《C++程序设计教程》

图书基本信息

书名：《C++程序设计教程》

13位ISBN编号：9787302114642

10位ISBN编号：7302114641

出版时间：2005-9

出版社：清华大学出版社

作者：钱能

页数：551

《C++程序设计教程》

内容概要

《C++程序设计教程》是《C++程序设计教程》的第二版。然而从指导思想、内容结构、写作特点等方面，都以全新的面貌呈现于读者。全书全部重新执笔，代码全部重写，涵盖了基本C++编程方法的全部技术特征。作者在长期的教学、科研实践以及ACM大学生程序设计竞赛培训工作中，总结出了许多难能可贵的教学经验，能使读者快捷而准确地找到编程技术要领，洞穿C++内部实现要害，直击抽象编程本质。与《C++程序设计教程》配套，《C++课程设计指导》、《C++程序设计习题及解答》、《C++程序设计教程详解》和《C++程序设计教程精粹》也将陆续面世。除此之外，还配有C++程序设计教程课件和源代码供读者下载。

《C++程序设计教程》

作者简介

钱能，1984年毕业于上海复旦大学计算机软件专业；1988年获电子工业部优秀科技青年称号；两次获得浙江省级优秀教学成果奖二等奖。

1999年在清华大学出版社出版“ C++程序设计系列教材 ”的第一批，共三种：《C++程序设计教程》、《C++程序设计实验指导》及《C++程序设计

《C++程序设计教程》

书籍目录

《C++程序设计教程》

编辑推荐

《普通高等教育精品教材·C++程序设计系列教材：C++程序设计教程（第2版）》荣获教育部全国高校优秀教材奖！以C++标准为蓝本，从过程化编程的基本描述，到对象化编程的方法展开，乃至高级编程的实质揭示，形成一条自然流畅的主线，通俗易懂，形象风趣。本书在内容结构上自成体系，并以独特的描述手法，辐射到计算机专业其他诸课程，体系严谨，结构独特。　　本书特色：　　1、第二版全方位改版，代码全部以标准C++重写，风格独特，极具模仿价值；文字诙谐生动，通俗易懂。　　2、自成体系，结构独特，整体关联，辐射计算机各门课程。　　3、引领读者由欣赏书中的初级精彩到享受国外经典名著的内在精彩。　　4、观点鲜明，客观褒贬C++，对术语的见解独到。　　5、C++内部特性和抽象编程并重，强化编程实践，以实际编程能力衡量计算机水平。　　第一部分为程序设计基础，分四章，包括概述，基本编程语句，数据类型和计算表达。其中基本编程语句和数据类型为重点，它们一个为算法描述的基础，一个为数据结构和抽象数据类型描述的基础。　第二部分为过程化程序设计，分三章，包括函数机制，性能和程序结构。其中函数机制和程序结构是重点，前者描述过程，后者描述过程组织。　　第三部分为分面向对象编程基础，实际上就是基于对象的编程方法，分四章，包括类，对象生灭，继承和基于对象编程。前三章对类机制作了全面描述，后一章是基于对象方法的一个归纳和实例。　　第四部分为高级编程，分四章，包括多态，抽象类，模板，异常。多态和抽象类是面向对象编程的核心内容，模板论述了泛型编程，异常则强化了面向对象编程中的可靠性和容错性。

《C++程序设计教程》

精彩短评

1、相当烂。
2、都是这该死的家伙，害得我C++看了好久都入不了门，又拖沓又无章法，真要入门推荐谭浩强的C++程序设计。
3、也是不错的C++入门书籍,话说初学者真不要啃C++ primer这样的大部头为好...
4、还不错，学习程序设计的好教程
5、学c++都可以用，书本的都差不多，这本讲的也还可以，但重要的是实际编程练习
6、这本书适合学完c语言的人学习，还不错。不过麻烦卖家把上面的土擦一下，太厚了！！！
7、力荐
8、东西是正品，很经典的教材！
9、俺们上课用的教科书，这本书真的不怎么滴，十分不推荐作为入门书籍，写的相当的混乱。
10、我太喜欢这个作者了，作者好萌！
一些例子真的很萌

