第六章

详细设计

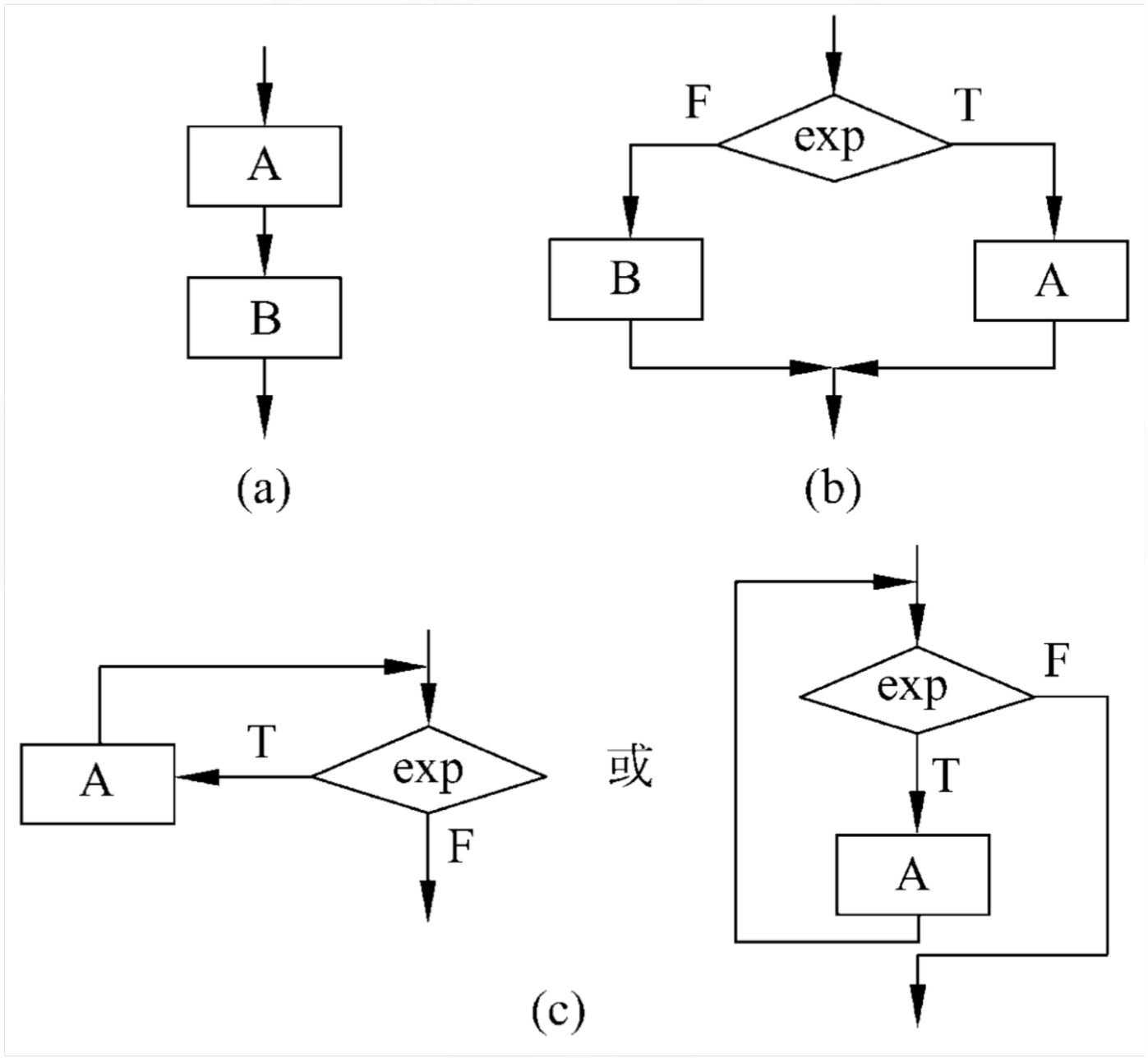
概述

- 详细设计阶段的根本目标
  - 确定应该怎样具体地实现所要求的系统
    - 得出对目标系统的精确描述
    - 在编码阶段可把描述直接翻译成程序
- 详细设计的结果基本上决定了最终的程序代码的质量
  - 考虑程序代码的质量时必须注意“读者”
    - 计算机
    - 人
  - 衡量程序的质量不仅要看它的逻辑是否正确，性能是否满足要求，更主要的是要看它是否容易阅读和理解

结构程序设计

- 结构程序设计的概念最早由E.W.Dijkstra提出
  - 1965年他在一次会议上指出：“可以从高级语言中取消GO TO语句”，“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3种基本的控制结构就能实现任何单入口单出口的程序
- 3种基本的控制结构是
  - 顺序
  - 选择
  - 循环

基本的控制结构



结构程序设计

- 理论上最基本的控制结构只有两种
  - 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)
- 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
  - 创立一种新的程序设计思想、方法和风格，以显著地提高软件生产率和降低软件维护代价

案例

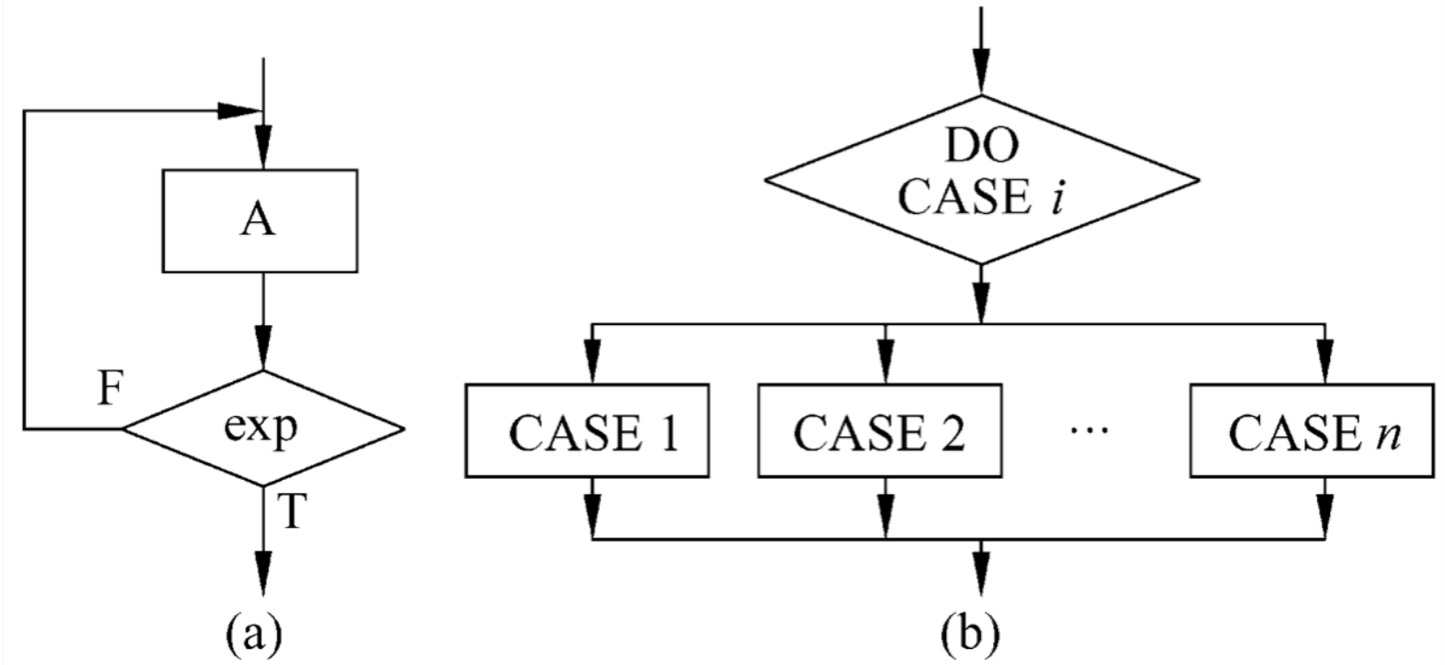
- 1971年IBM公司在纽约时报信息库管理系统的设计中成功地使用了结构程序设计技术，随后在美国宇航局空间实验室飞行模拟系统的设计中，结构程序设计技术再次获得圆满成功
  - 这两个系统都相当庞大
    - 前者包含83 000行高级语言源程序
    - 后者包含40万行源程序
    - 在设计过程中用户需求又曾有过很多改变
    - 两个系统的开发工作都按时并且高质量地完成目标
    - 软件生产率比以前提高了一倍，结构程序设计技术成功地经受了实践的检验
- 1972年IBM公司的Mills进一步提出，程序应该只有一个入口和一个出口，从而补充了结构程序设计的规则

结构程序设计的定义

- 结构程序设计的经典定义
  - “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。”
- 上述经典定义过于狭隘
  - 结构程序设计本质上并不是无GO TO语句的编程方法，而是一种使程序代码容易阅读、容易理解的编程方法
- 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。”

结构程序设计的定义

- 虽然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



结构程序设计的定义

- 有时需要立即从循环(甚至嵌套的循环)中转移出来，如果允许使用LEAVE(或BREAK)结构，则不仅方便而且会使效率提高很多
  - LEAVE或BREAK结构实质上是受限制的GO TO 语句，用于转移到循环结构后面的语句
- 经典的结构程序设计
  - 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
- 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
- 修正的结构程序设计
  - 如果再加上允许使用LEAVE(或BREAK)结构

