第 3 章 结构化设计

需求分析阶段结束后，系统必须“做什么”的结论已经明确了，下一步就要考虑如何实现系统的需求。如果系统比较简单，要求一经确定就可以立即开始编程序。但对于大型软件系统来说，为了保证产品的质量，提高软件的开发效率，必须先制订系统设计方案，确定软件结构，然后根据系统的特点选择适当的设计方法，而不必急于进入编写程序的阶段。

传统的软件工程方法学采用结构化设计（Structured Design，SD）完成软件设计工作。结构化设计分为概要设计和详细设计两个过程。

结构化设计（SD）的基本要点：

- 软件系统由层次结构的模块构成；
- 模块是单入口、单出口的；
- 模块构造和联结的基本准则是模块独立；
- 软件系统结构用图来描述。

本章重点：

- 软件结构设计；
- 过程设计工具；
- 人机界面设计。

3.1 软件设计步骤

软件设计工作分为概要设计和详细设计两个阶段。概要设计也称为总体设计，概要设计过程通常有确定设计方案和结构设计两个阶段，与此同时要进行数据库设计和制订测试计划。详细设计的任务是软件过程设计、系统接口设计和数据设计。

本节先介绍软件设计步骤，使读者对软件设计过程有一个初步的了解，后面章节再详细介绍其主要步骤的设计原理、方法和规则等。

3.1.1 概要设计步骤

概要设计阶段的主要任务是确定设计方案和软件结构设计。在概要设计阶段，还要在需求分析阶段的基础上进行数据文件设计，制订测试计划，制订出详细的软件工程进度计划，修订用户手册。

1．确定设计方案

系统结构设计是非常重要的，要经过系统分析员的仔细研究，并且要经过用户负责人

的批准才能确定。一般先由系统分析员设计出供选择的方案，并推荐其中最佳的实现方案，再由用户确定系统设计方案。

1）设计供选择的方案

需求分析阶段得出的数据流图是总体设计的出发点。把数据流图中的处理逻辑地进行组合，不同的组合可能就是不同的实现方案。分析各种方案，首先抛弃行不通的方案，然后提供各个合理方案的以下几方面资料：

- 数据流程图、IPO 图等；
- 组成系统的元素清单、数据字典；
- 成本/效益分析；
- 实现该系统的进度计划。

成本/效益分析方法将在 9.2 节进行介绍。一般应提供低成本、中成本、高成本的不同方案供用户选择。

进度计划可参考曾经实现的相当规模软件系统的计划执行情况来估计，在软件工程的后面几个阶段再进行适当调整。

2）推荐最佳实现方案

系统分析员应比较各个合理方案的利弊，选择一个最佳方案向用户推荐，并为所推荐的方案制订详细的实现计划。

用户和有关专家应认真审查分析员所提供的几种方案，如果确认某方案为最佳方案，且在现有条件下完全能实现，则应提请用户进一步审核。在使用单位的负责人审批接受了分析员所推荐的方案后，方可进入软件工程的下一步，即软件结构设计阶段。

2．软件结构设计

为了实现目标系统，必须设计出这个系统的所有程序和数据文件。对于大型系统，通常先进行结构设计，然后再进行详细设计。在概要设计阶段进行结构设计，确定系统由哪些模块组成，并确定模块之间的相互关系。在详细设计阶段确定每个模块的处理过程。

1）功能分解

为进行结构设计，首先把复杂的功能进一步分解为一系列比较简单的功能，此时数据流图和 IPO 图也可进一步细化。通常一个模块完成一个适当的子功能。

2）设计软件结构

分析员应把模块组织成层次结构，顶层模块能调用它的下一层模块，下一层模块再调用其下层模块，如此依次向下调用，最底层的模块能完成某项具体的功能。

3.2 节将介绍软件结构设计的基本原理、模块分割方法和模块设计准则等。软件的结构可用层次图或结构图来描绘。

3．数据文件设计

需求分析阶段所绘制的 E-R 图和数据字典是数据文件设计的依据。软件系统中常用数据文件存放数据，供系统中各模块共享或与系统外部通信时使用。数据文件设计主要是数据结构设计。对于管理信息系统，通常都用数据库来存放数据。分析员在需求分析阶段

对系统的数据要求做分析的基础上进行数据文件设计，主要是进行数据代码设计和数据库设计。

进行数据库设计时首先要确定数据库结构，还需要考虑数据库的完整性、安全性、一致性及优化等问题。数据库设计是计算机管理信息系统的一个重要阶段，应在数据库课程中介绍，本书不再赘述，这里只介绍有关数据代码设计的原则、分类和方法等。

4．制订测试计划

在软件开发的设计阶段提前考虑软件测试方案，有利于提高软件的可测试性。测试计划包括测试策略、测试方案、预期的测试结果和测试进度计划等。本书第4章将详细介绍软件测试的目标、步骤及测试方案的设计方法等。

5．书写概要设计文档

概要设计文档包括以下内容：

（1）系统说明：系统构成，成本/效益分析，对最佳方案的描述，细化的数据流图，用层次图或结构图描述的软件结构，IPO图，需求、功能和模块之间的关系等。

（2）用户手册：根据概要设计结果，修订需求分析阶段产生的初步的用户手册。

（3）测试计划。

（4）详细的软件工程进度计划。

（5）数据文件设计结果：包括代码设计和数据库设计的结果。

3.1.2 详细设计的基本任务

详细设计阶段主要进行接口设计和过程设计，同时为每个模块设计测试用例（包括模块功能、输入数据和预期的输出结果）。

1．数据结构设计和数据库设计

在概要设计的基础上，确定每个模块使用的数据结构，进一步设计软件的数据库结构。

2．接口设计

接口设计包括软件模块间的接口设计、模块与外部实体的接口设计和人机界面设计。

3．过程设计

过程设计应在系统结构设计、数据结构设计、接口设计完成之后进行，它是详细设计阶段应完成的主要任务之一。软件过程设计规定运用方法的顺序、应该交付的文档、开发软件的管理措施和各阶段任务完成的标志。

过程设计并不是具体地编写程序，而是从逻辑上设计能正确实现每个模块功能的处理过程。

过程设计可以采用面向数据流设计方法，也可以采用面向数据结构设计方法。

过程设计一般只用三种基本控制结构：顺序结构、条件选择结构和循环结构。用并且仅用这三种结构可以组成任何一个复杂的程序。过程设计就是通过顺序、选择和循环三种结构的有限次组合或嵌套，描述模块功能的实现算法。

过程设计阶段应当尽可能简明易懂地设计处理算法，以便在程序设计阶段，程序员依

据过程设计所描述的细节，可以直接而简单地编写程序代码。过程设计的结果基本上决定了程序的质量。

在进行过程设计时要描述程序的处理过程，可采用图形、表格或语言类工具。无论采用哪一类工具，都需对设计进行清晰、无歧义性的描述，应表明控制流程、系统功能、数据结构等方面的细节。过程设计可用流程图、N-S 图、PAD 图、判定表、判定树、过程设计语言（PDL）等进行描述。

4．代码设计、输入输出设计和网络设计等

代码是信息处理系统中使用的数字、文字等符号，用来代替自然语言，具有识别、分类和排序三项基本功能。代码的设计应当具有标准化、唯一性、可扩充性、简单性、规范化和可适应性。

数据输入设计需要对信息的发生、收集、介质化、输入和内容等方面进行详细的调查、研究后才能进行。同样，需要对用户的输出设备、输出介质、输出内容、输出方式（集中输出还是分散输出），以及数据通过什么途径、采用什么方式、什么周期、送给什么人等进行反复调查，然后才能进行数据的输出设计。设计时要防止数据的遗失、泄密和延误。

现在，很多软件系统之间相互连网、共享资源，因而网络设计往往是软件工程必不可少的。

5．编写详细设计说明书、软件系统的操作手册等文档

编写详细设计说明书时，可以根据实际需要包含以下内容：程序的功能、性能、输入项、输出项、算法、流程逻辑、接口、存储分配、注释设计、限制条件、测试计划、尚未解决的问题等。

在详细设计阶段，可以写出初步的用户操作手册，在程序编码阶段，再对其进行补充和修改。操作手册的内容含软件的结构、安装和初始化过程、每种可能的运行及其步骤、具体操作要求、输入输出过程、非常规过程、远程操作等。

6．复审

软件的详细设计完成以后，必须从软件的正确性和可维护性两方面对软件的逻辑结构、数据结构和人机界面等进行审查，以保证软件设计的质量。

3.2 软件结构设计

本节介绍与软件结构设计有关的基本概念、软件结构设计应遵循的基本原理及软件结构设计的启发规则等。

3.2.1 软件结构设计的基本原理

软件结构设计有以下基本原理：软件的模块化、模块独立性、抽象和逐步求精、信息隐蔽和局部化等。

1．模块

模块（Module）是能够单独命名，由边界元素限定的程序元素的序列。在软件的体系

结构中，模块能独立地完成一定的功能，是可以组合、分解和更换的单元。

模块有以下基本属性：

（1）名称。模块的名称必须表达该模块的功能，指明每次调用它时应完成的功能。模块的名称由一个动词和一个名词组成，如计算成绩总评分、计算日销售额等。

（2）接口。模块的输入和输出。

（3）功能。模块实现的功能。

（4）逻辑。模块内部如何实现功能及所需要的数据。

（5）状态。模块的调用与被调用关系。

一般地，模块从调用者那里获得输入数据，然后把产生的输出数据返回给调用者。

模块是程序的基本构件。模块是由边界元素限定的程序元素的序列。Pascal 或 Ada 这样的块结构程序设计语言中的 Begin…End，或 C、C++和 Java 语言中的{…}就是边界元素的例子。过程、函数、子程序和宏等都可作为模块。面向对象方法中的对象、对象内的方法也是模块。

模块化是指把系统分割成能完成独立功能的模块，模块独立性要求模块之间低耦合和模块内部高内聚。

2．抽象和逐步求精

抽象是人们认识复杂事物过程中经常使用的思维方式，先抽出事物本质的共同特性，而暂时不考虑它的细节，也不考虑其他因素。

逐步求精就是先抓住并解决主要问题，然后分阶段逐步深入考虑问题的细节。

人们在认识事物过程中，普遍遵守 Miller 法则：一个人在任何时候都只能把注意力集中在（7±2）个知识点上。软件设计时要考虑的问题通常不只有 7 个，因而不可能一下子解决所有问题。应当先考虑整体解决方案，以后逐步深入地考虑细节问题。

软件工程实施过程中，首先用抽象概念来理解和构造一个复杂系统，此后的每一步都可以看作是对软件抽象层次的一次细化。在软件需求分析阶段，要采取逐步求精的方法，在软件系统结构设计时，同样要采取逐步求精的方法。软件结构顶层的模块控制了系统的主要功能，然后逐步求精，逐步揭示各模块的细节，软件结构底层的模块完成对数据的一个具体处理。用自顶向下、由抽象到具体的逐步求精方法进行软件的设计和实现。

3．信息隐蔽和局部化

Parnas 提出了在模块划分时应遵循的信息隐蔽和局部化原则。

所谓信息隐蔽，是指在设计和确定模块时，使得一个模块内包含的信息（过程或数据）对于不需要这些信息的其他模块来说是不能访问的。在定义和实现模块时，通过信息隐蔽，对模块的过程细节和局部数据结构进行存取限制。这里“隐蔽”的不是模块的一切信息，而是模块的实现细节。有效的模块化通过一组相互独立的模块来实现，这些独立的模块彼此之间仅仅交换为完成系统功能所必需的信息，而将自身的实现细节与数据“隐藏”起来。

局部化就是把关系密切的软件元素放在一起。局部化有利于信息隐蔽。

一个软件系统在整个生命周期中要经过多次修改，信息隐蔽对软件系统的修改、测试

和维护都有好处，因此，在划分模块时要采取局部化措施，如采用局部数据结构，使得大多数过程（实现细节）和数据对软件的其他部分是隐藏起来的。这样，修改软件时偶然引入的错误所造成的影响只局限在一个或少量几个模块内部，不会影响其他模块，提高了软件的可维护性。

3.2.2 模块化

模块化（Modularization）是指把系统分割成能完成独立功能的模块，明确规定各模块及其输入输出规格，使模块的界面不会产生任何混乱。在软件工程中，模块化是大型软件设计的基本策略。

