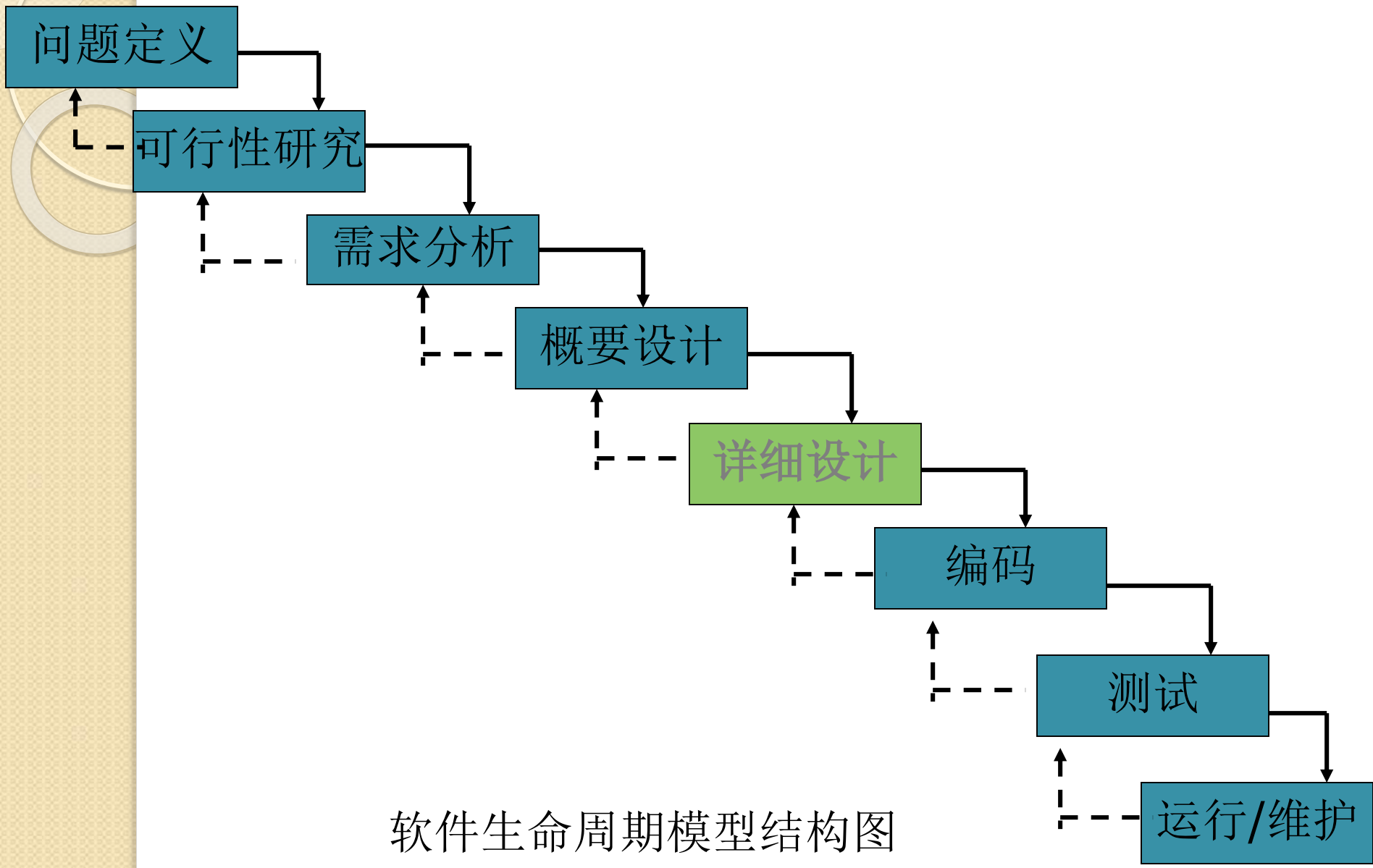
第11讲　详细设计

网络空间安全学院

芦效峰



软件生命周期模型结构图

1. 详细设计的概念

- 详细设计的任务是将概要设计阶段提出的解决问题的办法具体化
- 详细设计给出软件结构中各个模块内部过程的描述，从而在编码阶段可以把这个描述直接翻译成用某种程序设计语言书写的程序。
- 设计分为：
- 架构设计 ——规划房子的地基
- 概要设计 ——规划房子的钢筋混凝土框架
- 详细设计 ——规划管线和施工设计

详细设计的目的

Program Design ≠ Coding，详细设计不是真正的编写程序，而是设计出程序的详细规格说明（数据结构＋算法）。

从软件开发的工程化的观点来看，在进行程序编码以前，需要对系统所采用算法的逻辑关系进行分析，并给出明确、清晰的表述，为后面的程序编码打下基础，这就是详细设计的目的。

详细设计是否有必要？

- 1）当详细设计和编码人员**实际为同一人**时

观点：既然是同一人，自己做事心里有数，整体想清楚了就开始编码，而不愿意写详细设计文档。

- 2）当详细设计和编码人员**不是同一人**时：

观点：详细设计要画的很细才有效果，如果详细设计做的不够细，那还不如直接看需求文档。

文档有两个功能，**一个是指导下游工序进行工作，二是作为档案供将来回忆或新人学习之用**

2. 详细设计的任务

- 详细设计过程

1. 确定每个模块的算法
   - 选择适当工具表达算法执行过程
2. 确定每一个模块的数据结构
3. 为每一个模块设计一组测试用例
   - 输入数据、预期输出结果
4. 编写《详细设计说明书》
5. 设计评审

详细设计说明书的内容

- 程序描述：程序简要描述，意义和特点
- 功能：程序应具备的功能
- 性能：精度、灵活性和时间特性等
- 输入项
- 输出项
- 算法：具体的计算步骤和过程
- 接口：模块的隶属关系、调用方式和参数、与模块直接关联的数据结构（如数据库表、文件等）
- 存储分配（根据需要）
- 注释设计：程序注释说明
- 限制条件：程序运行中所受的限制条件

详细设计的内容

- 详细设计优化
  - 在不考虑时间因素的前提下开发并精化软件结构
  - 选出最耗时的模块，仔细设计处理算法
  - 孤立出大量占有处理机资源的模块
  - 必要时重新设计或用依赖于机器的语言重写大量占有资源的模块

3. 结构程序设计 structured programming

- 3.1 结构程序设计的提出
  - 结构程序设计概念最早1965年由E.W.Dijkstra提出
    - “可以从高级语言中取消GOTO语句”
    - “程序的质量与程序中所包含的GOTO语句的数量成反比”
  - 1966年bohm和Jacopini证明了：
    - 只用三种基本的控制结构能实现任何单入口单出口的程序
  - 1968年人们认识要创立一种新的程序设计思想
    - 以显著地提高软件生产率和降低软件维护代价

结构程序设计

- 1971年IBM公司成功地使用了结构程序设计技术
  - 纽约时报信息库管理系统
  - 美国宇航局空间实验室飞行模拟系统
- 1972年IBM公司的Mills进一步补充了结构程序设计的规则

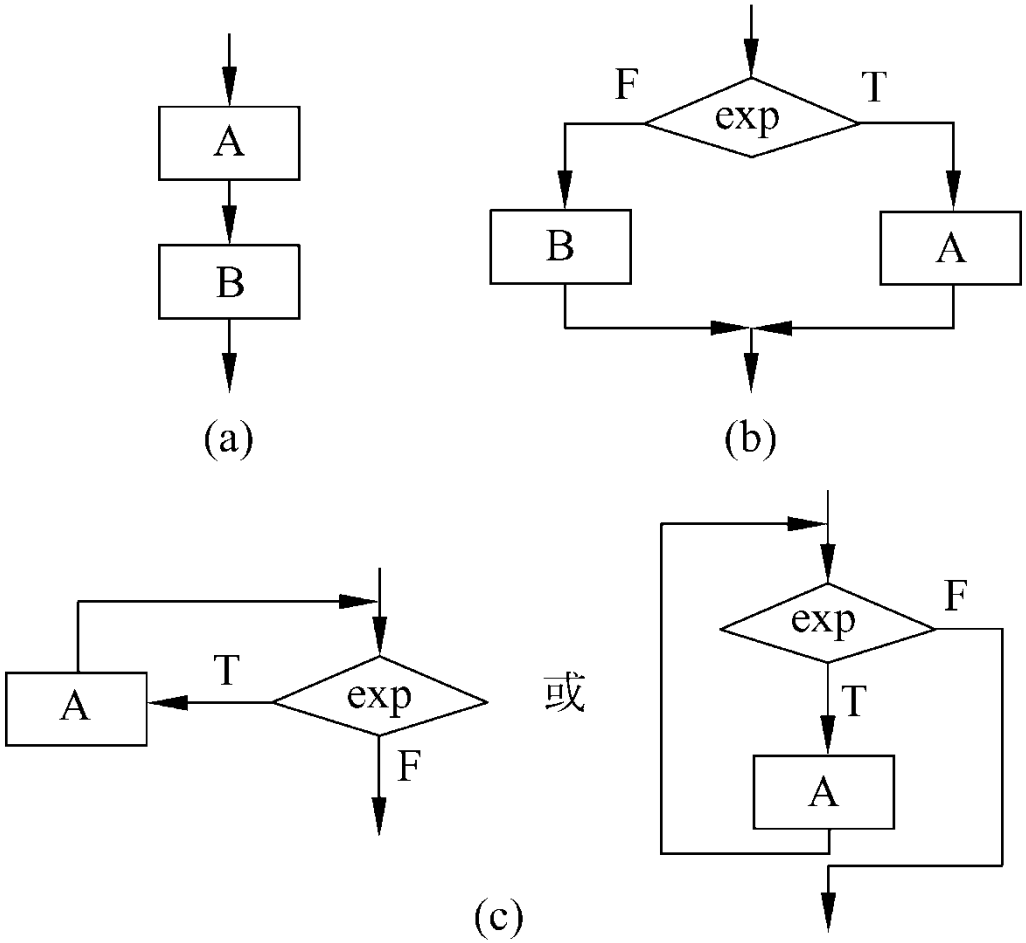
结构程序设计是尽可能少用GOTO语句的程序设计方法。最好仅在检测出错误时才使用GOTO语句，而且应该总是使用前向GOTO语句