人机界面设计

- 人机界面设计是接口设计的一个重要的组成部分
  - 对于交互式系统来说，人机界面设计和数据设计、体系结构设计及过程设计一样重要
  - 在个别系统中人机界面的设计工作量甚至占总设计量的一半以上

- 人机界面的设计质量，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

人机界面设计问题

- 常见的4个问题
  - 系统响应时间
  - 用户帮助设施
  - 出错信息处理
  - 命令交互

- 最好在设计初期就把这些问题作为重要的设计问题来考虑
  - 此时修改比较容易，代价也低

Q1-系统响应时间

- 系统响应时间
  - 指从用户完成某个控制动作(例如，按回车键或点击鼠标)，到软件给出预期的响应(输出信息或做动作)之间的时间
- 系统响应时间有两个重要属性
  - 长度
    - 如果系统响应时间过长，用户就会感到紧张和沮丧
    - 当用户工作速度是由人机界面决定的时候，系统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
  - 易变性
    - 指系统响应时间相对于平均响应时间的偏差
    - 即使系统响应时间较长，响应时间易变性低也有助于用户建立起稳定的工作节奏

Q2-用户帮助设施

- 常见的帮助设施可分为
  - 集成帮助设施：从一开始就设计在软件里面
    - 可以缩短用户获得帮助的时间，增加界面的友好性
  - 附加帮助设施：在系统建成后再添加到软件中的，常为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
- 具体设计帮助设施时，必须解决的问题
  - 在用户与系统交互期间，是否在任何时候都能获得关于系统任何功能的帮助信息?
    - 有两种选择：提供部分功能的帮助信息和提供全部功能的帮助信息。
  - 用户怎样请求帮助?
    - 有3种选择：帮助菜单，特殊功能键和HELP命令。
  - 怎样显示帮助信息?
    - 有3种选择：在独立的窗口中，指出参考某个文档(不理想)和在屏幕固定位置显示简短提示。
  - 用户怎样返回到正常的交互方式中?
    - 有两种选择：屏幕上的返回按钮和功能键。
  - 怎样组织帮助信息?
    - 有3种选择：平面结构，信息的层次结构和超文本结构

Q3-出错信息处理

- 出错信息和警告信息，是出现问题时交互式系统给出的“坏消息”
  - 出错信息设计得不好，将向用户提供无用的甚至误导的信息，反而会加重用户的挫折感
- 出错信息或警告信息应该具有的属性
  - 信息应该用用户可以理解的术语描述问题
  - 信息应该提供有助于从错误中恢复的建设性意见
  - 信息应该指出错误可能导致哪些负面后果(例如，破坏数据文件)，以便用户检查是否出现了这些问题，并在确实出现问题时及时解决。
  - 信息应该伴随着听觉上或视觉上的提示
    - 例如，在显示信息时同时发出警告铃声，或者信息用闪烁方式显示，或者信息用明显表示出错的颜色显示
  - 信息不能带有指责感情色彩

Q4-命令交互

- 用户既可以从菜单中选择软件功能，也可以通过键盘命令序列调用软件功能
- 在提供命令交互方式时，必须考虑下列设计问题
  - 是否每个菜单选项都有对应的命令?
  - 采用何种命令形式?有3种选择：控制序列(例如，Ctrl+P)，功能键和键入命令
  - 学习和记忆命令的难度有多大?忘记了命令怎么办?
  - 用户是否可以定制或缩写命令?
  - 是否提供了“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

用户界面的设计过程

- 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
- 用户界面工具箱 / 用户界面开发系统
  - 为简化窗口、菜单、设备交互、出错信息、命令及交互环境的许多其他元素的创建，提供了各种例程或对象
    - 既可以基于语言的方式也可以用基于图形的方式

用户界面设计的评估标准

- 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
- 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
- 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
- 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

人机交互设计的指导建议

- 一般交互指南
  1. 保持一致性
     - 应该为人机界面中的菜单选择、命令输入、数据显示以及众多的其他功能，使用一致的格式
  2. 提供有意义的反馈
     - 应向用户提供视觉的和听觉的反馈，以保证在用户和系统之间建立双向通信
  3. 在执行有较大破坏性的动作之前要求用户确认
     - 如果用户要删除一个文件，或覆盖一些重要信息，或终止一个程序的运行
  4. 允许取消绝大多数操作
     - UNDO或REVERSE功能
  5. 减少在两次操作之间必须记忆的信息量
  6. 提高对话、移动和思考的效率
     - 应该尽量减少用户击键的次数，设计屏幕布局时应该考虑尽量减少鼠标移动的距离
  7. 允许犯错误
     - 系统应该能保护自己不受严重错误的破坏
  8. 按功能对动作分类，并据此设计屏幕布局
  9. 提供对用户工作内容敏感的帮助设施
  10. 用简单动词或动词短语作为命令名

人机交互设计的指导建议

- 信息显示指南

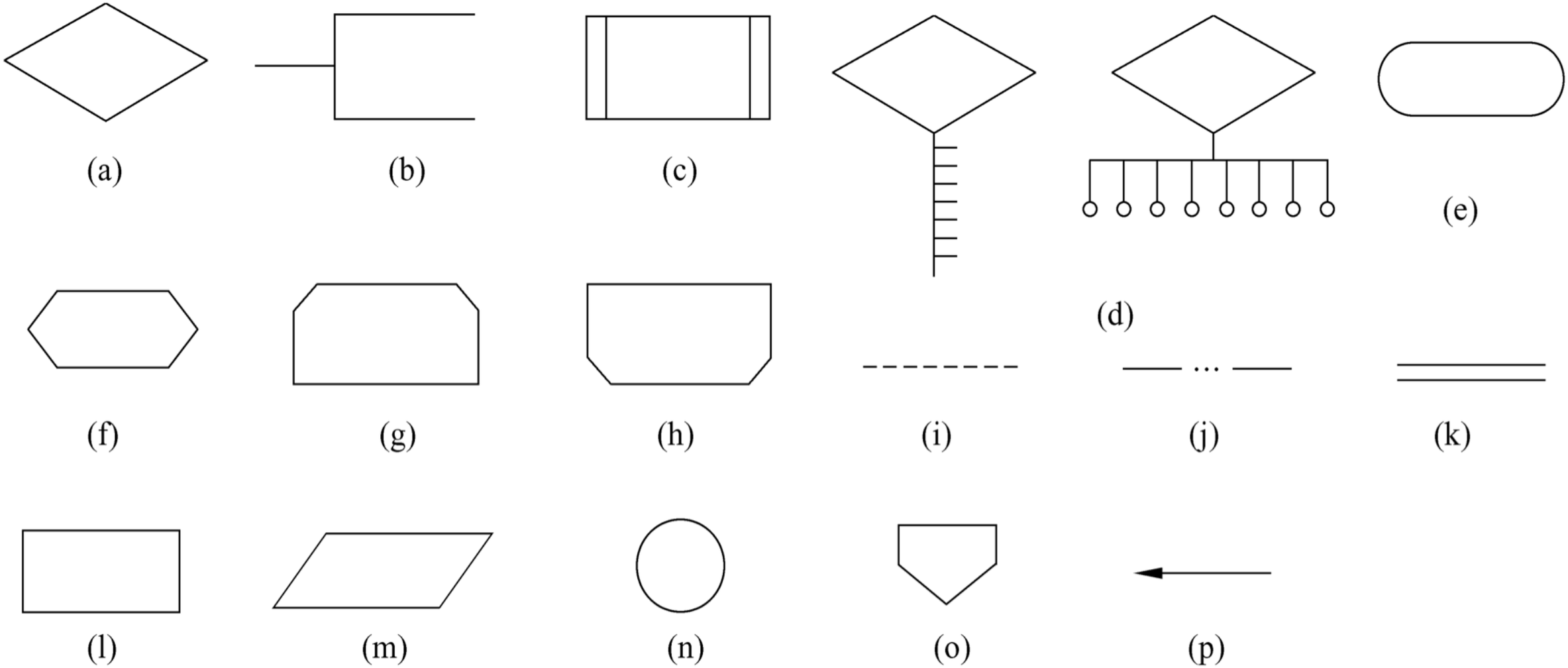
1. 只显示与当前工作内容有关的信息
   - 用户不必看到无关的数据、菜单和图形
2. 不要用数据淹没用户
   - 例如，可以用图形或图表来取代庞大的表格
3. 使用一致的标记、标准的缩写和可预知的颜色
   - 显示的含义应该非常明确，用户无须参照其他信息源就能理解
4. 允许用户保持可视化的语境
5. 产生有意义的出错信息
6. 使用大小写、缩进和文本分组以帮助理解
7. 使用窗口分隔不同类型的信息
8. 使用“模拟”显示方式表示信息，以使信息更容易被用户提取
9. 高效率地使用显示屏

人机交互设计的指导建议

- 数据输入指南
  1. 尽量减少用户的输入动作
  2. 保持信息显示和数据输入之间的一致性
     - 显示的视觉特征应该与输入域一致
  3. 允许用户自定义输入
  4. 交互应该是灵活的，并且可调整成用户最喜欢的输入方式
  5. 使在当前动作语境中不适用的命令不起作用
  6. 让用户控制交互流
  7. 对所有输入动作都提供帮助
  8. 消除冗余的输入

