软件工程导论（第6版）

第6章 详细设计

第6章　详细设计

根本目标：确定应该怎样具体地实现所要求的系统。

- 详细设计阶段的任务不是具体地编写程序，而是要设计出程序的“蓝图”。
- 详细设计的结果基本上决定了最终的程序代码的质量。

第6章　详细设计

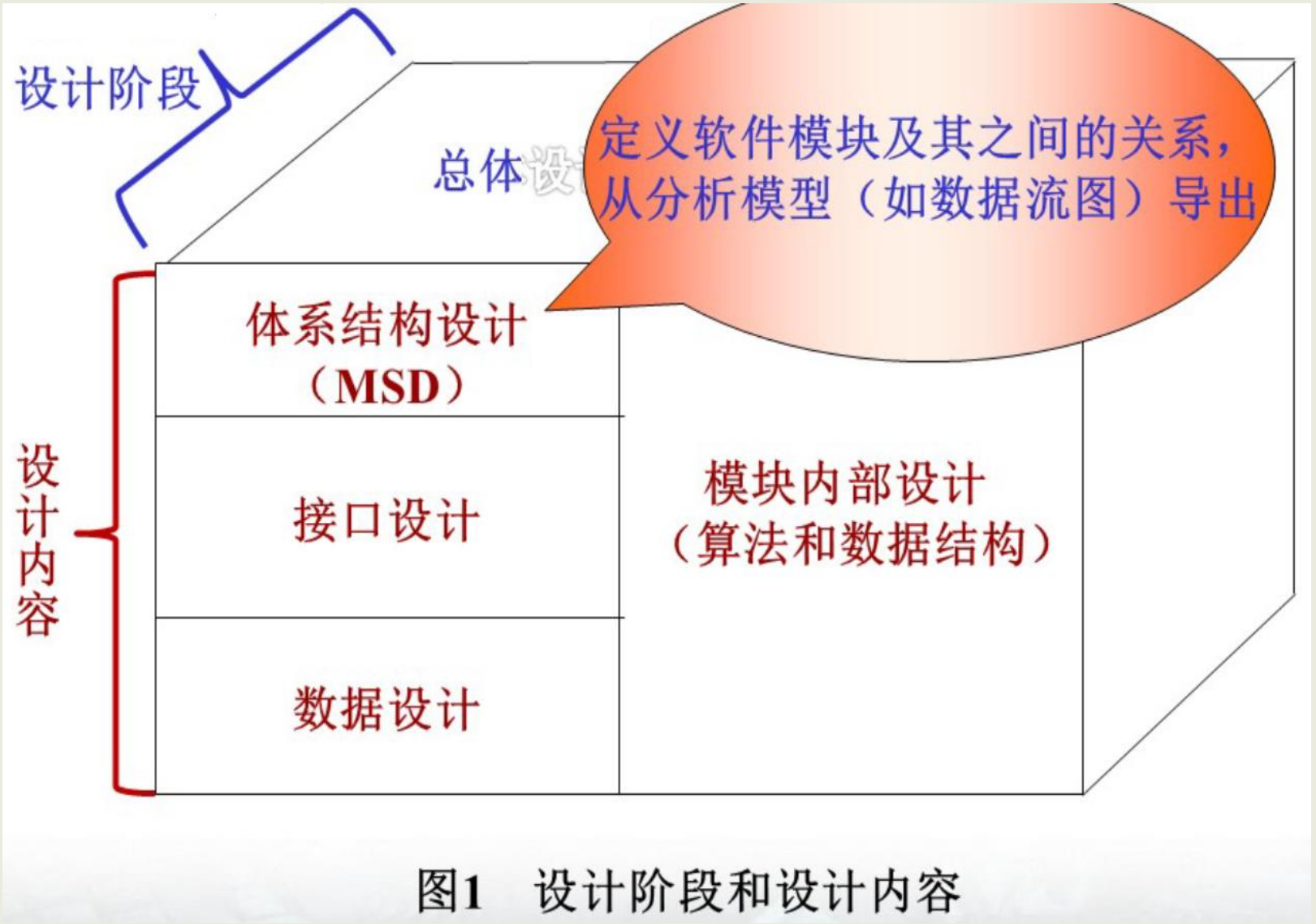


图1　设计阶段和设计内容

第6章　详细设计

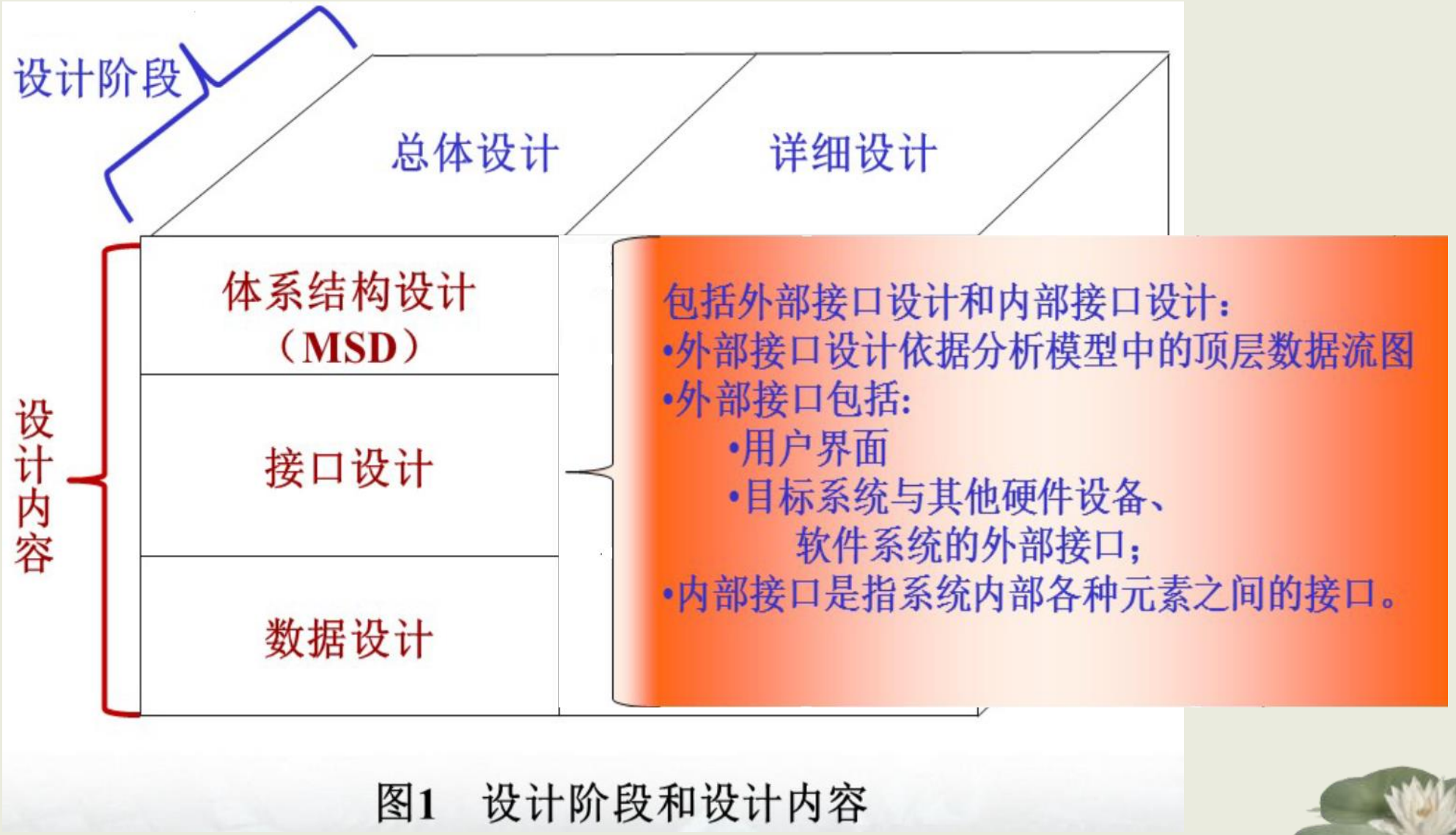


图1　设计阶段和设计内容

第6章　详细设计

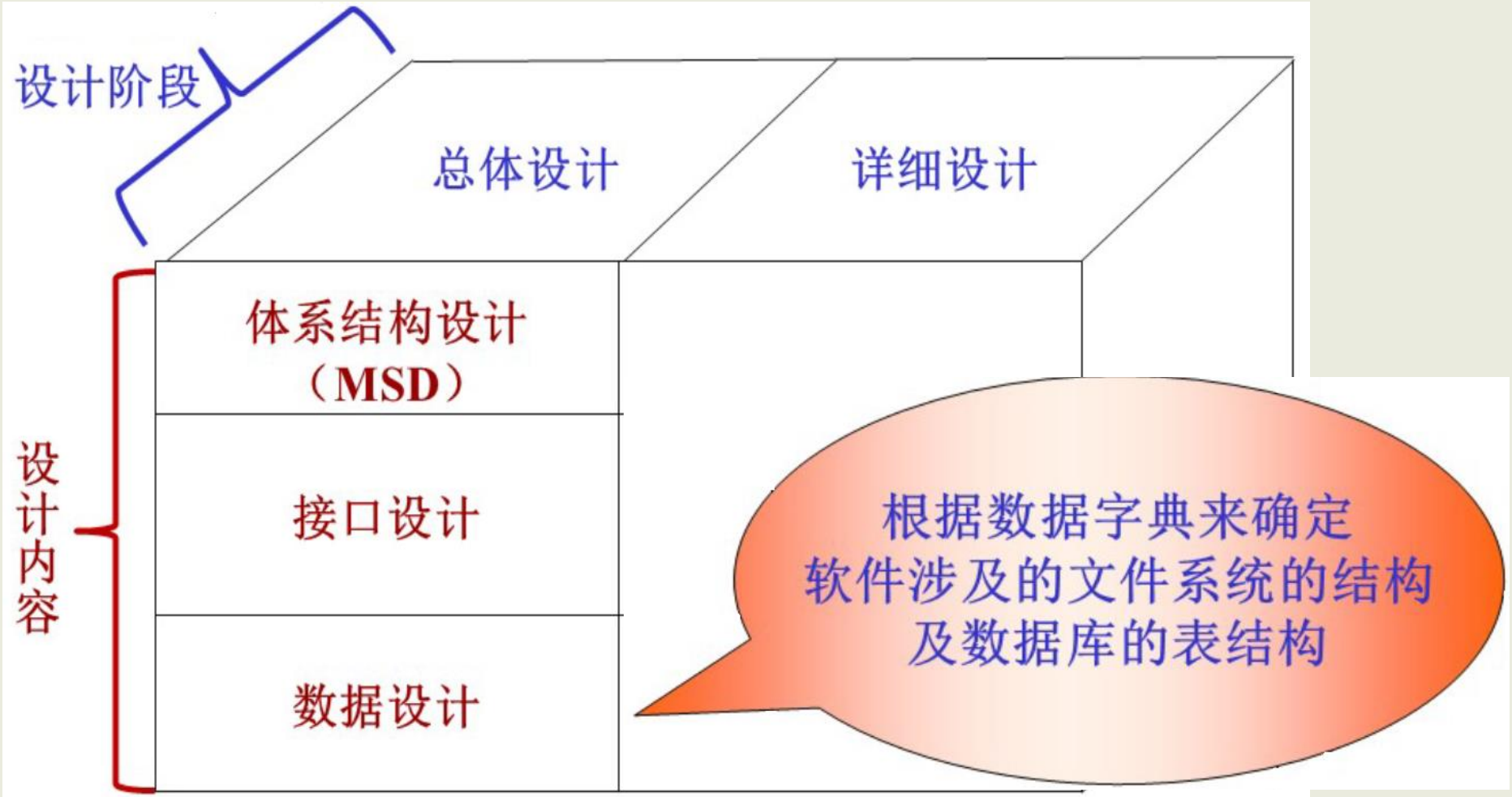


图1　设计阶段和设计内容

第6章　详细设计

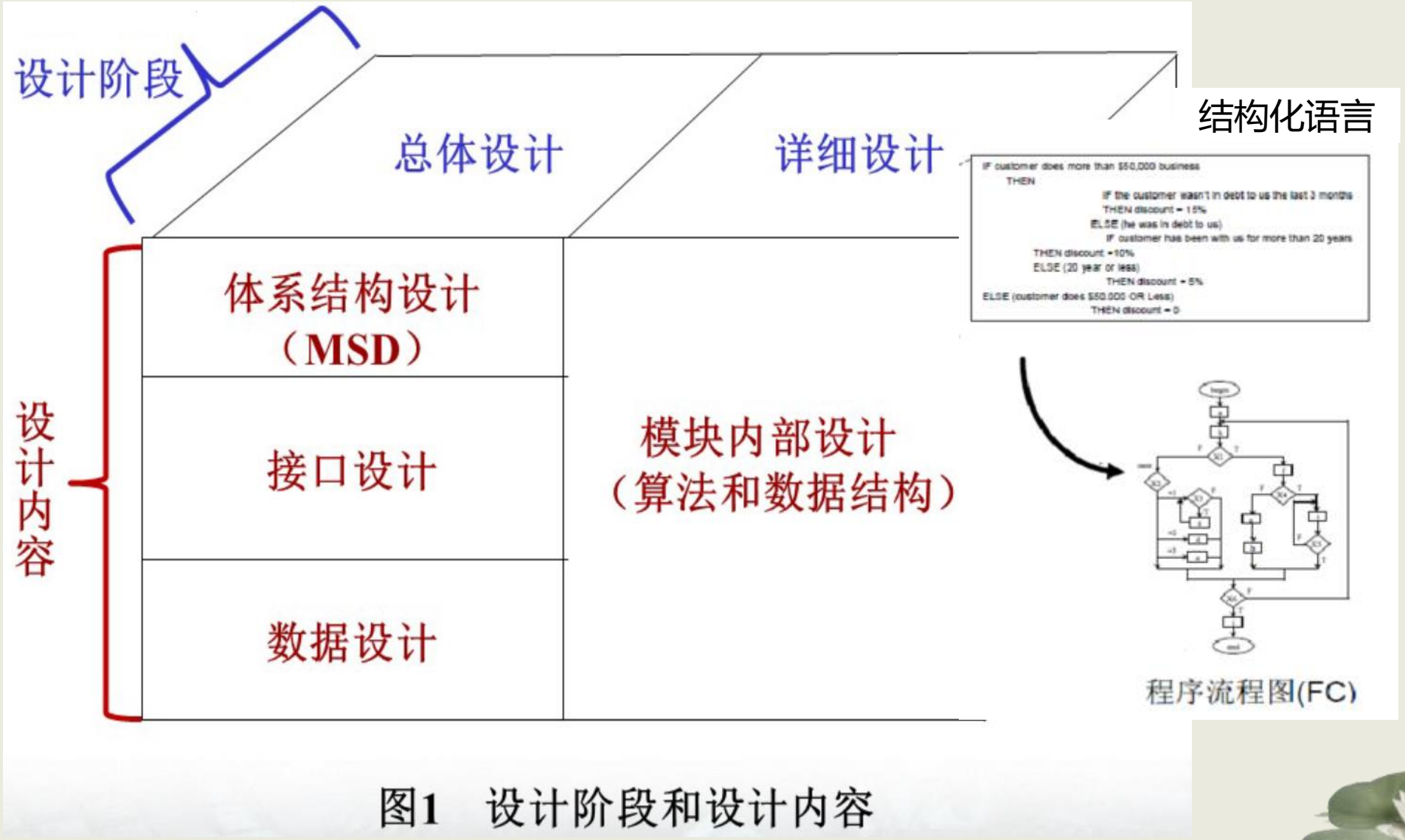


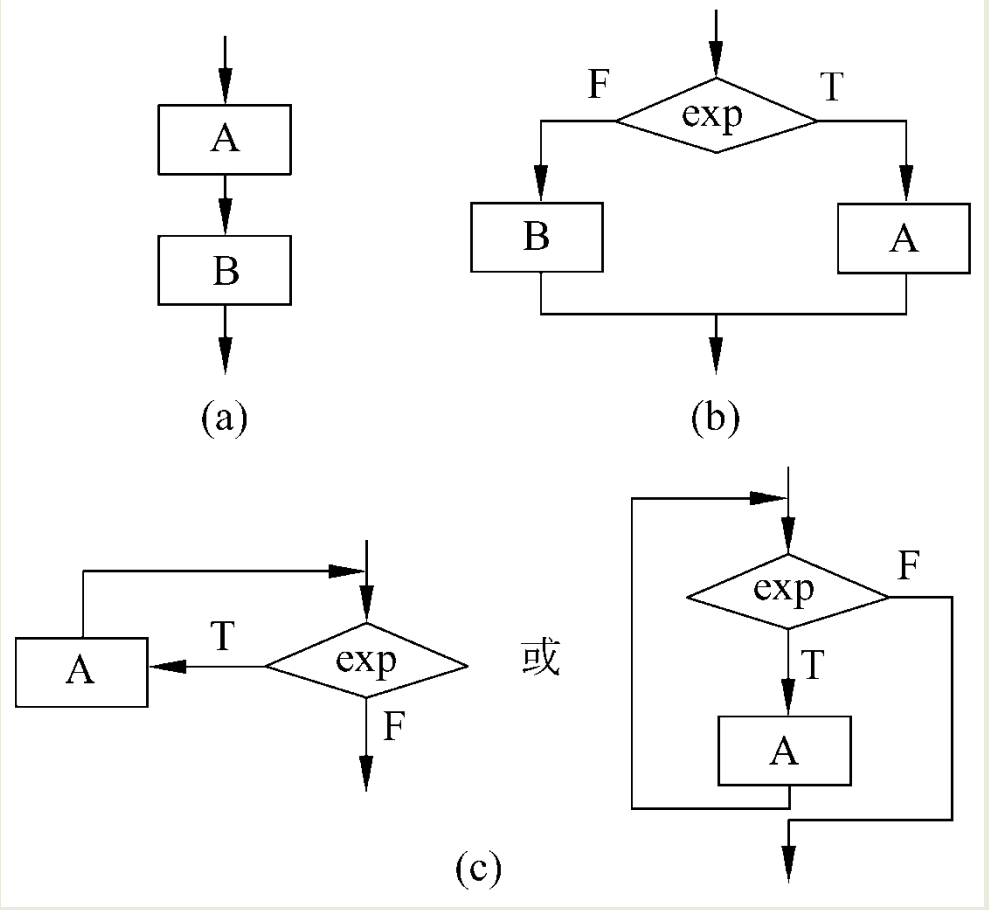
图1　设计阶段和设计内容

主要内容

6.1 结构程序设计

- 1965年结构程序设计的概念最早由E.W.Dijkstra提出：程序的质量与程序中所包含的GO TO 语句的数量成反比；

- 1966年Bohm和Jacopini证明了只用“顺序”、“选择”和“循环”控制结构就能实现任何单入口单出口的程序。



6.1 结构程序设计

实际上用顺序结构和循环结构(又称DO-WHILE结构)完全可以实现选择结构(又称IF-THEN-ELSE结构），因此，理论上最基本的控制结构只有两种。

- 1968年Dijkstra再次建议从一切高级语言中取消GO TO语句，只使用3种基本控制结构写程序。学界认识到不是简单地去掉GO TO 语句的问题，而是要创立一种新的程序设计思想、方法和风格。
- 1971年IBM公司在纽约时报信息库管理系统的设计中成功地使用了结构程序设计技术
- 1972年IBM公司的Mills进一步提出，程序应该只有一个入口和一个出口，补充了结构程序设计的规则。

