

# 第6章（结构化）详细设计

## 目录

- 6.1 结构程序设计
- 6.3 过程设计的工具
- 6.5 程序复杂度的定量度量

# 详细设计的目标

确定如何具体实现所要求的系统，即得出对目标系统的精确描述，从而在编码阶段可以把这个描述翻译成用某种程序设计语言书写的程序。

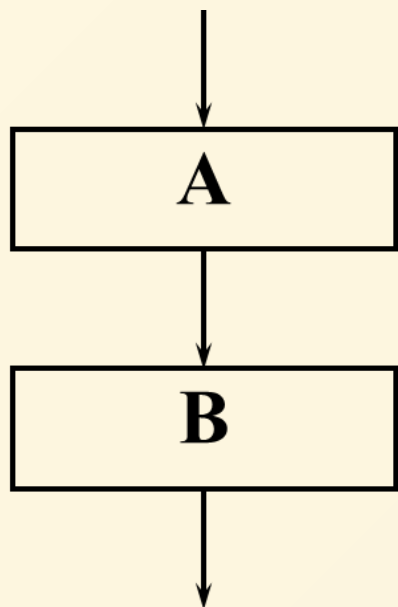
不是具体编写程序，而是设计程序的“蓝图”，以后程序员将根据这个“蓝图”写出实际的代码。

# 6.1 结构程序设计

- E.W.Dijkstra 最早提出结构程序设计：程序质量与程序中包含的Goto语句的数量成反比（1965）。
- 1966，Bohm, Jacopini，证明了只用“顺序”、“选择”、“循环”控制结构就能实现任何单入口单出口程序。

# 顺序结构

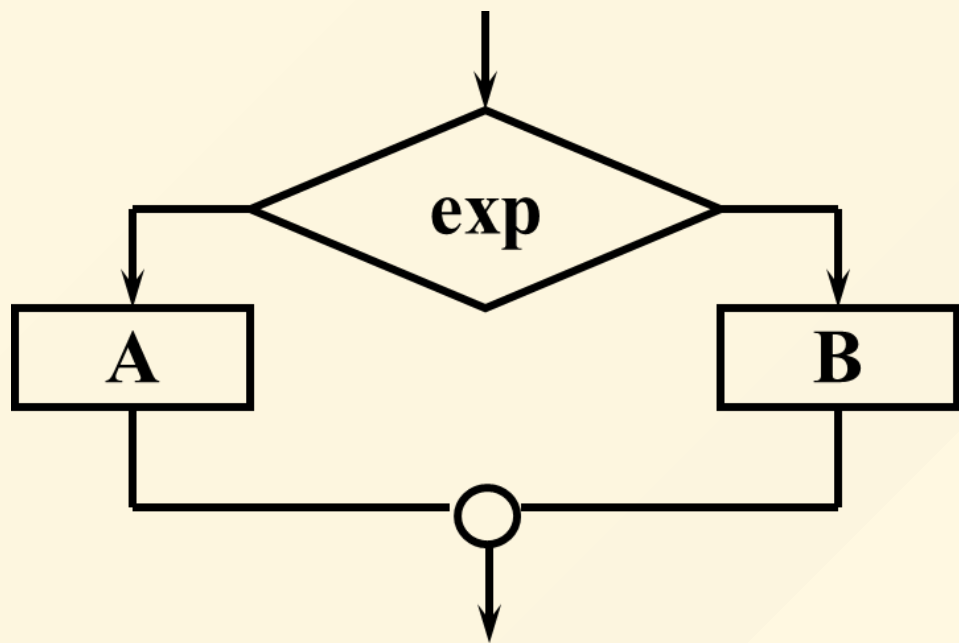
相当于“A、B”



**(a)**顺序结构

# 选择结构

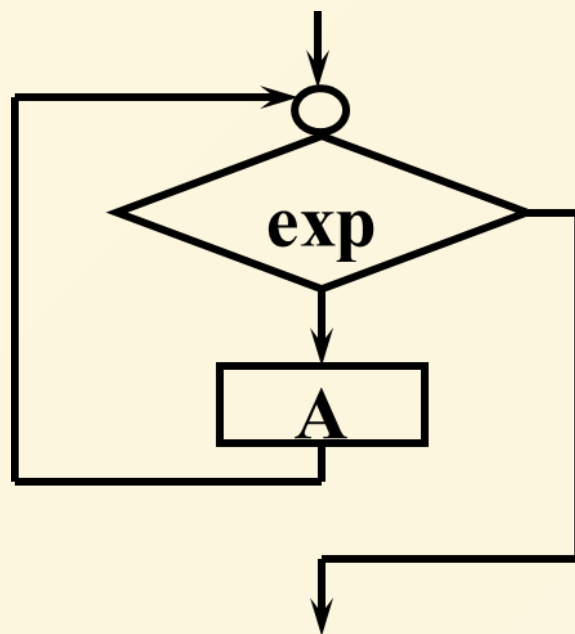
相当于“If exp then A else B endif ”