其实上课也是用这本书，认真看下来之后对面对对象的编程有了很好的理解
11、大学的C++教材。。。感觉质量一般，不过也可能当年老师实在讲的沉闷。。。
12、适合初学者 不错的基础教程
13、略看了一遍, C++ 名不虚传
14、书很厚实，写的面面俱到，适合刚入门的学者。
15、C++程序设计教程（第二版
16、刚学C++的时候读过点，当时觉得比C++ primer简单点，当然，不推荐
17、我以前学过C++，前两天为公司同事讲C++，基本按这本书讲的，循序渐近，能把复杂的东西讲得比较易懂。
18、本书是《C++程序设计教程》的第二版。然而从指导思想、内容结构、写作特点等方面，都以全新的面貌呈现于读者。全书全部重新执笔，代码全部重写，涵盖了基本C++编程方法的全部技术特征。本书以C++标准为蓝本，从过程化编程的基本描述，到对象化编程的方法展开，乃至高级编程的实质揭示，形成一条自然流畅的主线，通俗易懂，形象风趣。本书在内容结构上自成体系，并以独特的描述手法，辐射到计算机专业其他诸课程，体系严谨，结构独特。作者在长期的教学、科研实践以及ACM大学生程序设计竞赛培训工作中，总结出了许多难能可贵的教学经验，能使读者快捷而准确地找到编程技术要领，洞穿C++内部实现要害，直击抽象编程本质。与本书配套，《C++课程设计指导》、《C++程序设计习题及解答》、《C++程序设计教程详解》和《C++程序设计教程精粹》也将陆续面世。除此之外，还配有C++程序设计教程课件和源代码供读者下载。本书适用于大学计算机程序设计教学，也适合于立志自学成才的读者，帮助他们从零开始走向高级程序员。本书也旨在引导读者从欣赏C++入门的初级精彩到享受C++经典名作的内在精彩，因而，也是一本软件工作者不可多得的案头参考书
19、适合有一点编程基础的人入门cpp使用。
20、这是我们学校学C++的教材，应该挺不错的。我们宿舍直接团购了8本，算**当的满100减25的活动，每本只要25，比书店便宜多了，希望当当多搞活动以后就不去书店买书了XD。第一感觉是很厚，需要一定的基础来学习。
21、书很快就收到了，很好。浏览了一下，感觉内容挺丰富的，讲解也很仔细，挺适合从基础开始学的，希望自己能从中收获更多。
22、前面还不错,后面越来越罗嗦越来越罗嗦了.....神马时候才可以啃完primer啊.....
23、很不错,语言很通俗.
24、大学的教材。。
25、这本书是我的老师介绍给我的，很适合初学者使用，认真学习完后，编程不成问题，很受用。
26、十一五规划教材，里面讲解的和详细和透彻，很适合初学者，建议学C++一定要看这书。基础打好了再去看什么C++primer什么的。
27、c++拿了A+

