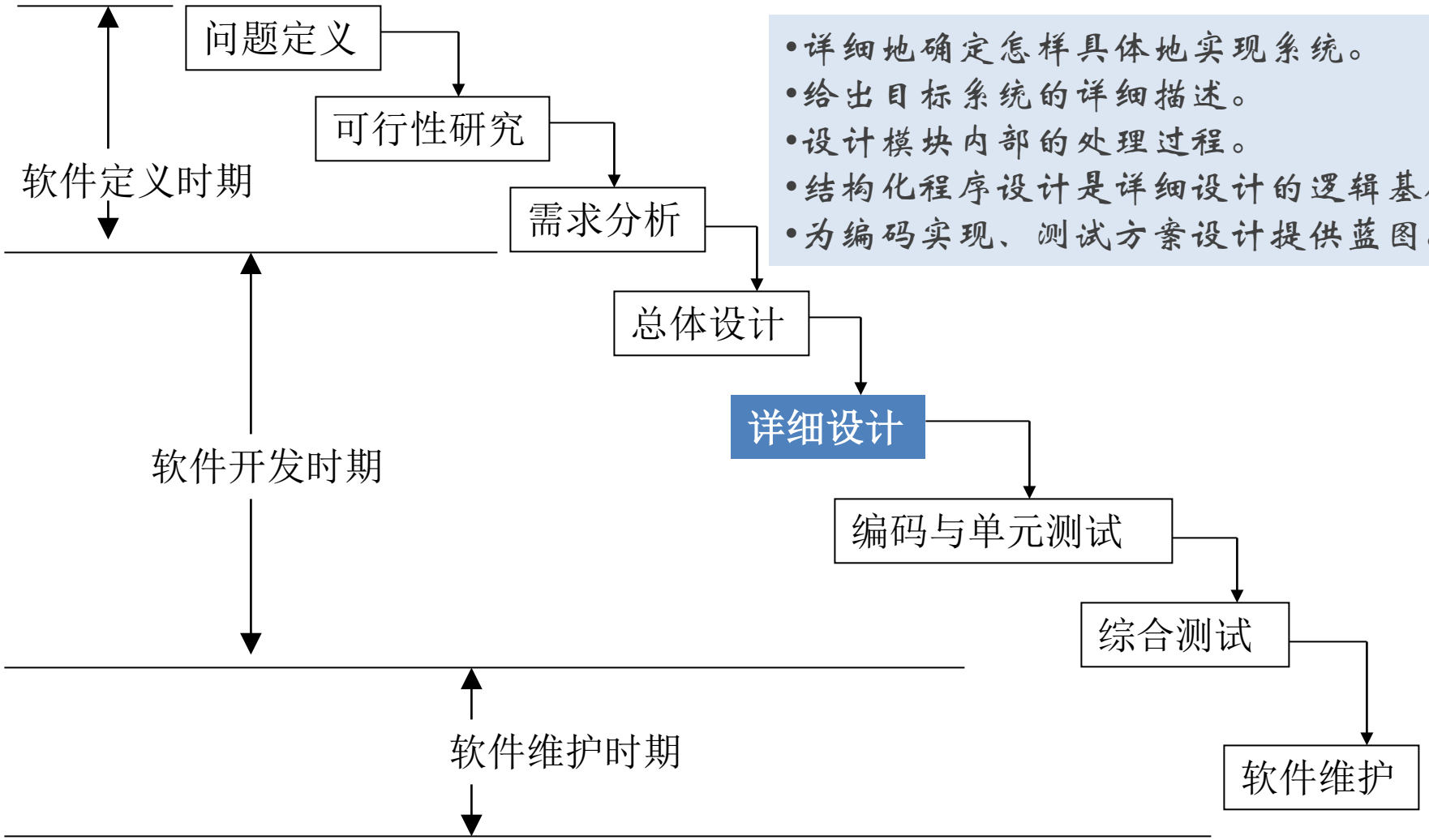
第六章 详细设计

How to do detailedly?

详细设计



- 详细地确定怎样具体地实现系统。
- 给出目标系统的详细描述。
- 设计模块内部的处理过程。
- 结构化程序设计是详细设计的逻辑基础。
- 为编码实现、测试方案设计提供蓝图。

第6章 详细设计

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

第6章 详细设计

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

6.1 结构程序设计

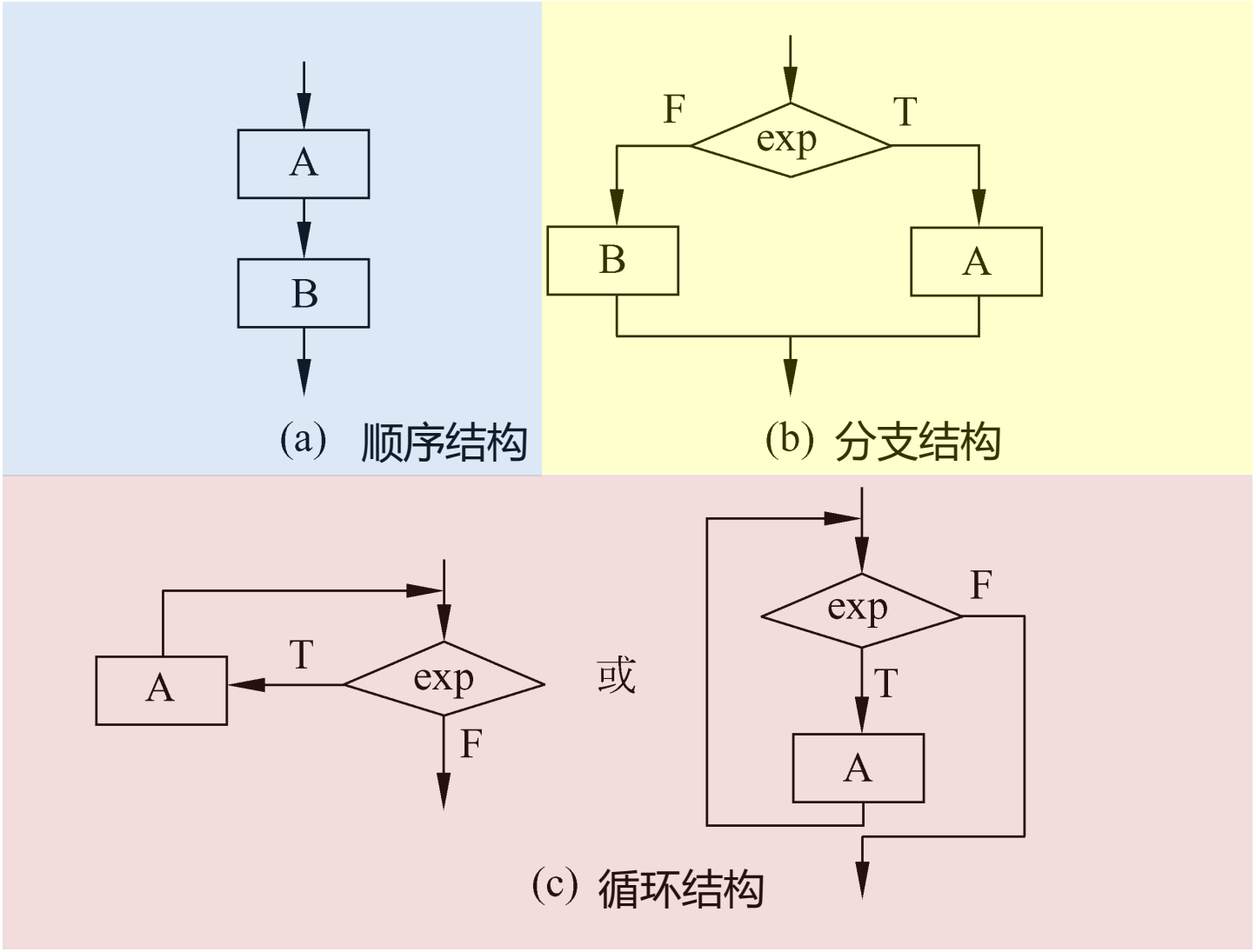
- 结构化程序设计
  - “结构程序设计是尽可能少用GO TO语句的程序设计方法。最好仅在检测出错误时才使用GO TO语句，而且应该总是使用前向GO TO语句”，“程序的质量与程序中所包含的GO TO 语句的数量成反比” 。

——E.W.Dijkstra,1965

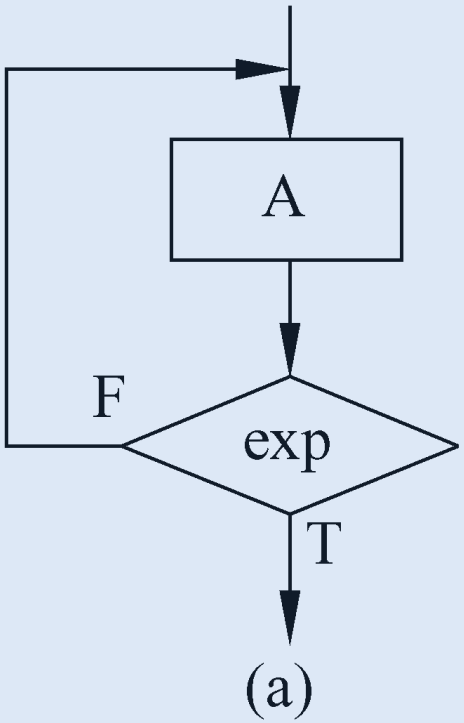
- 基本的程序控制结构
  - 3种基本的控制结构(顺序,选择,循环)就能实现任何单入口单出口的程序。

——Bohm and Jacopini,1966

6.1 结构程序设计——三种基本的控制结构

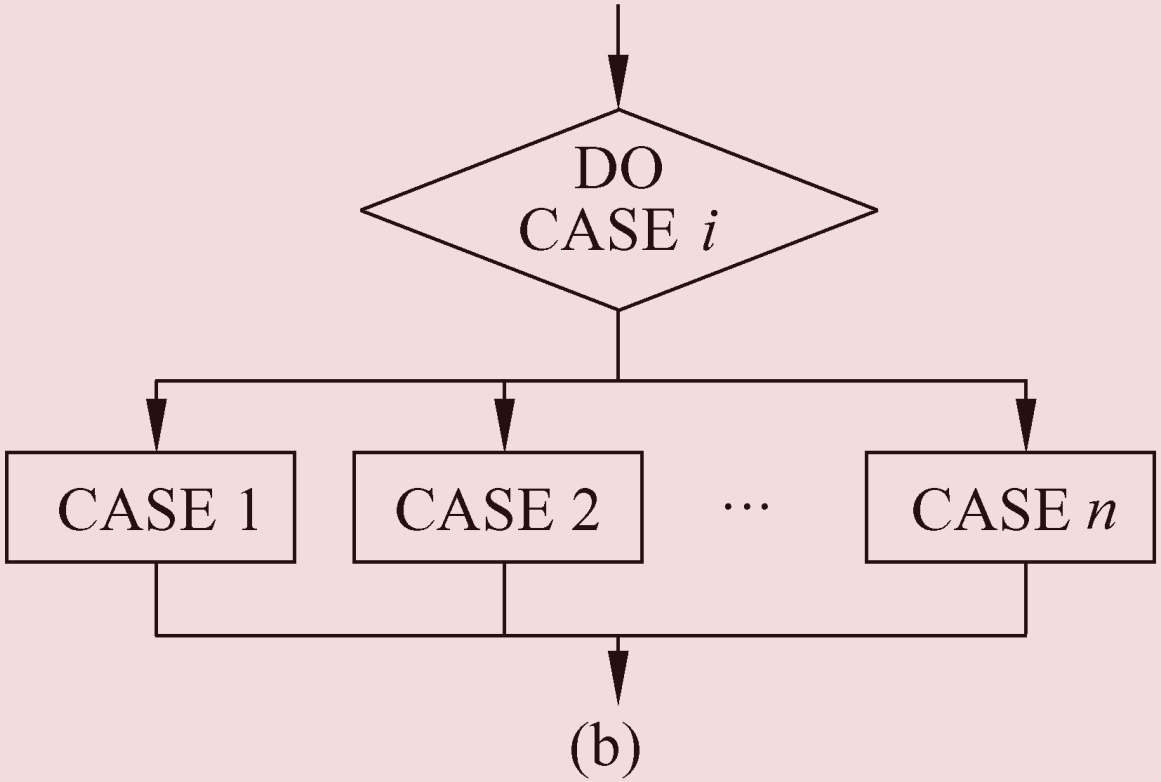


6.1 结构程序设计——其他常用的控制结构



(a)

