深入 C++构造函数

通常很多 C++程序员存在两种误解：

- 没有定义默认构造函数的类都会被编译器生成一个默认构造函数。
- 编译器生成的默认构造函数会明确初始化类中每一个数据成员。

在读《深度探索 C++对象模型》之前，我一直停留在上述二种误解上，所幸的是 Lippman 为我破除了藩篱。下面的部分我将随《深度探索 C++对象模型》对 C++默 认构造函数一探究竟。

C++标准规定：如果类的设计者并未为类定义任何构造函数，那么会有一个默认 构造函数被暗中生成，而这个暗中生成的默认构造函数通常是不做什么事的(无 用的)，下面四种情况除外。

换句话说，有以下四种情况编译器必须为未声明构造函数的类生成一个会做点事 的默认构造函数。我们会看到这些默认构造函数仅"忠于编译器"，而可能不会按 照程序员的意愿程效命。

1.包含有带默认构造函数的对象成员的类

若一个类 X 没有定义任何构造函数,但却包含一个或以上定义有默认构造函数的 对象成员，此时编译器会为 X 合成默认构造函数，该默认函数会调用对象成员的 默认构造函数为之初始化。如果对象的成员没有定义默认构造函数，那么编译器 合成的默认构造函数将不会为之提供初始化。例如类 A 包含两个数据成员对象， 分别为：`string str` 和 `char *Cstr`，那么编译器生成的默认构造函数将只提 供对 string 类型成员的初始化，而不会提供对 char*类型的初始化。

假如类 X 的设计者为 X 定义了默认的构造函数来完成对 str 的初始化，形 如： `A::A(){Cstr="hello"};`因为默认构造函数已经定义，编译器将不能再生成一 个默认构造函数。但是编译器将会扩充程序员定义的默认构造函数——在最前面插 入对初始化 str 的代码。若有多个定义有默认构造函数的成员对象，那么这些成员 对象的默认构造函数的调用将依据声明顺序排列。

2.继承自带有默认构造函数的基类的类

如果一个没有定义任何构造函数的类派生自带有默认构造函数的基类，那么编译 器为它定义的默认构造函数，将按照声明顺序为之依次调用其基类的默认构造函 数。若该类没有定义默认构造函数而定义了多个其他构造函数，那么编译器扩充 它的所有构造函数——加入必要的基类默认构造函数。另外，编译器会将基类的默 认构造函数代码加在对象成员的默认构造函数代码之前。

3.带有虚函数的类

带有虚函数的类，与其它类不太一样，因为它多了一个 vptr，而 vptr 的设置是由 编译器完成的，因此编译器会为类的每个构造函数添加代码来完成对 vptr 的初始 化。

4.带有一个虚基类的类

在这种情况下，编译器要将虚基类在类中的位置准备妥当，提供支持虚基类的机 制。也就是说要在所有构造函数中加入实现前述功能的的代码。没有构造函数将 合成以完成上述工作。

总结：简单来讲编译器会为构造函数做的一点事就是调用其基类或成员对象的默 认构造函数，以及初始化 vprt 以及准备虚基类的位置。

总的来说，编译器将对构造函数动这些手脚：

- 如果类虚继承自基类，编译器将在所有构造函数中插入准备虚基类位置的代 码和提供支持虚基类机制的代码。
- 如果类声明有虚函数，那么编译器将为之生成虚函数表以存储虚函数地址， 并将虚函数指针（vptr）的初始化代码插入到类的所有构造函数中。
- 如果类的父类有默认构造函数，编译将会对所有的默认构造函数插入调用其 父类必要的默认构造函数。必要是指设计者没有显示初始化其父类，调用顺 序，依照其继承时声明顺序。
- 如果类包含带有默认构造函数的对象成员，那么编译器将会为所有的构造函 数插入对这些对象成员的默认构造函数进行必要的调用代码，所谓必要是指 类设计者设计的构造函数没有对对象成员进行显式初始化。成员对象默认构 造函数的调用顺序，依照其声明顺序。
- 若类没有定义任何构造函数，编译器会为其合成默认构造函数，再执行上述 四点。

【2011/12/21 补】需要说明的是，从概念来上来讲，每一个没有定义构造函数 的类都会由编译器来合成一个默认构造函数，以使得可以定义一个该类的对象， 但是默认构造函数是否真的会被合成，将视是否有需要而定。C++ standard 将 合成的默认构造函数分为 trivial 和 notrivial 两种，前文所述的四种情况对 应于 notrivial 默认构造函数，其它情况都属于 trivial。对于一个 trivial 默认 构造函数，编译器的态度是，既然它全无用处，干脆就不合成它。在这儿要厘清 的是概念与实现的差别，概念上追求缜密完善，在实现上则追求效率，可以不要 的东西就不要。