编著◎张希伟　叶　枫

# 程序设计方法教程

河海大学出版社

# 程序设计方法教程

编著　张希伟　叶　枫

河海大学出版社

**图书在版编目(CIP)数据**

程序设计方法教程/张希伟，叶枫编著. —南京：河海大学出版社，2009.4
ISBN 978-7-5630-2592-3

Ⅰ. 程… Ⅱ. ①张… ②叶… Ⅲ. 程序设计—方法—高等学校—教材 Ⅳ. TP311.11

中国版本图书馆 CIP 数据核字(2009)第 063943 号

| | |
|---|---|
| 书　　名 | 程序设计方法教程 |
| 书　　号 | ISBN 978-7-5630-2592-3/TP·128 |
| 责任编辑 | 代江滨 |
| 责任校对 | 刘凌波 |
| 封面设计 | 杭永鸿 |
| 出版发行 | 河海大学出版社 |
| 地　　址 | 南京市西康路 1 号(邮编：210098) |
| 电　　话 | (025)83737852(总编室)　(025)83722833(发行部) |
| 经　　销 | 江苏省新华发行集团有限公司 |
| 排　　版 | 南京理工大学印刷厂 |
| 印　　刷 | 南京玉河印刷厂 |
| 开　　本 | 787 毫米×960 毫米　1/16　11.25 印张　300 千字 |
| 版　　次 | 2009 年 4 月第 1 版　2009 年 4 月第 1 次印刷 |
| 定　　价 | 28.00 元 |

# 前　言

程序是人类思维的产物，也是人类思维火花的时间定格，它呈现静态特性。作为产生程序的过程，程序设计却又是动态的，它反映了人类思维的规律和模式。

程序设计方法是20世纪70年代形成和发展起来的软件工程的一个分支，主要探讨程序的性质以及程序设计的理论和方法。近年来，随着程序设计理论的不断创新以及开发技术的改进，程序设计方法的内容也发生了很多的改变。为了介绍这一领域的一些基本思想方法和实际应用，作者在总结多年研究成果与教学实践的基础上，撰写完成本书。本书除了讨论程序设计方法中最基本和最成熟的知识外，还引入了目前流行的一些程序设计方法，主要包括UML方法和设计模式等。

本书的前两章简要描述了程序设计的基本概念以及程序设计的语言与环境；第3章主要讨论了面向过程的程序设计方法；第4、5、6章重点讨论面向对象的程序设计方法、设计的基本原则及设计模式等内容；第7章描述了程序设计优化方面的技巧；第8章通过一个实例来总结前文所涉及的各种程序设计方法。

与国内外同类书相比，本书系统性强、层次分明、通俗易懂、便于自学，并结合作者的理解和体会来阐述基本概念和特定问题，用大量的程序实例说明了程序设计的应用和正确性证明。另外，本书没有采用统一的语言来描述程序，这样可以使读者接触到更多的程序控制结构和设计风格，有利于读者阅读其他相关专著。

本书第1、2章由张希伟编著，第3、7、8章由俞佳编著，第4、5、6章由叶枫编著。全书由张希伟统稿。书中的程序部分由程明、刘海龙负责编写与调试。在本书的编写过程中，得到了河海大学计算机及信息工程学院的大力支持，我们在此表示衷心地感谢。

本书的所有内容都经过了作者的精心策划和安排。此外，本书的编写参考了大量的中外文献，并在参考文献中一一罗列，编者对这些文献著作者表示真诚的谢意。由于本书涉及的内容广，加之现代计算机技术的迅速发展，限于作者的水平和时间，难免有错误和不妥之处，恳请专家和广大读者批评指正。

编　者

2009年2月18日

# 目 录

# 第 1 章　程序设计概述

## 1.1　程序与程序设计

### 1.1.1　程序与软件

程序是对所要解决的问题的各个对象和处理规则的描述，或者说是为了解决某一问题而设计的一系列计算机能识别的指令（由操作码和操作数组成）。

程序设计是指程序的形成过程，是在计算机上使用可执行的程序代码来有效描述解决特定问题算法的过程。程序是人类思维火花的实现定格，呈现出静态特征，而作为产生程序的过程，程序设计却是动态的，它反映了人类思维的规律和模式。人类运用其逻辑思维能力以及符号处理能力来构造一个特定的符号处理器，使得借助于计算机这样一种设备，完成预定义的计算。程序设计及其相应文档，是指软件开发过程中的实现阶段的思维活动和相应的记录。

