps; //ps2和ps都有效，强引用计数为2；
//unique_ptr<string> ps(new string("I Love China"));
//unique_ptr<string> ps2 = ps;  //编译出错；

//虽然auto_ptr和unique_ptr都是独占式的，但unique_ptr这种情况，编译的时候就会报错；
//而不会默默的把ps的所有权转移到ps2上，避免后续如果使用ps导致程序崩溃的问题；
//当然如果你用移动语义，也能达到auto_ptr的效果：
unique_ptr<string> ps(new string("I Love China"));
unique_ptr<string> ps2 = std::move(ps); //运用了移动语义
```

```
//auto_ptr被废弃的主要原因是：设计的不太好，容易被误用引起潜在的程序崩溃等问题。所以c++11启用了unique_ptr来取代auto_ptr
//c++11表示不建议再使用auto_ptr，老师强烈建议大家，用unique_ptr取代；unique_ptr比auto_ptr更安全；
```

```
//（4.3）智能指针的选择
//shared_ptr,unique_ptr;
//a)如果程序要使用多个指向同一个对象的指针，应该选择shared_ptr;
//b)如果程序不需要多个指向同一个对象的指针，应该首选unique_ptr;
```