(b)选择结构

# 循环结构

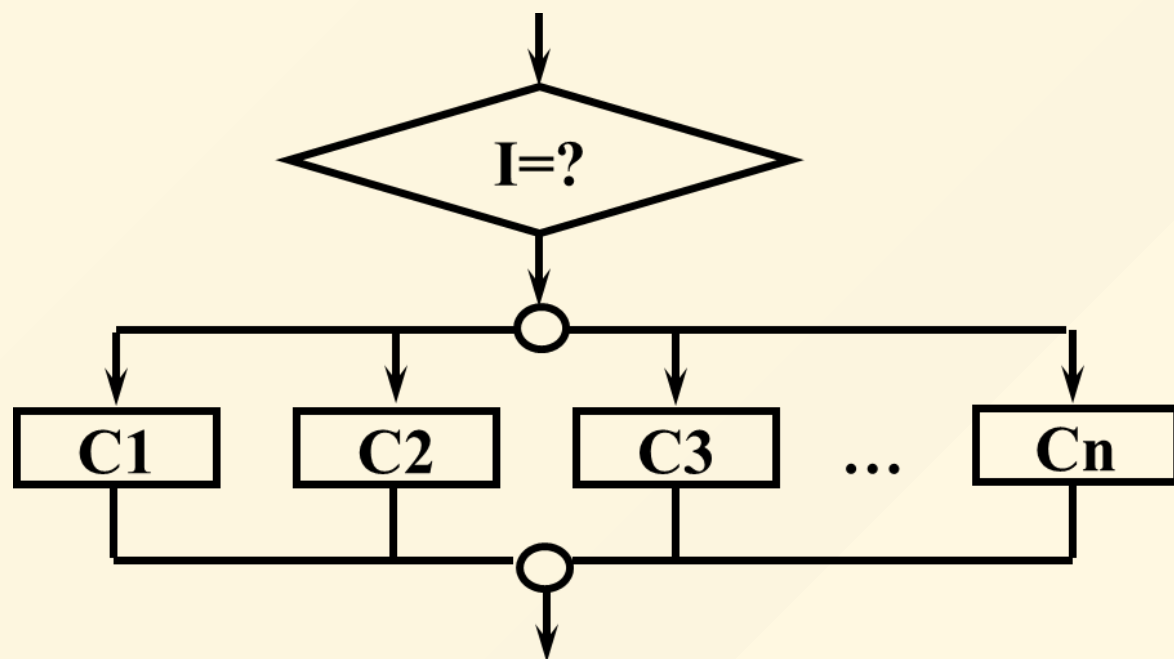
相当于“While exp do A”



(c) 循环结构

# 扩充：多分支结构

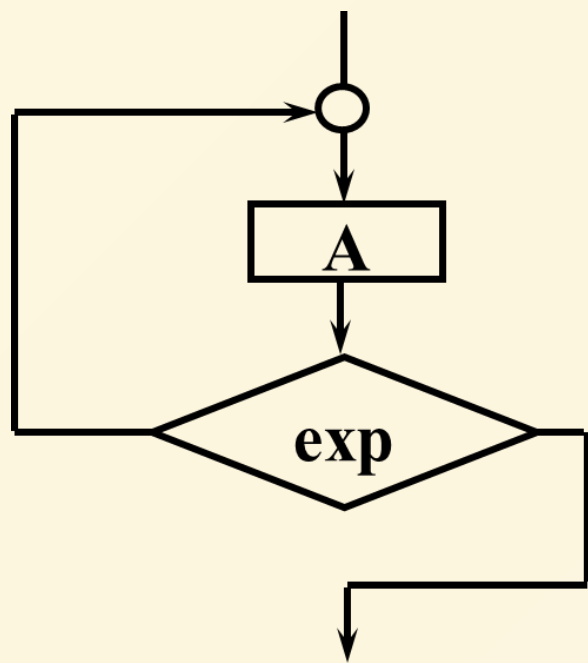
相当于“Case I of I=1:C1; I=2:C2; I=3:C3; ... ; I=n:Cn”



(d) 多分支结构

# 扩充：DO\_UNTIL循环结构

相当于“Repeat A Until exp”



(e)UNTIL循环



# 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、选择和循环这三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。
- 经典的结构程序设计：只允许使用顺序、IF\_THEN\_ELSE选择和WHILE\_DO循环
- 扩展的结构程序设计：除了三种基本控制结构，还使用多分支和DO\_UNTIL循环
- 修正的结构程序设计：除了三种基本控制结构和两种扩充结构，还使用CONTINUE和BREAK（从本次循环或整个循环结构中转移出来）。

# 使用结构程序设计技术的好处

- 提高软件开发工程的成功率和生产率；
- 系统有清晰的层次结构，容易阅读理解；
- 单入口单出口的控制结构，容易诊断纠正；
- 模块化可以使得软件可以重用；
- 程序逻辑结构清晰，有利于程序正确性证明。

## 6.3 过程设计的工具

### 程序流程图

- 是一种描述程序的控制结构流程和指令执行情况的有向图。
- 优点：历史悠久、使用广泛、直观描绘控制流程、便于初学者掌握。
- 缺点：
  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。

