```cpp
//using在用于定义类型（定义类型模板）的时候，是包含了typedef的所有功能的。
typedef unsigned int uint_t;
using uint_t = unsigned int;

typedef std::map<std::string, int> map_s_i;  //typedef定义类型的方法感觉像 定义一个变量 ：类型名 变量名
using map_s_i = std::map<std::string, int>;

typedef int(*FuncType)(int, int); //用typedef来定义函数指针
using FuncType = int(*)(int, int); //using定义类型的定义方法感觉像赋值。
```

```cpp
////c++11
//template<typename T>
//using str_map_t = std::map<std::string, T>;  //str_map_t是类型别名。
//                                             //using用来给一个“类型模板”起名字（别名）用的

template <typename T>
using myfunc_M = int(*)(T, T);  //定义类型模板，    是个函数指针模板

//using FuncType = int(*)(int, int);
```

```cpp
//c++98
template <typename wt>
struct map_s //定义了一个结构/类，只是结构的成员缺省都是public:
{
    typedef std::map <std::string, wt> type; //定义了一个类型    ，键 ，值
};

//c++11
template<typename T>
using str_map_t = std::map<std::string, T>;   //str_map_t是类型别名。
```