结构程序设计分类

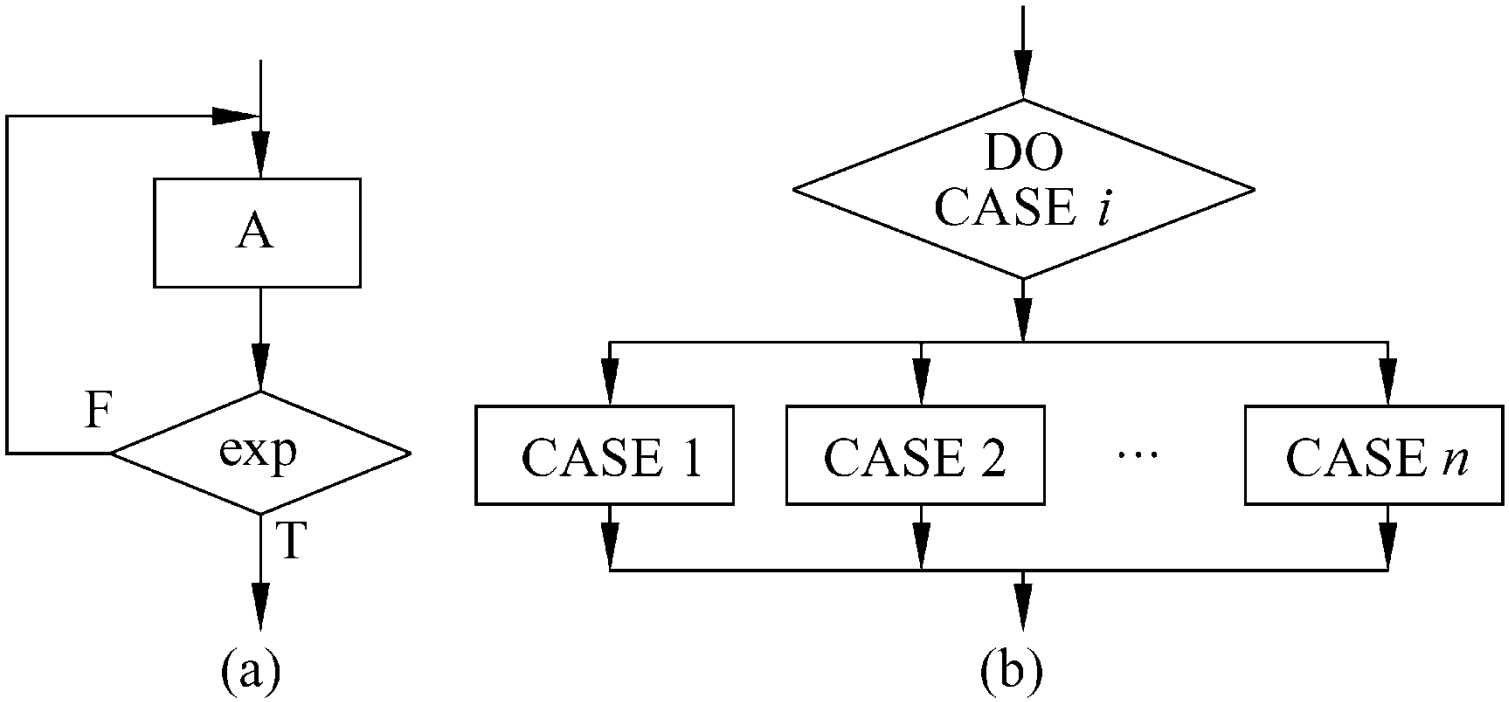
- 3.2 结构程序设计分类
  - 只允许使用顺序、IF-THEN-ELSE分支和DO-WHILE型循环这三种基本控制结构，称为经典的结构程序设计
  - 还允许使用DO-CASE多分支结构和DO-UNTIL循环结构，称为扩展的结构程序设计
  - 如果再加上允许使用LEAVE(或BREAK)结构，称为修正的结构程序设计

- 三种基本的控制结构：
  - ✓ 顺序
  - ✓ 选择
  - ✓ 循环
- 其他常用的控制结构：
  - ✓ DO-UNTIL
  - ✓ DO-CASE
- 以及：
  - ✓ LEAVE
  - ✓ BREAK

3种基本的控制结构



其他常用的控制结构



3.3 结构程序设计应遵循的基本原则

（一）采用自顶向下、逐步求精的模块化方法设计程序

- 自底向上的程序设计方法，编写出的程序往往局部结构较好，而整体结构不佳，有时会导致程序设计不理想或失败。
- 自顶向下逐步求精的方法符合人们解决复杂问题的普遍规律，整体结构清晰、合理。
- 软件可读性好，提高了软件的可维护性

（二）尽量使用 “基本结构”编程

（三）限制转向语句的使用

大量的资料数据表明：软件产品的质量与软件中goto语句的数量成反比；

3.4 结构程序设计的好处

- 自顶向下逐步求精的软件开发方法比较符合人们解决复杂问题的规律，因此可以提高软件开发的成功率和生产率。

- 由全局到局部，由整体到细节的逐步求精过程开发出的软件具有良好的层次结构，更易于人们的阅读和理解。

- 限制或不使用goto语句，使得程序的静态结构和动态执行情况较为接近，对于错误的诊断和纠正比较容易。

4. 过程设计工具

在详细设计阶段，要决定各个模块的实现算法，并精确地表达这些算法。

表达过程规格说明的工具叫做详细设计工具，它可以分为以下三类：

- 图形工具
- 表格工具
- 语言工具

(1) 图形工具

- 程序流程图(FC)
- 盒图(NS)
- 问题分析图(PAD)

(2) 表格工具

- 判定表
- 判定树

(3) 语言工具

- 过程设计语言(PDL)

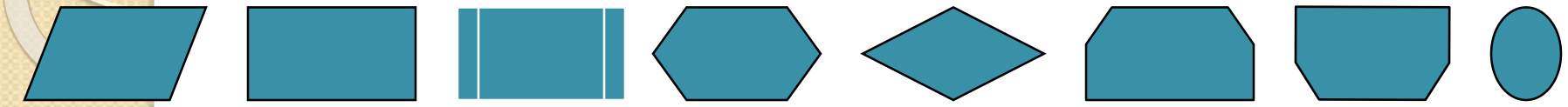
1. 程序流程图

1. 程序流程图

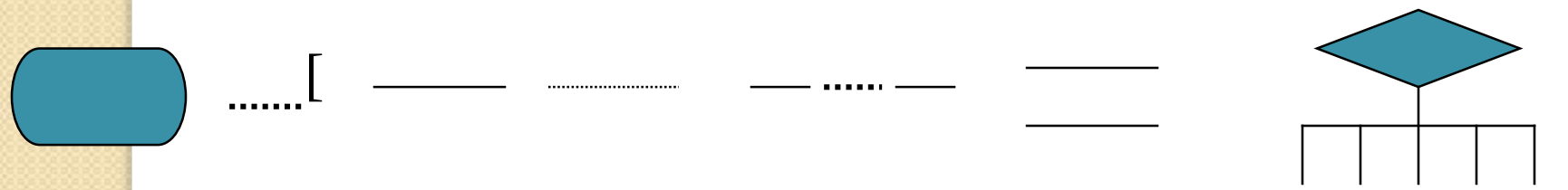
- 在40年代末到70年代中期，程序流程图一直是软件设计的工具
- 它以对控制流程的描绘直观、易于掌握而被设计人员青睐

