

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 (自学)

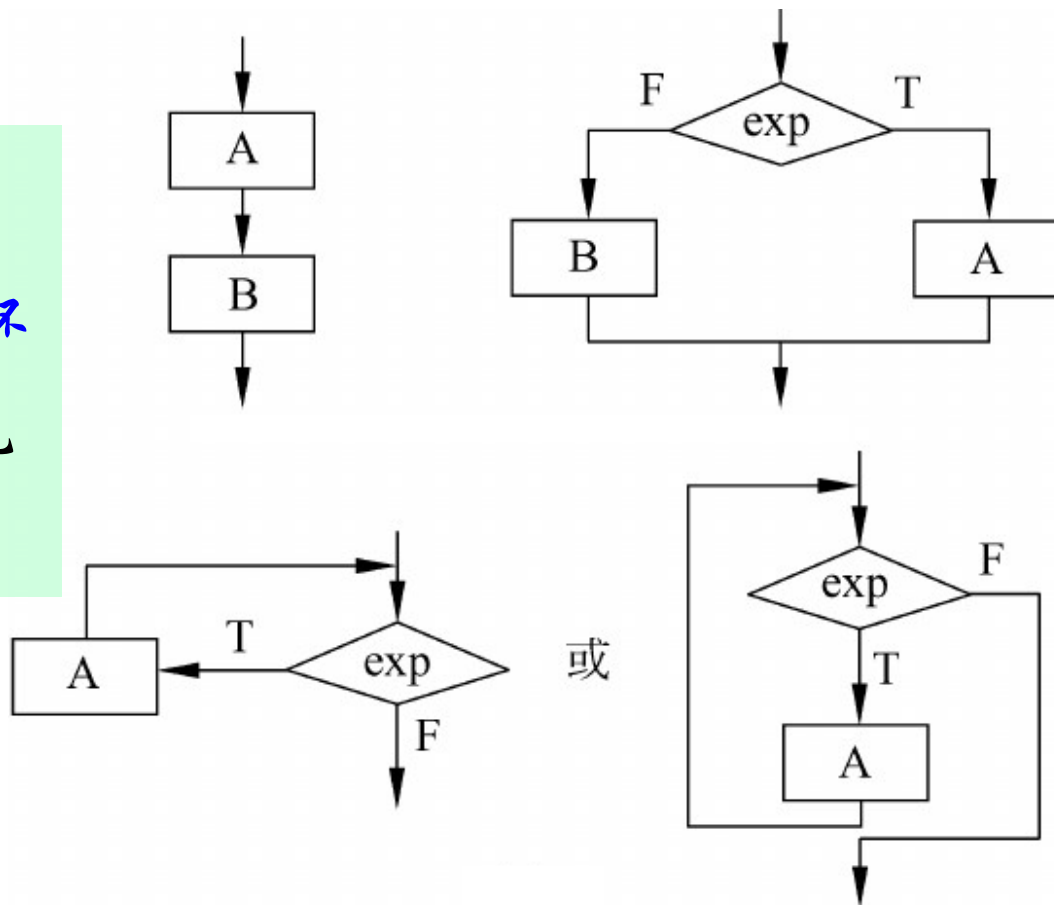
6.5 程序复杂度的定量度量 (自学)

- **根本任务**：确定应怎样具体实现所要求的系统。
为软件结构中的每个模块选择算法和块内数据结构，并用选定的某种表达工具给出清晰的描述。
- 详细设计结果基本上决定了最终程序代码的质量
- 详细设计的目标不仅仅是逻辑上正确地实现每个模块的功能，更重要的是设计出的**处理过程应该尽可能简明易懂**。

6.1 结构程序设计

□ 1965年最早由E.W.Dijkstra提出：可从高级语言中**取消goto语句**，程序的质量与所包含的goto语句数量成反比。

1966年，Bohm和Jacopini证明，只用**顺序**、**选择**和**循环**三种基本控制结构就能实现任何单入口单出口的程序。

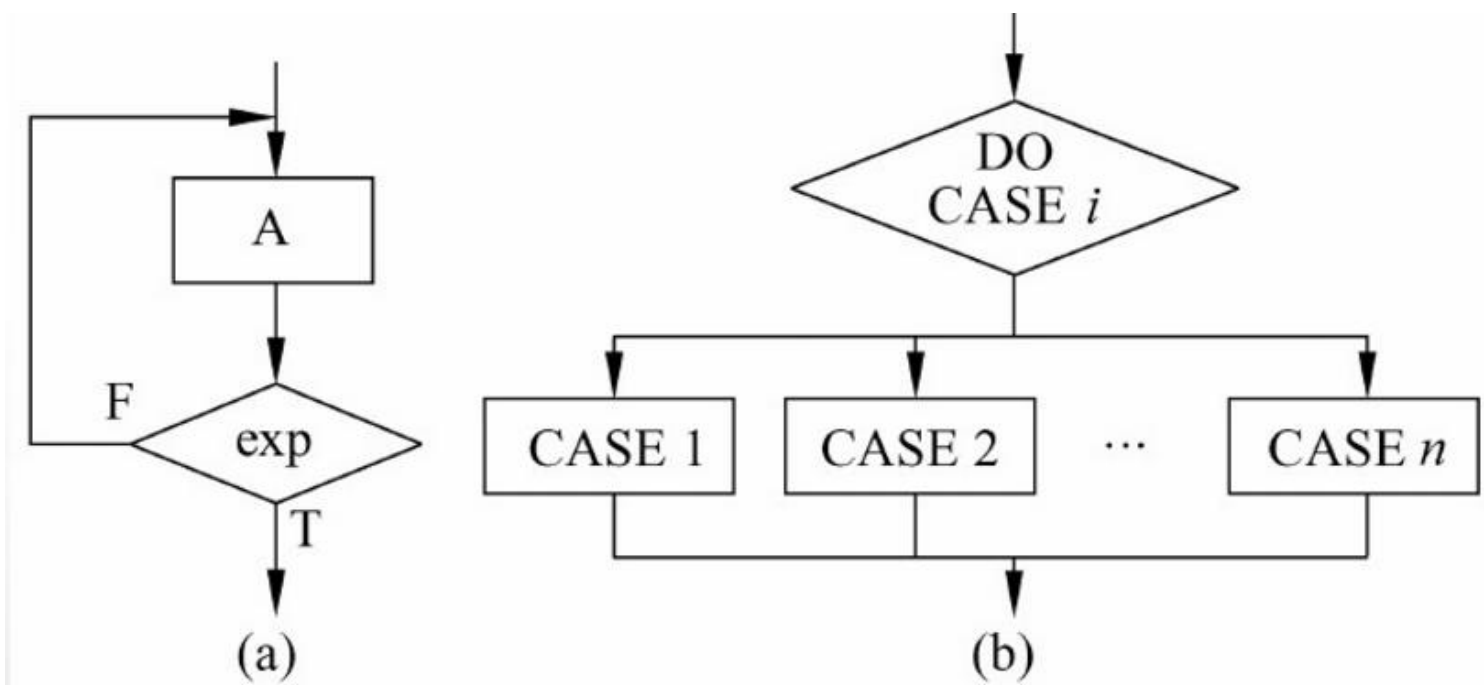


6.1 结构程序设计

- 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**。
- **结构程序的经典定义**：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连接，并且每个代码块只有一个入口和一个出口。
- **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。

