ISBN 7—5027—2878—3 / TP·128
定价：13.00 元

面向对象应用程序的图形编程工具

ObjectCraft 用户参考手册

阳 风　祥 子
陈剑飞　谢广平　编

海洋出版社

内 容 摘 要

ObjectCraft 是一种面向对象程序设计的简捷工具，它不是用面向对象的语言（C++或 Turbo Pascal5.5）编程来建立对象，而是以图形的方式画出对象，然后自动产生 C++或 Turbo Pascal 代码。本书在介绍 ObjectCraft 的同时，也详细介绍了 OOP 的基本原理，为 OOP 的新、老用户提供了许多范例。尤其对于 OOP 的初学者来说，ObjectCraft 可谓是一种十分理想的学习工具。

本书可供高等院校计算机系统、应用和管理等有关专业师生以及各部门的计算机工作者、科研、工程技术人员学习参考。

责任编辑：阎世尊

北京希望电脑公司 *OOP* 系列丛书

面向对象应用程序的图形编程工具
ObjectCraft 用户参考手册

白 枫　祥 子
陈剑飞　谢广平　编
希 望　审校

*　*　*　*　*

海洋出版社出版（北京市复兴门外大街 1 号）
海洋出版社发行　　兰空印刷厂印刷
开本：787×1092　1/16　印张：14.625　字数：328 千字
1991 年 8 月第一版　1991 年 8 月第一次印刷
印数：1-3000　册

*

ISBN7-5027-2878-3/TP・126　定价：13.00 元

前　言

ObjectCraft 是一种开发面向对象编程(OOP)技术和应用程序的软件工具,它可根据用户的想象自动生成编码.

OOP 作为一种新的编程方法，具有许多优点．诸如自然描述、可重用码和易于维护等，并给通常的问题分析和软件设计注入了新的生命力．OOP 的编程语言有 C++和 Turbo Pascal 的近期版本，这些语言目前已在各应用领域得到了广泛的应用．

ObjectCraft 能帮助用户学习 OOP 的方法，同时也能使用户更快、更有效的开发其应用程序．该工具无需用户用 OOP 语言写出代码，用户只需把要包含在应用程序中的对象以及对象之间的联系图画出来即可．一旦用户画出了应用程序图，ObjectCraft 即自动根据图形产生 C++或 Turbo Pascal 代码．

ObjectCraft 即适用于 OOP 的初学者，也适用于富有 OOP 编程经验的用户．对于初学者，ObjectCraft 是一种学习 OOP 基本原理和技术的快速工具；而对于后者，则能使其更有效的提高编程效率和编程技巧．

目　录

简　介

面向对象程序设计（OOP）是基于对程序设计的新的思路上的设计技巧．它已经导致新的语言、新的操作系统，面向对象的数据库，以及研制人机界面的新方法的诞生．它给予我们分析问题、设计软件以全新的思路．我们相信，在软件的发展过程中，在计算机指令中将会越来越强调对象这一关键概念．但是，目前大多数人刚刚开始学习有关 OOP 的知识．ObjectCraft 是一个理想的学习工具，另外，它也是研制面向对象程序的好方法．

采用 OOP 方法的关键优点是：自然的描述，代码可重用性，软件易于维持．OOP 也提供了新的、较好的对结构的应用方式．早期的 OOP 应用规模比较小，这些较小的应用导致程序易于修改和维护．换句话说：有很多理由表明，程序开发者很有必要学习 OOP 技巧．

ObjectCraft 是进行面向对象程序设计（OOP）的可视工具，不是用面向对象的语言（C++或 Turbo Pascal 5.5）编程来建立对象，而是以图形方式"画出"对象．ObjectCraft 提供了比 C++或 Pascal 5.5 更适用的编程环境．一旦使用 ObjectCraft 建立好应用系统，它就能够自动转化为 C++或 Turbo Pascal 代码．

