OOinC(1)：C语言中的类模拟和多态,继承 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/225/2021_2022_OOinC_1__EF_BC_c97_225713.htm 计算机等级考试训练软件《百宝箱》 在面向对象的语言里面，出现了类的概念。这是编程思想的一种进化。所谓类：是对特定数据的特定操作的集合体。所以说类包含了两个范畴：数据和操作。而C语言中的struct仅仅是数据的集合。 1.实例：下面先从一个小例子看起#ifndef C_Class #define C_Class struct #endif C_Class A { C_Class A *A_this；void （*Foo）（C_Class A *A_this）；int a；int b；}；C_Class B{ //B继承了A C_Class B *B_this； //顺序很重要void（*Foo）（C_Class B *Bthis）； //虚函数int a；int b；int c；}；void B_F2（C_Class B *Bthis）{ printf（"It is B_Fun\n"）；} void A_Foo（C_Class A *Athis）{ printf（"It is A.a=%d\n"，Athis->a）；//或者这里// exit（1）；// printf（"纯虚 不允许执行\n"）；//或者这里} void B_Foo（C_Class B *Bthis）{ printf（"It is B.c=%d\n"，Bthis->c）；} void A_Creat（struct A* p）{ p->Foo=A_Foo；p->a=1；p->b=2；p->A_this=p；} void B_Creat（struct B* p）{ p->Foo=B_Foo；p->a=11；p->b=12；p->c=13；p->B_this=p；} int main（int argc，char* argv[]）{ C_Class A *ma，a；C_Class B *mb，b；A_Creat（amp.b）；mb=amp.a；ma=（C_Class A*）mb；//引入多态指针printf（"%d\n"，ma->a）；//可惜的就是 函数 变量没有private ma->Foo（ma）；//多态a.Foo（amp.b）；//成员函数，因为效率问题不使用函数指针return 0；}输出结果：11 It is B.c=13 It is A.a=1 It is B_Fun 2.类模拟解说：我在网上看见过一篇文章

讲述了类似的思想（据说C＋＋编程思想上有更加详细的解说，可惜我没空看这个了，如果有知道的人说一说吧）。但是就象C 之父说的：“ C 和C是两种语言 ”。所以不要被他们在语法上的类似就混淆使用，那样有可能会导致一些不可预料的事情发生。 其实我很同意这样的观点，本文的目的也不是想用C模拟C＋＋，用一个语言去模拟另外一个语言是完全没有意义的。我的目的是想解决C语言中，整体框架结构过于分散、以及数据和函数脱节的问题。 C语言的一大问题是结构松散，虽然现在好的大型程序都基本上按照一个功能一个文件的设计方式，但是无法做到更小的颗粒化——原因就在于它的数据和函数的脱节。类和普通的函数集合的最大区别就在于这里。类可以实例化，这样相同的函数就可以对应不同的实例化类的变量。 自然语言的一个特点是概括：比如说表。可以说手表，钟表，秒表等等，这样的描述用面向对象的语言可以说是抽象（继承和多态）。但是我们更要注意到，即使对应于手表这个种类，还是有表链的长度，表盘的颜色等等细节属性，这样细微的属性如果还用抽象，就无法避免类膨胀的问题。所以说类用成员变量来描述这样的属性。这样实例并初始化不同的类，就描述了不同属性的对象。但是在C语言中，这样做是不可能的（至少语言本身不提供这样的功能）。C语言中，如果各个函数要共享一个变量，必须使用全局变量（一个文件内）。但是全局变量不能再次实例化了。所以通常的办法是定义一个数组。以往C语言在处理这样的问题的时候通常的办法就是这样，比如说socket的号，handel等等其实都是数组的下标。（不同的连接对应不同的号，不同的窗口对应不同的handel，其实这和不同的类有不

同的成员变量是一个意思）个人认为：两种形式（数组和模拟类）并无本质的区别（如果不考虑虚函数的应用的话），它们的唯一区别是：数组的办法将空间申请放在了“ 模块 ”内，而类模拟的办法将空间申请留给了外部，可以说就这一点上，类模拟更加灵活。 3.其他的话：我的上述思想还是很不成熟的，我的目的是想让C语言编程者能够享受面向对象编程的更多乐趣。我们仅仅面对的是浩瀚的“ 黑箱 ”，我们的工作是堆砌代码，而且如果要更改代码功能的时候，仅仅换一个黑箱就可以了。 而更大的目的是促使这样的黑箱的产生。或许有一天，一种效率很好，结构很好的语言将会出现。那个时候编程是不是就会象说话一样容易了呢？ 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com