28、很厚实的一本书，内容也很详细，结构独特，自成体系。
29、终于看完了，钱能的这本书总的来说还可以吧，感觉他更强调标准、性能和安全，强调利用STL库。感觉他写的好像比老谭的要深入一些（我看的是老谭比较薄的那本书）。但是钱能的书中的列子举得不好太长太罗嗦让人很厌烦。异常的那一章讲的不好，根本就没有对try、throw、catch等语法等进行介绍，最后还是看了老谭的书才明白过来。把钱能和老谭的结合起来看还不错……
30、收到货。书质量不错，送货也及时。当当要求评论不要让针对交易、配送等服务过程，不合理吧。
31、C++ Primer我没看过，只看过这本书我会出去乱说么。。。
32、比起其它某某语言教程，这本书要好许多了
33、很差
34、没什么存在感的教材，除了几个印刷错误，还可以的。
35、学C++时的教材，错很多，而且感觉不怎么样。。。
36、C++入门佳作
37、我用的教材~配合primer一起看的~　膜拜钱能老师。。。
38、不适合初学者。有基础的看看不错，书中很多实例还是很不错的，值得仔细研究，总的来说，是一本不错的书
39、此书不错，值得一看，但是需要有一定经验，不适合初学者，初学者还是看谭浩强的C++好点！
40、经典国内C＋＋教程，语言有文学色彩，让你在不经意间掌握C＋＋灵魂。从入门到精通，一门就够了
41、当年的教材
42、很可爱
43、我C++的入门书籍。之后又读了许多C++方面的书籍，虽然和国外的经典教材有所差距，但是在国内感觉已经很好了。
44、作为一本入门教程，还算可以，适乎也是被推荐的一本书作为C++入门书。
45、这本书是我们的教材，还不错
46、作为入门值得推荐!之前是先借了一本看，觉得内容各方面都介绍的比较全！思路比较清晰！书的纸张差得很！可以直接看到反面的字~！我在图书馆借得书比买的要厚1/5.
47、发货物流都很快，适合学习c++的人，简洁易懂
48、【入门专用】
49、的确是本高大全的入门教程 每次都是用这本书捡起来C++
50、国内少数标记“著”的语言书，原理讲的也很好
51、因为疏忽把手机写错了,害你们白跑一次,表示抱歉,谢谢你们在第二次还能及时迅速的把书送到
52、通俗易懂，形象风趣，在内容结构上自成体系，并以独特的描述手法，辐射到计算机专业其他诸课程，体系严谨，结构独特
53、如果学C++一开始看不动那2个大部头，可以先找这本来看下，厚度、内容上印象中都算合适，当然不是说里面没有错误，只是简单有个基础好看其他的书。
54、学习编程的人都推荐这本书买来一看确实不错
55、入门。。
56、刚拿到手时纸张感觉还不错，开始读中，买的时候就是想从这本书入门的！
57、有点基础的入门看这个挺好。
58、C++程序设计教程（第二版）　　C++学习的入门书
59、书的质量还是不错的，很厚实的一本书，到手时吓我一跳。
60、适合入门，为什么当时大学的C++教材不用钱能这版呢？
61、经典教材
62、书的内容就不必在这作评论了，是老师推荐的好书。当当送书的速度很快啊！书本质量也很好。总之，我很满意，因此有机会我就会在这买我需要的东西。
63、编程的好书
64、基础的实用。自己会思考，就是需要本书入门
65、钱能的书，很棒的说。。

66、这本书特别适合初学者，讲解透彻！
67、和谭浩强那本C++一起带着看的，感觉。。。
68、C++程序设计教程通俗易懂，物流很给力
69、觉得特别好，买过的计算机书也挺多的，但这本书给我的印象特别好，很多以前没弄明白的东西现在都明白了。甚至不觉得书里面有什么不足的地方:-)
70、之前看过的C++分量是这本的1/3 不到，看完基本上没学到什么，这本虽读起来比较花时间，但是实例充足，很有所获
71、很好的C++入门教材，值得一买！
72、这本书是我们的教材，也许在国内的教材中还算不错的了，但是跟国外的书比还是弱爆了，很多问题都没讲清楚。最近在看《C++ Primer》，这本书非常不错。
73、书中好几处都有错误
74、教科书就是教科书

75、国内c++教程的经典
76、C++程序设计教程可以的一本书哦将语法非常好的
77、额
78、教材
79、很好的一本书，我是初学者，这本书对于我入门很有用
80、大学教科书，汗，不堪回首. 连copy constructor是啥都不知道，竟然还拿到了这门课的奖学金. 好扯蛋.....
81、思路还是比较清晰的
82、不错的C++教程，比谭浩强的C++教程好。
83、入门级的，边看边回忆大学的学的东西。
84、入门级吧，C++入门就是用的这本书，并且还是作者教的我们。不过感觉内容偏少，很多东西都没讲。真的要用C++的时候感觉在这本书里学到的不多，建议进阶看C++primer！！！
85、2011年3月的读书任务，读的是第一版。
86、如果嫌c++primer太大部头了，那就先看这本吧，作者写的很用心看了这本后，effect c++中的知识点也差不多都可以懂了
87、从文字就能看出，这是作者亲身体验写成的。这本书不像其他书籍一样机械介绍知识点，而把更细节地上机实践的经验和经常出现的错误，以及理解上的偏差等写出来，使得书整体看来更易读和人性化，更实用。这也看出作者是一个细致，亲切，与人为善的人。此书编写，很有条理，并把面向过程和面向对象分别独立介绍，有一个递进的过程，使得读者更能看到两者的区别。可以说，作者倒数了C++的基本精髓。总体来说，这本书不错。
88、大学教材，作为入门书算是比较合适的
89、这本书，大概看看就行。有些该讲的只是泛泛的提了一下。有时间还是钻研一下primer的好。
90、入门书呗，没啥说的
91、C++程序方面的教程太多了，这本也是遍写的比较好的，很实用。
92、我那时刚学完C，接着买了这本书，觉得讲的很细，真的很生动，但书上的程序有的有问题，有的太啰嗦了，到了后面五六十页就比较难了。对于新手很适合，暴力推荐！！！
93、刚开始看"擦这么乱"，后来其实还编的挺有趣的。C++老师最可爱了！
94、　　你们真的不觉得这书废话太多么？想仿照欧美教材的写作风格，举例说明，却又没学到精髓，让人越读越烦。

