

书名、作者、ISBN

购书单 电子图书 豆瓣书店 2021年度榜单 2021书影音报告 购物车

深度探索C++对象模型



作者: [美] Stanley B. Lippman
出版社: 电子工业出版社
出品方: 博文视点
原名: Inside the C++ Object Model
译者: 侯捷
出版年: 2012-1-1
页数: 320
定价: 69.00元
装帧: 平装
丛书: 博文视点·传世经典书丛
ISBN: 9787121149528

豆瓣评分

9.1 381人评价

5星	63.8%
4星	29.9%
3星	5.8%
2星	0.5%
1星	0.0%

想读 在读 读过 评价: ☆☆☆☆☆

写笔记 写书评 加入购书单 分享到

推荐

内容简介

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。

在线试读：

 得到

作者简介

Stanley B.Lippman

微软公司Visual C++ 团队的架构师。他从1984年开始在贝尔实验室与C++的设计者Bjarne Stroustrup一起从事C++的设计与开发。他还著有Inside the C++ Object Model。

目录

本立道生（侯捷 译序） III
前言（Stanley B. Lippman） XIII
第0章 导读（译者的话） XXV
第1章 关于对象（Object Lessons） 1
加上封装后的布局成本（Layout Costs for Adding Encapsulation） 5
1.1 C++对象模式（The C++ Object Model） 6
简单对象模型（A Simple Object Model） 7
表格驱动对象模型（A Table-driven Object Model） 8
C++对象模型（The C++ Object Model） 9
对象模型如何影响程序（How the Object Model Effects Programs） 13
1.2 关键词所带来的差异（A Keyword Distinction） 15
关键词的困扰 16
策略性正确的struct（The Politically Correct Struct） 19
1.3 对象的差异（An Object Distinction） 22
指针的类型（The Type of a Pointer） 28
加上多态之后（Adding Polymorphism） 29
第2章 构造函数语意学（The Semantics of Constructors） 37
2.1 Default Constructor的构造操作 39
“带有Default Constructor”的Member Class Object 41

《LTspice仿真指南》电子书下载

Analog Devices

由谷歌提供的广告

当前版本有售

得到 33.99元 购买电子书

京东商城 46.20元 购买纸质书

当当网 73.70元 购买纸质书 限时抢

中图网 38.00元 购买纸质书

孔网 42.88元起 购买纸质书

+ 加入购书单

这本书的其他版本 (全部5)

华中科技大学出版社 (2001) 9.1分 1799人读过 展开有售 (2)

Addison-Wesley Professional (1996) 9.1分 185人读过

中国电力出版社 (2003) 9.6分 35人读过 展开有售 (1)

基峰 (1998) 暂无评分 4人读过

在哪儿借这本书

北京市公共图书馆(1)

上海图书馆(1)

广州图书馆(1)