软件是计算机程序、方法、规则以及所要求的文档资料和在计算机上运行时所必需的数据的总和。软件发展经历了三个时期：程序设计时期（20 世纪 40 年代末至 20 世纪 50 年代末）、软件 = 程序 + 说明时期（20 世纪 50 年代末至 20 世纪 70 年代末）及软件 = 程序 + 文档时期（20 世纪 70 年代初至今，即软件工程时期）。

程序是软件的一个组成部分，是一组预定义的工作指令流集合，通过执行程序，可以在计算机世界中智能地再现现实世界中具体的客观问题，从而使问题得到解决。计算机系统包括硬件系统和软件系统，硬件是由运算器、控制器、存储器、输入设备和输出设备组成，其中运算器和控制器称为中央处理器，它是计算机的核心，由大规模的数字集成电路组成。软件是在计算机硬件之上的功能扩充，它包括了使计算机运行所需的各种程序以及相关文档资料。软件工程师将解决问题的方法、步骤变成由一条条指令组成的程序，输入到计算机中。所谓指令就是计算机可识别的命令，相当于人与计算机之间交流的语言，计算机执行这一指令序列，便可完成预定的任务。

最初的程序设计全凭设计者个人经验和技艺独立进行，是一种典型的手工艺智力劳动。为解决软件研究和开发中面临的困难和混乱，软件的开发必须以工程化的思想为指导，运用标准和规范的方法来进行。

程序编写，即通常所说的“编程序”，是软件开发工程中的一个关键阶段。软件的质量主要是通过程序的质量来体现的，因此程序设计在软件开发过程中占有十分重要的地位。程序设计人员既需要有严密的思维能力，又需要有创意甚至是一定的艺术修养。成为一个优秀的程序设计者需要经过艰苦的努力。

对应于软件开发的分析、设计、实现、维护四个阶段，从一个具体的问题域到相应的软件系统的形成，人的思维中心也要经历多次转变，每一次转变都取决于本阶段的目标和现有的技术基础，同时，每一次转变又是建立在前一次的转变基础上，两者存在一定的联系。因此，程序设计与分析、设计阶段有一定的关系，是对分析、设计阶段成果的一种具体实现，这种实现局限于特殊环境和特殊语言，因此涉及环境和语言的特性。对于具体的环境和语言，同一个问题可以有不同的实现策略，反映不同的思维。可见，程序设计本身也会涉及思维问题，相对于分析、设计阶段而言，程序涉及的思维重心集中在最终系统的具体实现，如技术、语言、平台等，而与原始问题域的联系相对减弱。不同的思维重心反映了看待问题的不同抽象层次。尽管思维重心不同，但本质上仍然是人类的思维反映，可以理解为人类思维模式在不同认识域中的具体映射。

综上所述，程序设计与软件设计既有联系，又有区别。正确地理解两者之间的关系，对程序设计的学习具有指导意义。

### 1.1.2 程序设计的内涵

程序设计既是一种技术，又是一门文化，不同的文化产生不同的思维和行为模式，从而影响程序设计的发展。

从文化内涵来看，以逻辑为核心的西方刚性文化与以思辨为核心的东方柔性文化有着截然不同的内涵，西方思维的基本特征是演绎思维，而东方思维的基本特征是归纳思维。前者强调整体概念框架，由此演绎各种具体技术。后者则强调认识，通过各种技术归纳或建立整体的概念。计算机诞生于西方，自然地，程序设计以及由此衍生的各种软件工具、相关教材等，都明显地带有西方文化的痕迹，体现其思维特征。目前，所有的软件开发工具都提供了基于演绎的可视化设计方法，其蕴含的思想实质是西方的思维方式，即要求人们只要学会使用，按照设计者的实际思路做即可。他们认为这样做是顺理成章的事情，就应该这样，没有为什么。也就是说，在两个人配合默契、相互了解的前提下，每个人应该干什么、说什么，一切都在不言之中，都是自然的。然而，对于东方文化熏陶下的人而言，深层次的应用却

