嵌入式技术与应用丛书

面向对象的嵌入式软件开发

周颖颖 李洋 钱瑛 编著
林新华 主审

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

嵌入式技术与应用丛书

面向对象的嵌入式软件开发

周颖颖 李洋 钱瑛 编著
林新华 主审

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书立足编程实践，以 Linux 或者 Windows 为开发平台，从初学者的角度出发，以面向对象程序设计思想为主线，结合实际项目的开发需求，将隐藏在面向对象背后的关于 C++抽象、封装、继承、多态等机制和知识娓娓道来，用通俗易懂的语言展开讲解，不仅让读者知其然，更要让读者知其所以然，最终让这些知识再反作用于编程实践，帮助读者写出高质量的 C++代码。全书涉及面向对象的嵌入式软件开发的方方面面，具体说来，主要讨论包括面向对象语言的特点、MySQL 数据库的应用、QT 基础知识入门及项目开发过程等多个方面的话题。

本书既可作为高等院校相关专业师生的教材或教学参考书，也可供相关领域的工程技术人员查阅，对于面向对象编程的初学者，也可以使其掌握面向对象编程的特点。

图书在版编目（CIP）数据

面向对象的嵌入式软件开发/周颖颖，李洋，钱瑛编著. —北京：电子工业出版社，2018.8
（嵌入式技术与应用丛书）
ISBN 978-7-121-34743-6

Ⅰ. ①面… Ⅱ. ①周… ②李… ③钱… Ⅲ. ①软件开发－研究 Ⅳ. ①TP311.52

中国版本图书馆 CIP 数据核字（2018）第 157983 号

责任编辑：田宏峰
印　　刷：三河市君旺印务有限公司
装　　订：三河市君旺印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：24.5　字数：627 千字
版　　次：2018 年 8 月第 1 版
印　　次：2018 年 8 月第 1 次印刷
定　　价：88.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：tianhf@phei.com.cn。

前言

结合作者多年嵌入式软件开发培训的工作经验，从职业的可持续发展角度来说：要想成为一名优秀的程序员，不仅要看重“量”，还要看重“质”。

为什么这么说呢？因为每一段完整的代码都是程序员的作品，这段代码作品的质量直接代表了程序员的编程能力。就好比一位艺术家的艺术成就不是以其作品的数量而是以质量来衡量的。所以，作为一名程序员，无时无刻都要认真地对待自己编写的每一段代码。

如果一个程序员只是单纯地编写代码以完成工作量，那么对他的职业发展、技术提升并没有多少好处。开发好的应用是对程序员的基本要求之一，如果编写的代码质量不高，也就难以得到用人单位的重视。虽然是一名程序员，但绝不会是一名优秀的程序员。

好多初学者在开始学习编程语言时就疯狂地敲代码，这是我们一直提倡的，因为语言的学习过程本身是一个熟能生巧的过程。如果你不能理解代码的意义，那就不停地写，写多了，慢慢地就能理解代码的意义并能做到融会贯通。但我们后来发现大部分低级程序员普遍存在的问题是：代码结构混乱、算法效率低，只是停留在解决问题阶段，没有思考如何更好地解决问题。这样的代码在入门学习时是完全没问题的，但是企业却看不上，因为企业的产品是要给用户使用的，如果编写的代码结构混乱，会导致调试麻烦、效率低，代码算法不够高效会导致用户体验比较差，企业怎么会用这样的代码呢？所以一个要想成为企业急需的高端人才的程序员来说，虽然是从“量”开始，但在积累的过程中更要注重细节，不仅要坚持语言的学习，还应该深入理解计算机的工作原理，知道如何让计算机更加高效地工作，同时不断扩大自己的知识储备，最终才能写出高“质”的代码。

现在，在程序员间流行这样一句话，就是“大学老师教的，企业不用；企业用的，大学老师不教”。其实这句话说得并不准确，毕竟大学可让学生们掌握做开发时所需要的理论知识，这些理论知识会为以后的学习和工作奠定良好的基础。但是，现在学生普遍存在的问题是他们仅仅停留在了理论知识上，甚至连上面说的“量”都没有达到。这也是我们编写本书的目的，首先要帮助大家解决编写代码“量”的问题，然后在熟练掌握理论知识的基础上教会大家怎么能写出企业重视的高“质”的代码。“量”和“质”，一直都是我们强调的优秀程序员的必修之路。

本书从初学者的角度出发，以面向对象程序设计思想为主线，结合实际项目的开发需求，将隐藏在面向对象背后的关于C++的抽象、封装、继承、多态等机制和知识娓娓道来，用通俗易懂的语言讲解理论知识，不仅让读者知其然，更要让读者知其所以然，最终让这些理论知识再反作用于编程的实践，从而帮助读者写出高“质”的C++代码。全书涉及面向对象的方方面面，具体说来，主要包括面向对象语言的特点、MySQL 数据库的应用、Qt 基础知识入门，以及项目开发过程等多个方面的话题。