使用 ObjectCraft 的硬件要求是：带有 DOS 2.0 或更高的 DOS 版本的个人计算机，其内存不小于 512K 字节，带有能够显示增强的 EGA（EGA 板上带有不少于 256K 随机存贮器）或 VGA 图形的监视器，以及一个鼠标器．不需要 Turbo Pascal 5.5，Glockenspiel C++，Borland C++，或者 Zortech C++．如果想把用 ObjectCraft 实现的应用系统转变成 C++或 Turbo Pascal 5.5，还需要它们的编译程序．本书的后面附有 ObjectCraft 的命令形式．

使用 ObjectCraft 有很多种方式．对于 OOP 的初学者来说，ObjectCraft 是一个学习工具．可以在计算机屏幕上建立对象，试试这些对象如何工作，据此可以掌握 OOP 的一些概念．通过使用 ObjectCraft，可以养成分析和设计好的 OOP 的习惯，却不必处理由 C++或 Turbo Pascal 引出的许多新的、复杂的语法．而且，ObjectCraft 确保用户集中精力建立面向对象的结构，从而充分利用面向对象途径的优点，即不会掉进使用 C++之类的面向对象的语言实现传统应用时存在的陷阱．

如果一个熟练的 OOP 编程者使用 ObjectCraft，可以有多种方式．可以用它帮助开发 C++或 Turbo Pascal 程序，通过它迅速建立一个原型，这样做可以提高开发效率．可以用 ObjectCraft"画出"程序的数据结构、数据流，以及有关的接口．然后，ObjectCraft 将自动把这些流图转换为代码．

用 ObjectCraft 生成的流图不仅可以转换成代码，而且它们本身就是可以运行的程序，因为 ObjectCraft 可以在"解释"的环境中运行．也就是说，所建立的对象都是有一定的功能的．利用这个特点，一旦对象建立后，可以马上检查它的性能，如果它的性能令人满意，再把它转换成代码．或者，就用这些流图作为可运行程序使用，不再把它们转换成代码．本手册是对 ObjectCraft 软件包的关键部分的说明．它首先概要地介绍了 OOP，接着一步一步地介绍了 ObjectCraft 的使用，ObjectCraft 程序的规范的参考资料，生成面向对象应用软件的一般步骤．本手册分五个部分．

★ 第一部分　引　论

第一章介绍了 OOP，第二章介绍了安装和运行 ObjectCraft 软件包的一些特定的知识，它也详细地介绍了 ObjectCraft 编程环境，包括全部的下拉菜单命令和图形"绘画"环境.

一些读者希望在使用 ObjectCraft 之前获得一些有关 OOP 的背景知识，另外一些读者喜欢先获得一些实际编程和操作知识，再返回来学习一般性介绍．两种安排都是有效的.

希望先获得有关 OOP 的知识的读者，可以从第一章开始，另外一些读者可从第二章开始，接着阅读本手册第二部分的章节.

★ 第二部分　初学者指导

第二部分通过介绍一些精心设计的练习使读者熟悉如何使用 ObjectCraft．到这部分的结尾读者将已经开发了一个小的应用软件，附录中列举了其 C++代码和 Turbo Pascal 代码.

★ 第三部分　ObjectCraft 的基本操作

第三部分详细地讨论了 ObjectCraft 的基本操作．事实上，第三部分就是 ObjectCraft 的参考手册.

★ 第四部分　对 ObjectCraft 的进一步浏览

第四部分给出了三个小的但有趣的应用软件，通过它们的示范作用，介绍 ObjectCraft 的更高级的特点.

★ 第五部分　生成面向对象程序

第五部分一步步地介绍了分析、设计、生成面向对象应用软件的方法.

★ 附录

附录提供了一些 C++或 Turbo Pascal 5.5 的代码例子，这些代码都是由第二部分介绍的应用软件通过不同的代码生成选择转换而来的.