6.1 结构程序设计

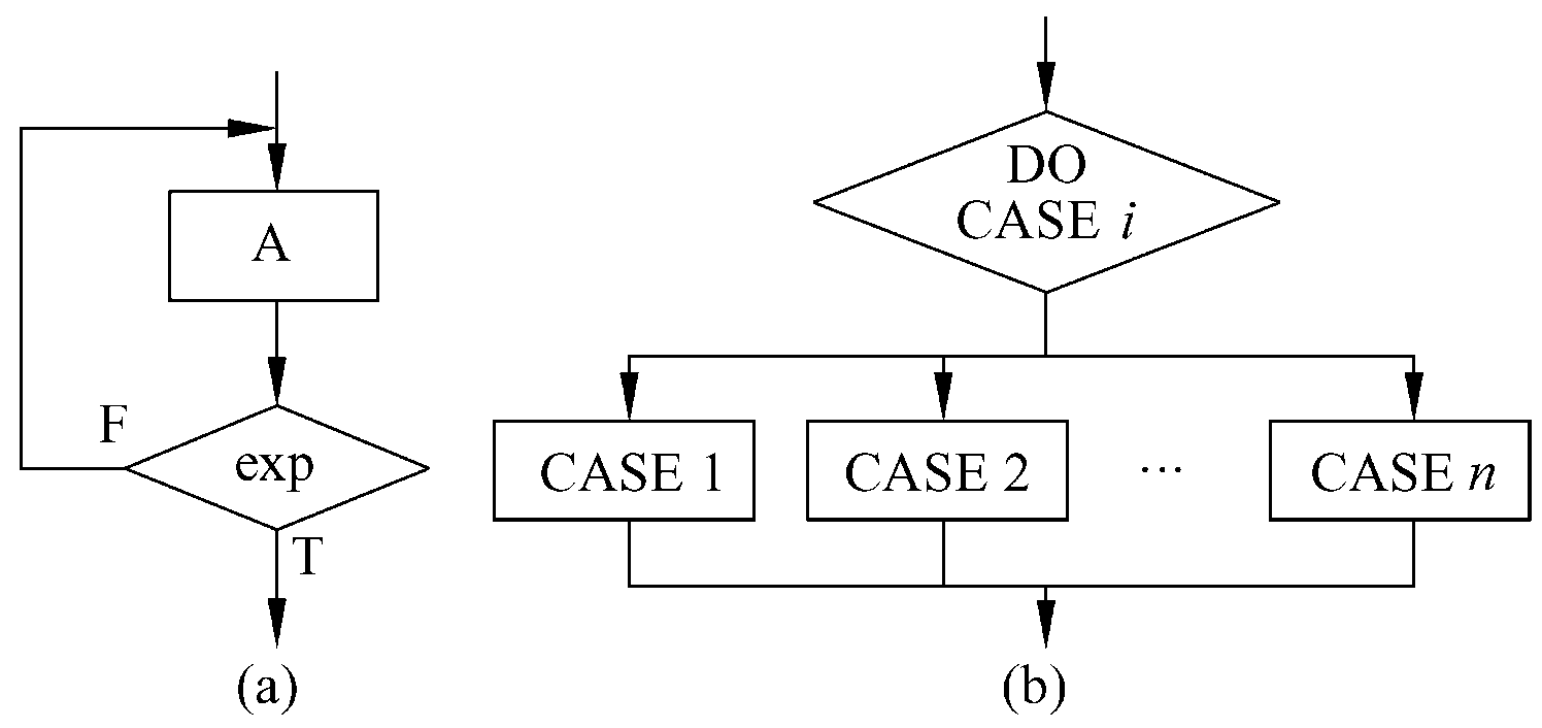
结构程序设计经典定义：如果一个程序的代码块仅仅通过顺序、选择和循环这3种基本控制结构进行连接，并且每个代码块只有一个入口和一个出口，则称这个程序是结构化的。

结构程序设计更全面的定义：结构程序设计是尽可能少用GO TO语句的程序设计方法。最好仅在检测出错误时才使用GO TO语句，而且应该总是使用前向GO TO语句。

- **使用结构程序设计技术的好处：**
  - **1）提高软件开发工程的成功率和生产率；**
  - **2）系统有清晰的层次结构，容易阅读理解；**
  - **3）单入口单出口的控制结构，容易诊断纠正；**
  - **4）模块化可以使得软件可以重用；**
  - **5）程序逻辑结构清晰，有利于程序正确性证明。**

6.1 结构程序设计

从理论上说只用上述3种基本控制结构就可以实现任何单入口单出口的程序，但是为了实际使用方便起见，常常还允许使用DO-UNTIL和DO-CASE两种控制结构



6.1 结构程序设计

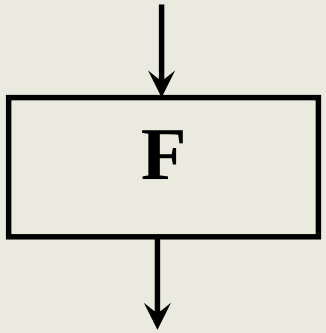
•经典的结构程序设计：如果只允许使用顺序、IF-THEN-ELSE型分支和DO-WHILE型循环这3种基本控制结构，则称为经典的结构程序设计；

扩展的结构程序设计：如果除了上述3种基本控制结构之外，还允许使用DO-CASE型多分支结构和DO-UNTIL型循环结构，则称为扩展的结构程序设计；

修正的结构程序设计：如果再允许使用LEAVE(或BREAK)结构，则称为修正的结构程序设计。

6.1.1 结构化程序

6.1.1.1 控制结构



函数结点

•流程图通常由三种结点组成：

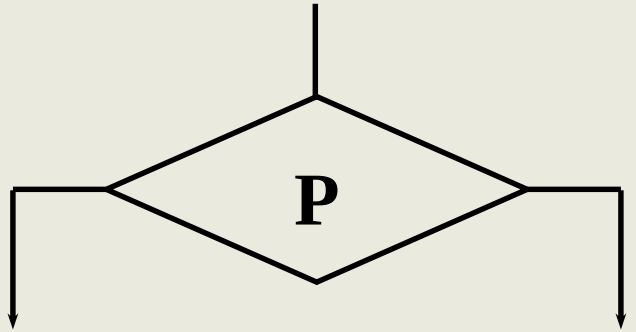
•1）**函数结点**

• 如果一个结点有一个入口线和一个出口线，则称为函数结点。

•由于函数结点一般对应于赋值语句，所以 F 也表示了这一个结点对应的函数关系。

•2）**谓词结点**

• 如果一个结点有一个入口线和两个出口线，而且它不改变程序的数据项的值，则称为谓词结点。

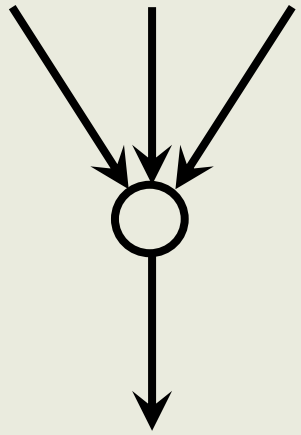


谓词结点

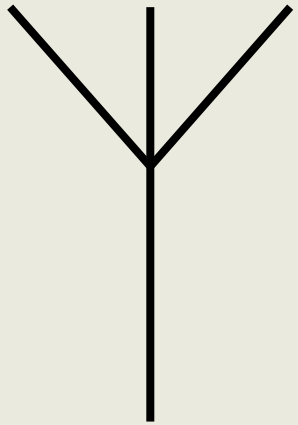
P是一个谓词，根据P的逻辑值（T或F），结点有不同的出口。

•3）汇点

• 如果一个结点有两个或多个入口线和一个出口线，而且它不执行任何运算，则称为汇点 。



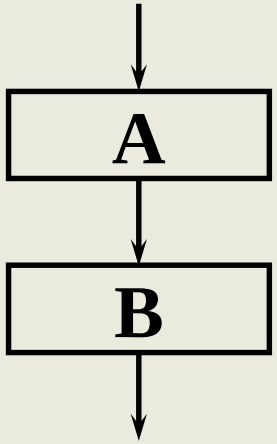
汇点



汇点的简略表示

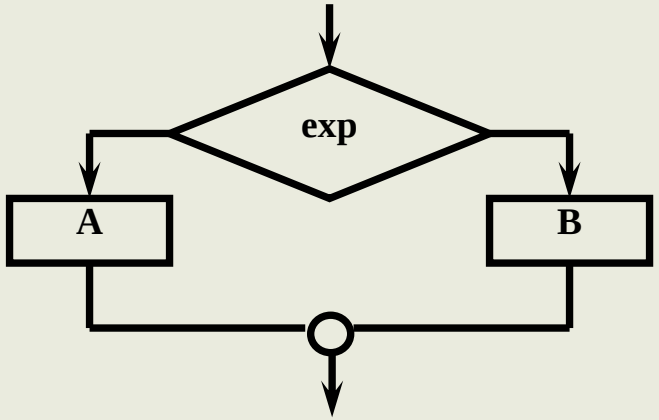
2. 三种基本控制结构

- 1）**顺序结构**：相当于“A、B”



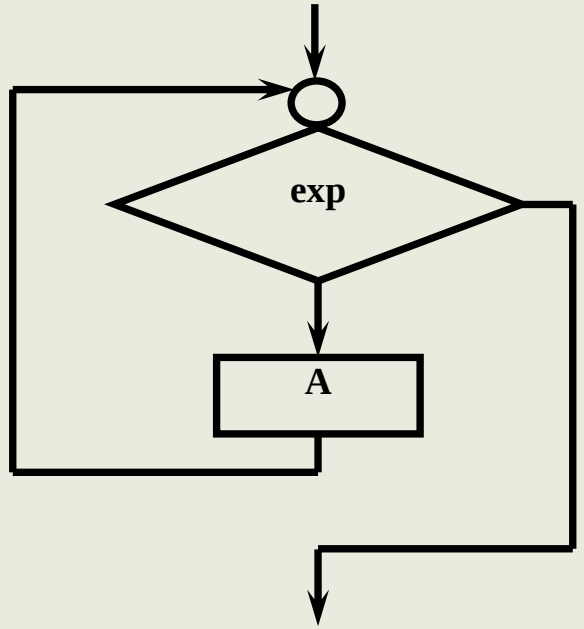
(a)顺序结构

- 2）**选择结构**
- 相当于“If exp then A else B endif”



(b)选择结构

- 3）**循环结构**：
- 相当于“While exp do A”

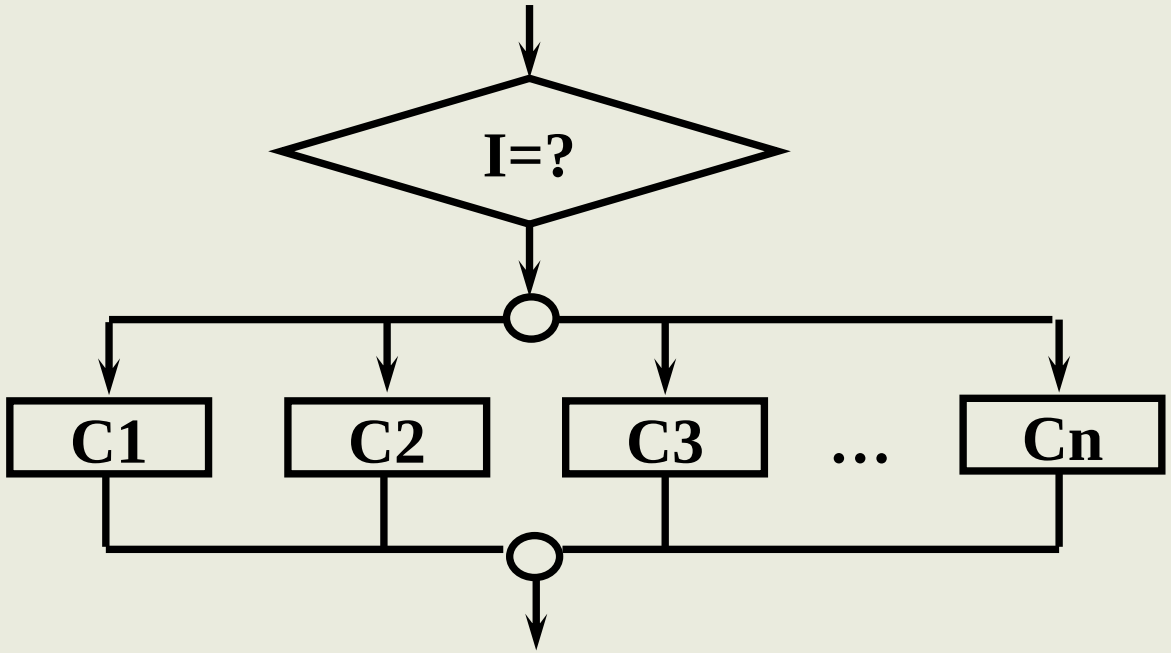


(c)循环结构

3. 扩充两种控制结构

•1）多分支结构

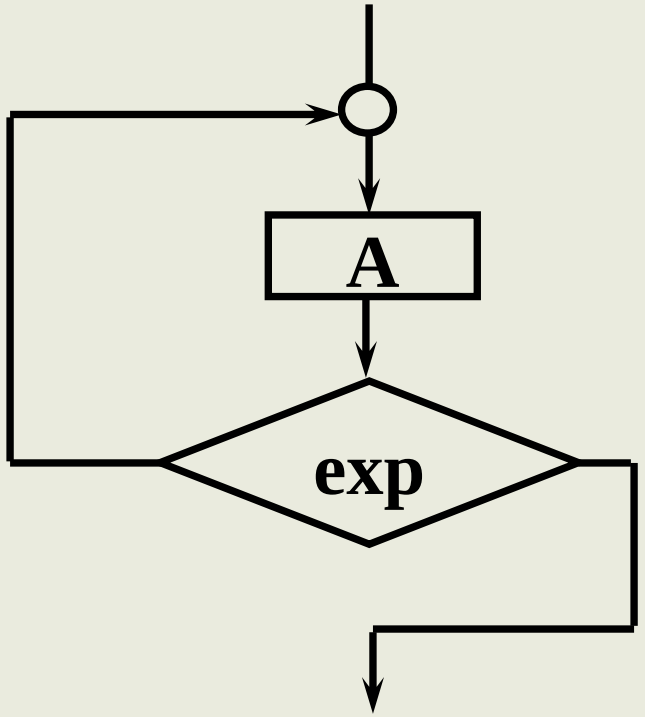
• 相当于“Case I of I=1:C1; I=2:C2; I=3:C3; … ; I=n:Cn”



(d) 多分支结构

•2）UNTIL循环结构

• 相当于“Repeat A Until exp”

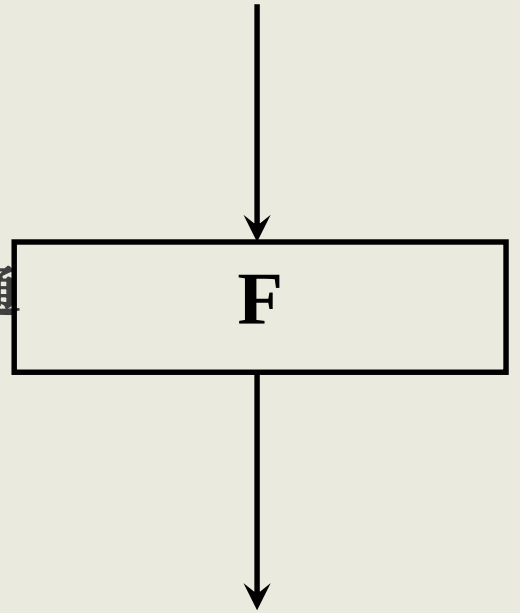


(e)UNTIL循环

•6.1.1.2 正规程序（定义，定理）

•**定义1**：一个流程图程序如果满足下面两个条件，称为正规程序：

- 1）具有**一个入口线和一个出口线**；
- 2）对每一个**结点**，都有一条从入口线到出口线的通路通过该结点。



函数结点

•由于正规程序有一个入口线和一个出口线，因而一个正规程序总可以抽象为一个**函数结点**。

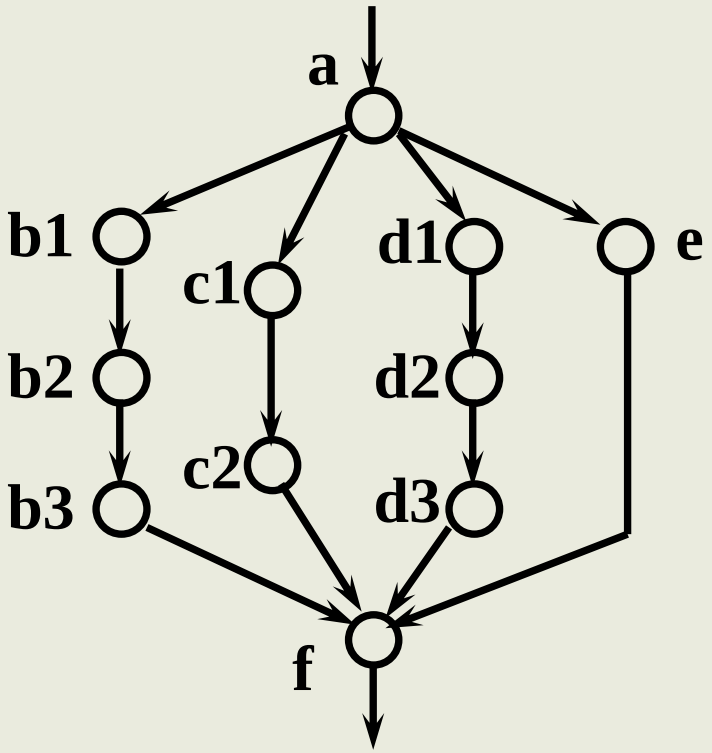
•**定义2**：如果一个正规程序的**某个部分**仍然是正规程序，那么称它为该正规程序的正规子程序。