# 盒图（N-S图）

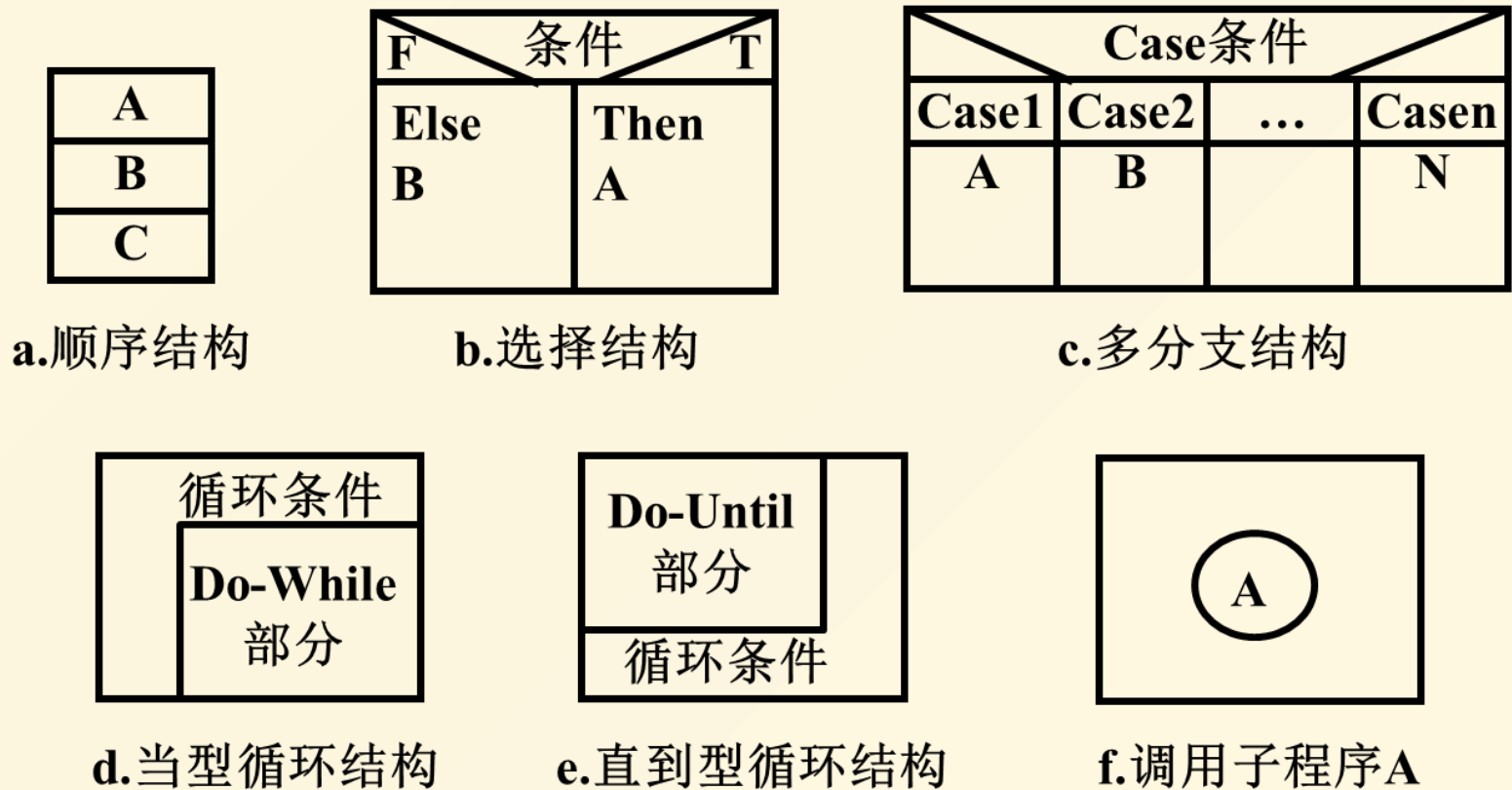