6.1 结构程序设计

□ 理论上只用上述3种基本控制结构就可以实现任何单入口单出口的程序，但为了实际使用方便，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**：只允许使用顺序、IF_THEN_ELSE型分支和DO_WHILE型循环这三种基本控制结构。
- **扩展的结构程序设计**：除上述三种基本控制结构外，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**：再允许使用BREAK(LEAVE)结构。

6.2 人机界面设计

- 人机界面是人与计算机之间搭建的一个有效的交流媒介，主要解决用户如何来操作软件系统，是系统的外在表现形式。
- 软件的用户界面作为人机接口起着越来越重要的作用，它直接关系到用户对软件可接受程度，直接影响到软件的竞争力和寿命。

6.2 人机界面设计

一、设计问题

1、系统响应时间

- 从用户完成某个控制动作(如按回车键或单击鼠标), 到软件给出预期的响应(输出信息或做动作)之间的这段时间。
- 系统响应时间有两个重要属性。
 - **长度**: 时间过长, 用户就会感到紧张; 过短, 迫使用户加快操作节奏, 可能会犯错误。
 - **易变性**: 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。

6.2 人机界面设计

2、用户帮助设施

- 常见的帮助设施可分为**集成的**和**附加的**两类。
- 具体设计帮助设施时，必须解决下述的一系列问题

1) 用户与系统交互期间，是否任何时候都能获得关于系统功能的帮助信息？

- 提供部分功能的帮助信息和提供全部功能的帮助信息

2) 用户怎样请求帮助？

- 帮助菜单、特殊功能键和HELP命令

6.2 人机界面设计

3)怎样显示帮助信息?

- 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。

4)用户怎样返回到正常交互方式中?

- 屏幕上的返回按钮和功能键。

5)怎样组织帮助信息?

- 平面结构(关键字)、信息的层次结构(逐层展开)和超文本结构。

6.2 人机界面设计

3、出错信息处理

❑ 出错信息或警告信息是出现问题时交互式系统给出的“坏消息”。一般给出的出错或警告信息具有下述属性：

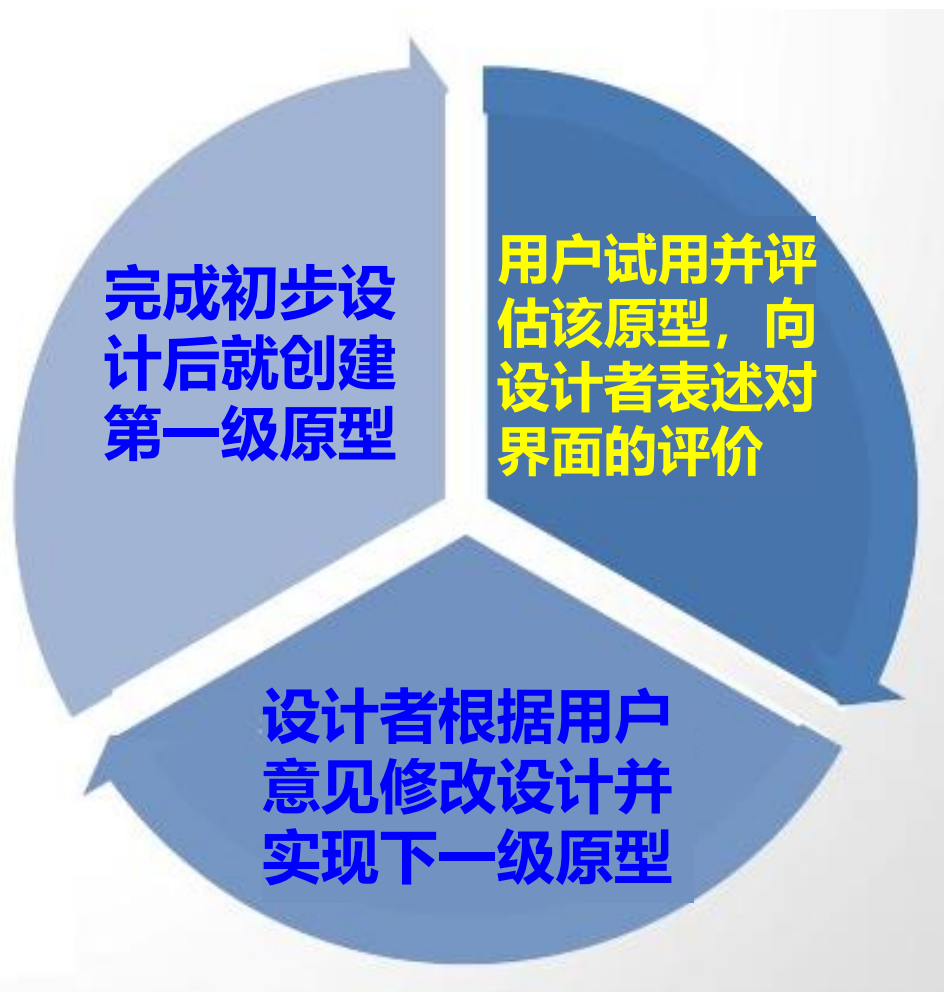
- 用用户可以理解的术语描述问题。
- 提供有助于从错误中恢复的建设性意见。
- 指出错误可能导致哪些负面后果（如破坏数据文件），以便用户检查是否出现了这些问题，并在确实出现问题时及时解决。
- 伴随听觉或视觉上的提示。
- 不能带有指责色彩，即不能责怪用户。



6.2 人机界面设计

二、设计过程

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



6.2 人机界面设计

三、人机界面设计指南

1、一般交互指南：

- 1) 保持一致性。如菜单选择、数据显示等使用一致的格式。
- 2) 提供有意义的反馈。
- 3) 在执行有较大破坏性的动作之前要求用户确认。
- 4) 允许取消绝大多数操作。如撤销命令。
- 5) 减少在两次操作之间必须记忆的信息量。

6.2 人机界面设计

- 6) 提高对话、移动和思考的效率。尽量减少用户按键的次数，设计屏幕布局时应考虑尽量减少鼠标移动的距离，避免用户出现“这是什么意思”的疑问
- 7) 允许犯错误。系统应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。

6.2 人机界面设计

2、信息显示指南：

- ① 只显示与当前工作内容相关的信息。
- ② 不要用数据淹没用户，使用便于用户迅速吸取信息的方式表现数据，如图形。
- ③ 使用一致的标记、标准的缩写和可预知的颜色。
- ④ 产生有意义的出错信息。
- ⑤ 使用大小写、缩进和文本分组以帮助理解。
- ⑥ 使用窗口分隔不同类型信息。
- ⑦ 使用“模拟”显示方式表示信息，如图形方式。

