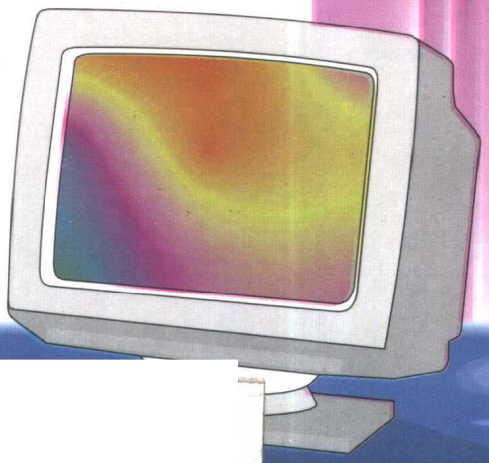
面向对象

Visual C++ 和 Windows

编程指南与实例

袁清珂 编著



西安交通大学出版社

面向对象 Visual C++ 和 Windows 编程指南与实例

袁清珂　编著

西安交通大学出版社

内 容 提 要

本书全面系统地介绍了面向对象编程语言 Visual C++。全书共分三篇：Visual C++编程环境及快速入门、Visual C++面向对象编程技术与实例以及 Windows 程序设计技术与实例；全面系统地介绍了 Visual C++编程环境及实用工具，主要包括 Visual C++工作平台、Visual C++编辑器、实用工具 AppWizard、实用工具 AppStudio、实用工具 ClassWizard；全面系统地介绍了 Visual C++面向对象编程技术，主要包括面向对象技术导论、类与对象、函数重载与算符重载、派生类与继承性、虚函数与多态性、输入和输出；全面系统地介绍了 MFC 基本类库与 Windows 程序设计技术，主要包括 MFC 基本类库、Windows 程序设计的基本知识、Windows 程序开发工具、Windows 应用程序的开发与实例。在附录部分给出了 Visual C++运行库函数参考和 MFC 基本类库类参考。

本书覆盖面广、条理清晰、论述充分、实例丰富、实用性强、易学易用，是学习 Visual C++程序设计和 Windows 程序设计的实用教科书、参考书、工具书，适合广大工程技术人员、各级程序员以及计算机爱好者阅读与参考。

(陕)新登字 007 号

面向对象 Visual C++和 Windows 编程指南与实例

袁清珂 编著

责任编辑 林全

*

西安交通大学出版社出版发行

(西安市咸宁西路 28 号 邮政编码 710049)

西安向阳印刷厂印装

各地新华书店经销

*

开本：787×1092 1/16 印张：20.375 字数：493 千字

1996 年 7 月第 1 版 1996 年 7 月第 1 次印刷

印数：1——5000

ISBN7-5605-0846-4/TP・132 定价：19.00 元

前 言

面向对象技术被誉为“研究高技术的好方法”，是近年来迅速发展的一个研究与应用领域，在程序设计与开发方面获得了广泛的应用。

面向对象程序设计是计算机程序设计的崭新模式，它提供了一种与结构化程序设计完全不同的程序设计、开发、研究的新方法，使计算机解决问题的方式更接近人类活动本身。面向对象程序设计比结构化程序设计具有无比的优越性，被软件开发者广泛接受与采用。

面向对象程序设计是程序设计与开发的最先进技术，因此必须下大力气尽快学好、用好这一技术。本书满足你的要求！

Microsoft C/C++7.0 和 Borland C++3.1 是目前 C 语言编译器中较为出色的两个，但有许多程序设计人员抱怨 Microsoft C/C++7.0 缺少 Borland C++3.1所具有的极好的集成工具，而且更感到奇怪的是，它居然缺少 Windows 宿主环境。当然，毋庸置疑，首先推出 Windows 的 Microsoft 公司有能力做得更好，Visual C++的出现便证实了这一点。

Visual C++是一种强大的开发工具，它提供了一个集成环境，用于建立调试 Visual C++程序和 Windows 应用程序，大大简化了复杂的开发工作，提高了工作效率。Visual C++分为两个版本：标准版和专业版，这两个版本都具有集成开发环境，大部分内容是相同的，只有少许区别。

