条款5：优先使用auto而非显式类型声明

① 类型名字过长!!!

欣赏一下C++给你的大份

```
typename std::iterator_traits<It>::value_type currentI;
```

没错，上述语句仅仅声明了一个变量，而使用auto可以让一切交给编译器

② C++14后lambda中形参也可使用auto

```
auto fun=[](const std::unique_ptr<Widget> &p1,
            const std::unique_ptr<Widget> &p2){
return *p1 < *p2}
```

auto

③ lambda表达式的返回值一定要用auto.

②中的auto可替换为什么呢

lambda为匿名函数，底层生成匿名类，因此根本无法获得类的名字

但可使用 `std::function<bool(const std::unique_ptr<Widget>&, const std::unique_ptr<Widget>&)> fun`

好了，又变成大份了，std::function还会额外性能损耗

④ 与类型快捷方式有关的问题

```
std::vector<int> V;      unsigned sz = V.size();
```

V.size()的标准返回类型为std::vector<int>::size_type

而在不同的系统上unsigned与std::vector<int>::size的大小可能不同

意味着当s1足够大时，数据可能会溢出使s2成为乱码，一旦之后再使s1必定出现bug，并且不好复现

⑤避免类型写错而导致的无用拷贝

```
int a=10; float b= &a;  const float c=&a;
```

```
std::map<std::string, int> testmap;
for(const std::pair<std::string, int> &p : m){};
```

错误写法，出现多余拷贝