沈阳师范大学图书馆

绍兴图书馆

“带有Default Constructor”的Base Class 44

“带有一个Virtual Function”的Class 44

“带有一个Virtual Base Class”的Class 46

总结 47

2.2 Copy Constructor的构造操作 48

Default Memberwise Initialization 49

Bitwise Copy Semantics（位逐次拷贝） 51

不要Bitwise Copy Semantics！ 53

重新设定Virtual Table的指针 54

处理Virtual Base Class Subobject 57

2.3 程序转化语意学（Program Transformation Semantics） 60

显式的初始化操作（Explicit Initialization） 61

参数的初始化（Argument Initialization） 62

返回值的初始化（Return Value Initialization） 63

在使用者层面做优化（Optimization at the User Level） 65

在编译器层面做优化（Optimization at the Compiler Level） 66

Copy Constructor：要还是不要？ 72

摘要 74

2.4 成员们的初始化队伍（Member Initialization List） 74

第3章 Data语意学（The Semantics of Data） 83

3.1 Data Member的绑定（The Binding of a Data Member） 88

3.2 Data Member的布局（Data Member Layout） 92

3.3 Data Member的存取 94

Static Data Members 95

Nonstatic Data Members 97

3.4 “继承”与Data Member 99

只要继承不要多态（Inheritance without Polymorphism） 100

加上多态（Adding Polymorphism） 107

多重继承（Multiple Inheritance） 112

虚拟继承（Virtual Inheritance） 116

3.5 对象成员的效率（Object Member Efficiency） 124

3.6 指向Data Members的指针（Pointer to Data Members） 129

“指向Members的指针”的效率问题 134

第4章 Function语意学（The Semantics of Function） 139

4.1 Member的各种调用方式 140

Nonstatic Member Functions（非静态成员函数） 141

Virtual Member Functions（虚拟成员函数） 147

Static Member Functions（静态成员函数） 148

4.2 Virtual Member Functions（虚拟成员函数） 152

多重继承下的Virtual Functions 159

虚拟继承下的Virtual Functions 168

4.3 函数的效能 170

4.4 指向Member Function的指针（Pointer-to-Member Functions） 174

支持“指向Virtual Member Functions”的指针 176

在多重继承之下，指向Member Functions的指针 178

“指向Member Functions之指针”的效率 180

4.5 Inline Functions 182

形式参数（Formal Arguments） 185

局部变量（Local Variables） 186

第5章 构造、析构、拷贝语意学（Semantics of Construction, Destruction, and Copy） 191

纯虚函数的存在（Presence of a Pure Virtual Function） 193

虚拟规格的存在（Presence of a Virtual Specification） 194

虚拟规格中const的存在 195

重新考虑class的声明 195

5.1 “无继承”情况下的对象构造 196

抽象数据类型（Abstract Data Type） 198

为继承做准备 202

5.2 继承体系下的对象构造 206

虚拟继承（Virtual Inheritance） 210

vptr初始化语意学（The Semantics of the vptr Initialization） 213

5.3 对象复制语意学（Object Copy Semantics） 219

5.4 对象的效能（Object Efficiency） 225

5.5 析构语意学（Semantics of Destruction） 231

第6章 执行期语意学（Runtime Semantics） 237

6.1 对象的构造和析构（Object Construction and Destruction） 240

全局对象（Global Objects） 242

局部静态对象（Local Static Objects） 247

对象数组（Array of Objects） 250

Default Constructors和数组 252

以下书单推荐 · · · · · (全部)

C++: From Novice to Professional (Kimmy)

计算机科学 (月亮)

程序员典藏大系 (恒量)

评分9分以上的计算机图书 (子苓)

9分以上（软件开发相关） (博士奶爸)



豆瓣豆品 | 给“好书”一个拥抱
豆瓣豆品

广告

谁读这本书？



南瓜不说话
昨天 想读



楚凤
8月8日 想读



坤坤的里世界
8月8日 想读



WXY
8月4日 读过

> 140人在读

> 404人读过

> 1065人想读



广告

二手市场

4本二手书欲转让 (20.00 至 40.00元)

有1065人想读,手里有一本闲着? [在豆瓣转让](#)

转让给其他二手平台? [孔网上门收书](#)

订阅关于深度探索C++对象模型的评论:

[feed: rss 2.0](#)

6.2 new和delete运算符 254

针对数组的new语意 257

Placement Operator new的语意 263

6.3 临时性对象 (Temporary Objects) 267

临时性对象的迷思 (神话、传说) 275

第7章 站在对象模型的尖端 (On the Cusp of the Object Model) 279

7.1 Template 280

Template的“实例化”行为 (Template Instantiation) 281

Template的错误报告 (Error Reporting within a Template) 285

Template中的名称决议法 (Name Resolution within a Template) 289

Member Function的实例化行为 (Member Function Instantiation) 292

7.2 异常处理 (Exception Handling) 297

Exception Handling快速检阅 298

对Exception Handling的支持 303

7.3 执行期类型识别 (Runtime Type Identification, RTTI) 308

Type-Safe Downcast (保证安全的向下转换操作) 310

Type-Safe Dynamic Cast (保证安全的动态转换) 311

References并不是Pointers 313

Typeid运算符 314

7.4 效率有了, 弹性呢? 318

动态共享函数库 (Dynamic Shared Libraries) 318

共享内存 (Shared Memory) 318

..... (收起)

"深度探索C++对象模型"试读

关于对象 (Object Lessons) 在C语言中, “数据”和“处理数据的操作 (函数)”是分开来声明的, 也就是说, 语言本身并没有支持“数据和函数”之间的关联性。我们把这种程序方法称为程序性的 (procedural), 由一组“分布在各个以功能为导向的函数中”的算法所驱动, 它们处理的是共同的外部数据。举个例子, 如果我们声明一个struct Point3d, 像这样: typedef struct...

第一章-关于对象

..... (查看全部试读)

原文摘录 (全部)

在这四种情况中, 程序可以被正确编译并执行, 但是效率不彰。 (查看原文)

 邻家の躺平人 2013-07-28 22:49:57

—— 引自第75页

You must use the member initialization list in the following cases in order for your program to compile 1. When initializing a reference member 2. When initializing a const member 3. When invoking a base or member class constructor with a set of arguments In the fourth case, the program compiles and executes correctly. But it does so inefficiently. (查看原文)

 邻家の躺平人 2013-07-28 22:49:57

—— 引自第75页

> 全部原文摘录

丛书信息

博文视点·传世经典书丛 (共15册), 这套丛书还有 《Effective C++》, 《Essential C++ 中文版》, 《提高C++性能的编程技术》, 《编程匠艺》, 《代码大全 (第2版)》等。