本书共分三篇 15 章并有两个附录，全面系统地介绍了面向对象编程语言 Visual C++的编程环境、面向对象编程技术及 Windows 程序设计技术。书中包含大量的实例，以帮助读者加深对概念的理解、编程技巧的掌握及上机实践。

第一篇为 Visual C++编程环境及快速入门，包括面向对象技术导论、Visual C++的安装与使用、编写第一个 Visual C++程序、Visual C++程序设计技巧与实例、Visual C++的工作平台、Visual C++编辑器等 6 章。主要介绍了面向对象技术的基本思想及基本概念、Visual C++的基本知识及基本编程技巧、Visual C++的编程环境及使用方法。使读者能够尽快掌握和使用 Visual C++。

第二篇为 Visual C++面向对象编程技术与实例，包括类与对象、函数重载与算符重载、派生类与继承性、虚函数与多态性、输入和输出等 5 章。主要介绍了面向对象编程(OOP)的概念与技术、输入和输出流的概念与编程技巧。使读者结合大量实例的学习，尽快掌握 Visual C++的 OOP 技术。

第三篇为 Windows 程序设计技术与实例，包括 MFC 基本类库、Windows 程序设计的基本知识、Visual C++提供的 Windows 开发工具、Windows 应用程序的开发与实例等 4 章。主要介绍了 MFC 类层次及常用类的使用、Windows 编程的基

本知识、实用开发工具 AppWizard、AppStudio、ClassWizard，最后给出了开发实例。通过本篇的学习，使读者了解 MFC 基本类库、掌握实用开发工具的使用及 Windows编程技术。

附录 A 为 Visual C++运行库函数参考，附录 B 为 MFC 基本类库类参考。这两个附录分别列出了运行库函数和 MFC 库类，以方便读者在学习和编程过程中查阅。

本书是作者在教学与科研实践的基础上，经过进一步消化、加工、补充与完善而形成的。本书对概念及编程思想分析透彻，表达清晰，并附有大量实例，以帮助读者理解与掌握。书中所有程序都很典型，具有一定的代表性，并且经过上机调试均通过。

作者

目　录

第 1 篇　Visual C++编程环境及快速入门

第 1 章　面向对象技术导论

第 2 章　Visual C++的安装与使用

第 3 章　编写第一个 Visual C++程序

第 4 章　Visual C++程序设计技巧与实例

第 5 章 Visual C++的工作平台

第 6 章 Visual C++ 编辑器

第 2 篇　Visual C++面向对象编程技术与实例

第 7 章　类与对象

第 8 章　函数重载与算符重载

第 9 章　派生类与继承性

第 10 章　虚函数与多态性

第 11 章　输入和输出

第 3 篇 Windows 程序设计技术与实例

第 12 章 MFC 基本类库

第 13 章 Windows 程序设计的基本知识

第 14 章 Visual C++提供的 Windows 开发工具

第 15 章 Windows 应用程序的开发与实例

附录

第1章

面向对象技术导论

被称为“研究高技术的好方法”的面向对象技术正引起全世界的高度关注。80年代以来，面向对象(Object－Oriented，简称O－O)技术，从O－O认识方法论、O－O系统分析与设计、O－O编程等领域强烈地影响、推动和促进着一系列高技术的发展和多学科的综合应用。面向对象技术在程序设计与开发方面已获得了广泛的应用。

1.1 面向对象技术的产生与发展

我们对世界或一个系统的认识是一个渐进的过程，是在继承了以往的有关知识的基础上而逐渐深化的。然而，传统的用于分析、设计和实现一个系统的过程和方法大部分是“瀑布”型的，即后一步是实现前一步所提出的要求，或者是进一步发展前一步所得出的结果。因此，当接近系统实现的后期时，若要对前期的结果作修改，则是非常困难的事情。同时，也只有在系统实现的后期，才能发现前期的一些差错。当系统越大、越复杂时，这些困难也就越大。有时，前期累计的差错在后期是无法解决的。这样，整个系统就不得不重新开始。