第一部分　引　论

第一章　面向对象的程序设计

1.1　传统的程序设计

这一章介绍了面向对象程序设计（OOP）中的基本概念和技术．本章的目的是使读都习惯 OOP 的特性及操作．本章介绍了 OOP 的历史以及 OOP 的种类，使读者对不同 OOP 语言、工具在功能上、术语上的差别有所了解．首先简单地讨论一下传统的程序设计语言，接着介绍 OOP 语言的不同之处．

对于有 ObjectCraft 软件，而且急于开始使用该软件的读者，可以跳过本章，从第二章开始，在第二章介绍了如何安装和检查 ObjectCraft 软件．然后，去阅读第二部分开始处的指导性介绍，经过一些实际使用后再回到本章来．另一方面，如果读者希望首先系统地了解 OOP 的概念，建议从本章开始．

程序设计始于第二次世界大战以后，这时第二代计算机已诞生了．第一位程序员先用机器码再接着用汇编语言书写程序．也就是说，程序员汇编了一连串顺序式的语句，然后启动系统，期待着系统会按顺序一步步执行这些顺序式的语句（程序）．

在 50 年代早期，高水平的函数式或过程式语言诞生了 —— 例如，FORTRAN，COBOL，以及 ALGOL．这些第三代语言鼓励了程序员把编程任务分成两个部分。首先，程序员列出他们预计要使用的数据项，接着，把整个程序流程分解成块．显然，有些块将被重复使用，这一点鼓励了程序员把这些块的代码写进过程中，以便于反复调用这些代码．这整个过程一般被称为功能分解．逐渐地这种方法导致了更高水平上的抽象 —— 划分并确定最通常的几步（称作程序抽象）—— 然后再对确定的每一步进行提炼和细分，这个过程可以反复进行，直到程序员满意为止．这样做，可以开发出一些子程序或子进程，它能被程序的不同地方反复调用．

为了编程更方便，已经有子进程库派生．例如，大多数 C 语言编程人员可能用过子进程，例如 printf（ ），但并不知道其中的代码是什么．

在 70 年代后期，结构化程序设计方法派生，这种设计方法帮助程序设计者分析问题，开发大的软件系统．早期的结构化方法只不过是对功能分解方法的原理的系统性确立．在最近的几年中，不管怎么说，这种方法越来越强调程序设计的两个方面，既要求设计者以数据流图的形式描述程序流，也要求设计者以数据图的形式描述数据，后者常被称作实体关系图（ER）．导致这种情况的原因是，使用者如果弄不清楚其中数据的表示和操作，就不能弄清复杂的应用软件．尽管如此，许多程序设计者仍然关注于这种方法．

1.2 面向对象的程序设计（OOP）

面向对象的程序设计（OOP）代表了程序设计方法的主要变化：程序概念化。一个对象就是一个数据结构，其中含有数据和子程序。

面向对象软件是真实世界中部分领域的精确模型。现实世界中的事物在软件中用对象来表示。软件中对象对输入的响应，就像真实世界中该对象对接收到的信息的响应。这种模拟使得程序设计任务变得容易些。最初的问题领域的流图便是程序的结构。

图 1.1 从下到上显示了从机器语言到被分析的真实世界环境的一系列层次。先前人们认为程序是一连串的顺序代码，计算机将执行这些顺序代码，慢慢地人们的观念转向了程序应是真实世界的模型。

熟悉程序化语言的人们往往难以理解把程序作为现实世界的模型这一新的观念。这些人已经习惯了这种想法：当计算机硬件执行程序时，程序就是一连串将要发生的动作。然而，开发过带有复杂的人机界面的交互式软件的程序设计人员却认识到，把程序理解为一连串动作是困难的。高度交互式软件的数据结构往往很丰富，常常由一些经常被调用的小程序组成。