依据国家标准(GB1526-89)，列出有关程序流程图的基本符号

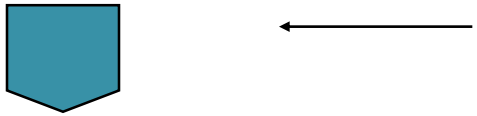
程序流程图符号



数据　处理　待处理　准备　判断　循环上、下界　连接



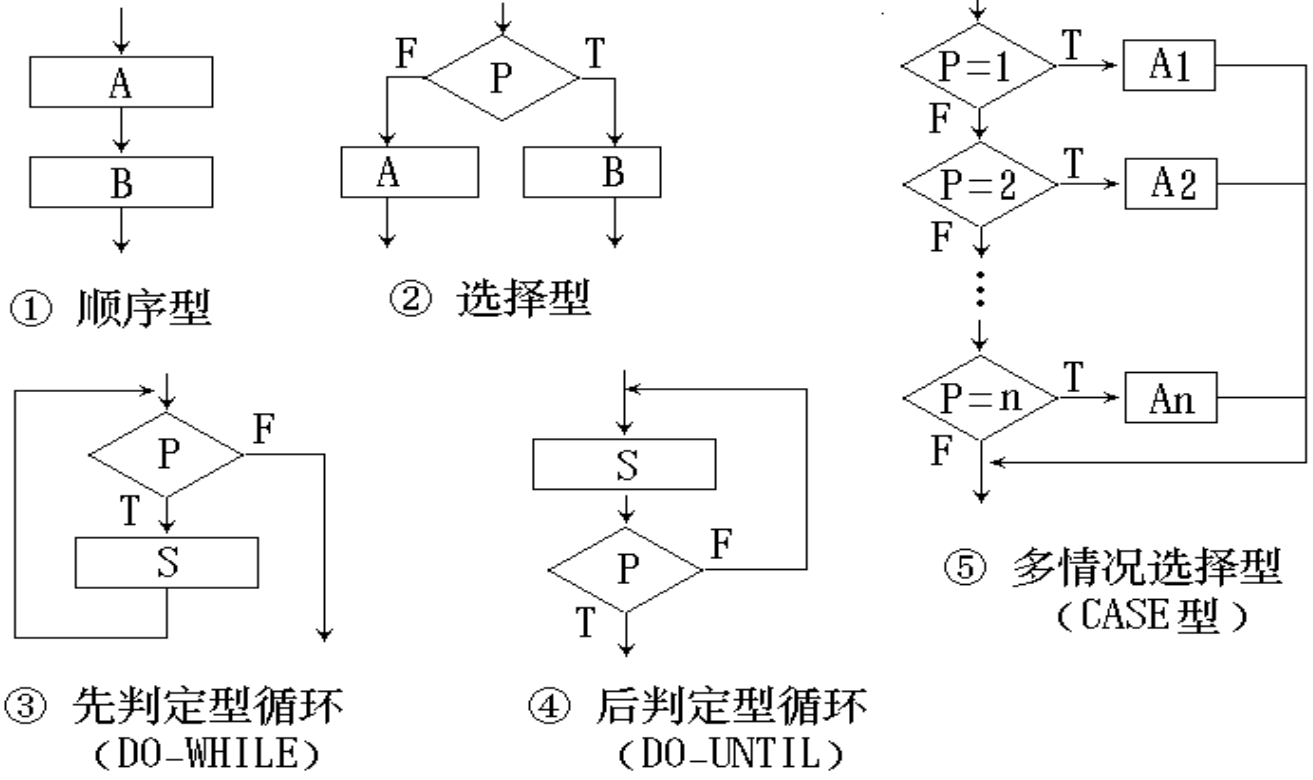
端点符　注解符　流线　虚线　省略符　并行方式　多出口判断



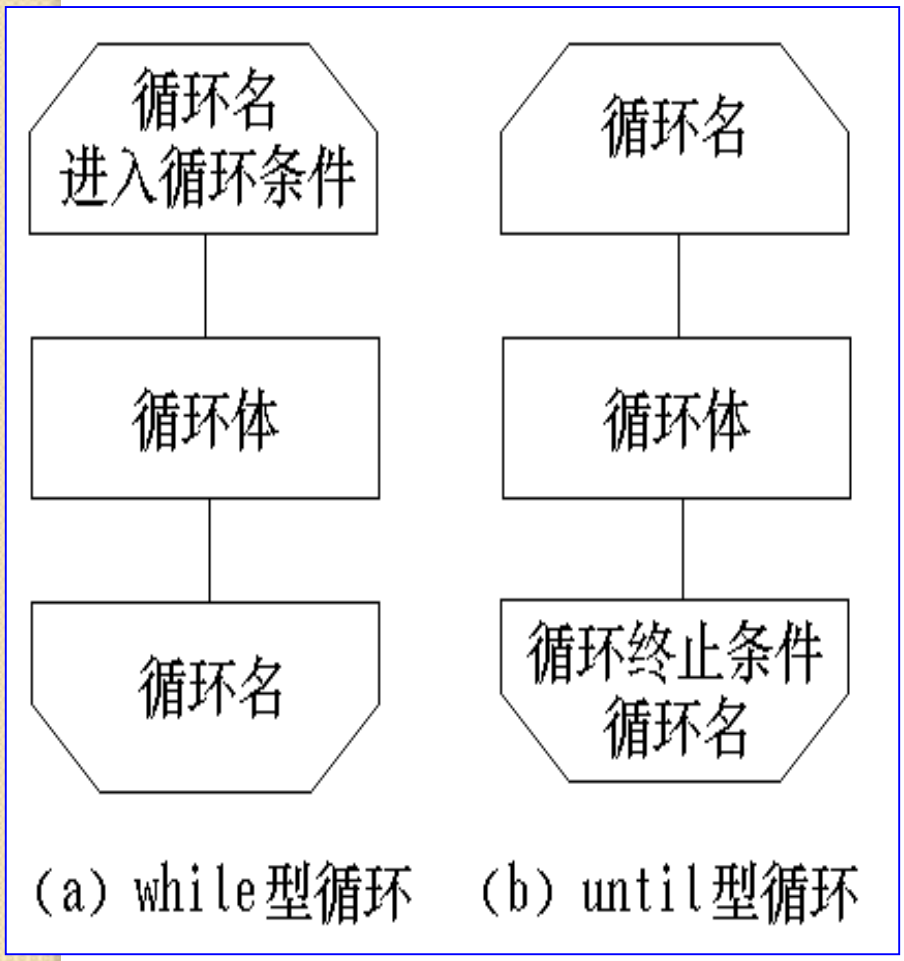
换页连接　控制流

程序流程图基本结构

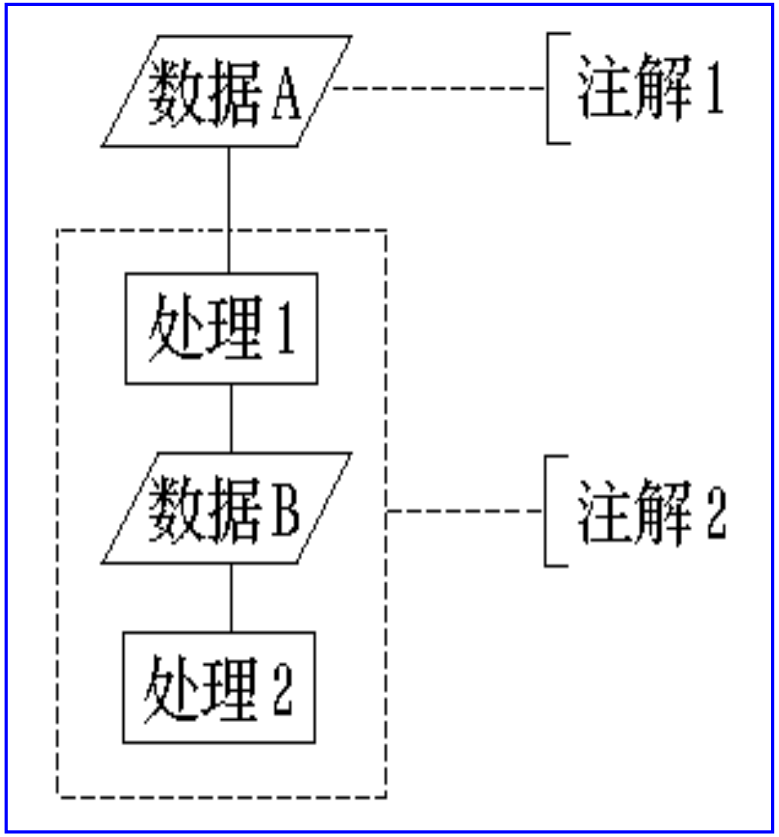
- **程序流程图也称为程序框图，程序流程图使用五种基本控制结构是：**



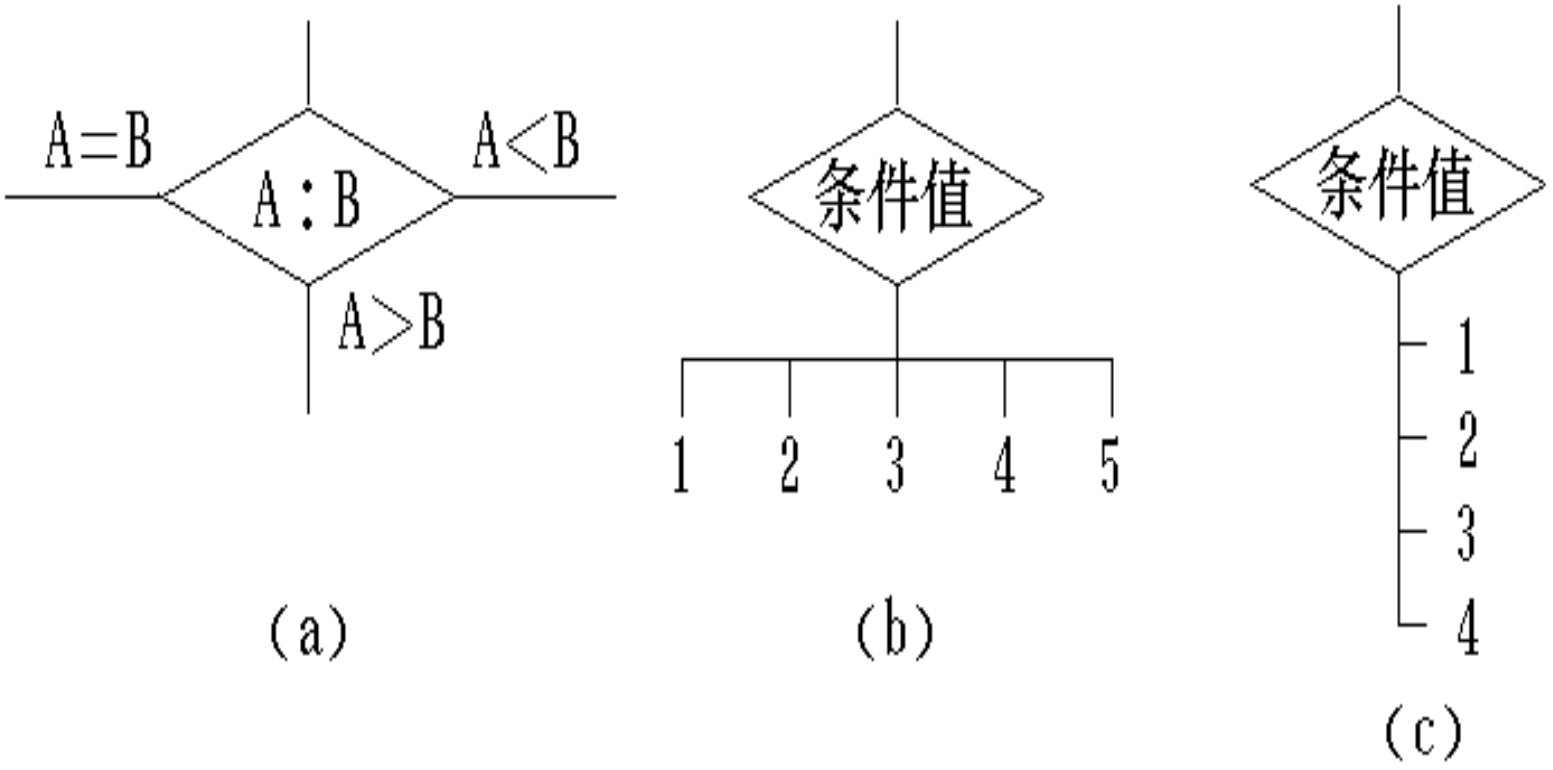
循环的标准符号



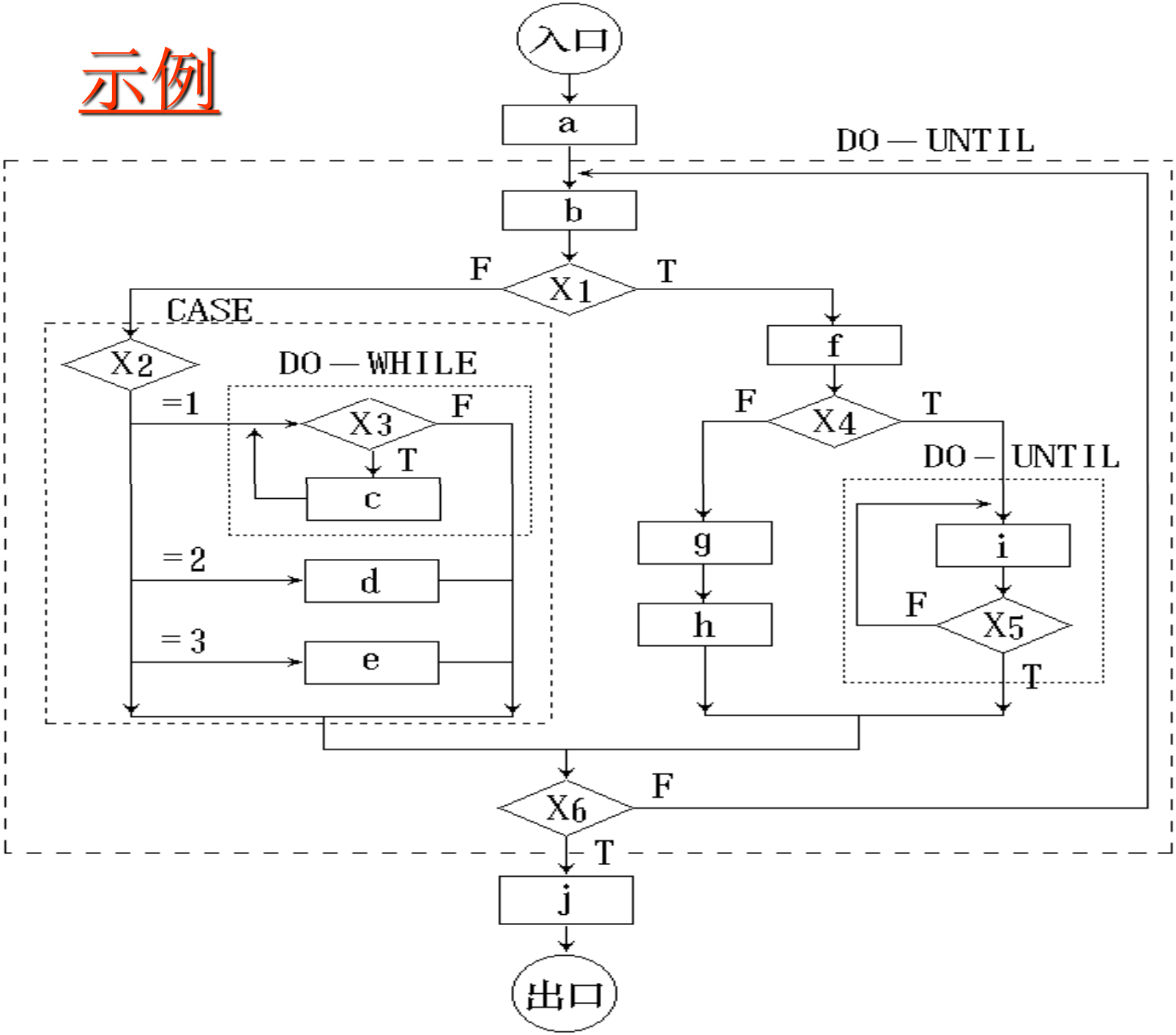
注解的使用



多出口判断



示例


入口
a
DO — UNTIL
b
F
X1
T
CASE
X2
DO — WHILE
=1
X3
F
T
c
=2
d
=3
e
f
F
X4
T
DO — UNTIL
g
h
i
F
X5
T
F
X6
T
j
出口