6.1.1.3 基本程序

- 如图：封闭结构为
- { a - [b1 - b2 - b3 ; c1 - c2 ; d1 - d2 - d3 ; e] - f }

•先给出一个概念：**封闭结构**

•**定义3**：流程图中，两个结点之间所有没有重复结点的通路组成的结构称为封闭结构。



6.1.1.3 基本程序

定义4：一个正规程序，如果满足以下两个条件，则称之为基本程序：

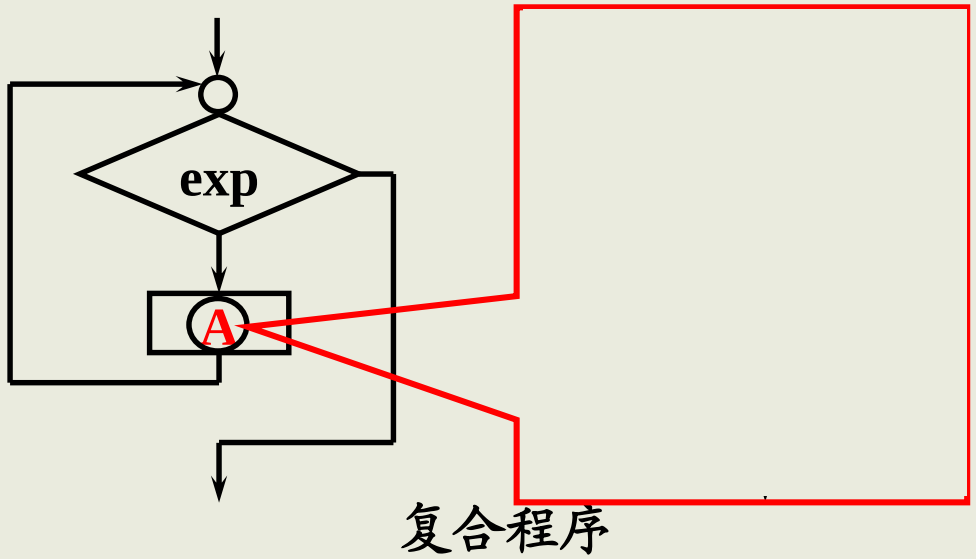
- 1）不包括多于一个结点的正规子程序，即它是一种不可再分解的正规程序；（程序自身不可视为正规子程序）
- 2）如果存在封闭结构，封闭结构都是正规程序。
- 基本程序形式有多种，前面提到的三种基本控制结构（顺序结构、选择结构、循环结构）和两个扩充控制结构（多分支结构、UNTIL循环结构）都是基本程序。

·定义5：

- 用以构造程序的基本程序的集合称为**基集合**。
- 如：**{顺序，if-then-else，while do}，{顺序，if-then-else，repeat-until}**都是基集合。

·定义7：由基本程序的一个固定的基集合构造出的复合程序，称为结构化程序

·定义6：如果一个基本程序的函数结点用另一个基本函数程序替换，产生的新的正规程序称为复合程序。



复合程序

循环结构的A函数结点用另一循环结构代替，即嵌套循环，就产生了复合程序。

由于复合程序是由一些基本程序组成，因此，无论从总体上看或是从每个组成部分看，都满足“一个入口，一个出口”的原则，这样的程序就是通常说的好结构程序，或者结构化程序。

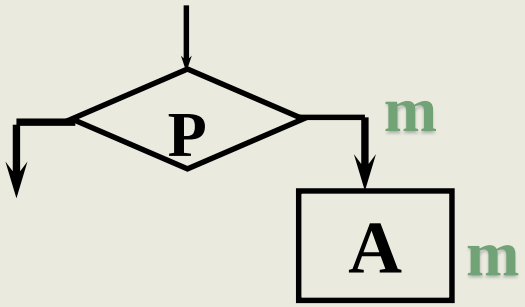
6.1.2 结构化定理

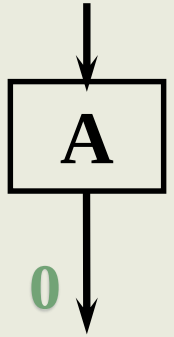
•结构化定理：任一正规程序都可以函数等价于一个由基集合{顺序，If-else-then，While-do}产生的结构化程序。

• 实际上，只要能证明可以将任一正规程序转换成等价的结构化程序就可以证明这个结构化定理。

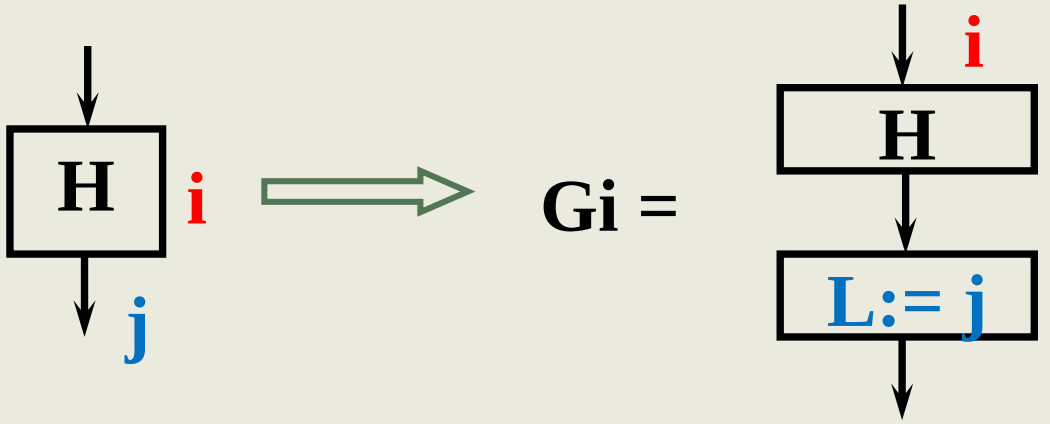
• 证明：（分三步进行结构化程序的转换）

•步骤一：从程序入口处开始给程序的函数结点和谓词结点编号：1，2，3，…，n，同时，将每个函数和谓词结点的出口线用它后面的结点的号码进行编号，如果出口线后面没有结点，也就是说该结点的出口线与程序的出口线（结束）相连时，出口线编号为0。

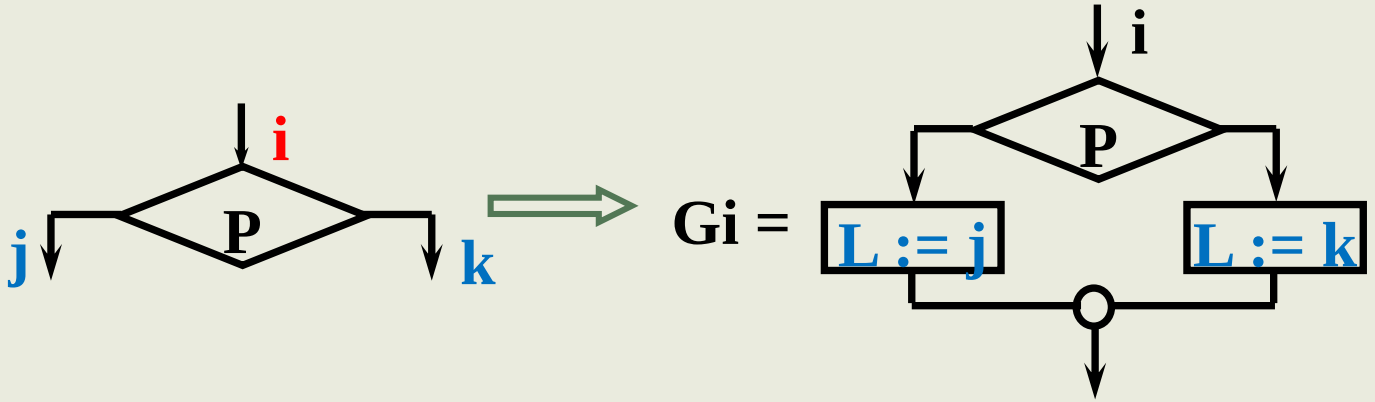




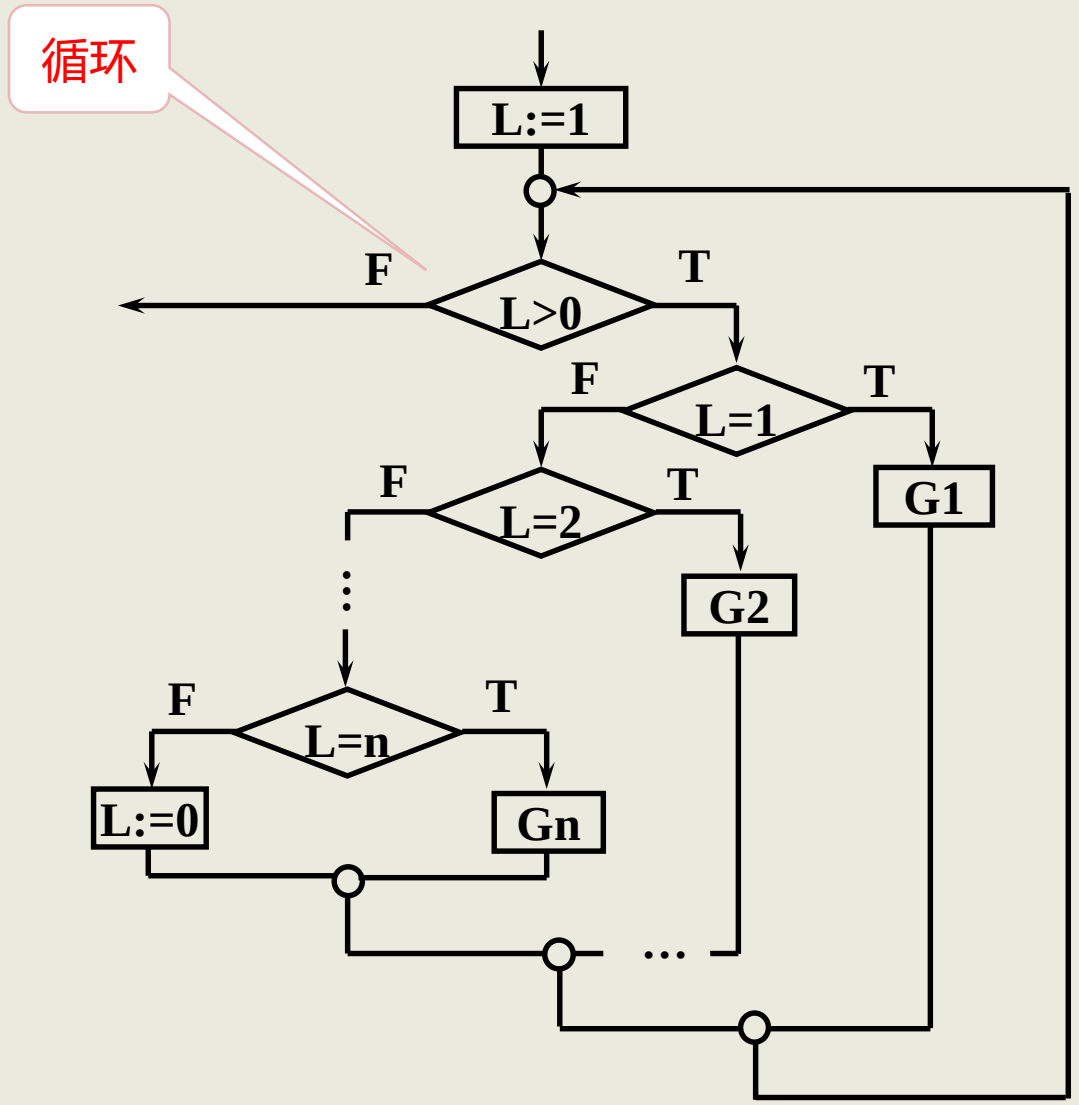
• 步骤二：对原程序中每一个编号为i，出口线编号为j的**函数结点**H，构造一个新的序列程序Gi，如图：



•类似地，对于每个编号为i，出口线分别为j和k的**谓词结点**，构造一个新的选择程序Gi，如图：



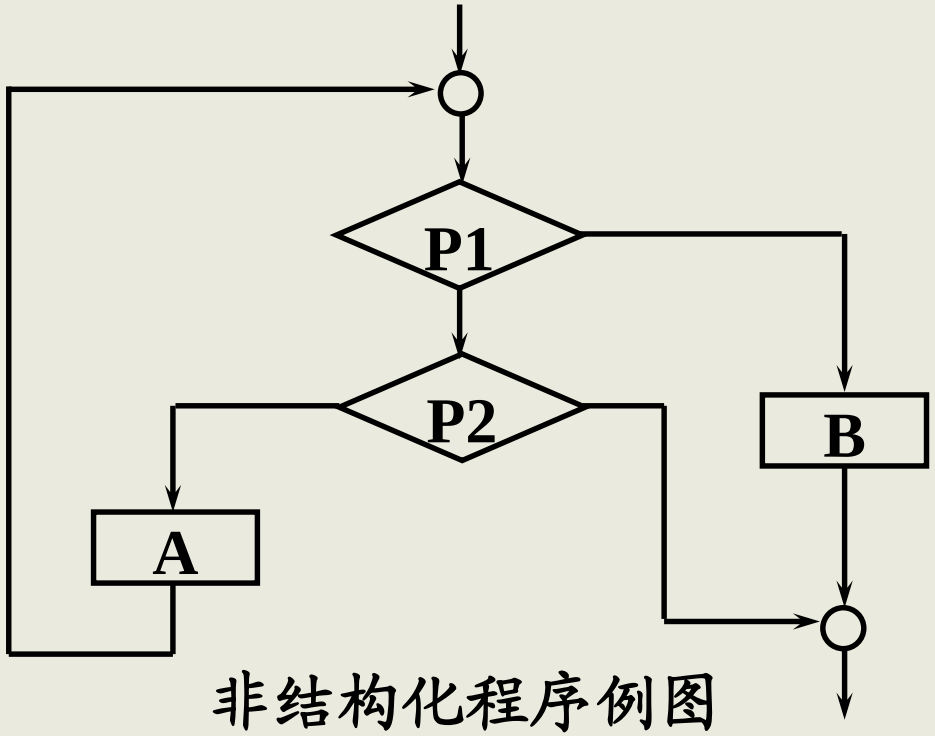
•步骤三：利用已经得到的一些Gi程序（i=1,2,3,…,n），按下图的形式构造一个While-do循环。



- 图中的循环体是一个对L从1到n的嵌套选择（if-then-else）程序，转换后的程序与原程序是等价的，是由基集合{顺序、选择、循环} 所复合成的结构化程序。
- 这种方法并不是唯一的把程序转变为结构化程序的方法，所得的程序也不一定是最好的。
- 它的目的是为了证明结构化定理。

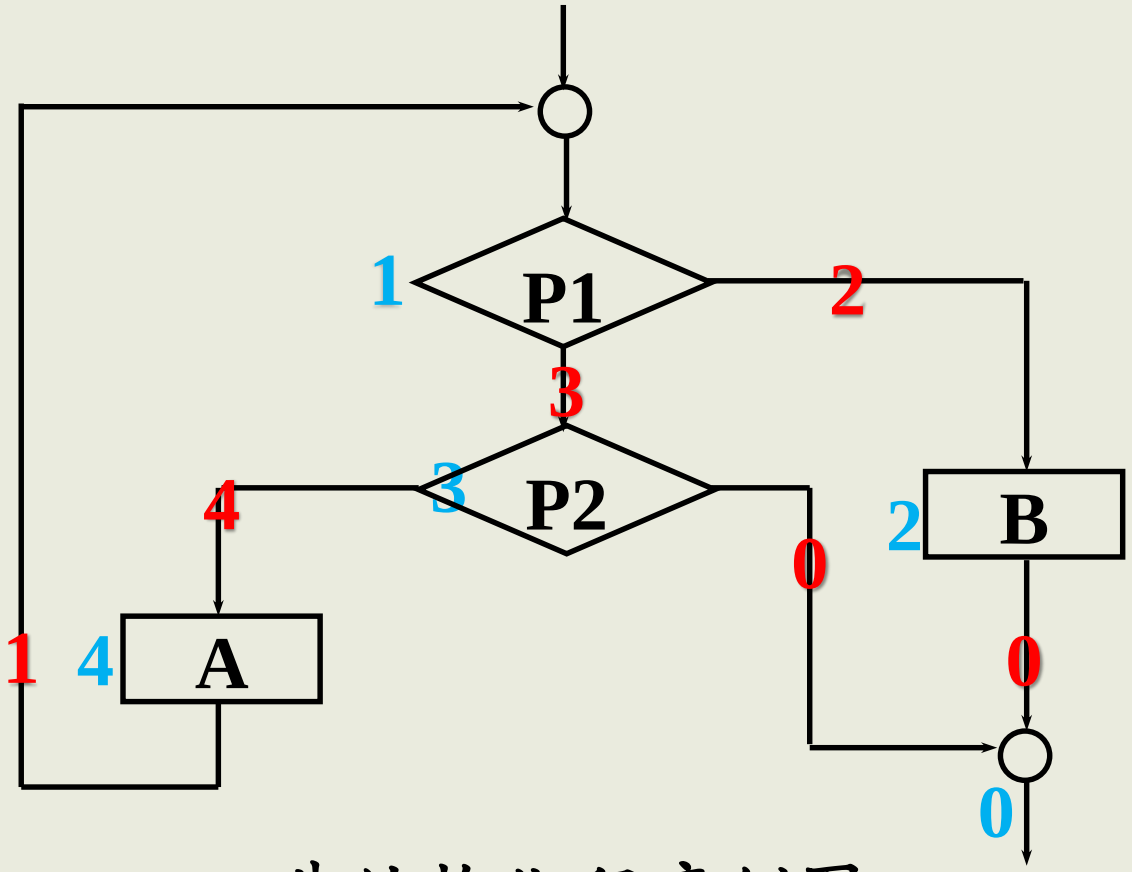
6.1.3 非结构化程序到结构化程序的转换

•例1：把图示的非结构化程序转换成结构化程序（用结构化定理证明过程提供的方法转换）



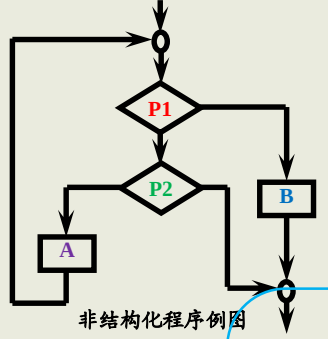
非结构化程序例图

• 1）进行结点及其出口线的编号；

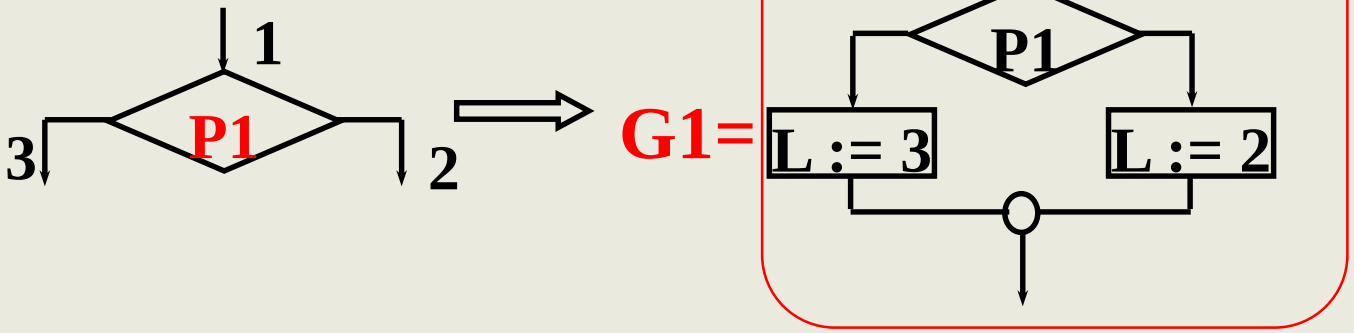


非结构化程序例图

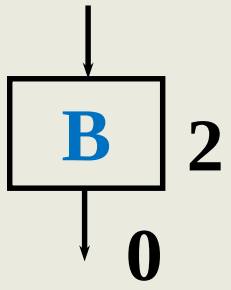
2）将图中的四个结点构造新的程序G1、G2、G3、G4;



