深度探索C 对象模型(2) PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/133/2021_2022__E6_B7_B1_E5_BA_A6_E6_8E_A2_E7_c97_133880.htm 史列因：我刚看了你写的“ 深度探索C 对象模型(1) ”，感觉很不错。不过我有一个建议：你说“ 谁知第一章便如此的难以消化，已经反复读了3遍，还是有些夹生 ”是很自然的。第一章是一个总览，如果你能全看懂，后面的就没什么看的必要了。第一章的内容后面都有详细介绍，开始只要有个大概印象就可以了。这本书中很多内容都是前后重复的。我建议你先不管看懂看不懂，只管向后看，之后再从头看几遍，那样效果好得多。 我想史列因说的应该是一种非常好的阅读方式，类似《深度探索C 对象模型》这样的技术书籍，需要的是理解，和学习英文不同，不能靠死记硬背，如果出现理解不了的情况，那你不妨将书放下，打一盘红警（俺骄傲的说，我是高手）。或者跳过去也是一个不错的方法。好了，我们还是继续研究C的对象模型吧。 简单的对象模型 看书上的例子（注释是表示solt的索引） Class Point { public: Point(float xval). //1 virtual ~Point(). //2 float x() const. //3 static int PointCount(). //4 protected: virtual ostreamamp.os) const. //5 float _x. //6 static int _point_count. //7 } 每一个Object是一系列的Slots,每一个Slots指向一个members。 表格驱动对象模型 当构造对象时便会有一个类似指针数组的东西存放着类数据成员在内存中位置的指针，还有指向成员函数的指针。为了对一个类产生的所有对象实体有一个标准的表达，所以对象模型采用了表格，把所有的数据成员放在数据成员表中，把所有的成员函数的地址

放在了成员函数表中，而类对象本身有指向这两个表的指针。 为了便于理解，雷神来举个不恰当的例子说明一下，注意是不很恰当的例子 我们把写字楼看成一个类，写字楼中的人看成是类的数据成员，而每一个租用写字楼的公司看成类的成员函数。我们来看一个实体，我们叫它雷神大厦。雷神大厦的物业管理部门需要登记每个出入写字楼的人，以便发通行证，并且需要登记每个公司的房间号，并制作了一个牌子在大厅的墙上。实际上这便是类的对象构造过程。你可以通过大厅墙上的公司列表找到任何一家在雷神大厦租房的公司，也可以通过物业提供的花名册找到任何一个出入雷神大厦的人。 真是一个考验大家想象力的例子。（如果你有更好例子的别忘了和雷神交流一下）。 C 的对象模型 C 对象模型是从简单对象模型派生得来，并对内存空间和存取时间做了优化。它引入了虚函数表（virtual table）的方案。每个类产生一堆指向虚函数的指针，放在表格中。每个类的对象被添加了一个指针（vptr），指向相关的虚函数表（virtual table）。而这个指针是由每一个类的constructor、destructor和copy assignment运算符自动完成。 我们还用上面的雷神大厦举例，物业管理为了提高效率，对长期稳定的公司和人员不再登记，指对不稳定或不能确定的公司进行登记，以便于管理。 再次考验大家的想象力。 得出结论，C 对象模型和双表格对象模型相比，提高了空间和存储时间的效率，却失去了弹性。试想一下，没有整个雷神大厦人员和公司的名录，如果他们发生变化，则需要物业管理部门做很多工作。重新确定长期稳定的公司和人员是那些。对应应用程序则需要重新编译。（这次更离谱，但为了保持连贯，大家请进行理解性的思考

，不要局限字面的意思）这篇笔记是分成多次一点点写的，甚至每天抽出一个小时都不能保证（没办法最近实在忙），因此可能会有不连贯，如果你读起来很不爽认为雷神的思维短路了，那属于正常。不过雷神还是再上传之前努力的将思路进行了一下整理。希望能把这些支言片语串起来。 最后说一句阅读《深入C 对象模型》一书感觉没有什么可以被成为重点的东西，感觉每一个字都不应该放过，全是重点。经过反复阅读，雷神好象有些开窍，继续努力呀，我和大家都是。