要解决这一问题，就应该使我们分析、设计和实现一个系统的方法尽可能接近我们认识一个系统的方法。于是，就产生了面向对象技术。

面向对象技术的基本方法学认为，客观世界是由许多各种各样的对象所组成的，每种对象都有各自的内部状态和运动规律，不同对象之间的相互作用和联系就构成了各种不同的系统，构成了我们所面对的客观世界。面向对象技术就是要面对客观世界，以对象为基本单元，分析、设计和实现一个系统。对象是其核心，它具有以下基本特征：

(1) 模块性和隐蔽性。一个对象是一个可独立存在的实体。从外部看这个模块，只了解这个模块具有哪些功能，至于这个模块的内部状态，以及如何实现这些功能的细节都是不可见的，即是隐蔽在模块内部的。一个模块的内部状态是不受(或很少受到)外界影响的，同时，一个模块内部状态的改变也不会影响到其它模块的内部状态。因此，各模块之间的依赖性是很小的。所以各种模块才有可能较为独立地为各个系统所选用。

(2) 继承性和类比性。人们是通过客观世界中的各种对象进行分类与合并等来认识世界的，每个具体的对象都是在它所属的某一类对象(类)的层次结构中占据一定的位置。因此，下一层次的对象应具有上一层次对象的某些属性，这种性质称为继承性。另一方面，当人们发现一些不同的对象具有某些相同的属性时，也常常把它们归并成一个类，这种归并性质称为类比性。

(3) 动态连接性。在客观世界中，由于存在各种各样的对象，以及它们之间的相互连接和

作用，从而构成了各种不同的系统。因此，我们把对象和对象之间所具有的一种统一、方便、动态地连接传递消息的能力与机制称为动态连接性。

(4) 易维护性。任何一个对象都是把如何实现本对象功能的细节隐藏在该对象的内部。因此，无论是完善本对象的功能，还是改变功能实现的细节，都在该对象内部处理，而不传给外部，这就增强了对象和整个系统的易维护性。

面向对象技术的发展可以追溯到 60 年代，70 年代已提出了一些基本概念，到 1980 年推出了 Smalltalk－80，树立了面向对象技术的一个里程碑。在 80 年代，面向对象技术得到了很大的发展和广泛的应用，主要有以下几个方面：

●面向对象的设计 OOD(Object－Oriented Design)。系统的设计过程可以看成是把系统所要求解的问题分解为一些对象之间传递消息的过程。

●面向对象的语言 OOL(Object－Oriented Language)。在这种语言中，可以把数据和处理数据的过程结合为一个对象。对象既可以像数据一样被处理，又可以像过程一样描述处理的流程和细节。

●面向对象的数据库 OODB(Object－Oriented Data Base)。把对象作为存取和检索的单位，把传统数据库定义的数据定义语言和数据操作语言融为一体，这种概念也是构成语义数据库和知识库的基础。

●面向对象的智能程序设计 OOIP(Object－Oriented Intelligent Programming)。把知识系统中的智能实体作为对象，一个对象所具有的知识是该对象的动态属性，一个对象所具有的知识处理方法则是该对象的智能行为的体现。

1.2 面向对象的编程语言

面向对象技术的发展与面向对象的编程语言(简称为 O－OPL)是密切相关的。随着计算机应用日益普及，编程人员致力于研究和开发各种各样的高级语言，研究编程的构造、规范以及基本原理，以便更有效地编写各种应用程序，同时，也力求使编写、调试好的程序便于运行和维护。

1.2.1 编程中的问题

50 年代后期，在编写 Fortran 的大型程序时出现了变量名在不同的程序部分发生冲突的问题。为此，Algol 语言的设计者决定采用“阻挡”(Barriers)来隔开程序段中的变量名。这样就产生了 Algol 60 以“Begin…End”为标识的程序块(Block)。由于程序块内的变量名是局部的，它们的使用就不会与程序中其他块的同名变量相冲突。这是在编程语言中首次提供保护(Protection)或封装(Encapsulation)的尝试。现在，程序块结构已广泛用于 C、Pascal、Ada 等各种语言中。