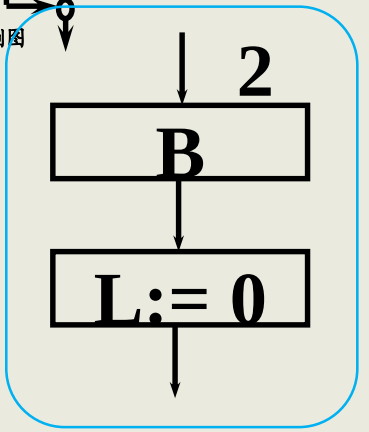
非结构化程序例图

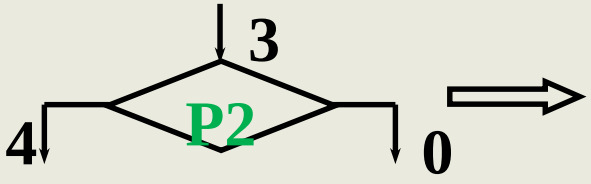


G1=

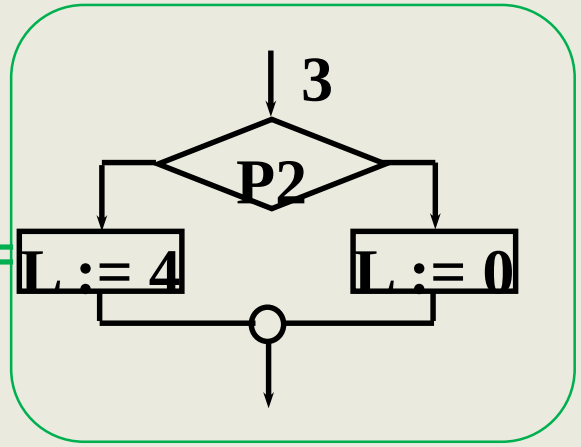


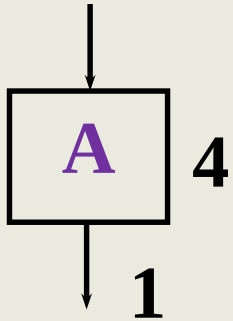
G2 =



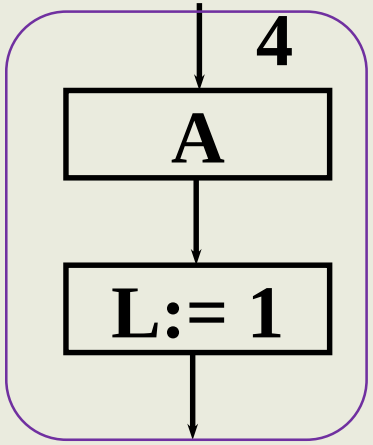


G3 =

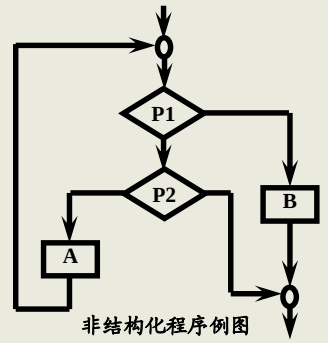


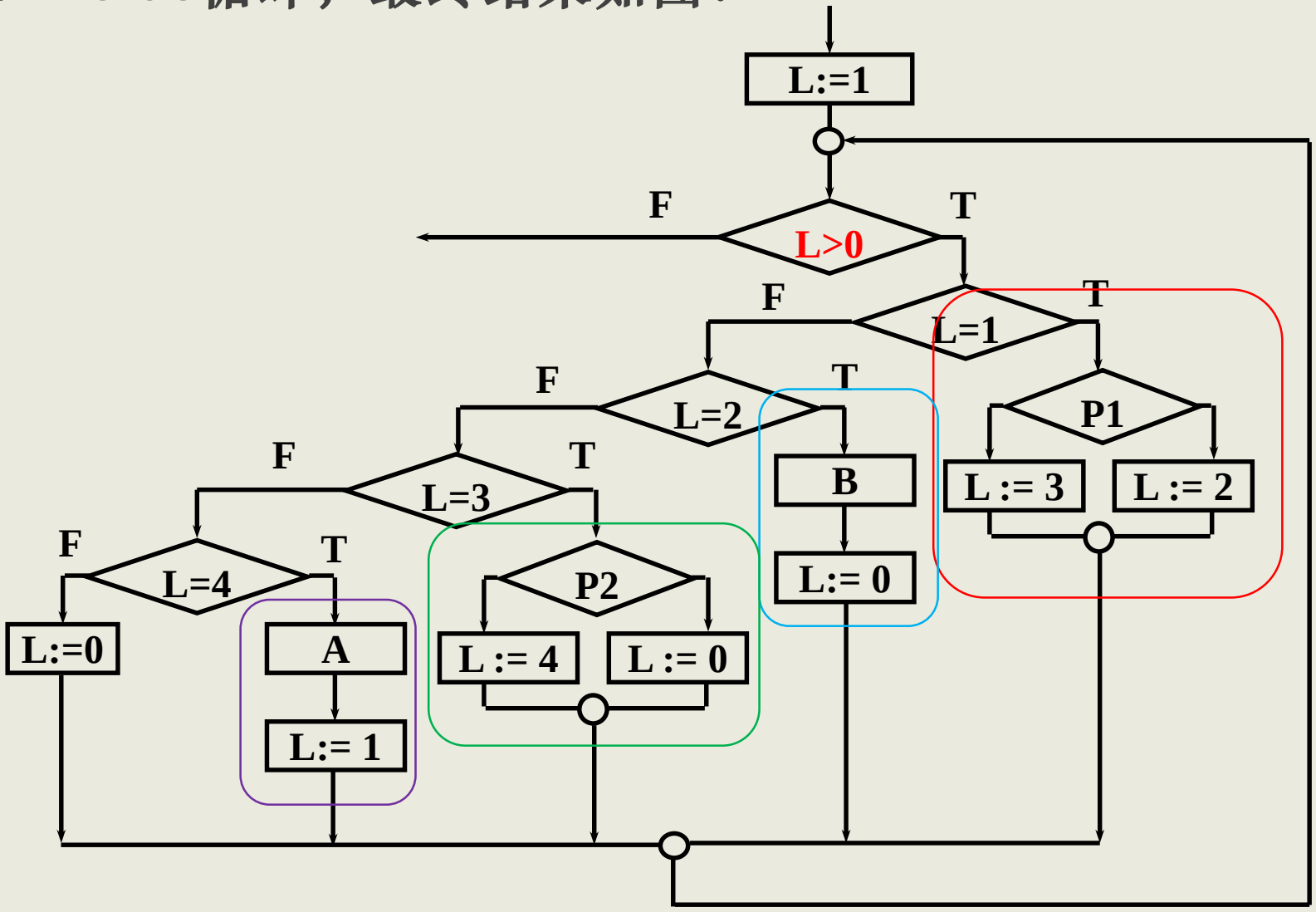


G4 =



- 3）利用得到的G1、G2、G3、G4按介绍的方法构造一个While-do循环，最终结果如图：





转化后的结构化程序图

第 19 卷 第 2 期 延安大学学报(自然科学版) Vol
2000 年 6 月 Journal of Yanan University (Natural Science Edition) June 2000

窗口弹出 ×

非结构化程序到结构化程序的转化[Ξ]

白晓虹[1], 刘东玲[2], 马 燕[1]
(1 延安大学, 陕西 延安 716000; 2 西安飞机工业公司 61 厂, 陕西 西安 710089)

摘 要: 通过对非结构化程序与结构化程序的比较, 讨论了结构化程序的优良特点, 并归纳总结了三种非结构化程序到结构化程序的转化方法.

关键词: 结构化程序; 函数结点; 正规程序; 程序函数
中图分类号: O 244 文献标识码: A 文章编号: 10042602X (2000) 0220027206

随着计算机硬件技术的不断发展, 程序设计方法也在不断地发展. 程序设计方法的发展过程, 大体上可以简单地根括为: 从语言发展的角度看, 由机器语言、汇编语言发展为高级程序设计语言; 从程序设计方法的角度看, 从手工艺式的设计方法发展为结构化程序设计、面向对象程序设计、可视化程序设计等[1,2]. 从手工艺式的设计方 法到结构化程序设计方法的这一发展是程序设计方法论上一次质的飞跃.

1 结构化程序设计方法的产生

结构化程序设计方法的产生, 是由 20 世纪 60 年代末 70 年代初爆发的"软件危机"所引起的. 60 年代末 70 年代初, 随着计算机硬件技术的发展和应用范围的不断扩大, 人们已经开发了各种大型软件系统, 如操作系统、数据库管理系统、航空和旅馆订票系统等. 开发这些大型软件系统, 编制周期长, 工作量大, 并且由于传统程序设计方法的局限性, 设计出的软件系统可靠性差, 往往隐藏着许多错误, 软件的开发和维护过程中遇到了一系列严重的问题. 一方面客观上需要研制大量大型软件系统; 另一方面, 按照原有的方法研制软件周期长、可靠性差、维护困难, 已不能适应新形势的要求, 这就是"危机"之所在.

面对这种严峻的现实, 软件人员不得不重新考虑过去所忽视的程序设计中的一些基本问

3 非结构化程序到结构化程序的转化

含有 GO TO 语句的非结构化程序, 到底能否消去 GO TO 语句转化为一个结构良好的结构化程序? 或者说一个任意的正规程序能否用一个和它完全等价的结构化程序来表示? 结构化定理对这一问题给出了肯定的回答[1].

3.1 程序函数

已知一正规程序 P, 对于每一个初始的数据状态 X, 若程序是终止的, 那么有确定的最终数据状态 Y. 如果对于每一个给定的 X, 值 Y 是唯一的, 那么所有的有序对的集合{(x, y)}定义了一个函数, 称这个函数为程序 P 的程序函数.

例如, 若程序 P 由语句

if x≤y then z = 23 x+ 5 else z = y

构成, 则它的程序函数为{(x, y, z)|x≤y→z= 23 x+ 5 ∨ x>y→z = y}

3.2 函数等价

如果程序 P_1 和程序 P_2 有相同的程序函数, 则称它们是函数等价的, 简称为等价.

3.3 结构化定理

任一正规程序都可以函数(语义)等价于一个由基集合{顺序, if—then—else, while—do}产生的结构化程序.

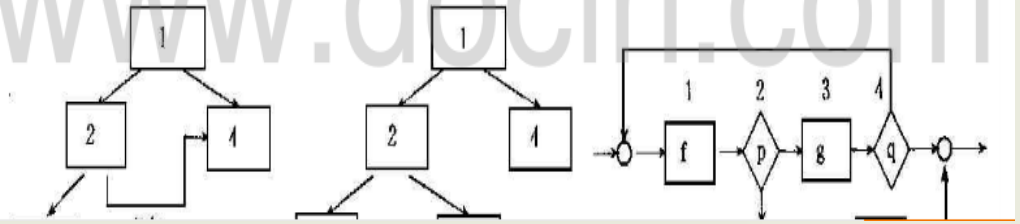
结构化定理表明, 任何程序都可以用顺序、条件和循环三种结构表示出来, 使之成为好结构的程序.

4 非结构化程序转化为结构化程序的一般方法

通常, 可以使用如下三种方法将一个非结构化程序转化为结构化程序.

4.1 重复编码法

如图(5)转化为图(6)



主要内容

6.1 结构程序设计

6.2 人机界面设计

6.3 过程设计的工具

6.4 面向数据结构的设计方法

6.5 程序复杂程度的定量度量

6.2 人机界面设计

6.2.1 设计问题

- **1. 系统响应时间；**
- **2. 用户帮助；**
- **3. 出错信息处理；**
- **4. 命令交互**

④ 命令交互。

许多高级用户仍然偏爱面向命令行的交互方式

在提供命令交互方式时，必须考虑下列设计问题。

(1) 是否每个菜单选项都有对应的命令?

(2) 采用何种命令形式?有3种选择：控制序列(例如，Ctrl+P)，功能键和输入命令。

(3) 学习和记忆命令的难度有多大?忘记了命令怎么办?

(4) 用户是否可以定制或缩写命令?

在越来越多的应用软件中，人机界面设计者都提供了“命令宏机制”。

在理想的情况下，所有应用软件都有一致的命令使用方法。

6.2 人机界面设计

6.2.2.设计过程

用户界面设计是一个迭代的过程，通常先创建设计模型，再用原型实现这个设计模型，并由用户试用和评估，然后根据用户意见进行修改。

建立起用户界面的原型，就必须对它进行评估，评估可以是非正式的也可以使正式的。

用户界面的评估周期

6.2 人机界面设计

创建了用户界面的设计模型之后，可以运用下述评估标准对设计进行早期复审。

- (1) 系统及其界面的规格说明书的长度和复杂程度，预示了用户学习使用该系统所需要的工作量。
- (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) 设计模型中包含的动作、命令和系统状态的数量，预示了用户学习使用该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及用户接受该界面的程度。

6.2 人机界面设计

6.2.3 人机界面设计指南

① **一般交互指南：**涉及信息显示、数据输入和系统整体控制

- 保持一致性。
- 提供有意义的反馈。
- 在执行有较大破坏性的动作之前要求用户确认。
- 允许取消绝大多数操作。
- 减少在两次操作之间必须记忆的信息量。
- 提高对话、移动和思考的效率。
- 允许犯错误。
- 按功能对动作分类，并据此设计屏幕布局。
- 提供对用户工作内容敏感的帮助设施
- 用简单动词或动词短语作为命令名。

6.2 人机界面设计

② **信息显示指南：**多种不同方式“显示”信息：用文字、图形和声音；按位置、移动和大小；使用颜色、分辨率和省略。

- 只显示与当前工作内容有关的信息。
- 不要用数据淹没用户，应该用便于用户迅速吸取信息的方式来表示数据。
- 使用一致的标记、标准的缩写和可预知的颜色。
- 允许用户保持可视化的语境。
- 产生有意义的出错信息。
- 使用大小写、缩进和文本分组以帮助理解。
- 使用窗口分隔不同类型的信息。
- 使用“模拟”显示方式表示信息，以使信息更容易被用户提取。
- 高效率地使用显示屏。

6.2 人机界面设计

③ **数据输入指南**：用户的大部分时间用在选择命令、输入数据和向系统提供输入。

- 尽量减少用户的输入动作。
- 保持信息显示和数据输入之间的一致性。
- 允许用户自定义输入。
- 交互应该是灵活的，并且可调整成用户最喜欢的输入方式。
- 使在当前动作语境中不适用的命令不起作用。
- 让用户控制交互流。
- 对所有输入动作都提供帮助。
- 消除冗余的输入。
- 高效率地使用显示屏。