程序流程图的优缺点

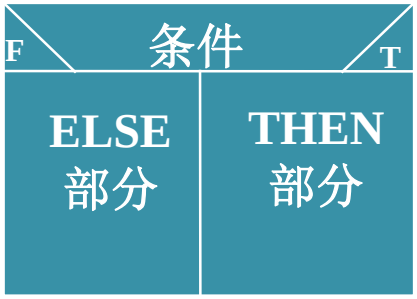
- 优点：容易掌握，且历史“悠久”，使用广泛。
- 缺点：本质上不具备逐步求精的特点，对于提高大型系统的可理解性作用甚微；

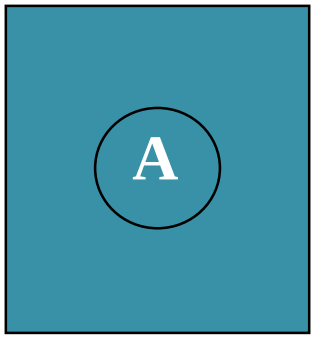
  不易表示数据结构；

  可以随心所欲地画控制流程，容易造成非结构化的程序结构。
- 趋势：停止使用

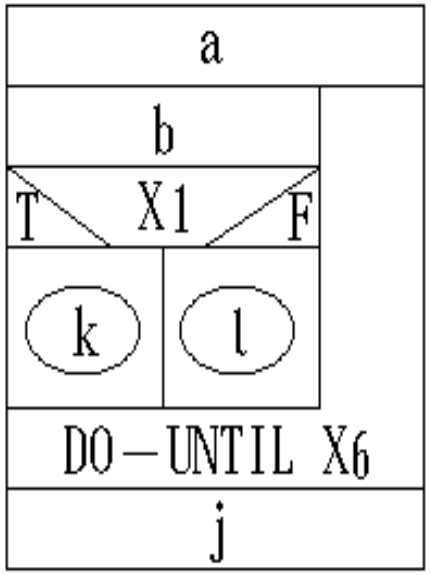
2 过程设计工具： 盒图

- N-S图是Nassi和Shneiderman提出来的，它体现了结构程序设计精神，是目前过程设计中广泛使用的一种图形工具。

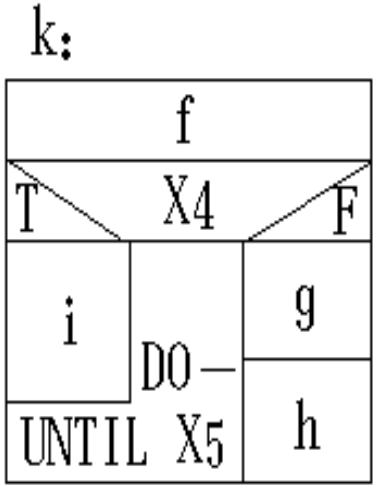




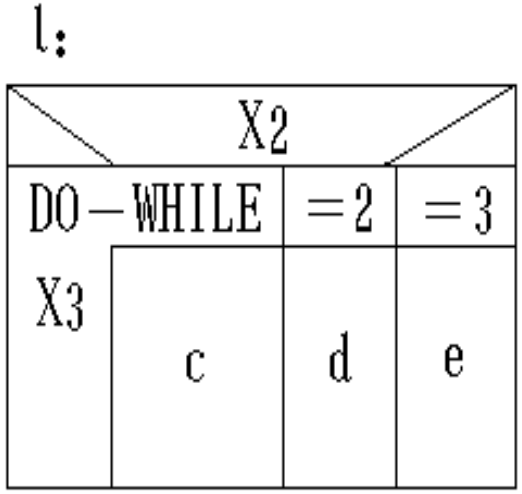
N-S图的嵌套定义形式



(a)



(b)



(c)