1．模块化的效果

（1）减少复杂性。

对复杂问题进行分割后，每个模块的信息量小，问题简单，便于对系统的理解和处理。

设函数 C(X)定义问题 X 的复杂程度，函数 E(X)确定解决问题 X 所需要的工作量（时间）。对于问题 P1 和问题 P2，如果

$$C(P1) > C(P2)$$

显然有

$$E(P1) > E(P2)$$

根据一般经验，另一个有趣的规律是

$$C(P1+P2) > C(P1)+C(P2)$$

由问题 P1 和 P2 组合而成的问题的复杂程度大于分别考虑每个问题时的复杂程度之和。

从而得到不等式

$$E(P1+P2) > E(P1)+E(P2)$$

即独立解决问题 P1 和 P2 所需的工作量比把 P1 和 P2 合起来解决所需的工作量少。

由此可见，模块化是控制复杂性的最好办法。先独立地对各部分进行分析，确定解决问题的途径的正确性，最后才对整体进行验证，是行之有效的办法，有利于提高软件的开发效率。

（2）提高软件的可靠性。

程序的错误通常出现在模块内及模块间的接口中，模块化使软件易于测试和调试，有助于提高软件开发的可靠性。

（3）提高可维护性。

软件模块化后，即使对少数模块进行大幅度的修改，由于其他模块没有变动，对整个系统的影响较小，这样可使系统能适应各种环境变化，从而及时地进行维护。

（4）有助于软件工程的组织管理。

承担各模块设计的人员可以独立地、并行地进行开发，有利于软件开发团队的任务安排，可将设计难度大的模块分配给技术熟练的程序开发员。

（5）有助于信息隐蔽。

在设计和确定模块时，尽量设法将可变性因素隐蔽在一个或几个局部模块中。具体的做法是先将可能发生变化的因素列出，然后在划分模块时将这些因素隐蔽在某几个模块

内，使其他模块与这些可变因素无关。这样可避免软件维护时错误的传递，使得所列出的因素的任何一个变化，仅影响与其相关的模块，不会影响其他模块，从而提高软件的可维护性。

2．模块分割方法

模块化的关键问题是如何分割模块和如何设计系统的模块结构。

进行模块分割时，由于人们在认识问题时遵守 Miller 法则，一次最多只能有（7±2）个知识点，因而一个模块可分为 7 个左右的子模块，不要超过 9 个子模块。模块化的过程是自顶向下、由概括到具体的过程，软件结构顶层的模块控制系统的主要功能，每个功能模块可细分为若干个子模块，软件结构底层的模块完成对数据的一个具体处理。自顶向下分析和构造软件的层次结构。

模块的分割主要根据功能的各种差异来进行。根据系统本身的特点，可采用以下几种不同的分割方法。

1）横向分割

根据输入输出等功能的不同来分割模块，如绘图软件包 AutoCAD 可划分为绘实体、文件管理、外设配置、版本转换、绘图机输出、打印机输出等不同功能模块。

2）纵向分割

根据系统对信息进行处理过程中不同的阶段来分割。

例如，例 2.3 招聘考试成绩管理系统中数据处理要逐步进行，前一步结束了后一步才可进行，前一步数据有误会影响后一步工作的数据的正确性。该系统可划分为以下几个模块：输入考生基本情况，考前处理，输入考试成绩，计算考生成绩总分，排序，录用，输出录用名单，统计考试情况。

3．模块分割顺序

模块分割顺序是先确定中心控制模块，由控制模块指示从属模块，逐次进行。把各个功能层次化、具体化，各个功能模块最好只有一个入口，一个出口。例如，图书馆管理系统的用户有图书馆工作人员和读者。图书馆工作人员又分为采购、图书编码、读者管理、借书和还书等岗位，各司其职、职责分明。读者只能查询图书信息，不能进入图书流通和读者管理等图书馆工作人员才能进入的功能模块。在进入该系统时，要根据系统用户的权限，确定允许其进入哪个模块。该系统的模块分割方法就是先确定中心控制模块，再将每个从属模块的功能逐步细化。

3.2.3 模块独立性

在软件系统模块化时，最重要的原理是模块独立性。评价模块分割好坏的标准主要有以下 4 个方面。

（1）模块大小。模块的大小和问题的复杂程度相关，如果模块太大，过于复杂，会使设计、调试、维护工作十分困难。如果模块太小，使功能意义消失，反而会使模块之间的关系增强，影响模块的独立性，从而影响整个系统结构的质量。模块的大小以模块的功能意义、复杂程度、易于理解、便于控制为标准。

（2）模块之间的联系程度（Coupling，耦合）。

（3）模块内软件元素的联系程度（Cohesion，内聚）。

（4）模块信息的隐蔽程度。

其中，衡量模块独立程度的两个定性度量标准是耦合和内聚。

1．耦合

软件结构中模块之间互相依赖的程度用耦合来度量。耦合的强弱取决于模块间接口的复杂程度，一般由模块之间的调用方式、传递信息的类型和数量来决定。在设计软件结构时应追求尽可能松散的耦合。如果系统中两个模块彼此间完全独立，不需要另一个模块就能单独地工作，则这两个模块之间耦合程度最低。事实上，一个软件系统中不可能所有模块之间都没有任何联系。

模块之间传递的信息有以下三种：

（1）数据信息：记录某种事实，一般可用名词表示，如考生成绩。

（2）描述标志信息：描述数据状态或性质，如已录用、未被录用等。

（3）控制标志信息：要求执行非正常的动作或某个功能，如显示“准考证号超范围，重新输入”。

耦合包括以下几类：

1）数据耦合

两个模块彼此间交换的信息仅仅是数据，那么这种耦合称为数据耦合。数据耦合是低耦合。例如，当某个模块的输出数据是另一个模块的输入数据时，这两个模块是数据耦合。一般两个数据耦合的模块共同完成一个任务。

2）控制耦合

两个模块之间传递的信息中有控制信息，则称这种耦合为控制耦合。有时这种控制信息是以数据形式出现的。控制耦合是中等程度的耦合，它增加了系统的复杂程度。

例如，模块 B 用于打印会计收支账目统计报表，可以是日报表、月报表、年报表等不同报表。模块 A 用于调用模块 B，调用时必须传递的信息是要打印日报表、月报表，还是年报表。若是日报表，必须指明日期；若是月报表，必须指明年、月；若是年报表，必须指明年份。这是因为在调用数据库内容时三种报表所调用的内容是不同的。日报表只调出某日的所有收支情况，然后统计总共收入多少，支出多少，收支差额多少；而月报表是统计某月内所有收支情况；年报表则是将某年中每月收支的合计情况分别打印出来并汇总成年收支情况。这里三种报表的打印格式相同，但实际工作过程不同。模块 A 和模块 B 是控制耦合，如图 3.1 所示。

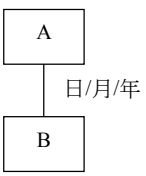


图 3.1　控制耦合

3）特征耦合

被调用的模块可以使用的数据多于它实际需要的数据，就是特征耦合。这将导致可能对数据的访问失去控制，从而给计算机犯罪提供机会。

4）公共环境耦合

两个或多个模块共享信息，这几个模块的耦合称为公共环境耦合。

这里公共环境可以是全程变量、内存的公共覆盖区、可供各个模块使用的数据文件、物理设备等。这种设计方案的复杂程度随耦合的模块个数的多少而不同，如果多个模块交叉共用大量的数据，那么设计方案是不易理解的：必须确定某个模块究竟用了哪些数据，

某个数据究竟被哪几个模块使用（必须在数据字典里加以说明）。

这种设计方案不利于修改，当需要改动某个变量名字或类型时，难以确定这一改动会影响到哪几个模块，如果没有搞清楚改动所涉及的模块范围就进行修改，极易引发潜伏的错误。

这种设计方案可靠性差，当某个模块发生错误，而引起全程变量出错时，这些错误就会扩散到使用这些全程变量的其他模块，还会通过公共数据蔓延扩散到系统的其他部分。因此，通常应限制公共环境耦合的使用。

如果两个模块共享的数据很多，通过参数传递不方便，此时可以利用公共环境耦合，但应注意共享数据的命名，使其含义明确，变量名应适当地加长一些，避免不相干的模块使用相同名字的变量，从而引起不必要的麻烦。

5）内容耦合

两个模块之间有下列情况之一时产生内容耦合：

（1）某个模块直接访问另一个模块的内部数据。

（2）两个模块有相同的程序段。

（3）一个模块直接进入另一个模块的内部。

（4）一个模块有多个入口，即模块有多个功能。

耦合程度最高的是内容耦合，应避免使用内容耦合。

总之，为了降低模块间的耦合程度，应采用以下设计原则：

（1）在传递信息时尽量使用数据耦合，少用控制耦合和特征耦合。在耦合方式上，通过语句调用，用参数传递信息，不采用直接引用方式（内容耦合），尽量控制公共环境耦合。

（2）模块之间相互调用时，传递的参数最好只有一个，最多不超过4个。

（3）在设计模块时尽量做到把模块之间的连接限制到最少，确保模块环境的任何变化都不应引起模块内部发生改变。

2．内聚

一个模块内各个元素彼此结合的紧密程度用内聚来度量。理想的模块只完成一个功能，模块设计的目标之一是实现尽可能高的内聚。

内聚和耦合是进行模块化设计时应考虑的密切相关的两个工具，模块的高内聚往往会导致模块间的松耦合，但事实证明内聚更加重要，设计时应更多地考虑如何提高模块的内聚程度。

内聚包括以下7类：

1）偶然内聚（Contingent Cohesion）

模块完成一组任务，这些任务之间关系松散，实际上没有什么联系时称为偶然内聚。假如模块A、B、C、D中都含有某些相同的语句段，程序员将A、B、C、D放在同一模块T中，共同使用相同的语句段，则A、B、C、D将成为同一模块，即模块T的几个成分，这样就形成了偶然内聚的模块T。

2）逻辑内聚（Logical Cohesion）

将逻辑上相同或相似的一类任务放在同一模块中，称为逻辑内聚。例如，对某数据库

中的数据可以按各种条件进行查询，这些不同的查询条件所用的查询方式也不相同，设计时将不同条件的查询放在同一个“查询”模块中，这就是逻辑内聚。

3）时间内聚（Temporal Cohesion）

将需要同时执行的成分放在同一模块中，称为时间内聚。

例如财务软件中，“年终结算”就是在年终时需要做的一系列任务，如第四季度结算、年结算、年底经费结余额转入下一年度的“经费来源”、下一年度的“支出”取初始值为零等，把这些任务放在同一模块中。

以上三种内聚的模块内部各成分并没有共用数据，属于很弱的块内联系。下面两种内聚则属于中等程度的内聚。

4）过程内聚

如果一个模块内的处理元素是相关的，必须以特定次序执行，则称为过程内聚。

通过数据流程图确定模块的划分，往往得到过程内聚的模块。

5）通信内聚（Communicational Cohesion）

模块中的各成分引用共同的数据，称为通信内聚。例如模块中含有 4 个部分，这 4 个部分使用同一数据文件产生不同的报表，属于通信内聚。例如财务软件的流水账文件中含有某个月内全部收支流水账记录，利用该文件可分别产生该月上旬、中旬、下旬的三种不同统计报表，以及该月总统计报表（如图 3.2 所示）。若模块中几个部分产生同一个输出数据，则也属于通信内聚。

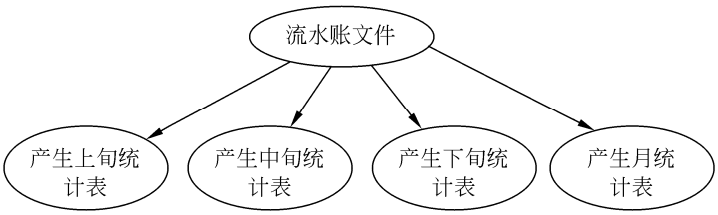


图 3.2　通信内聚

6）顺序内聚（Sequential Cohesion）

如果模块内某个成分的输出是另一成分的输入，因而这两个模块必须依次执行，则称为顺序内聚。图 3.3 中模块 A 和模块 B 都属于顺序内聚。

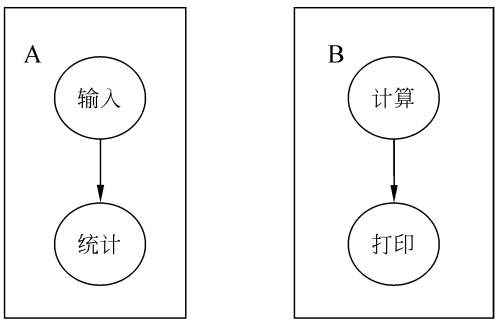


图 3.3　顺序内聚

7）功能内聚（Functional Cohesion）

一个模块内所有元素都是完成某一功能所必需的处理对象，由这些元素组成一个整体，从而完成一个特定的功能，则称为功能内聚。功能内聚是最高程度的内聚。

内聚的概念是由 Constantine、Yourdon 和 Stevens 等人提出的。按照他们的观点，把上述几种内聚按紧密程度从高到低排列次序为功能内聚、顺序内聚、通信内聚、过程内聚、时间内聚、逻辑内聚、偶然内聚，但紧密程度的增长是非线性的。偶然内聚和逻辑内聚联系松散，前面几种内聚相差不多；功能内聚最理想，一个模块完成一个功能，独立性强，内部结构紧密。