95、一如既往的经典。现在市面上的烂书太多了，看书还是看经典吧
96、看了三篇，還是不懂……
97、比较全 就是写的有点乱感觉 不太系统
98、买书到现在已经好几个月了，看完这本书了，真的不错，建议一读。
99、这本书应该是国内介绍C++最好的书。
100、C++教材经典

101、全面也好读，毕业那会找工作就靠这书了。

《C++程序设计教程》

精彩书评

1、我只是把它当作资料来查阅了一下，没想到书里面有很多常识性的错误，作者妄加猜测，而不经过实践检测。我只看了string部分和关于const指针的解释两部，就已经发现了很多问题。==========我在此先举一例技术性错误，钱能真的是误人子弟！第97页，原文如下----------int* const cp = &b; //const修饰指针*cp--指针常量int const* dp = &b; //等价于上一句--指针常量----------显然dp和cp完全不是一回事。dp是非常量指针，所指向的是整型常量。另外，作者还在文中咬文嚼字地说什么指针常量、常量指针，起这么短的名字，自以为很聪明，不如写a constant pointer to int和a pointer to constant int来区分你所说的整型“指针常量”和整型“常量指针”。==========另外，还有非技术性错误。比如，第82页，方框里的输出结果应该是：E:\ch03>f03061443231383524各位读者适可而止吧，去读The C Programming Language，C++ Plus。嫌太厚了，就读essential C++。
2、入门级吧，C++入门就是用的这本书，并且还是作者教的我们。不过感觉内容偏少，很多东西都没讲。另外我也看了谭浩强的，感觉也差不多。真的要用C++的时候感觉在这本书里学到的不多，建议进阶看C++primer！！！
3、刚买了书，挺兴奋的。我同学看到了，说他也喜欢这书，不过他看了一点觉得有地方不太明白，就找了个猎豹网校，那里有视频课程，有老师教的。他说那样学比自己看书直观多了。我要比较一下再定了。
4、你们真的不觉得这书废话太多么？想仿照欧美教材的写作风格，举例说明，却又没学到精髓，让人越读越烦。//这短吗？
5、涵盖基本上所有语法知识点和编程细节，也算国内难得的原创书了，对标准库介绍偏少，其他方面都比较OK。
6、比第一版有趣多了，感觉应该能很快读完吧。。。。。。。我的评论太短了？？？？我的评论太短了？？？？我的评论太短了？？？？我的评论太短了？？？？我的评论太短了？？？？我的评论太短了？？？？我的评论太短了？？？？我的评论太短了？？？？

《C++程序设计教程》

章节试读

1、《C++程序设计教程》的笔记-第139页

第五章，指针参数。
直接包括进传值参数就行了，独立一节显得略显拖沓，也不利于理解。钱能这本书很多地方都有点玩弄概念的意味。

2、《C++程序设计教程》的笔记-第474页

类模板是一种模板，它通过在类定义上铺设类型参数的形式，表示具有相似操作的系列类。
模板类是指从类模板产生的类。通常通过传递给类模板以类型实参而得到模板类。

函数模板属于模板，是一种函数族。
模板函数是普通函数的一种，通常通过传递给函数模板类型实参得到模板函数。

3、《C++程序设计教程》的笔记-第417页

类型转换：
1.动态转换，//只适合有虚函数的基类。不含想要的指针则返回0
dynamic_cast<Children*>(Base);
dynamic_cast操作时专门针对有虚函数的继承结构来的。它将基类指针转换成想要的子类指针。
2.静态转换
static_cast<Class1*>(Class2);
用来转换相关类型的类。static_cast转换并不是专门针对指针的，只要是相关类型的转换，都可以操作。
3.常量转换
char*p=const_cast<char*)(const char * f);
从type类型转换到const type类型是允许的，但是从const type类型转换到type类型是不行的。为了将const type类型转换为type类型。可以引入const_type来将const_type类型转换为type类型。