过程设计的工具

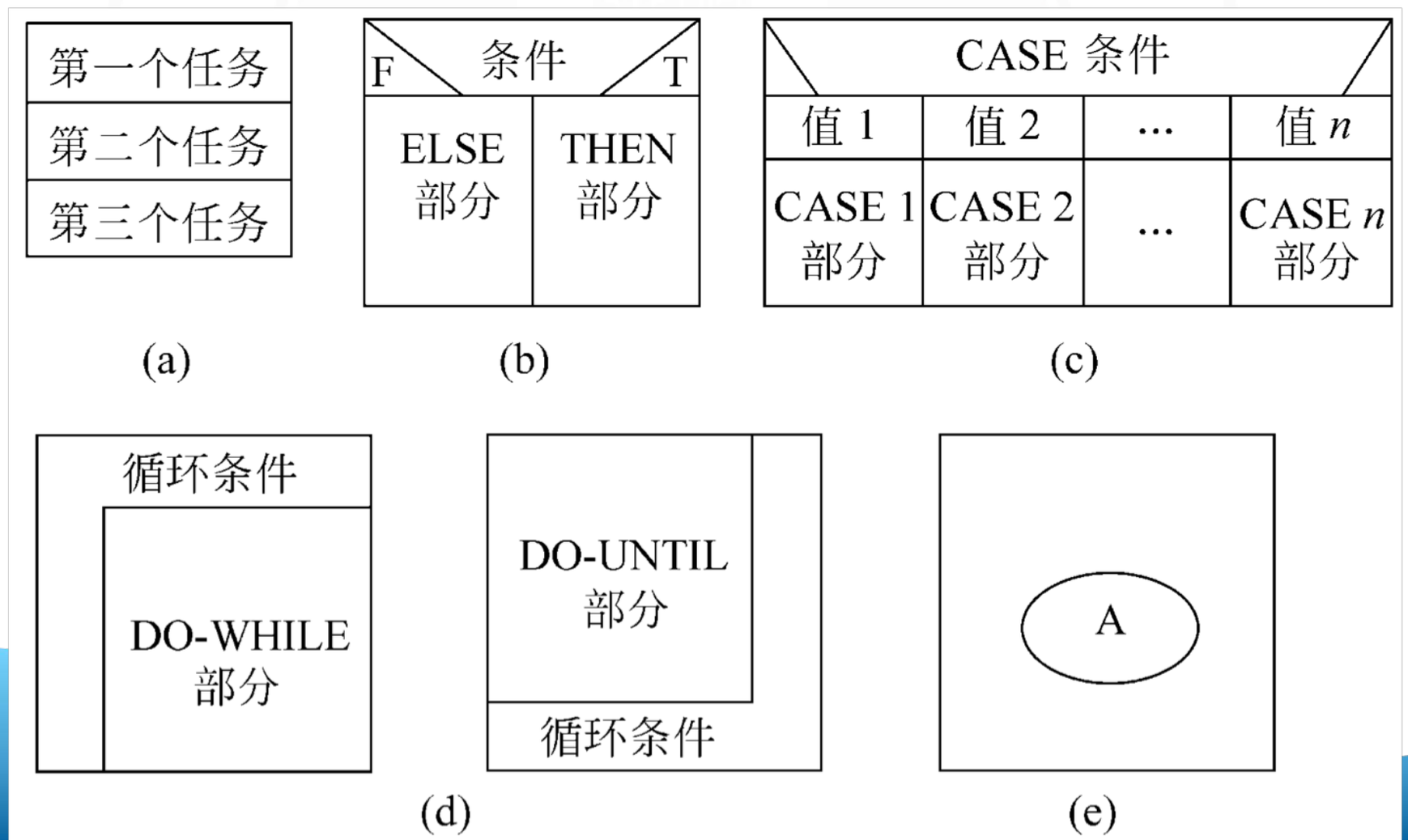
- 程序流程图
  - 程序流程图本质上不是逐步求精的好工具
    - 容易误导程序员过早地考虑程序的控制流程，而不去考虑程序的全局结构
  - 程序流程图中用箭头代表控制流
    - 程序员不受任何约束，可以完全不顾结构程序设计的精神，随意转移控制。
  - 程序流程图不易表示数据结构



过程设计工具

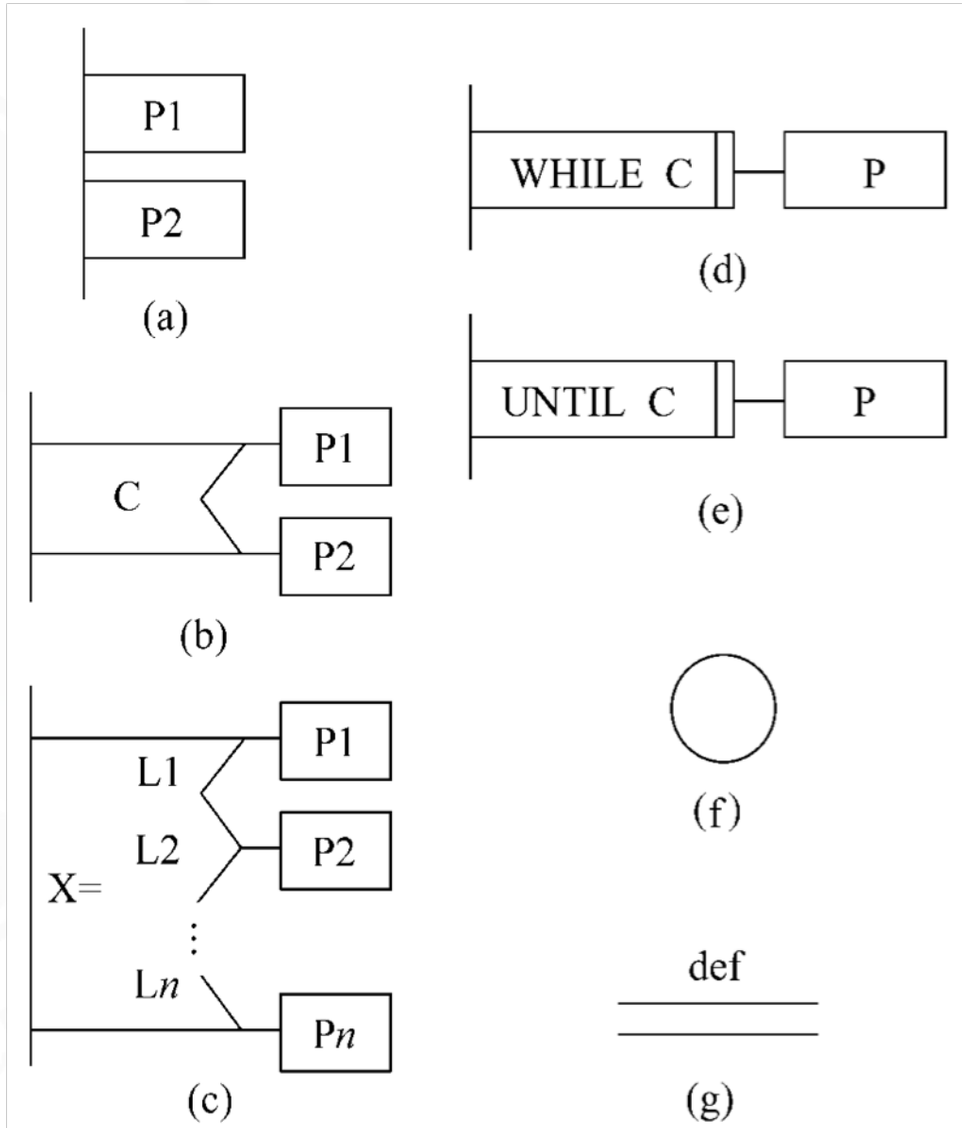
- 盒图(N-S图)
  - Nassi和Shneiderman提出了盒图，又称为N-S图

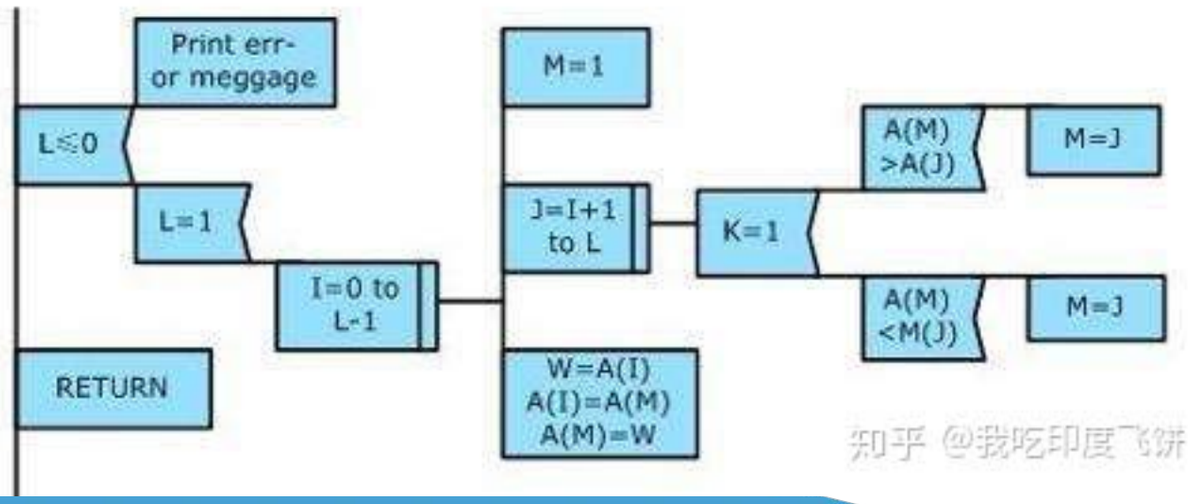
(1)功能域(即，一个特定控制结构的作用域)明确，可以从盒图上一眼就看出来
(2)不可能任意转移控制
(3)很容易确定局部和全程数据的作用域
(4)很容易表现嵌套关系，也可以表示模块的层次结构