OOP 把设计者的注意力从功能块的顺序设计转移到描述问题的数据结构上。但是，这仅仅是信息化模型方法的特点，OOP 还不止这些特点，它强调数据，建立了一种新的数据结构：数据和程序式代码封装在一起。而且，OOP 比传统的"结构化设计方法"要有效得多，因为程序的基本结构模拟了现实世界。把数据和程序式代码封装在一起听起来简单，但它使得程序更简单、更容易维护和修改。

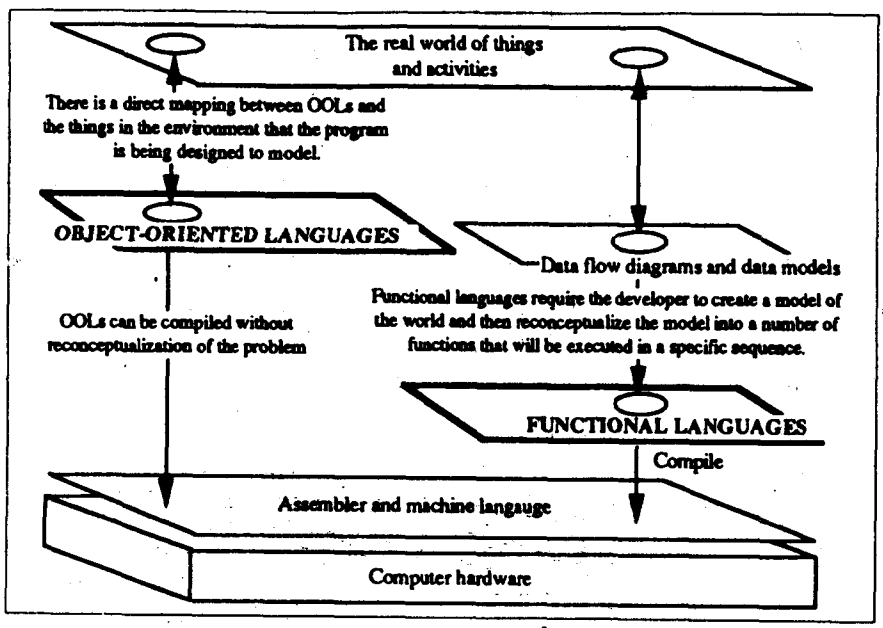

图 1.1 从计算机硬件到现实世界的一系列层次

已经介绍了 OOP 的优点，下面介绍 OOP 是什么．实际上，关于组成面向对象语言（OOL）的要素有哪些一直存在着争论．为了本手册以后介绍的需要，我们假定 OOL 有四个特点：

1. 对象包括数据以及对数据的操作
2. 对象能被划分成类（classes）和实例（instances）
3. 继承是从类到实例、从上到下的继承．
4. 操作是由对象之间传递的信息来激活．

OOL 通常有两类："纯"OOL 语言和"混合式"OOL 语言．纯 OOL 从单一的根或类派生而来的，这个根或类规定了派生的对象能做什么．Smalltalk 就是这样的一个好例子．大多数 OOL 是好的原型化工具，但在传统的带有计算工作的问题领域里，这些语言增加了程序设计人员的许多负担，从而使程序设计的效率降低．

混合式 OOL 是在现有的一种高水平语言的基础上发展起来的语言，例如，C++是由 C 语言发展来的，Turbo Pascal 是由 Pascal 发展来的．对于刚开始从事应用软件开发人员来说，混合式 OOL 用起来要稍稍困难些，但对比较熟练的程序设计人员来说，混合 OOL 存在着语法方面的优点，它们的编译、连接、以及运行更加有效．

ObjectCraft 结合了两者的优点．ObjectCraft 允许程序设计人员使用 C++或 Turbo Pascal 5.5 中没有的 OOP 技巧．同时，ObjectCraft 允许使用程序化语言的一些语法规定．然后，ObjectCraft 自动生成 C++或 Turbo Pascal 5.5 代码．另外，ObjectCraft 能自动建立一些代码（例如，构造函数）并加以注释．因此，ObjectCraft 使程序设计人员能集中精力考虑 OOP 的采用．