后置循环结构



(b)

多项选择结构

6.1 结构程序设计——结构程序设计指导准则

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：

- 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：

- 除上述结构以外，还允许使用LEAVE(或BREAK)结构。

使用GO TO语句的原则:

- 结构程序设计是不使用（或尽可能少用）GO TO语句的程序设计方法。仅在检测出错误时才使用GO TO语句，而且总是使用后向GO TO语句。

第6章 详细设计

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

6.2.2 设计过程

6.2.3 人机界面设计指南

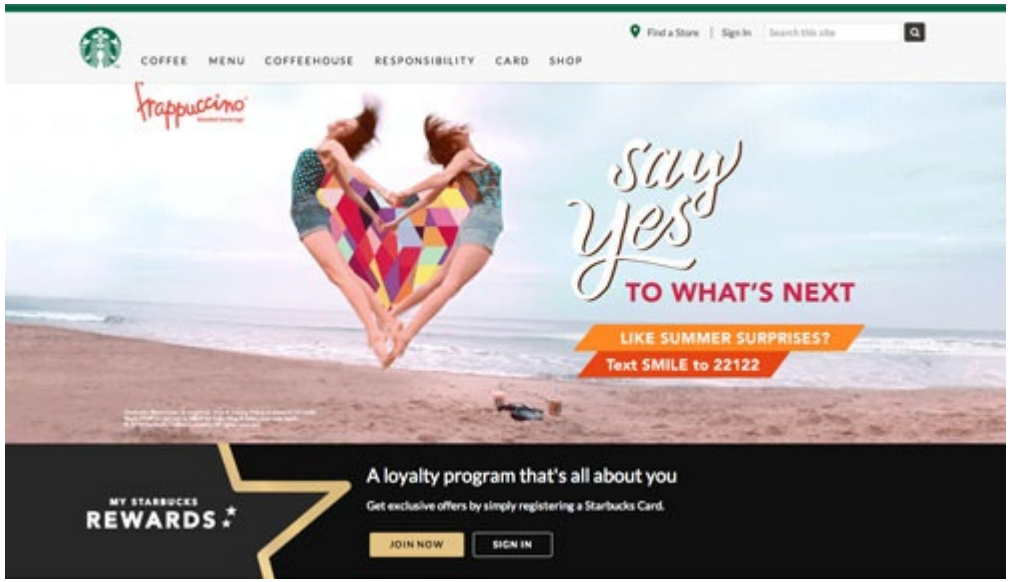
6.2.1 设计问题

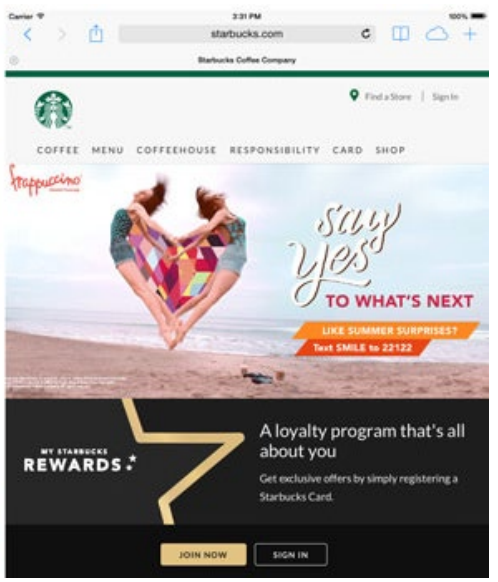
1. 系统响应时间
2. 用户帮助设施
3. 出错信息（及警告信息）处理
4. 命令交互

6.2.1 设计问题

1. 系统响应时间

- 响应时间的长度：适中
- 相应时间的易变性：同一类操作的响应时间不要差异太大。

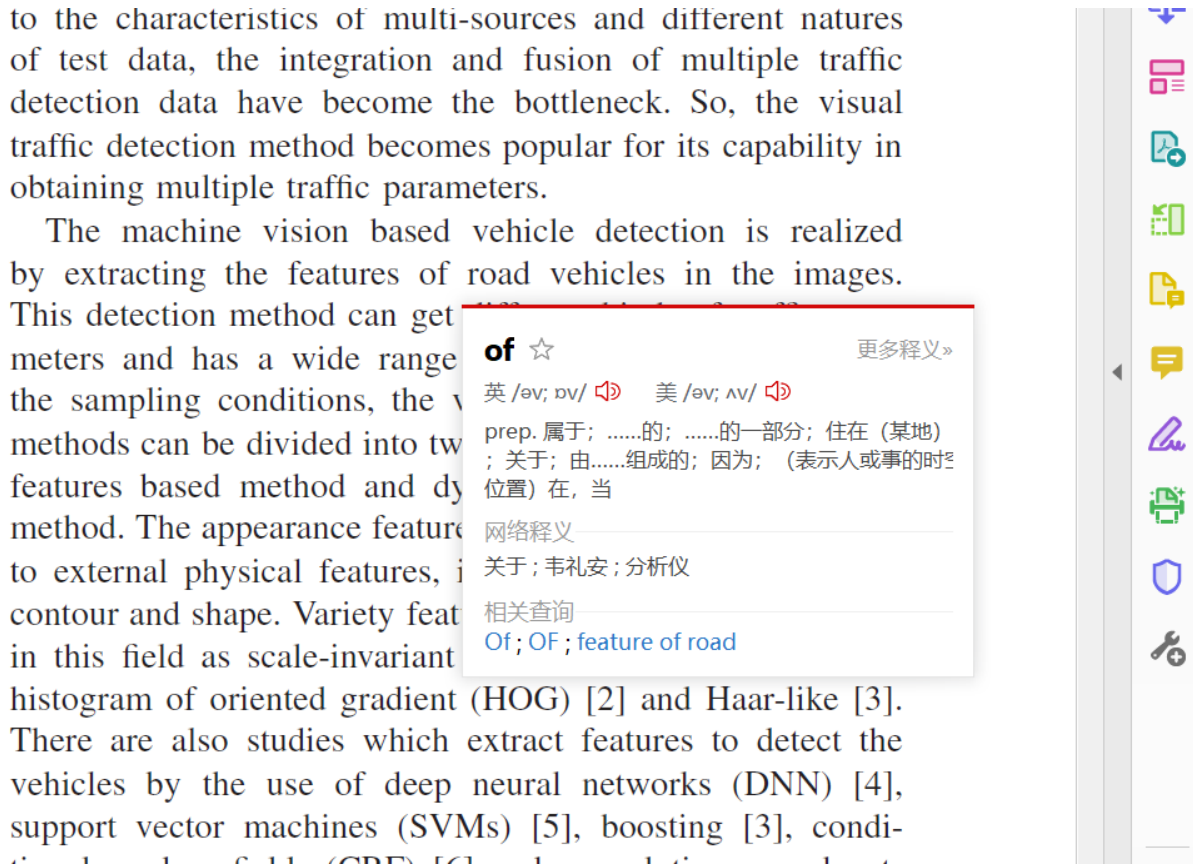




3G网络下，加载33个外部JavaScript文件, 6个CSS文件

6.2.1 设计问题

1. 系统响应时间
2. 用户帮助设施
   - 脱机帮助手册
   - 附加的联机帮助
   - 集成的联机帮助
   - 上下文相关的联机帮助



6.2.1 设计问题

1. 系统响应时间
2. 用户帮助设施
3. 出错信息（及警告信息）处理
   - 如何办, 用户心理

6.2.1 设计问题

1. 系统响应时间
2. 用户帮助设施
3. 出错信息（及警告信息）处理
4. 命令交互
   - 多媒体(图,文,声,光)
   - 自定义宏指令
   - 控制序列（CTRL+字母 ）

2000年开始，人机交互方式急剧繁荣

6.2.1 设计问题

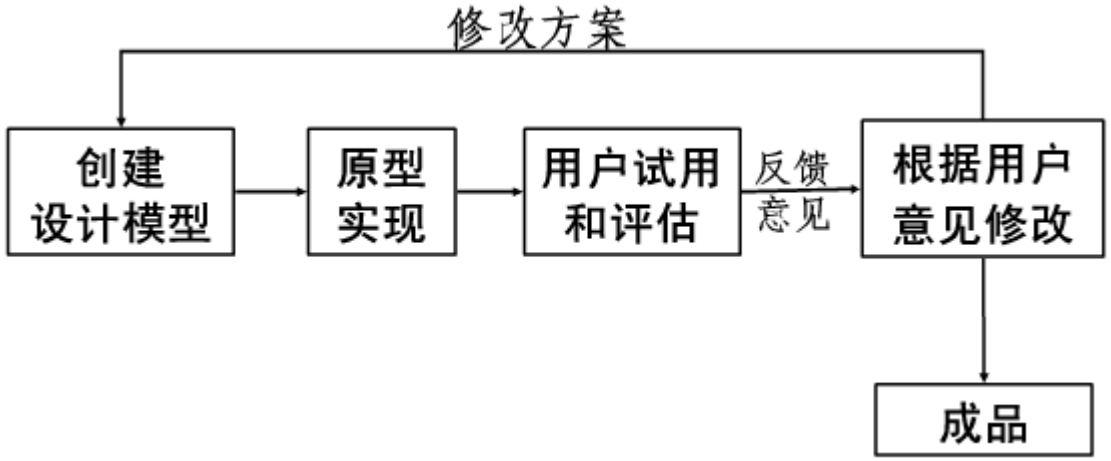
1. 系统响应时间
2. 用户帮助设施
3. 出错信息（及警告信息）处理
4. 命令交互

设备和人眼的距离	0.01-0.03米	0.3米	0.6米	3米	和人员距离关系不大
设备	眼镜设备	智能手表、手机	笔记本、平板、PC	智能电视、游戏终端	语言驱动的智能设备
交互的挑战	如何更方便的输入？	如何有效地在小型屏幕上显示关键信息？	屏幕多点触控、众多交互方式中如何提高效率？	专用设备怎么提供更细粒度的交互？体感交互？	如何提高语言的识别率？如何理解语义？

6.2.2 设计过程

迭代的设计过程

- 创建设计模型
- 原型实现
- 评估复审
- 进一步精化



评估复审的准则

- 系统规格说明书的长度和复杂程度—系统工作量
- 动作、命令中的包含的平均参数个数及操作个数—交互时间，效率
- 设计模型中包含的动作、命令、状态的数量—用户学习时间
- 界面风格、帮助设施、出错处理的友好性—用户接受程度