■顺序结构

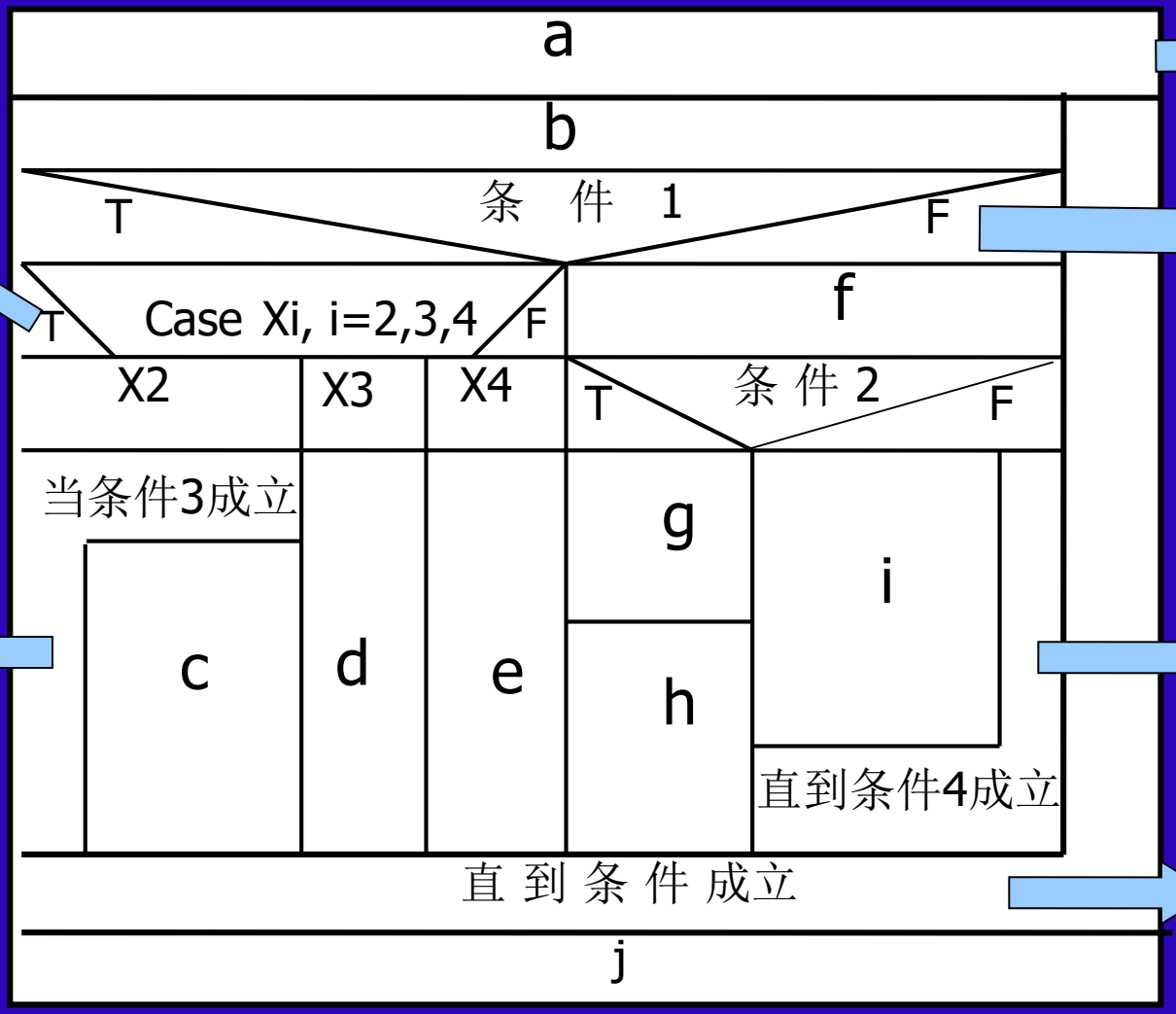
■case多分支选择结构

■if选择结构

■Do-while循环

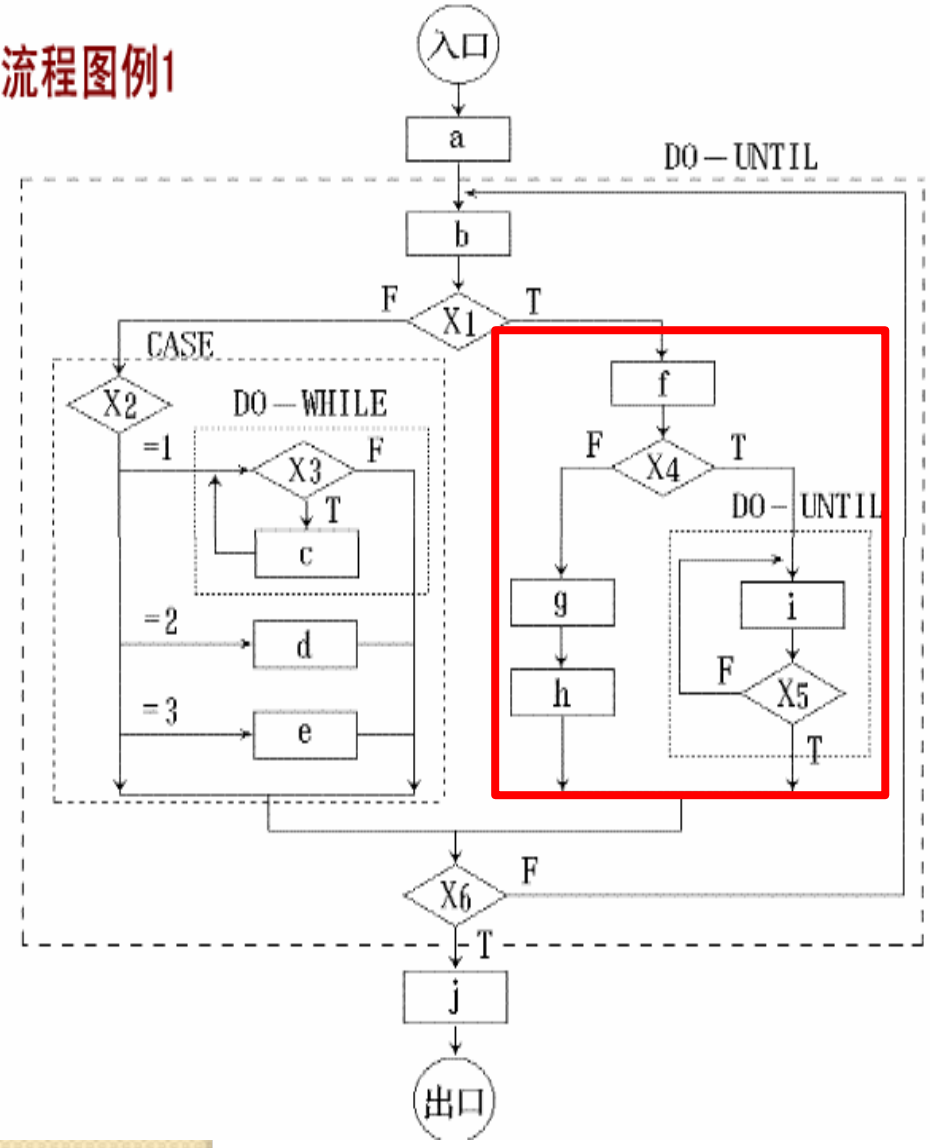
■Do-until循环

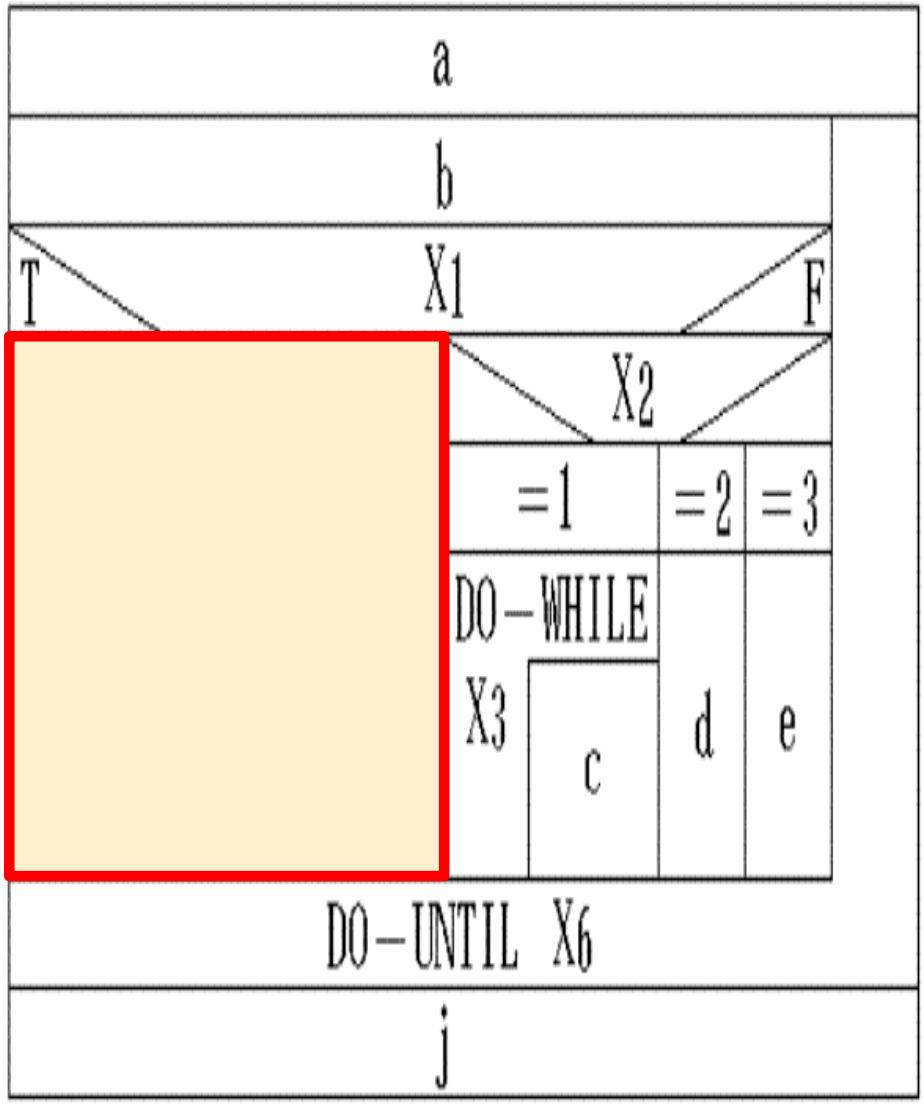
■Do-until循环



练习：请画出图中红色框所对应的N-S图

流程图例1

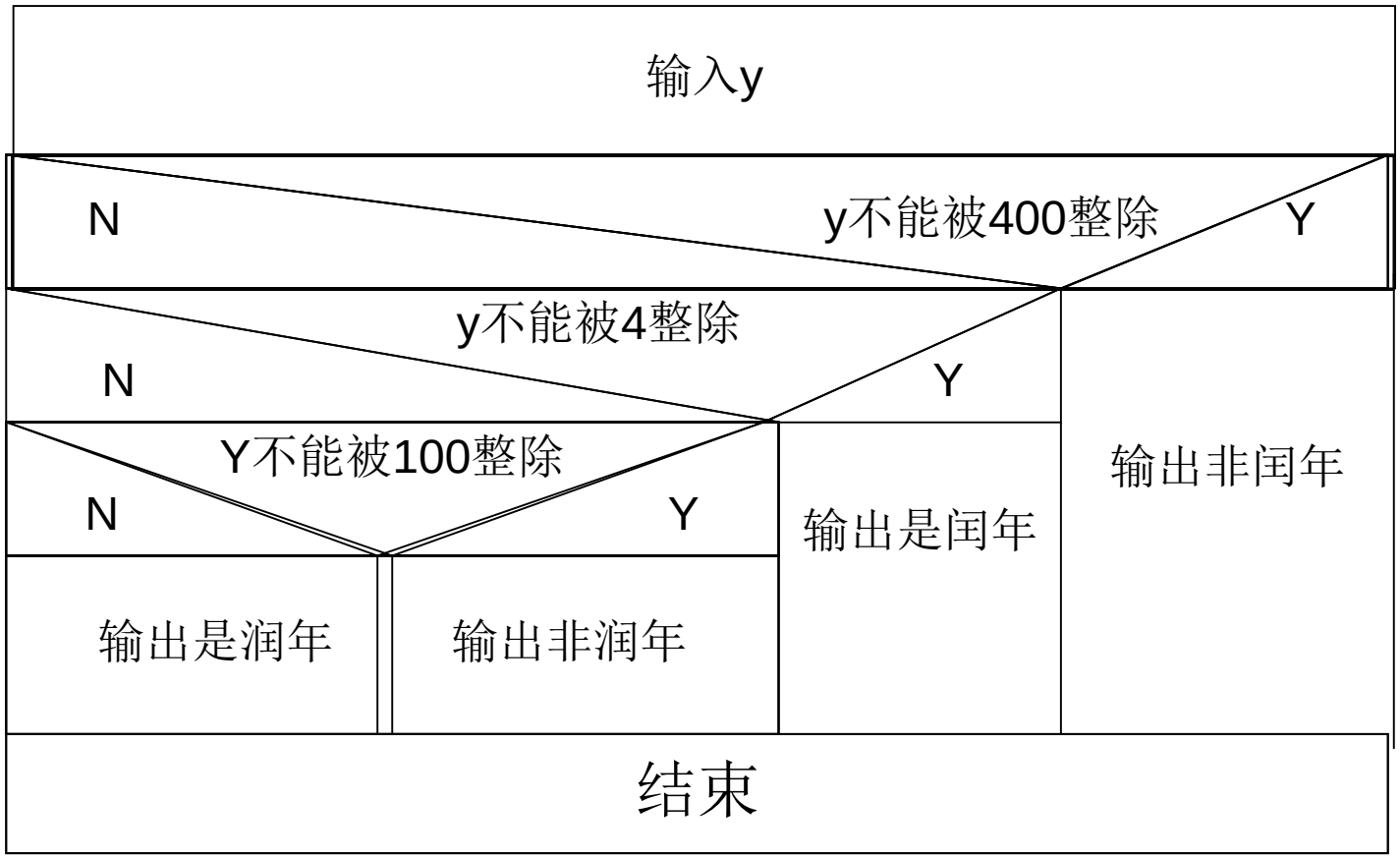




N-S图的特点

- 优点
  - 功能域(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。
  - 没有箭头不允许随意转移控制，不可能任意转移控制。
  - 很容易确定局部和全程数据的作用域。
  - 很容易表现嵌套关系，也可以表示模块的层次结构。
- 缺点
  - 随着程序内嵌套的层数增多时，内层方框越来越小，会增加画图的难度，影响清晰度。

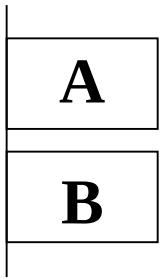
例2、判定y是否为润年(能被4整除却不能被100整除或能被400整除的年份是闰年)



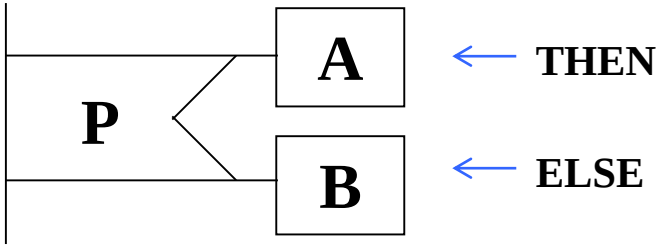
3 过程设计工具： PAD图

- PAD（Problem Analysis Diagram）问题分析图
- 日立公司中央研究所在1973年研究开发
- 使用二维树形结构图描述程序的逻辑
- PAD图是一种由左往右展开的二维树型结构。
- PAD图的控制流程为自上而下、从左到右地执行