本文先简短地介绍组成 OOP 的基本概念和方法．接着，回顾了 OOP 的发展历史，以及 C++，Turbo Pascal 5.5，ObjectCraft 在 OOP 的整个发展过程中充当的角色．

本文每次介绍 OOP 的一个关键特征，顺序如下：(1) 对象，(2) 类和实例，(3) 层次和继承，(4) 操作和信息，(5) 多态性、动态赋值以及虚拟操作．不幸的是，由于这些概念相互之间不是独立的，所以不能不考虑其它概念而单独详细介绍某一概念．如果文中使用了未加定义的概念，后面将用几页纸的篇幅详细介绍它．

1.3 对象 (Objects)

OOP 中的基本概念是对象．一个对象是由数据和程序组合而成的数据结构．图 1.2 显示了基本的对象．图中用两种方式图示对象．左边采用的是有关 OOP 的书籍中常用方式，右边是 ObjectCraft 中表示对象的常用方式．

左图强调了对象对数据和程序化代码的封装性，右图适于在计算机屏幕上建立或操作．而且，ObjectCraft 允许在表示对象的椭圆形中隐藏数据和程序式代码．这样做并且打开一个对象以后，屏幕上将生成一个包含组成该对象的所有数据和程序式代码的窗口．

对象由四个特征来定义：(1) 对象名，(2) 父对象，(3) 属性（槽）的名字、种类、数值大小，(4) 程序式代码表示的操作．

★ 对象名：每个对象有一个名字．

★ 父对象：如果一个对象是从其它对象派生而来的，父对象的名字必须被保留在该

对象中．如果没有父对象，即该对象是根对象，那么该对象中没有指向更高层次对象的指针．

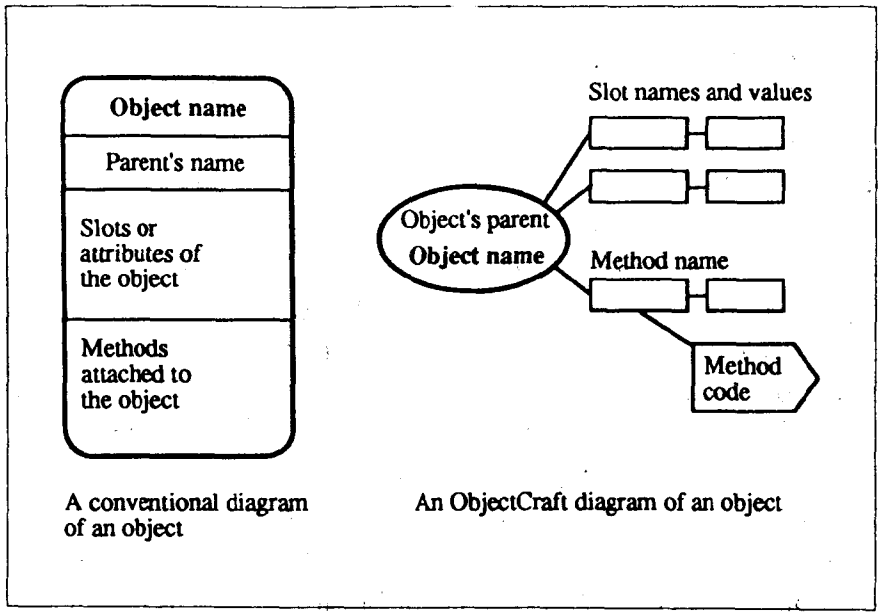

图 1.2　对象的两种图形表示方式

★ 属性（槽）：在本文中，对象中的数据项被称作属性（槽）．

属性（槽）包括数据项的名字、数据的种类、数据的实际大小．

有些作者把属性（槽）中的不同部分当作不同分面．因此，属性（槽）名是一个分面，属性（槽）中数据大小是另一个分面．

一些 OOL 允许一个属性（槽）中可以有多个数值，但 C++、Turbo pascal 5.5，以及 ObjectCraft 不允许这样做．

有些语言称属性（槽）为"变量"．

一个属性（槽）可能包括一个数值．因此，描述"人们"的对象中可能有一个其值为某人姓名的属性（槽）．相同的对象可能还有地址（槽）．在这种情况下，地址（槽）可能是一个指向不同对象的指针．这个不同对象可能又有许多属性，包括：街道、城市、邮政编码、国家，以及电话号码．