用户的第一印象

软件设计者要给用户什么样的第一印象？

所有功能都出现在用户面前？

可以考虑的两点：

1. 如何尽快让目标用户找到响应就的功能入口？
2. 怎么样让用户在第一次使用的时候，少花时间在对用户没有价值的部分（软件配置、填写用户属性等），而把大部分时间花在实际功能上？

用户的第一印象


电源
设置
自定义
待机
音量
TV/AV
静音
资讯
股票
声道
短信
游戏
音量+
喜爱
互动切换
音量-
指南
频道+
回看
确认
菜单
退出
频道-
上页
显示
下页
快退
播放/暂停
快进
点播
时移
1
2ABC
3DEF
4GHI
5JKL
6MNO
7PQRS
8TUV
9WXYZ
电视
0
广播
湖北广电网络


电源
GA870WJSA


OK
音量
节目
静音
显示
回看

用户的第一印象


OK
音量
节目
USB
静音
1.@
2abc
3def
4ghi
5jkl
6mno
7pqrs
8tuv
9wxyz
显示
0
回看


音量
节目

6.2.3 人机界面设计指南

1. 界面的基本形式
2. 界面设计的任务
3. 界面的类别
4. 界面应具有的基本特性
5. 界面设计指南

6.2.3 人机界面设计指南

1. 界面的基本形式
2. 界面设计的任务
3. 界面的类别
4. 界面应具有的基本特性
5. 界面设计指南

用户界面的基本形式举例

菜单：程序中功能的选择、数据的选择

图象：将数据可视化地展示

图表：显示统计数据、对比的数据

仪表盘：动态变化的数据、仿真的数据

对话框：初始数据、任意数据的输入

窗口：综合界面，可包含数据输入、信息显示、功能选择操作等。

用户界面的基本形式举例

菜单、图象、图表、仪表盘、对话框、窗口

设备和人眼的距离	0.01-0.03米	0.3米	0.6米	3米	和人员距离关系不大
设备	眼镜设备	智能手表、手机	笔记本、平板、PC	智能电视、游戏终端	语言驱动的智能设备

6.2.3 人机界面设计指南

1. 界面的基本形式
2. 界面设计的任务
3. 界面的类别
4. 界面应具有的基本特性
5. 界面设计指南

用户界面设计的任务

1、用户特性分析 — 用户模型

- 用户类型：外行型、初学型、熟练型、专家型。
- 用户特性度量：与用户使用模式和用户群体能力有关。包括：用户使用频度、用户用机能力、用户的知识、思维能力等。

2、用户界面的任务分析 — 任务模型

- 是对系统内部活动的分解，不仅要进行功能分解（用DFD图描述），还要包括与人相关的活动（人工操作）。每个加工即一个功能或任务。

3、确定用户界面类型

- 从用户角度出发，用户界面设计的类型主要有问题描述语言，数据表格、图形与图标、菜单、对话框及窗口等。每一种类型都有不同的特点和性能。

6.2.3 人机界面设计指南

1. 界面的基本形式
2. 界面设计的任务
3. 界面的类别
4. 界面应具有的基本特性
5. 界面设计指南

用户界面的类别

1. 一般交互界面
   - “一般交互”包括信息显示、数据输入、系统整体控制。这一部分指南具有全局性意义，对系统界面好坏影响极大。
2. 信息显示界面
   - 指显示信息时要注意的问题。显示的信息应该是完整的、无二义的、好理解的，这样才能满足用户要求。
   - 信息“显示”的不同方式：文字/图形/声音；不同位置/移动；不同大小；不同颜色/分辨率；等等。
3. 数据输入界面
   - 数据输入界面，是系统的重要组成部分。主要从输入效率和减少出错率考虑。
   - 用户一般会花费大量的时间在命令操作和数据输入操作，因此友好的数据输入操作非常必要。
   - 不同输入手段（介质/设备）：键盘、鼠标、数字化仪、扫描仪、语音、照片、视频等等。

6.2.3 人机界面设计指南

1. 界面的基本形式
2. 界面设计的任务
3. 界面的类别
4. 界面应具有的基本特性
5. 界面设计指南

一般交互界面设计指南

- 保持界面格式的一致性
- 提供有意义的反馈信息
- 重要操作要确认：如新增0次确认、修改1次确认、删除2次确认
- 为大部分操作提供“回退”(Undo)
- 减少两次操作之间的记忆的信息量（自动补充缺省值）
- 提高操作效率：思考、对话、按键、光标移动等效率
- 容许错误操作，出错后不会对系统造成严重影响
- 操作功能/动作分类，屏幕合理布局
- 提供操作的帮助设施
- 简单易懂的命令名

案例：如何把无意识引入到交互设计中

信息显示界面设计指南

- 只显示当前工作相关的信息
- 以合适的形式显示信息：图形？图表？清单？
- 以统一的风格显示信息：统一的标记、标准的缩写、预定的颜色、确定的位置等
- 提供可视化的显示环境
- 产生有意义的出错信息
- 使用格式化的显示板式：大小写、缩进、文本分段等
- 使用窗口分隔不同类型信息
- 使用“模拟”显示方式表示信息：数字仪表盘技术
- 合理高效地使用物理显示屏

案例：手机屏幕尺寸扩展是如何影响用户体验设计的

信息显示界面设计指南

人民币流通和反假币举报机制投诉电话和投诉电子邮箱：
0xLJB3F6C4C9D3EBCFD6BDF0B9DCC0EDzFJPJKFBCNYXJGL@mail.notes.bank-of-china.com

0xLJ = 链接
B3F6 = 出
C4C9 = 纳
D3EB = 与
CFD6 = 现
BDF0 = 金
B9DC = 管
C0ED = 理
z = 小写
FJPJKFB =反假票据客服部的拼音缩写
CNYXJGL = 出纳与现金管理的拼音缩写
@mail.notes.bank-of-china.com

减少用户在使用软件时出现错误

- 利用某种方式提醒用户可能出错
- 如何设置出错信息
  - 不要只告诉用户操作无法完成或者操作失败
  - 不要仅仅给出出错代码，还应当给出该错误的含义
  - 不要在出错信息中使用用户无法理解的术语
  - 错误要尽可能明确
  - 错误信息要有建设性，要让用户看出怎样才是正确的
  - 不要给出误导性的出错信息
  - 向用户提出解决问题的建议

减少用户在使用软件时出现错误

- 利用某种方式提醒用户可能出错
- 如何设置出错信息

:30 答辩地点：6教北

名	学生姓名	学号	班级	专业
	王逸飞	17072224	17184111	软件工程
	邓金旭	17051112	17052311	计算机科学
	张卓群	17051608	17052316	计算机科学
	姜亚定	17051316	17052313	计算机科学
	魏文毓	17051703	17052313	计算机科学
	方进福	17051911	17052312	计算机科学
	田虹虹	17120352	17052317	计算机科学
	肖雪	16051106	16052311	计算机科学
	刘强华	17196119	17052318	计算机科学
	钟维真	17051237	17052312	计算机科学
	童奕超	17051927	17184112	计算机科学

答辩地点：6教北

学生姓名	学号	班级	专业
王逸飞	17072224	17184111	软件工程
邓金旭	17051112	17052311	计算机科学
张卓群			计算机科学
姜亚定			计算机科学
魏文毓			计算机科学
方进福			计算机科学
田虹虹			计算机科学
肖雪			计算机科学
刘强华			计算机科学
钟维真	17051237	17052312	计算机科学
童奕超	17051927	17184112	计算机科学

以文本形式存储的数字
转换为数字(C)
有关此错误的帮助
忽略错误
在编辑栏中编辑(F)
错误检查选项(O)...

数据输入界面设计指南

- 尽量减少用户的输入动作，使按键最少，如对相同内容输入设置默认值、自动填入、列表选择或点击选择等。
- 保持信息显示和数据输入一致性：视觉效果一致
- 容许用户自定义输入：为专家级用户提供的机制
- 提供灵活的交互方式：键盘、鼠标等，适应不同用户
- 休眠(disable)当前动作语境中不用的命令
- 让用户控制交互流：灵活的操作顺序和恢复机制
- 对所有输入动作都提供帮助
- 消除冗余的输入

案例：网页注册登录界面设计

第6章 详细设计

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

6.3.2 盒图（N-S图）

6.3.3 问题分析图（PAD图）

6.3.4 判定表

6.3.5 判定树

6.3.6 过程设计语言（PDL）

6.3 过程设计的工具

6.3.1 程序流程图

6.3.2 盒图（N-S图）

6.3.3 问题分析图（PAD图）

6.3.4 判定表

6.3.5 判定树

6.3.6 过程设计语言（PDL）

6.3.1 程序流程图

程序流程图的作用

- 程序流程图作为一种算法表达工具, 是人们对解决问题的方法、思路或算法的一种描述。(国家标准GB1525-89, 国际标准ISO5807—85)