1.2.2 面向对象编程语言的产生

60 年代中后期，Simula－67 语言的设计者(Dahl 和 Nygaard 在 1966 年以及 Myhrbang 在 1970 年)采用了 Algol 的程序概念，并加以推进，提出了“对象”(Object)的概念。虽然 Simula 源出于 Algol，但它主要用于仿真；Simula 的对象具有“自身独立存在”并能在仿真过程中以一定

的含义彼此通信的特点。从此，在编程语言中开始使用数据封装(Data Encapsulation)的概念。

1.2.3 面向对象编程语言的发展

70年代，随着对管理大型程序的迫切需要的增长，许多语言设计者追求实现“数据抽象”的概念。由Xerox Paloalt公司经过对Smalltalk－72、74、76连续不断的研究、改进之后，终于在1980年推出商品化的Smalltalk－80。它在系统设计中强调对象概念的统一，引入对象、对象类、方法、实例等概念和术语，采用动态联编和单继承性机制。它还建立以O－O编程语言为核心，集各种软件开发工具为一体，建立O－O计算环境，配有很强的图形功能和多窗口用户界面。因此，Smalltalk的商品化标志了面向对象的编程语言已建立了较完整的概念和理论，并推向实用。Smalltalk作为一种新的、纯粹的面向对象编程语言，同时又体现和发展了面向对象方法的许多重要的概念，对面向对象方法学的形成和发展起了重大的作用。正是通过Smalltalk－80的研制与推广应用，使人们注意到面向对象方法所具有的模块化、信息封装与隐藏、抽象性、继承性、多态性等独特之处，这些优异特性为解决大型软件管理、提高软件可靠性、可重用性、可扩充性和可维护性提供了有效的手段与途径。

到目前已有多种面向对象的编程语言，如美国Xerox公司的Smalltalk－80、美国PPI公司的Objective－C、美国Interactive Software Enginering公司的Eiffel，以及C＋＋(包括Tuber C＋＋、Borland C＋＋、Micresoft C＋＋、Visual C＋＋等)。这些面向对象编程语言经常用到下列概念和术语：

●对象(Object)：是一个由信息及有关对它进行处理的描述所组成的包。

●类(Class)：是对一个或几个相似对象的描述。

●消息(Message)：是对某种对象处理的说明。

●方法(Method)：是类似于过程的一个实体，是对当某个对象接受了某一消息后所采取的一系列操作的描述。

●实例(Instance)：是被某一个特定的类所描述的一个对象。因此，每个对象都是某个类的一个实例，而类是对各个实例的全部相似性的描述。

●程序设计环境(Programming Environment)：是一个用于设计、生产和使用软件系统的环境。

●继承(Inheritance)：是指对象间具有相同性(可重用部分)和差异变化的关系。

●方法字典(Method Dictionary)：是消息选择符和方法之间的一个相关集合。

●元类(Metaclass)：当某个类的单个实例本身就是一个类时，这个类就被称为元类。

●子类(Subclass)：是在共享其他类的描述后，再对这个描述作某些修改而构成的类。

1.3 面向对象技术的基本概念与术语

用面向对象技术来分析问题和解决问题的基本出发点，是尽可能地按照人们认识世界的方法和思维方式来进行的。客观世界是由许多具体的事物或事件、抽象的概念、规划等组成的。因此，我们将任何感兴趣或要加以研究的事、物、概念都称为对象。面向对象的方法正是以对象作为最基本的元素，它也是分析问题、解决问题的核心。由此可见，O－O方法很自然地符合人的认识规律。下面介绍面向对象技术的基本概念和术语。

1.3.1 对象 (Object)