3.2.4 模块设计启发规则

长期以来人们在计算机软件开发的实践中积累了丰富的经验，总结这些经验可得出一些设计规则，这些规则往往能有助于软件设计师提高软件质量，下面介绍几条模块设计的启发规则。

1．尽力提高模块独立性

设计软件结构时，力求提高内聚、降低耦合，获得较高的模块独立性。

模块独立的重要性有以下两点：

（1）易于开发。

要使一个模块在总体设计中起到应有的作用，它必须满足一定的功能，对每一个模块的设计目标应能简明地理解该模块的功能，方便地掌握功能范围，知道模块应该做什么，这样才能有效地构造该模块。有时可以通过模块的分解或合并，减少控制信息的传递和对全程数据的引用，降低模块之间接口的复杂程度。

（2）易于测试和维护。

独立的模块修改所需的工作量较小，错误传播的范围小，要扩充功能时也较容易。

总之，模块独立是软件设计好坏的关键，也是决定软件质量的关键环节。

2．注意模块的可靠性、通用性、可维护性和简单性

（1）可靠性：模块运行应无差错，才可把它整体地加到系统中去。在设计时应先对模块做测试，使差错尽可能少。

（2）通用性：在设计模块时应尽可能使其通用化，扩大其应用性。

（3）可维护性：设计完善的模块应易于修改。可维护性在第 5 章还将进一步讨论。

（4）简单性：减少复杂性，有利于人们理解，使模块易于设计和使用。

3．模块应大小适中

模块不宜太大。根据经验，模块规模最好在一页以内（通常不超过 50 行语句），这种规模的模块易于阅读和理解。由于种种原因，一个模块可能会大于一页，只要不影响程序的清晰性，可允许大于一页，但应仔细分析一下，是否应进一步分解模块。不过不能以模块长度为绝对标准，还应根据具体情况仔细斟酌，最主要的是要使模块功能不太复杂且边界明确，模块分解不应降低模块独立性。

过小的模块有时不值得单独存在，可以把它合并到上级模块中去。模块数目过多会导

致系统接口复杂。

4．模块的深度、宽度、扇出和扇入要适当

深度指软件结构中模块的层数，如果层数过多则应考虑是否有某些模块过于简单，应适当合并。

宽度指软件结构内同一层次的模块数的最大值。一般来说，宽度越大，系统结构越复杂。

扇出指一个模块所调用的模块数。扇出太大，会使模块过于复杂，所控制的下级模块太多。扇出过小也不好，有时可把下级模块合并到上级模块中去以减少扇出。

扇入指有多少上级模块调用它，扇入大，说明共享该模块的上级模块数目多，这是有好处的，但不要违背模块独立原理而一味追求高扇入。

通常设计的软件结构，顶层扇出大，中间扇出较小，下层调用公共模块。

5．模块接口要简单和清晰

模块接口要简单和清晰，便于理解，易于实现、测试与维护。

3.3 软件结构设计的图形工具

前面已经介绍了需求分析阶段使用的一些分析工具。在进行软件结构设计时也有一些设计工具可利用。进行软件系统结构设计需描绘系统模块的层次结构，可采用层次图、HIPO图（层次图加输入/处理/输出图）和结构图。

3.3.1 层次图（或 HIPO 图）

层次图适合于描绘软件的层次结构，特别适合于在自顶向下设计时使用。

在层次图（HIPO 图）里除了顶层之外，每个方框里都加编号。编号的规律是每个处理的下层处理的编号在其上层编号后加“.”号及序号，序号可用数字，也可用英文字母表示。像这样带编号的层次图称为 HIPO（Hierarchy plus Input Process Output，层次图加输入/处理/输出）图。

在系统设计时，一般最上层的模块含有退出、输入、处理、输出、查询和系统维护模块。根据系统的具体要求，下层再将功能进一步细化。例如，查询可以用多种方式，按不同的条件进行。数据库里存放的数据可以进行插入、修改、删除等操作；系统的状态或数据可以进行初始化；处理的过程可以逐步详细描述等。一般将类似的功能放在一个模块中，不同类型的功能放在不同的模块中。

【例 3.1】 画出第 2 章例 2.4 医疗费管理系统的 HIPO 图。

如图 3.4 所示，该系统共有 4 个主要模块。数据输入（1.0）分为报销（1.A）、结算（1.B）和累加（1.C）三个子模块。系统维护（4.0）分为改医疗费限额（4.A）、初始化（4.B）和人员调动（4.C）三个功能。统计模块有 5 个功能。查询打印可以从 7 种内容中选择一种。主菜单还有“退出”系统的功能。

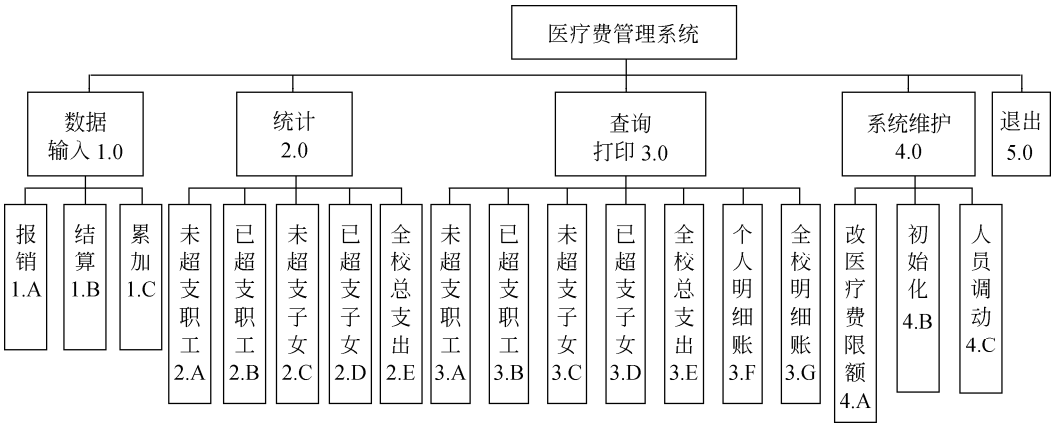


图 3.4　医疗费管理系统 HIPO 图

3.3.2　结构图

结构图和层次图相似，是用于描述软件结构的图形工具。

1974 年，美国的 W. Stevens、G. Myers 和 L. Constantine 三人联名在 *IBM System Journal*（《IBM 系统杂志》，Vol.13，No.2）上发表 *Structured Design*（《结构化设计》）的论文，第一次提出了结构系统设计的思想，指出可用一组标准的工具和准则进行系统设计。结构图（Structure Chart，SC）就是一项主要工具，用于表达系统内部各分量之间的逻辑结构和相互关系。

结构图的形态特征为深度、宽度、扇出和扇入。

1．结构图的符号

结构图的主要内容有三个：模块、模块的调用关系和模块间的信息传递。

结构图的符号主要有方框、箭头及选择结构或循环结构的框图。

（1）方框代表模块，框内注明模块的名字和主要功能。

（2）方框之间的大箭头或直线表示模块的调用关系。

（3）带注解的小箭头表示模块调用时传递的信息及其传递方向。尾部加空心圆的小箭头表示传递数据信息，尾部加实心圆的小箭头表示传递控制信息。

（4）选择结构：如图 3.5 所示，条件符合时调用模块 A，条件不符合时调用模块 B。

（5）循环结构：如图 3.5 所示，模块 H 循环调用模块 A、B、C。

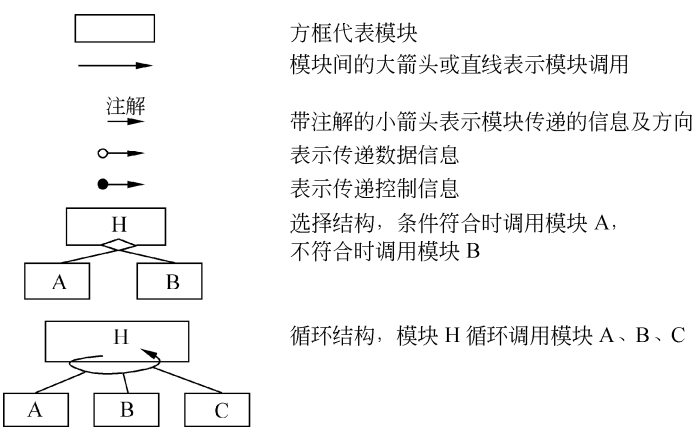


图 3.5　结构图的符号

2．绘制结构图

【例 3.2】 画出例 2.3 招聘考试成绩管理系统的初始结构图。

招聘考试成绩管理系统主要调用输入、处理和输出三个模块。该系统的初始结构图如图 3.6 所示，细化后的结构图如图 3.7 所示。

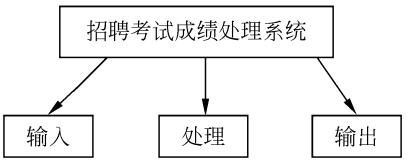


图 3.6　招聘考试成绩处理系统的初始结构图

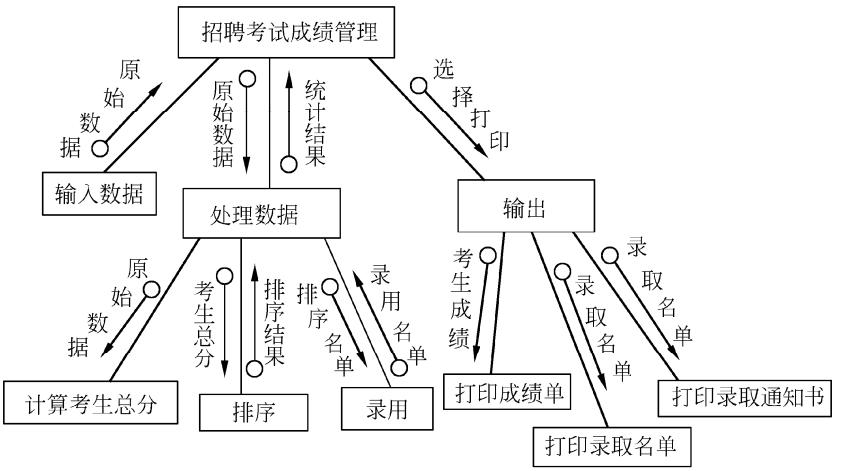


图 3.7　招聘考试成绩管理系统的结构图

结构图只描述一个模块调用哪些模块，没有描述调用次序，也没有表明模块内部的成分，通常上层模块除了调用下层模块的语句之外还可以有其他语句，结构图体现不出这种情况。

画结构图可以作为检查设计正确性和模块独立性的方法，通过检查数据传递情况，分析数据传递是否齐全，是否正确，是否有不必要的数据传递；还可分析模块分解或合并的合理性，以便选用最佳方案。

3.4　面向数据流的设计方法

过程设计不是具体地编写程序，而是从逻辑上设计正确实现每个模块功能的处理过程，过程设计应当尽可能简明易懂。传统方法采用结构化设计方法进行过程设计。

结构化设计是国际上应用最广，技术上也较完善的系统设计方法。SD 方法是由 L.L.Constantine 和 E.Yourdon 等人提出的，基于面向数据流的设计方法（Data Flow-Oriented Design）。数据流是软件开发者分析设计的基础，在需求分析（SA）阶段用数据流程图（Data