主要内容

6.1 结构程序设计

6.2 人机界面设计

6.3 过程设计的工具

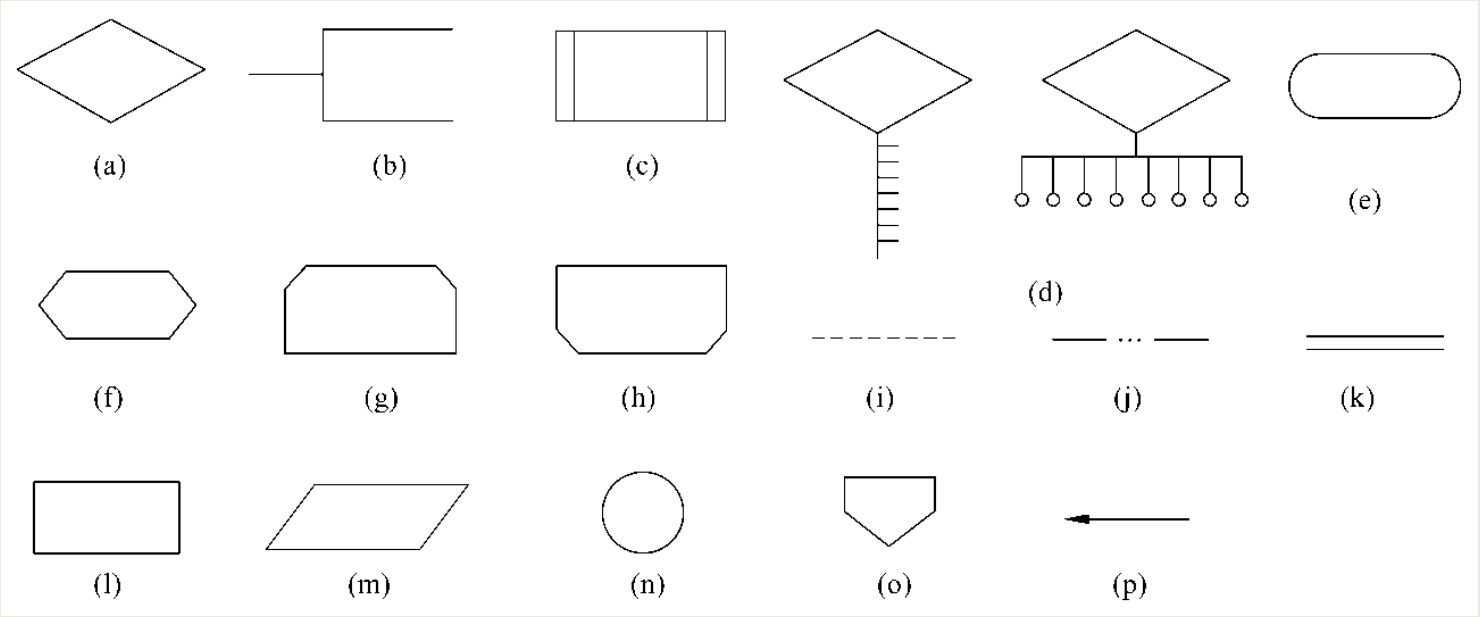
6.4 面向数据结构的设计方法

6.5 程序复杂程度的定量度量

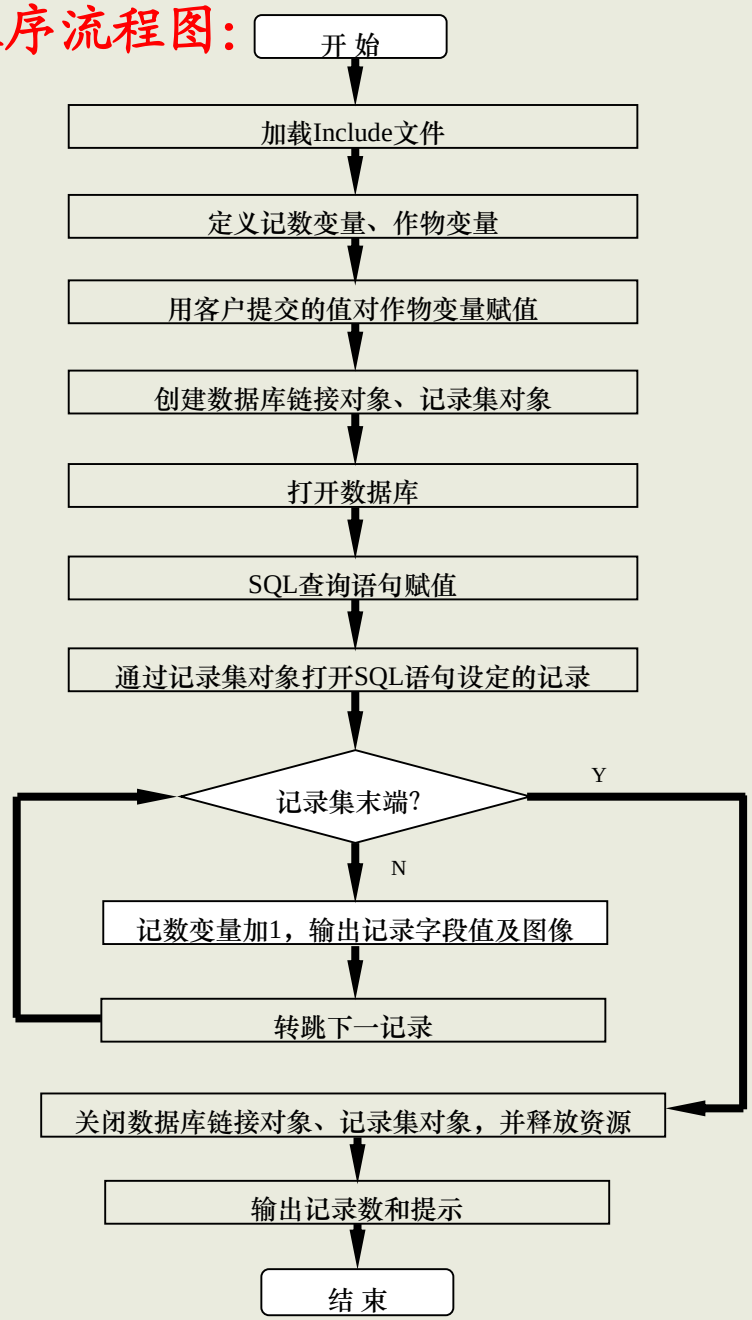
6.3 过程设计的工具

6.3.1 程序流程图

程序流程图又称为程序框图，它是使用最广泛的描述过程设计的方法。程序流程图中使用的符号(a) 选择(分支)； (b) 注释； (c) 预先定义的处理； (d) 多分支； (e) 开始或停止； (f) 准备； (g) 循环上界限； (h) 循环下界限； (i) 虚线； (j) 省略符； (k) 并行方式； (l) 处理； (m) 输入输出； (n) 连接； (o) 换页连接； (p) 控制流



ASP检索程序流程图：



总的趋势是越来越多的人不再使用程序流程图了。程序流程图的主要缺点如下。

- (1) 程序流程图本质上不是逐步求精的好工具，它诱使程序员过早地考虑程序的控制流程，而不去考虑程序的全局结构。
- (2) 程序流程图中用箭头代表控制流，因此程序员不受任何约束，可以完全不顾结构程序设计的精神，随意转移控制。
- (3) 程序流程图不易表示数据结构。

6.3 过程设计的工具

6.3.2 盒图

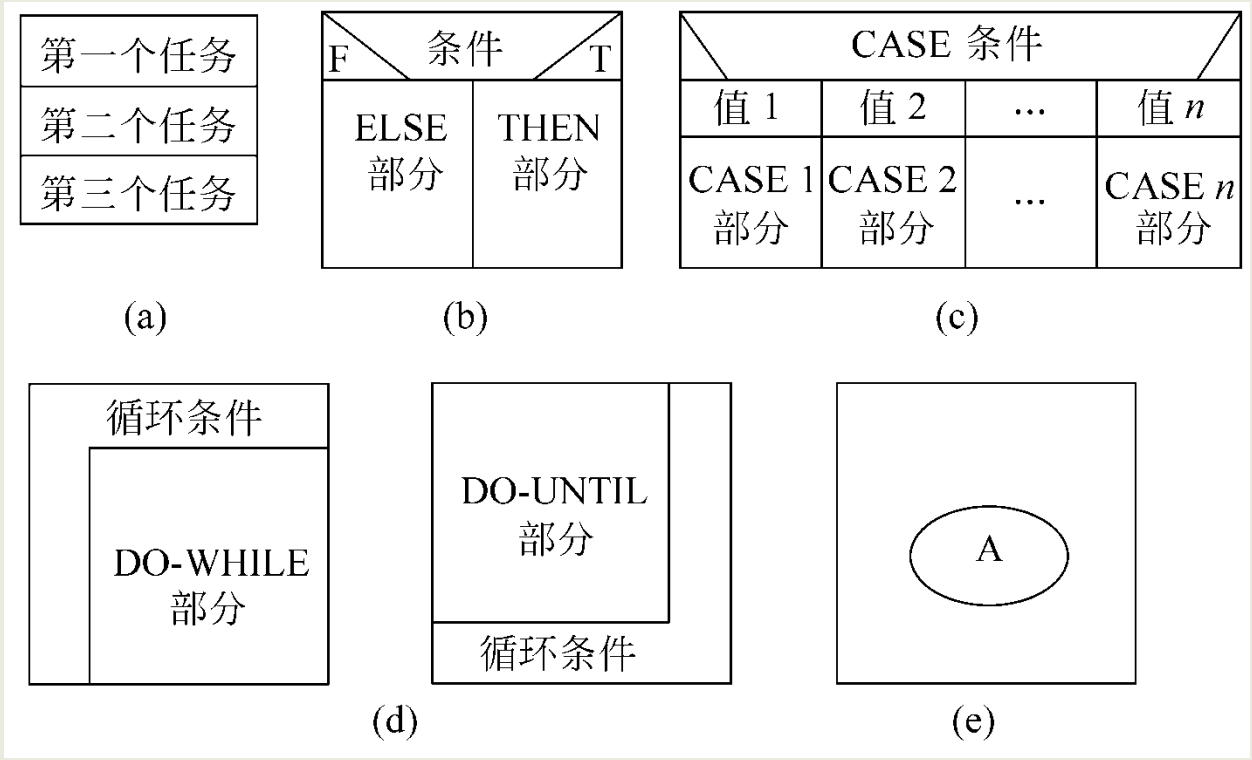
出于要有一种不允许违背结构程序设计精神的图形工具的考虑，Nassi和Shneiderman提出了盒图，又称为N-S图。它有下述特点。

- (1) 功能域(即一个特定控制结构的作用域)明确，可以从盒图上一眼就看出来。
- (2) 不可能任意转移控制。
- (3) 很容易确定局部和全程数据的作用域。
- (4) 很容易表现嵌套关系，也可以表示模块的层次结构。

6.3 过程设计的工具

图给出了结构化控制结构的盒图表示，也给出了调用子程序的盒图表示方法。其中

基本符号(a) 顺序； (b) IF_THEN_ELSE型分支； (c) CASE型多分支； (d) 循环； (e) 调用子程序A



6.3 过程设计的工具

6.3.3 PAD图

PAD是问题分析图(problem analysis diagram)的英文缩写，用二维树形结构的图来表示程序的控制流。

基本符号

(a)顺序(先执行P1后执行P2)；

(b)选择(IF C THEN P1 ELSE P2);

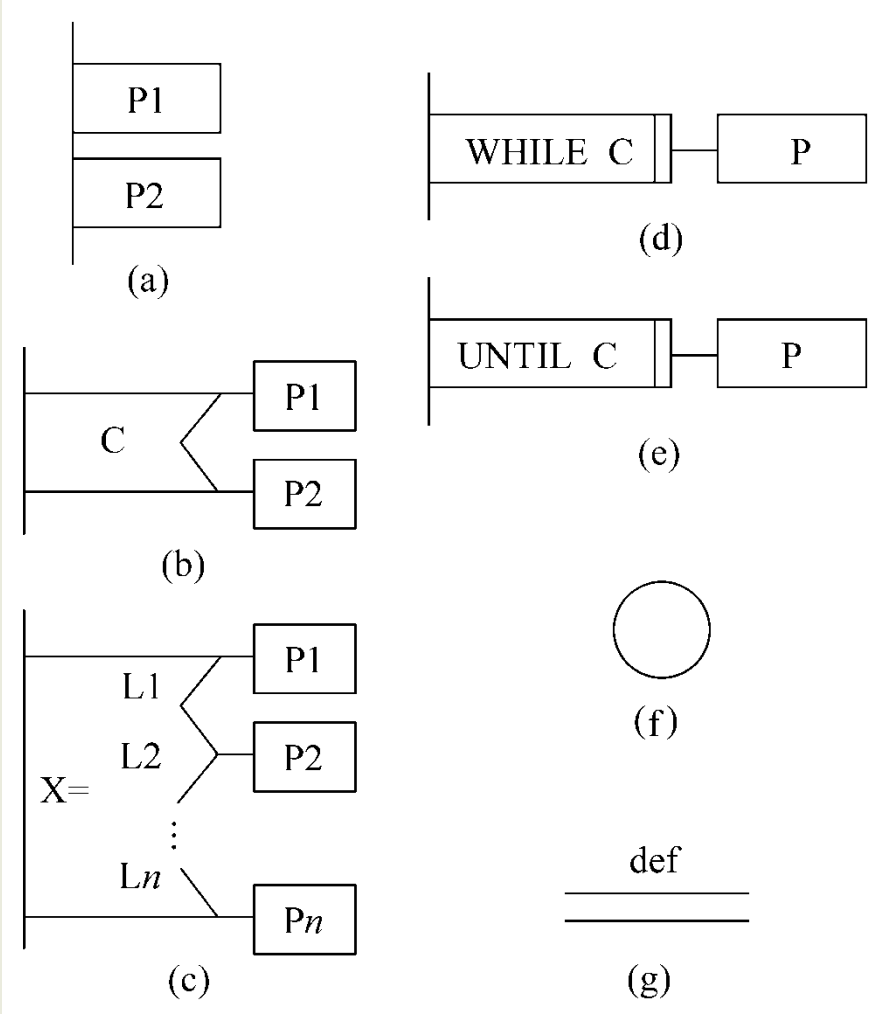
(c)CASE型多分支；

(d)WHILE型循环(WHILE C DO P);

(e)UNTIL型循环(REPEAT P UNTIL C)；

(f)语句标号；

(g)定义



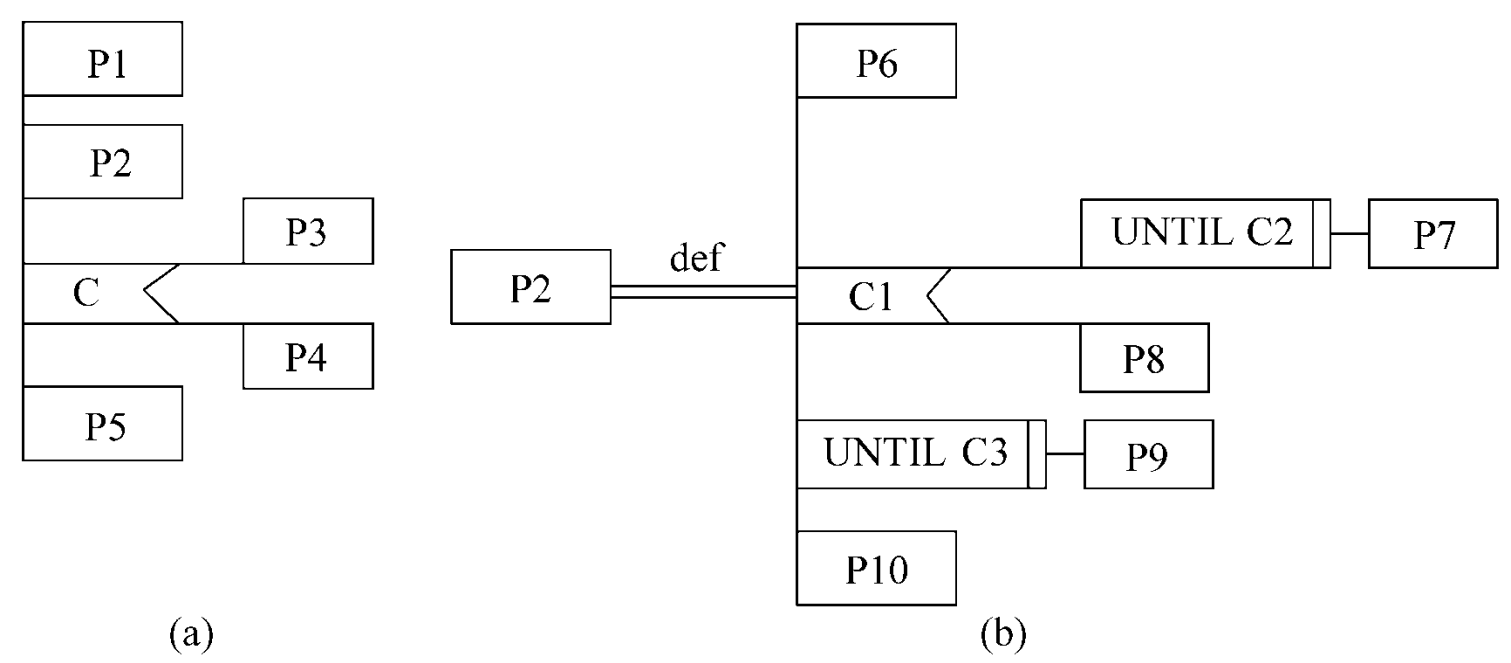
6.3 过程设计的工具

PAD图的主要优点如下：

- (1) 使用表示结构化控制结构的PAD符号所设计出来的程序必然是结构化程序。
- (2) PAD图所描绘的程序结构十分清晰。
- (3) 用PAD图表现程序逻辑，易读、易懂、易记。
- (4) 容易将PAD图转换成高级语言源程序，这种转换可用软件工具自动完成，从而可省去人工编码的工作，有利于提高软件可靠性和软件生产率。
- (5) 即可用于表示程序逻辑，也可用于描绘数据结构。
- (6) PAD图的符号支持自顶向下、逐步求精方法的使用。

6.3 过程设计的工具

例子



(a) 初始的PAD图； (b) 使用def符号细化处理框P2

6.3 过程设计的工具

6.3.4 判定表

判定表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。

判定表由4部分组成，

1. 左上部列出所有条件（条件桩）
2. 左下部是所有可能做的动作（动作桩）
3. 右上部是表示各种条件组合的一个矩阵
4. 右下部是和每种条件组合相对应的动作。

		1	2	3	4	5	6	7	8
问题	觉得疲倦？	Y	Y	Y	Y	N	N	N	N
	感兴趣吗？	Y	Y	N	N	Y	Y	N	N
	糊涂吗？	Y	N	Y	N	Y	N	Y	N
建议	重读					√			
	继续						√		
	跳下一章							√	√
	休息	√	√	√	√				

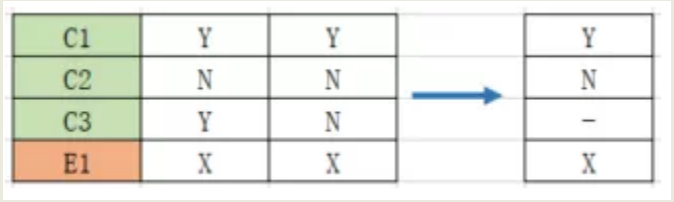
判定表右半部的每一列实质上是一条规则，规定了与特定的条件组合相对应的动作。

如果有3个条件，每个条件有2个取值，则有 $2x2x2 = 2^3 = 8$ 种规则，判定表中则有 8 列。

判定表中，规则的数量是否一定写成2的n次方？

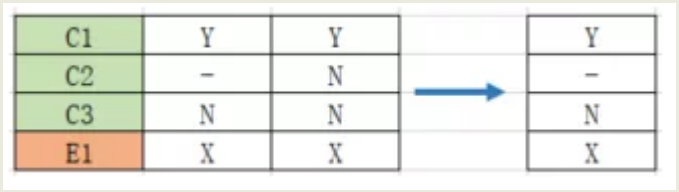
判定表的化简与合并

- 化简工作是以合并相似规则为目标。如果表中有两条或多条规则具有相同的动作，并且其条件项之间存在极为相似的关系，即可将其合并。
- 下图中，E1的取值都是X，且C1和C2的取值相同，C3的取值不同，那么可以得出，只要C1和C2取值为Y/N，无论C3怎么取值，结果都是X



C1	Y	Y		Y
C2	N	N	→	N
C3	Y	N		-
E1	X	X		X

- 下图中E1的取值为X，C1和C3的取值相同，且第一列的C2取值包含了第二列的C2取值范围，所以第二列是重复的，可以去掉第二列。



C1	Y	Y		Y
C2	-	N	→	-
C3	N	N		N
E1	X	X		X

判定表的化简与合并

	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
行李重量W≤30kg	T	F	F	F	F	F	F	F	F
免费	X								
(W-30)X2				X					
(W-30)X3					X				
(W-30)X4		X						X	
(W-30)X6			X						X
(W-30)X8						X			
(W-30)X12							X		

从表可以看出，只要行李重量不超过30kg，不论这位乘客持有何种机票，是中国人还是外国人，是残疾人还是正常人，一律免收行李费，这就是表右部第一列(规则1合并了8个规则)表示的内容。

当行李重量超过30kg时，根据乘客机票的等级、乘客国籍及是否残疾人而使用不同算法计算行李费，这就是从规则2到规则9所表示的内容。

从上面这个例子可以看出，判定表能够简洁而又无歧义地描述处理规则。

但是用判断表描述循环比较困难。有时，判断表可以和结构化英语结合起来使用。

课堂练习 打印机判定表的合并化简

- 在日常工作中，我们经常会使用到打印机。那么如何才能正常使用打印机呢？按照正常流程，我们需要确保电脑有打印机的驱动，打印机正常工作、打印机的纸张充足、打印机墨粉充足才能满足打印。请对以下打印机判定表进行合并和化简

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
条件桩	C1：打印机驱动	1	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0
	C2：打印机上电	1	1	1	0	1	1	0	0	1	0	1	0	0	1	0	0
	C3：纸张	1	1	0	1	1	0	1	0	0	1	1	0	1	0	0	0
	C4：墨粉	1	0	1	1	1	0	0	1	1	1	0	1	0	1	0	0
动作桩	E1：正常打印	1															
	E2：提示驱动异常					1				1	1	1	1	1	1		1
	E3：提示无法连接打印机				1			1	1							1	
	E4：提示没有纸张			1			1										
	E5：提示没有墨粉		1														

