Borland C++ 3.0 系列丛书之三

技术参考手册

王小华 主编
阿 毛 罗 静
王 力 李 燕

陕西电子杂志社

Borland C++ 3.0

基本教程

陕西电子杂志社

前 言

时代在发展、知识在更新，随着计算机的普及与广泛应用，推动着现代科学技术迅猛发展。我们知道，软件是计算机的核心，可以说，离开了软件的支持，计算机就是一个毫无意义的空壳，就象人没有头脑和灵魂一样。软件的研究从发展走向成熟，在80年代末期，一种新的程序设计方法逐步形成，这就是当今最为流行的面向对象的程序设计方法，它取代了传统的结构化程序设计方法，使设计对象更自然、更直接、更形象地反映真实的问题，从而使程序设计更接近于自然语言的描述。

Borland公司推出的C++（3.0）语言，以其支持多重覆盖窗口，支持自动管理覆盖、支持内部汇编、支持鼠标输入；集编辑、编译、运行、调试于一体，并以其良好的性能而深得广大用户喜爱。

为了尽快跟上时代步伐，赶上软件技术新潮流，我们组织编写了这套丛书，这套丛书共包括三个部分：

① 库函数使用手册

② 使用入门

③ 技术参考手册

这套丛书是在吸收了国内外各类有关C++语言书籍的基础上，择其精华而编选出来的，根据读者口味的不同，采取由浅入深、叙述与例子相结合的方法，并为求使读者从书中学到真正的东西。

《库函数使用手册》中介绍了库函数分类、主函数main( )及运行库各函数的注释与使用，并结合实例给予分析，通俗易懂。

《使用入门》讲述了C++软件的安装、集成环境与命令行环境的使用，C++语言新增的功能与函数介绍。

《技术参考手册》通过以面向对象的程序设计方法为基础，重点介绍了C++语言的函数、指针、类、对象、继承性、流库等，结合以丰富的例子程序。

这套丛书经过精心选编，编者站在作者的角度反复阅读原稿，并不断修改以适应各类不同层次的读者。我们的宗旨是：力求向读者奉献最新最好的资料。

当然，在这套丛书的编选过程中，由于时间仓促与水平的限制，虽然我们力求全面与准确，但存在的局限性与错误也在所难免，这一点，愿广大读者能给予理解，编者愿听到读者善意的批评。编者与读者的心始终是相通的。

在这套丛书的组稿过程中，始终得到了陕西电子杂志社总编张忠智的支持和关怀，陕西电子杂志社的其他同志也给予了有力的协助，在些一并表示谢意。

王小华

1993年9月于西安

目 录

目 录

第五章 函数

第六章 数据类型

第七章 对象类

第一章 面向对象的程序设计方法

§1.1 起源与发展

近年来，一种新颖、独特的程序设计方法正引起全世界越来越强烈的关注，它被誉为"研究高技术的好方法"，这就是面向对象的方法(Object-Oriented Paradigm，简称为O-O方法)．许多专家与学者预言：70年代的结构方法对计算机应用与发展产生了巨大影响．80年代以后，O-O方法将从O-O认识方法论，O-O系统分析与设计方法，O-O编程风格等领域推动一系列高新技术的发展和多学科的综合．

面向对象起源于面向对象的编程语言（O-O PL），随着计算机的普及，对于开发和研究各种高级语言，研究编程的构造，规范以及基本原理，更有效地利用各种应用程序，便于调试与维护等都有极大的帮助．

在以往的编程中，譬如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情况，Algol语言的设计者决定采用"阻挡"的办法来隔绝程序段中的变量名．于是在这种思想的指导下产生了Algol 60以"Begin…End"为标识的程序块．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它块的同名变量发生冲突，这样就起到了保护的作用．这就是模块化结构，目前这种结构也广泛应用于C、Pascal、Ada等多种高级语言中．

1.1.1 面向对象语言的产生

大约在60年代中后期，Simula-67语言的设计者（Dahl和Nygaard）采用了Algol程序块概念，并加以改进和提高，提出了"对象"（Object）的概念．Simula的对象具有"自身独立存在"并能在仿真过程中以一定含义彼此通讯的特点（虽然Simula源于Algol，但它主要用于仿真）．这便是在编程中开始使用"数据封装"（data encapsulation）的开始．但是，由于Simula编译器价格昂贵，当时没能在市场上流传开来．