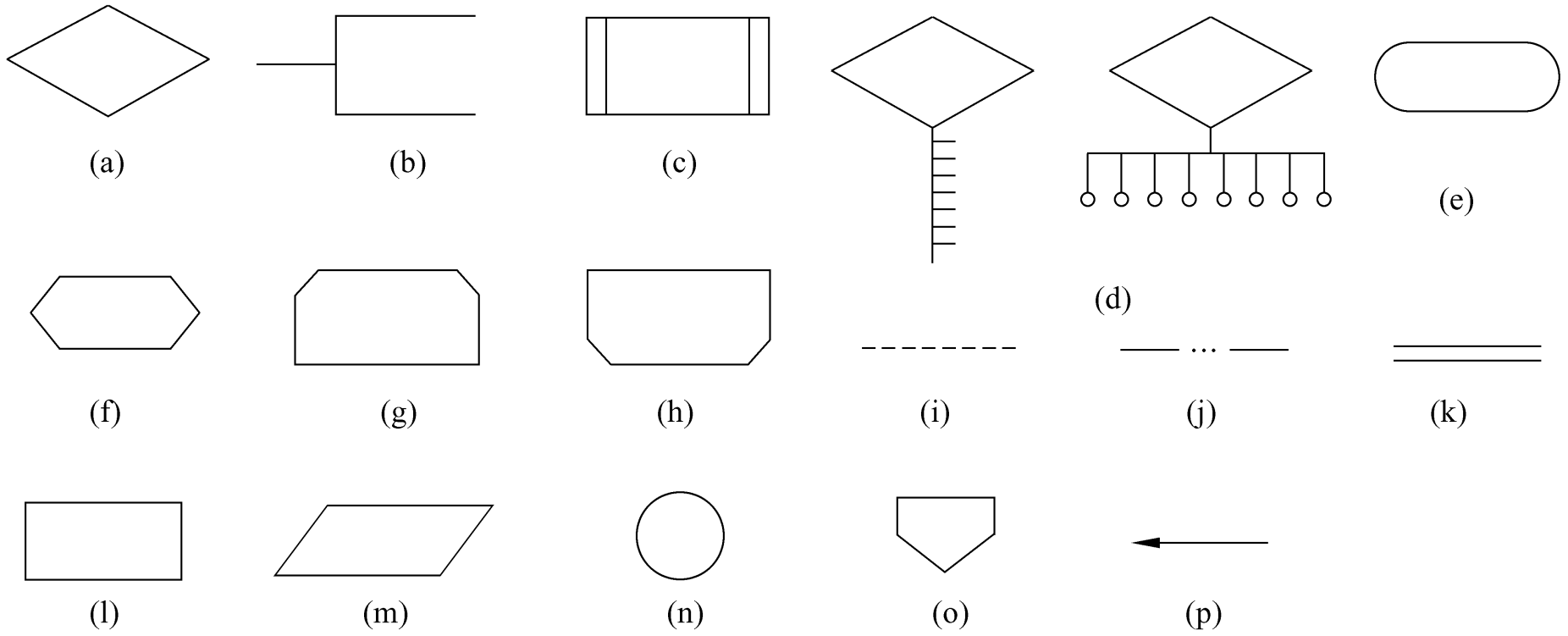
优点

- 采用简单规范的符号，画法简单；
- 结构清晰，逻辑性强；
- 便于描述，容易理解。

缺点

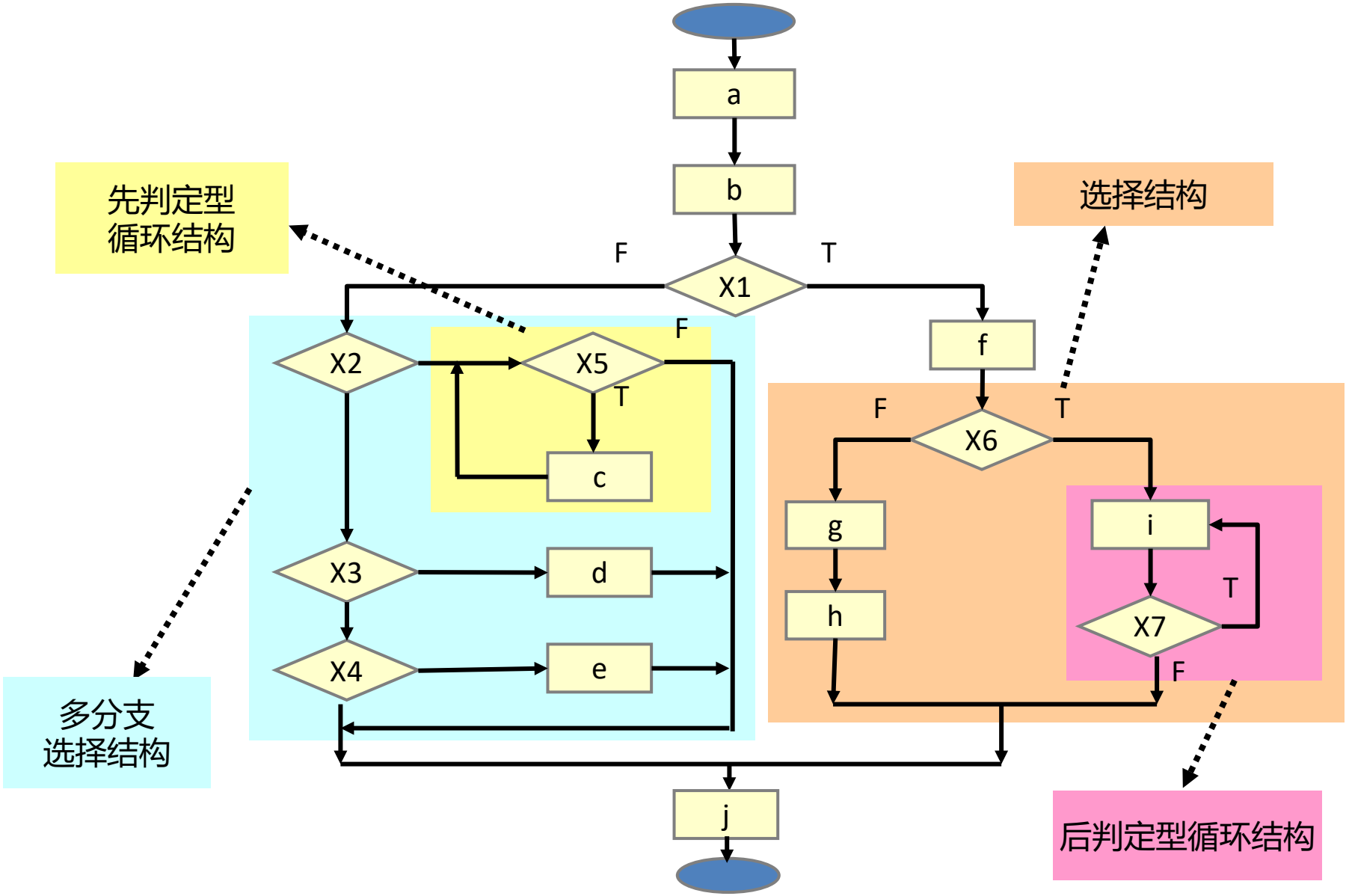
- 本质上不是“自顶向下、逐步求精”的设计工具（容易引导过早关注细节）
- 不是结构化设计工具（流向线可以随意连）

6.3.1 程序流程图——符号



程序流程图中使用的符号

6.3.1 程序流程图——图例


a
b
X1
F
T
X2
X5
T
c
F
X3
d
X4
e
f
X6
F
T
g
h
i
T
X7
F
j
先判定型
循环结构
选择结构
多分支
选择结构
后判定型循环结构

6.3 过程设计的工具

6.3.1 程序流程图

6.3.2 盒图（N-S图）

6.3.3 问题分析图（PAD图）

6.3.4 判定表

6.3.5 判定树

6.3.6 过程设计语言（PDL）

6.3.3 问题分析图（PAD图）

PAD

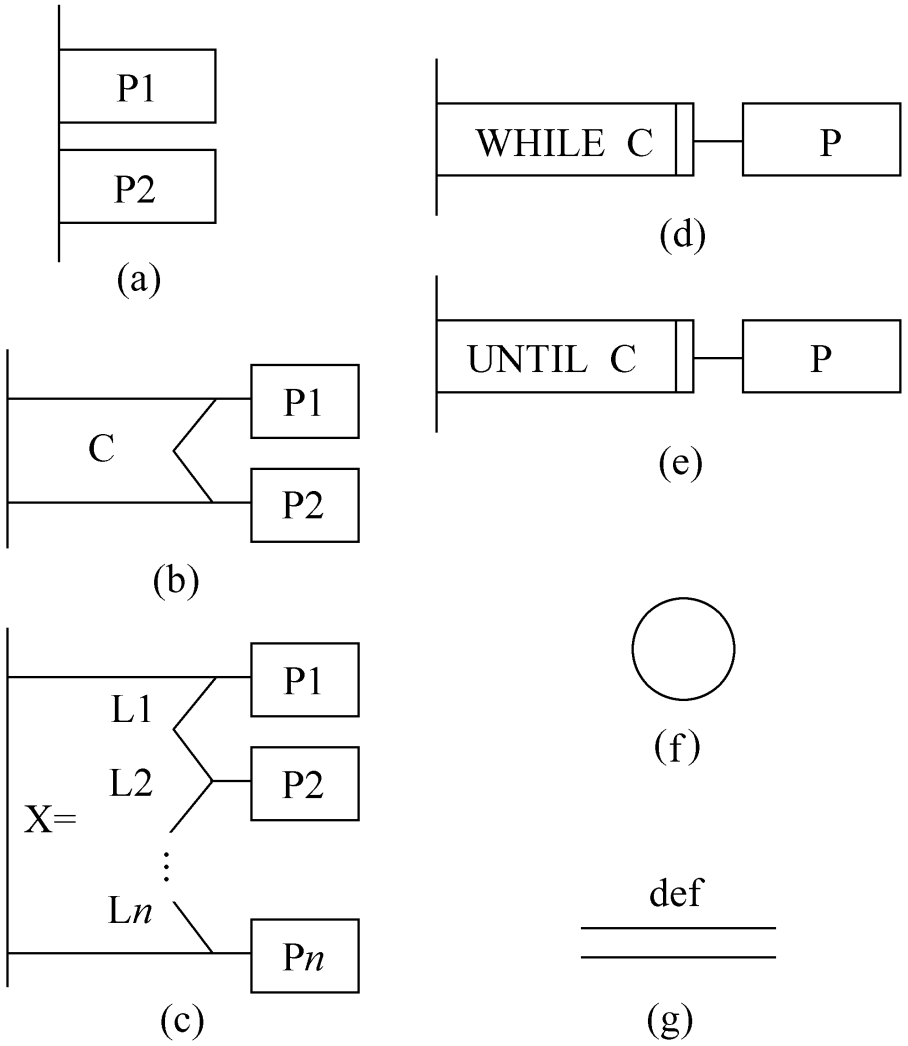
- PAD（Problem Analysis Diagram）是日立公司创立的一种采用二维、树形结构表示程序的控制结构的结构化设计方法。
- PAD为多种高级程序设计语言（Fortran、Cobol、Pascal等）根据其语言结构特点都设计了一套符号。

PAD的优点——结构化

- 程序结构清晰：垂直（竖线）方向表明了结构的顺序，水平方向（结构）表明了结构嵌套的层次及深度（竖线的数目）。
- 程序逻辑清晰：很好地表现程序逻辑，易读、易懂、易记。自上而下、从左向右执行，可遍历程序的所有节点（图论中的“广度优先”遍历）。
- 易于转换成高级程序设计语言的源程序：PAD规范，易于自动转换，有利于提高软件可靠性和生产率。
- 表达能力强：既可容易地表达程序逻辑，也可表达数据结构。
- 支持“自顶向下、逐步求精”的设计方法：在上层进行抽象，使用“def”符号在下层逐步细化。