并不简单。首先要接受西方文化的思维模式，或者了解各种技术、工具的产生背景，然后才能深入理解技术、工具本身，进而对其灵活运用，最后融入东方文化内涵，创造性地进行技术、工具的扩展与应用。

事实上，尽管东方文化的思维在形式化方面比较薄弱，但却有较高的感悟能力。形式逻辑是按部就班的思维特征，只要知道是什么(What)；而辩证逻辑要求的是原理理解，不仅需要知道是什么，更要知道为什么(Why)，只有知道为什么，才能融会贯通地知道是什么。可见归纳思维解决的是认识本体问题，而演绎思维解决的是在假设前提下的一致问题。

从本质上讲，具有东方文化基础对于程序设计应该是有益的，这也就是中国学生(在此是指已经熟悉了西方思维的学生)能力比较强的原因。然而中国的软件产业仍然十分落后，IT人才的流动特别频繁，这些现象的产生固然有多种原因，但作者认为，深层的原因在于目前的程序设计课程体系：知识定位于语言和部分算法，沿用的是西方文化的逻辑体系，没有充分挖掘东方文化及其思想内涵。由此而得的教学思想、策略等，制约了东方文化对程序设计的作用，延长了程序设计学习的时间，甚至可能导致学习方法、思维方法的畸形发展，从而制约程序设计能力的提高乃至软件产业的发展。

另一方面，尽管每个人的思想不同，但人类的认识活动存在着一定的规律，而与某个具体个体的认识活动的关系可以看作是普遍性与特殊性之间的辩证关系。普遍性是指共性和规律，可以映射成为程序设计活动中的各种规律、经验模式，是从大量程序设计个体活动中抽象出来的、有意义的、通用的问题处理方法和思想。特殊性是指在普遍性指导下，个体认识活动的具体展开过程。它是根据共性和规律，对具体的问题进行灵活应用的过程和能力。特殊性可以映射为每一个程序员的程序设计过程，该设计过程显然应该在普遍性原则指导下进行。

因此程序设计的本质在于对两种文化及其思维框架的认识，从认识论的高度去理解程序设计。正确理解两种文化及其思维框架，以及由此诞生的各种技术和工具，并注重捕捉设计活动中的经验规律和应用模式，是掌握程序设计精髓的关键。

图1-1演绎了程序设计的内涵。从各种技术、工具出发，理解其背后隐藏的技术思想和原理，更进一步地由各种技术思想感悟其深厚的文化内涵及其思维逻辑，从而深刻领会程序设计的丰富内涵，即编程之道。如此，可以降低学习新思想、新技术、新工具的成本，笑傲瞬息万变的IT世界，同时完成“知道—体道—得道”的过程。正如老子在《道德经》中所说：“为学日益，为道日损，损之又损，以至于无为，无为而无不为。”

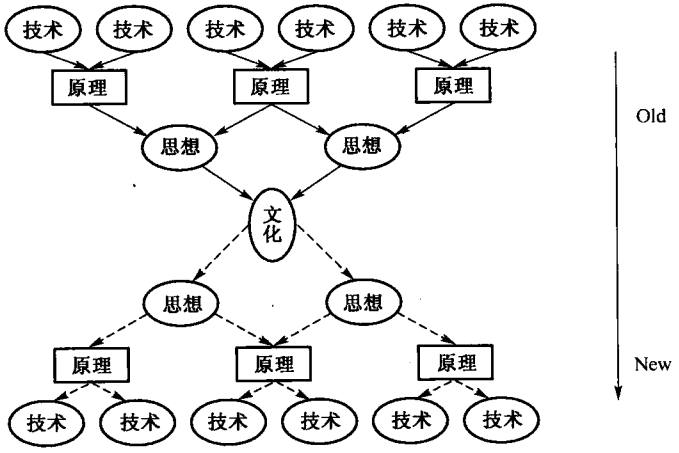



**图 1-1　程序设计的内涵**

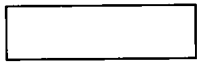
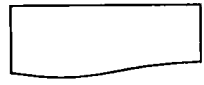
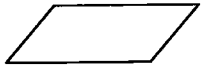
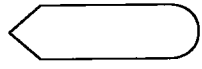
### 1.1.3　程序设计的表示方法

