A.7 自动推导变量类型

C++ 是静态语言：所有变量的类型，都会在编译时被准确指定。所以，作为程序员你需要为每个变量指定对应的类型。

有些时候就需要使用一些繁琐类型定义，比如：

```
std::map<std::string,std::unique_ptr<some_data>> m;
std::map<std::string,std::unique_ptr<some_data>>::iterator
      iter=m.find("my key");
```

常规的解决办法是使用typedef来缩短类型名的长度。这种方式在 C++ 11中仍然可行，不过这里要介绍一种新的解决办法：如果一个变量需要通过一个已初始化的变量类型来为其做声明，那么就可以直接使用 auto 关键字。这样，编译器就会通过已初始化的变量，去自动推断变量的类型。

```
auto iter=m.find("my key");
```

当然， auto 还有很多种用法：可以使用它来声明const、指针或引用变量。这里使用 auto 对相关类型进行了声明：

```
auto i=42; // int
auto& j=i; // int&
auto const k=i; // int const
auto* const p=&i; // int * const
```

变量类型的推导规则是建立一些语言规则基础上：函数模板参数。其声明形式如下：

```
some-type-expression-involving-auto var=some-expression;
```

var变量的类型与声明函数模板的参数的类型相同。要想替换 auto ，需要使用完整的类型参数：

```
template<typename T>
```

```
void f(type-expression var);
f(some-expression);
```

在使用 auto 的时候，数组类型将衰变为指针，引用将会被删除(除非将类型进行显式为引用)，比如：

```
int some_array[45];
auto p=some_array; // int*
int& r=*p;
auto x=r; // int
auto& y=r; // int&
```

这样能大大简化变量的声明过程，特别是在类型标识符特别长，或不清楚具体类型的时候(例如，调用函数模板，等到的目标值类型就是不确定的)。