Flow Diagram，DFD）来描述数据从系统的输入端到输出端所经历的一系列变换或处理，在系统设计阶段要将 DFD 图表示的系统逻辑模型转化为软件结构设计的描述，可用结构图（SC 图）描述。这就是包括 SA 与 SD 在内的基于数据流的系统设计方法。结构化设计主张采用自顶向下、逐步求精的设计方法。SD 对系统的功能进行逐步的分解，能将复杂问题逐步分解为容易理解的较小问题。这些较小的问题具有良好的边界（目标和接口）定义。

结构化设计方法的步骤如下：

（1）对 DFD 图进行复审，必要时修改或细化。

（2）根据 DFD 图确定软件结构是变换型，还是事务型。

（3）把 DFD 图映射成 SC 图。

（4）改进 SC 图，使设计更完善。

这里可采用下面介绍的变换型或事务型设计方法来完成上述步骤（2），根据 3.2 节介绍的模块化设计原则和设计方法来完成上述步骤（3）和步骤（4）。

把 DFD 图映射成 SC 图要先区分 DFD 图的类型是变换型还是事务型。

1．变换型

变换型设计分为以下三个步骤：

（1）对变换型数据流图，要划分出数据输入、数据输出和变换中心三个部分，在 DFD 图上用虚线标明分界线。

（2）画出初始的 SC 图，顶层是主控模块，下层（第一层）一般包括输入、输出和变换三个模块。沿数据调用线标注数据流的名称。

（3）根据 DFD 图来逐步细化分解输入、输出和变换三个过程，将 SC 图也细化和优化。根据输入、输出、变换各需要几个模块，逐步由顶向下分解，直至画出每个底层模块为止。

【例 3.3】 画出例 2.3 招聘考试成绩管理系统的结构图。

该系统初始结构图如图 3.6 所示，细化后的结构图如图 3.7 所示。

2．事务型

事务型设计分为以下三个步骤：

（1）在 DFD 图中确定事务中心、接受数据和全部处理路径三个部分。

（2）画出初始 SC 图框架，把 DFD 图的三个部分分别转换为事务控制模块、接受模块和处理模块。

（3）分解和细化接受分支和处理分支。事务中心常是各条处理路径的起点，由事务中心通往受事务中心控制的所有处理路径。向事务中心提供启动信息的路径是系统接受数据的路径，有时不止一条路径。处理路径通常有多条，每条路径的结构可以不一样，有的可能是变换型，有的则是事务型。这一步主要是分解处理路径。在结构图画出后，要进行细化和优化。模块大小要适中，模块的扇入、扇出不能过大。

【例 3.4】 画出图书馆管理系统结构图。

该系统含有图书采编、读者管理、图书流通和查询等功能。

该系统执行时，先输入一个数据，根据此数据选择执行的路径：对购入图书进行登记；图书编目调用图书采编功能；借书、还书调用流通功能；查询调用查询功能。因而该系统

属于事务型系统，如图 3.8 所示。其系统结构图如图 3.9 所示。

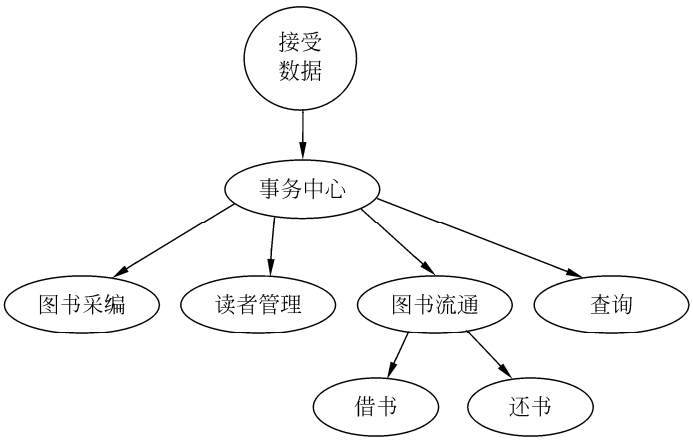


图 3.8　图书馆管理系统示意图

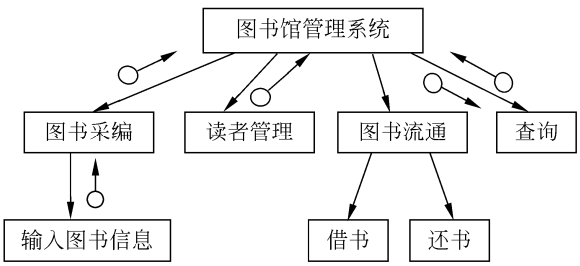


图 3.9　图书馆管理系统结构图

3.5　过程设计工具

在详细设计阶段进行过程设计时，要描述程序处理过程，可采用图形、表格、语言类工具，无论采用哪类工具，都需对设计进行清晰、无二义性的描述，应表明控制流程、系统功能、数据结构等方面的细节，以便在系统实现阶段能根据详细设计的描述直接进行编程。

本节介绍过程设计使用的工具：流程图、N-S 图、问题分析图（PAD 图）、判定表、判定树、过程设计语言（PDL）等。

3.5.1　流程图

流程图是对某一个问题的定义、分析或解法的图形表示，图中用各种符号表示操作、数据、流向及装置等。

传统的程序流程图又称为程序框图，用来描述程序设计，是历史最悠久、使用最广泛的方法。然而传统的程序流程图的一些缺点使得越来越多的人不再使用它。

传统的程序流程图的主要缺点如下：

（1）不利于逐步求精的设计。

（2）图中用箭头可随意地将控制进行转移，这是不符合结构程序设计精神的。

（3）不易表示系统中所含的数据结构。

中华人民共和国国家标准 GB/T 1526—1989《信息处理——数据流程图、程序流程图、系统流程图、程序网络图和系统资源图的文件编制符号及约定》等都采用国际标准 ISO 5807—985。根据此标准画的流程图避免了控制的随意转移。

1．流程图的分类

在国家标准 GB/T 1526—1989 中规定，流程图分为数据流程图、程序流程图、系统流程图、程序网络图和系统资源图 5 种。

1）数据流程图

数据流程图表示求解某一问题的数据通路，同时规定了处理的主要阶段和所用的各种数据媒体，包括以下几项：

- 指明数据存在的数据符号，这些数据符号也可指明该数据所使用的媒体；
- 指明对数据进行处理的处理符号，这些符号也指明该处理所用到的机器功能；
- 指明几个处理和（或）数据媒体之间数据流的流线符号；
- 便于读、写数据流程图的特殊符号。

在处理符号的前后都应是数据符号，数据流程图以数据符号开始和结束。

2）程序流程图

程序流程图表示程序中的操作顺序，包括以下几项：

- 指明实际处理操作的处理符号，它包括根据逻辑条件确定要执行路径的符号；
- 指明控制流的流线符号；
- 便于读、写程序流程图的特殊符号。

3）系统流程图

系统流程图表示系统的操作控制和数据流，包括以下几项：

- 指明数据存在的数据符号，这些数据符号也指明该数据所使用的媒体；
- 定义要执行的逻辑路径及指明对数据执行的操作的处理符号；
- 指明各处理和（或）数据媒体间数据流向的流线符号；
- 便于读、写系统流程图的特殊符号。

4）程序网络图

程序网络图表示程序激活路径和程序与相关数据流的相互作用。在系统流程图中，一个程序可能在多个控制流中出现，但在程序网络图中，每个程序仅出现一次。

程序网络图包括以下几项：

- 指明数据存在的数据符号；
- 指明对数据执行操作的处理符号；
- 表明各处理的激活和处理与数据间流向的流线符号；
- 便于读、写程序网络图的特殊符号。

5）系统资源图

系统资源图表示适合于一个问题或一组问题求解的数据单元和处理单元的配置，包括

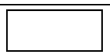
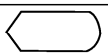
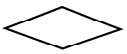
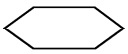
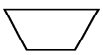
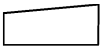
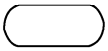
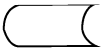
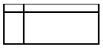
以下几项：

- 表明输入、输出或存储设备的数据符号；
- 表示处理器（中央处理机、通道等）的处理符号；
- 表示数据设备和处理器间的数据传送，以及处理器之间的控制传送的流线符号；
- 便于读、写系统资源图的特殊符号。

2．流程图符号

国家标准 GB/T 1526—1989《信息处理——数据流程图、程序流程图、系统流程图、程序网络图和系统资源图的文件编制符号及约定》中的信息处理流程图符号如表 3.1 所示。

表 3.1　信息处理流程图符号

符　号	名　称	符　号	名　称
	处理		显示
	判断		循环开始
	准备		循环结束
	人工操作		连接符
	人工输入		端点
	数据符号	……	省略符
	数据存储	——	流线
	文件		内存储器

3．流程图使用约定

（1）符号的用途是用图形来标识它所表示的功能，而不考虑符号内的内容。

（2）图中各符号均匀地分配空间，连线应保持合理长度，尽量少用长线。

（3）在符号内列出说明性文字时，不要改变符号的角度和形状，尽可能统一各种符号的大小。

（4）应把理解某符号功能所需的最少量的说明文字置于符号内。应按从左到右和自上而下的方式来书写，与流向无关。若说明文字太多，可使用一个注解符。

如果使用注解会干扰或破坏图形流程，应将正文写在另外一页上，并注明引用符号。

（5）符号标识符为赋予某个符号的标识符，其作用是便于其他文件引用该符号。符号标识符要写在符号的左上角。

（6）符号描述符用于交叉引用，表示一个符号的特定用途或进一步理解某个图形符号的功能。符号描述符要写在符号的右上角。在系统流程图中，一个描述数据媒体的符号在很多情况下既可表示输出媒体，又可表示输入媒体。表示输出媒体符号的流程图说明性文字要写在符号的右上角；表示输入媒体符号的流程图说明性文字要写在符号的右下角。

（7）分支符号如图 3.10 所示，多分支符号如图 3.11 所示。每个出口应加标识符，以反映其逻辑通路。

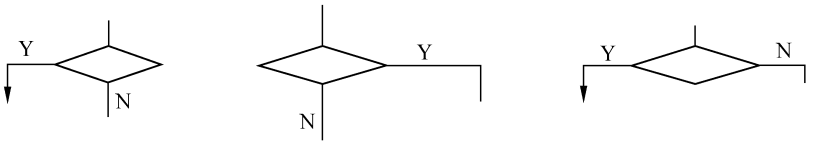


图 3.10　分支符号

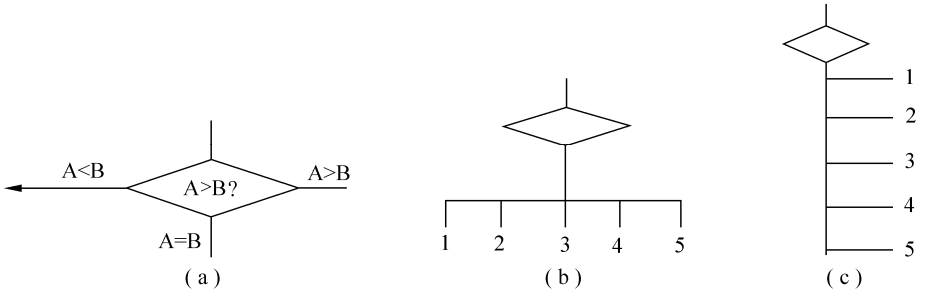


图 3.11　多分支符号

（8）流线。

- 流线可以表示数据流或控制流。流向一般从左到右，自上而下。否则要用箭头表示流向，无论何时都可以用箭头指示流程方向；
- 流线的交叉：应当尽量避免流线的交叉。即使出现流线交叉，也不表示它们有逻辑上的关系，不对流向产生任何影响；
- 两根或更多的进入流线可以汇集成一根输出线，各连接点应相互错开以提高表述清晰度，并在必要时使用箭头表示流向。