6.2 人机界面设计

3、数据输入指南

- ① 尽量减少用户的输入动作。如减少击键的数量，用鼠标选择菜单代替击键等。
- ② 交互应该是灵活的，可调整为用户喜欢的输入方式
- ③ 使在当前动作语境中不适用的命令不起作用
- ④ 让用户控制交互流（跳过、下一步、上一步、取消等）
- ⑤ 为输入动作提供必要的帮助
- ⑥ 消除冗余输入。如不需要用户输入数据的单位
- ⑦ 防止用户出错。对已输入的数据删除时必须进行确认；要对输入的数据进行有效性验证。

6.3 过程设计的工具

□ 详细设计工具分为**图形、表格和语言**三类。








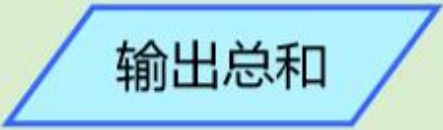

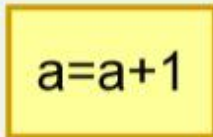
主要工具有：

- 程序流程图
- 盒图（N-S图）
- PAD图
- 判定表
- 判定树
- 过程设计语言（PDL）

一、程序流程图



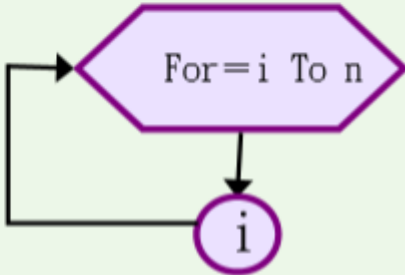

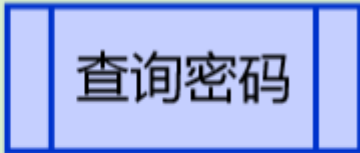


□ 程序流程图是使用较为广泛的描述过程设计的方法。

■ 程序流程图中经常使用的基本符号

	名 称	意 义	范 例
	开始 (Start) 终止 (End)	表示程序的开始或结束	 
	路径(Path)	表示流程进行的方向	 
	输入(Input) 输出(Output)	表示数据的输入或结果的输出	
	处理(Process)	表示执行或处理某一项工作	

一、程序流程图

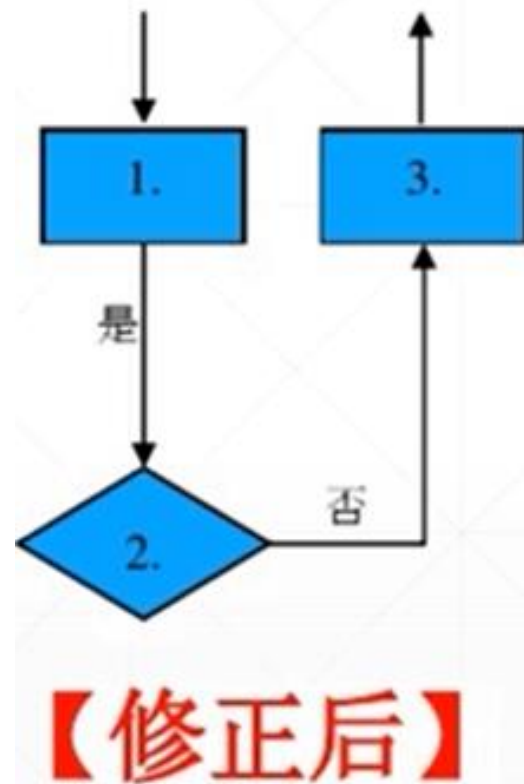
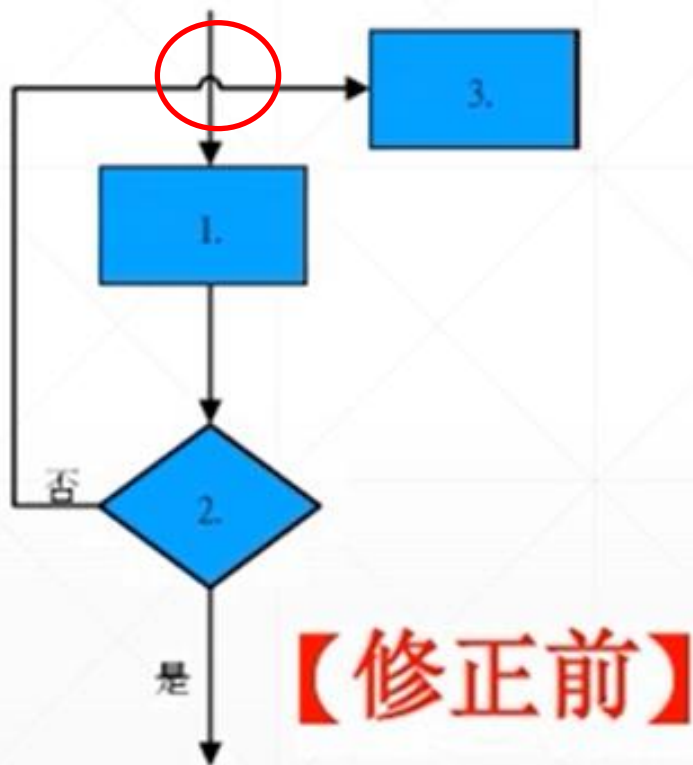
■ 程序流程图中经常使用的基本符号

	名 称	意 义	范 例
	决策判断 (Decision)	针对某一条件进行判断	
	循环 (Loop)	表示循环控制变量的初始值及终值	
	子程序 (Subroutine)	用以表示一群已经定义流程的组合	
	文件(Document)	指输入输出的文件	