答案

- 化简
  - 列 5/9/10/11/12/13/15/16 中，结果都是E2 = ,1，且不管C2/C3/C4的值是什么，只要C1 = 0，所以可以合并这几列。
  - 列 4/7/8/15中，结果都是 E3 = 1，不管 C3和C4取什么值，只要 C1 = 1和 C2 = 0，所以合并这几列。
  - 列 3/6 中，结果都是 E4 = 1， 不管 C4取什么纸，只要 C1=1 、 C2=1 和 C3=0，所以这两列可以合并。

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
条件桩	C1：打印机驱动	1	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0
	C2：打印机上电	1	1	1	0	1	1	0	0	1	0	1	0	0	1	0	0
	C3：纸张	1	1	0	1	1	0	1	0	0	1	1	0	1	0	0	0
	C4：墨粉	1	0	1	1	1	0	0	1	1	1	0	1	0	1	0	0
动作桩	E1：正常打印	1															
	E2：提示驱动异常					1				1	1	1	1	1	1		1
	E3：提示无法连接打印机				1			1	1							1	
	E4：提示没有纸张			1			1										
	E5：提示没有墨粉		1														

		1	2	14	3&6	4&7&8*15	5&9&10&11&12&13&16
条件桩	C1：打印机驱动	1	1	0	1	1	0
	C2：打印机上电	1	1	1	1	0	-
	C3：纸张	1	1	0	0	-	-
	C4：墨粉	1	0	1	-	-	-
动作桩	E1：正常打印	1					
	E2：提示驱动异常			1			1
	E3：提示无法连接打印机					1	
	E4：提示没有纸张				1		
	E5：提示没有墨粉		1				

6.3 过程设计的工具

6.3.5 判定树

判定树是判定表的变种，它也能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。

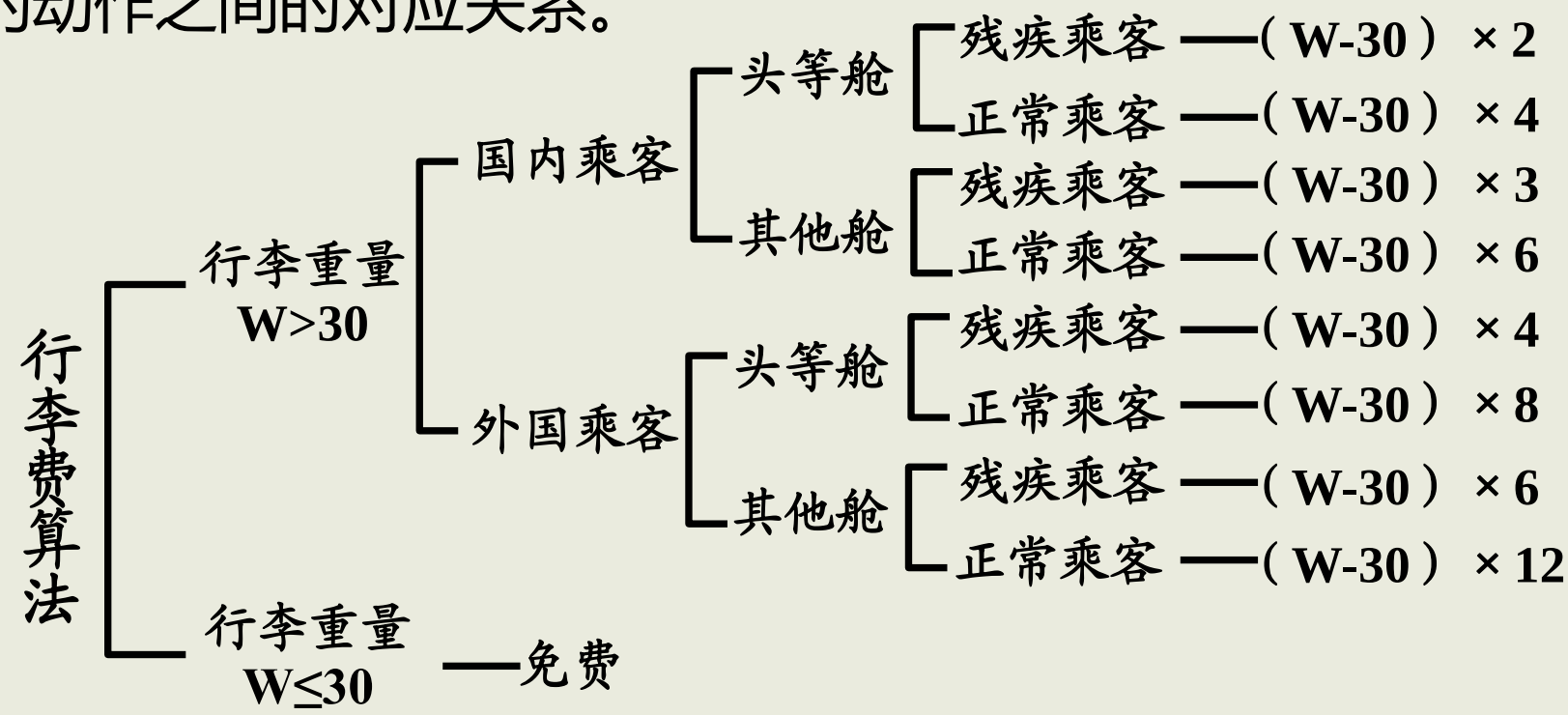


图 用判定树表示计算行李费的算法

6.3 过程设计的工具

6.3.6 过程设计语言

过程设计语言（PDL）也称为伪码。是用正文形式表示数据和处理过程的设计工具。

PDL有下述特点：

(1)**关键字的固定语法**，它提供了结构化控制结构、数据说明和模块化的特点。如，if…fi(或endif)等

(2) **自然语言的自由语法**，它描述处理特点。

(3) **数据说明**的手段。应该既包括简单的数据结构(例如纯量和数组)，又包括复杂的数据结构(例如，链表或层次的数据结构)。

(4) **模块定义和调用**的技术，应该提供各种接口描述模式。

如：if I>0 then

执行订单数据输入模块

else

报告出错信息

end if

6.3 过程设计的工具

PDL有下述优点：

(1) 可以作为注释直接插在源程序中间。

(2) 可以使用普通的正文编辑程序或文字处理系统，很方便地完成PDL的书写和编辑工作。

(3) 已经有自动处理PDL的程序存在，而且可以自动由PDL生成程序代码。

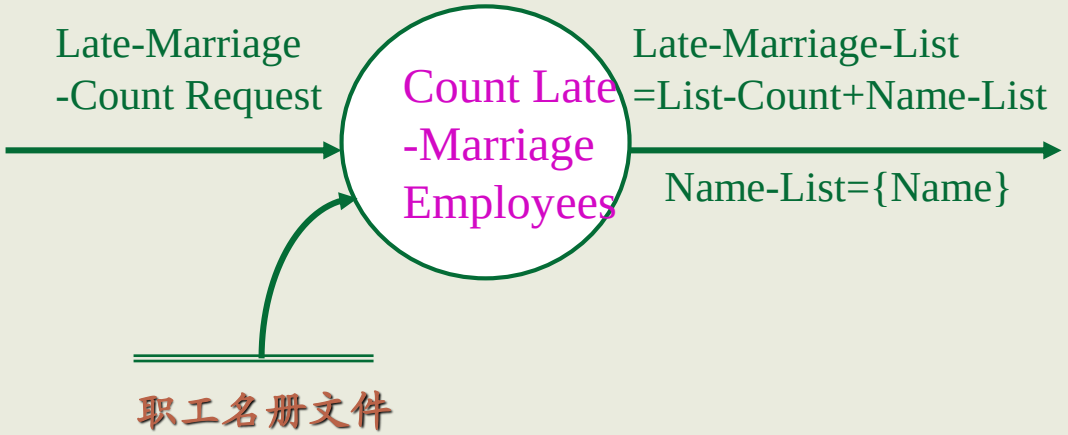
PDL的缺点是不如**图形工具**形象直观，描述复杂的条件组合与动作间的对应关系时，不如**判定表**清晰简单。

结构化语言 自然语言+结构化形式

选择结构	如果<条件> <策略>		If <condition> <policy>	
	如果<条件> 则 <策略1> 否则 <策略2>	情况1 <条件> <策略1> … … 情况n <条件> <策略n>	If <condition> then <policy1> Otherwise <policy2>	case 1 <condition> <policy1> … … case n <condition> <policyn>
循环结构	对 … , <策略>	重复以下 <策略> 直至 <条件>	For each … , <policy>	Repeat the following: <policy> Until <condition>

英文

例1：请写出下列在 DFD 图中给出的 “统计晚婚职工”的加工说明



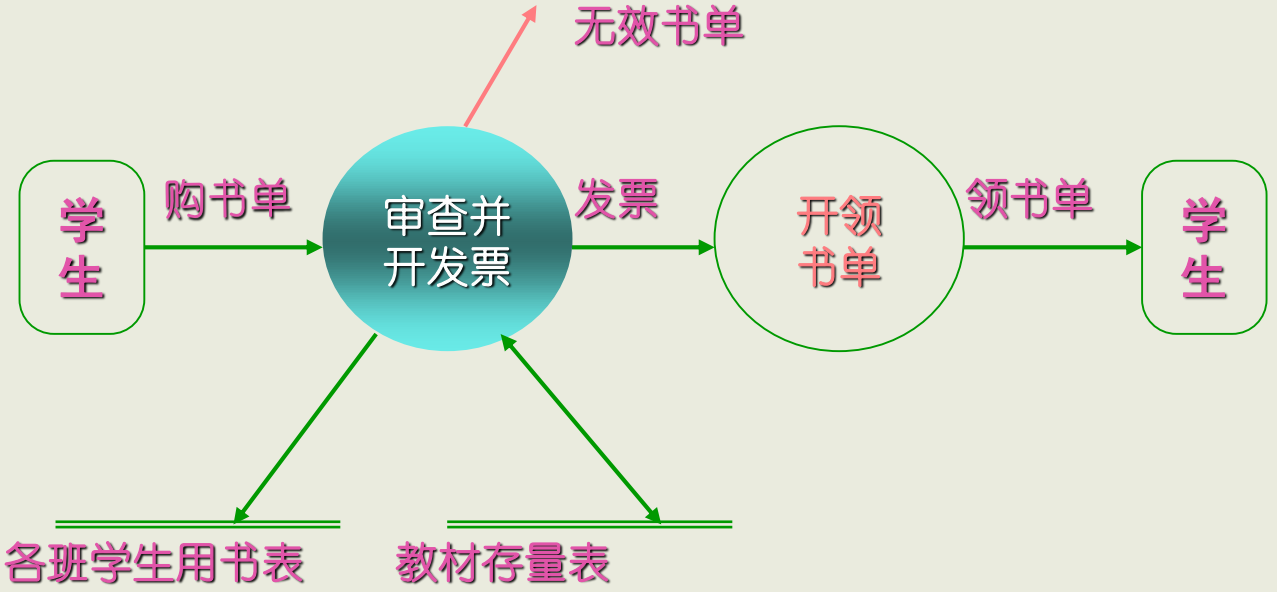
该加工的任务是，每当接到统计晚婚职工的请求时，便将职工名册中年瞒30岁的未婚男职工和年满26岁的未婚女职工的人数统计出来，然后列出他们的名字。

结构化 英文:

Count Late-Marriage Employees Policy
For each Late-Marriage-Count request:
  Repeat the following;
    Access the staffs-Record.
    If status is single,
      If sex is maie and Age is over 30 or
        sex is female and Age is over 26
        Write Name to Name-List.
        Increment List-Count.
  Until there are no more Staff-Records.
Combine List-Count and Name-List.
Write Up Late-Marriage-List.

结构化 中文

例2：请为下列DFD中的“审查并开发票”加工点写中文伪码



----学生购买教材的 系统逻辑模型

对每张购书单

把学生学号和姓名写到发票上

按购书单上学生的年级和系、专业与班号

检索“各班学生用书表”文件，获得该生当年的书单

对 购书单上的每一书号

如果 书单上无此书号

则 把书号写到出错通知单上

否则

按书号检索 “教材存量表”文件，

从而获得该书的单价与库存量

如果 库存量〈 购书单的数量

则 将书号写到出错通知单上

否则

将书号、单价、数量、总价等项写入到发票上；

更新存书量，并写回“教材存量表” 文件；

累计书费合计

把书费合计写到发票上

主要内容

6.1 结构程序设计

6.2 人机界面设计

6.3 过程设计的工具

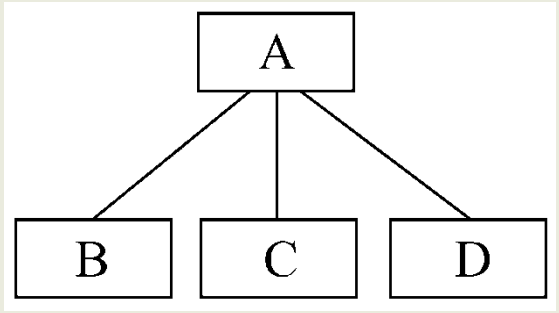
6.4 面向数据结构的设计方法

6.5 程序复杂程度的定量度

6.4 面向数据结构的设计方法

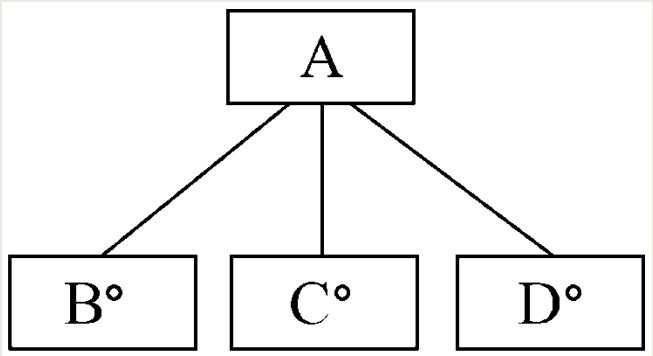
6.4.1 Jackson图

顺序结构



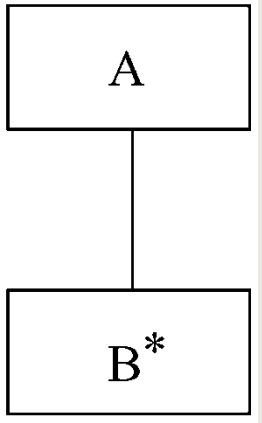
A由B、C、D 3个元素顺序组成(每个元素只出现一次，出现的次序依次是B、C和D)

选择结构



根据条件A是B或C或D中的某一个(注意，在B、C和D的右上角有小圆圈做标记)

重复结构



A由B出现N次(N≥0)组成(注意，在B的右上角有星号标记)

总结了COBOL事务处理程序中的开发方法后而发展起来的, 欧洲较流行, 特别适合设计企业业务管理类的数据处理系统

重点不是自顶向下、逐步求精, 而是在数据结构的基础上进行构造, 根据I/O的数据结构建立程序结构

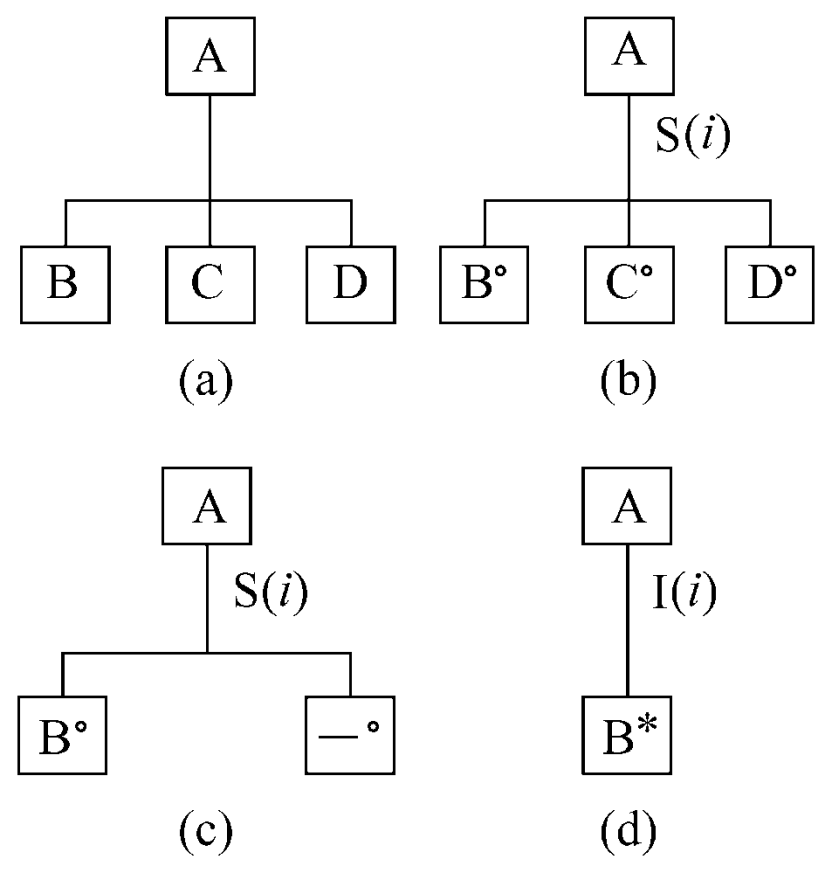
6.4 面向数据结构的设计方法

6.4.2 改进的Jackson图

Jackson图的不足:

1. 用其表示选择或重复时, 选择条件或循环结束条件不能直接在图上表示出来, 影响了图的表达能力

2. 不易在行式打印机上打印



(a) 顺序结构，B、C、D中任一个都不能是选择出现或重复出现的数据元素(即不能是右上角有小圆圈或星号标记的元素)；

(b) 选择结构，S右面括号中的数字i是分支条件的编号；

(c) 可选结构，A或者是元素B或者不出现；

(d) 重复结构，循环结束条件的编号为i。

6.4 面向数据结构的设计方法

6.4.3 Jackson法

Jackson结构程序设计方法基本上由下述5个步骤组成。

(1) 分析并确定输入数据和输出数据的逻辑结构，并用Jackson图描绘这些数据结构。

(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) 用下述3条规则从描绘数据结构的Jackson图导出描绘程序结构的Jackson图。

① 为每对有对应关系的数据单元，按照它们在数据结构图中的层次在程序结构图的相应层次画一个处理框

② 根据输入数据结构中剩余的每个数据单元所处的层次，在程序结构图的相应层次分别为它们画上对应的处理框。

③ 根据输出数据结构中剩余的每个数据单元所处的层次，在程序结构图的相应层次分别为它们画上

6.4 面向数据结构的设计方法

(4) 列出所有操作和条件(包括分支条件和循环结束条件)，并且把它们分配到程序结构图的适当位置。

(5) 用伪码表示程序。

Jackson方法中使用的伪码和Jackson图是完全对应的，

下面是和3种基本结构对应的伪码。

顺序结构

```
A seq
   B
   C
   D
A end
```

选择结构

```
A select cond1
    B
A or cond2
    C
A or cond3
    D
A end
```

重复结构

```
A iter until(或while) cond
    B
A end
```

6.4 面向数据结构的设计方法

下面结合一个具体例子进一步说明Jackson结构程序设计方法。

例：一个正文文件由若干个记录组成，每个记录是一个字符串。（**如：**

Record 1：How many stages are there in the traditional software development model?