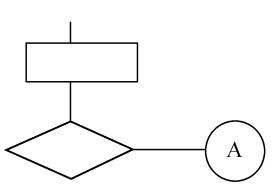
（9）连接符号。

在出口连接符号中和与它相对应的入口连接符号中应记入相同的文字、数字或名称等识别符号表示衔接（如图 3.12 所示）。

图 3.12 是流程图的一部分，图 3.12（a）中有一个分支 A，而分支 A 的详细流程在图 3.12（b）中画出。

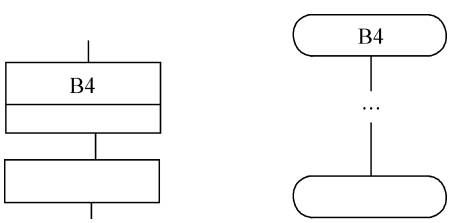
（10）详细表示线。

在处理符、数据符或其他符号中画一横线，表示该符号在同一文件集的其他地方有更详细的表示。横线加在图形符号内靠顶端处，并在横线上方写上详细表示的标识符。详细表示处始末均应有端点符号，始端写上与加横线符号相同的标识符。图 3.13（a）所示流程图的 B4 中加有横线，表示该处理另有详细描述的流程图。B4 的详细流程图如图 3.13（b）所示，其开始和结束都有端点符号，始端标有 B4。



（a）流程图中有一分支 A　（b）分支 A 的流程图

图 3.12　流程图连接符号



（a）流程图中 B4 有一横线　（b）B4 的详细流程图

图 3.13　流程图符号加横线

（11）同类介质重复使用的表示。

同一符号按同一方向的多次重复依次写上序号，顺序一律从前往后，方向可向右上、右下、左上、左下，但使用介质的优先顺序不因此而改变，如图 3.14 所示。

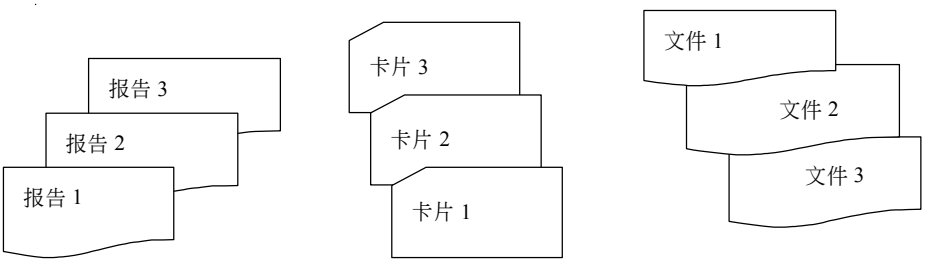
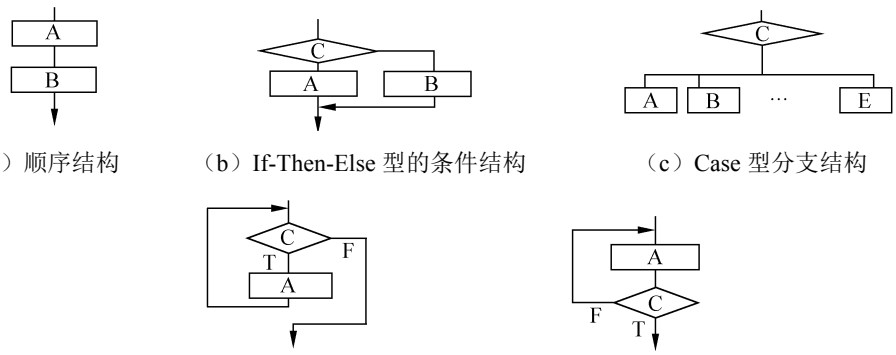


图 3.14　同类介质重复使用

4．流程图的三种基本结构

图 3.15 是用流程图表示的三种基本结构（顺序结构、条件选择结构、循环结构），图 3.15 中的 C 表示判定条件。图 3.15（a）是顺序结构；图 3.15（b）是 If-Then-Else 型的条件结构；图 3.15（c）是 Case 型分支结构；图 3.15（d）是先判断结束条件的 While 型循环结构；图 3.15（e）表示后判断结束条件的 Repeat-Until 型循环结构。



（a）顺序结构　（b）If-Then-Else 型的条件结构　（c）Case 型分支结构

（d）While 型循环结构　（e）Repeat-Until 型循环结构

图 3.15　流程图的三种基本结构

为了克服流程图随意转移控制和不利于结构化的缺陷，在画流程图时，只用以上三种基本控制结构进行组合或完整的嵌套，不要出现基本结构相互交叉的情况，以此保证流程图是结构化的。

3.5.2　盒图

盒图是 Nassi 和 Shneiderman 提出的，又称为 N-S 图。盒图没有箭头，不允许随意转移，只允许程序员用结构化设计方法来思考问题、解决问题。

1．盒图的符号

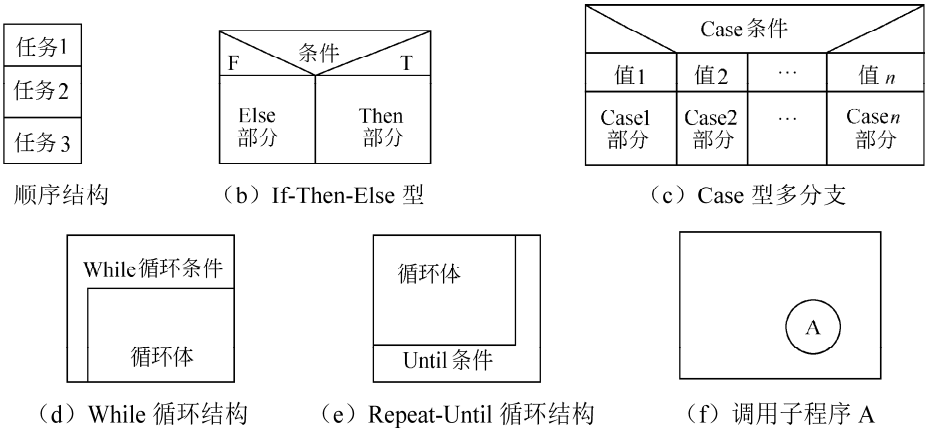
（1）顺序结构，如图 3.16（a）所示。

（2）If-Then-Else 型分支，如图 3.16（b）所示。

（3）Case 型多分支，如图 3.16（c）所示。

（4）循环结构有 While 型和 Repeat-Until 型两种，如图 3.16（d）和图 3.16（e）所示。

（5）调用子程序 A，如图 3.16（f）所示。



（a）顺序结构　（b）If-Then-Else 型　（c）Case 型多分支

（d）While 循环结构　（e）Repeat-Until 循环结构　（f）调用子程序 A

图 3.16　盒图的符号

2．N-S 图的特点

（1）清晰地描述功能域。

（2）不允许任意转移控制，因而只能表示结构化设计结构。

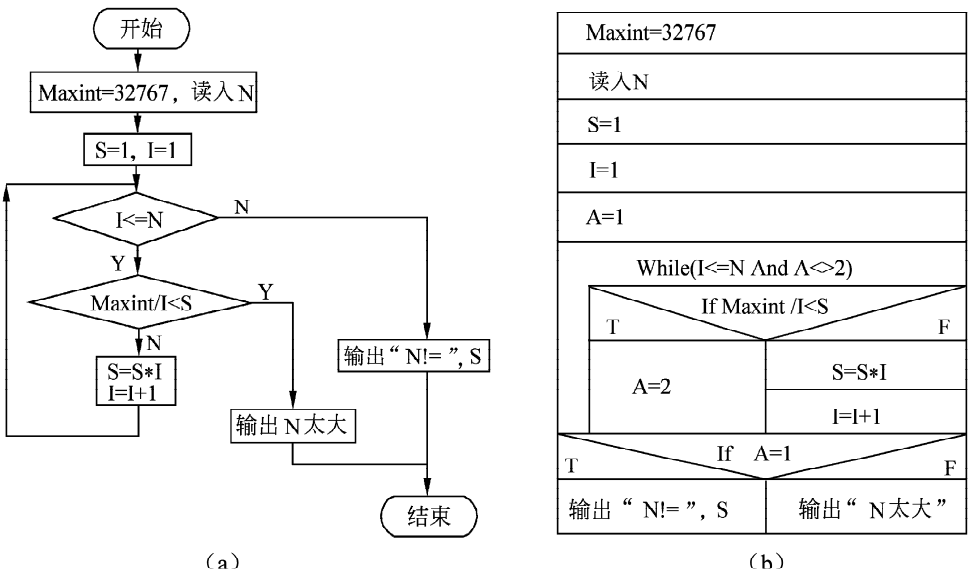
（3）易于确定数据的作用域是全局量还是局部量。

（4）易于描述系统的层次结构和嵌套关系。

【例 3.5】 将下述含有 Goto 语句的程序流程图改为 N-S 图。

该问题的程序流程图如图 3.17（a）所示。

Maxint/I 小于 S 时，能跳出循环，可以设置一个标记值 A=1。在上述条件不符合时，标记值不变，循环正常进行，直至算出 N!；满足该条件时，标记值变为 A=2，循环立即结束。该问题的 N-S 图如图 3.17（b）所示。



（a）　（b）

图 3.17　程序流程图改为 N-S 图

【例 3.6】 例 2.3 招聘考试成绩管理系统的 N-S 图。

画出招聘考试成绩管理系统的 N-S 图，如图 3.18 所示。

图 3.18 中，SUM1 是考生人数，SUM2 是录用人数。

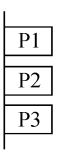
图 3.18　例 2.3 招聘考试成绩管理系统的 N-S 图

3.5.3　PAD 图

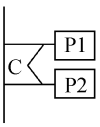
PAD（Problem Analysis Diagram，问题分析图）图是日本日立公司于 1973 年发明的。按照中华人民共和国国家标准 GB/T 13502—1992《信息处理——程序构造及其表示的约定》的规定，PAD 图的符号和特点如下。

1．PAD 图的基本符号

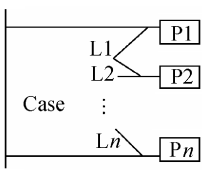
PAD 图的基本符号如图 3.19 所示。



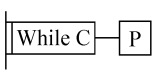
（a）顺序结构（以 P1、P2、P3 的顺序执行）



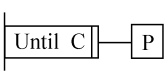
（b）选择结构（If C Then P1 Else P2）



（c）Case 型多分支结构



（d）先检测 While 型循环



（e）后检测 Until 型循环

（f）语句标号

（g）定义

图 3.19　PAD 图的基本符号

2．PAD 图的特点

（1）用PAD 图表示的程序从最左边竖线的上端开始执行，自上而下、自左向右执行。

（2）用 PAD 符号设计的过程必然是结构化的程序结构。

（3）结构清晰、层次分明。

（4）既可表示程序逻辑，也可用于描绘数据结构。

（5）用 def 逐步详细描述，可支持自顶向下、逐步求精的设计方法。

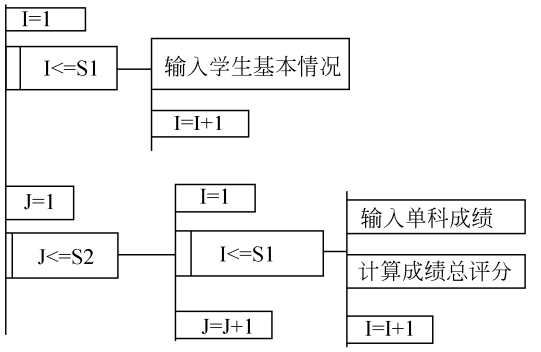


图 3.20　学生成绩管理系统的 PAD 图

PAD 图为常用高级程序设计语言的各种控制语句都提供了对应的图形符号，显然将 PAD图转换为对应的高级语言程序是很容易的。

【例 3.7】 例 2.2 学生成绩管理系统的 PAD 图。

例 2.2 学生成绩管理系统输入数据部分的 PAD 图如图 3.20 所示，其中 S1 是班级学生人数，S2 是课程数。每门课程都要输入全班级每位学生的成绩并算出成绩总评分。

3.5.4　判定表

有一类问题，其中含有复杂的条件选择，用前面介绍的程序流程图、盒图、PAD 图和结构图等都不易表达清楚。此时，可用判定表清晰地表示复杂的条件组合与应做的工作之间的对应关系。

（1）判定表的组成。左上部列出所有条件；左下部列出所有可能要做的工作；右上部每一列表示出各种条件的一种可能组合，所有列表示条件组合的全部可能情况；右下部的每一列是和每一种条件组合所对应的应做的工作。

（2）判定表中的符号。右上部用 T 表示条件成立，用 F 表示条件不成立，空白表示条件成立与否没有影响。右下部画 X 表示在该列上边规定的条件下做该行左边列出的那项工作，空白表示不做该项工作。