程序设计的表示方法可以分为图形(程序流程图、盒图、问题分析图)、表格(判定表)和语言(过程设计语言)。不论是哪种工具,对它们的基本要求都是:能提供对设计的无歧义的描述,也就是应该能指明控制流程、处理功能、数据组织以及其他方面的实现细节,从而能把对设计的描述直接翻译成程序代码。下面对程序流程图、判定表和过程设计语言进行简单介绍。

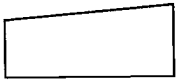
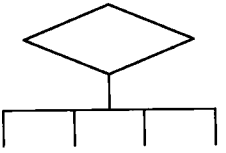
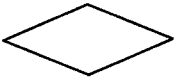
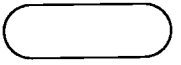
1) 程序流程图

程序流程图又称为程序框图,它是历史最悠久、适用最广泛的描述程序设计的方法。绘制流程图的常用符号如表 1-1 所示。

**表 1-1　绘制程序流程图的常用符号**

| 符　号 | 名　称 | 符　号 | 名　称 |
|---|---|---|---|
|  | 处理 |  | 文档 |
|  | 输入/输出 |  | 磁盘 |
|  | 数据流 |  | 显示 |

（续　表）

| 符　号 | 名　称 | 符　号 | 名　称 |
| --- | --- | --- | --- |
|  | 人工输入 |  | 多分支 |
|  | 选择(分支) |  | 终止 |

下面给出一个简单的使用程序流程图的例子。

某装配厂有一座存放零件的仓库，仓库中现有的各种零件的数量以及每种零件的库存量临界值等数据记录在库存清单主文件中。当仓库中零件数量有变化时，应该及时修改库存清单主文件，如果哪种零件的库存量少于库存量的临界值，则应该报告给采购部门以便订货，规定每天向采购部门送一次订货报告。

该装配厂使用一台小型计算机处理更新库存清单主文件和产生订货报告的任务。零件库存量的每一个变化称为一个事务，通过放在仓库中的终端输入到计算机中；系统中的库存清单程序对事务进行处理，更新存储在磁盘上的库存清单主文件，并且把必要的订货信息写在磁盘上。最后，每天由报告程序读一次磁盘，并且打印出订货报告。

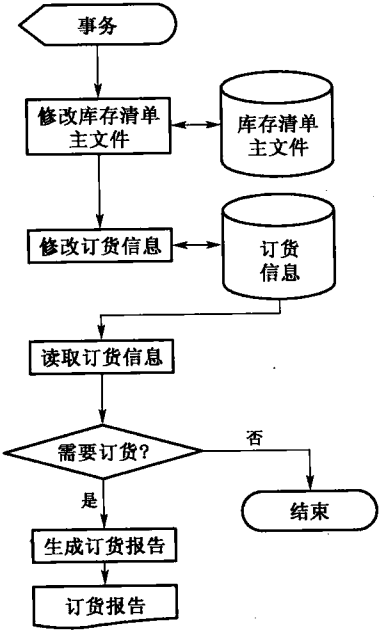



图 1-2　库存清单系统的程序流程图

2）判定表

当算法中包含多重嵌套的条件选择时，用程序流程图不易描述清楚，但判定表却能够清晰地表示复杂的条件组合与应做的动作之间的对应关系。

一张判定表由四部分组成：左上部分列出所有条件；左下部分是所有可能的动作；右上部分是表示各种条件组合的一个矩阵；右下部分是和每种条件组合对应的动作。判定表右半部分的每一列实质上是一条规则，规定了与特定的条件组合相对应的动作。

