通过对第六章的学习，我了解到了什么是模板和异常处理。模板是泛型编程的基础，泛型编程即以一种独立于任何特定类型的方式编写代码。模板是创建泛型类或函数的蓝图或公式。库容器，比如迭代器和算法，都是泛型编程的例子，它们都使用了模板的概念。每个容器都有一个单一的定义，比如 向量，我们可以定义许多不同类型的向量，比如 vector <int> 或 vector <string>。“C++ 异常处理涉及到三个关键字:try、catch、throw。 throw:当问题出现时,程序会抛出一个异常。这是通过使用throw关键字来完成的。 catch:在您想要处理问题的地方,通过异常处理程序捕获异常。catch关键字用于捕获异常。 try:try块中的代码标识将被激活的特定异常。它后面通常跟着一个或多个 catch 块。 如果有一个块抛出一个异常,捕获异常的方法会使用try和catch关键字。

通过对这一章的学习我对c++这门编程语言有了跟更加深刻的了解，让我对掌握c++这门编程语言有了更大的信心。利用 C++ 模板，您可以定义可对不同类型的信息运行的一系列函数或类。 Microsoft C++ 模板的实现基于 ISO/ANSI C++ 标准。在导致重复多个类型的同一代码的情况下应使用模板。 例如，可以使用函数模板创建将相同的算法应用于不同数据类型的函数集。 还可以使用类模板开发 typesafe 类集。 模板有时候是比 C 宏和 void 指针更好的解决方案，当与集合（MFC 中的模板的主要用途之一）和智能指针一起使用时，模板尤其有用。

模板（有时称为参模板（有时称为参数化类型）是用于生成基于类型参数的函数和类的机制。通过使用模板，可以设计操作多种类型的数据的单个类或函数，而不必为每种类型创建单独的类。明和定义模板在某种程度上类似于其他函数和类，同时存在一些主要差异。模板声明不完全定义函数或类；它仅定义类或函数的语法主干。通过称作实例化的过程从模板中创建实际类或函数。创建的单个类或函数被引用为已实例化的。例如，类模板：

template <class T> struct A { . . . };

可用于实例化 A<int>、A<char>、A<int\*>、A<MyClass\*> 等的类。

可以显式或隐式执行类或函数的实例化。显式实例化是一种对外调用代码中将生成模板的版本的方式。隐式实例化允许在第一次使用模板的位置根据需要实例化模板。模板也可以由值参数进行参数化，在这种情况下，声明模板参数的方式与声明函数的参数的方式类似。禁止将浮点型和类类型作为值参数。模板的常见问题是，它们可以是通用型解决方案，这意味着同一代码适用于所有类型。如果需要自定义特定类型的模板的行为，则可以使用专用化。通过使用显式专用化，模板可专用于特定的实际类型而不是泛型类型。也可以对类模板进行部分专用化，这在您的模板有多个类型参数且您只想自定义与该模板的部分而非所有参数相关的行为时会很有用。部分专用化仍为泛型且需要实际模板参数才能生成实际的实例化类。

template 声明指定参数化类或函数的集

“模板参数列表” 是模板参数的逗号分隔的列表，它可能是在窗体“类”、“标识符”、“类型名称” “标示符” 或 模板<”模板参数列表“>类“标示符” 中的类型或在模板主体中使用的非类型参数。模板参数的语法是以下的其中一个：

必须实例化类模板，就像您要实例化普通类一样，但是在尖括号 (<>) 内必须包括模板参数。这些模板参数可以是任何类型，如果模板参数列表包含类或“类名称”关键字，如果该参数是非类型参数则为合适类型的值。不需要特殊语法调用函数模板，尽管，可能需要尖括号和模板参数，如果模板参数不能从参数推导到函数。

“模板参数列表” 是指定紧跟其后的代码哪一个部分将改变的模板函数所使用的参数列表。例如：

template< class T, int i > class MyStack...

在这种情况下，该模板可以接到类型 (class T) 和常量参数 (int i)。在实例化时，模板将使用类型 T 和常量整数 i。在 MyStack 声明体内，您必须引用T 标识符。

模板声明本身不生成代码；它指定类或函数的系列以及当被其他代码引用时生成的一个或多个代码。

模板声明有全局、命名空间或类范围。它们不能在函数内声明。