<<深度探索C++对象模型>>

图书基本信息

书名：<<深度探索C++对象模型>>

13位ISBN编号：9787121149528

10位ISBN编号：7121149524

出版时间：2012-1

出版时间：电子工业出版社

作者：斯坦利·B.李普曼

页数：356

译者：侯捷

<<深度探索C++对象模型>>

内容概要

作者Lippman参与设计了全世界第一套C++编译程序cfront，这本书就是一位伟大的C++编译程序设计者向你阐述他如何处理各种explicit（明确出现于C++程序代码中）和implicit（隐藏于程序代码背后）的C++语意。

本书专注于C++面向对象程序设计的底层机制，包括结构式语意、临时性对象的生成、封装、继承，以及虚拟——虚拟函数和虚拟继承。
这本书让你知道：一旦你能够了解底层实现模型，你的程序代码将获得多么大的效率。
Lippman澄清了那些关于C++额外负荷与复杂度的各种错误信息和迷思，但也指出其中某些成本和利益交换确实存在。
他阐述了各式各样的实现模型，指出它们的进化之道及其本质因素。
书中涵盖了C++对象模型的语意暗示，并指出这个模型是如何影响你的程序的。

作者Lippman参与设计了全世界第一套C++编译程序cfront，这本书就是一位伟大的C++编译程序设计者向你阐述他如何处理各种explicit（明确出现于C++程序代码中）和implicit（隐藏于程序代码背后）的C++语意。

本书专注于C++面向对象程序设计的底层机制，包括结构式语意、临时性对象的生成、封装、继承，以及虚拟——虚拟函数和虚拟继承。
这本书让你知道：一旦你能够了解底层实现模型，你的程序代码将获得多么大的效率。
Lippman澄清了那些关于C++额外负荷与复杂度的各种错误信息和迷思，但也指出其中某些成本和利益交换确实存在。
他阐述了各式各样的实现模型，指出它们的进化之道及其本质因素。
书中涵盖了C++对象模型的语意暗示，并指出这个模型是如何影响你的程序的。

对于C++底层机制感兴趣的读者，这必然是一本让你大呼过瘾的绝妙好书。

<<深度探索C++对象模型>>

作者简介

作者:(美)Lippman

<<深度探索C++对象模型>>

书籍目录

章节摘录

版权页： 插图： Member Function的实例化行为（Member Function Instantiation）对于template的支持，最困难的莫过于template function的实例化（instantiation）。
目前的编译器提供了两个策略：一个是编译时期策略，程序代码必须在program text file中备妥可用；另一个是链接时期策略，有一些meta.compilation工具可以导引编译器的实例化行为（instantiation）。
下面是编译器设计者必须回答的三个主要问题：1.编译器如何找出函数的定义？
答案之一是包含template program text file，就好像它是一个header文件一样。
Borland编译器就遵循这个策略。
另一种方法是要求一个文件命名规则，例如，我们可以要求，在Point.h文件中发现的函数声明，其template program text一定要放置于文件Point.C或Point.cpp中，依此类推。
cfront就遵循这个策略。
Edison DesignGroup编译器对这两种策略都支持。
2.编译器如何能够只实例化程序中用到的member functions？
解决办法之一就是，根本忽略这项要求，把一个已经实例化的class的所有member functions都产生出来。
Borland就是这么做的——虽然它也提供#pragmas让你压制（或实例化）特定实例。
另一种策略就是模拟链接操作，检测看看哪一个函数真正需要，然后只为它（们）产生实例。
cfront就是这么做的。
Edison DesignGroup编译器对这两种策略都支持。
3.编译器如何阻止member definitions在多个.o文件中都被实例化呢？
解决办法之一就是产生多个实例，然后从链接器中提供支持，只留下其中一个实例，其余都忽略。
另一个办法就是由使用者来导引“模拟链接阶段”的实例化策略，决定哪些实例（instances）才是所需求的。
目前，不论是编译时期还是链接时期的实例化（instantiation）策略，均存在以下弱点：当template实例被产生出来时，有时候会大量增加编译时间。
很显然，这将是template functions第一次实例化时的必要条件。
然而当那些函数被非必要地再次实例化，或是当“决定那些函数是否需要再实例化”所花的代价太大时，编译器的表现令人失望！
C++支持template的原始意图可以想见是一个由使用者导引的自动实例化机制（use—directed automatic instantiation mechanism），既不需要使用者的介入，也不需要相同文件有多次的实例化行为。
但是这已被证明是非常难以达成的任务，比任何人此刻所能想象的还要难（请参考[S7ROUP94]）。
ptlink，随着cfront 3.0版所附的原始实例化工具，提供了一个由使用者驱动的自动实例化机制（use—drivenautomatic instantiation mechanism），但它实在太复杂了，即使是久经世故的人也没法一下子了解。

<<深度探索C++对象模型>>

编辑推荐

《深度探索C++对象模型》重点探索了“面向对象程序所支持的C++对象模型”下的程序行为，对于“面向对象性质的基础实现技术”以及“各种性质背后的隐含利益交换”提供了一个清楚的认识，检验了由程序变形所带来的效率冲击，提供了丰富的程序范例、图片，以及面向对象观念和底层对象模型之间的效率测量。

<<深度探索C++对象模型>>

版权说明