过程设计工具

- PAD图(problem analysis diagram)
  - 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
  - PAD图所描绘的程序结构十分清晰
  - 用PAD图表现程序逻辑，易读、易懂、易记
  - 容易将PAD图转换成高级语言源程序，可用软件工具自动完成
  - 即可用于表示程序逻辑，也可用于描绘数据结构
  - PAD图的符号支持自顶向下、逐步求精方法的使用





过程设计工具

- 判定表
  - 判定表却能够清晰地表示复杂的条件组合与应做的动作之间的对应关系
- 判定表由4部分组成

  (1)左上部列出所有条件

  (2)左下部是所有可能做的动作

  (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
  - 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

实例

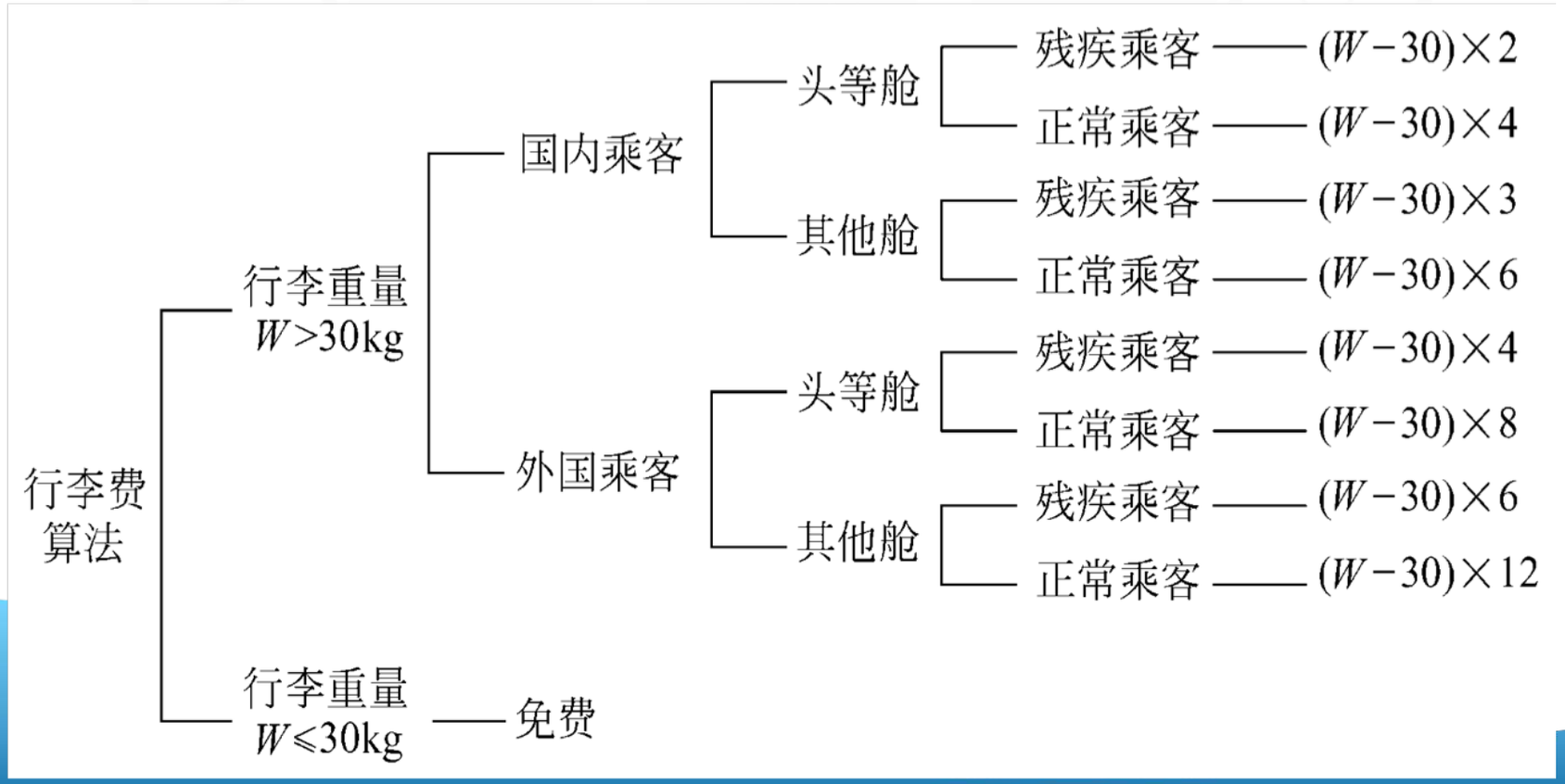
- 下面以行李托运费的算法为例说明判定表的组织方法
- 假设某航空公司规定，乘客可以免费托运重量不超过30kg的行李
  - 当行李重量超过30kg时
  - 对头等舱的国内乘客超重部分每公斤收费4元
  - 对其他舱的国内乘客超重部分每公斤收费6元
  - 对外国乘客超重部分每公斤收费比国内乘客多一倍
  - 对残疾乘客超重部分每公斤收费比正常乘客少一半
- 用判定表可以清楚地表示与上述每种条件组合相对应的计算行李费的算法

判定表

	规则								
	1	2	3	4	5	6	7	8	9
国内乘客		T	T	T	T	F	F	F	F
头等舱		T	F	T	F	T	F	T	F
残疾乘客		F	F	T	T	F	F	T	T
行李重量 $W \leqslant 30$kg	T	F	F	F	F	F	F	F	F
免费	×								
$(W-30)\times 2$				×					
$(W-30)\times 3$					×				
$(W-30)\times 4$		×						×	
$(W-30)\times 6$			×						×
$(W-30)\times 8$						×			
$(W-30)\times 12$							×		

过程设计工具

- 判定树
  - 判定表虽然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，但其含义不易读，需要间断学习
  - 当数据元素的值多于两个时(例如，假设对机票需细分为头等舱、二等舱和经济舱等多种级别时)，判定表的简洁程度也将下降
  - 判定树是判定表的变种，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



过程设计工具

- 过程设计语言
  - PDL，也称为伪码
  - 特点
    - 关键字的固定语法，提供了结构化控制结构、数据说明和模块化的特点
    - 自然语言的自由语法，它描述处理特点。
    - 数据说明的手段
    - 模块定义和调用的技术，应该提供各种接口描述模式

过程设计工具

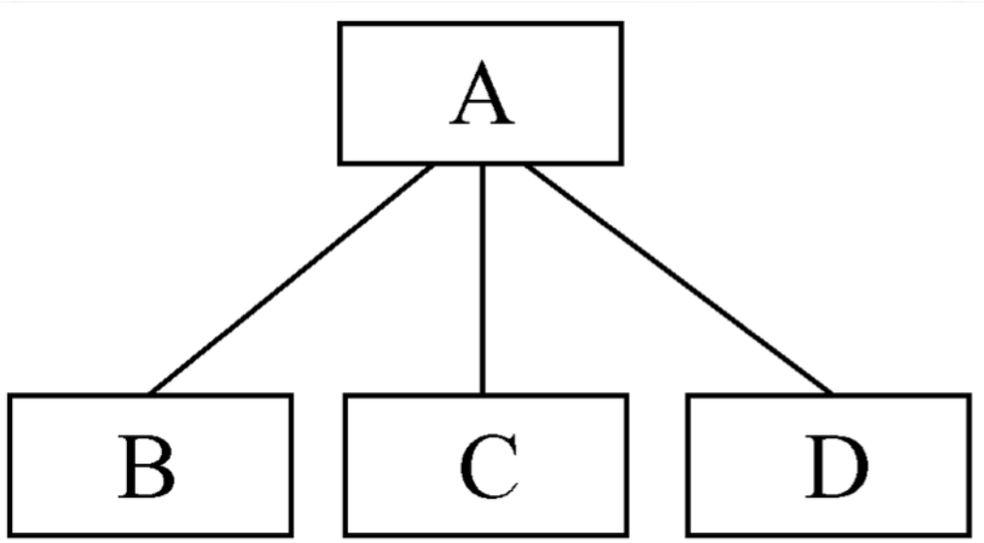
- 过程设计语言的优点
  - 可以作为注释直接插在源程序中间
  - 可以使用普通的正文编辑程序或文字处理系统书写和编辑
  - 已经有自动处理程序存在，而且可以自动由PDL生成程序代码
- 缺点
  - 不如图形工具形象直观，描述复杂的条件组合与动作间的对应关系时，不如判定表清晰简单