判定表不适合于用做通用的设计工具，不能表示顺序结构、循环结构。

【例 3.8】 某校各种不同职称教师的课时津贴费判定表。

某校对各种不同职称的教师，根据其是本校专职教师还是外聘兼职教师，决定其讲课的课时津贴费。本校专职教师每课时津贴费：教授 80 元，副教授 60 元，讲师 50 元，助教 40 元。外聘兼职教师每课时津贴费：教授 90 元，副教授 80 元，讲师 60 元，助教 50 元。

用判定表表示，如表 3.2 所示。

表 3.2　教师课时津贴判定表

教授	T	F	F	F	T	F	F	F
副教授	F	T	F	F	F	T	F	F
讲师	F	F	T	F	F	F	T	F
助教	F	F	F	T	F	F	F	T
专职	T	T	T	T	F	F	F	F

续表

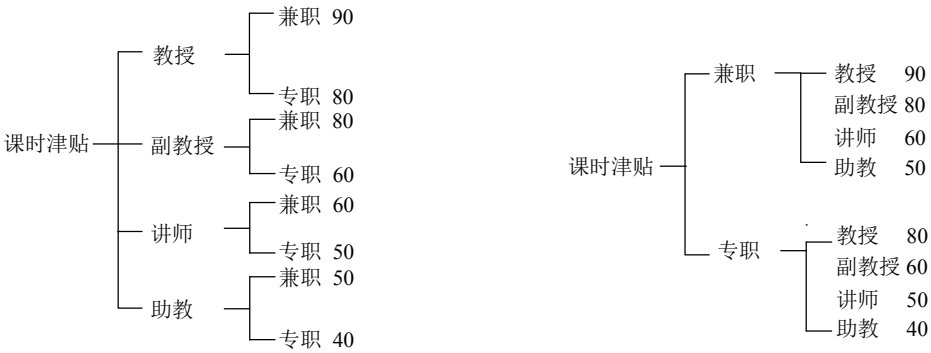
90	X
80	X X
60	X X
50	X X
40	X

3.5.5 判定树

判定表可以清晰地表示复杂的条件组合所对应的处理。判定树和判定表一样，也能表明复杂的条件组合与对应处理之间的关系。由于判定树是一种图形表示方式，更易被用户理解。

【例 3.9】 某校各种不同职称教师的课时津贴费判定树。

将例 3.8 改为用判定树表示各类教师课时津贴费，可先按职称分类，再按专职、兼职分类，如图 3.21（a）所示；也可先按专职、兼职分类，再按职称分类，如图 3.21（b）所示。



（a）先按职称分类　　（b）先按专职、兼职分类

图 3.21　教师的课时津贴费判定树

3.5.6 过程设计语言

过程设计语言（Program Design Language，PDL）也称为伪码，是一种混杂语言，混合使用叙述性说明和某种结构化程序设计语言的语法形式。

PDL 应具有下述特点：

（1）关键字应有固定语法，提供结构化的控制结构和数据说明，并在控制结构的头尾都加关键字，体现模块化的特点，如 If-EndIf、While-EndWhile 和 Case-EndCase 等。

（2）用自然语言叙述系统处理功能。

（3）具有说明各种数据结构的手段。

（4）描述模块定义和调用及模块接口模式。

PDL 作为软件设计工具与具体使用哪一种编程语言无关，但能方便地转换为程序员所选择的任意一种编程语言（转换的难易程度有所区别）。

PDL 的缺点是描述算法不如图形工具那样形象直观，在描述复杂的条件组合及对应处理之间关系时不如判定表那样清晰。

3.6 系统人机界面设计

对于交互式软件系统来说，人机界面设计是接口设计的一个组成部分。现在人机界面设计在系统软件设计中所占的比例越来越大，个别情况甚至占了总量的一半。

人机界面设计的质量直接影响用户对软件产品的评价，从而影响软件产品的竞争力和寿命，应对人机界面设计给以足够的重视。

本节对以下几个方面进行介绍：人机界面设计问题、界面设计过程、界面设计指南。

3.6.1 人机界面设计问题

人机界面设计要考虑以下问题：系统响应时间、用户帮助设施、出错信息处理、命令交互。

1．系统响应时间

从用户完成某个控制动作（按回车键或单击鼠标）到软件给出预期的响应（输出或做动作）之间的时间称为系统响应时间。响应时间有两个属性：长度和易变性。

（1）长度。响应时间过长，用户会不满意；响应时间过短，会迫使用户加快操作节奏，从而可能会犯错误。

（2）易变性。响应时间相对于平均响应时间的偏差。响应时间易变性低，有助于用户建立稳定的工作节奏。响应时间的变化暗示系统工作出现异常。

2．用户帮助设施

几乎每个用户都需要帮助，用户帮助设施可使用户不离开用户界面就可解决问题。

常见的帮助设施有两类：

（1）集成的帮助设施。集成的帮助设施设计在软件里，对用户工作内容敏感，用户可从与操作有关的主题中选择一个，请求帮助。可以缩短用户获得帮助的时间，增加界面的友好性。

（2）附加的帮助设施。附加的帮助设施实际是一种查询能力有限的联机用户手册。

集成的帮助设施优于附加的帮助设施。具体设计时必须解决以下问题：

① 提供部分功能的帮助信息还是提供全部功能的帮助信息。

② 请求方式：帮助菜单、特殊功能键、HELP 命令。

③ 显示帮助信息方式：独立窗口、指出参考某文件、在屏幕固定位置显示提示。

④ 返回正常交互方式：屏幕上的返回按钮、功能键。

⑤ 帮助信息的组织方式：平面结构（通过关键字访问）、层次结构、超文本结构。

3．出错信息处理

出错信息和警告信息是出现问题时给出的坏消息。出错信息设计得不好，会起不到作用或误导用户，增加用户的挫折感。出错信息和警告信息应具有以下属性。

（1）信息应以用户可理解的术语描述问题。

（2）信息应提供有助于从错误中恢复的建设性意见。

（3）信息应指出错误可能导致的负面后果（如破坏数据文件），以便用户检查是否出

现了这些问题，并在问题出现时予以改正。

（4）信息应伴随听觉上或视觉上的提示，在显示信息的同时发出警告声或用闪烁方式显示，或用明显的颜色表示出错信息。

（5）信息不能指责用户。

有效的出错信息能提高交互系统的质量，减少用户的挫折感。

4．命令交互

面向窗口、单击和拾取方式的界面已减少了用户对命令行的依赖，但还是有些用户偏爱使用命令方式进行交互。提供命令交互方式时，应考虑下列设计问题：

（1）每个菜单项都应有对应的命令。

（2）命令形式：控制序列（如 Ctrl + P）、功能键、输入命令。

（3）学习和记忆命令的难度，忘记了命令怎么办。

（4）用户是否可以定制或缩写命令。

（5）命令宏：代表一个常用的命令序列。只需输入命令宏的名字就可以顺序执行它所代表的全部命令。

（6）所有应用软件都应有一致的命令使用方法。

例如设计时定义宏命令 Ctrl+D。假如有的定义其含义是删除，有的定义其含义为复制，则会使用户感到困惑，并往往会导致错误。

3.6.2 人机界面设计过程

人机界面的设计过程如下：

（1）先创建设计模型，实现模型—用户界面原型。

（2）用户试用并评估该原型，向设计者反馈对界面的评价。

（3）设计者根据用户的意见修改设计并实现下一级原型。

（4）不断进行下去，直到用户感到满意为止。

3.6.3 评估界面设计的标准

评估界面设计的标准如下：

（1）系统及其界面的规格说明的长度和复杂程度，预示了用户学习使用该系统所需的工作量。

（2）命令或动作的数量、命令的平均参数个数或动作中单个操作的个数，预示了系统的交互时间和总体效率。

（3）动作、命令和系统状态的数量，预示了用户学习使用系统时需要记忆的内容的多少。

（4）界面风格、帮助设施和出错处理协议，预示了界面的复杂程度和用户对该界面的接受程度。

3.6.4 界面设计指南

人机界面设计主要依靠设计者的经验。力求设计友好、高效的人机界面。下面介绍三

类人机界面设计指南：一般交互、信息显示、数据输入。

1．一般交互

（1）保持一致性。菜单选择、命令输入、数据显示、其他功能要使用一致的格式。

（2）提供有意义的反馈。提供视觉、听觉的反馈，建立双向通信。

（3）要求确认。在执行有破坏性动作之前要求用户确认，如删除、覆盖信息、终止运行等，提示“是否确实要……”。

（4）允许取消操作。能方便地取消已完成的操作。

（5）尽量减少记忆量。

（6）提高效率。对话、移动和思考等要提高效率。尽量减少击键的次数，减少鼠标移动的距离，避免用户问“什么意思？”。

（7）允许用户犯错误。系统应保护自己不受致命错误的破坏。

（8）按功能对动作分类。如下拉菜单。应尽力提高命令和动作的内聚性。

（9）提供帮助设施。

（10）命令名要简单。用简单动词或动词短语作为命令名。

2．信息显示

（1）人机界面显示的信息应完整、清晰、易于理解。

（2）可用不同方式显示信息：用文字、图片、声音表示信息；按位置、移动、大小来显示信息；使用颜色、分辨率和省略。

（3）只显示与当前工作内容有关的信息。

（4）使用一致的标记、标准的缩写、可预知的颜色，显示的含义应非常明确。

（5）允许用户保持可视化语境。若对图形进行放缩，原始图形应一直显示着。

（6）产生有意义的出错信息。

（7）使用大小写、缩进和文本分组帮助理解。

（8）使用窗口分隔不同类型的信息。

（9）使用模拟显示方式表示信息。例如，垂直移动的矩形表示温度、压力等，用颜色变化表示警告信息等。

（10）高效率地使用显示屏。使用多窗口时，应使每个窗口都有空间显示信息。屏幕大小应选择得当。

3．数据输入

用户的绝大部分时间用在选择命令、输入数据和向系统提供输入上。

（1）尽量减少用户的输入动作。用鼠标从预定的一组输入中选择一个；然后在给定的值域中指定输入值；利用宏命令把一次击键转变为复杂的输入数据集。

（2）保持信息显示和数据输入的一致性，如文字大小、颜色、位置与输入一致。

（3）允许用户自己定义输入，如定义专用命令、警告信息和动作确认等。

（4）允许用户选择输入方式（键盘、鼠标）等。

（5）使当前不适用的命令不起作用。

（6）让用户控制交互流程。例如，跳过不必要的动作，改变工作的顺序，从错误状态

中恢复正常。

（7）对所有的输入动作提供帮助。

（8）消除冗余的输入。绝对不要要求用户提供程序可以自动获得或计算出来的信息。例如，当前日期、时间等可以自动获得；又如姓名、性别、部门和职称等信息预先存放在数据库里，只要输入职工号，就可以立即通过程序调用显示其对应的姓名、性别、部门和职称等信息；整数后面不要求用户输入“.00”。

3.7 数据代码设计

在计算机软件系统中如果存放的数据量很大，则需要对数据进行代码设计。在实际应用中常常会因为软件人员缺乏代码设计知识而出现代码设计不合理的现象。本节介绍数据代码设计的目的、代码的功能和性质、代码的种类及代码设计方法等。

3.7.1 代码设计的目的

代码设计的目的是将自然语言转换成便于计算机处理的、无二义性的形态，从而提高计算机的处理效率和操作性能。

下面介绍代码的定义、功能和代码的性质。

1．代码的定义和功能

代码是为了对数据进行识别、分类、排序等操作所使用的数字、文字或符号。

代码具有识别、分类和排序三项基本功能。尤其是在信息处理系统中，代码应用涉及的面广、量大，必须从系统的整体出发，综合考虑各方面的因素，精心设计信息代码。

在一个系统中，如果使用的代码复杂、量大，人们无法准确地记忆，可以用代码词典记录代码与数据之间的对应关系，必要时可以设计代码联机查询功能，以方便用户的使用和查找。有时代码需要随时进行增加、删除、修改或查询等，可以设计相应的代码管理功能。

2．代码的性质

代码具有简洁性、保密性、可扩充性和持久性。

（1）简洁性。代码可以减少存储空间，要求消除二义性。

（2）保密性。不了解编码规则的人不知道代码的含义。

（3）可扩充性。设计代码时要留有余地，以便在软件生命期内增加代码。

（4）持久性。代码应在软件生命期内可以长久使用。要考虑到代码的变换会影响数据库和程序。

3.7.2 代码设计的原则

代码的设计原则是标准化、唯一性、可扩充性、简单性、规范化和适应性。

- 标准化：尽可能采用国际标准、国家标准、部颁标准、行业标准或遵循惯例，以便于信息的交换和维护。如会计科目编码、身份证号码、图书资料分类编码等，要根据国家标准来编码。