1.1.2 面向对象编程语言的发展

随着对管理大型程序的迫切需要，大约在70年代，许多语言设计者追求实现"数据"抽象的概念．为了应合这个时期的需要，由Xerox Paloalt公司经过对Smalltalk 72,74,76连续不断地研究，改进之后，于1980年推出了商品化的Smalltalk-80．这个语言的特点是在系统设计中强调对象概念的统一，于是引入了对象，对象类，方法，实例等概念和术语，并采用动态联编和继承性机制．它还建立了以O-O编程语言为核心，集各种软件开发工具于一体，建立O-O计算环境，配有很强的图形功能及多窗口用户界面，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，并有较高

的应用价值。

通过 Smalltalk-80 的研制和推广应用，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、多形性等独特之处，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和方法。

近十年来，由于一系列高技术项目研究，如智能计算机、计算机集成制造系统(CIMS)、计算机辅助系统工程（CASE）、办公系统自动化等项目的研究，迫切要求进一步改进系统研究的方法、提高软件研究的质量。由于传统的结构化方法已很难满足这些复杂大系统的要求，随着对发展的需要，因而出现了更先进，更有效的方法。O-O 方法便是这个时代的产物。从面向对象的编程语言进一步迈向 O-O 认识方法学、O-O 系统分析方法学和 O-O 系统设计方法学等等，使得这些领域有了长足的发展。

1.1.3 面向对象方法的研究动态

目前，在研究 O-O 方法的大气候下，主要有以下几个方面的研究领域：

1. 第五代计算机的研究

在第五代计算机（智能计算机）的研制中，日本正将 O-O 方法引入研究机制，如 PSI 机的系统程序设计语言 ESP 引入了 O-O 编程风格，PIM 的知识程序设计系统 MA DALA 中也引入了 O-O 方法。

2. 新一代操作系统的研究

采用 O-O 方法来设计新一代操作系统，主要有以下几个优点

①采用对象(Object)来描述信息，如系统所需要设计、管理的文件、打印机、其它外设等诸类信息，使语言描述更接近自然。

②O-O 方法对于处理操作系统的诸多事务、如命令、同步、保护、管理等更易于实现和便于维护。

③O-O 方法对于多机、并发控制可提供强有力的支持，并能适当地管理网络与通讯。

3. 多科学的综合研究

当前，人工智能、数据库、程序设计语言的研究有汇合的趋势，O-O 方法是一个很有可能的汇聚点。

4. 新一代面向对象的硬件系统的研究

要支持采用 O-O 方法设计和实现的软件环境，必须有更理想的能支持 O-O 方法的硬件环境。目前采用松耦合（分布主存）结构的多处理机系统更接近于 O-O 方法的思想，因为松耦合的对象可占用不同的处理机，使多处理机都能处于忙碌状态，从而充分发挥每个处理机的效能。最新出现的神经网络计算机的体系结构与 O-O 方法的体系结构更具有惊人的相似，并能相互支持与配合：一个神经元就如一个小粒度的对象，神经元的连接机制与 O-O 方法的消息传送也有类似的关系；譬如，可以把一次连接看成一次消息的发送。

由上可见，O-O 方法具有广阔的发展前景。

§1.2 面向对象方法的基本概念

1.2.1 对象（Object）

客观世界是由许多具体的事物或事件、抽象的概念、规则等组成的，为此可以将要加以研究的事、物、概念等统称为对象。面向对象的方法正是以对象作为最基本的元素，它是分析问题，解决问题的核心和关键。在不同的领域，不同的学者对"对象"的概念也不尽一样。尽管目前尚无统一的认识，但可以抽取其共识，主要有以下几点：

1. 对象是要研究的事物

对象的范围可大可小，对象所以从简单的数字（如1、2、3、4、5）到极其庞大的物体（如航天飞机），或社会组织（如某集团组织）。对象不仅能表现具体的实体，也可以表示抽象的规则、计划、方针、策略等。从目前研究较多的对象来说，主要有以下几种对象类型：

