**C++ STL学习**

C++标准库的出现相当程度上改变了C++程序的风貌以及学习模式，整个标准库没有太多的面向对象(OOP)，倒是无处不在的泛型编程(GP)，标准库的所有组件包括熟知的string、stream，占比约标准库80%的STL，全部以template覆盖。作为C++编程者，想要充实数据结构与算法，提高泛型编程技法，追踪STL源码是提升功力的最佳路线。借用林语堂的一段话与大家共勉，使用一个东西,却不明白它的道理,不高明！只知道 how，不知道 why，当养成了习惯之后，可能会默认成公理，而真正思考起来，却发现若有所失的样子。

STL提供六大组件，六大组件相互关联，容器通过配置器取得数据存储空间、算法通过迭代器存取容器的内存、仿函数可以协助算法完成不同的策略变化、适配器可以修饰容器、仿函数、迭代器。本文以STL六大部件出发,了解STL内部的关系。

1.容器（containers）：置物之所也，数据的特定排列方式，有利于搜寻或排序或其他特殊目的，STL容器即将运用最广的一些数据结构实现出来。STL容器分为序列式容器：array，vector，list，forwardlist deque。关联式容器：set，map，multiset，multimap，unorderedset，unordermap，unorder\_multiset，unorder\_multimap。从实现的角度来看，STL容器是一种class template。就体积而言，这一部分很像冰山在海面下的比率。STL容器又可以衍生出一些其他容器，也可称为适配器，这里的衍生，并非继承派生关系，而是内含关系，例如heap内含一个vector，priority-queue内含一个heap，stack和queue都含一个deque**.**

2.算法（algorithms）：算法，问题之解法也，以有限的步骤解决逻辑或数学问题，STL各种常用算法如sort，search，copy，erase…。从实现的角度来看，STL算法是一种function template。

3.迭代器（iterators）：算法用来对付容器的,处理容器。算法与容器各自闭门造车,各自独立,算法看不见容器，对其一无所知，算法所需要的一切信息都必须从迭代器取得,而迭代器（由容器提供）必须能够回答算法的所有提问,才能搭配该算法的所有操作。如果算法提出问题,迭代器回答不了,那么编译到那一行就出错了。扮演容器与算法之间的胶合剂，是所谓的“泛型指针”。共有五种类型，以及其它衍生变化。从实现的角度来看，迭代器是一种将operator\*，operator->，operatort+，operator--等指针相关操作予以重载的class template。所有STL容器都附带有自己专属的迭代器——是的，只有容器设计者才知道如何遍历自己的元素。

4.仿函数（functors）：仿函数只为算法服务,当算法需要一些特定的准则/操作,就可以利用函数/仿函数。从实现的角度来看，仿函数是一种重载了operator（）的class或class template.一般函数指针可视为狭义的仿函数。仿函数可拥有自己的成员变量，这意味着仿函数可以保存状态。一般函数中如果需要保存状态，则需要使用全局变量。仿函数可以定义内部的类型，也可以继承。仿函数可以包装函数指针、使调用更安全

5.配接器（adapters）：把某一个既有的东西,已经存在的东西，功能也不错的东西稍微改造一下,比如函数的接口改一下,类的名称改一下等。用来修饰容器、仿函数或迭代器（iterators）接口的东西。例如，STL提供的queue和stack，虽然看似容器，其实只能算是一种容器配接器，因为它们的底部完全借助deque，所有操作都由底层的deque供应。改变functor接口者，称为function adapter；改变container 接口者，称为container adapter；改变iterator接口者，称为iterator adapter。对于配接器只要抓住配接器的关键共性,思路就豁然开朗。例A把B改造之后,A就代表B了。A就要面对大家,给大家使用，大家就不会再看到B了,而是会看到A。但是A自己所做的主要的事情,就是交给B去做,因为B是本来就有的一个好东西。所以A身为一个桥梁,是使用者与隐藏在幕后的B中间的一个桥梁。这就是A的作用,也就是adapter的作用。

A要去取用B的功能。我们在编程的技术上，技巧上。两种做法。一个就是用继承的方式的。A继承了B,当然也就可以取用它所有的资源。另外一种方式就是A内含B,我们就是要拥有或复合。

6.配置器（allocators）：STL使用者基本接触不到配置器，我们在程序开发中经常需要频繁动态申请，释放很多的小块内存，会出现内存碎片，频繁调用malloc\free，导致系统性能下降。allocators作为STL中很重要的一个幕后英雄，负责空间配置与管理。从实现的角度来看，配置器是一个实现了动态空间配置、空间管理、空间释放的class template。