面向数据结构的设计方法

- 最终目标
  - 得出对程序处理过程的描述
- 最适合于在详细设计阶段使用
  - 在完成了软件结构设计之后，用于设计每个模块的处理过程

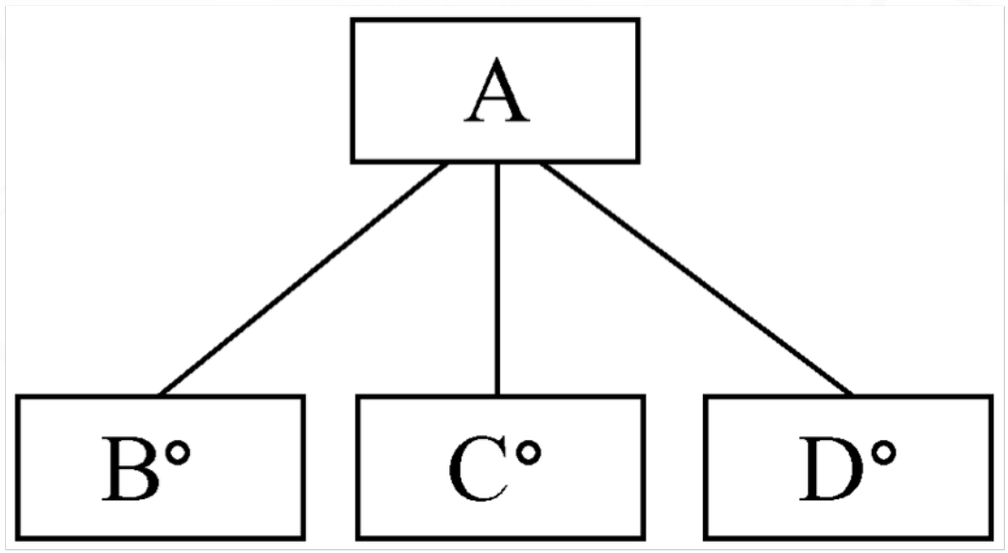
面向数据结构的设计方法

- Jackson图
  - 顺序结构
    - 顺序结构的数据由一个或多个数据元素组成，每个元素按确定次序出现一次



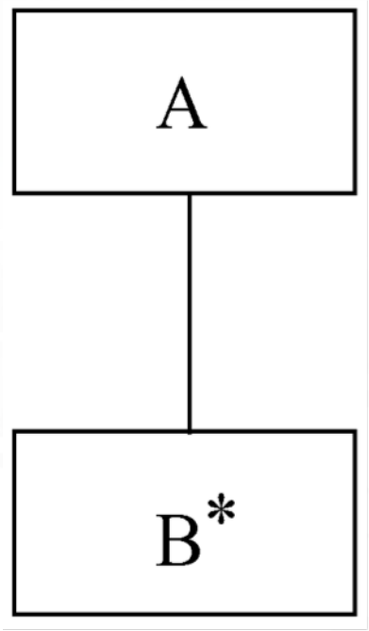
面向数据结构的设计方法

- Jackson图
  - 选择结构
    - 选择结构的数据包含两个或多个数据元素，每次使用这个数据时按一定条件从这些数据元素中选择一个



面向数据结构的设计方法

- Jackson图
  - 重复结构
    - 重复结构的数据，根据使用时的条件由一个数据元素出现零次或多次构成

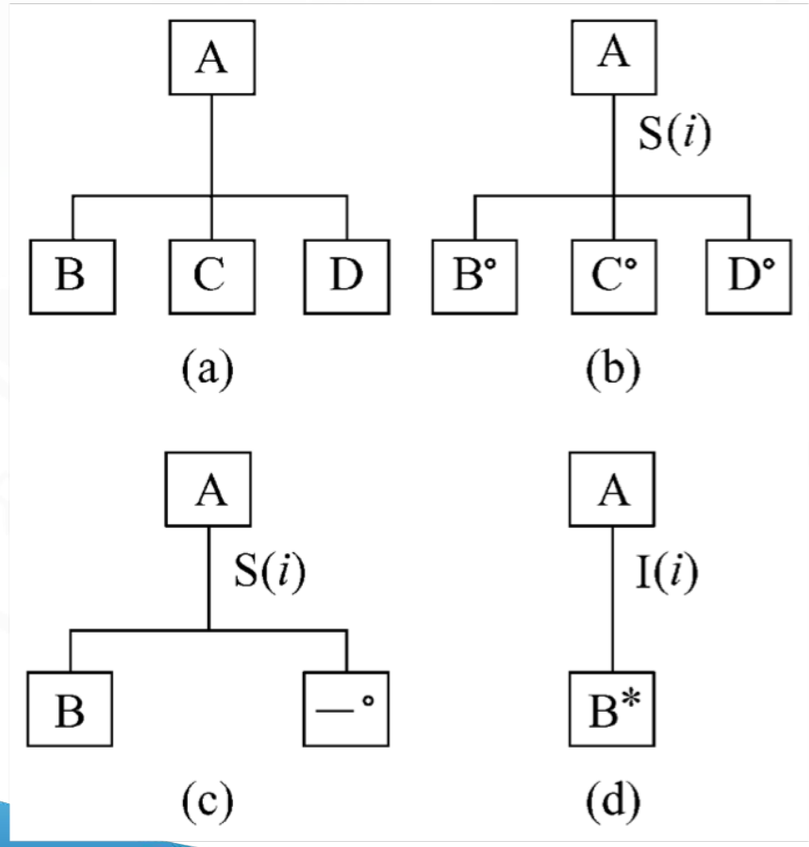


面向数据结构的设计方法

- Jackson图的优点
  - 便于表示层次结构，而且是对结构进行自顶向下分解的有力工具
  - 形象直观可读性好
  - 既能表示数据结构也能表示程序结构
    - 因为结构程序设计也只使用上述3种基本控制结构
- Jackson图的缺点
  - 用这种图形工具表示选择或重复结构时，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，也不易直接把图翻译成程序

面向数据结构的设计方法

- 改进的Jackson图
  - 用这种图形工具表示选择或重复结构时，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，也不易直接把图翻译成程序



面向数据结构的设计方法

- Jackson方法的执行步骤

(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

- 所谓有对应关系是指
  - 有直接的因果关系，在程序中可以同时处理的数据单元
    - 对于重复出现的数据单元必须重复的次序和次数都相同才可能有对应关系

面向数据结构的设计方法

- Jackson方法的执行步骤

(3) 用以下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

- 第一，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
  - 注意，如果这对数据单元在输入数据结构和输出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之中在数据结构图中层次低的那个对应
- 第二，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
- 第三，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
- 描绘程序结构的Jackson图应该综合输入数据结构和输出数据结构的层次关系而导出
  - 在导出程序结构图的过程中，由于改进的Jackson图规定在构成顺序结构的元素中不能有重复出现或选择出现的元素，因此可能需要增加中间层次的处理框

面向数据结构的设计方法

- Jackson方法的执行步骤

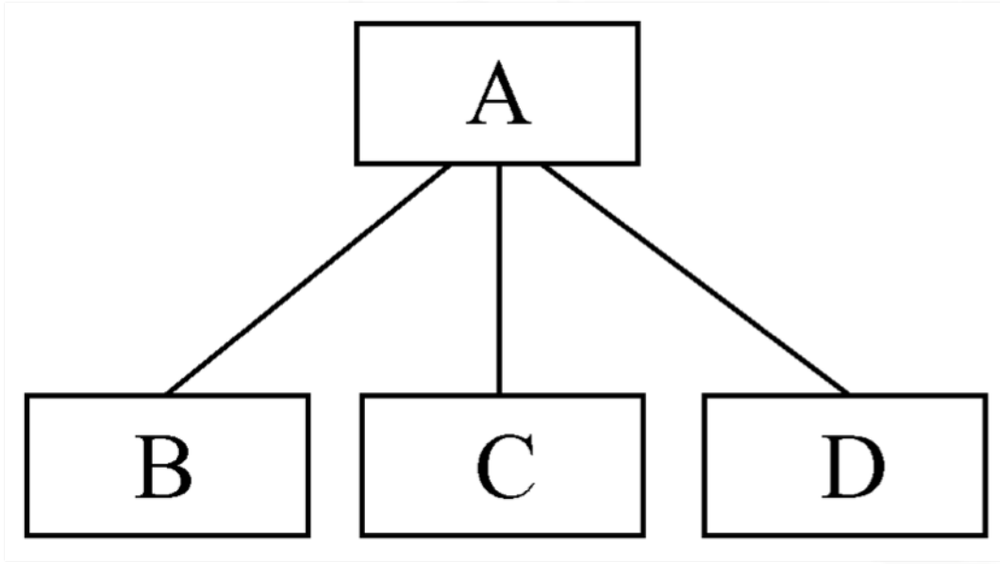
(4) 列出所有操作和条件(包括分支条件和循环结束条件)，并且把它们分配到程序结构图的适当位置