PAD的不足——水平嵌套层次太深时，表达显得有点凌乱。

6.3.3 问题分析图(PAD图)——基本控制结构



(a) 顺序结构

(b) 选择结构

(c) 多分枝选族结构

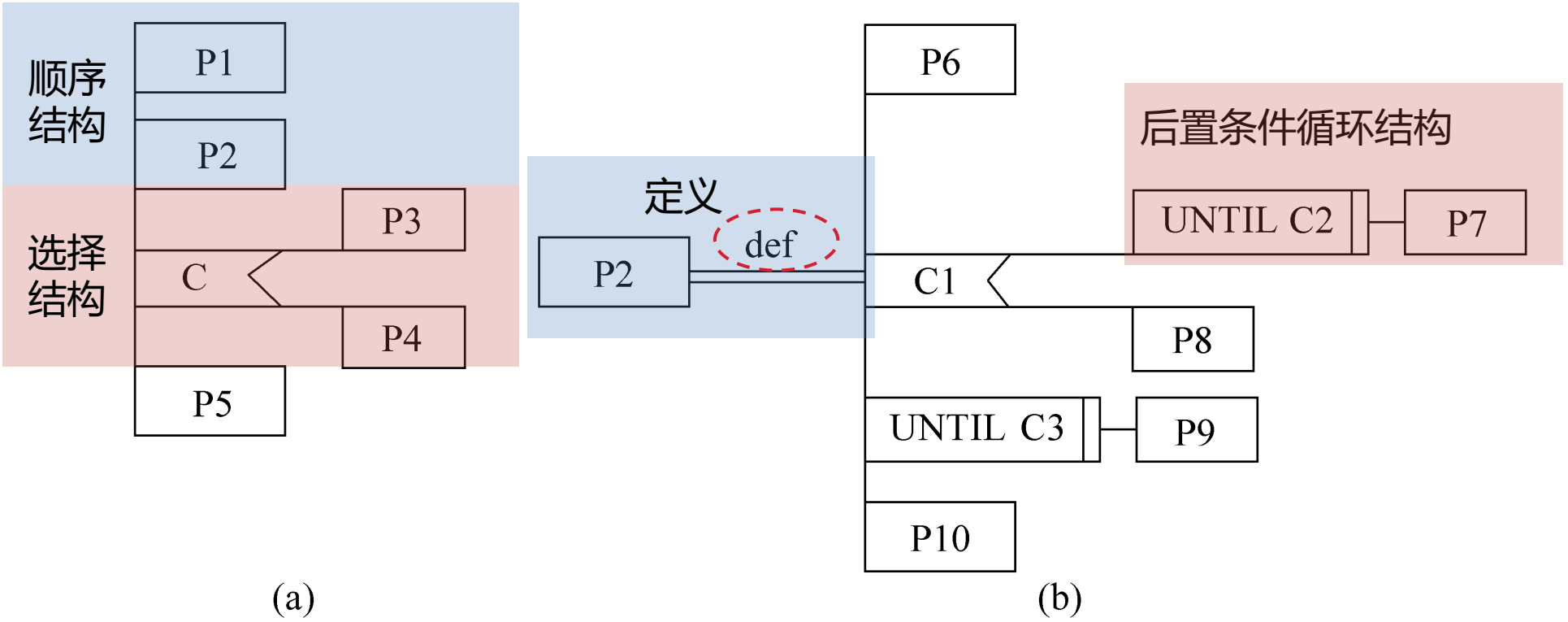
(d) 前置条件循环结构

(e) 后置条件循环结构

(f) 语句标号

(g) 定义

6.3.3 问题分析图——使用“def”逐步求精示例



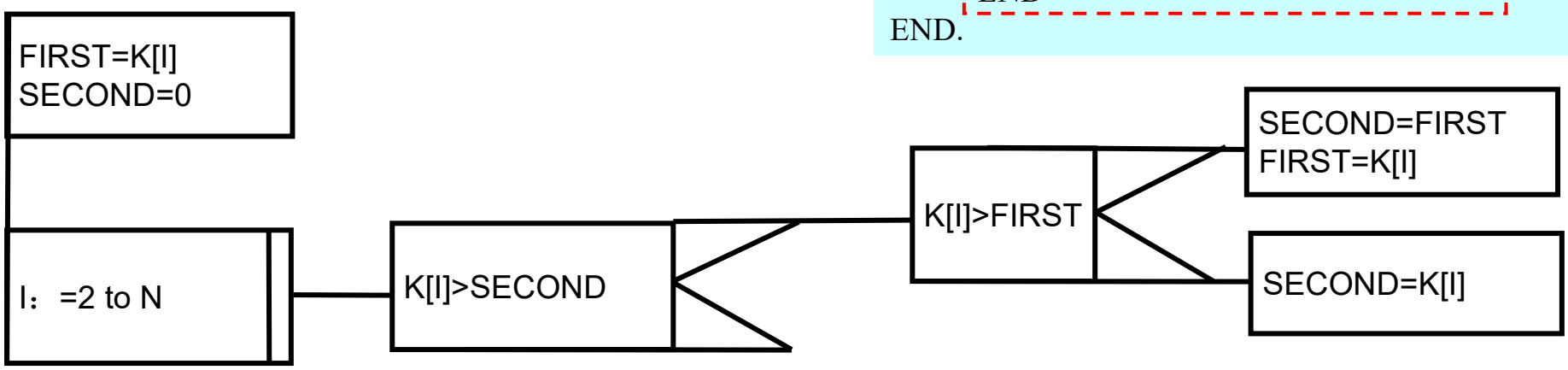
使用PAD图提供的定义功能来逐步求精的例子

PAD应用示例

算法功能：在数组K[1..N]中（所有元素不小于0），找出最大（FIRST）和次大（SECOND）的数。

PASCAL源程序段

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



6.3 过程设计的工具

6.3.1 程序流程图

6.3.2 盒图（N-S图）

6.3.3 问题分析图（PAD图）

6.3.4 判定表

6.3.5 判定树

6.3.6 过程设计语言（PDL）

6.3.4 判定表

判定表

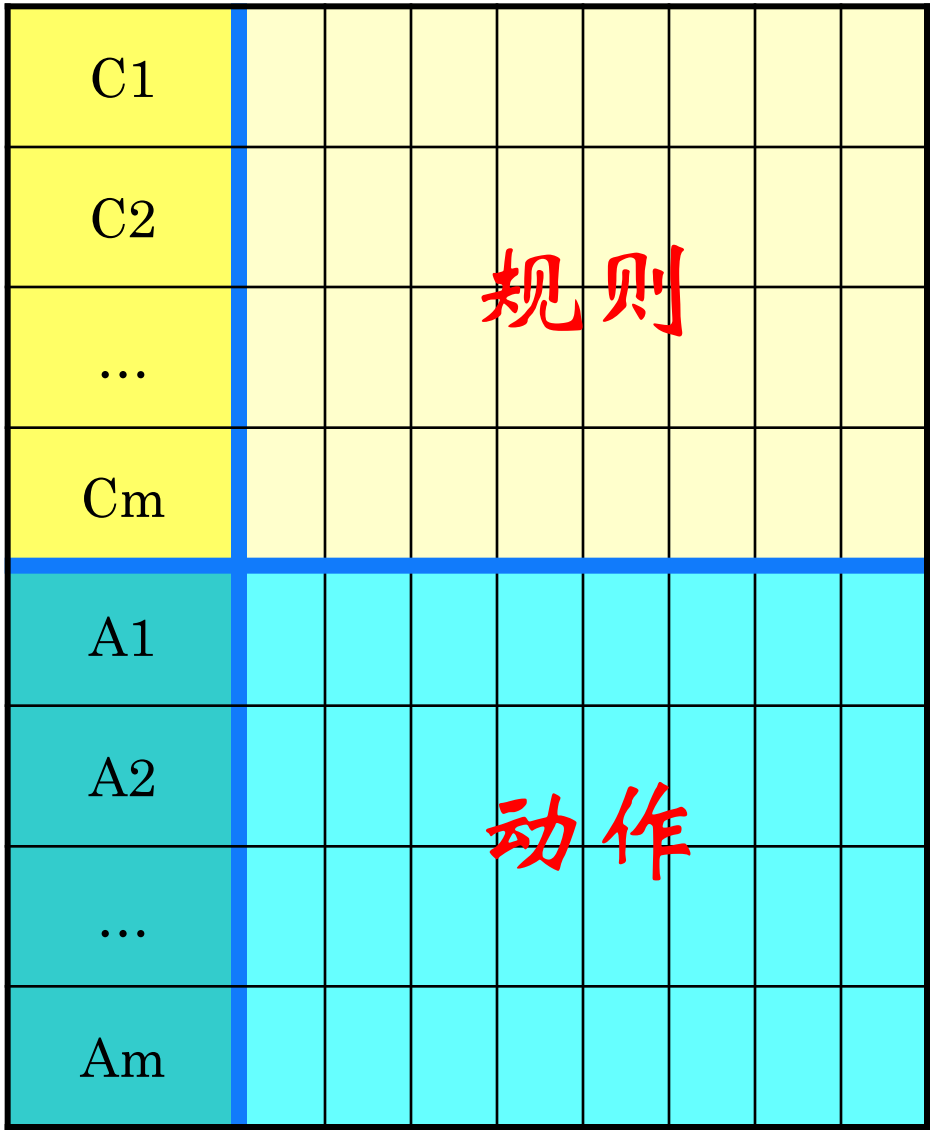
- 表达复杂的条件组合与需执行的动作之间的对应关系的一种关系表。

判定表由四部分组成：

- 基本条件
- 条件组合矩阵 （规则，或条件表达式）
- 基本动作
- 动作组合矩阵（条件组合对应要执行的动作）

判定表特点

- 可准确地表达“规则-操作”
- 条件表达式（规则）可简化
- 只适用于此，作为其他设计工具的补充



6.3.4 判定表：行李托运示例1

假设某航空公司规定，

1. 不超过30kg的行李：免费
2. 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。

6.3.4 判定表：示例2

假设某大学要从学生中挑选男子篮球队队员，基本条件是：

- 各门课程的平均分在 70 分以上，
- 身高超过 1.80 米，
- 体重超过 75 公斤。

需要从学生登记表中挑选出符合上述条件的男同学，并列出他们的姓名和住址，以便进一步选拔。

		1	2	3	4	5
条件	是男同学	T	F			
	平均分高于70分	T		F		
	身高超过1．80米	T			F	
	体重超过75千克	T				F
动作	并列出他们的姓名和住址	Y				
	拒绝这个学生		Y	Y	Y	Y

6.3 过程设计的工具

6.3.1 程序流程图

6.3.2 盒图（N-S图）

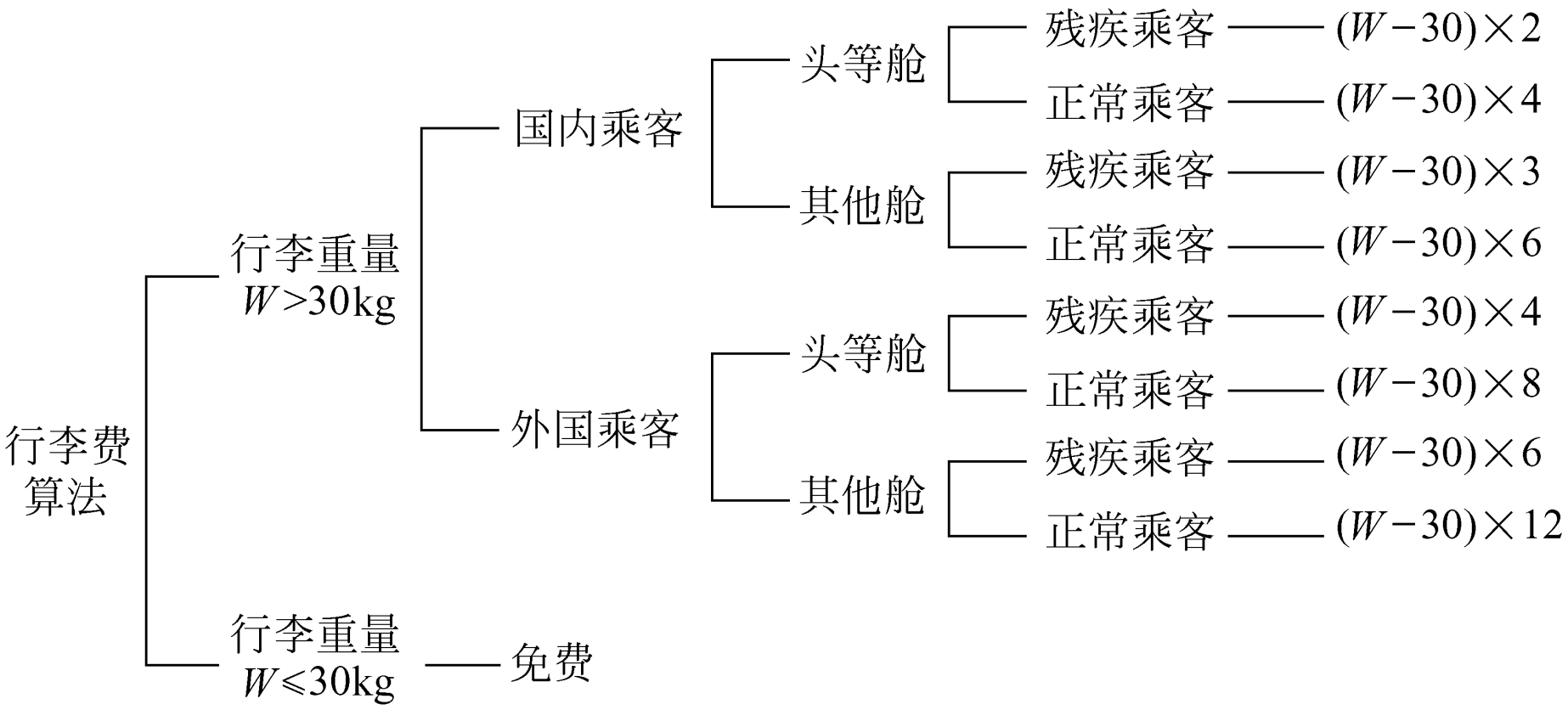
6.3.3 问题分析图（PAD图）

6.3.4 判定表

6.3.5 判定树

6.3.6 过程设计语言（PDL）

6.3.5 判定树例



用判定树表示计算行李费的算法

6.3 过程设计的工具

6.3.1 程序流程图

6.3.2 盒图（N-S图）

6.3.3 问题分析图（PAD图）

6.3.4 判定表

6.3.5 判定树

6.3.6 过程设计语言（PDL）

6.3.6 过程设计语言（PDL）

PDL（Procedure Design Language）过程设计语言

- 也称为伪码，是介于自然语言和程序设计语言之间的软件设计语言。
- PDL是非形式化比较灵活结构化的语言，用于描述模块内部过程的具体算法，以便在开发人员之间比较精确的进行交流。