- 唯一性：一个代码只代表一个信息，每个信息只有一个代码。
- 可扩充性：设计代码时要留有余地，方便代码的更新、扩充。
- 简单性：代码结构简单、尽量短，便于记忆和使用。
- 规范化：代码的结构、类型和缩写格式要统一。
- 适应性：代码要尽可能反映信息的特点，唯一地标识某些特征，如物体的形状、大小、颜色，或材料的型号、规格和透明度等。

另外，有一些实用规则可供参考：

（1）只有两个特征值的，可用逻辑值代码，如电路的闭合、断开。

（2）特征值的个数不超过 10 时，可用数字代码。

（3）特征值的个数不超过 20 时，可用字母代码。

（4）数字、字母混用时，要注意区分相似的符号。

例如：

数字	字母
1	I
0	O、D
8	B
5	S
2	Z
9	q

（5）若代码会出现颠倒使用的情况，要注意区分数字 6 和数字 9、字母 M 和字母 W。

（6）代码可能会正反使用时，要注意区分字母 p 和字母 q。

3.7.3 代码种类

在代码设计中，可用数字、符号的组合构成各种编码方式，一般分为顺序码、信息块码、归组分类码、助记码、数字式字符码和组合码等。

1．顺序码

按数字的大小或字母的前后次序排列的组合作为代码使用称为顺序码。

这是最简单的代码体系，例如在财务凭证、售票发票、银行支票等票据类数据中用顺序码表示单据号。

2．信息块码

将代码按某些规则分成几个信息块，在信息块之间留出一些备用码，每块内的码是按顺序编排的，这样编成的代码称为信息块码。

例如，中华人民共和国行政区划代码（GB2260.1995）就是典型的信息块码，其代码结构由 6 位数字组成，形式如 XXYYZZ。其中：

（1）前两位 XX 代表省、直辖市：如 11 代表北京市、12 代表天津市、31 代表上海市、32 代表江苏省等。

（2）中间两位 YY：01～20、50～70 表示省辖市，21～49 表示地、州、盟。

（3）后两位 ZZ：01～18 表示市辖区或地辖区，21～80 表示县、旗，81～99 表示省直辖县级市。如 320106 表示江苏省南京市鼓楼区。

学生的学号也可以用信息块码来编码。将学号分为几个信息块：学生的入学年份、系的代码、专业代码、班级代码等，每个信息块内部按顺序排列，信息块之间留出备用码。

3．归组分类码

将信息按一定的标准分为大类、中类、小类，每类分配顺序代码，就构成归组分类码。

与信息块码不同的是，不是按整个代码分组，而是按代码的代号分组，对各组内的位数没有限制。表 3.3 是归组代码的例子。

表 3.3　归组代码的例子

信　　息	代　　码
哲学	100
宗教	200
社会科学	300
法律	320
商法	325
公司法	3252
股份公司法	32524
合股公司法	32525

4．助记码

助记码是将数据的名称适当压缩组成代码，以便于记忆。

助记码多由汉语拼音、英文字母和数字等混合组成。例如，12 英寸电视机的代码是 12TV，29 英寸电视机的代码为 29TV。

5．数字式字符码

按规定的方式，将字符用数字表示，所形成的代码称为数字式字符码。

计算机中通用的 ASCII 码就是数字式字符码，表 3.4 列出了部分 ASCII 码。

表 3.4　部分 ASCII 码

ASCII 码	字　　符	ASCII 码	字　　符
048	0	065	A
049	1	066	B
050	2	067	C
051	3	068	D
052	4	069	E
053	5	070	F
054	6	071	G
055	7	072	H
056	8	073	I
057	9	074	J

6．组合码

在很多应用中，如果仅选用一种代码形式进行编码往往不能满足要求，而选用几种形

态的代码合成编码会产生很好的效果。这样的代码使用起来十分方便，只是代码的位数较多。

3.7.4 代码设计方法

代码设计时一定要遵循简单、唯一、标准化、可扩充、规范化的设计原则。记住设计代码的目的是提高信息的处理效率。

对于软件系统中的主要数据，尤其是汉字词语，一般都要编码，这样有利于识别、分类和检索。

代码设计的基本步骤如下。

1．确定编码对象

选择采用代码后可以提高输入、输出、查询效率的数据作为编码的对象，如学校的学生、教师、图书和设备，商场的商品、供货单位、财务科目等。

2．明确编码目的

确定编码后需要进行识别、分类、排序等的重要性，编码目的不同则所选用的代码种类也不相同。

3．确定代码的个数

确定当前的代码数目和将来可能扩充的代码数目。

4．确定代码使用范围和使用期限

确定代码可以使用的范围及使用期限。

5．确定代码体系和代码位数

这是编码的关键，要使设计的代码简短、易记、不易混淆。要根据代码使用的目的来确定采用哪种编码和代码位数。

6．确定编码规则

编码规则的确定要通过与使用人员密切合作、认真讨论，根据使用的需求、计算机处理的方便、容易记忆和维护综合考虑。

7．编写代码

在明确编码目的、编码对象、确定编码规则、代码的个数、代码的使用范围和期限、代码体系和位数等问题后，就可以进行代码编写。

8．编写代码词典

对于数据量小的代码，要在数据字典中将代码与数据的对应关系一一列出。

对于数据量较大的代码，应当编写代码词典。代码词典应记录数据与代码的对应关系、代码使用方法和示例、修改代码的手续及规则、代码管理的部门和权限等。

3.8 面向数据结构的设计方法

面向数据结构的设计方法是按输入、输出及计算机内部存储信息的数据结构进行软件结构设计的，把对数据结构的描述变换为对软件结构的描述。

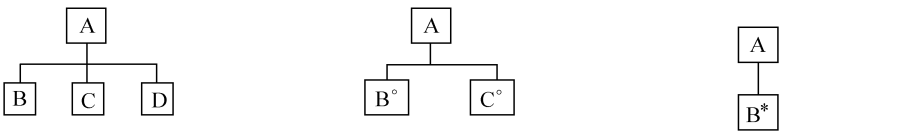
在许多应用领域中，信息的结构层次清楚，输入数据、输出数据及内部存储的信息

有一定的结构关系。数据结构不仅影响软件的结构设计，还影响软件的处理过程。例如，重复出现的数据通常由循环结构来控制；一个数据结构具有选择特性，既可能出现也可能不出现，就采用条件选择程序来控制；如果一个数据结构为分层次的，软件结构也必然为分层次的。因此，数据结构充分地揭示了软件结构。使用面向数据结构的设计方法，首先需要分析确定数据结构，并用适当的工具清晰地描述数据结构，最终得出对程序处理过程的描述。

面向数据结构的设计方法有两种：Jackson 方法和 Warnier 方法。这两种方法只是使用的图形工具不同。本节只介绍 Jackson 方法，Warnier 方法可参考本书 2.8.2 节介绍的 Warnier 图进行分析设计。

Jackson 方法由英国的 M. Jackson 提出，在欧洲较为流行，它特别适合于设计企事业管理类的数据处理系统。Jackson 方法的主要图形工具是 Jackson 图，它既可以表示数据结构，也可以表示程序结构。

Jackson 把数据结构（或程序结构）分为以下三种基本类型，如图 3.22 所示。



（a）数据 A 由 B、C、D 顺序组成　　（b）数据 A 由 B 或 C 中选一个　　（c）数据 A 由 B 重复组成

图 3.22　数据结构的三种基本类型

1．顺序结构

顺序结构的数据由一个或多个元素组成，每个元素依次出现一次。图 3.22（a）表示数据 A 由 B、C、D 三个元素顺序组成。

2．选择结构

选择结构的数据包含两个或多个元素，每次使用该数据时，按一定的条件从这些元素中选择一个。图 3.22（b）表示数据 A 根据条件从 B 或 C 中选择一个。B 和 C 的右上方加符号“°”表示从中选择一个。

3．重复结构

重复结构的数据，根据条件由数据元素出现 0 次或多次组成。图 3.22（c）表示数据 A 由 B 出现 0 次或多次组成。数据 B 的右上方加符号“*”表示重复。

Jackson 图有以下特点：

- 能对结构进行自顶向下的分解，可以清晰地表示层次结构。
- 结构易读、形象、直观。
- 既可表示数据结构，也可表示程序结构。

Jackson 设计方法采用以下 4 个步骤：

（1）分析并确定输入数据和输出数据的逻辑结构。

（2）找出输入数据结构和输出数据结构中有对应关系的数据单元。

（3）从描述数据结构的 Jackson 图导出描述程序结构的 Jackson 图。

（4）列出所有的操作和条件，并把它们分配到程序结构图中去。

下面结合具体例子进一步说明 Jackson 结构设计方法。

【例 3.10】 用 Jackson 方法对学生成绩管理系统进行结构设计。

例 2.2 学生成绩管理系统在学生入学时输入学生基本信息。每次单科成绩是按班级内学生学号的顺序依次输入每位学生的平时成绩和考试成绩，成绩输入格式如表 3.5 所示。然后由计算机计算每位学生的单科成绩总评分。输出的学生个人成绩单格式如表 3.6 所示，班级各科成绩汇总表格式如表 3.7 所示。

表 3.5　班级单科成绩表格式

上海××大学

2004—2005 年第一学期

成绩表

课程号：1090　课程名：计算机网络基础　系：计算机科学与技术　班级：04104111

学　号	姓　名	性　别	平时成绩	考试成绩	总　评

分数段	人数	百分比
90 分以上		
80～89 分		
70～79 分		
60～69 分		
不及格		

任课教师签名：

日期：　　年　月　日

表 3.6　学生个人成绩单格式

上海××大学

2004—2005 年第一学期

学生成绩单

学号：041011116　姓名：王力　系：计算机科学与技术　班级：0410111

课程名	平时成绩	考试成绩	总　评	考试/考查

表 3.7　班级各科成绩汇总表格式

上海××大学

2004—2005 年第一学期

班级成绩汇总表

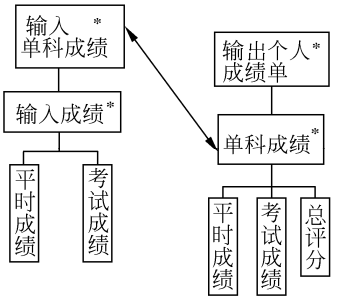
系：计算机科学与技术　　班级：0410111

学　号	姓　名	高等数学	计算机网络基础	英　语	政　治	体　育

根据以上输入数据和所需的输出表格，可写出输入数据结构、输出数据结构及程序结构的 Jackson 图，步骤如下：

（1）输入数据结构的 Jackson 图如图 3.23（a）所示。

（2）输出数据结构的 Jackson 图如图 3.23（b）所示。



（a）输入数据结构　（b）输出数据结构

图 3.23　学生成绩管理系统输入、输出数据结构的对应关系图

（3）根据输入、输出数据结构的 Jackson 图用双向箭头画出对应关系图，如图 3.23 所示。

（4）从输入、输出数据结构关系导出程序结构 Jackson 图。

（5）列出所有操作和条件，并把它们分配到程序结构图的适当位置，如图 3.24 所示。

- 输入学生基本情况；
- 输入学生单科成绩；
- 计算单科成绩总评分：总评分＝平时成绩×0.3+考试成绩×0.7；
- 输出班级单科成绩表；
- 输出学生个人成绩单；
- 输出班级成绩汇总；
- 输出重修名单；
- 输出留级名单。