本书既可作为高等院校相关专业师生的教材或教学参考书，也可供相关领域的工程技术人

员查阅，对于面向对象编程的初学者，也可以使其掌握面向对象编程的特点。

限于作者水平，书中难免有不足之处，敬请广大读者批评指正。

作　者

2018 年 5 月

目录

第1章

面向对象概述

1.1 C++概述

1.1.1 C++的发展

1980 年，Bjarne Stroustrup 博士开始着手创建一种模拟语言，使它能够具有面向对象的程序设计特色。当时，面向对象编程还是一个比较新的理念，Stroustrup 博士并不是从头开始设计新语言的，而是在 C 语言的基础上进行创建的，这就是 C++语言。

1985 年，C++开始在慢慢流行。经过多年的发展，C++已经有了多个版本，为此，ANSI 和 ISO 的联合委员会于 1989 年着手为 C++制定标准。1994 年 2 月，该委员会发布了第一份非正式草案，1998 年正式推出了 C++的国际标准。

C++语言的标准如下。

（1）C++98 标准。C++标准第一版，于 1998 年发布，正式名称为 ISO/IEC 14882:1998。

（2）C++03 标准。C++标准第二版，于 2003 年发布，正式名称为 ISO/IEC 14882:2003。

（3）C++11 标准。C++标准第三版，于 2011 年发布，正式名称为 ISO/IEC 14882:2011。

C++11 标准对容器类的方法做了三项主要修改：首先，新增的右值引用能够给容器类提供移动语义；其次，由于新增了模板类 initilizer_list，因此新增了将 initilizer_list 作为参数的构造函数和赋值运算符；第三，新增的可变参数模板（variadic template）和函数参数包（parameter pack）可以提供就地创建（emplacement）方法。

（4）C++14 标准。C++标准第四版，于 2014 年发布，正式名称为 ISO/IEC 14882:2014。

C++14 是 C++11 的更新，主要是支持普通函数的返回类型推演、泛型 lambda、扩展的

lambda 捕获、对 constexpr 函数限制的修订，以及 constexpr 变量模板化等[22]。

1.1.2 为什么要学习 C++

在 2014 年 3 月世界编程语言的排行榜中，C++语言位列第 4 位，从这个排名中我们可以看出 C++语言的应用是非常广泛的。C++语言可以用于应用软件开发、娱乐游戏开发、多媒体音/视频处理、网络通信和智能识别等。

1. 应用软件开发

操作系统可以分为两块：内核，以及内核以外的一些应用程序。内核用于控制最底层的硬件设备，应用程序则用于完成一系列的任务。应用程序是通过调用系统提供的接口（如 Windows API）操作硬件来实现一系列功能的。

要想从事应用软件开发，除了需要掌握基本的 C++语法，还需要对 Windows 系统及其他系统提供的 API 或 SDK 有一定的了解。与之相对应的岗位主要有软件开发工程师、算法工程师、架构工程师等。

2. 游戏开发

掌握了 C++语言基本语法之后，从事游戏开发也是一个不错的选择，目前工业级的 3D 游戏引擎主要是用 C 或 C++语言编写的。

虽然一个人无法去开发一个庞大的网络游戏，但是从编写一些简单的小游戏开始，然后逐渐深入、循序渐进并最终加入大型游戏开发团队还是一个非常好的选择。与之相对应的岗位主要有游戏开发工程师、游戏引擎架构工程师等。

3. 多媒体开发

目前多媒体技术已经渗入了人们的日常生活中，音/视频已经成为人们获取信息的一个非常重要的手段。音/视频在传输过程中都是经过压缩并且按照一定规则打包过的，音/视频的编码技术从最开始的 H.261 到如今的 H.265，经历了 30 多年的发展，而且实现代码全部是由 C 或 C++语言实现的。

最新的 HEVC 编码标准就是由 C++代码实现的，如果对此感兴趣的话，在掌握 C++语法后可以去 ITU（国际电信联盟）官网下载源码。与之相对应的岗位有图像算法工程师、音/视频编码工程师、音/视频转码工程师等。

4. 人工智能

人工智能、机器学习等方向也少不了 C 或 C++语言的身影，人工智能已经走入我们的生活，随着科技的飞速进步，其前景必将更加广泛。

需要强调的是，虽然 C++语言可以应用的方向非常广泛，但是仅仅掌握 C++语法是远远不够的，在上述的应用领域，C++语言是基础，进入这些领域还需要进一步深入学习相关领域的专业知识。千里之行，始于足下！下面本书将一一介绍 C++语言的基本语法，以期能够帮助大家熟练掌握 C++语言，为今后的发展奠定良好的基础。