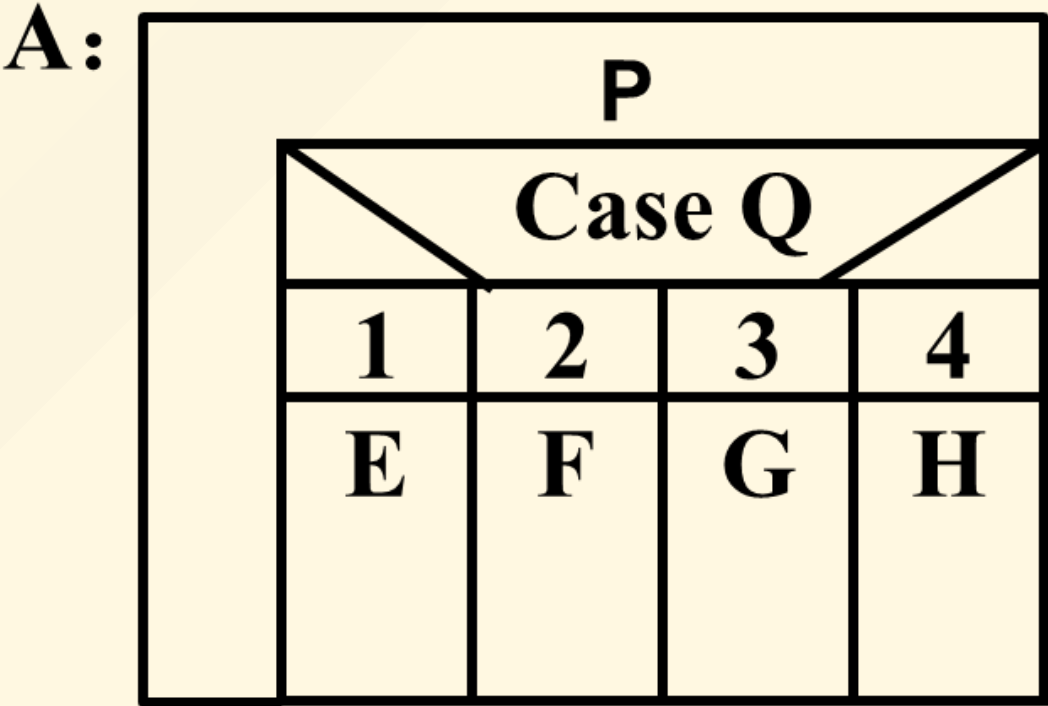
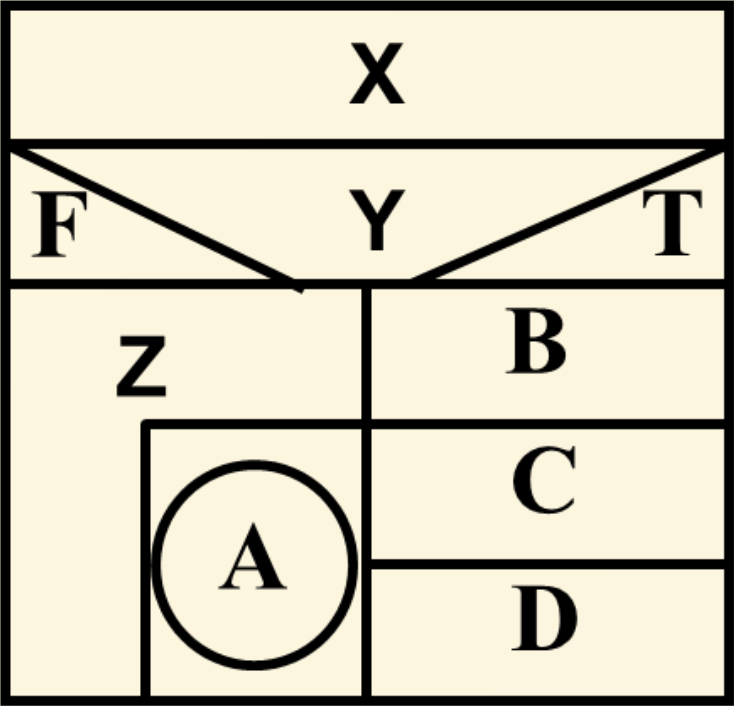


