decltype关键字进行编译时类型推导，这点类似auto，但decltype以表达式做为参数进行类型推导，而auto从变量声明的初始化表达式获得变量的类型。

decltype不对表达式进行求值。

例：程序stl\_test11

*map*<*string*, float> coll;

coll.*insert*(*pair*<*string*, float>("abc", 4));

coll["bcd"] = 5;

// 通过decltype获取类型

typedef decltype(coll.*begin*()) map\_type;

for (map\_type i = coll.*begin*(); i != coll.*end*(); ++i)

{

*cout* << (\*i).*first*.*c\_str*() << " " << (\*i).*second* << *endl*;

}

scoped enum：域化枚举，强类型枚举

// 非scoped enum

enum Color

{

yellow,

green

};

// auto yellow = false; // error，重定义枚举数

Color c = Color::green;

if (c < 14.5) // ok，c可以隐式转换为整型

{

}

// scoped enum

enum class Color1

{

black,

white

};

auto black = false; // ok

Color1 c1 = Color1::black;

//if (c1 < 14.5) // error，c1不能隐式转换为整型

//{

//}

关键字typename用来指明紧跟其后的是个类型。

template <typename T>

class MyClass {

typename T::SubType\* ptr;

}

在这里typename用来说明SubType是个类型，定义在Class T内。因此，ptr是一个指针，指向类型为T::SubType。如果没有typename，SubType会被视为类型为T的一个MyClass的静态成员。

在C++中，template内的任何标识符都被视为一个value，除非为它加上typename。