Record 2： After entering the room, walk to the person sitting nearest to you and greet him/her with a “high five”.

Record 3： What are encapsulated into an object?

Record 4： What diagram is the following diagram? Simply describe the meaning of it.）

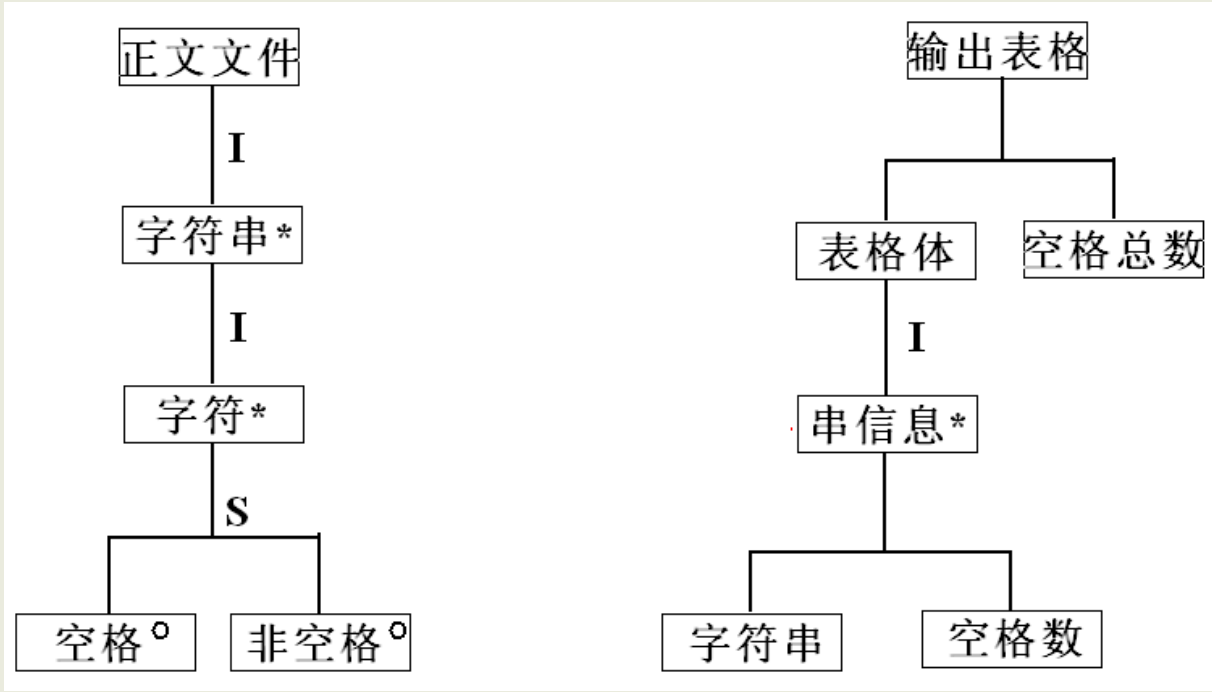
要求：1）统计每个记录中空格字符的个数，以及文件中空格字符的总个数。

2）输出数据格式是，每复制一行输入字符串之后，另起一行印出这个字符串中的空格数，最后印出文件中空格的总个数。

Jackson系统开发方法

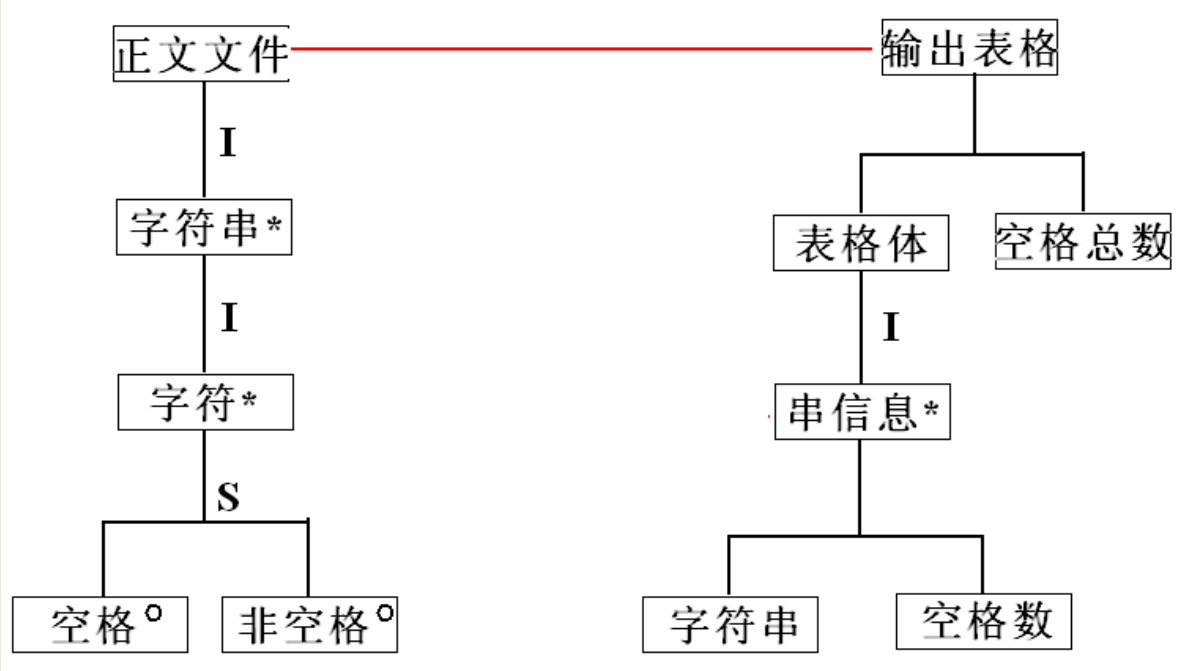
- 例：Jackson方法

- 第1步：分析并确定I/O数据的逻辑结构



Jackson系统开发方法

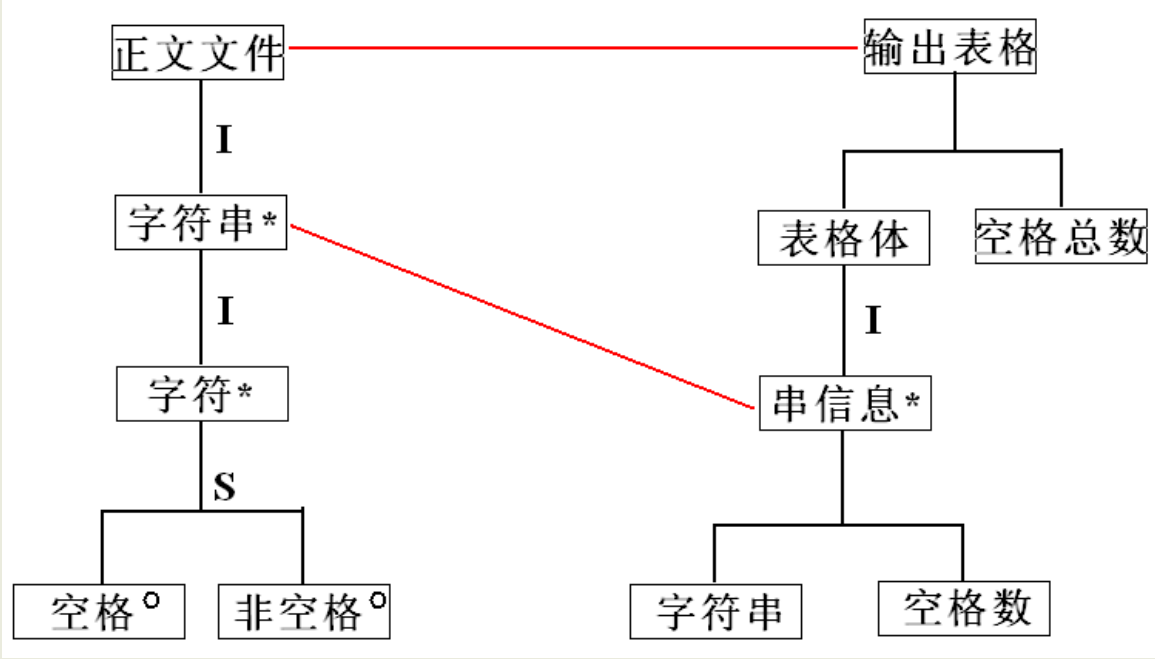
- 第2步：找出I/O数据结构中的对应数据单元



输出数据总由输入数据经处理得到，它们总有对应关系

Jackson系统开发方法

- 第2步：找出I/O数据结构中的对应数据单元



都是重复出现，且出现次序和重复次数都相同

Jackson系统开发方法

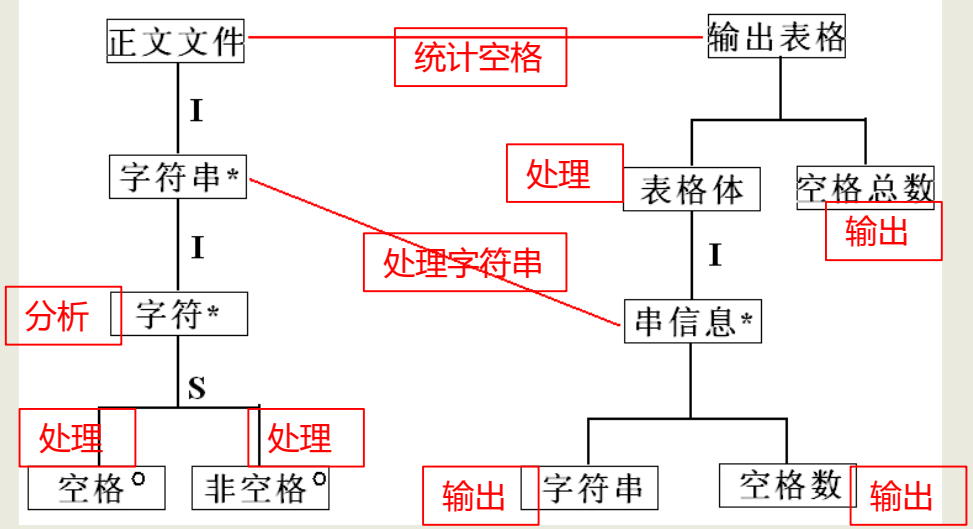
- 第3步：从数据结构图导出程序结构图

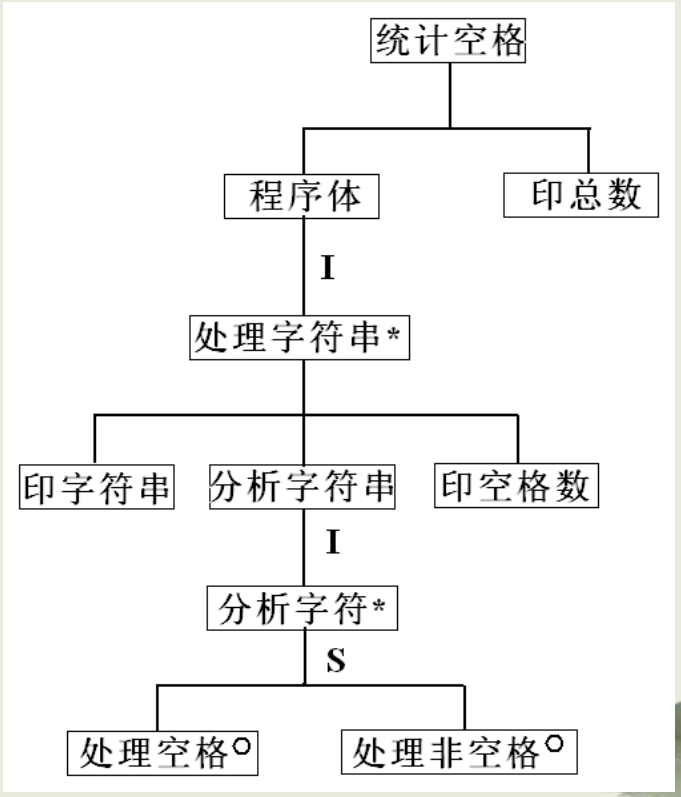
1) 为每对有对应关系的**数据单元**，按照它们在数据结构中的层次在程序结构图的相应层次画一个**处理框**；

2) 根据**输入数据结构中**剩余的每个数据单元所处的层次，在程序结构图的相应层次分别为它们画上对应的**处理框**；

3) 根据**输出数据结构中**剩余的每个数据单元所处的层次，在程序结构图的相应层次分别为它们画上对应的**处理框**；

注: 若这对数据单元在I/O数据结构中所处的层次不同，则和它们对应的处理框在程序结构图中所处的层次与它们中在数据结构图中层次较低的那个对应





Jackson系统开发方法

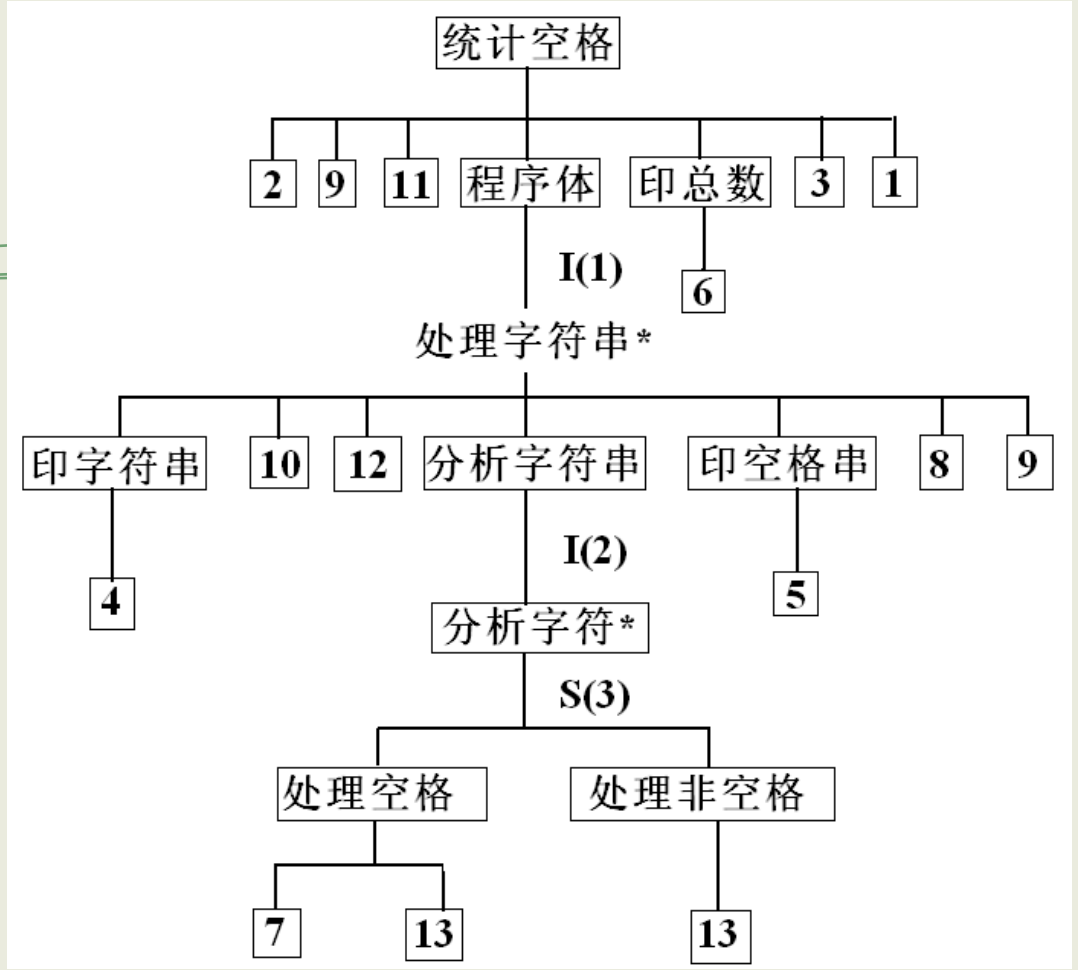
- 第4步：列出所有操作和条件，并把它们分配到程序结构的适当位置

1.停止　2.打开文件　3.关闭文件

4.印出字符串　　5.印出空格数目　6.印出空格总数

7.sum:=sum+1
8.totalsum:=totalsum+sum
9.读入字符串
10.sum:=0　　11.totalsum:=0
12.pointer:=1
13.pointer:=pointer+1

I(1)文件结束
I(2)字符串结束
S(3)字符是空格



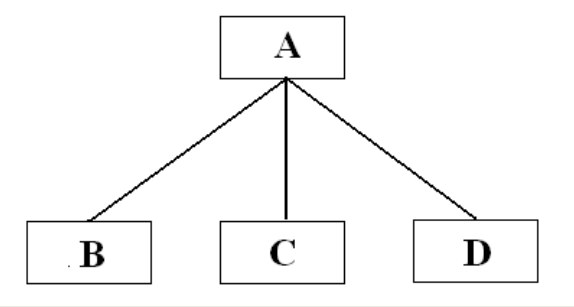
Jackson系统开发方法

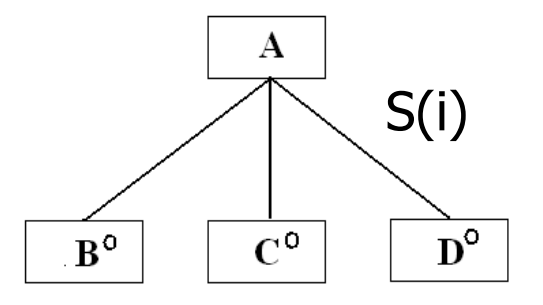
- 第5步：用伪码表示程序

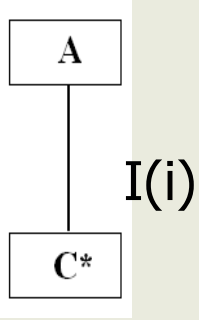
A seq
  B
  C
  D
A end

A select cond1
  B
A or cond2
  C
A or cond3
  D
A end

A iter untill (or while) cond
  C
A end







Jackson系统开发方法

- 第5步：用伪码表示程序

```
统计空格 seq
  打开文件
  读入字符串
  totalsum:=0
  程序体 iter until 文件结束
    处理字符串 seq
      印字符串 seq
        印出字符串
      印字符串 end
      sum:=0
      pointer:=1
      分析字符串 iter untill 字符串结束
        分析字符 select 字符是空格
          处理空格 seq
            sum:=sum+1
            pointer:=pointer+1
          处理空格 end
        分析字符 or 字符不是空格
          处理非空格 seq
            pointer:=pointer+1
          处理非空格 end
        分析字符 end
      分析字符串 end
      印空格数 seq
        印出空格数目
      印空格数 end
      totalsum:=totalsum+sum
      读入字符串
    处理字符串 end
  程序体 end
  印总数 seq
    印出空格总数
  印总数 end
  关闭文件
  停止
统计空格 end
```