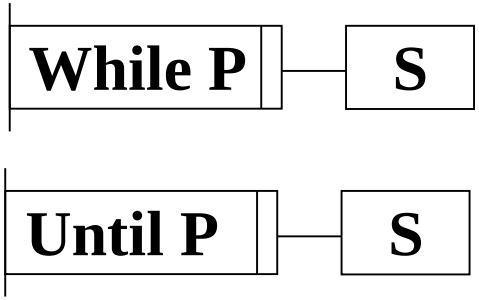
过程设计工具



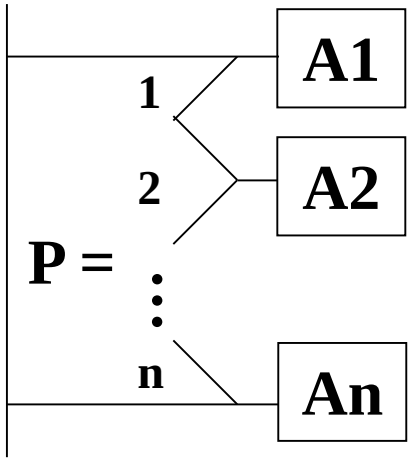
Sequential



Selective



Loops



Case

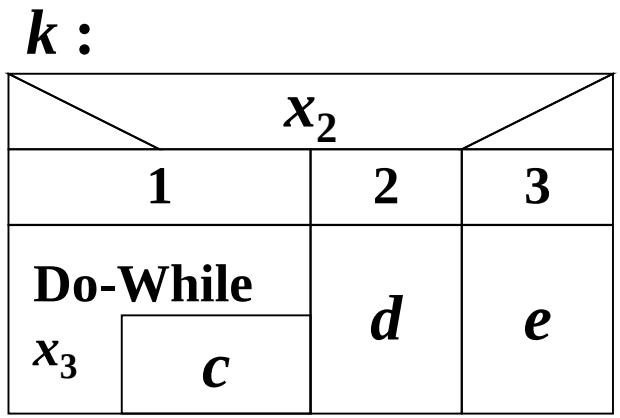
Statement Index

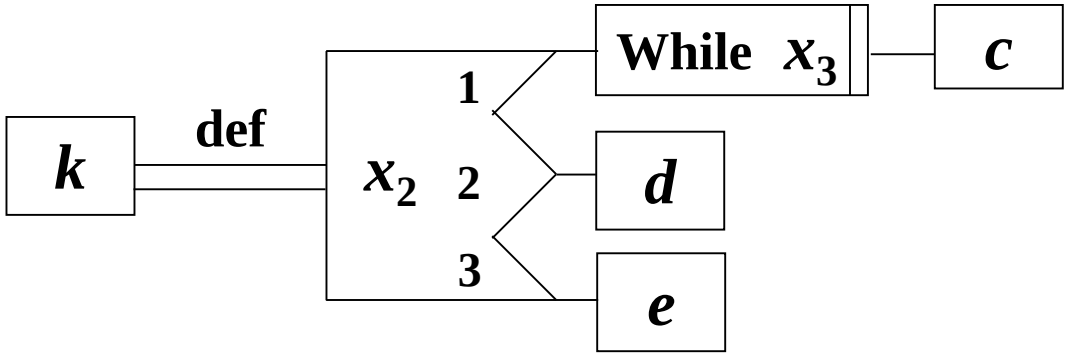
def

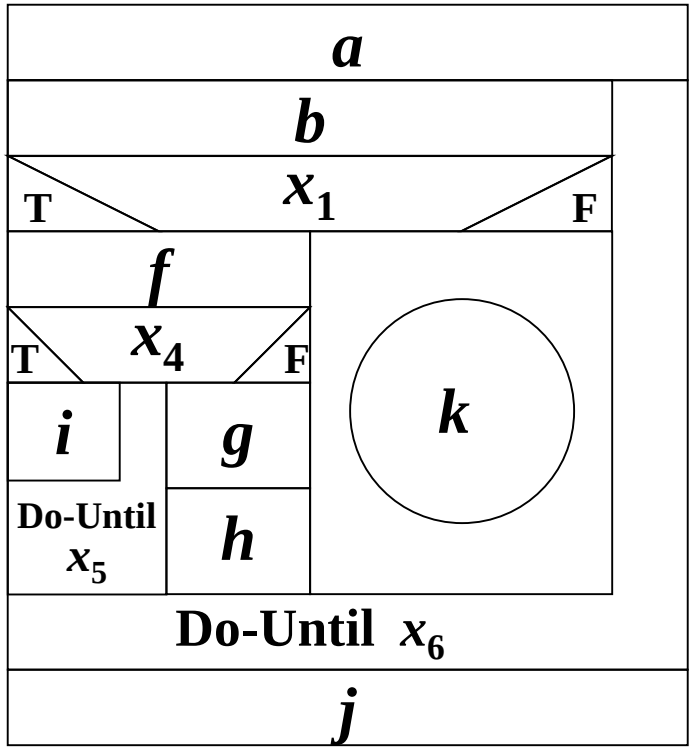
Definition

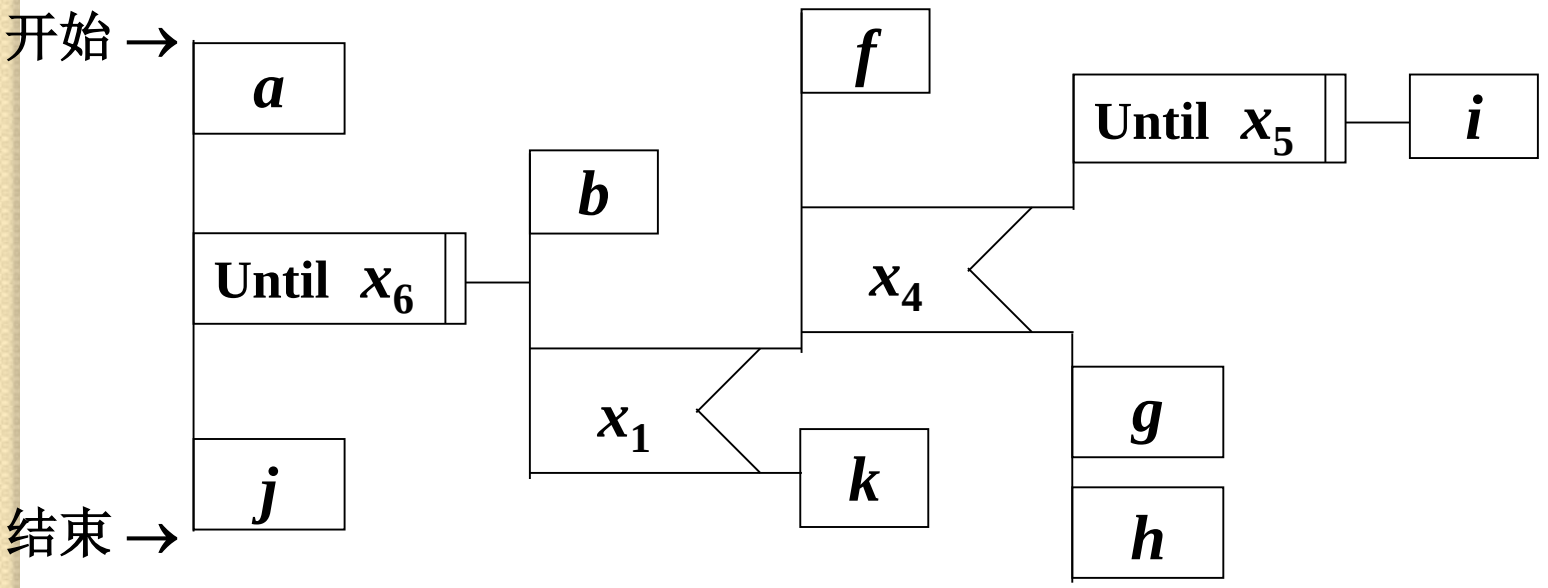
过程设计工具

例：N-S图与PAD的转换






a
b
T
x_1
F
f
T
x_4
F
i
g
k
Do-Until
x_5
h
Do-Until x_6
j


开始 →
a
b
Until x_6
j
结束 →
f
Until x_5
i
x_4
x_1
k
g
h

PAD图的主要优点如下：

(1)程序结构清晰，结构化程度高。图中的竖线为程序的层次线，最左边竖线是程序的主线， 其后一层一层展开，层次关系一目了然。