注意：类型转换常常会引入很多其他错误，所以要慎用。

4、《C++程序设计教程》的笔记-第475页

模板参数可以被默认。模板参数默认的规则与函数参数默认的规则相同，从参数列表右边开始默认起。我们熟悉的向量容器其实有两个模板参数，只不过，第二个模板参数是默认参数，在通常的编程中用的是默认值。向量的声明形式为：
template<typename T,typename Allocator=allocator<T>>
class vector;
向量模板的第二个参数是内存分配器，它负责分配和释放需要操作的空间。向量容器的默认内存分配器是STL提供的，除非我们自己定义向量容器的内存分配器，一般都是默认采用内部的动态内存分配器。
假如我们自定义了一个内存分配器Alloc。则在建立向量时，可以：
class Alloc；
vector<double> a(1000);
vector<double,Alloc> b(1000);

5、《C++程序设计教程》的笔记-第440页

抽象类做界面的设计方案，既可以达到彻底分离程序模块的目的，又克服了繁琐的成员函数转换，既干净又利落。在彻底分离编程内容和职责的前提下，就可以建立完全独立的模块——动态链接库了
在上节中，动态链接库可由IDATA类的派生类即DATA类构成，只要IDATA界面不动，动态链接库还可以在背后悄悄更换，甚至连DATA类定义都可以改，而应用程序无须做任何改动面向对象的模块就是完整的类模块，它通过抽象界面与用户联系，而所有的实现却隐在连源代码都可以隐藏的类似动态链接库中

6、《C++程序设计教程》的笔记-第476页

模板实例化：
类模板的实例化即生成模板类不是一次性将类中的成员函数全部实例化的，没有被使用的函数将不会被实例化。即：类模板成员函数的实例化是被单独激活的。函数模板实例化仅发生在第一次调用该模板函数时。

可以显示实例化一个类模板让其生成对应的模板类
template List<double>;
template List<int>;
显示实例化会马上将所有的类模板的成员函数都实例化成对应的模板成员函数

7、《C++程序设计教程》的笔记-第424页

一般地，一个容器，其元素师基类对象的指针或引用，才有多态可言，若没有指向基类的操作，子类就不能行使多态。

理解：要使用多态，就必须要一个共同的基类，而要调用多态函数则需要通过基类的指针和引用来动态调用多态函数，这就出现了一个问题：必选要有基类对象指针或引用，但是常常我们并不需要这么一个多余的基类对象。解决方法是引入抽象类来解决，因为抽象类不能实例化对象。

抽象类：含有纯虚函数是抽象类的唯一标识。抽象类不能创建对象。纯虚函数在抽象类中不要实现。

8、《C++程序设计教程》的笔记-第480页

局部定做：
类模板的模板参数多于一个是很正常的。
template<typename T1,typename T2>
class A{....};模板定做不一定要全部定做，即不一定非得要将类模板定做成普通类。如果一个类模板的类型参数多于一个，而定做其中一部分类型参数为确定的类型，则称为局部定做
模板一旦局部定做，则产生的模板铸件就仍然是类模板。
template <typename T>
class A<T,T>{...};

Class Cat；
template<typename T>
class A<T,Cat>{...};

9、《C++程序设计教程》的笔记-第464页

template <typename T>
template <class T>
void swap(T&a,T&b)
{
T temp =a;
a=b;
b=temp;
}根据数据实参的类型----》匹配数据形参的类型---》确认模板实参---》推得模板形参的过程称为数据实参的演绎
以函数模板名为函数名的函数调用，以数据实参推演出模板实参，进而生成模板函数定义的过程称为函数模板的实体化或实例化
利用函数模板隐式生成模板函数：swap(A,B)
利用函数模板显式生成模板函数：swap(int)(A,B)
函数模板具有苛刻的类型匹配：
隐式调用的参数必须和模板函数的参数严格匹配。
可用显式生成模板函数的方法来解决
也可以调用类型转换的方式来改变实参类型