主要内容

6.1 结构程序设计

6.2 人机界面设计

6.3 过程设计的工具

6.4 面向数据结构的设计方法

6.5 程序复杂程度的定量度量

6.5 程序复杂程度的定量度量

定量度量程序复杂程度的方法很有价值：

a) 把程序的复杂程度乘以适当常数即可估算出软件中错误的数量以及软件开发需要用的工作量，

b) 定量度量的结果可以用来比较两个不同的设计或两个不同算法的优劣；

c) 程序的定量的复杂程度可以作为模块规模的精确限度。

6.5 程序复杂程度的定量度量

6.5.1 McCabe方法

① 流图

McCabe方法根据程序控制流的复杂程度定量度量程序的复杂程度，这样度量出的结果称为程序的环形复杂度。

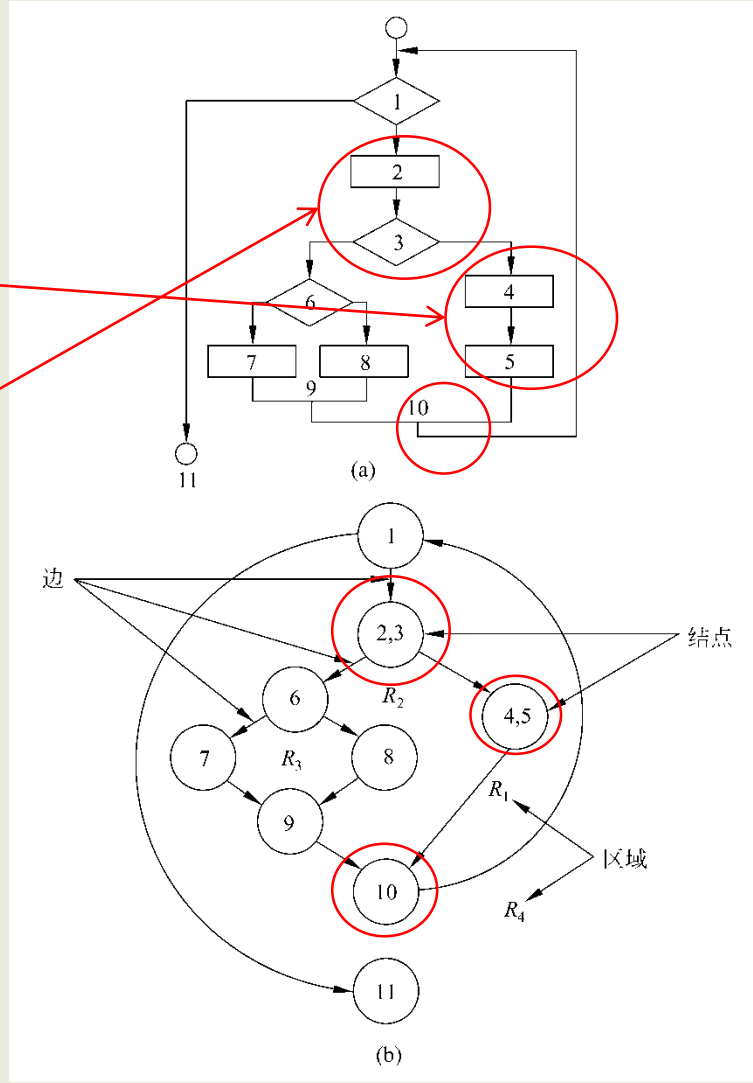
流图实质上是“退化了的”程序流程图，描绘程序的控制流程，不表现对数据的具体操作以及分支或循环的具体条件。

6.5 程序复杂程度的定量度量

① 流图（流程图到流图）

- 一个圆代表一条或多条语句；
- 一个顺序结构可以合并一个结点；
- 汇点也是结点；
- 一个顺序处理框序列和一个判断框可映射成一个结点。

- 流图中的箭头线称为边，代表控制流；
- 在流图中一条边必须终止于一个结点



6.5 程序复杂程度的定量度量

① **流图**（由PDL翻译成的流图）

- 一个圆代表一条或多条语句；
- 一个顺序结构可以合并一个结点；
- 汇点也是结点；
- 一个顺序处理框序列和一个判断框可映射成一个结点。

```
PDL
procedure:sort
    do while records remain
        read record;
        if record field 1=0
          then process record;
            store in buffer;
            incremert counter;
        elseif record field 2=0
          then reset counter;
          else process record;
            store in file;
        endif
        endif
    enddo
    end
```

do while records remain

read record;
if record field 1=0

elseif record field 2=0

then process record;
store in buffer; .
incremert counter;

else process record;
store in file;

then reset counter;

endif
endif

7b: enddo

8: end

6.5 程序复杂程度的定量度量

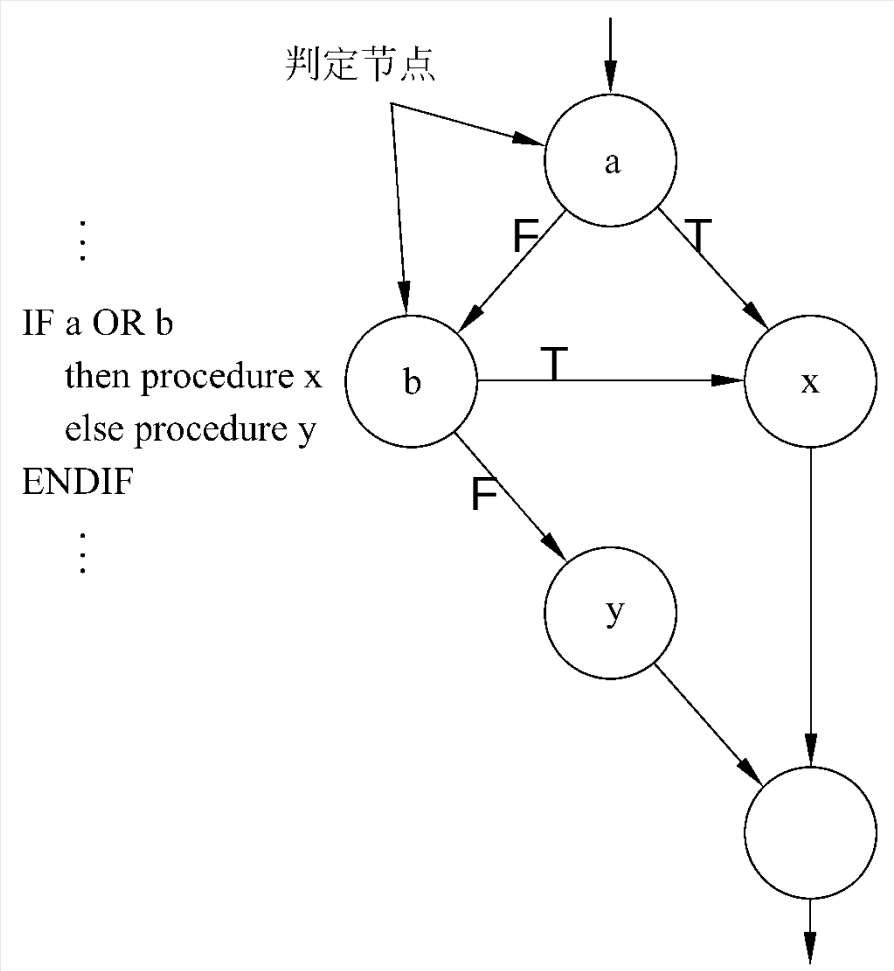
- 复合条件，就是在条件中包含了一个或多个布尔运算符(逻辑OR，AND，NAND，NOR)；
- 右图为a OR b的图示；
- 想一想，a AND b的怎么画？

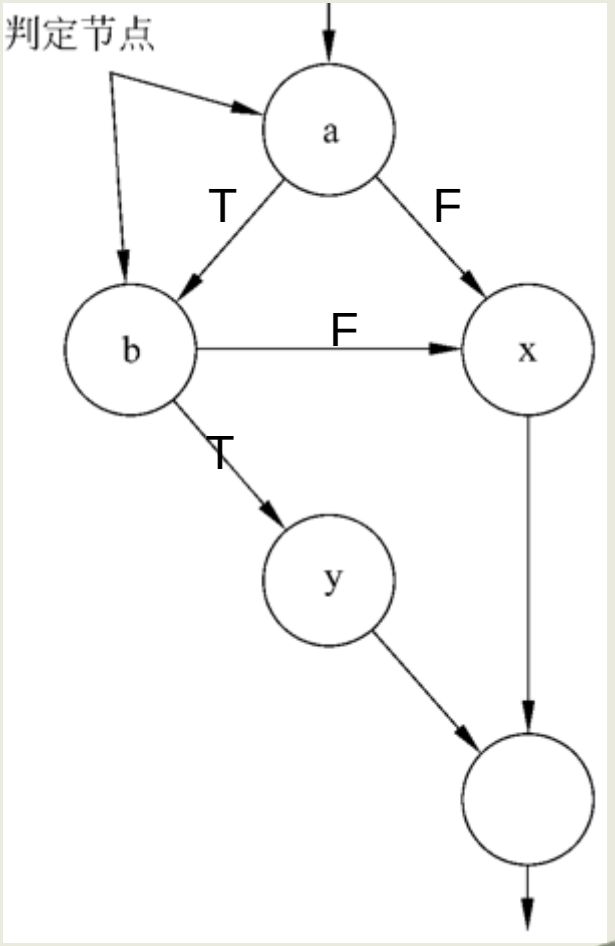
IF a AND b

then procedure y

else procedure x

EDNIF





6.5 程序复杂程度的定量度量

② 计算环形复杂度的方法

- (1) 流图中线性无关的区域数等于环形复杂度。
- (2) 流图G的环形复杂度V(G)=E-N+2,其中，E是流图中边的条数，N是结点数。
- (3) 流图G的环形复杂度V(G)=P+1，其中，P是流图中判定结点的数目。

- 例：计算下列程序图的程序复杂度

PDL

```
procedure:sort
1:   do while records remain
2:       read record;
         if record field 1=0
3:         then process record;
             store in buffer;
             increment counter;
4:       elseif record field 2=0
5:         then reset counter;
6:         else process record;
             store in file;
7a:      endif
         endif
7b:  enddo
8:   end
```

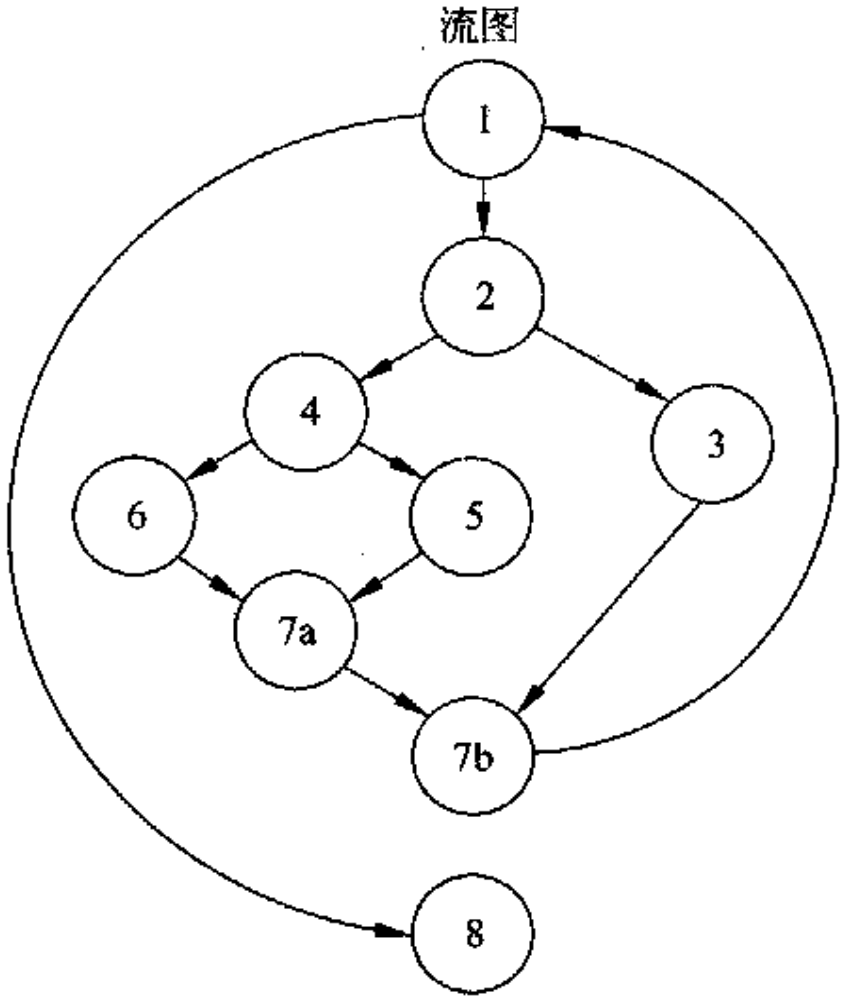
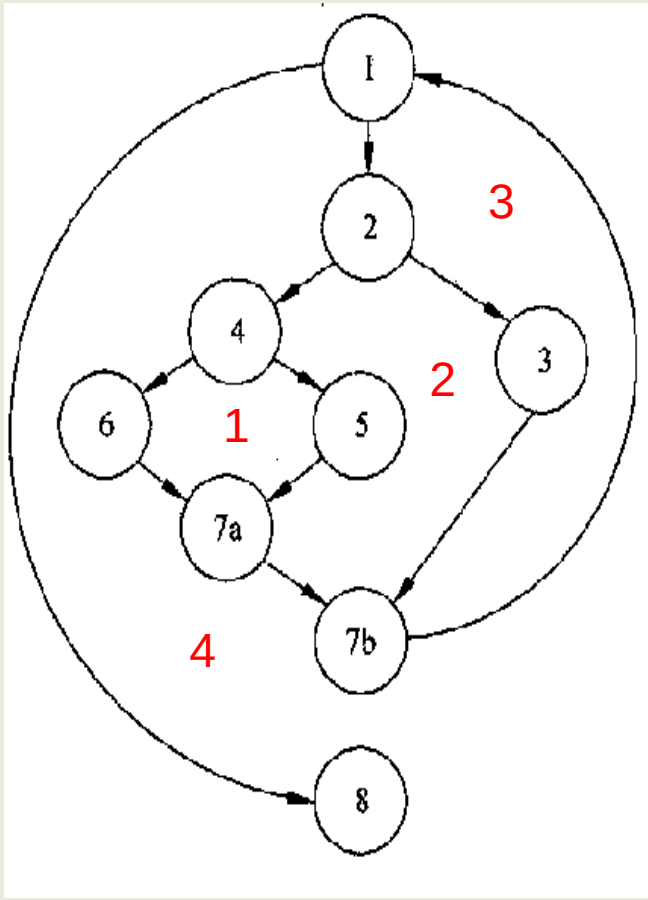


图 6.16　由 PDL 翻译成的流图

•解：

- 方法一：程序图把平面分为4个区域，程序复杂度V（G）＝4；
- 方法二：边的条数E＝11，结点数N＝9，程序复杂度V（G）＝E－N＋2＝4；
- 方法三：判定结点为1、2、4点，数目为P＝3个，所以V（G）＝P＋1＝4。



③ 环形复杂度的用途

对测试难度的一种定量度量，也能对软件最终的可靠性给出某种预测。

实践表明，模块规模以V(G)≤10为宜

课堂练习：以下程序代码的环形复杂度为多少？

```
set
{
    Product previousValue = this._Product.Entity;
    if (((previousValue != value)
                || (this._Product.HasLoadedOrAssignedValue == false)))
    {
        this.SendPropertyChanging();
        if ((previousValue != null))
        {
            this._Product.Entity = null;
            previousValue.Products_Relateds.Remove(this);
        }
        this._Product.Entity = value;
        if ((value != null))
        {
            value.Products_Relateds.Add(this);
            this._SKU = value.SKU;
        }
        else
        {
            this._SKU = default(string);
        }
        this.SendPropertyChanged("Product");
    }
```

6.5 程序复杂程度的定量度量

2. Halstead方法

- 根据程序中运算符和操作数的总数来度量程序的复杂程度。
- 令N1为程序中运算符出现的总次数，N2为操作数出现的总次数，程序长度N定义为：
  - $N=N1+N2$
- 程序中使用的不同运算符(包括关键字)的个数n_1，以及不同操作数(变量和常数)的个数n_2。Halstead给出预测程序长度的公式如下：
  - $H=n_1 \log_2 n_1+n_2 \log_2 n_2$
- 多次验证都表明，预测的长度H与实际长度N非常接近。
- Halstead还给出了预测程序中包含错误的个数的公式如下：$E=N \log_2 (n_1+n_2)/3000$

本章小结

1.结构程序设计技术是进行详细设计的逻辑基础。

2.人机界面设计必须重视。

3.过程设计是详细设计阶段完成的主要工作。

4.在开发有清楚的层次结构时可采用面向数据结构的设计方法完成设计过程设计。

5.使用环形复杂度可以定量度量程序的复杂程度。

《概要设计说明书》－1

1 前言

1.1 目的

1.2 范围

1.3 定义、缩写词、略语

1.4 参考资料

2 任务概述(项目概述)

2.1 目标

2.2 运行环境

2.3 需求概述

2.4 条件与限制

《概要设计说明书》－2

3 总体设计

3.1 处理流程

3.2 总体结构和模块外部设计

3.3 功能分配

3.4 参考资料

4 接口设计

4.1 外部接口

4.2 内部接口

《概要设计说明书》－3

5 数据结构设计

5.1 逻辑结构设计

5.2 物理结构设计

5.3 数据结构与程序设计

6 运行设计

6.1 运行模块的组合

6.2 运行控制

6.3 运行时间

《概要设计说明书》－4

7　出错处理

7.1　出错输出信息

7.2　出错处理对策

8　安全保密设计

9　维护设计

《详细设计说明书》－1

1 前言

1.1 目的

1.2 范围

1.3 定义、缩写词、略语

1.4 参考资料

2 总体设计

2.1 需求概述

2.2 软件结构

敏捷——用户故事的例子

有什么启示？能否用结构化方式进行这些系统的分析和设计？

Epic主题	Story	Task
拍卖系统	用户可以在拍卖平台上选择商品，然后投标购买 ……	1. 拍卖功能数据库结构的设计 2. 拍卖流程的规则设计 3. 拍卖页面的设计……
	用户可以在拍卖平台上查看并删除拍卖纪录，可以删除过期的拍卖……	1. 查询页面的设计 2. 删除功能的设计……
华为云上的Devops项目凤凰商城案例——汽车零部件配件电子商城	管理员可以使用系统进行会员的积分管理	1. 积分规则设计 2. 积分功能数据库表结构设计 ……
	管理员可以使用系统进行会员级别的设置	1. 会员级别设置网页的设计 2. 会员管理功能数据库表结构设计……
	管理员可以使用系统进行用户分析	1. 用户数据库结构分析 2. 用户分析页面的设计……
	用户可以管理所有的门店网络 ……	1. 查看门店网络的页面设计 ……

《详细设计说明书》－2

3　程序说明

3.1 功能

3.2 性能

3.3 输入、输出项目

3.4 算法

3.5 程序逻辑

3.6 接口

3.7 存储分配

3.8 限制条件

3.9 测试要点

• 课堂练习