①有形的实体：指一切看得见、摸得着的实物，如机器、计算机、钢笔等。

②作用：指人或组织所起的作用，如医生、教师、职工、公司、部门等。

③事件：指在特定时间所发生的事：如飞行、演出、开会等。

④性能说明：如产品的性能说明等。

对象不仅能表现结构化的数据，而且也能表示抽象的事件、规则以及复杂的工程实体等。这些是结构化方法所不能做到的。因为，对象具有很强的表达能力和描述功能，更接近于自然语言。

2. 数据与操作的结合

对象具有状态，是用数据来描述的，对象的操作用以改变对象的状态，对象及其操作就是对象的行为。

对象的行为是千变万化、丰富多采的。传统的结构化方法只能呆板地描述对象并采用极其复杂的算法才能操纵对象来获得求解，这就产生了某些不能解决的问题。面向对象的方法试图尽可能自然而灵活地模拟现实世界，其中关键的一点就是改变传统方法中将数据（也可以是函数或过程）与操作相分离的做法，实现了将数据与操作封装在对象的统一体中。O-O方法还提供一种机制，使得对象内的私有数据能有效地隐藏起来，从而可灵活地专门描述对象的独特行为，在某些问题上可直接模拟现实世界中的对象，这样设计的软件易于理解和维护。由于对象封装了数据与操作，从而使对象具有较强的独立性和自治性，使其内部状态少受外界的影响，也就是说，它亦具有很好的模块化特点，可为软件重用奠定基础。

3. 对象的识别功能

在对象建立时，由系统给予对象以唯一的对象标识符（Object Identify，或OID），OID之值可用来唯一而且永久地标识对象。亦即在对象的整个生命期中，它的OID值不会改变，不同的对象有不同的OID值。对象被清除后，OID值可以回收，再由系统分配给新创建的对象。此外，对象还具有对象名，对象名代表了数据与操作的一体化。

4. 对象参与对象类

对象应当参与一个或一个以上的对象类，并成为类的实例。（在O-O方法中，对象

与类实例是同义词)。

在大多数 O-O 方法中，类反过来也可以看作是对象，从而实现了类与对象的统一。由于类实例总是由类来创建的，因此，亦可将创建类的类称为元类(Metaclass)，这时，类就可作为元类的实例。

1.2.2 对象类

将具有相同结构，操作，并遵守相同约束规则的对象聚合在一起，这组对象的集合就称之为类。类至少包括以下内容:

①类名。

②外部接口：用于操纵类实例的外部操作。

③内部表示。

⑤接口的内部表现。

1. 类说明(Class Specification)

类说明亦称为类的外部特征或称为外部接口，它往往由一组操作符组成，这些操作符可以送到目标对象中。类说明是为了让用户了解对象类是什么和能够做什么事，因而也可以称为与用户沟通的"外部接口"或"界面"。

以"公司"类的操作符为例说明如下:

①操作符名：检索利润

结　　果：某公司近年的利润

②操作符名：检索子公司

结　　果：返回某公司下属的全部子公司名

③操作符名：增加子公司

变　　量：新子公司名

结　　果：在某公司的子公司名单中，增加一个新的子公司。

2. 类实现(Class Implementation)

类实现是类的内部表示及类说明的具体实现方法，也就是详细设计对象类所说明的功能应当如何做，通常由系统开发人员通过编程实现，对用户来说，是信息隐藏的。

如果仍以"公司"类为例，解释如下:

①"公司"类的内部表示：公司名、地址、支出、税收、职工等描述公司的属性。

②"公司"类的内部实现：实现各种操作的详细模块。通常也可称这些程序模块为"方法"。

对象类的显著特色是将数据的结构与数据的操作都封装在对象类中，并实现了类的外部特性与类实现的隔离，也就是实现了将使用对象、对象类的用户与具体实现对象，对象类的开发者区分开，从而使 O-O 方法具有较好的模块化特性，进而为复杂大系统的分析、设计、实现提供了先进的方法。

注意：对象类(class)和类型(type)是不同的概念。类不是类型，类除了具有本身的结构特性、操作之外，也可以作为具体的例子。类型仅代表数据的抽象描述，即值的集及其

