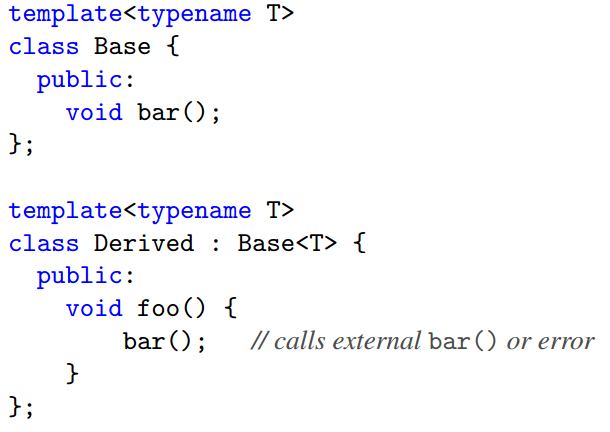
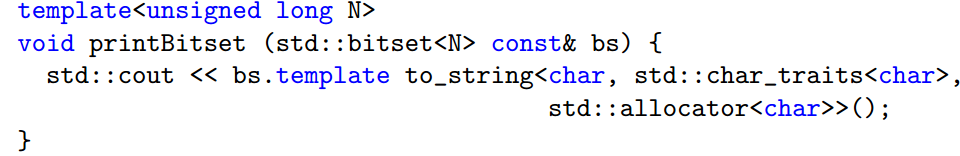
1. C++中，可以在编译期进行操纵的实体称为元数据（metadata），大致分两范畴：类型（types）和非类型（non\_types）。
2. Using this->. For class templates with base classes that depend on template parameters, using a name x by itself is not always equivalent to this->x, even though a member x is inherited. For example:

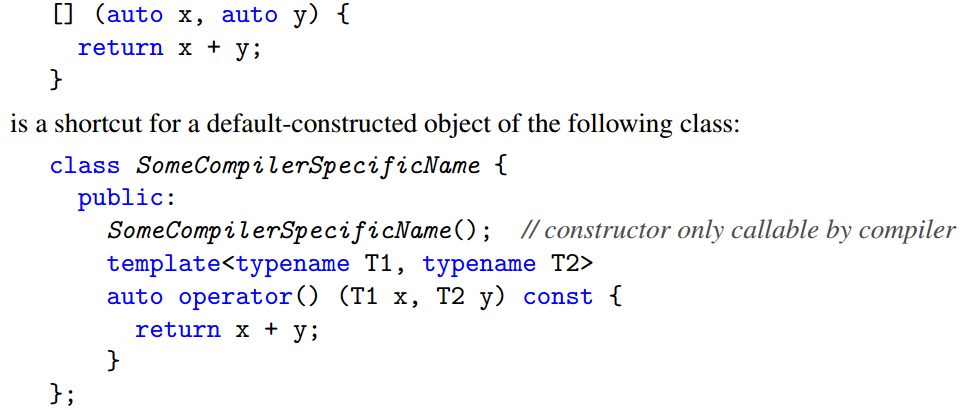


As a rule of thumb, we recommend that you always qualify any symbol that is declared in a base that is somehow dependent on a template parameter with this-> or Base<T>::.

1. Templates for raw arrays and string literals
2. The .template construct. Sometimes, it is necessary to explicitly qualify template arguments when callign a member template. In that case, you have to use the template keyword to ensure that a < is the beginning of the template argument list.



Generic Lambdas and Member Templates.



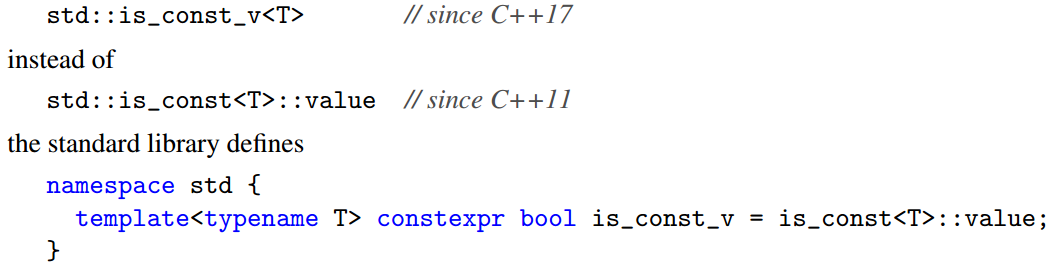
1. Variable templates

Variables also can be parameterized by a special type. Such a thing is called a variable template. Yes, we have very similar terms for very different things: A variable template is a variable that is a template (variable is a noun here). A variadic template is a template for a variadic number of template parameters (variadic is an adjective here).

For example, you can use the following code to define the value of pi while still not defining the type of the value:



Type traits suffix \_v:

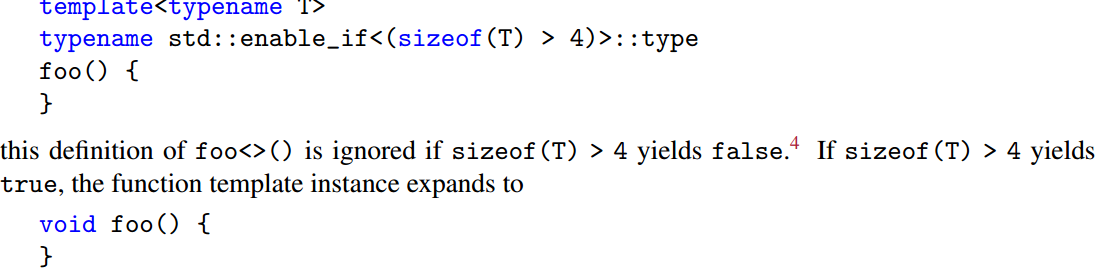


1. Move semantics：

You can use it to optimize copying and assignments by moving (“stealing”) internal resources from a source object to a destination object instead of copying those contents. This can be done provided the source no longer needs its internal value or state (because it is about to be discarded).

1. Disable templates with enable\_if<>

C++ std lib provides a helper template std::enable\_if<> to ignore function templates under certain compile-time conditions.



If the expression yields false, the member type is not defined. Due to a template feature called SFINAE (substitution is not an error), this has the effect that the function template with the enable\_if expression is ignored.

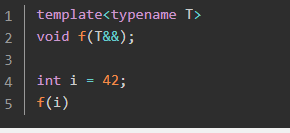
1. C++11 move 和forward

新规则：

1：引用折叠：如果间接创建一个引用的引用，则这些引用就会“折叠”。所有情况下（除了一个例外），引用折叠成一个普通的左值引用类型。一种特殊情况下，引用会折叠成右值引用，即右值引用的右值引用，T&& &&

* X& &、X& &&、X&& &都折叠成X&
* X&& &&折叠为X&&

2：（右值引用的特殊类型推断）：当将一个左值传递给一个参数是右值引用的函数时，且此右值引用指向模板类型参数（T&&），编译器推断模板参数类型为实参的左值引用，如：



3：虽然不能隐式将一个左值转换为右值引用，但可以通过static\_cast显示地将一个左值转换为一个右值。

std::move执行到右值的无条件转换，std::forward执行到右值的有条件转换，在参数都是右值时，二者就是等价的。

1. Using concepts to simplify enable\_if<>Expression