下面以行李托运费的算法为例说明判定表的使用方法。

假设某航空公司规定，乘客可以免费托运质量不超过 30 kg 的行李。当行李的质量超过 30 kg 时，对头等舱的国内乘客超重部分每 kg 收费 4 元，对其他舱的国内乘客超重部分每 kg 收费 6 元，对外国乘客超重部分每 kg 收费比国内乘客多一倍，对残疾乘客超重部分每 kg 收费比正常乘客少一半。用判定表来表示上述的规则，如图 1-3 所示。

| | 规则 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 国内乘客 | | T | T | T | T | F | F | F | F |
| 头等舱 | | T | F | T | F | T | F | T | F |
| 残疾乘客 | | F | F | T | T | F | F | T | T |
| 行李质量 W<30 或 W=30 | T | F | F | F | F | F | F | F | F |
| 免　费 | × | | | | | | | | |
| (W−30) * 2 | | | | × | | | | | |
| (W−30) * 3 | | | | | × | | | | |
| (W−30) * 4 | | × | | | | | | × | |
| (W−30) * 6 | | | × | | | | | | × |
| (W−30) * 8 | | | | | | × | | | |
| (W−30) * 12 | | | | | | | × | | |

**图 1-3　用判定表表示计算行李费的算法**

在表的右上部分中，“T”表示它左边的那个条件成立，“F”表示条件不成立，空白表示这个条件的成立与否并不影响对动作的选择。判定表右下部分中画“×”表示做它左边的动作，空白表示不做这个动作。

3）过程设计语言(PDL)

PDL 也称为伪码，这是一个笼统的名称，现在有很多种不同的过程设计语言在使用。它是用正文形式表示数据和处理过程的设计工具。下面描述一种类 Pascal 语言的 PDL 的语法规则。

在伪码中，每一条指令占一行(else if 除外)，指令后不跟任何符号，这和 Pascal 和 C 语言中语句要以分号结尾有所不同。

书写上的“缩进”表示程序中的分支程序结构。这种缩进风格也适用于 if-then-else 语句。用“缩进”取代传统 Pascal 中的 begin 和 end 语句来表示程序的块结构可以大大提高代码的清晰性。同一模块的语句有相同的“缩进”量，次一级模

块的语句相对于其父级语句又有一个“缩进”量。

例如：

```
line 1
line 2
  sub line 1
  sub line 2
    sub sub line 1
    sub sub line 2
  sub line 3
line 3
```

而在 Pascal 中这种关系用 begin 和 end 的嵌套来表示：

```
line 1
line 2
  begin
    sub line 1
    sub line 2
      begin
        sub sub line 1
        sub sub line 2
      end;
    sub line 3
  end;
line 3
```

在 C 中这种关系用“{”和“}”的嵌套来表示：

```
line 1
line 2
{
  sub line 1
  sub line 2
  {
    sub sub line 1
    sub sub line 2
  }
  sub line 3
```

}

Line 3

PDL 作为一种设计工具有如下优点：

(1) 可以作为注释直接插在源程序中间。这样做能促使维护人员在修改程序代码的同时也相应地修改 PDL 注释，因此，有助于保持文档和程序的一致性，提高了文档的质量。

(2) 可以使用普通的正文编辑软件或者文字处理系统，可以很方便地完成 PDL 的书写和编辑工作。

(3) 已经有自动处理程序的存在，而且可以自动由 PDL 生成程序源代码。

PDL 的缺点是不如图形工具形象直观，描述复杂的条件组合与动作间的对应关系时，不如判定表清晰简单。

## 1.2 程序设计方法的发展

### 1.2.1 面向计算机的程序设计

人类最早的编程语言是由计算机可以直接识别的二进制指令组成的机器语言。显然机器语言便于计算机识别，但对人类来说却是晦涩难懂。这一阶段，在人类的自然语言与计算机编程语言之间存在着巨大的鸿沟，这一时期的程序设计属于面向计算机的程序设计，设计人员关注的重心是使程序尽可能地被计算机接受并按指令正确地执行，至于计算机的程序能否让人理解并不重要。软件开发的人员只能是少数的软件工程师，因此软件开发的难度大，周期长，而且开发出的软件功能也很简单，界面也不友好，计算机的应用仅限于科学计算。