★ 操作：在本文中，对象中的代码块被称作操作．操作可以被划分成过程和函数，下面将仔细讨论它们．

有些作者称操作为"过程"，另有些作者则称它们为"函数"．Turbo Pascal 5.5 和 ObjectCraft 等语言要求程序设计人员区分"过程"和"函数"的不同，因此本文采用通用术语操作描述过程或函数．操作将由对象之间信息激活，或由一些外部程序来激活．由于信息是动态的，不用图形来表示信息，虽然可以从对象中的操作观察出一些信息来．后面将进一步介绍操作，子程序、函数以及信息．

本文中用对象作为通用术语，既表示类对象，也表示实例对象。当有必要区分类对象和实例对象时，文中采用“类对象”和“实例对象”来加以区分，而不是笼统地采用术语“对象”。

1.4 封装

一个对象就是一个自组织单元，它包括数据和处理这些数据的操作，这种机制叫“封装”。换言之，数据被封装在一个对象之中，如果不通过该对象中的操作，其它对象或程序是不能访问或修改该数据。这种做法是由早期的“信息隐藏”或“数据抽象”演变而来的。这些做法都强调把数据隐藏起来，只能被特定的程序访问。

如果封装成功的话，封装保证了安全性。它保证程序设计人员不用担心一个函数或子程序会处理不该处理的数据。

Smalltalk，C++，还有其它几种 OOL，都支持封装。ObjectCraft 和 Turbo Pascal 5.5 都不禁止程序设计者编制非封装代码。因此，程序设计者可以设计操作访问不同层次对象中的数据。ObjectCraft 允许这种做法，目的是让程序设计人员在正式生成软件之前用原型化方法快速生成软件。Turbo Pascal 5.5 不支持封装。C++语言提供支持封装性，但设计都可以利用该封装性，也可以不利用。无论是 C++，ObjectCraft，还是 Turbo Pascal 5.5，都强调程序设计人员最后编制成的程序中要避免对数据任意的直接访问，而应当采用封装性。如果编制程序时没有采用封装性，便是放弃了 OOP 的一个关键性优点，编制成的程序将难以维护。

1.5 类和实例

对象有两种：类和实例。类比较抽象。它们就相当于描述由该类派生出实例的共性的模板（Templates）。实例是针对现实世界中特定的存在。以数据库来打比方，类像文件头，它确定记录的格式，但自身并不包含任何数据元素。一个实例相当于一个记录，它含有特定的值。进一步地，属性（槽）相当于数据库中的字段。至于操作，数据库中没有与之相当的内容。

本书中用“对象”作为一般术语，用“实例”和“类”区分两种对象，有时也常用“类对象”“和“实例对象”更清楚地区分这两种对象。大部分书籍避免建立明确的术语，但这样做是值得的，因为这样做可以避免混淆。类对象和类、实例对象和实例是等同的。

1.5.1 类对象

类对象，或者类，是为建立实例对象准备的模板。

C++称类对象为“类”（classes），Turbo Pascal 5.5 称类对象为“对象种类”（object types）或者“种类”（types）。

一个类对象由与定义对象采用的相同的四个特征来定义：（1）对象名；（2）与其它对象（父对象）的继承关系；（3）属性（槽）的名字、种类、数值大小；（4）操作。

★ 类对象名：每个类对象必须有唯一的名字。