操作，但不可能有具体的例子。类和类型的另一个显著差别还在于静态和动态之别。类型往往是静态数据的抽象描述。而类具有动态性，亦即可以从类动态地生成新的属于该类的对象。

3. 类层次与类格(Class Hierarchy and Class lattice)

对象类之间可能具有层次的结构或格的结构关系。类的层次结构和格结构可用来描述现实世界中概括的抽象的关系。一般情况下，越是在上层的类越具有普遍性和共性，越在下层的类越专门化，不具普遍性。

如："学生"类包括：小学生、中学生、大学生、研究生。"教师"类包括：教授、讲师、助教。"人"类又包括：工人、农民、学生、教师。其结构层次关系如图 1-1 所示。其中，处于最上层的对象类"人"称为超类（又称基类），第二层的工人、农民、学生，教师等对象类就是"人"类的"子类"，或称导出类。同理，小学生、中学生、大学生，研究生可看成是"学生"类的"子类"，即导出类，而学生又可称为它们的超类（基类）。该结构就是类层次结构，其特点是每个子类都只有一个超类。

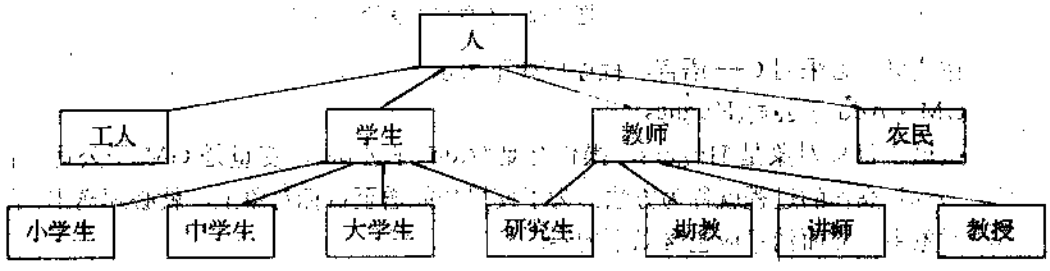

图 1-1　类层次结构图

但是，如果子类有一个以上的超类，就称为类格结构。例如：若研究生子类中，既有攻读学位的，又有从事教学工作的在职研究生。于是其超类既有"学生"类，又有"教师"类。这就是类格结构。

1.2.3　方法和消息 (Methods and Messages)

类通过说明它能执行的操作来决定类实例的行为。在调用一个操作时，应指明以下几点：

①目标对象；

②操作名；

③操作变量；

④调用实现具体操作的程序，并将参数联编到调用程序的实际变量中去。

现举例说明如下：

若 GM 代表通用汽车公司，由此，GM 是"公司"类的一个实例。若 Hughes 也是"公司"类的一个实例，要求将 GM 公司作为 Hughes 的子公司，可用"加入子公司"这个操作来执行，完成这个要求。如图 1-2。

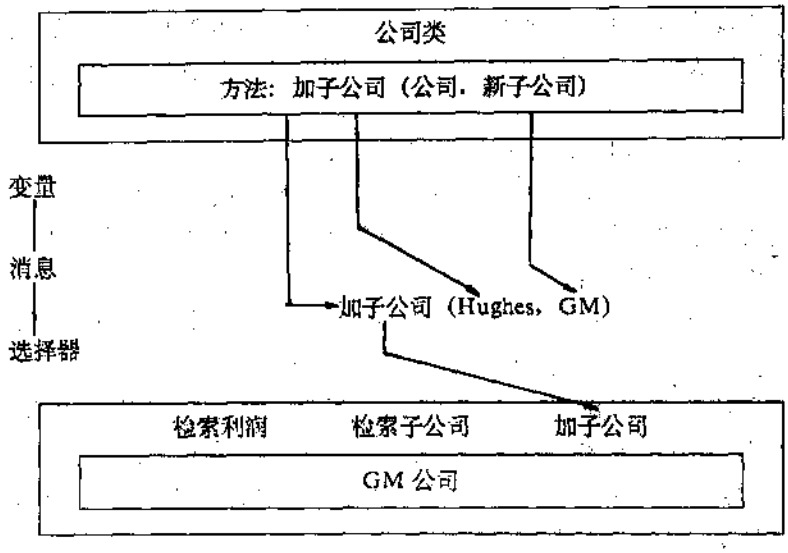

图 1-2 消息与方法例

在这里，若采用 C++语言，程序应如下表示

GM · Add 子公司(Hughes)