PDL的语法

- PDL的语法是开放式的，其外层语法是确定的，采用类似于一般程序设计语言控制结构和关键字。为了区别关键字，关键字一律大写，其它单词一律小写。
- 内层语法则不确定，一般使用自然语言（或半形式化语言）来描述处理特性。内语法比较灵活，只要写清楚就可以，不必考虑语法错，以利于人们可把主要精力放在描述算法的逻辑上。
- 这种语法一般称为“类-程序设计语言”，如Like-C，Like-JAVA，Like-Pascal等。

6.3.6 PDL特点

结构描述：

- 固定的关键字词法；
- 固定的程序结构语法；
- 结构化的控制结构；
- 模块化。

处理描述：

- 采用自然语言或半形式化语言描述操作、处理。

数据描述：规范的数据说明，包括：

- 简单数据结构的定义（如标量、数组）
- 复杂数据结构的定义（如记录、链表）

6.3.6 PDL的优点和缺点

PDL优点：

- PDL程序易于编写和编辑，使设计人员关注程序的逻辑而不是程序的语法。
- PDL程序转化成程序设计语言程序时，可用作很好的注释，提高了程序的可读性、可维护性。
- 存在将PDL转化成程序代码的自动化工具。

PDL缺点：

- 描述有些算法不如图形化工具形象直观。
- 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。

6.3.6 PDL控制结构语法例

顺序结构

```
p1;
P2;
……
pn
```

选择结构

```
IF  c THEN
    p1;
ELSE
    p2;
ENDIF
```

多分支选择结构

```
DO CASE  OF c
    c1: p1
    c2: p2
    ……
    cn: pn
    otherewise:    pn+1
ENDCASE
```

前置循环结构

```
WHILE c DO
    p
ENDDO
```

后置循环结构

```
REPEAT
    p
UNTIL c
```

6.3.6 PDL控制结构语法例

```
BEGIN
IF 九点以前 THEN
        do 私人事务;
IF 9点到18点 THEN
        工作;
ELSE
        下班;
ENDIF
END
```

6.3.6 PDL控制结构语法例

```
PROCEDURE spellcheck
BEGIN
 --* 分词
 LOOP get next word
     add word to word list in sort order
     EXIT  WHEN all words processed
 END  LOOP
     --* 查询词典
 LOOP get word from word list
          IF word not in dictionary THEN
          --* 显示不在词典中的单词
          display word prompt on user terminal
          IF user response says word OK THEN
             add word to good word list
          ELSE
             add word to bad word list
          ENDIF
     ENDIF
     EXIT WHEN all words processed
END LOOP
--* create a new words dictionary
  dictionary : = merge dictionary and good word list
END spellcheck
```

6.3.6 课堂练习

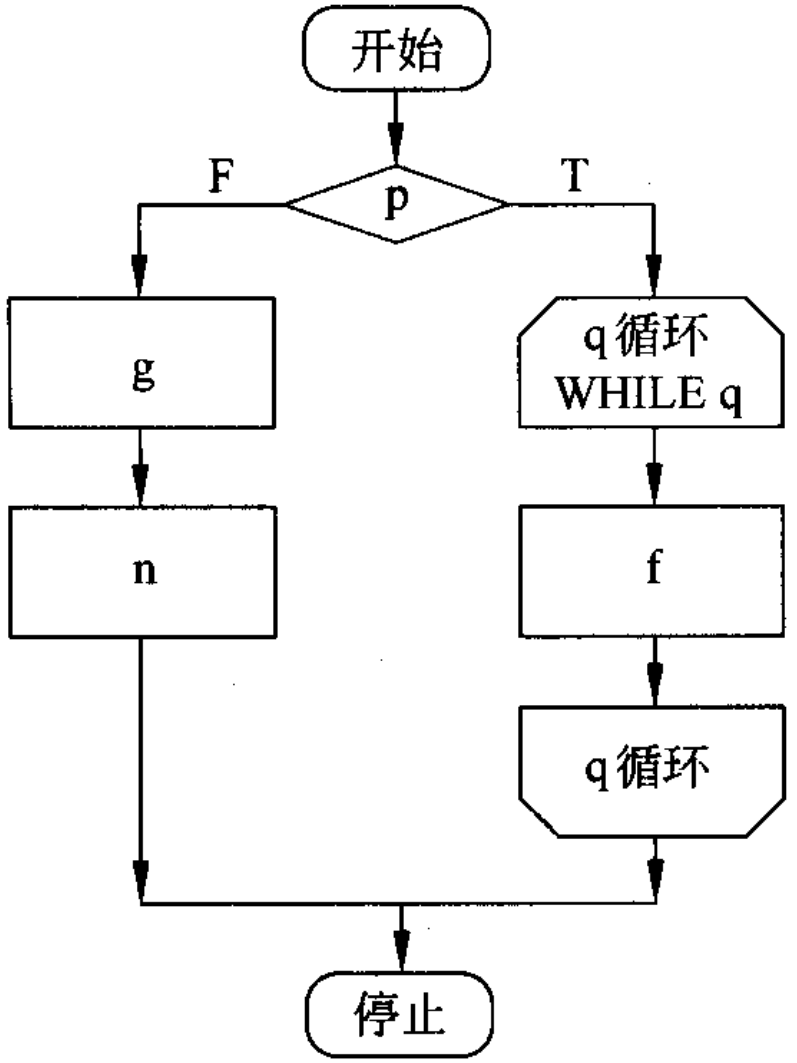
画出下列伪码程序的程序流程图和盒图

```
START
IF  p  THEN
    WHILE  q  DO
              f
    END  DO
ELSE
    BLOCK
    g
    n
    END  BLOCK
END  IF
STOP
```

6.3.6 课堂练习

画出下列伪码程序的程序流程图和盒图

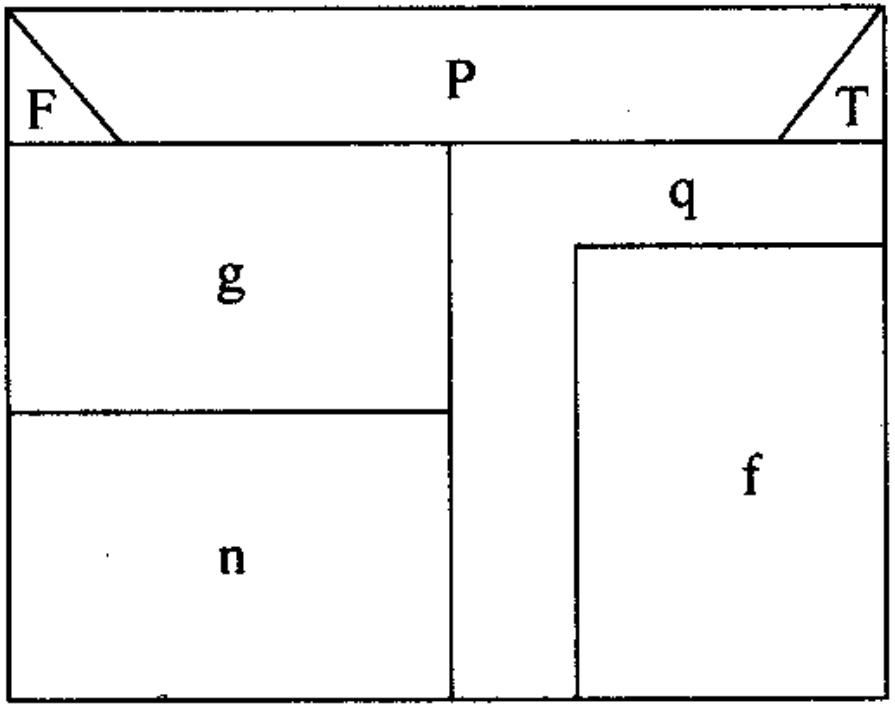
```
START
IF  p  THEN
    WHILE  q  DO
                f
    END  DO
ELSE
    BLOCK
    g
    n
    END  BLOCK
END  IF
STOP
```



6.3.6 课堂练习

画出下列伪码程序的程序流程图和盒图

```
START
IF  p  THEN
    WHILE  q  DO
                f
    END  DO
ELSE
    BLOCK
    g
    n
    END  BLOCK
END  IF
STOP
```



第6章 详细设计

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

6.4 面向数据结构的设计方法(Jackson方法)

6.4.1 Jackson图

6.4.2 改进的Jackson图

6.4.3 Jackson方法

6.4 面向数据结构的设计方法(Jackson方法)

6.4.1 Jackson图

6.4.2 改进的Jackson图

6.4.3 Jackson方法

6.4.1 JSD系统方法简介

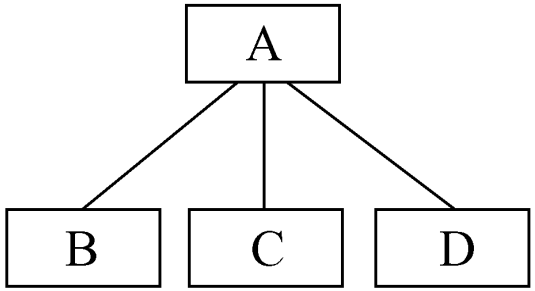
1. 结构化开发方法是一种面向数据流的开发方法，
2. Jackson系统开发方法则是面向数据结构的开发方法。其基本思想实现建立输入输出的数据结构，再将其转换为软件结构。

6.4.1 Jackson图

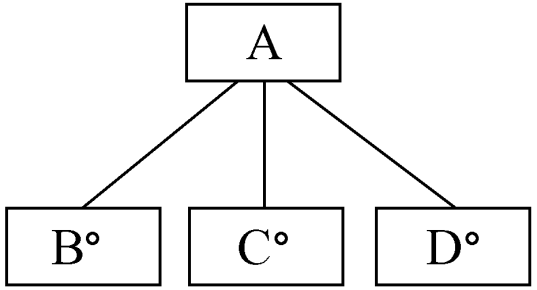
数据元素彼此间的基本逻辑关系只有顺序、选择和重复3类，因此逻辑数据结构也只有这3类。