一、程序流程图

■ 流程图绘制原则

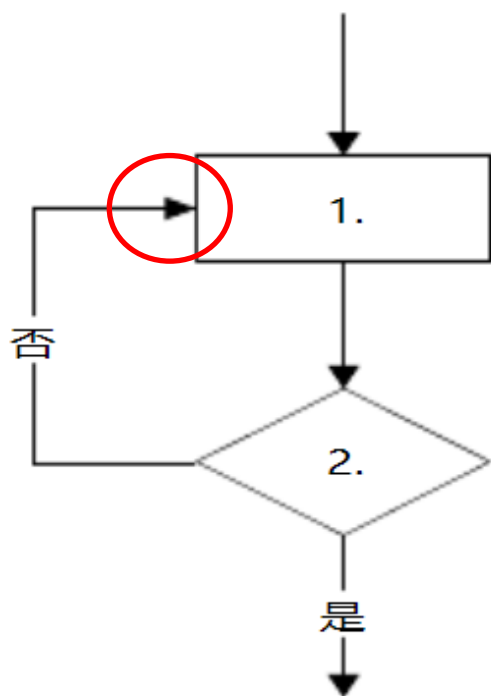
路径符号宜避免互相交叉



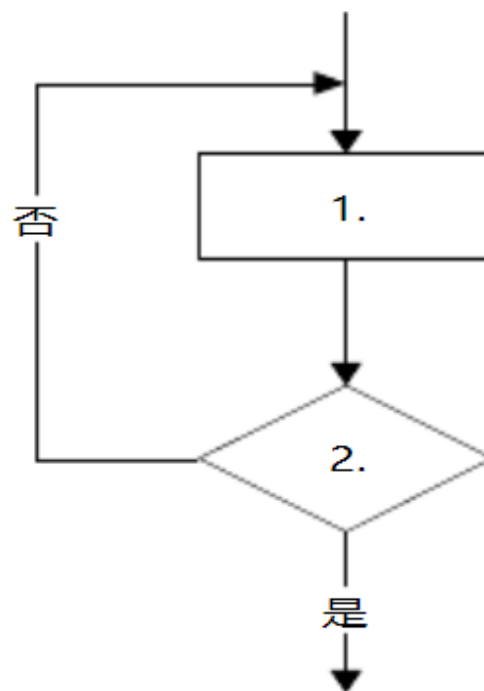
一、程序流程图

■ 流程图绘制原则

处理程序必须以单一入口与单一出口特性绘制！！！！



【修正前】

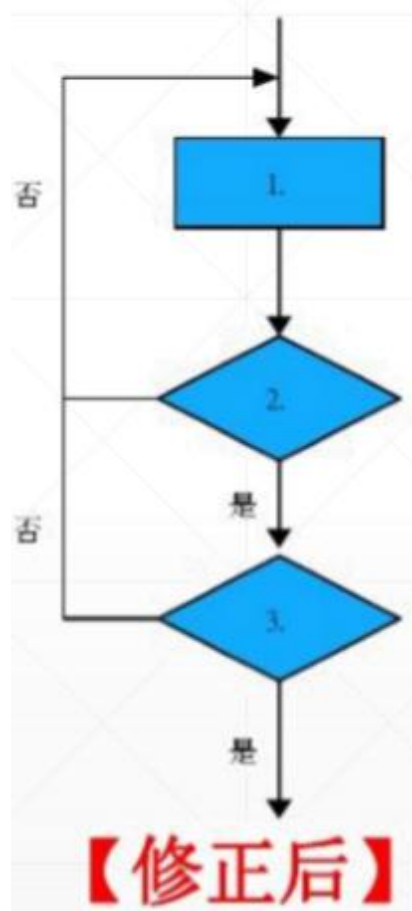
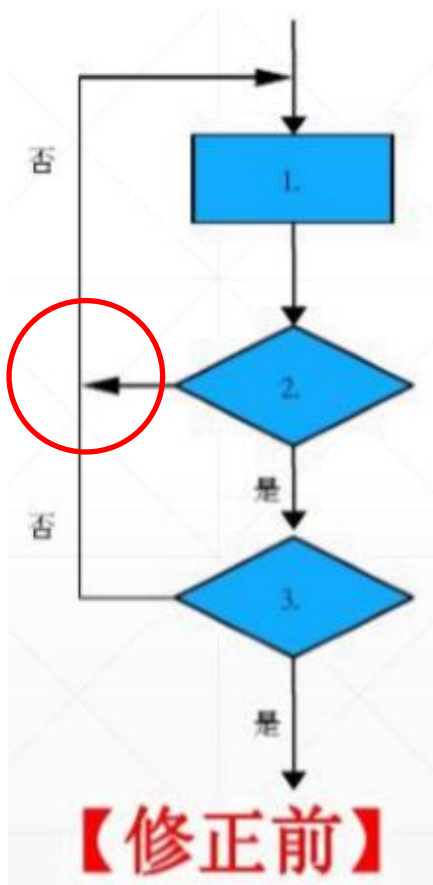