图6.4 盒图的基本符号

# 盒图的特点

1. 功能域明确，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。

# 盒图举例



# PAD图

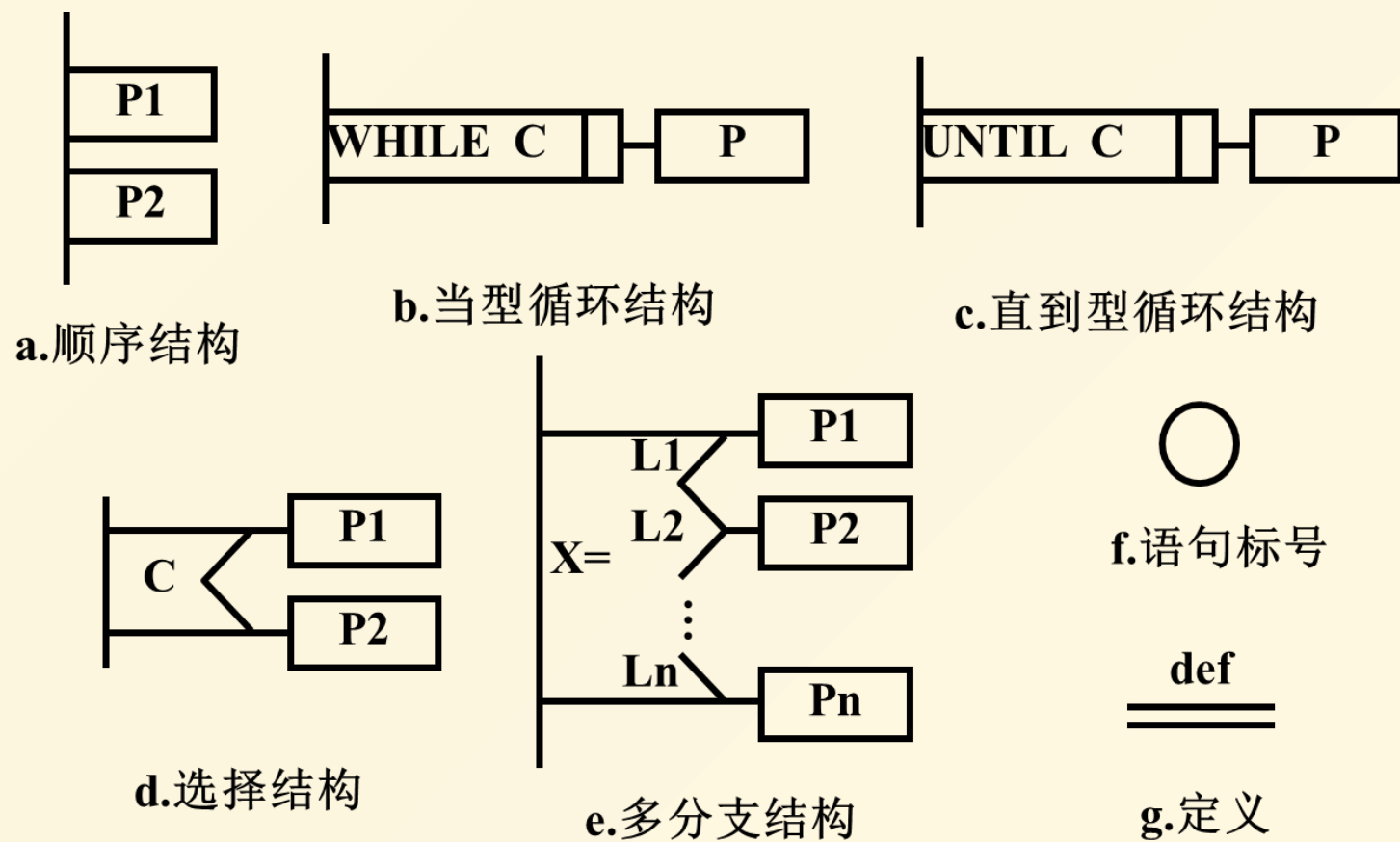
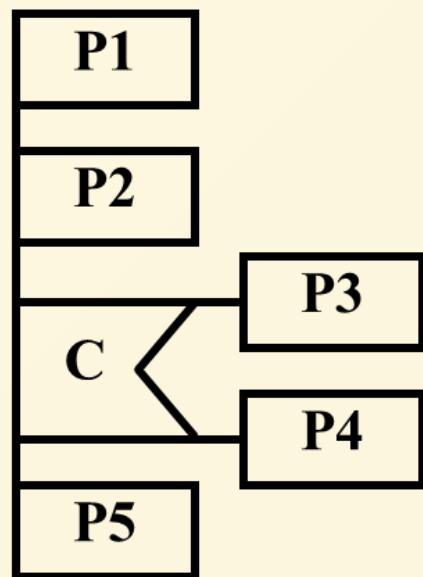
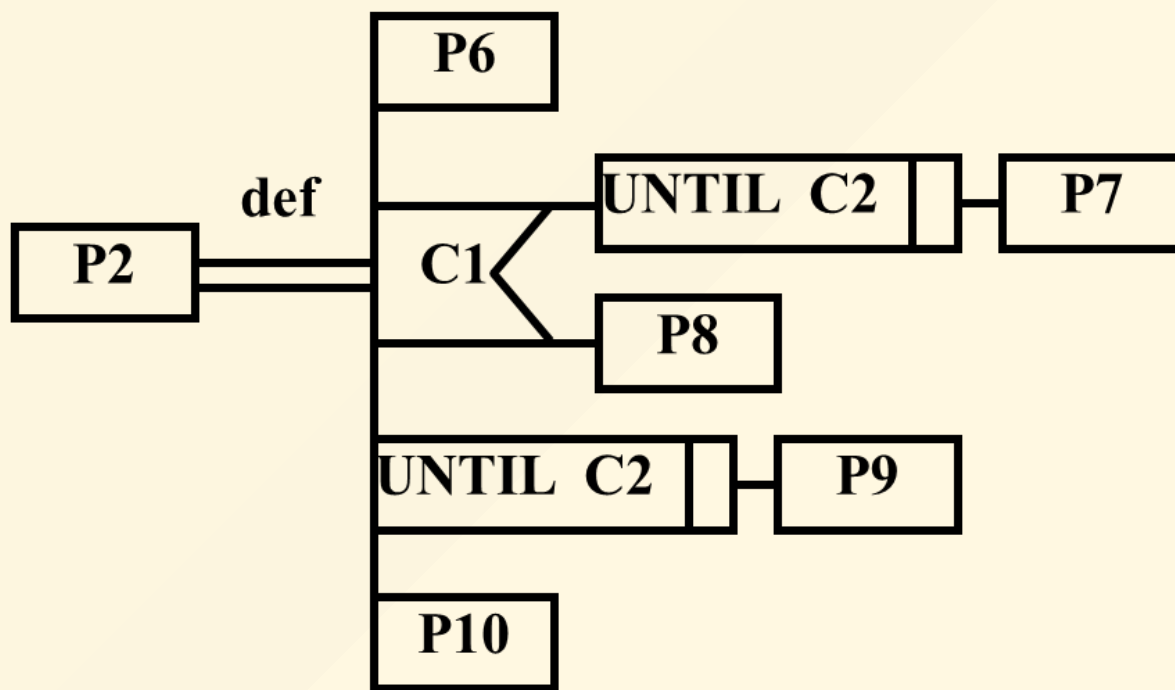


图6.5 PAD图的基本符号

# PAD图举例



a. 初始的PAD图



b. 使用def符号细化处理框P2

图6.6 PAD图例子



# PAD图的优点

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，通俗易懂，程序从图中最左竖线上端的结点开始执行，自上而下，从左向右顺序执行，遍历所有结点。
4. 容易将PAD图转换成高级语言源程序，这种转换可以用软件工具自动完成。
5. 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、逐步求精的方法。

# 判定表

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，对其它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。