1.2 面向过程和面向对象

如果问 C 语言和 C++语言有什么区别，很多人都会马上回答出来，C 是面向过程的语言，

C++是面向对象的语言。下面我们就来讲解下什么是面向过程，什么是面向对象。

面向过程（Procedure Oriented）是一种以过程为中心的编程思想，也可称为面向记录编程思想。面向过程其实是最为实际的一种思考方式，或者说是一种基础的方法，它考虑的也是实际的实现。一般的面向过程是从上往下步步求精，所以在面向过程编程思想中，最重要的是模块化的思想方法。当程序规模不是很大时，程序的流程很清楚，按照模块与函数的方法可以很好地组织程序。面向过程的编程思想具有如下特点：

- 强调做（算法）；
- 大程序被分隔为许多小程序，这些小程序称为函数；
- 数据开放地从一个函数流向另一个函数，函数把数据从一种形式转换为另一种形式。

面向对象（OOP）是一种以事物为中心的编程思想，它汲取了结构化程序设计中好的思想，并将这些思想与一些新的、强大的理念相结合，从而为程序设计提供了一种全新的方法。在面向对象的程序设计中，通常会将一个问题分解为一些相互关联的子集，每个子集内部都包含了相关的数据和函数；同时，还会以某种方式将这些子集分为不同的等级。一个对象就是已定义的某个类型的变量，当定义了一个对象时，就隐含地创建了一个新的数据类型。面向对象编程具有以下优点：易维护、易复用、易扩展，由于面向对象具有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统更加灵活、更加易于维护。

有人说面向对象比面向过程强，在我看来，这好比高等数学是初等数学的延伸，高等数学照样要用到方程、代数、四则运算。

很多学校老师会告诉你，万物皆对象，面向对象能更好地模拟现实，面向对象就是拖控件，面向对象就是封装继承多态，等等，这些都不完全正确。

面向对象的动机很简单，就是为了开发更大规模的软件，开发更容易扩展和维护的软件，便于更多人协同地开发软件。事实上，这也是面向过程的动机，只是面向对象扩展了这一点，这好比自行车的出现使得我们的出行更快、更省力，而汽车的设计目标也是如此。

面向对象是对面向过程的延伸，而不是否定，所以为了理解为什么要面向对象，我们首先看看为什么要面向过程。

我们知道，与非面向过程相比，在面向过程编程中，我们将重复的代码提炼成一个个的函数，然后调用这些函数。通过这种方式，我们可以将一个大的软件分成了很多模块，每个模块又可分成很多子模块，与非面向过程相比，这种组织形式可以使管理更加有序，在开发更大规模的软件时更有效。维护代码的关键是，仅在一处修改代码就能改变软件的功能，同时修改代码不会影响到别的代码。面向过程的优势在于，函数内定义的是局部变量，而函数相当于一个黑盒，在修改这个函数时，只要保持接口不变，那么程序的其他地方就不会受到影响。而程序中相同的功能被定义为函数，修改了函数，所有对函数的调用自然也就修改了。

多人协同开发软件的困难之处在于，系统中很多代码是别人写的，要调用这些代码，就必须理解它们。面向过程的好处在于，只需要理解函数的输入/输出约定、函数的功能，而不用理解函数的实现过程和具体代码，就可以调用它，换言之，你和编写这些代码的人就能协同工作了。

随着软件的规模进一步扩大呢，我们用面向过程解决的问题又被重新提出来了。

（1）原先有很多代码，被整理成几个函数。现在函数也变得很多了，这时就需要一种更高形式的组织。有人会说，这个还不简单吗？函数套函数，功能套功能。但是问题是，这样一来，某个子功能和位于另一个大模块下的子功能，如果它们有相近之处，怎么组织呢？单

一的层次结构很难组织这种结构，将层次关系转变为网状的话，结果是代码成为一团乱麻，要么坚持它们不发生关系，功能相似的代码在项目中出现两份。

面向对象按照类来组织程序，每个类实现某个功能或者特定的概念，并且把相关的代码放在一起。类和类之间相对独立，这时软件就如同积木，更容易组织。

（2）维护代码期待更少的修改。要做到这一点，必须把容易修改的代码从不容易修改的代码中剥离出来单独定义。面向对象的继承很好地做到了这一点，派生类的本质就是将类中需要修改和扩展的东西提取出来，组成的代码集合。

（3）在团队开发中，我们定义函数的目的有两个，一个是定义函数给自己用，另一个目的是给别的程序员用。但是面向过程并没有区分这两者的不同，在实际中我们都有这种体会，一旦写了一个函数给人家用，我们就不能随意修改甚至删除它，因为我们不知道这个函数被多少人调用了，贸然删除就会导致很严重的后果。解决的办法就是，一旦修改这个函数就必须让团队所有人知道。这是不是很不利于协作？

面向对象则区分了仅仅自己调用的函数（私有的）和公开的函数。对于前者，我们可以随意修改、删除，只要保证对外的接口和原来一样就可以了。这是不是更利于协作呢？