【修正后】

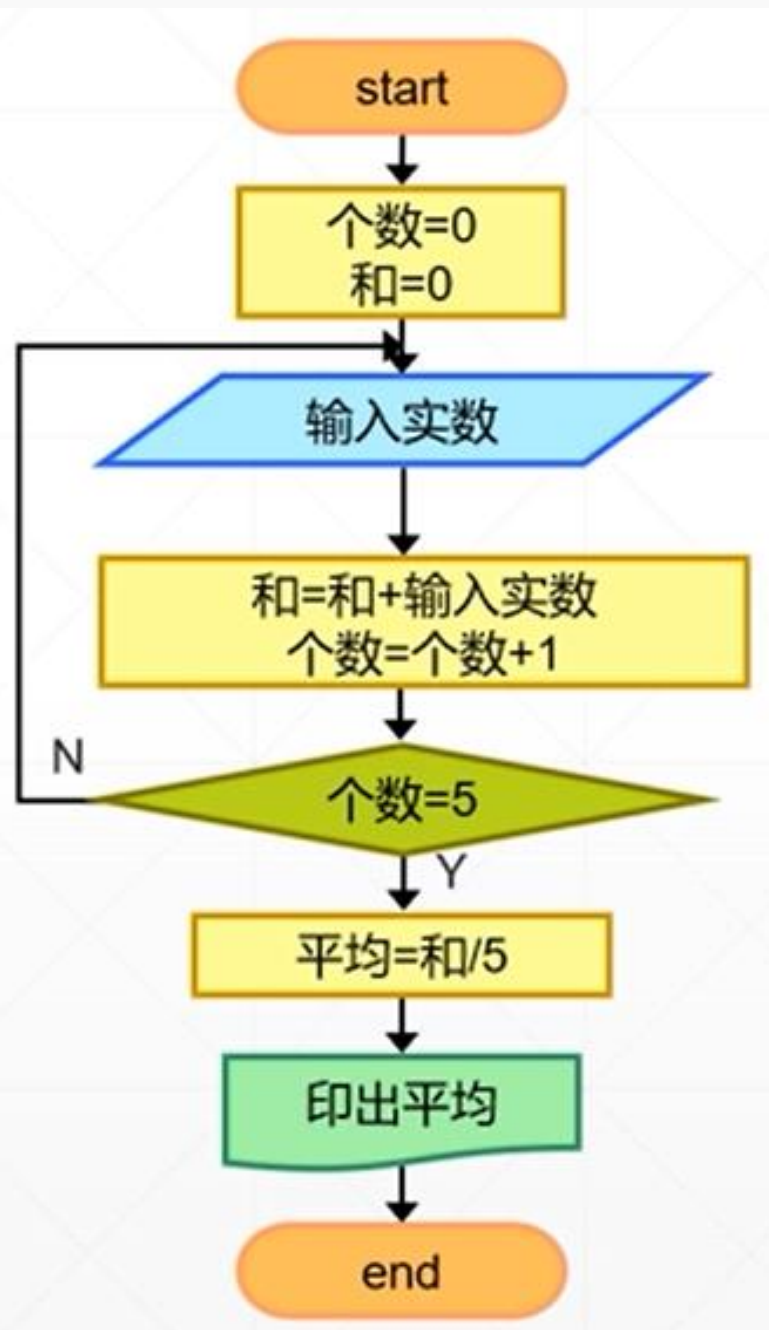
一、程序流程图

■ 流程图绘制原则

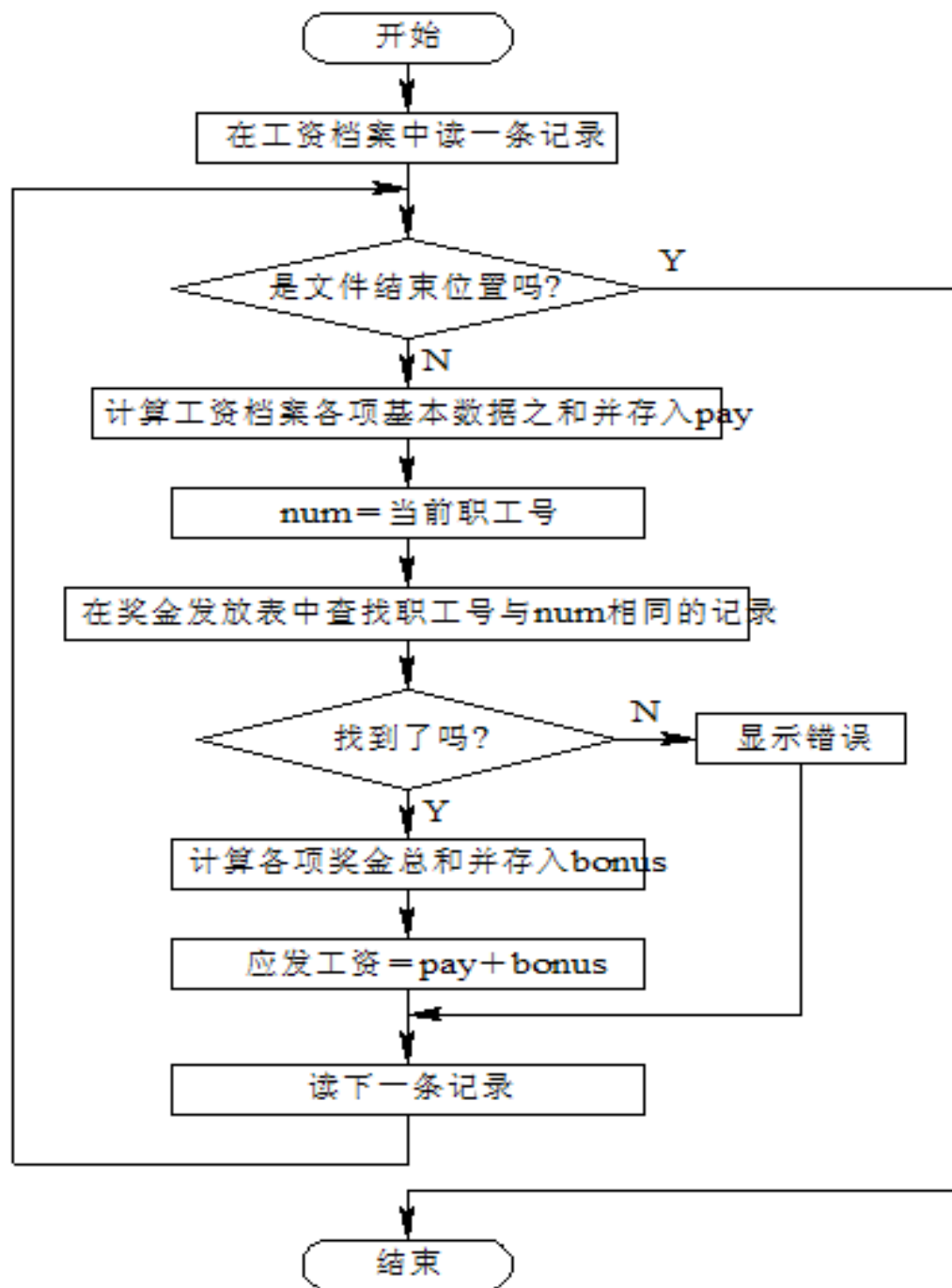
同一路径符号的指示箭头应只有一个!!!



**示例：输入5个实数，
计算平均值，然后再
打印该平均值**



采用程序流程图描述 计算应发工资模块



一、程序流程图

□ 优点:

- 描述直观、清晰，使用灵活，便于阅读和掌握。

□ 缺点:

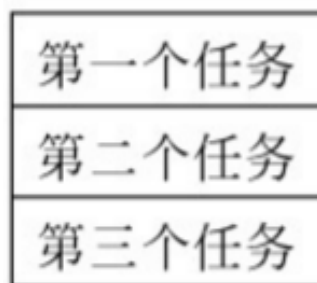
- (1) 程序流程图本质上不是逐步求精的好工具，使程序员过早地考虑程序的控制流程，而未考虑程序的全局结构。
- (2) 用箭头代表控制流，程序员不受任何约束，可以不顾结构程序设计的精神，随意转移控制。
- (3) 难以表示系统中的数据结构。

二、盒图

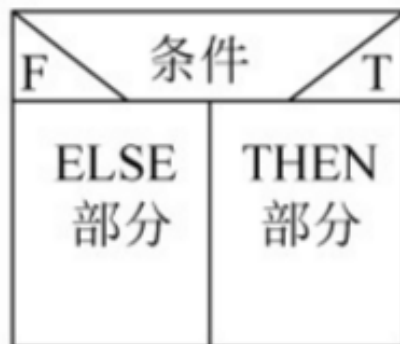
□ 出于需要不允许违背结构程序设计精神的图形工具的考虑，Nassi和Shneiderman提出了**盒图**，又称**N-S图**。它有下列特点：

- (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。

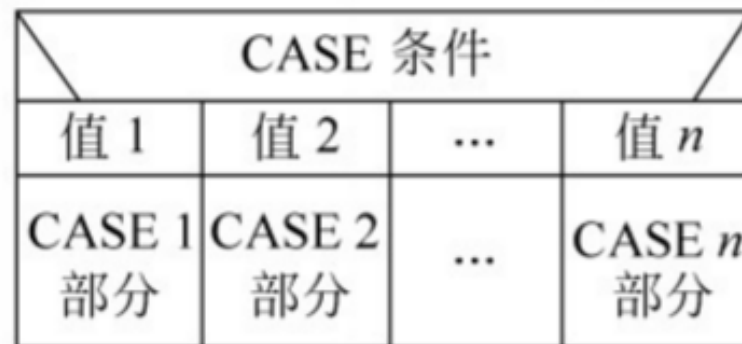
二、盒图



(a)



(b)



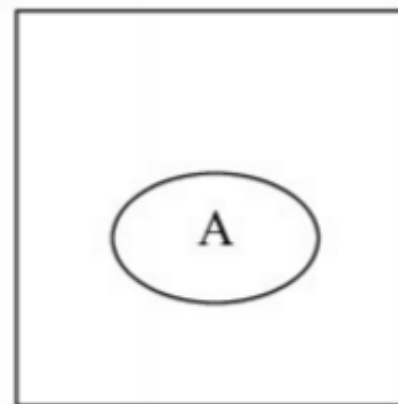
(c)



(d)

