优先使用C++的别名声明（using）来替换typedef

Posted on 2021-04-24 14:56 闪之剑圣 阅读(99) 评论(0) 编辑 收藏 举报

C++98中，我们如果想用简写的方式表达一个类型，那么可以使用typedef关键字：

```
typedef std::unique_ptr<std::unordered_map<std::string, std::string>> UPtrMapSS;
```

C++11中，提供了别名声明的语法来取代typedef:

```
using UPtrMapSS = std::unique_ptr<std::unordered_map<std::string, std::string>>;
```

既然这两种语法都可以代表对一种类型的简写，那为什么一定要用using而不是typedef呢?
理由之一是，用using来声明函数指针时会更直观一些：

```
typedef void (function*) (int, int);
using function = void (*) (int, int);
```

直观与否，见仁见智，这其实不是一个非常必要的原因。using真正强大的地方在于它对模板的支持。
比方说我们要声明一个模板类型的简写，用using可以非常简单地实现：

```
template<typename T>
using MyAllocList = std::list<T, MyAlloc<T>>;

MyAllocList<Widget> lw;
```

如果要用typedef的话，就会变得非常麻烦了：

```
template<typename T>
struct MyAllocList{
  typedef std::list<T, MyAlloc<T>> type;
};
MyAllocList<Widget>::type lw;
```

除此之外，如果我们想在模板类中使用它，且MyAllocList的类型T是从模板形参指定的话，我们还要加上typename前缀：

```
template<typename T>
class Widget
{
  private:
    typename MyAllocList<T>::type l;
}
```

之所以要加typename是因为在这里编译器无法分辨MyAllocList::type究竟是MyAllocList里定义的类型还是一个成员变量，因此需要手动通过typename来标识一下。
如果使用的是using就完全不用关心这个问题，因为using本身就是在告诉编辑器这是一种类型。

```
template<typename T>
using MyAllocList = std::list<T, MyAlloc<T>>;

template<typename T>
class Widget
{
  private:
    MyAllocList<T> l;
}
```

因此在日常写代码中，还是应当尽量都转换到以using来声明类型的别名，而不再使用typedef。

分类: C++ Tricks