1. 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。
2. 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。
3. 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。

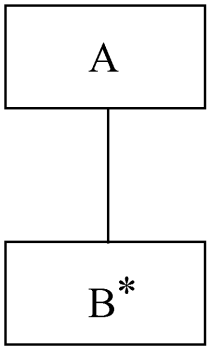
6.4.1 Jackson图的三种结构



顺序结构
A由B、C、D 3个元素顺序组成



选择结构
根据条件A是B或C或D中的某一个



重复结构
A由B出现N次(N≥0)组成

6.4 面向数据结构的设计方法(Jackson方法)

6.4.1 Jackson图

6.4.2 改进的Jackson图

6.4.3 Jackson方法

6.4.2 改进的Jackson图

Jackson图的缺点：

- 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；
- 框间连线为斜线，层次结构不严谨。

改进Jackson图：

- 上、下层之间的连线改成组织结构层次连线。
- 在重复结构的连线上标注循环条件。
- 在选择结构的连线上标注选择条件。
- 增加单分支选择结构，即在无操作的选择臂上使用“无关”结点框。

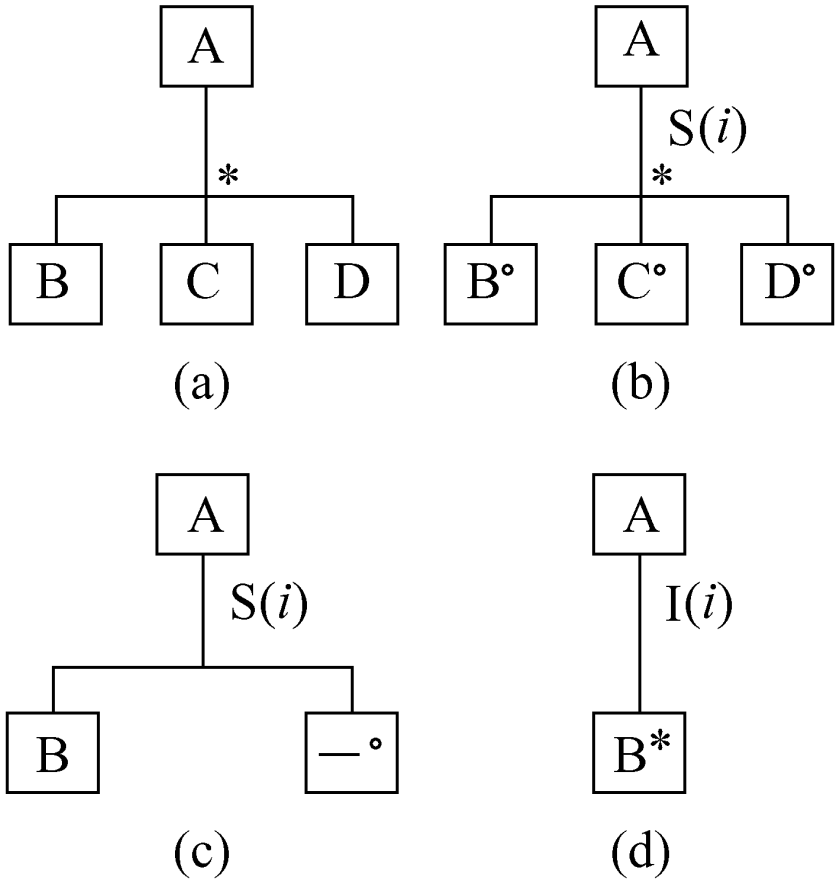
6.4.2 改进的Jackson图

(a)顺序结构

(b)多重选择结构，带选择条件

(c)单臂选择结构，带选择条件

(d)重复结构，带重复选择条件



6.4.2 Jackson图和软件结构层次图不同

Jackson图和描绘软件结构的层次图形式类似，但是含义却很不相同：

1. 方框中内容不同
   - 层次图中的一个方框通常代表一个模块；
   - Jackson图的一个方框只代表几个语句。
2. 上下方框之间的关系不同
   - 层次图表现的是调用关系；
   - Jackson图表现的是组成关系，一个方框中包括的操作仅仅由它下层框中的那些操作组成。

6.4 面向数据结构的设计方法(Jackson方法)

6.4.1 Jackson图

6.4.2 改进的Jackson图

6.4.3 Jackson方法

6.4.3 Jackson方法

JACKSON方法

- SD法是一种面向数据流的设计方法，
- JACKSON方法是一种面向数据结构的设计方法。强调程序结构与问题结构相对应。

JACKSON方法的构成

- JACKSON方法=JSP（Jackson Structured Programming)+JSD(Jackson System Development) 。
- JSP法主要体现程序结构的设计，不严格区分软件概要设计和详细设计。可以根据JSP的规则直接导出程序结构。一般用于规模不大的数据处理系统，而且I/O数据结构容易描述的情况。
- JSD法是对JSP法的扩充，针对JSP的缺陷提出解决方案。其主要特点是：用“分而治之”的策略控制系统的复杂性，解决I/O结构的冲突问题。

6.4.3 JSD方法的步骤

需求分析阶段

1、实体动作分析

2、实体结构分析

3、定义初始模型

软件设计阶段

4、功能描述：详细说明与已定义的动作相对应的功能。

5、绝对系统时间特性：对进程调度特性进行评价和说明。

6、实现：设计组成系统的硬件和软件。

6.4.3 JSD方法的步骤

1、实体动作分析：输入数据和输出数据的逻辑结构

2、实体结构分析：找出输入与输出有对应关系的数据单元

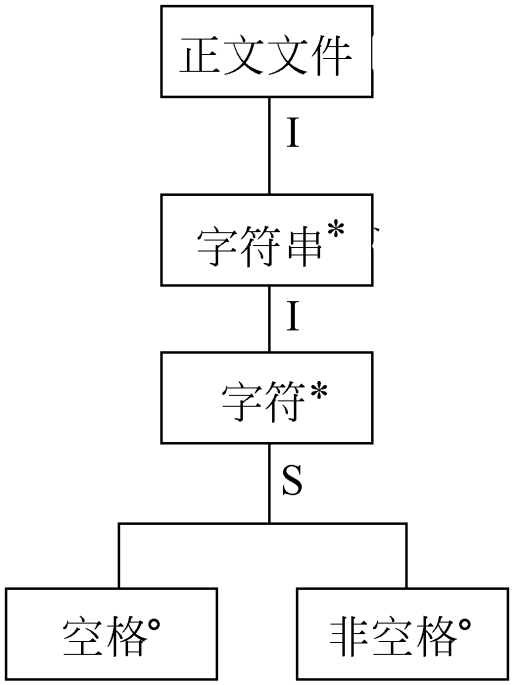
3、定义初始模型：从数据结构导出程序结构图

4、列出所有操作和条件

5、伪代码

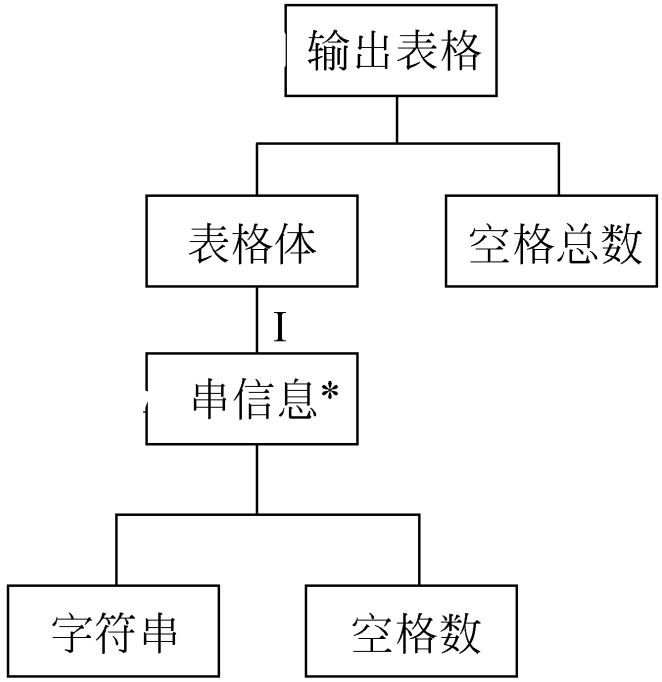
6.4.3 Jackson方法

1、实体动作分析：确定输入数据和输出数据的逻辑结构



(a)

输入数据结构

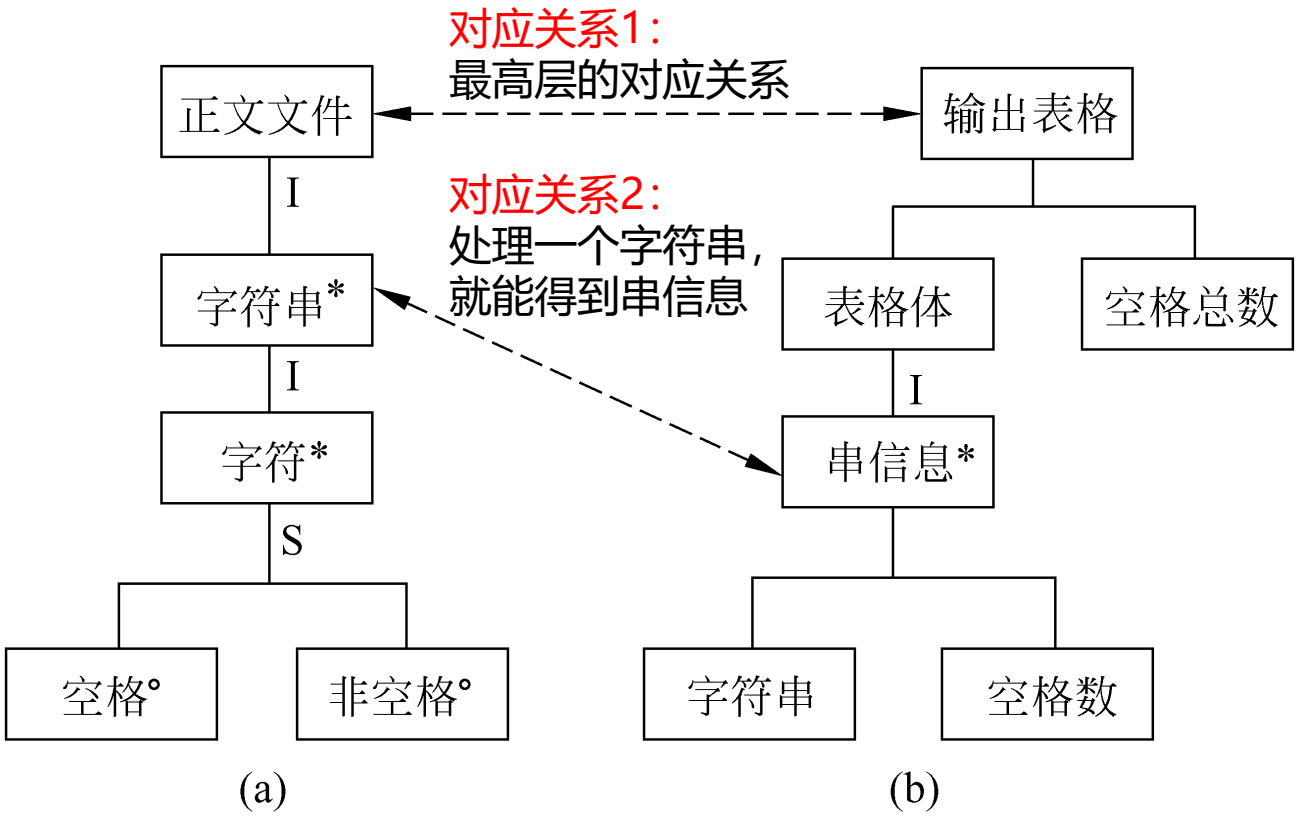


(b)