喜欢读"深度探索C++对象模型"的人也喜欢的电子书

支持 Web、iPhone、iPad、Android 阅读器



Essential C++
中文
Stanley B. Lippman
侯捷 译
13.00元



C++标准库 (第2版)
The C++ Standard Library
A Tutorial and Reference, Second Edition
37.20元



Python源码剖析
——从源代码理解Python核心技术
17.99元

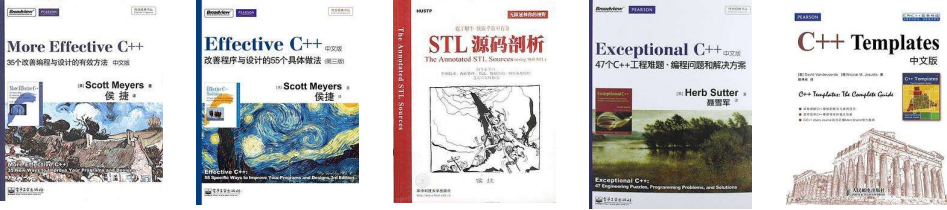


编程之美
——微软技术面试心得
19.28元



思考的乐趣
Matrix67 数字笔记
18.00元

喜欢读"深度探索C++对象模型"的人也喜欢 · · · · ·



More Effective C++ (中文版) Effective C++ STL源码剖析 Exceptional C++ (中文版) C++ Templates 中文版



程序员的自我修养 UNIX环境高级编程 (第3版) C++语言的设计与演化 深入理解计算机系统 (原书第2版) Linux/UNIX系统编程手册

短评 · · · · · (全部 151 条)

我来说两句

热门 / 最新 / 好友

略略学长 2012-10-27 07:57:02 0 有用
写作风格非常奇怪 以及小错误非常多 造成了理解上的困难

道满 2019-04-20 13:27:30 0 有用
面试必备

ZFHuang 2022-05-23 14:18:32 0 有用
按筒介读了1, 3, 4章. 学到的东西足够应付面试了. 书里讲了很多编译器底层的内容, 可惜实在晦涩难懂, 一方面是这些内容本身高深, 另一方面作者笔力也属实不够. 这本书已经出版20多年, 希望未来能有更现代且更易懂的书推出

冲不停🍷 2022-03-16 12:11:40 1 有用
重看一遍还是没什么耐心。翻译太磨叽了，加一起三十页能说完的东西要说三百页，是侯捷老师的文风。提醒一句，读书的时候分清语言标准与编译器行为，别陷进去。多没必要看的直接跳，比如多重继承虚继承。C++另一种“语言联邦”的属性体现为：语法+编译器行为+编程准则，语法是天马行空的，但是不要离开编程准则太远。时代变了，编译器的进步比程序员快得多。

usless 2021-04-11 15:58:32 0 有用
看过c++逆向，这个就有点重复

> 更多短评 151 条

深度探索C++对象模型的书评 · · · · · (全部 62 条)

我要写书评


热门 / 最新 / 好友 / 只看本版本的评论

 老鼠快跑 2011-05-11 21:21:33 华中科技大学出版社2001版

蛋疼的作者蛋疼的书，蛋疼的译者蛋疼幽默，但是如果你错过，你会在你的c++生涯中一直蛋疼下去！

如果你跟我一样是个智商在250整点上的青年，如果你也觉得看看技术书,尤其是web青年都不待见的C++相关的技术书非常流弊，来试试吧，这本书最适合你了，看了之后也不用跟着写几行代码，顶多一把一把的薅头发，反正不费电！以前看书，书里总得告儿我别这样别那样，要这样要那样，... (展开)

△ 165 ▽ 10 36回应

 为有牺牲多壮志 2008-04-05 22:58:53 华中科技大学出版社2001版

宜通读，不宜细读

理由有以下几条 一、书比较老了，是在C++ 98标准出来之前写的 二、细节错误比较多，侯捷的翻译导读里说他更正了不少，可谁知道还有多少细节错误没发现；而且，我发现他的译文有些地方也有问题，原本本来正确的，却纠正成错的 了三、除非从事编译器方面的工作，没必要死抠细... (展开)

△ 32 ▽ 3 15回应



argpunk

2012-01-30 14:33:45

华中科技大学出版社2001版

不仅仅是语言

其实完全可以用软件设计的视角来阅读这本书，在这本书中学到的不仅仅是语言，而是语言背后的东西——代码应该怎么写才会更有效率。软件设计中功能性的实现总是最低层次的，而软件背后的效率和设计的思想才更值得人去关注，linus反对用C++，因为用C完全能做到C++的效果，... (展开)

△ 13 ▽ 1 3回应



树懒

2011-02-13 16:01:57

华中科技大学出版社2001版

读后的简单总结

这本书写得很拗口，侯捷的翻译也有很多不符合习惯的用词，所以读起来颇为费力。总结一下： 1. 虚函数的实现：为每个带有虚函数的类，建立一个虚函数表，存放这个类的每个虚函数的地址。基类和派生类的虚函数表有着不同的内容。每个有虚函数的类的对象带... (展开)