对象是面向对象技术的核心。它具有以下基本特征。

(1) 对象是人们要进行研究或感兴趣的任何事物。对象可以是有形的实体、一个事件等。

(2) 对象实现了数据与操作的结合。对象具有状态:通常用数据来描述;对象还应当有操作,用以改变对象的状态。数据与操作都装在对象的统一体中。

(3) 对象具有唯一的识别功能。建立对象时,必须给对象赋以唯一的标识符,用来唯一而且永久地标识该对象。

(4) 对象必须属于某一个类,并成为该类的实例。

1.3.2 类 (Class)

类是指具有相同结构和操作方法,并遵守相同的约束规则的对象的集合。实质上,类定义的是一种对象类型,它描述了属于该类型的所有对象的性质。对象是在执行过程中由其所属类动态生成的,一个类可以生成多个不同的对象。同一个类的所有对象具有相同的性质,即其外部特性(外部接口)和内部实现都是相同的。一个对象的内部状态只能由其自身来修改,任何别的对象都不能改变它。

类的本质在于将数据的结构和对数据的操作封装在一起,并实现了类的外部特性与类的内部实现相隔。对于用户来说,只了解其外部特性,即该类需要输入什么信息,完成哪些功能,至于这些信息是如何处理、如何产生结果的内部实现方法对用户来说是隐藏的。这也就是常说的封装性和隐蔽性。

类具有层次性,即一个类的上层可以有超类(Super Class),这种层次结构的一个重要特点是继承性,一个类继承其上一层类的所有特性,并且这种继承具有传递性。如 Class2 继承 Class1,Class3 又继承 Class2,则 Class3 继承了 Class1 的所有特性。一个类可以有多个子类,也可以有多个超类(或父类)。有一个父类的继承称为单继承,有多个父类的继承称为多重继承。

抽象类是一种不能直接建立实例(对象)的类,抽象类将有关的类组织在一起,提供一个公共的根,其它一系列的子类从这个根派生出来。抽象类刻画了公共行为的特征,并将这些特征传给它的子类。通常一个抽象类描述了与该类有关的接口,但没有实现,具体实现留给子类来完成。

综上所述,类是对一组对象的抽象,它将该种对象所具有的共同特征集中起来,由该种对象所共享。在系统构成上,则形成了一个具有特定功能的模块和一种代码共享的手段。

1.3.3 消息与方法(Message and Method)

类通过说明它能执行的操作来决定对象的行为。消息就是用来请求对象执行某一处理、某一行为的信息,或者说对象完成某一功能所需求提供的信息。消息统一了数据流和控制流。某一对象在执行相应的处理时,如果需要,它可以通过传递消息请求其它对象完成某些处理工作,或回答某些信息。其它对象在执行所要求的处理活动时,同样可以通过传递消息向别的对象联系。因此,程序执行是靠对象间传递消息来完成的。消息实现了对象与外界、对象与其它对象之间的联系。送消息的对象称为发送者,接收消息的对象称为接收者。消息中包含发送者的要求,它告诉接收者要完成哪些处理,但并不指示接收者应该怎样完成这些处理。消息完全

由接收者解释，接收者独立决定采用什么方式完成所需处理。一个对象能为接收不同形式、不同内容的多个消息，相同形式的消息可以送往不同的对象。不同的对象对于形式相当的消息可以有不同的解释，做出不同的反映。对于传送来的消息，对象可以返回相应的回答信息，但这种返回并不是必须的。消息的形式用消息模式来表示，一个消息模式定义了一类消息，它可以对应内容不同的消息。对于同一消息模式的不同消息，同一对象所做的解释和处理都是相同的，只是处理的结果可能不同。对象的固有处理能力按消息模式分类，一个消息模式定义了对象的一种处理能力。这种处理能力是通过该模式及消息引用表现出来的。所以只要给出对象的所有消息处理模式及相应于每个消息模式的处理能力，也就定义了对象所能受理的消息，而且还定义了对象的固有处理能力，它是定义对象的唯一信息。

1.4 面向对象技术的特征

1.4.1 抽象性(Abstraction)

抽象是对复杂的现实世界的简明表示，它强调了我们所关心(感兴趣)的信息，而将不重要的信息予以忽略。抽象在系统分析、系统设计以及程序设计语言的发展中一直起着主导作用，整个软件工程的进展总是和抽象程度的提高紧密相连。在O—O技术中，抽象性是通过类与对象来实现的。