这里，目标对象是 Hughes，操作名是"Add 子公司"，变量是 GM 子公司，而实现"加子公司"操作的过程就称为方法。为了使操作能送至目标对象上，就必须送出"消息"到目标对象中，故而消息应包含以下内容：

①选择器（或称方法名）

②一个或多个变量。

由选择器从目标对象的有关方法中选择出合适的方法。当选中相应的方法后，就将消息中的变量联编到方法的参数中，该过程类似于传统结构化编程语言的过程调用，其最大的差异在于 O-O 方法的消息中，变量联编既可在编译时进行，也可以在运行时进行。这种变量在运行时联编的过程就称为迟后联编（late binding）或称动态联编（Dynamic binding）。

由以上可粗略知道，对象类的操作是由方法来具体实现的，其方法通常是不可见的，静态的。当对象之间要进行相互联系和通讯时，是通过"消息"传送作为唯一的途径来实现的。对象一旦接收到一条具体的消息，就要激活与此消息相匹配的某一方法，并执行这个方法，执行的结果又生成一个新的对象。

§ 1.3 面向对象方法的特点

1.3.1 抽象性(Abstraction)

抽象是对复杂的现实世界的简明表示，它着重强调我们所要研究的重要信息，而忽略所不重要的信息。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，推动了整个软件工程的进展。

之所以说 O—O 方法比传统的结构化方法具有更高的抽象性，它还有如下几个方面的特点：

(1) 抽象的表达能力：O—O 方法采用对象来表达一切事物，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，甚至包括复杂的工程实体、图形、声音以及规划等等。由于对象的这种高度抽象的表达能力，使面向对象的方法具有很强的建模能力。

(2) 抽象的数据类型：在对象抽象的基础上，O—O 方法进一步提出了对象类这一基本概念。对象类实现了更高级的抽象，即具有相同或类似特性的一组对象组成对象类之后，对对象的共性加以抽取，并进行统一的说明，从而省略了对每个对象共性的重复说明。

对象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有的更高级的抽象，即类定义时，既可由系统，也可由用户来定义对象类所具有的数据类型，称为抽象的数据类型。更突出的一点是，它可根据需要由用户灵活地定义数据类型，从而使 O—O 方法具有很强的解决复杂问题的能力。

另外，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，这样就使相同消息各作用于不同的数据类型。因此，O—O 方法具有结构化方法所不可比拟的高度的抽象性和灵活性。

1.3.2 封装性(Encapsulation)

封装性是保证软件部件具有良好模块性能的基础，主要包括以下几个部分：

①所有软件部件的内部都有明确的范围以及清楚的外部边界。

②每个软件部件都具有较好的用户界面，用以说明软件部件之间的相互联系与相互作用。

③完全保护软件的内部实现，有利于用户集中精力去考虑所开发的系统，各模块之间的关系等问题，而模块的内部实现没有必要再作为用户的研究对象。

O—O 方法提供的完整的封装性是指以下几点：

①对象类是封装良好的模块，类定义将其说明与实现显式地区分开来，其内部实现按具体定义的作用域提供保护，可分为私有、局部、全局等。如：类所定义的私有量仅共本类所使用，它类不可访问，类所定义的全局量则允许所有类访问。

②对象是封装的最基本单位，对象类定义为本类所有对象的共性。在采用 O—O 方法解决实际问题时，需要在类定义的基础上加以实例化，亦即要建立具体的该类的对象。每当创建一个新的类实例（即对象）时，除了具有类定义的共性并加以量化（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。

③O—O 方法的封装优于结构化方法的封装。结构化方法不可能实现完整封装。结构化封装是以库函数形式进行的，库函数包含一个以上的函数和有关的数据抽象，但库函数的内部实现与外部接口的界限分不清楚，仅具有部分封装特性。