		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 $W \leq 30$		T	F	F	F	F	F	F	F	F
免费		×								
$(W-30) \times 2$					×					
$(W-30) \times 3$						×				
$(W-30) \times 4$			×						×	
$(W-30) \times 6$				×						×
$(W-30) \times 8$							×			
$(W-30) \times 12$								×		

# 判定表的优缺点

- 优点：能够简洁而又无歧义地描述处理规则。
- 缺点：不能表示顺序和重复等处理特性。

# 判定树

判定树是判定表的变种。

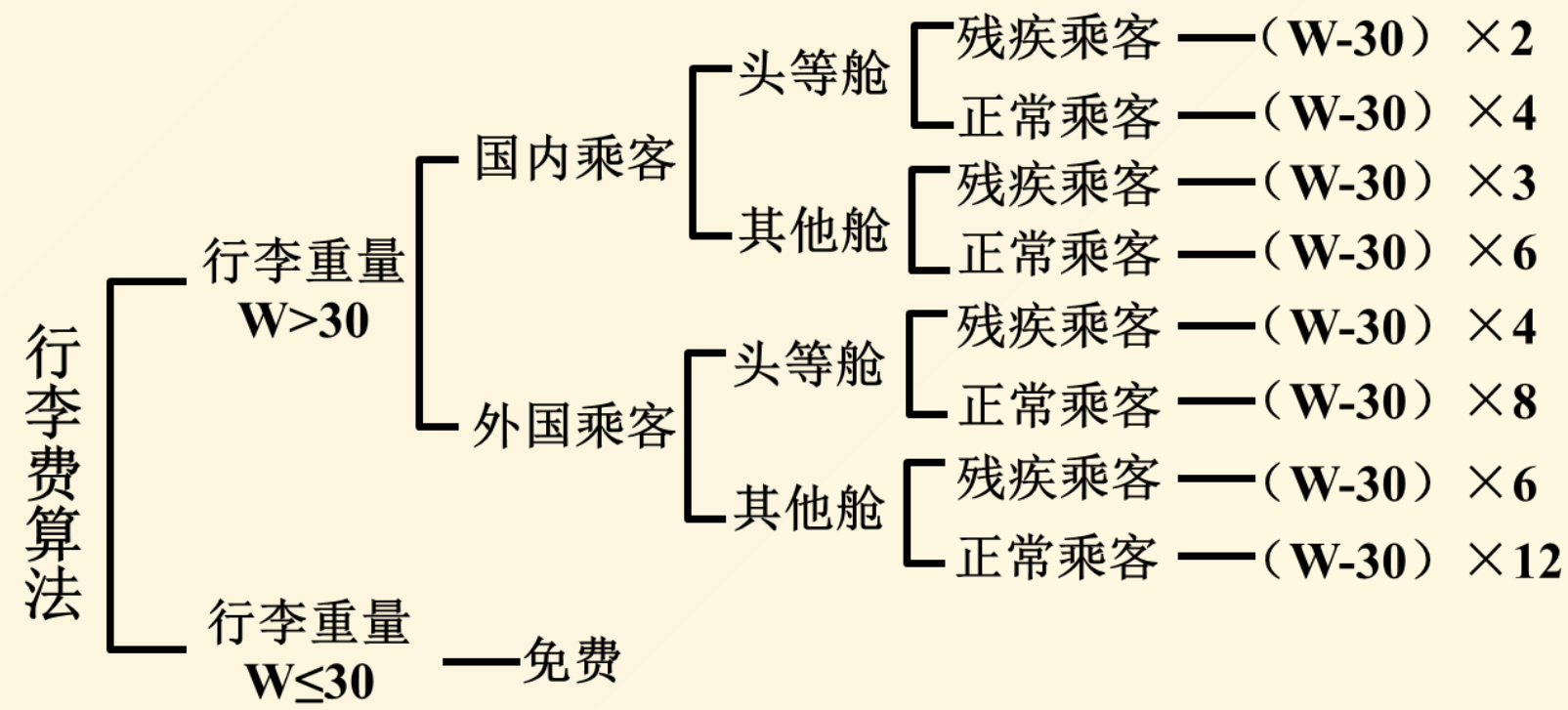


图6.7 用判定树表示计算行李费的算法

# 过程设计语言（PDL）

PDL也称为伪码（ pseudocode ）。

如：

```
if  I>0  then
    执行订单数据输入模块
else
    报告出错信息
end if
```

# PDL的优缺点

- 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来完成PDL的书写和编辑工作
  3. 现在已经有一些自动处理程序可以自动地把PDL生成程序代码
- 缺点：