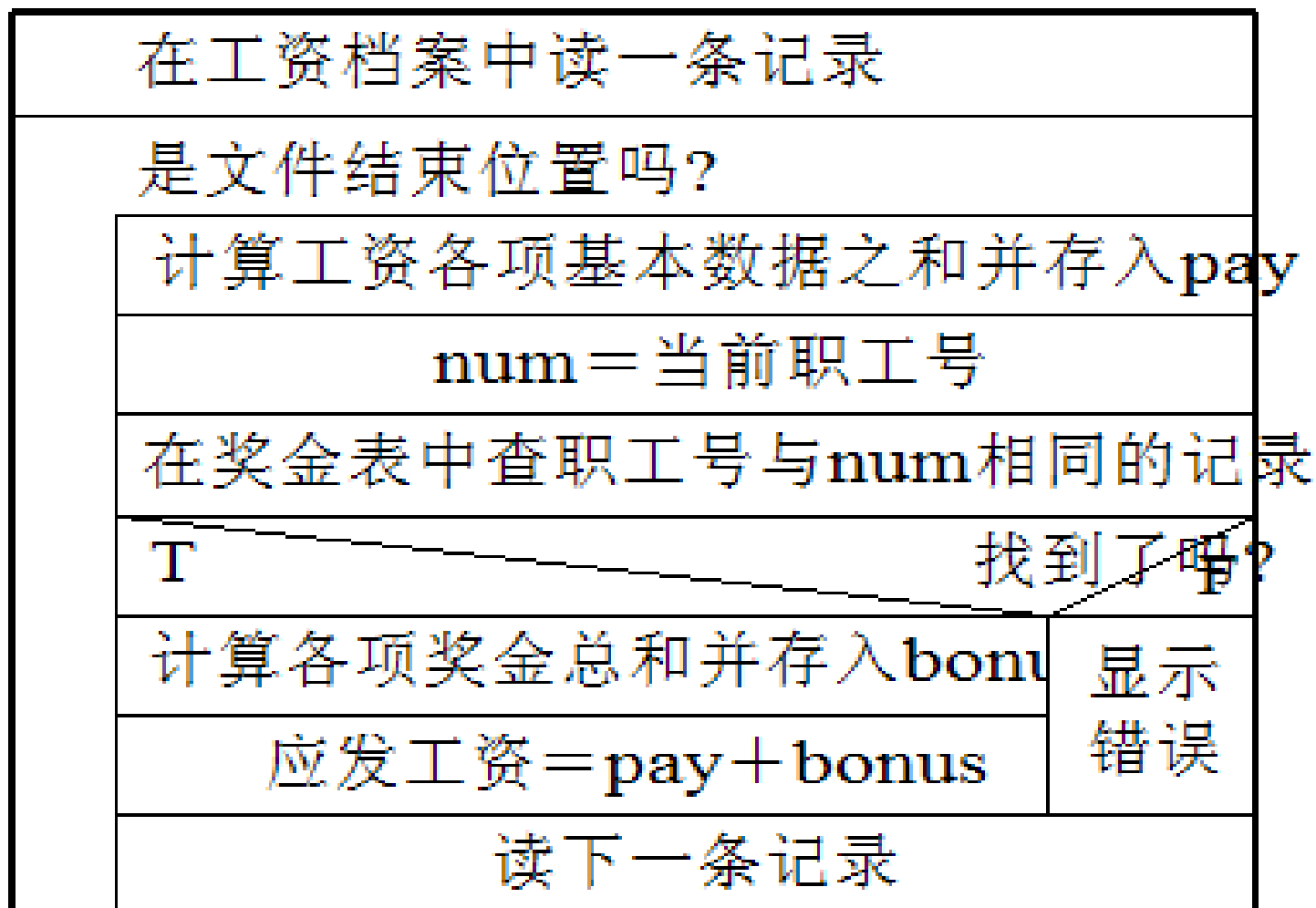


(e)



盒图的基本符号：(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

二、盒图



采用盒图描述计算应发工资模块

二、盒图

□ 优点:

- 所有的程序结构均使用矩形框表示，可以清晰地表达结构中的嵌套及模块的层次关系。
- 没有流程线，不可能随意转移控制，因而表达出的程序结构必然符合结构化程序设计的思想，有利于培养软件设计人员的良好设计风格。

□ 缺点:

- 当所描述的程序嵌套层次较多时，盒图的内层方框会越画越小，不仅影响可读性而且不易修改。

三、PAD图

PAD(Problem Analysis Diagram)是日立公司在20世纪70年代提出的又一种用于详细设计的图形表达工具。

■ 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

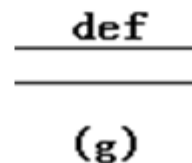
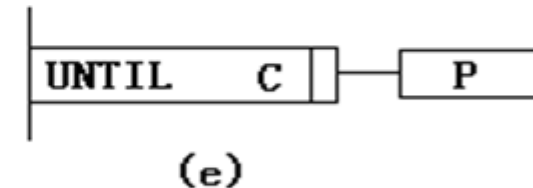
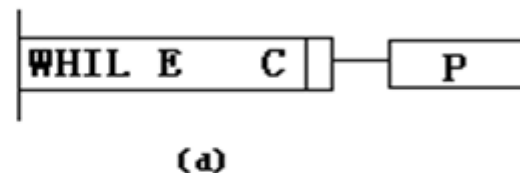
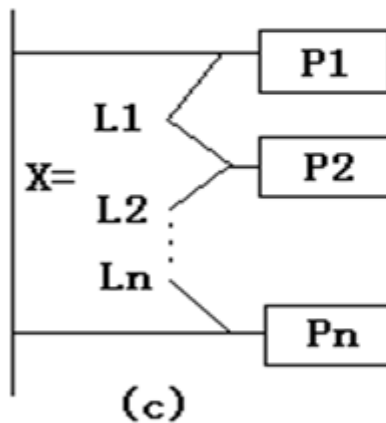
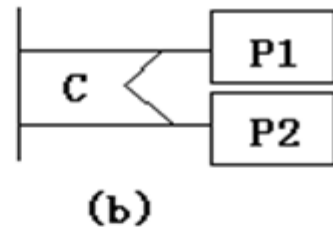
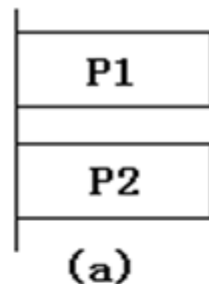
(WHILE C DO P);