(1) 对象具有极强的抽象表达能力。对象可以用来表达一切事物。对象不仅可表达结构化的数据，而且可以表达传统的结构化方法所不能够表达的非结构化数据，包括复杂的工程实体、图形、声音、规则等等。正是由于对象这种高度的抽象表达能力，使面向对象技术具有很强的建模能力。面向对象的建模方法冲破了传统建模的约束，能够更自然、更充分地表达现实世界中存在的语义。

(2) 类实现了抽象的数据类型。在抽象的基础上，O—O技术进一步提出了类这一独特的概念，类实现了更高级的抽象。首先，将具有相同语义特性(即结构特性、操作特性、约束特性)的一组对象组成为类之后，就可进一步将这些对象的共性加以抽取，并进行统一的说明，从而省略了各个对象对共性的重复说明。其次，类将数据结构上的抽象与功能上的抽象结合起来，实现了传统方法所不具备的更高级的抽象，即类定义时，既可由系统、也可由用户来定义类所具有的数据类型，称为抽象的数据类型。尤其值得强调的是，由用户按需要灵活地定义数据类型是极其有用的机制，它使O—O技术具有很强的解决复杂问题的功能。此外，由于相同的消息名可送至不同的对象中，这些对象可以有不同的数据类型，这样就使相同消息名作用于不同的数据类型，具有结构化方法所不可比拟的高度的抽象与灵活。

1.4.2 封装性(Encapsulation)

封装性是指将数据的结构和对数据的操作结合在一个统一体中的性质，O—O技术提供了完整的封装性。

类是封装良好的模块。类定义将其“说明”(用户可见的外部接口)与“实现”(用户不可见的内部实现细节)显式地区分开。其内部实现按具体定义的作用域提供保护，可分为私有、保护与公有。如类所定义的私有变量(或成员、或数据)仅为本类所使用，其它类是不能访问它的。若

定义为公有变量，则允许所有类访问。

对象是封装的最基本的单位，类定义为本类的所有对象提供了共性。但在采用O—O方法解决实际问题时，往往要在类定义的基础上加以实例化，也就是要建立具体的该类的对象。每当创建一个新的类实例(对象)时，除了具有类定义好的共性并加以量化(取具体数值)之外，还应当定义仅由该对象所私有的特性。因此，对象的封装比类封装更具体、更细致，成为O—O方法封装的最基本的单位。

1.4.3 继承性(Inheritance)

继承性是自动地共享类、子类和对象中的方法和数据的机制，是O—O技术的重要特征之一。

继承性最重要的是体现于类层次中的共享机制，也就是说，子类能自动地继承其超(父)类的语义特性；若类层次具有多层的话，这种继承还具有传递作用，即最下层的子类可继承其上各层超类的全部语义特性。

子类可从多个超类中继承它们的语义特性，这种继承称为多重继承。采用多重继承性可将现存的几个类结合起来产生集成的新类，这个新类具有多个超类的功能与特性。

继承性具有下列优点：

(1) 继承性使所建立的软件系统具有开放性。当前软件发展的重要趋势是开放体系结构。这主要指系统的开发应当尽可能选用公用的接口，实现信息的交换与共享，模块应当尽量多地重用。O—O技术由于具有继承性，使要建立的系统并非完全从空白(从头)开始，而是尽可能利用已建立的系统或已建立的类，以它们为基础进行扩充，考虑向上与向下的接口方法。

(2) 继承性是信息组织与分类的行之有效的方法。类层次反映了现实世界中普遍存在的一般与特殊的语义联系：越靠上层(根部)表示着更为普遍或更概括的概念，而越往下层(叶部)表示更专门、更细化或更具体的概念。

(3) 继承性显著地简化了对象、类的创建工作量。通过声明新创建对象参与已存在的某个类；或通过声明新创建的类是已存在类的子类就可自动地继承类或超类的共性。这样，使创建对象或类的定义变得简单和容易，也进而增强了以O—O技术开发系统的可扩充性。