△ 7 ▽ 2回应



Warren

2010-05-21 08:03:52

华中科技大学出版社2001版

所有面向对象的程序员都应该读的一本书

4年前临近毕业的时候无意中从学校图书馆发现了它，从第一页开始就被深深吸引住了，如饥似渴，一连几天都铺在上面，虽然后面的内容大多没有看懂。后来工作之后的两年中，此书起码通读了不下3遍，每次都有不同的体会-----经典的书籍就是这样，随着时光的流逝，更加沉淀出它的芬芳... (展开)

△ 5 ▽ 0回应



一无所知

2006-03-06 11:55:55

华中科技大学出版社2001版

适合对C++有较深了解的读者

第一代C++编译器开发主管所写。如果你想成为真正的C++高手，看这本书，他为你讲述了编译器在处理各种语法时在“后台”所做的事。对C++有较深入了解的读者会在读后有恍然大悟之感。侯杰翻译，质量相当不错，但内容太深，只适合对C++有较深了解的读者。 (展开)

△ 7 ▽ 1 2回应



primer

2013-09-08 09:39:28

非常有深度

很好很强大，也非常深奥，比Effective系列深太多了。只有前5章勉强看懂。这本书偏重的是C++内部实现，而effective则是讲C++的常用技术。虽然在技术上没有太大帮助，但绝对大大提高内功，会给你剖析了C++ class的实现机制。建议想深入C++的人都读一读，会有一种醍醐灌顶的作用... (展开)

△ 4 ▽ 0回应



Charlie

2006-12-27 17:41:05

华中科技大学出版社2001版

深入浅出

这本书买了好几年了。开始看的时候，感觉很深入艰涩，比较高难。这种感觉是因为初学之时注重表面上的“用法”。当渐渐发现这些“用法”非常复杂，难以掌握驾驭的时候，回头再看这本书，却又发现它讲的都是最简洁基本的内容。如果学习的过程中能不时从这本书里得到基本原理的启... (展开)

△ 2 ▽ 0回应



颠颠De我

2021-11-20 23:12:49

好书，有不少干货

这本书我买的很早，在我还连C++基础语法都没搞明白的时候，我就买下了这本大家都在推荐的“C++进阶必看书籍”。结果就是直到毕业工作了两年后，我才翻开这本书，开始我的C++进阶之路， =。= 这本书总体来说还是不错的，一些核心的观点讲得还是比较清楚的，比如说对象的内存... (展开)

△ 3 ▾ 0回应

 LV8 2015-11-28 10:21:53

C++编译器说明书 + virtual 关键字详解！

这本书非常适合用来装逼！人活着不为了装逼，那跟咸鱼又有什么区别！整本书几乎没讲C++的任何语法，任何编程技巧，任何使用经验，说的内容就如标题所言：C++编译器说明书 + virtual 关键字详解！看这本书的时候，完全不需要写任何代码去验证。为什么？举个例子... (展开)

△ 3 ▾ 0回应

> 更多书评 62篇

读书笔记 ······ (共16篇)

按有用程度 按页码先后 最新笔记

我来写笔记

 第5章 构造、析构、拷贝语意学

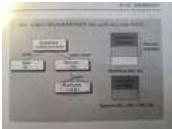
夏夜寂寞属壁虎 (像树一样自由)

foobar()函数中的L5，有个 Point object local，同样也是既没有被构造也没有被析构。当然啦，Point object local如果没有先经过初始化，可能会成为一个潜在的程序“臭虫”——万一第一次使用它就需要其初值的话（像L7）。至于 heap object在L6的初始化操作： 6) Point * heap = new Point; 会被转为对new运算符（由library提供）的调用： Point *heap = __new(sizeof(Point)); 再一次强调一下，并没有 default constructor...

2021-01-26 15:14:49

 第58页，第59页

夏夜寂寞属壁虎 (像树一样自由)



P.58：然而如果企图以一个RedPanda object作为little_critter的初值，编译器必须判断“后续当程序员企图存取其ZooAnimal subobject时是否能够正确地执行”（这是一个理性的程序员所期望的）critter的准确解释是living creature，动物。P.59：


2021-01-17 17:03:26

 第275页

Leuckart

“临时性对象的生命规则的第二个例外是当一个临时性对象被一个reference绑定时”。这句话不严谨，准确地说，一个临时性对象只能被一个const reference绑定，非const的reference是绑定不了临时性对象的。

2020-01-30 00:31:29

 第217页

Leuckart

PVertex构造函数的扩展形式中，未初始化_next。

2020-01-27 22:08:13

> 更多读书笔记 (共16篇)

论坛 ······

在这本书的论坛里发言