(e) UNTIL型循环

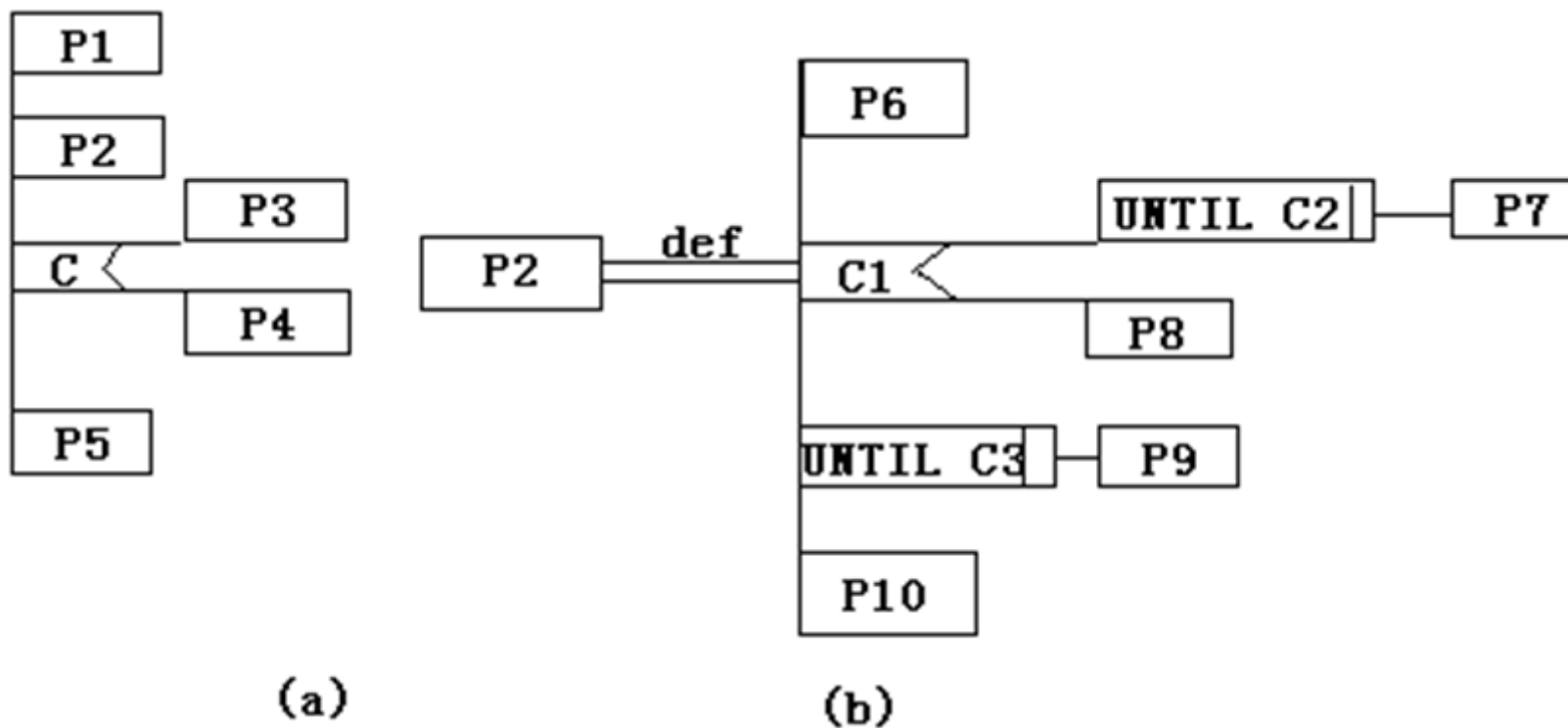
(REPEAT P UNTIL C)

(f)语句标号;

(g)定义

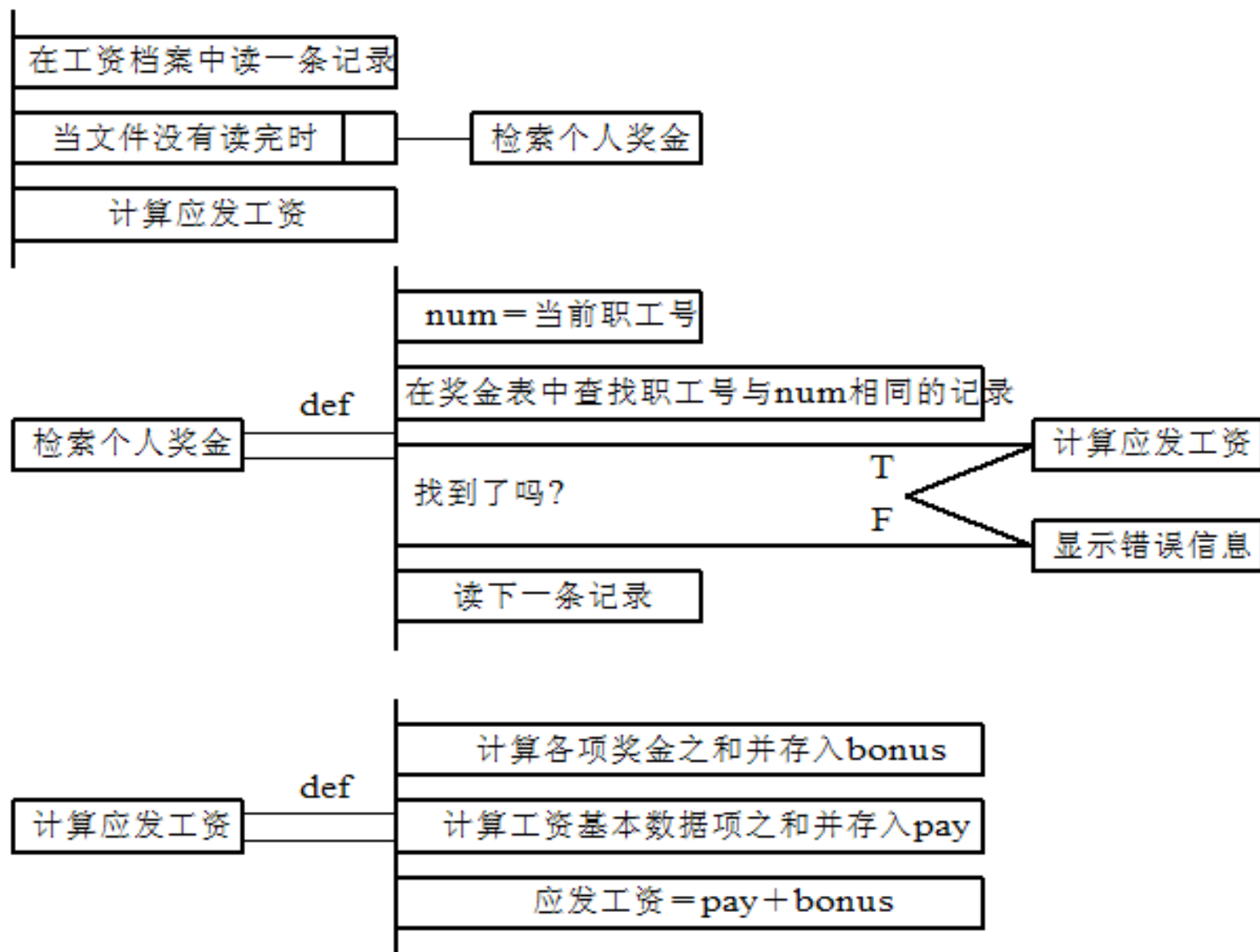


三、PAD图



(a) 初始的PAD图

(b) 使用def符号细化处理框P2



采用PAD图描述计算应发工资模块

三、PAD图

□ 优点:

- (1) 使用PAD图描述的程序结构层次清晰，逻辑结构关系直观、易读、易记、易修改。
- (2) PAD图为多种常用高级语言提供了相应的图形符号，易于PAD图向高级语言源程序转换。
- (3) 支持自顶向下、逐步求精的设计过程。

当算法中包含多重嵌套的条件选择时，用上述方法都不易清楚描述，这时可以选择判断表来表达复杂的条件组合与应做的动作之间的对应关系。

四、判定表

□ 判定表用于表示程序的静态逻辑，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，一般由四个部分构成：

- 左上部列出所有**条件**；
- 左下部列出所有可能产生的**动作**；
- 右上部列出所有可能的**条件组合**；
- 右下部列出每种**条件组合**相对应的**动作**。