10、《C++程序设计教程》的笔记-第526页

异常机制通过规定异常发生的可能区域（try），以及异常扑捉（catch），来处理发生异常的善后问题。对于抛出来的异常，往往是跨越数个函数调用而被捕捉处理。捕捉是根据抛出来的对象类型与捕捉对象类型的匹配来完成的。这个匹配是完全匹配不能进行任何类型转换

异常可以申述，可以在函数中规定抛出异常的种类，以便让预料异常和未料到异常处于不同的捕捉区，让程序能够分门别类地处理突发事件

抛出得异常被捕捉后，还可以重新抛出，因而可以形成一张捕捉网。在一个相对完整的程序中，异常一般构成一个用户的类层次体系，辅之以标准异常的类层次体系，就可以基本上满足地处理好程序中的各处异常了。

构造函数是必须借助异常处理的典型示例，引用的动态转换和动态类型信息的获取也要靠异常帮忙，因为它们的一个共同点是，不能等到一个操作完整地执行后，再来判断其成功与否，操作失败的结果相当于不可思议地创建整形变量(int a)失败，对程序来说，在异常没有推出之前，这种意外只有粗暴停机。

异常给我们带来了另外一个好处就是非错误处理。尤其是递归函数中，嵌套搜索树结构中的结点时，在找到相应树结点后迅速撤离现场到原始发出搜索调用的函数处，异常处理表现得异常独特。

11、《C++程序设计教程》的笔记-第486页

模板的多态：

多态：相同的表达式在不同的场合表示不同的实体时，其捆绑的操作因实体不同而表现出不同的行为

，称为多态。

动多态：通过继承基类的虚函数来动态绑定不同的子类，当子类调用该虚函数的时候，通过滞后捆绑的方式实现动态多态。

用模板实现静多态：用函数模板将函数形参设置为待确定类型。不同对象调用函数模板时根据参数生成不同模板函数从而到达多态；

动静多态的差异：
动多态适合于类层次结构，而且基类必须为多态类，即含有虚成员函数。
静多态则适合所有的类，不需要虚函数。

表现动多态的函数总是通过指针或者引用传递参数，实参一般是该指针的类型以下（含)的实体地址；
静多态则可以是传递对象，也可以传指针或引用

表现动多态的函数只处理特定的类层次结构中的系列对象，一个类系列，配一个表现动多态的专用函数。
静多态则随着传递实参的类型变化，进行函数模板的特例化（生成模板函数）

12、《C++程序设计教程》的笔记-第457页

手柄类是专门拿来处理有多态表现的指针的，这些指针所指向的对象有一个共同点，都是通过某个创建函数来产生对象的，而且该对象的实体一般不复制，传递都是通过指针或引用。
手柄类的性质：
1.对象通过指针参数的形式创建，不另外申请内存空间创建指针所指向的对象，但是要重新开辟基数。
2.对象通过别的对象创建时挂接相同对象，对象计数加1
3对象复制时，原对象需要析构，然后挂接相同对象，对象计数加1
4析构时，计数值减1.只有在计数值减至0时，才释放指针指向的对象

```
class SonyHandle
{
   Sony *sp;          //抽象基类
   int *count;

public:
   SonyHandle(Sony *pp):sp(pp),count(new int (1){}
   SonyHandle(const SonyHandle&amp;amp; sh):sp(sh.sp),count(sh.count)
   {
      (*count)++;
   }
   Sony* operator-&gt;()
   {
      return sp;
   }
   SonyHandle&amp;amp; operator=(const SonyHandle&amp;amp; sh)
   {
      if(sh.sp==sp)
         return *this;
```

```
    (*this).~SonyHandle();

    sp=sh.sp;
    count=sh.count;
    (*count)++;
    return *this;
  }

  ~SonyHandle ()
  {
    if(--(*count)==0)
    {
      delete sp;
      delete count;
    }
  }
```

};手柄类实际上已经包含了高级编程中的智能指针和引用计数，这两项技术是使面向对象编程走向真正实用的基本技术。这种技术还真能推动面向对象的如意编程。

13、《C++程序设计教程》的笔记-第402页

基类与子类的虚函数不能只是名字上的重载，而是其声明要一模一样。不然即使用virtual申明了，也不会被滞后处理的。

重载函数如果仅仅是返回值不同的话，编译器会报错，因为编译器是根据函数参数来识别不同的重载函数的。
覆盖有一种特殊的情况下可以允许返回值不同。即返回类的指针或引用，此时类系中该虚函数都是返回本类的指针或引用的。

