# Using

## 定义别名

C++11中using可以用来定义别名，特别适用于在模板的情况下：

using u4 = uint32\_t;

using u8 = uint64\_t;

重定义模板,使得使用的时候更方便：

template <typename T>

using MyVector = std::vector<T, MyAllocatorAdapter<T>>;

如果不适用using，那么还可以采用旧的方式：

typedef uint32\_t u4;

typedef uint64\_t u8;

<https://blog.csdn.net/chen802311/article/details/53008118>

定义一个具体类型的别名的时候，typedef与using是一样的；

对于模板的话，只能用using，不能用typedef.

代码示例：

*#include <iostream>*

*using namespace std;*

*// using用于定义一个具体类型的别名， using 与 typedef 一样的效果*

*// 但是模板的别名只能用using（如果typedef也能的话，cpp11就不会加using这个特性了）*

*// 以下效果一样*

*using u\_flags = std::ios\_base::fmtflags;*

*typedef std::ios\_base::fmtflags f\_flags;*

*using u\_func = void(\*) (int, int); // 感觉using方式看上去更优雅*

*typedef void(\*f\_func) (int, int);*

*template<typename T>*

*struct Node { T val; };*

*template<typename T>*

*using u\_Test = Node<T>; // ok*

*//template<typename T>*

*//typedef Node<T> f\_Test; // error，因为 Node<T> 不是一个具体类型*

*// 这样：*

*typedef Node<int> f\_Test; // ok, 具体类型*

*// 模板别名*

*template<class T>*

*using ptr = T\*;*

*int main(){*

*u\_Test<int> t1;*

*f\_Test t2;*

*ptr<int> x;*

*int ntmp = 90;*

*x = &ntmp;*

*cout << \*x << endl;*

*return 0;*

*}*