条件列表	条件组合
动作列表	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。

四、判定表

例：航空行李托运费的算法

重量不超过30公斤的行李可免费托运。重量超过30公斤时，对超运部分，头等舱国内乘客收4元/公斤；其它舱位国内乘客收6元/公斤；外国乘客收费为国内乘客的2倍；残疾乘客收费为正常乘客的1/2。

□ 判定表的生成步骤：

- 提取问题中的所有条件和动作。
- 列出所有的条件组合。
- 填写每种条件组合下对应的动作。
- 若表中存在不同规则对应相同动作且其条件组合存在某种关系时，可进行必要的化简。

四、判定表

规则

条件

国内乘客
头等舱
残疾乘客
行李重量
 $W \leq 30\text{kg}$

1	2	3	4	5	6	7	8	9
	T	T	T	T	F	F	F	F
	T	F	T	F	T	F	T	F
	F	F	T	T	F	F	T	T
T	F	F	F	F	F	F	F	F

免费
(W-30)X2
(W-30)X3
(W-30)X4
(W-30)X6
(W-30)X8
(W-30)X12

X								
			X					
				X				
	X						X	
		X						X
					X			
						X		

动作

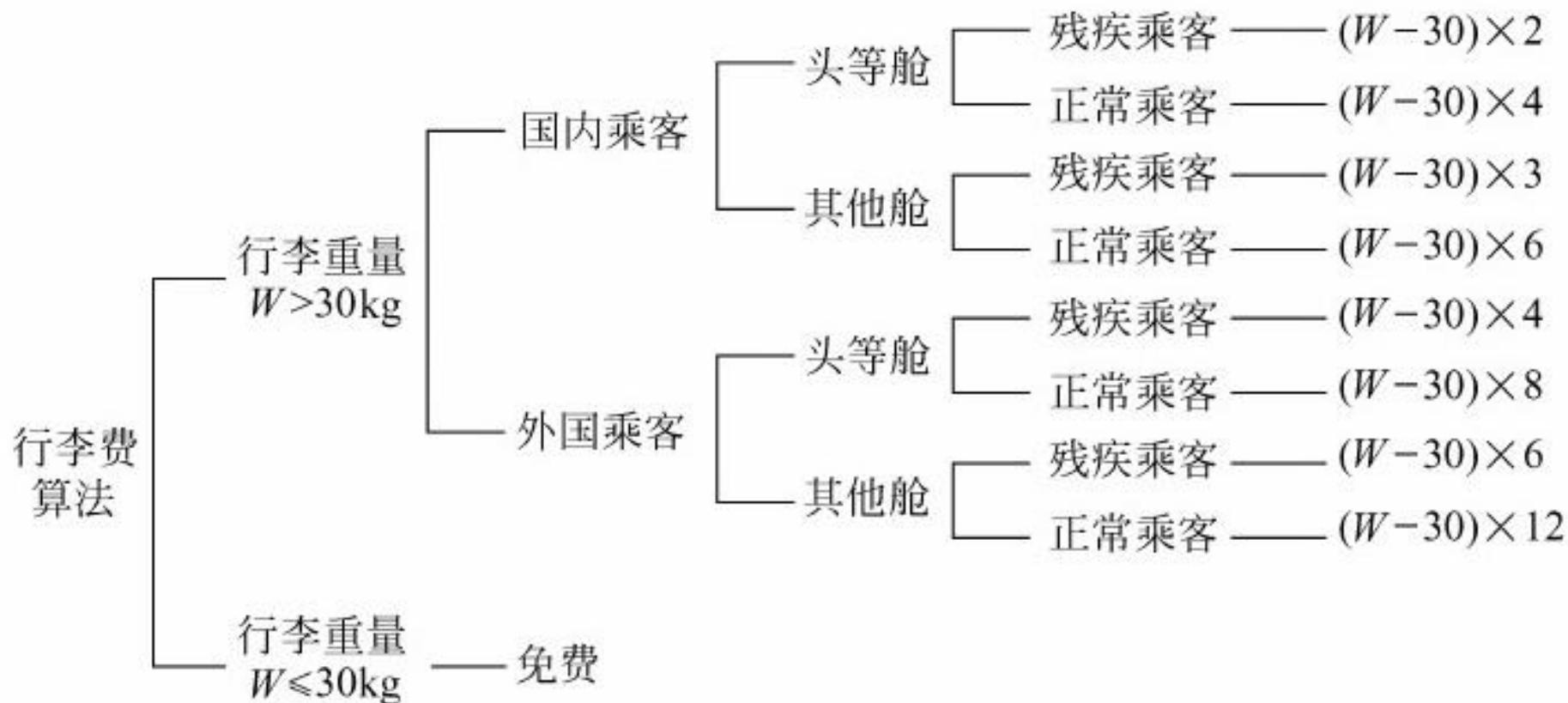
四、判定表

某单位工资档案管理系统中“职务津贴计算”判定表如下所示：

条件组合		1	2	3	4	5	6	7	8	9
条件	职务	助工	工程师	高工	助工	工程师	高工	助工	工程师	高工
	工龄	<10	<10	<10	10~20	10~20	10~20	>20	>20	>20
动作	奖金基数 350	√			√			√		
	奖金基数 400		√			√			√	
	奖金基数 500			√			√			√
	上浮 20%				√	√				
	上浮 30%						√	√		
	上浮 35%								√	
	上浮 40%									√

五、判定树

- 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。



五、判定树

- **优点：**判定树是判定表的图形表示，它与判定表的作用大致相同，但比判定表更加直观，更易于理解和掌握。
- **缺点：**逻辑上没有判定表严格，用户在使用判定树时容易造成个别条件的遗漏。

判定表与判定树并不适用于作为一种通用的设计工具，不能表示顺序结构、循环结构等复杂处理

六、PDL语言

PDL(Process Design Language)语言即过程设计语言，是一种用于描述程序算法和定义数据结构的伪代码。

□ 特点：

- **PDL语言是一种兼有自然语言和结构化程序设计语言语法的“混合型”语言。**
- **自然语言的采用使算法的描述灵活自由、清晰易懂。**
- **结构化程序设计语言的采用使控制结构的表达具有固定的形式且符合结构化设计的思想。**

六、PDL语言

PDL应用示例： 以××系统主控模块详细设计为例

PROCEDURE模块名()

清屏;

显示××系统用户界面;

PUT(“请输入用户口令: ”);

GET(password);

IF password < > 系统口令

提示警告信息;

退出运行

ENDIF

显示本系统主菜单;

六、PDL语言

WHILE(true)

接收用户选择Choice;

IF Choice=“退出”

Break;

ENDIF

调用相应下层模块完成用户选择功能;

ENDWHILE;

清屏;

RETURN

END

将下列伪代码转换成用NS图和PAD图表示。

```
S1;  
if (x>5) then S2  
else S3;  
while (y<0) S4;  
S5;  
if u>0 then  
    {  
        S6;  
        while (k>5) S7;  
    }  
S8;
```

作答

正常使用主观题需2.0以上版本雨课堂

课堂练习