(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

面向数据结构的设计方法

- Jackson图
  - 顺序结构
    - 顺序结构的数据由一个或多个数据元素组成，每个元素按确定次序出现一次



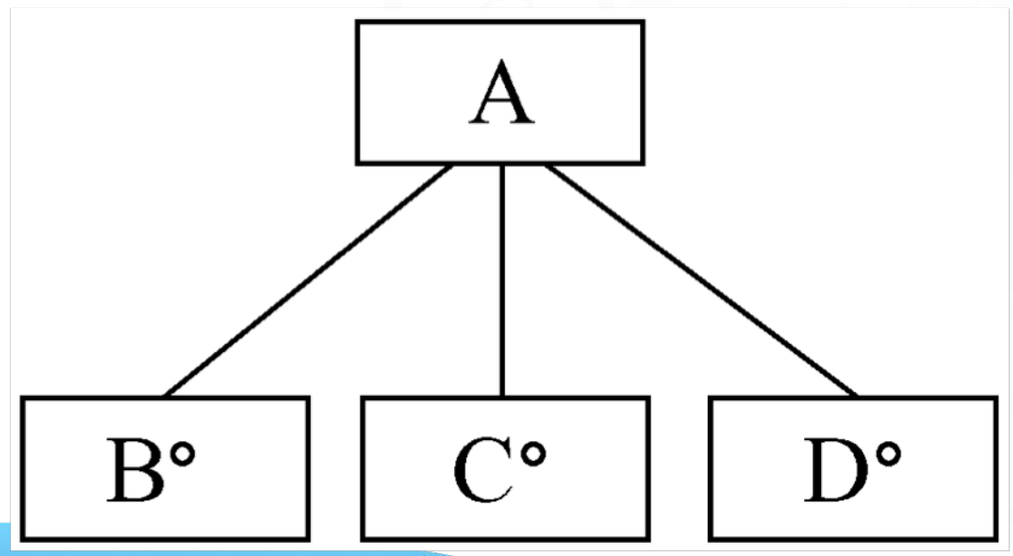
顺序结构对应的伪码

其中‘seq’和‘end’是关键字：

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

面向数据结构的设计方法

- Jackson图
  - 选择结构
    - 选择结构的数据包含两个或多个数据元素，每次使用这个数据时按一定条件从这些数据元素中选择一个



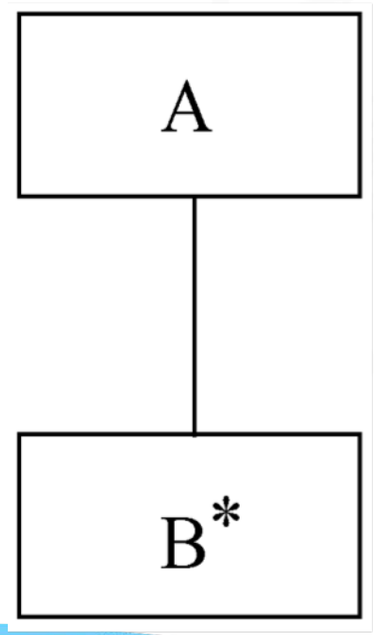
选择结构对应的伪码

其中'select'、'or'和'end'是关键字，cond1、cond2和cond3分别是执行B、C或D的条件

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

面向数据结构的设计方法

- Jackson图
  - 重复结构
    - 重复结构的数据，根据使用时的条件由一个数据元素出现零次或多次构成



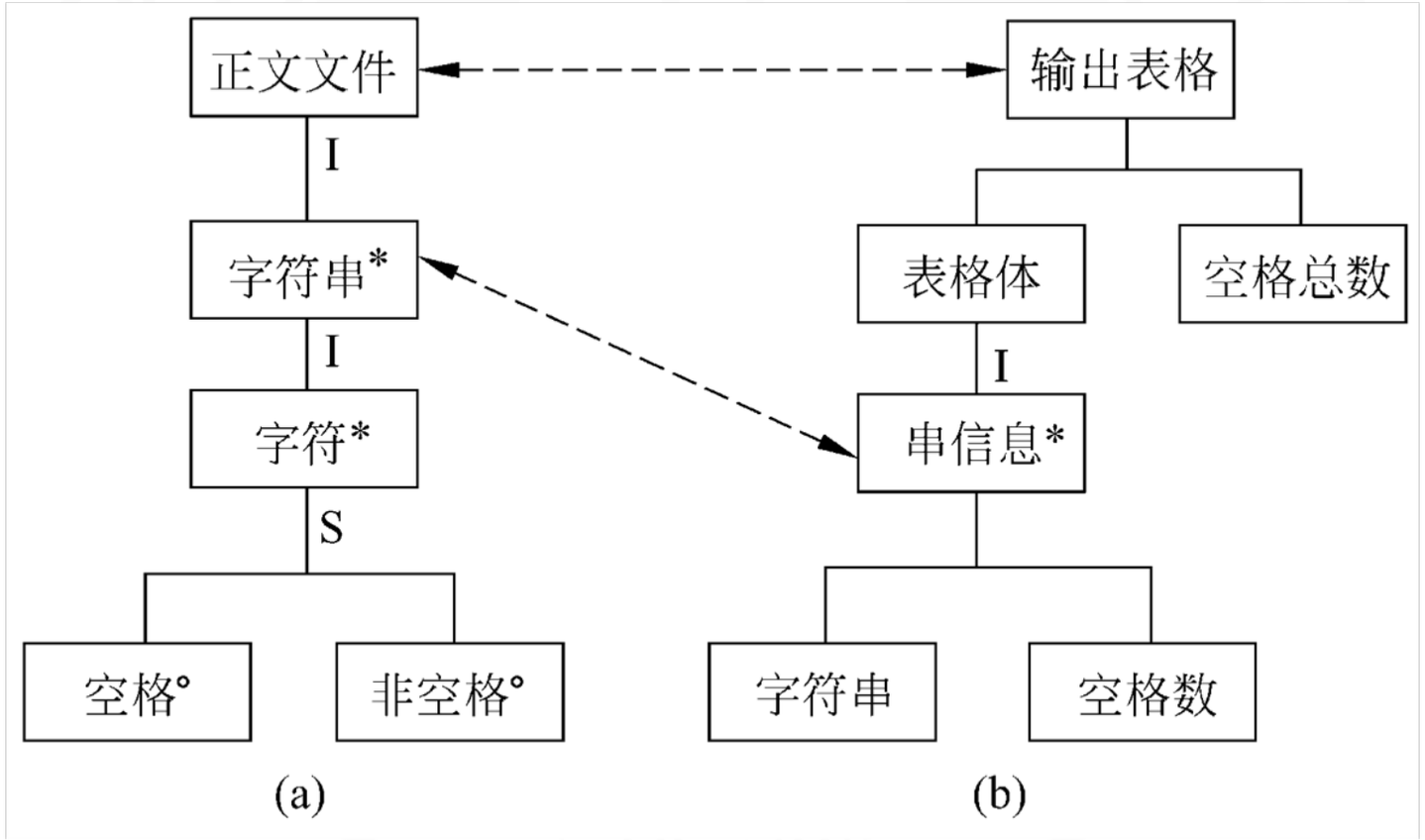
重复结构对应的伪码

其中’iter’、‘until’、‘while’和‘end’是关键字，cond是条件：

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

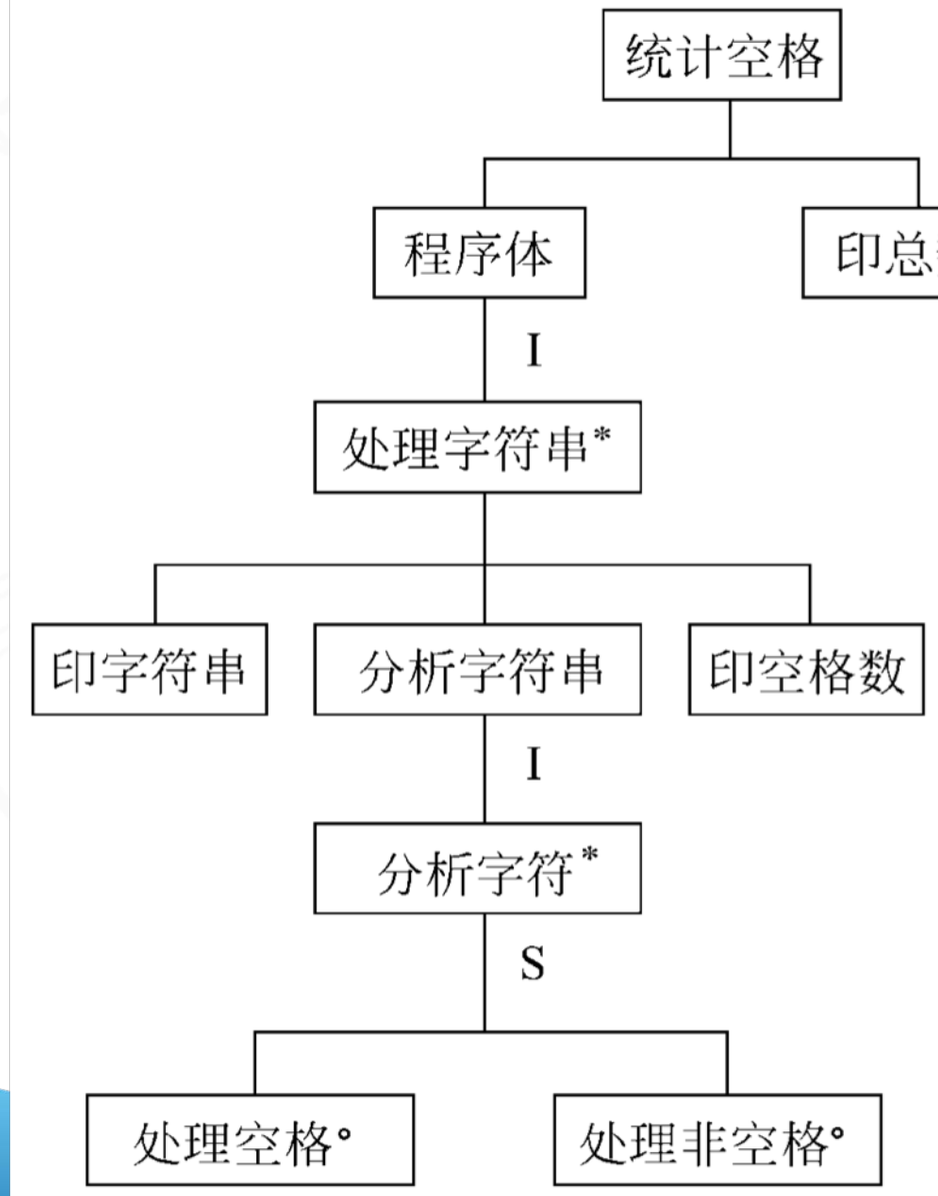
实例

- 一个正文文件由若干个记录组成，每个记录是一个字符串。要求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。要求的输出数据格式是，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