1.3.3 继承性(Inheritance)

继承性只有 O-O 方法才有，其它方法一概不具有，因而是构成 O-O 方法的一大特点。

继承性体现并扩充了 O-O 方法的共享机制，有以下几种共享的情况：

①对象共享它所在类的结构、操作与约束等语义特性，换句话说，当创建一个新对象时，只要声明它所参与的类，这个新对象就能自动地继承该类的全部语义特性。

②传递性：类层次的继承具有传递性，子类能自动继承其超类的全部语义特性；若类层次具有多层的话，这种继承就具有传递性，最下层的子类能继承其上各层超类的全部语义特性。因此，也可称之为一种垂直的多层共享机制。

③类格的继承性：这是从横向实现共享的机制，即子类可从多个超类中继承它们的语义特性，也称为多重继承。

根据继承的以上特性，继承的作用亦可概括为以下几点：

①采用继承性使所建立的软件系统具有开放性。系统的开发应当尽可能地选用公用接口，实现信息的交换与共享，模块应可能地多重用。由于 O-O 方法的继承性，尽可能利用已建立的系统或已经建立的类，以它们为基础而进行扩充，不必重新开始设计整个系统。

②类层次或类格的继承性是信息组织与分类的行之有效的方法。类层次和类格反映了现实生活中普遍存在的一般与特殊的语义联系，越靠上层的类越具有普遍性和共性，越靠下层的类越具有特殊性。

②简化了对象、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对象类是已存在对象类的子类，就可自动继承类或超类的共性。这样，使创建对象或对象类的定义变得简单和容易，也增强了以 O-O 方法开发的系统的可扩充性。

④增强了代码的可重用率，从而提高了软件系统的可靠性。在 O-O 方法中，代码的重用是通过继承超类的方法来实现的。

1.3.4 多形性(Polymorphism)

多形性是测重强调 O-O 方法中的为可在不同时间内保存、取用及返回不同类的类型值。亦即相同的操作（过程或函数）可作用于多种类型的对象上并获得不同的结果。具体的多形性行为主要表现为以下几个方面：

①运算符重载，指同一运算符可作用于多种数据类型上。对 O-O 方法来说，不仅能作用于系统定义的数据类型上，还能作用于用户定义的数据类型上。

②函数名重载。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。

③虚函数与动态联编。当重载的函数名前通过标识符（如用 Virtual）标识其为虚函数时，该函数就具有较大的灵活性，它既可以表示子类中的同名函数，也可以表示超类中的同名函数。还有一个突出的优点是允许在运行时（而非编译时）才按照具体的数据类型

和参量数来确定选用哪一个函数，这种方式也称为动态联编。

由以上特性可知，多形性亦有如下两点长处。

①多形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，增强了操作的透明性、可理解性及可维护性。

②多形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。多形性与继承性相结合，使软件具有更广泛的重用性和可扩充性。

§1.4 从结构化编程语言到面向对象的编程语言

1.4.1 高级语言的发展

随着计算机的不断普及与推广应用，与其密切相关的计算机语言也经历了由"机器语言"到"汇编语言"进一步发展到"高级语言"。

高级语言克服了机器语言难读难写难于理解的缺点，使人们采用类似于自然语言的形式与计算机系统打交道，从而推动计算机事业的发展。通过不断研究和改进，高级语言的发展可分为三个阶段：

①基础语言阶段：基础语言是人们熟悉、应用最为广泛的语言，其共同特点是具有很好的表达能力，如 Fortran、Cobol、Algol、Basic 等都属于基础语言。随着软件规模的扩大，这些以语句序列作为程序的基础语言就不便于管理和维护，于是便陷入了软件危机，从而迫切需要一种新的模式取代语句的语言形式，在这种思想的指导下，出现了结构化的程序设计语言。

②结构化语言阶段：为了克服 60 年代出现的软件危机，1968 年由北约组织提出"软件工程"的概念，对程序设计语言的认识从强调以表达能力为重点转向以结构化和简明化为重点，将程序从语句序列转向将程序作为相互作用的模块的集合。于是 Pascal、C 语言得以蓬勃发展。这种语言支持结构化的程序设计，程序中的任何逻辑问题均可用"顺序"、"条件"和"重复"三种基本结构加以描述。