14、《C++程序设计教程》的笔记-第475页

模板参数主要是类型参数，但也有值参数，或称非类型模板参数。模板参数可以是既包含类型参数，又包含值参数的一个参数列表
如：位集合模板类bitset<11000> Bit

一般来说，模板值参数大多数使用描述空间大小或对象实体数量的整型值，在描述整型值的时候，要求其值为常量或字面值

15、《C++程序设计教程》的笔记-第455页

手柄类的引入：

对象指针问题：在纷繁的程序模块中，指针传递来复制去，一不小心就会重复释放，导致运行错误。同样，一不小心就会遗漏释放，导致内存泄露。
对象指针的外套：<原文开始><指针真是烫手！解决的办法是将抽象基类Sony的外面套一个手柄。做一个SonyHandle的手柄类，它里面的数据成员就只有一个Sony类的指针，成员函数有一个转换间访操作和一个构造函数。这样设计的目的：一是装了套子就可以再套子上再装饰，从外部提高类体

系的战斗力和防御力；二是让多态行为平凡化，打破非得传递指针和引用才能表现多态的偏见。/原文结束>

```
class SonyHandle{
    Sony *sp;

public:
    Sony* operator-&gt;()
    {
        return sp;
    }

    SonyHandle (Sony* pp):sp(pp) {};
};
```

16、《C++程序设计教程》的笔记-第403页

虚函数的若干限制：
1.只用类的成员函数才能声明为虚函数
2.静态成员函数不能是虚函数
3.内联函数不能是虚函数
4.构造函数不能是虚函数
5.析构函数可以是虚函数且通常声明为虚函数

17、《C++程序设计教程》的笔记-第477页

可以用模板实参来定做模板类
类模板的模板实参通过实例化，则构成了模板的实例，它是定义好了的模板类。如果不想用预定义的类模板来生成模板类，而是以该模板类名自己专门重写一个模板类，则得到模板铸件。得到模板铸件的过程，称为模板定做。模板定做时，必须以template<>开始，然后再class后跟类模板名，即定义需要定做的类。定做时，成员可以在原先类模板的基础上随心所欲地增删，因此，定做的实现可以与类模板的实现完全不同。定做的成员函数不再是模板函数而是普通函数了，类模板中的成员中的类型参数T将被定做的类型取代。模板定做机制使得所产生的模板类可以偏离基本类模板的框架，带来了产生模板实例的灵活性

18、《C++程序设计教程》的笔记-第435页

用抽象类做界面
因为面向对象编程，总是要涉及类层次结构下的多态，所以处理的对象，往往是通过指针或引用方式传递的，因此对应用编程来说，关键是在类层次结构下的各种对象能够表现出多态。所以我们把目标转向抽象类，一是因为抽象类是类层次结构的基类，基类的指针操作可以表现多态行为；二是因为抽象类可以只提供纯虚函数，不牵扯成员函数定义的细节，不提供任何私有数据，干净利落。
用抽象类来做应用程序的界面，不仅可以避免用户知道类中的细节，又避免了因为类中成员的改变需要来改变应用程序。

例如：假如我们的应用程序要使用某个类DATA 。我们可以将DATA类的公有成员函数提取出来做成纯虚函数。构成一个抽象类IDATA。 再将DATA类继承与IDATA。对于应用程序，就用这个抽象基类与之打周到；对于DATA类，从该抽象基类派生，并实现创建对象的函数CreateData

其中DATA可做成动态链接库

19、《C++程序设计教程》的笔记-第461页

C++程序是一些类型和函数，编程就是设计类型和函数，然后将它们按C++的程序结构组织起来。由于事物的相似性，设计的类型和函数有时也表现出相似甚至相同性。将这些相似的类型和函数归纳起来构成一个类族或函数族，用一种统一的方式来编程就是模板编程。由模板可以得到一系列的相似类型或相似函数，这些相似类型和相似函数涉及的数据其类型可能不同，但处理数据却具有相同的表现形态。
这些相似类型或相似函数就是将涉及的数据之类型为参数来生成的模板，其相似类型称为模板类，相似函数称为模板函数。

《C++程序设计教程》

版权说明