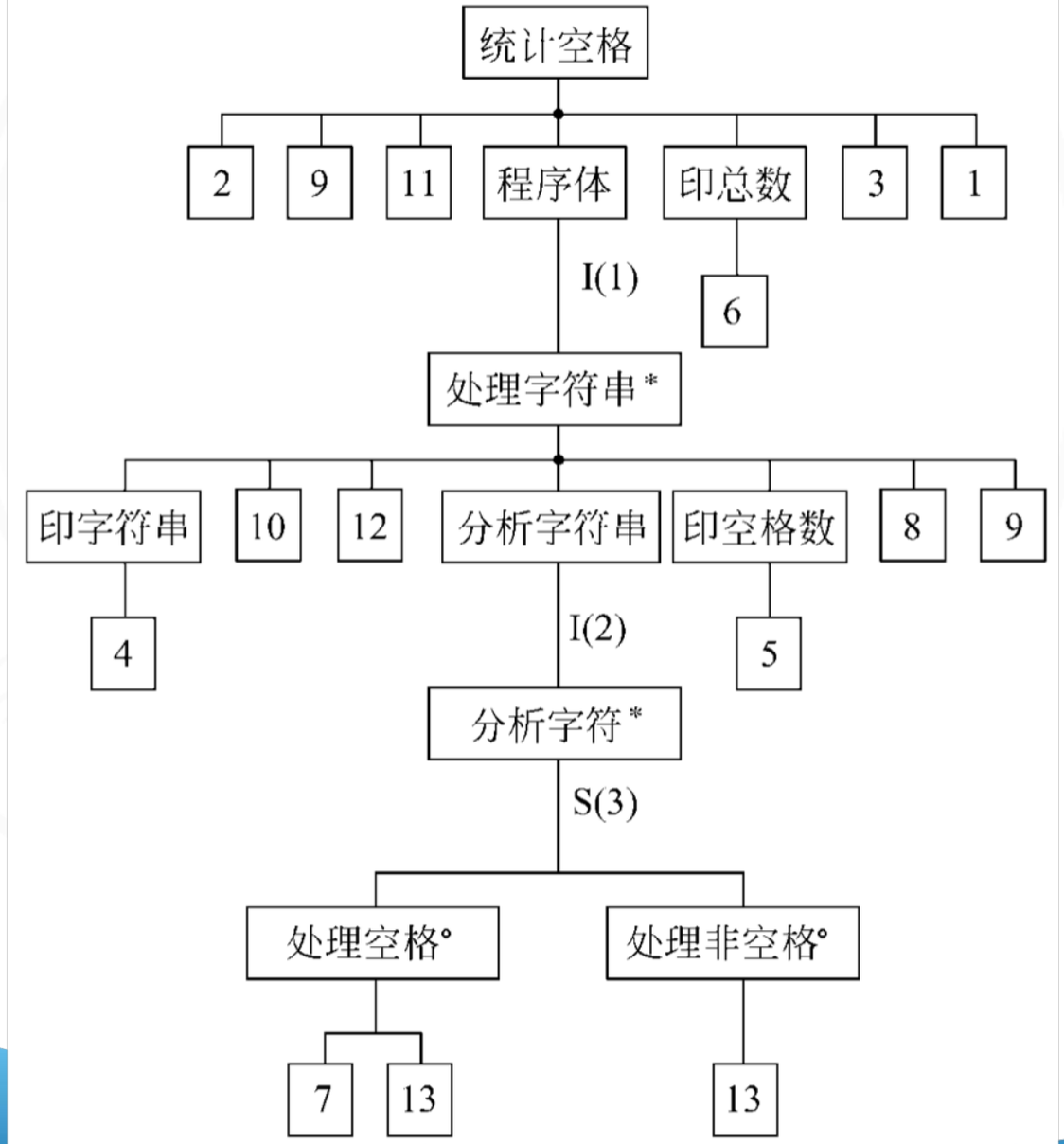
实例

- 一个正文文件由若干个记录组成，每个记录是一个字符串。要求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。要求的输出数据格式是，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



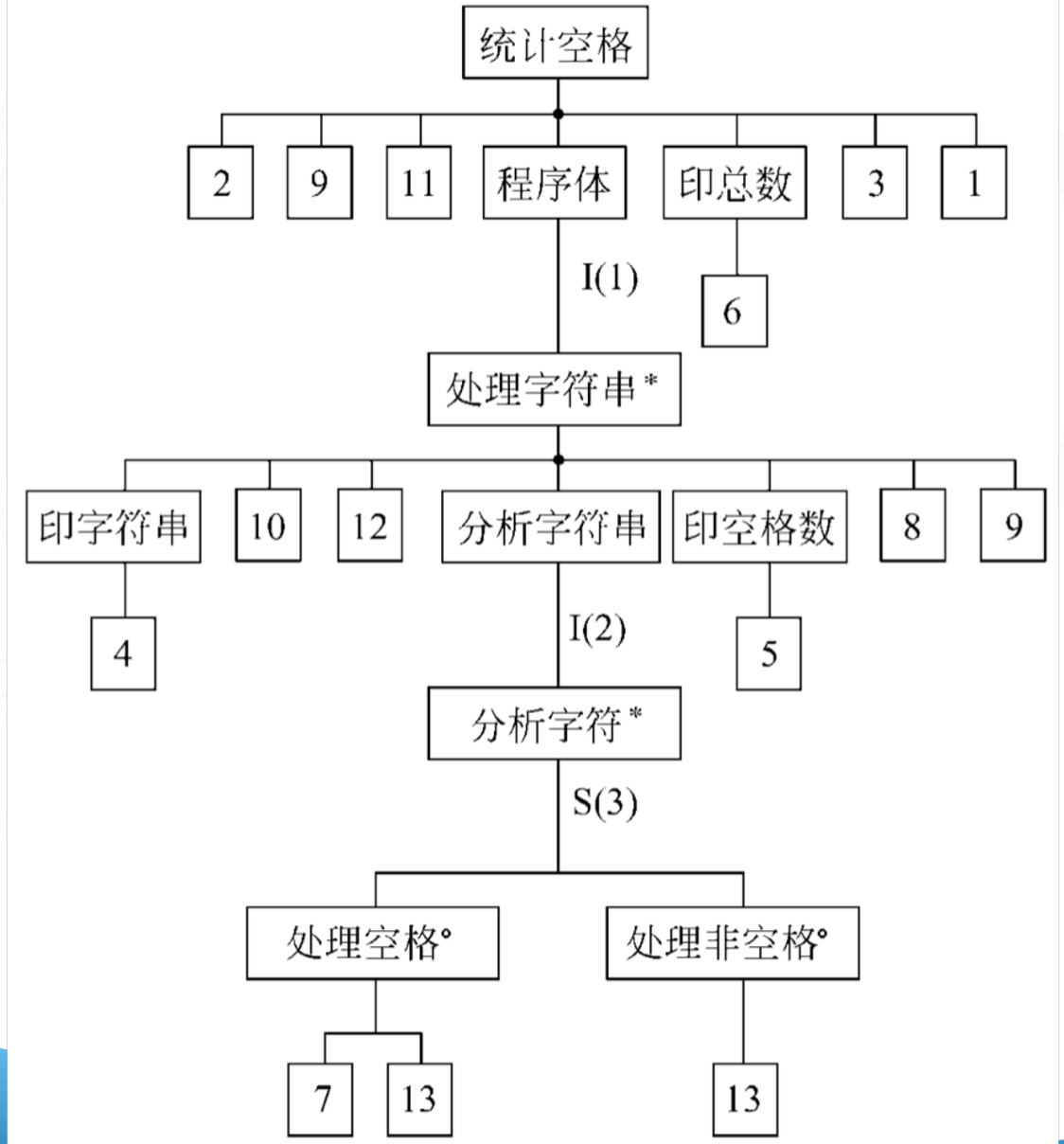
实例

- 一个正文文件由若干个记录组成，每个记录是一个字符串。要求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。要求的输出数据格式是，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



实例

- 一个正文文件由若干个记录组成，每个记录是一个字符串。要求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。要求的输出数据格式是，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



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 until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

程序复杂程度的定量度量

- 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
- 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
- 程序的定量的复杂程度可以作为模块规模的精确限度