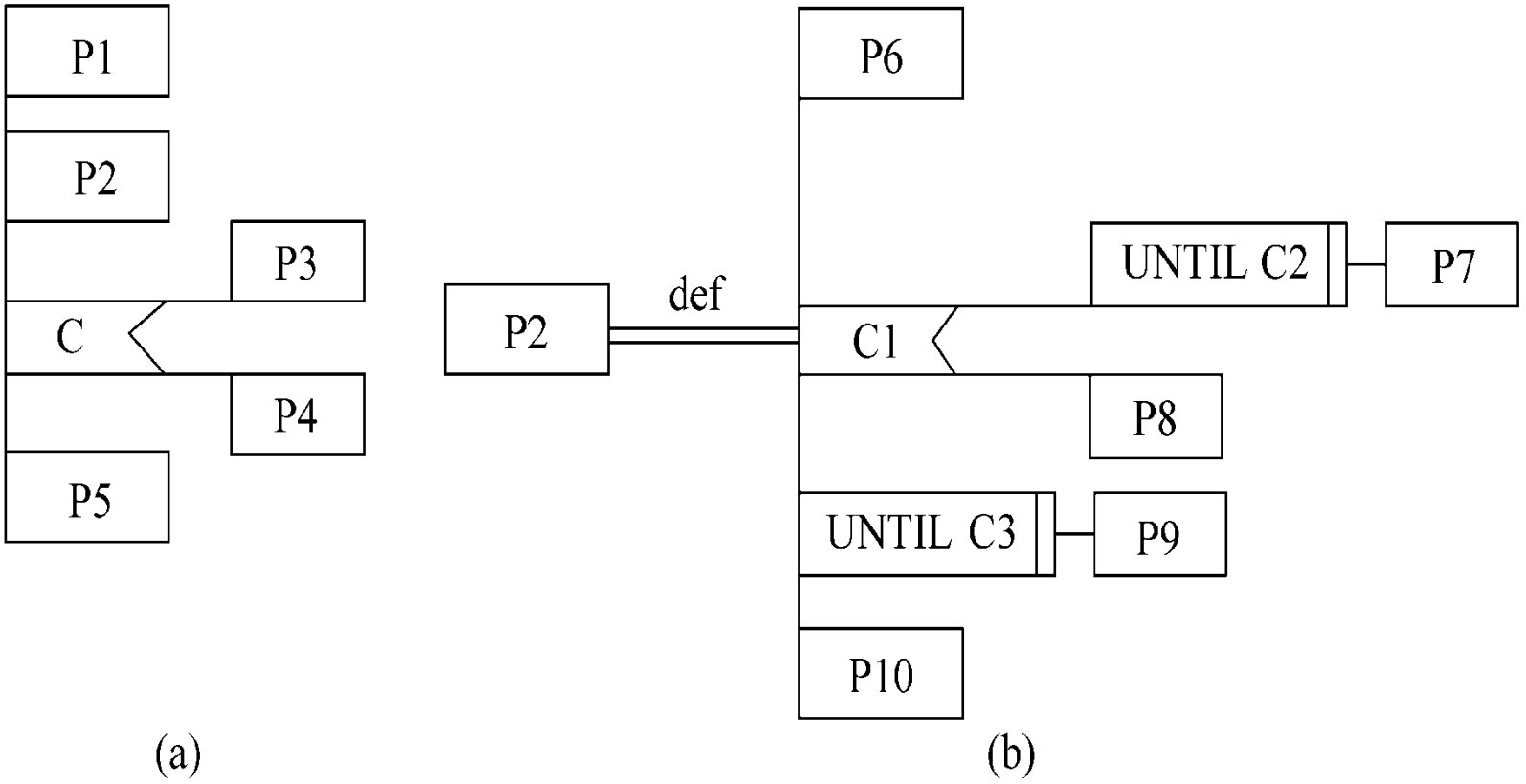
(2)支持自顶向下，逐步求精的设计方法。

(3)既可以表示程序逻辑，也可以描绘数据结构。

(4)用PAD图表现程序逻辑，易读易写，使用方便。

(5)容易转换成高级语言源程序，也可用软件工具实现自动转换。

(6)易读易写， 使用方便。 ■

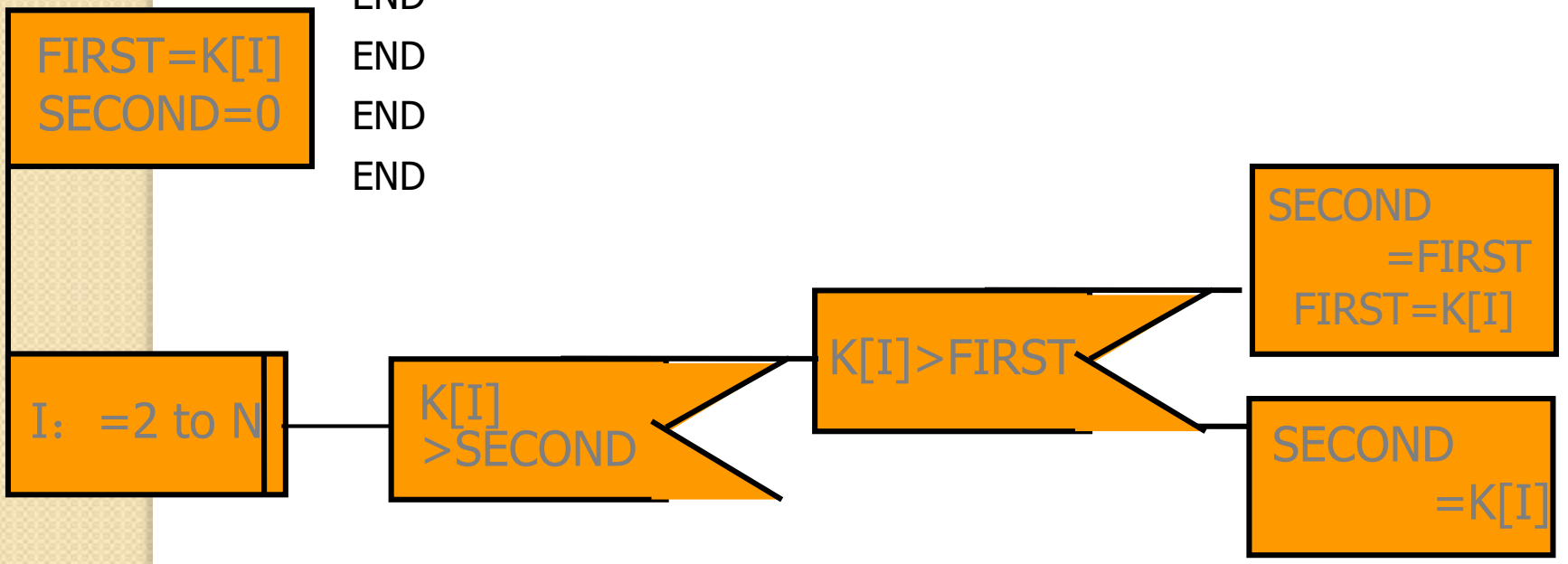


使用PAD图提供的定义功能来逐步求精的例子

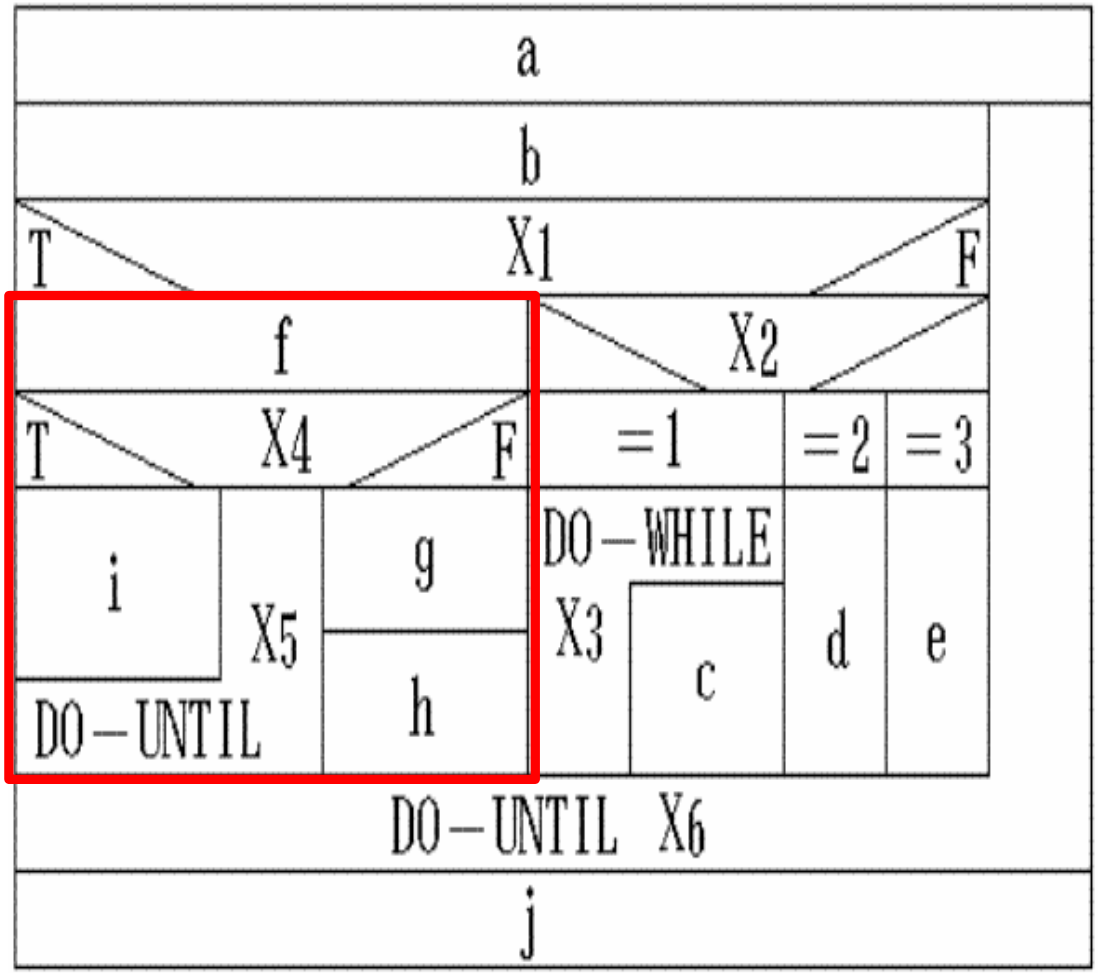
PAD图转换成代码举例

```
BEGIN
    FIRST:=K[1];
    SECOND:=0;
BEGIN FOR I:=2  TO  N  DO
BEGIN  IF  K[I]>SECOND THEN
BEGIN    IF  K[I]>FIRST
THEN   BEGIN SECOND:=FIRST;
                    FIRST:=K[I];
 ELSE    SECOND:=K[I];
 END
 END
 END
 END
```