不如图形工具形象直观

# 6.5 程序复杂度的定量度量

## 定量度量程序复杂度的作用

1. 可估算软件中错误的数量及软件开发工作量；
2. 度量的结果可用来比较不同设计或不同算法的优劣；
3. 程序的复杂度可作为模块规模的限度。

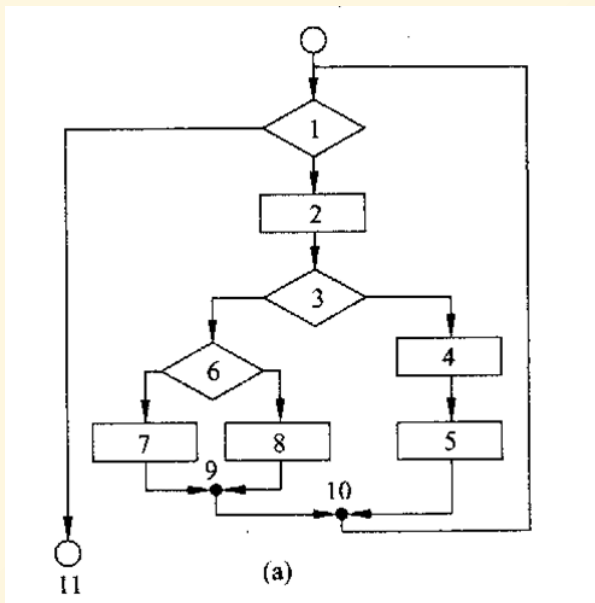
# McCabe方法

## 流图

“退化”的程序流程图，仅描绘程序的控制流程，不表现对数据的具体操作及循环、选择的条件。



## 程序流程图 → 流图



一个顺序的处理框序列 → 一个结点；  
一个顺序的处理框序列和一个判定框 →  
一个结点；  
汇点 → 结点。

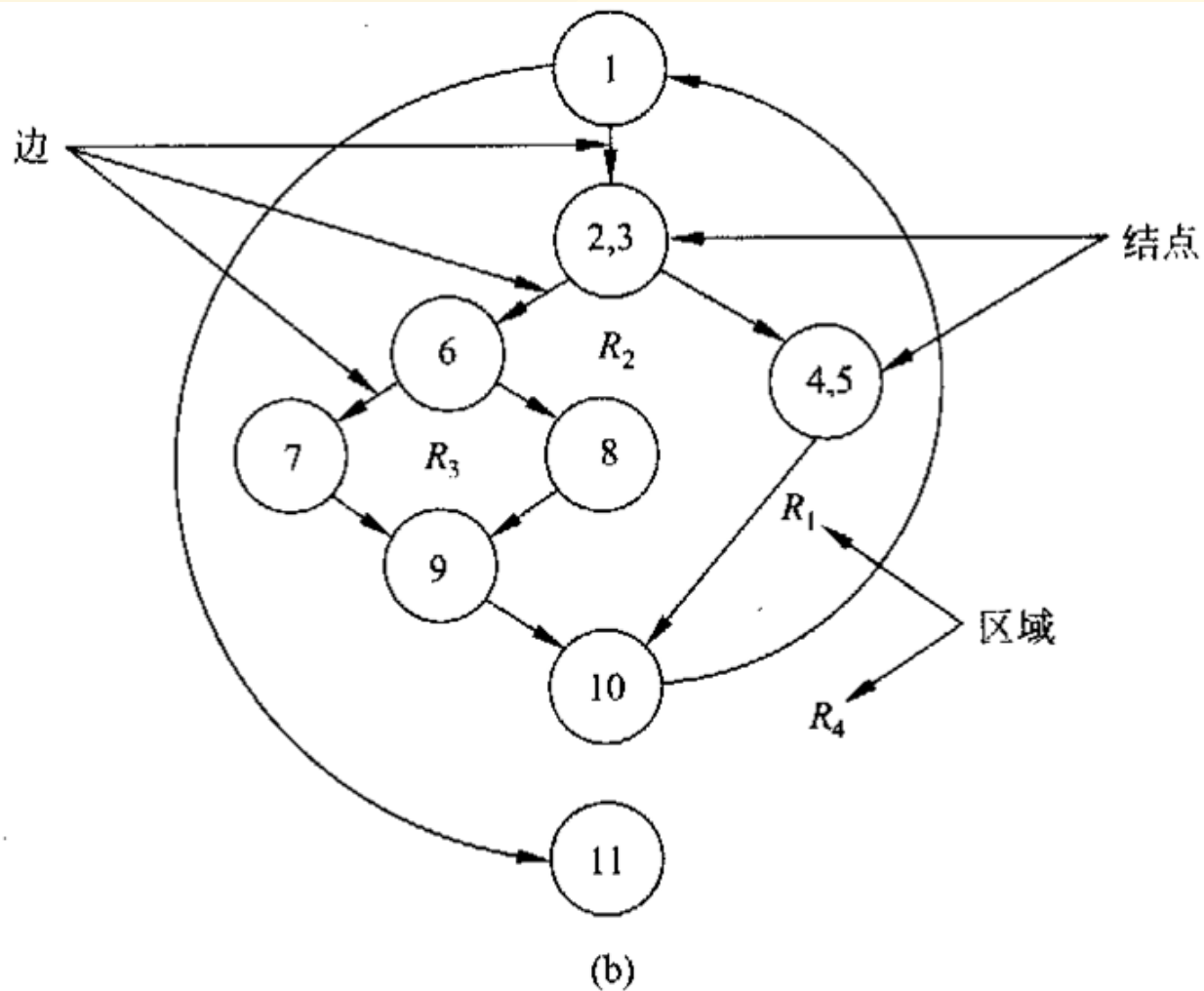


图 6.15 把程序流程图映射成流图

## 对复合条件的处理

- 复合条件包含了一个或多个布尔运算符（OR、AND、NOR等）。
- 应把复合条件分解为简单条件，每个条件对应一个结点。

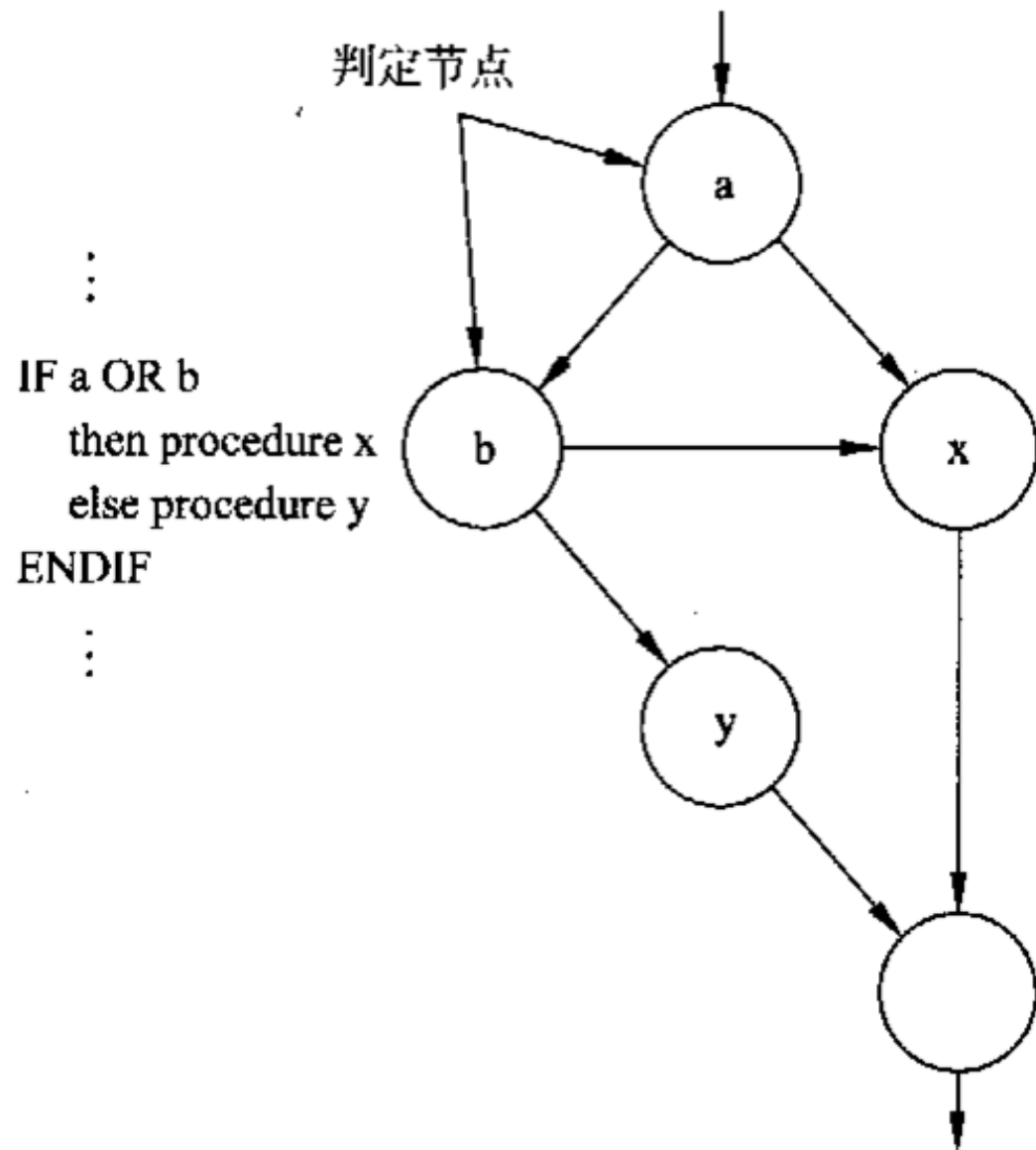


图 6.17 由包含复合条件的 PDL 映射成的流图

# 计算环形复杂度的方法

以下三种方法等价：

1. 环形复杂度  $V(G)$  等于流图中的区域数；
2. 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. 环形复杂度  $V(G) = P + 1$ ，其中P为流图中判定结点的数目。

图6.15的环形复杂度  $V(G) = 4$ 。

# 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  $V(G) \leq 10$  为宜。

# Halstead方法

根据程序中运算符和操作数的总数来度量程序复杂度。

$$N = N_1 + N_2$$

其中：

$N$  定义为程序长度；

$N_1$  为程序中运算符出现的总次数；

$N_2$  为操作数出现的总次数。

# 预测程序长度 $N$

Halstead给出预测程序长度的公式为：

$$H = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

其中：

$H$  定义为程序预测长度；

$n_1$  为程序中使用的不同运算符（包括关键字）的个数；

$n_2$  为程序中使用的不同操作数（变量和常量）的个数。

多次验证都表明，程序的预测长度 $H$ 和实际程序长度 $N$ 非常接近。

# 预测程序错误个数

Halstead还给出了预测程序中包含错误的个数的公式：

$$E = N \log_2 (n_1 + n_2) / 3000$$

# 详细设计说明书

- 着重描述每一模块是怎样实现的，包括实现算法、逻辑流程等。
- 详细设计说明书是详细设计的主要工作成果。



# 本章重点

- 结构化程序设计的概念
- 各种过程设计工具
- 程序复杂度的定量度量