结构化语言的编程方法就是确定所需要的过程和采取能找到的最好算法，其中，过程的设计为整个语言构成的核心。结构化语言支持过程设计是通过将参数传给函数，并由函数返回值。

结构化方法较之语句逻列式方法的特点是：显著地减少了软件的复杂性，提高了软件的可靠性、可测试性及可维护性。这一特点使 70 年代被誉为"结构化语言"的年代，从而使广大的程序员在这一时期取得了辉煌的成就。

③面向对象编程语言阶段：进入 80 年代，尤其是九十年代以来，由于一系列高新技术的研究，如第五代计算机、CAD / CAM / CIMS、知识工程等一系列项目的研究和发展，都迫切要求大型软件系统的支持，它们所使用的数据类型也超出了常规的结构化数据类型的范畴，从而提出了对图形、语言、规则等非结构化的信息的管理。

为了适应时代的要求，强烈要求模块应具有更强的独立自治性，以便大型软件的管

理、扩充和重用。

由于结构化语言的数据类型较为简单，不能胜任对非结构化数据的定义与管理；采用过程调用机制也不够灵活，独立性差；在规模庞大的复杂软件面前，它显得无能为力或力不从心，因此，强烈要求一种新型的语言结构来取代它。

为了适应高新技术发展的需要，消除由于结构化编程语言的局限，自 80 年代以后，出现了面向对象的程序设计语言(Object-Oriented Programming Language)，这种设计方法将计算或解题看作一个系统的开发过程，系统包括若干对象类，经一连串的消息传送产生相应的对象状态的变化以获取所需的结果。

由于面向对象的程序设计语言（简称 O-OPL）的对象、对象类具有高度的抽象性，能很好地表达任何复杂的数据类型，也允许用户灵活地定义自己所需的数据类型，对象类本身具有完整的封装性，从而使它作为编程中的模块单元，满足模块独立性高的要求。另外，由于其有继承性和多形性的特点，更简化了大型软件的开发。

1.4.2 O-OPL 的基本特征

基于对象的语言是这样定义对象的：对象具有一个操作集合和一个"记忆"操作结果的局部共享状态。对一个对象实施某种操作所返回的值取决于对象的状态及操作的变元。对象的状态如同局部存储器，为该对象上的所有操作所共享，尤其是已执行过的其它操作可能影响特定操作之返回值。不同语言中的对象概念，基于对象的不同语言中，大致可分为四类：

①函数式对象。如 OBJ_2 面向对象的函数式语言以及 Valcan 面向对象的逻辑型语言中的对象，均属于函数式对象。它们的接口类似对象，但对象在各操作之间不再是同一的，用于状态的变换的操作会产生带有给定接口以及新状态的新对象。

②被动式对象。在同步消息或远程过程调用激活它们的操作时，对象才变为主动。如 C++或 Smalltalk 等传统的对象均属于被动的对象。

③主动式对象。即使无其它对象的请求或调用，该对象也能自动地执行操作，这类对象就属于主动式对象。如共享变量的并发程序设计语言的重要构造进程、强耦合并发对象以及分布式并发对象均属于此类。

④基于槽的对象。根据其实例变量（即槽 Slot）进行定义的对象就称为基于槽的对象。这是知识的框架表示的映射。Flavors 和 Commonlisp 语言中的对象均属于此类。

按照对象是一个具有局部状态和一个操作集的定义，对象可形式化描述为：

对象::＝<OID, MS, DS, MI>

其中：OID 是对象标识符

MS 是对象的操作集合 $MS=\{OP_0, OP_1, \cdots, OP_n\}$

DS 是对象的数据结构 $DS=\{S_0, S_1, \cdots, I_n\}$

MI 是消息名集，也是外部接口 $MI=\{I_0, I_1, \cdots, I_n\}$

在建立了对象的概念之后，我们再了解一下类，类是对具有相同语义特性的一组对象的抽象。类将这组对象的操作特性、存储特性、约束规则等加以归纳，统一由类定义其共