随后出现了汇编语言，它将机器指令映射为一些能读懂的助记符。如 ADD、SUB 等。此时的汇编语言与人类的自然语言之间的鸿沟略有缩小，但仍与人类的思想相差甚远。因为它的抽象层次太低，程序员需要考虑大量的机器细节。此时的程序设计仍很注重计算机的硬件系统，它仍属于面向计算机的程序设计。面向计算机的程序设计的基本思想可归纳为：注重机器、逐一执行。

### 1.2.2 面向过程的程序设计

随着计算机应用范围的扩大，人们感觉到机器语言和汇编语言的不足，机器语言太注重计算机的硬件，而汇编语言也不太适合人类的思维习惯。这时设计了更接近人类思维习惯的高级语言。它撇开了计算机的硬件细节，提高了语言的抽象层次，程序中可采用具有一定含义的数据命名和容易理解的执行语句。这使得在

写程序时可以联系到程序所描述的具体事物。上世纪 60 年代末开始出现的结构化程序设计的思想便是面向过程的程序设计思想的集中表现。

结构化程序设计的思想是：自顶向下、逐步求精。其程序结构是按功能划分为若干个基本模块(基本程序)，这些模块形成一个树状结构，各模块之间的关系尽可能简单，在功能上相对独立：每一个模块内部均是由顺序、条件、循环三种基本结构组成，其模块化实现的具体方法是使用子程序。结构化程序设计由于采用了模块分化与功能分解，自顶向下、分而治之的方法，因而可将一个较复杂的问题分解为若干个子问题，各子问题分别由不同的人员解决，从而提高了速度，并且便于程序的调试，有利于软件的开发和维护。

结构化程序设计的自顶向下、逐步求精的思想可用图 1-4 中的树状结构表示(其中 P 表示程序，$P_1$、$P_2$、$P_3$ 等表示子程序，以此类推，$P_{311}$、$P_{312}$等表示基本程序)。

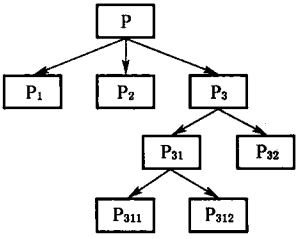



**图 1-4　结构化程序的设计思想示意图**

结构化程序设计思想的核心是功能的分解。当程序员用 C 或 Pascal 语言来设计程序解决一个实际问题时，首先要做的工作就是将一个问题按功能的不同分解成若干个模块，然后根据模块的功能来设计一系列用于存储数据的数据结构，最后编写一些过程(或函数)对这些数据进行操作。最终的程序就是由这些数据和操作构成的。显然，这种方法将数据结构和操作过程作为两个独立的实体来对待，设计人员编程之前首先考虑如何将功能分解，在每个过程中又要着重安排程序的操作序列，并且程序员在编程的同时又必须时时考虑数据结构，因为毕竟要将操作作用在数据上。不仅如此，程序员在编程过程中，不能保证数据结构始终没有变化，而一旦数据结构发生变化，作用在数据上的操作必然会相应地发生变化，这样给软件开发人员造成了沉重的负担。

我们知道，客观世界中的问题是错综复杂和不断变化的，软件开发人员开发的软件往往不是一成不变的。随着社会的发展，用户对软件提出了更多的要求，因此软件的更新日益加快。而面向过程的程序设计中，由于数据与操作的分离，使程序的可重用性差，维护代价高，不便于程序的更新换代。为了克服这一缺点，人们提出了面向对象的程序设计思想。

### 1.2.3　面向对象的程序设计

面向对象的程序设计思想是：注重对象、抽象成类。在程序系统中，将客观世界中的事物看成对象，对象是由数据及对数据的操作构成的一个不可分离的整体。对同类型的对象抽象出其共性，形成类。类中大多数数据，只能用本类的方