(4) 继承性进一步增强了代码的可重用率，从而提高软件系统的可靠性。代码的重用是通过继承超类的方法来实现的。

1.4.4 多态性(Polymorphism)

多态性与编程语言有较密切联系，O—O技术由于源于面向对象的编程语言，而在O—OPL中，多态性又得到充分的发挥并使O—O技术增强了灵活性和可理解性。

多态性强调O—O技术中的行为(操作)可在不同时间内保存、取用以及返回不同的类型值。也就是相同的操作(或函数、过程)可作用于多种类型的对象上并获得不同的结果。具体的多态行为主要有如下的表现：

(1) 运算符重载。指同一个运算符可作用于多种数据类型上，对O—O技术来说，不仅能作用于系统定义的数据类型上(这与结构化方法相同)，还能作用于用户定义的数据类型上(这是结构化方法所没有的性能)。

(2) 函数重载。指相同的函数名可作用于不同的对象类型上并产生不同的行为效果。例

如，函数名“Open”既可加于“数据流”对象类型上，也可加于“窗口”对象类型上而且产生完全不同的结果。

(3) 虚函数与动态联编。这是O—O技术(具体指O—OPL)所特有的动态性质。当重载的函数名前通过标识其为虚函数(例如用 Virtual 标识，置于函数名之前)时，该函数就具有较大的灵活性；既可表示子类中的同名函数，也可表示超类中的同名函数，它具有较大的灵活性，允许在运行时(而不是编译时)才按具体的数据类型和参量数来确定选用哪一个函数，这种方式就称为动态联编(Dynamic Binding)。

由于对象封装了操作，就可利用同名的操作，让各个“对象”自行解释同名操作的具体含义，而解释的多义又不会导致混乱，从而更方便于用户。这样就可更有效地进行高层次的设计(概念设计)，因为设计者无需过问数据的结构与类型，可将注意力集中在程序设计的逻辑上是否合理与可行，从而大大提高系统设计的效率与质量。

多态性具有以下优点：

(1) 多态性允许每个对象以适合自身的方式去响应共同的消息。这样，就增强了操作的透明性、可理解性和可维护性。用户不必为相同的功能的操作但作用于不同类型的对象而费心去识别，这为软件的开发与维护提供了很大的方便。

(2) 多态性增强了软件的灵活性和重用性。尤其采用虚函数与动态联编机制后，允许用户以更为明确、易懂的方式去建立通用的软件；多态性与继承层次相结合，使软件具有更广泛的重用性和可扩充性。

1.5 面向对象的程序设计

为了弄清什么是面向对象的程序设计，首先应了解和回顾一下传统的结构化程序设计方法及其设计思想，程序结构与特点，以便比较对照二者之间的差异并领会 OOP 确是 SP 的更高层次的继承、丰富、发展和突破。

1.5.1 结构化程序设计(SP)

SP(Structure Programming)是 60 年代诞生的、针对当时爆发的所谓“软件危机”的挑战、而在 70 年代—80 年代遍及全球成为所有软件开发设计领域、每个程序员都广为采用和熟知的传统的结构化程序设计方法与技术之简称。它的产生和发展形成了现代软件工程学的基础。SP 的总的设计思路是：自顶向下，层次化；逐步求精，精细化。

其程序结构是按功能划分基本模块为树型结构，使模块间的关系尽可能简单、独立，并从而可单独验证模块的正确性。每一模块均由顺序、选择和循环三种基本结构组合而成。总而言之，此即所谓“模块化”。因此，SP 的程序通过使用局部变量与子程序，使其具有如下的基本特点：

- 按层次组织模块；
- 每一模块只有一个入口，一个出口；
- 代码和数据分离(程序＝数据结构＋算法)。

SP 的优点可以概括为：子程序对程序其它部分没有或尽可能少有副作用，从而为共享代码奠定基础。由于 SP 方法的模块化分解与功能抽象、自顶向下、分而治之的手段，从而能有效