重复条件 sum1：对所有学生都执行一次。

重复条件 sum2：总评成绩不及格人数。

重复条件 sum3：留级人数。

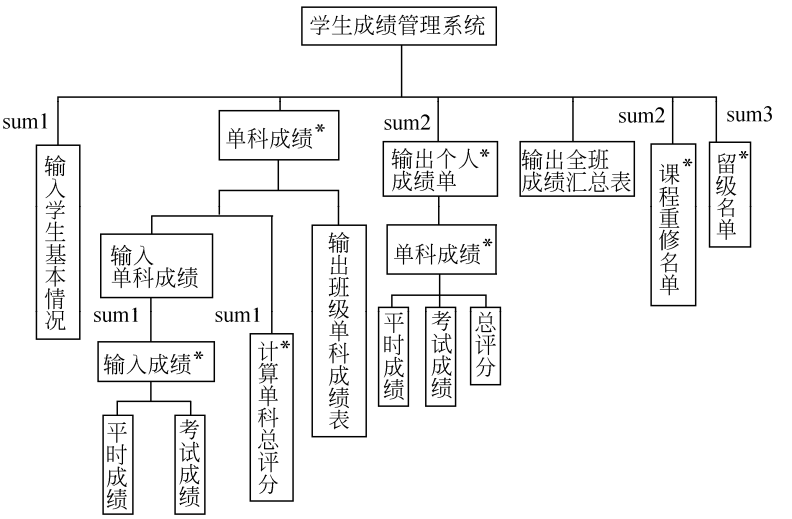


图 3.24　学生成绩管理系统程序结构 Jackson 图

3.9　软件设计文档

3.9.1　概要设计说明书

本节介绍概要设计说明书编写提示和概要设计的复审。

1．概要设计说明书的内容

概要设计说明书的内容如下：

1）引言

- 编写目的。
- 背景说明。
- 定义。
- 参考资料。

2）总体设计

- 需求规定。
- 运行环境。
- 基本设计概念和处理流程。
- 结构。
- 功能需求与程序的关系。
- 人工处理过程。
- 尚未解决的问题。

3）接口设计

- 用户接口。
- 外部接口。

- 内部接口。

4）运行设计

- 运行模块组合。
- 运行控制。
- 运行时间。

5）系统数据结构设计

- 逻辑结构设计要点。
- 物理结构设计要点。
- 数据结构和程序的关系。

6）系统出错处理设计

- 出错信息。
- 补救措施：后备技术、降低效果技术、恢复及再启动技术。
- 系统维护设计。

2．概要设计复审

概要设计复审的参加人员包括结构设计负责人、设计文档的作者、课题负责人、行政负责人、对开发任务进行技术监督的软件工程师、技术专家，以及其他方面代表。

概要设计复审的主要内容是审查软件结构设计。要充分彻底地评价数据流图，严格审查结构图中参数传输情况、检查全局变量和模块间的对应关系，对系统接口设计进行检查。纠正设计中的错误和缺陷，使有关设计者了解和任务有关的接口情况。

3.9.2 数据库设计说明书

数据库设计说明书的编写目的是对所设计的数据库中所有的标识、逻辑结构和物理结构做出具体的设计规定。数据库设计说明书的编写内容如下：

1．引言

（1）编写目的。

（2）背景说明。

（3）定义。

（4）参考资料。

2．外部设计

（1）标识符和状态。

（2）使用它的程序。

（3）约定。

（4）专门指导。

（5）支持软件。

3．结构设计

（1）概念结构设计。

（2）逻辑结构设计。

（3）物理结构设计。

4．运用设计

（1）数据字典设计。

（2）安全保密设计。

3.9.3 详细设计说明书

本节介绍详细设计说明书的内容和详细设计的复审。

1．详细设计说明书内容

1）引言

- 编写目的。
- 背景说明。
- 定义。

2）程序系统的结构

用一系列图表列出本软件系统内每个程序（包括每个模块和子程序）的名称、标识符和它们之间的层次结构关系。每个程序根据实际需要说明以下内容，并不是每个程序都需要写出下列全部内容。

- 程序描述。
- 功能。
- 性能。
- 输入项。
- 输出项。
- 算法。
- 流程逻辑。
- 接口。
- 存储分配。
- 注释设计。
- 限制条件。
- 测试计划。
- 尚未解决的问题。

2．详细设计的复审

软件的详细设计完成后，必须从软件的正确性和可维护性两个方面对它的逻辑、数据结构和界面等进行审查。

详细设计的复查可用下列形式之一完成：

（1）设计者和设计组的另一个成员一起进行静态检查。

（2）由检查小组进行较正式的软件结构设计检查。

（3）由检查小组进行正式的设计检查，对软件设计质量给出评价。

实践证明，正式的详细设计复审工作，在发现设计错误方面的作用与软件测试同样有效，并且其发现设计错误更加容易。

3.9.4 操作手册编写提示

在详细设计阶段描述了系统功能如何实现的具体算法，因而可以写出初步的用户操作手册，在程序编码阶段再对操作手册进行补充和修改。操作手册的编写内容如下：

1．引言

（1）编写目的。

（2）背景说明。

（3）定义。

（4）参考资料。

2．软件概述

1）软件的结构

结合软件系统所具有的功能，包括输入、处理和输出提供该软件的总体结构图表。

2）程序表

列出本系统内每个程序的标识符、编号和简称。

3）文卷表

列出将由本系统引用、建立或更新的每个永久性文卷，说明它们各自的标识符、编号、简称、存储媒体和存储要求。

3．安装和初始化

具体说明为使用本软件而需要进行的安装与初始化过程，包括程序的储存形式、安装与初始化过程中的全部操作命令、系统对这些命令的反应与答复、表明安装工作完成的测试实例等。如果有的话，还应说明安装过程中所需用到的专门软件。

4．运行说明

所谓运行是指提供一个启动控制信息后，直到计算机系统等待另一个启动控制信息时为止的计算机系统执行的全部过程。

1）运行表

列出每种可能的运行，说明每个运行的目的，指出每个运行所执行的程序。

2）运行步骤

逐个说明每个运行及完成整个系统运行的步骤。

以对操作人员最方便、最有用的形式说明运行的有关信息。例如，运行控制、操作信息和运行目的。

操作要求：启动方法、预定时间启动等，预计的运行时间和解题时间，操作命令，与运行有联系的其他事项。

输入输出文卷，占用硬设备的优先级及保密控制等。

输出：提供本软件输出的有关信息、输出媒体、文字容量、分发对象、保密要求。

输出的复制：提供有关信息、复制的技术手段，纸张或其他媒体的规格，装订要求，分发对象，复制份数。

5．非常规过程

提供有关应急操作或非常规操作的必要信息，如出错处理操作、向后备系统的切换操作，以及其他必须向程序维护人员交代的事项和步骤。

6．远程操作

如果本软件能够通过远程终端控制运行，则说明操作过程。

小　　结

概要设计阶段通常完成确定设计方案和结构设计两个任务。

详细设计阶段完成三个任务：过程设计、接口设计和数据设计。

结构化设计的基本要点如下：

（1）软件系统由层次结构的模块构成。

（2）模块是单入口、单出口的。

（3）模块构造和联结的基本准则是模块独立。

（4）软件系统结构用图来描述。

软件结构设计基本原理：软件的模块化，模块独立性，抽象和逐步求精，信息隐蔽和局部化等。

评价模块分割好坏的标准主要有以下4个方面：

（1）模块的大小。

（2）模块之间的联系程度——耦合。尽量使用数据耦合，少用控制耦合和特征耦合，不采用内容耦合，控制公共环境耦合。

（3）模块内的联系程度——内聚。内聚按紧密程度从高到低排列次序为功能内聚、顺序内聚、通信内聚、过程内聚、时间内聚、逻辑内聚、偶然内聚。

（4）模块信息的隐蔽程度。

模块设计启发式规则如下：

（1）尽力提高模块独立性。

（2）注意模块的可靠性、通用性、可维护性、简单性。

（3）模块的大小应适中。

（4）模块的深度、宽度、扇出和扇入要适当。通常顶层扇出高，中间扇出较少，下层调用公共模块。

进行软件系统结构设计可采用层次图、HIPO图和结构图描绘系统模块的层次结构。

对于交互式软件系统来说，人机界面设计是接口设计的一个组成部分。人机界面设计的质量直接影响用户对软件产品的评价，应对人机界面设计给予足够的重视。

详细设计阶段使用的工具有流程图、N-S图、问题分析图（PAD图）、判定表、判定树、过程设计语言（PDL）等。

习　题　3

1．什么是概要设计？其基本任务是什么？

2．什么是模块？模块有哪些属性？

3．什么是模块化？划分模块的原则是什么？

4．什么是软件结构设计？

5．画例 2.2 学生成绩管理系统的 HIPO 图。

6．画例 3.4 图书馆管理系统的 HIPO 图。

7．选择填空

在众多设计方法当中，结构化设计（SD）方法是最广泛应用的一种，这种方法可以同分析阶段的 A 方法及编码阶段的 B 方法前后衔接。SD 方法是建立良好程序结构的方法，它提出衡量模块结构质量的标准是模块间联系与模块内部联系的紧密程度，SD 方法的最终目标是 C 。用于表示模块间调用关系的图叫 D 。划分模块的信息隐蔽原则方法称为 E 方法。

供选择的答案：

A，B：① Jackson　② 结构化分析 SA　③ 结构化程序设计 SP　④ Parnas

C：① 模块间联系紧密，模块内联系紧密
② 模块间联系紧密，模块内联系松散
③ 模块间联系松散，模块内联系紧密
④ 模块间联系松散，模块内联系松散

D：① PAD　② SC　③ N-S　④ HIPO

E：① Jackson　② Parnas　③ Turing　④ Wirth

8．选择填空

模块内聚性是衡量模块内各成分之间彼此结合的紧密程度。若一组语句在程序多处出现，为节省内存而把这些语句放在一个模块中，该模块的内聚性称为 A 。而将几个逻辑上相似的成分放在同一个模块中，该模块的内聚性是 B 。如果模块中所有成分引用共同的数据，该模块的内聚性是 C 。而模块内某个成分的输出是另一个成分的输入，该模块内聚性是 D 。当模块中所有成分结合起来完成一项任务，该模块的内聚为 E 。

供选择的答案：

A，B，C，D，E：① 功能内聚　② 顺序内聚　③ 通信内聚　④ 过程内聚
⑤ 偶然内聚　⑥ 瞬时内聚　⑦ 逻辑内聚

9．选择填空

结构化分析方法（SA）、结构化设计方法（SD）和 Jackson 方法是软件开发过程中应用的方法。人们使用 SA 方法可以得到 A ，该方法的基本手段是 B ；使用 SD 方法可以得到 C ，并可以实现 D ；而使用 Jackson 方法可以实现 E 。

供选择的答案：

A，C：① 程序流程图　② 具体的语言程序
③ 模块结构图及模块功能说明书　④ 分层数据流图和数据字典

B：① 分解与抽象　② 分解与综合　③ 归纳与推导　④ 试探与回溯

D，E：① 从数据结构导出程序结构　② 从数据流图导出初始结构图
③ 从模块结构导出数据结构　④ 从模块结构导出程序结构

10．某旅行社根据旅游淡季、旺季及是否团体订票，确定旅游票价的折扣率。具体规定如下：人数在 20 人以上的属团体，20 人以下的是散客。每年的 4～5 月、7～8 月、10 月为旅游旺季，其余为旅游淡季。旅游旺季，团体票优惠 5%，散客不优惠。旅游淡季，团体票优惠 30%，散客优惠 20%。试分别用判定表和判定树表示旅游订票的优惠规定。

11．下面是用 PDL 写的程序段，请分别画出对应的 N-S 图和 PAD 图。

```
While  C  do
       If  A>0  Then   A1   Else  A2    Endif
       If  B>0  Then   B1
                     If  C>0  Then  C1   Else  C2    Endif
              Else  B2
          Endif
       B3
EndwhileS
```

12．请画出下列伪码程序对应的盒图。

```
START
IF P THEN
  WHILE q DO
  f
  END DO
 ELSE
  BLOCK
  g
  n
  END BLOCK
  END IF
  STOP
```

13．研究下面的伪码程序：

```
LOOP:
    Set I to (START+FINISH) / 2
    If  TABLE(I)=ITEM goto F()UND
    If  TABLE(I)<ITEM Set START to(I+1)
    If  TABLE(I)>ITEM Set FINISH to(I-1)
    If(FINISH—START)>1 goto LOOP
    If TABLE(START)=ITEM goto FOUND
    If TABLE(FINISH)=ITEM goto FOUND
   Set FLAG to 0
   Goto DONE
   FOUND: Set FLAG to l
   DONE: Exit
```

要求：

（1）画出程序流程图。

（2）程序是结构化的吗？说明理由。

（3）若程序是非结构化的，请设计一个等价的结构化程序并且画出程序流程图。

（4）此程序的功能是什么？它完成预定功能有什么隐含的前提条件吗？