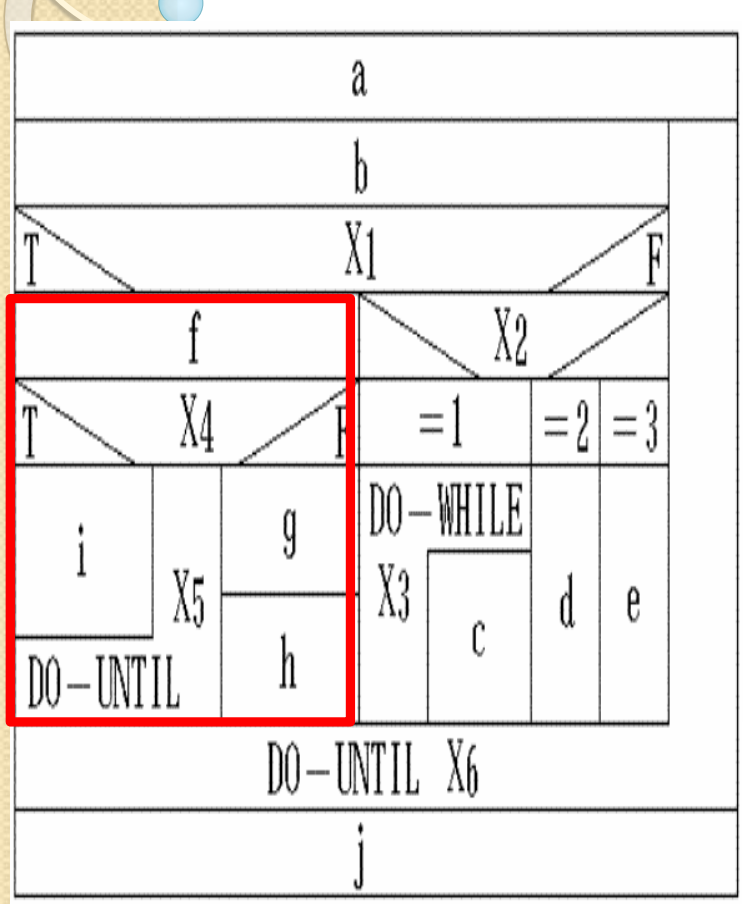
- PAD图

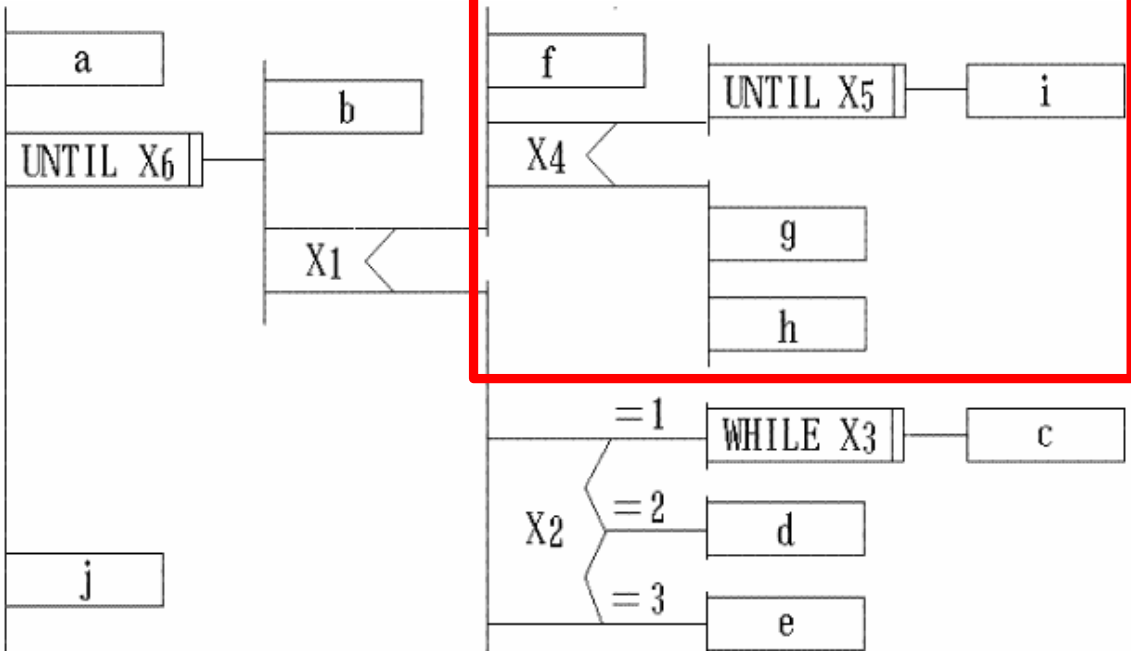


练习： 下图中红框内转化为PAD图



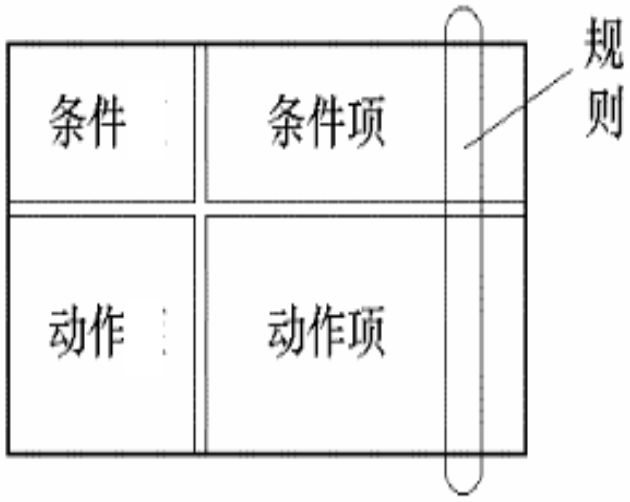
练习： 下图中转化为PAD图





4 过程设计工具：判定表

- 判定表能够清晰地表示复杂的条件组合与所产生的动作之前的关系。
- 一张判定表由四部分组成：
  - 左上部列出所有的条件 I
  - 左下部是所有可能的操作III
  - 右上部是各种条件的组合矩阵 II
  - 右下部是每种条件组合对应的动作



4 判定表举例

例:

- 某航空公司规定，乘客可以免费托运重量不超过30公斤的行李。当行李重量超过30公斤时，头等舱国内乘客超重部分每公斤收费4元，其他舱的国内乘客超重部分每公斤收费6元，对于外国乘客超重部分每公斤收费比国内乘客多一倍，对于残疾乘客超重部分每公斤收费比正常乘客减少一半。

国内乘客		T	T	T	T	F	F	F	F
头等仓		T	F	T	F	T	F	T	F
残疾乘客		F	F	T	T	F	F	T	T
行李重量≤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							√		

判定表的特点

- 1、当算法中包含多重嵌套的条件选择时，用程序流程图、N-S图或PAD都不易清楚地描述。然而，判定表却能清晰地表达复杂的条件组合与应做动作之间的对应关系。
- 2、判定表缺乏通用性，没法和PAD图那样良好的转换成代码。
- 3、无法清晰表示顺序和重复等处理

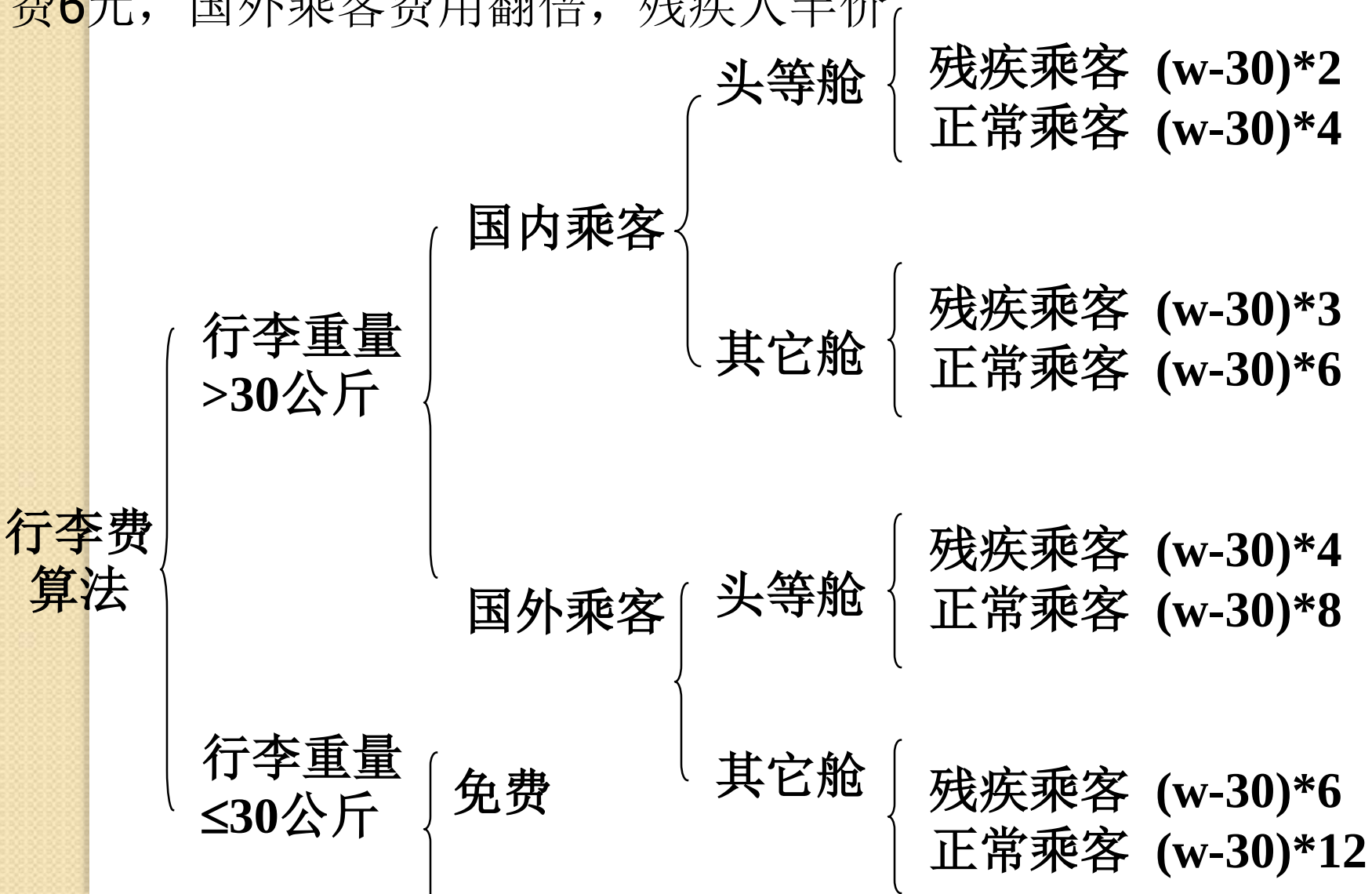
5. 过程设计工具： 判定树

1. 判定树的定义

- 判定树又称决策树，适合描述问题处理中具有多重判断，即每个决策与若干条件有关。
- 使用判定树进行描述时，应该从问题的文字描述中分清哪些是判定条件，哪些是判定的决策。

2. 判定树的符号

例:某航空公司规定乘客可以免费托运30KG的物品，超出部分收费，头等舱国内乘客1kg收费4元，其他舱国内乘客1kg收费6元，国外乘客费用翻倍，残疾人半价



判定树的特点

- 表示复杂的条件组合与应做的动作之间的对应关系
- 判定树形式简单，长期以来一直受到重视
- 判定树的简洁性不如判定表
  - 经常出现同一个值重复写多遍
  - 且叶端重复次数急剧增加
- 由于判定树的分枝次序对于最终画出的判定树的简洁程度有较大影响
  - **所以选择哪一个条件作为第一个分枝是至关重要的.**

6. 过程设计工具：伪码和过程设计语言(PDL)

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

- 是一种“混杂式语言”，采用了某种语言（如英语或自然语言）的词汇，另一种语言（某种结构化程序设计语言）的语法。

- PDL的特点：

(1) 正文用严格的程序语言的基本控制结构分割，称为“外语法”，如 If … … Then … … Else 、 While … … DO 、 Repeat … … Until 、 Case … … of

(2) 操作用自然语言表示，描述处理特点，称为“内语法”。

(3) 具有数据说明的手段，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) 具有模块定义和调用技术，提供各种接口描述模式。

PDL程序结构

1) 顺序结构 ■

采用自然语言描述顺序结构： ■

处理S1 ■

处理S2 ■

... ■

处理Sn ■

2) 选择结构 ■

(1) IF-结构： ■

IF 条件 IF 条件 ■

处理S_1 或 处理S ■

ELSE ENDIF ■

处理S_2 ■

ENDIF ■

(2) IF-ORIF-ELSE结构： ■

IF 条件1 ■

处理S_1 ■

ORIF 条件$_2$ ■

✔ ... ■

ELSE　　处理Sn ■

ENDIF ■

(3) CASE结构：　■

CASE OF ■

CASE(1) ■

处理S_1 ■

CASE(2) ■

处理S_2 ■

✔ **... ■**

ELSE 处理S_n ■

ENDCASE ■

3) 重复结构 ■

(1) FOR 结构： ■

FOR i=1 TO n ■

循环体 ■

END FOR ■

(2) WHILE 结构： ■

WHILE 条件 ■

循环体 ■

ENDWHILE ■

(3) UNTIL 结构： ■

REPEAT ■

循环体 ■

UNTIL 条件 ■

4) 出口结构 ■

(1) ESCAPE 结构(退出本层结构): ■

WHILE 条件 ■

处理**S1** ■

ESCAPE L IF 条件

处理S_2 ■

ENDWHILE ■

L： ... ■ ■

5) 扩充结构 ■

(1) 模块定义： ■

PROCEDURE 模块名(参数) ■

✔ **... ■**

RETURN ■

END ■

(2) 模块调用： ■

CALL 模块名(参数) ■

(3) 数据定义： ■

DECLARE 属性 变量名， ... ■

属性有： 字符、 整型、 实型、 双精度、 指针、 数组及结构等类型。

(4) 输入/输出： ■

GET(输入变量表) ■

PUT(输出变量表) ■

例：商店业务处理系统中“检查发货单”

```
IF   发货单金额超过$500   THEN
       IF 欠款超过了 60 天 THEN
            在偿还欠款前不予批准
       ELSE （欠款未超期）
            发批准书，发货单
       END IF
ELSE   （发货单金额未超过$500 ）
       IF 欠款超过 60 天 THEN
          发批准书，发货单及赊欠报告
       ELSE （欠款未超期）
             发批准书，发货单
       END IF
END IF
```

- **PDL的优点：**

(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：**

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。