程序复杂程度的定量度量

- 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，度量出的结果称为程序的环形复杂度
  - 所谓流图实质上是“退化了的”程序流程图
    - 它仅仅描绘程序的控制流程
    - 完全不表现对数据的具体操作以及分支或循环的具体条件
    - 用任何方法表示的过程设计结果，都可以翻译成流图
- 画法
  - 在流图中用圆表示结点，一个圆代表一条或多条语句
    - 程序流程图中的一个顺序的处理框序列和一个菱形判定框，可以映射成流图中的一个结点。
  - 流图中的箭头线称为边，它和程序流程图中的箭头线类似，代表控制流
    - 在流图中一条边必须终止于一个结点，即使这个结点并不代表任何语句
      - 实际上相当于一个空语句
  - 由边和结点围成的面积称为区域
    - 当计算区域数时应该包括图外部未被围起来的区域

PDL

procedure:sort

1: do while records remain

2: read record;

if record field 1=0

3: then process record;

store in buffer;

incremert counter;

4: elseif record field 2=0

5: then reset counter;

6: else process record;

store in file;

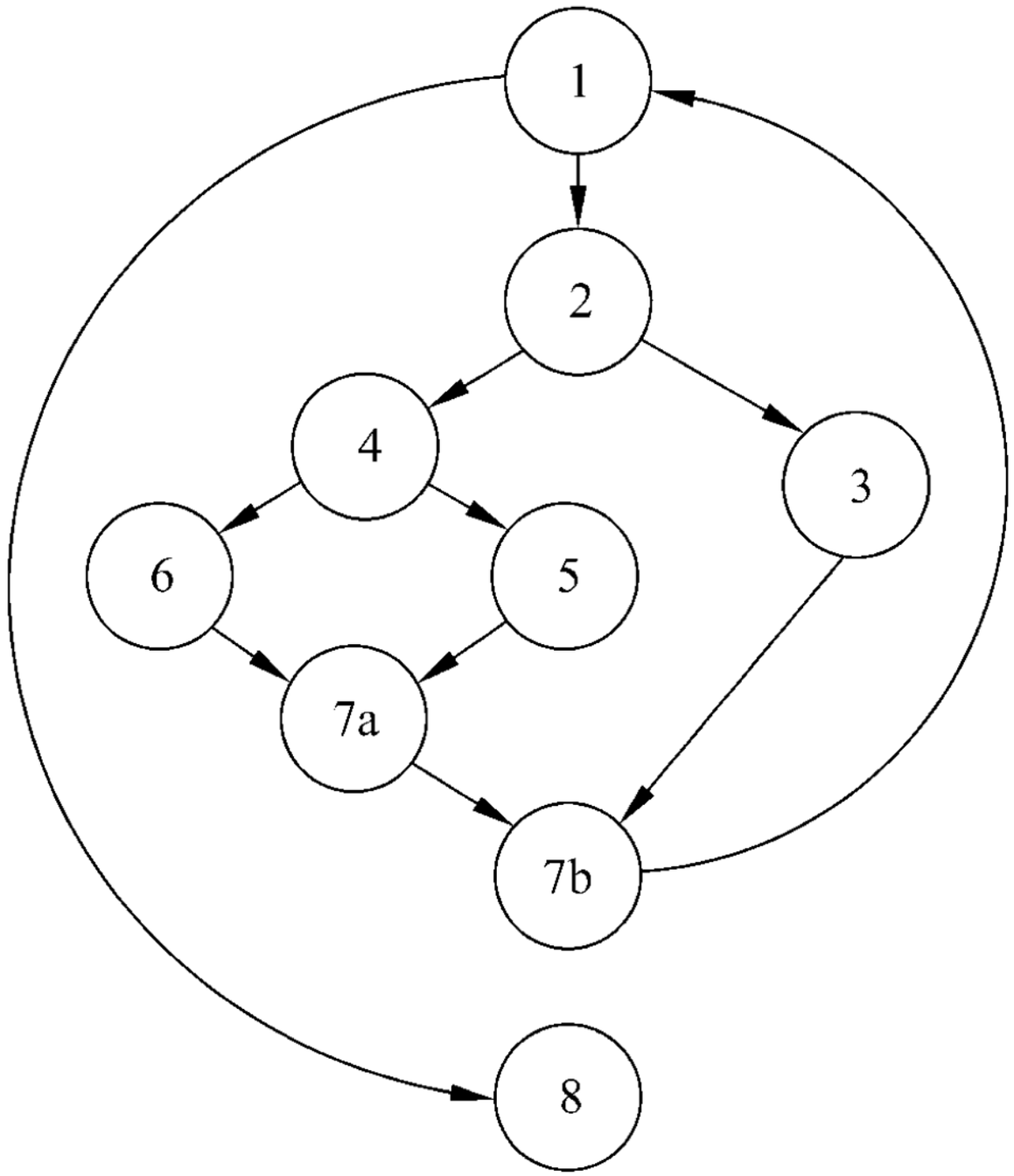
7a: endif

endif

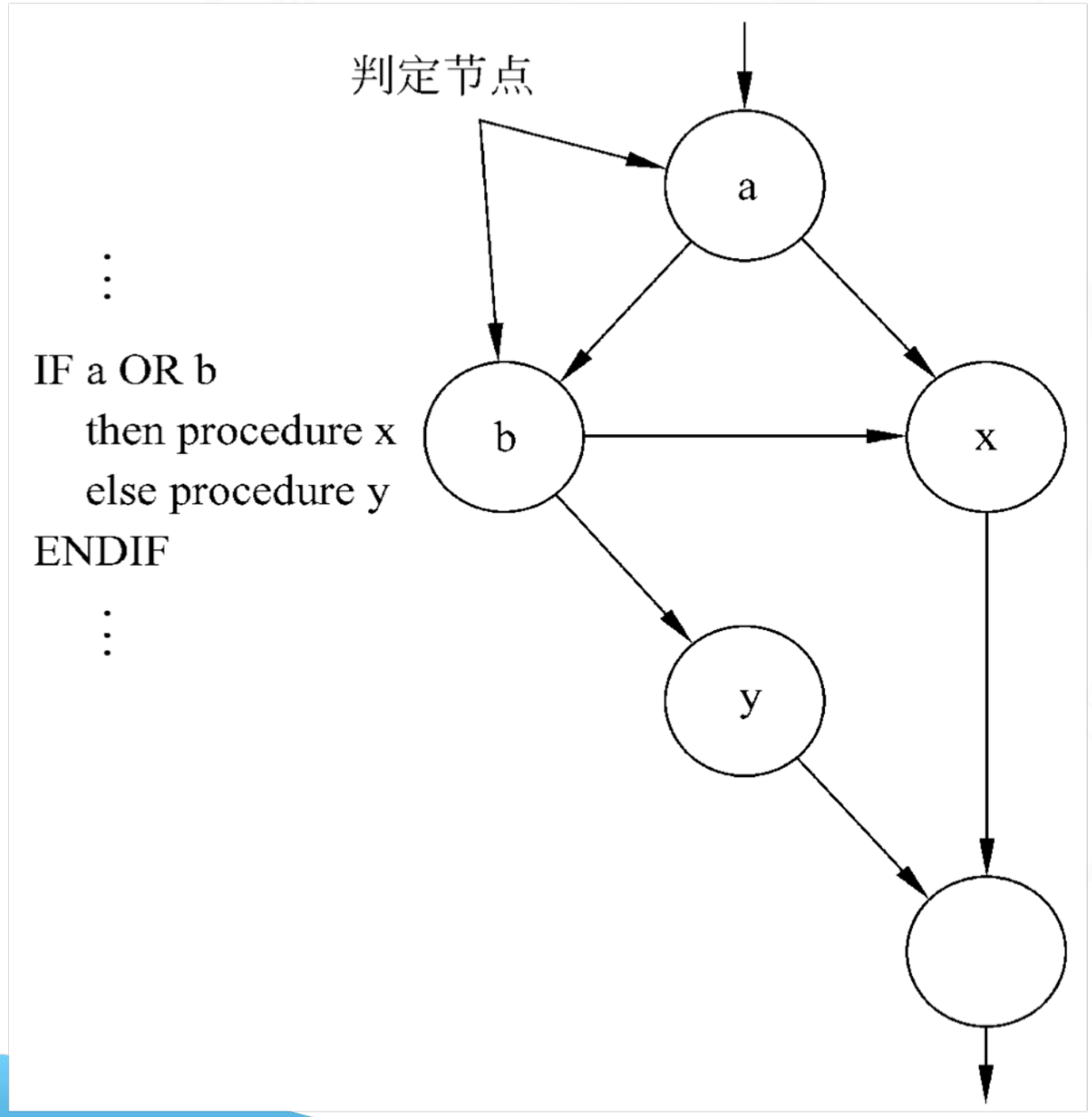
7b: enddo

8: end

流图



❖ 由包含复合条件的PDL映射成的流图



计算环形复杂度的方法

- 环形复杂度定量度量程序的逻辑复杂度
- 基于流图，可以用下述3种方法中的任何一种来计算环形复杂度。

(1) 流图中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

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

P是流图中判定结点的数目

计算环形复杂度的方法

- 环形复杂度高的程序往往是最困难、最容易出问题的程序
  - 实践表明，模块规模以V(G)≤10为宜
    - V(G)=10是模块规模的一个更科学更精确的上限