输出数据结构

6.4.3 Jackson方法

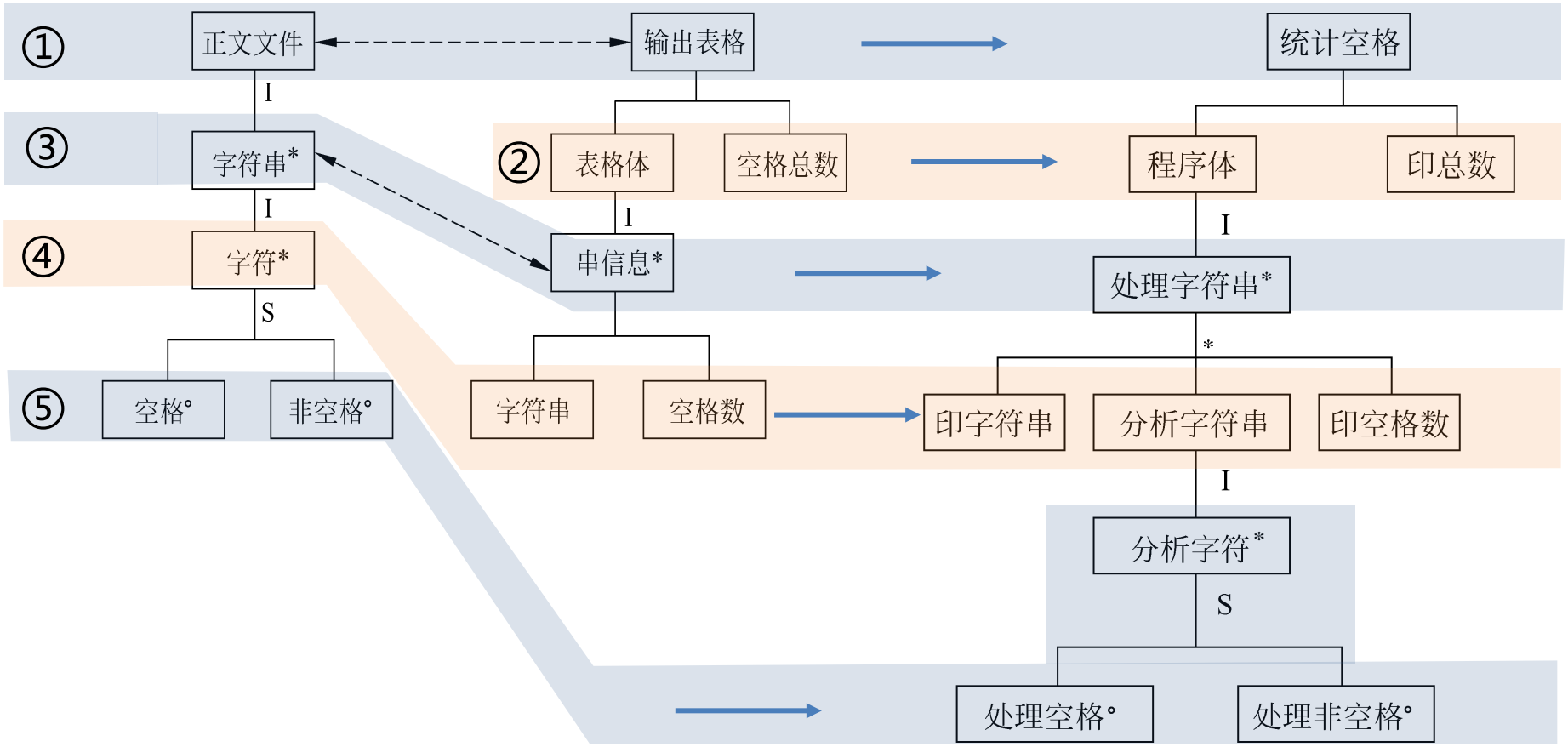
2、实体结构分析：找出输入与输出有对应关系的数据单元



输入、输出数据结构的Jackson图

6.4.3 Jackson方法

3、定义初始模型：从数据结构导出程序结构图



6.4.3 Jackson方法

4、把操作和条件分配到程序结构图的适当位置

(1) 停止　　(2) 打开文件

(3) 关闭文件　　(4) 打印字符串

(5) 打印空格数目　　(6) 打印空格总数

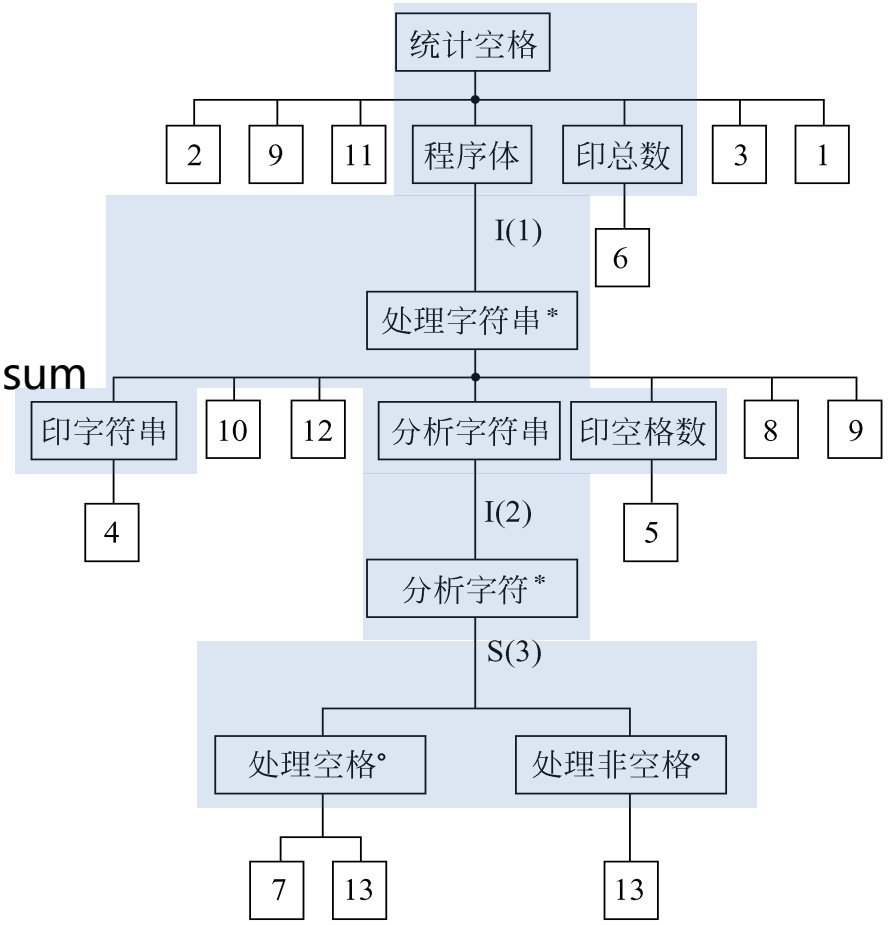
(7) sum:=sum+1　　(8) totalsum:= totalsum+sum

(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

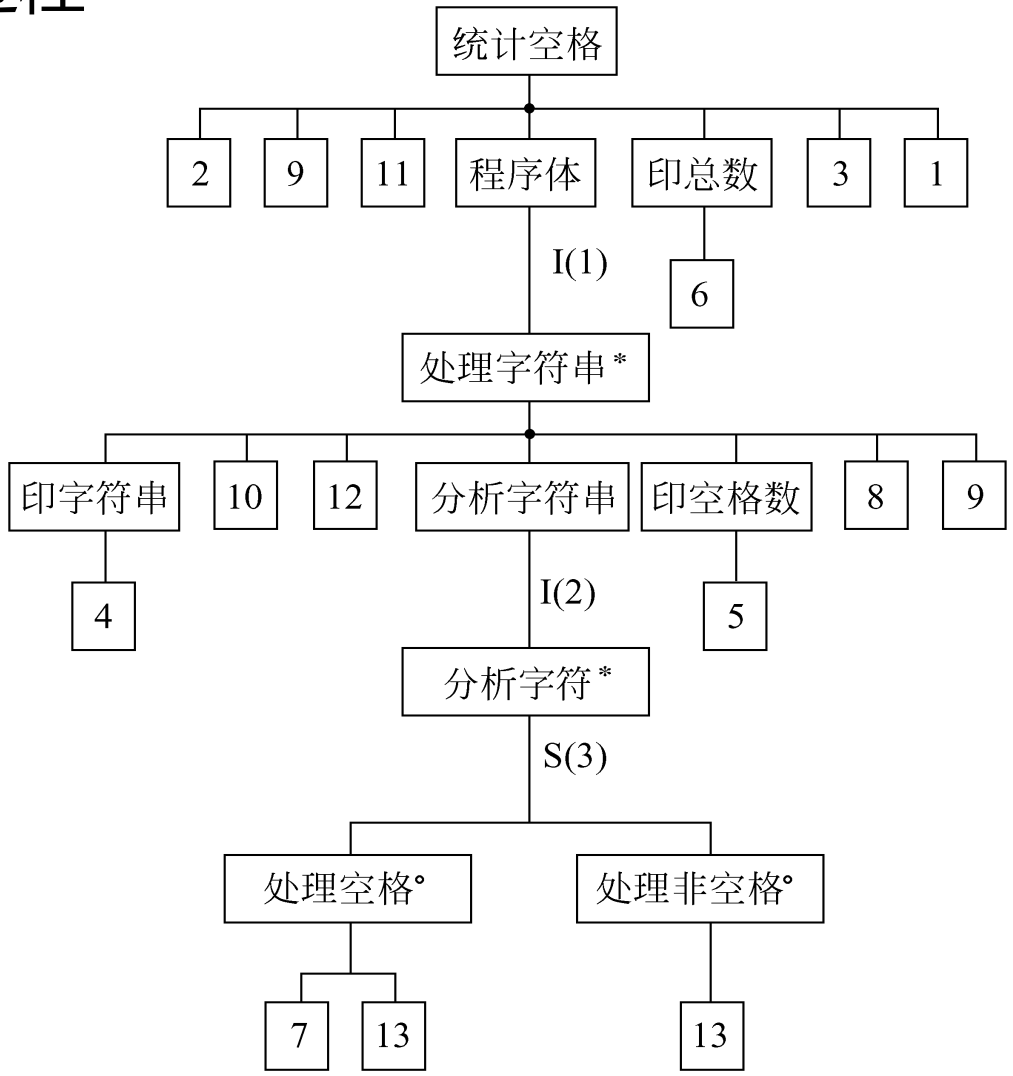
I (2)字符串结束　　S(3)字符是空格



6.4.3 Jackson方法

5、用伪代码表示程序处理过程

```
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
    ……
    ……
        处理字符串 end
    程序体 end
    ……
    关闭文件
    停止
统计空格 end
```



6.4.3 JACKSON方法的优点和缺点

优点：

- 结构清晰、易理解、易修改。
- 不会过多依赖于设计者的经验。

缺点：

- 当系统规模及复杂度大时，确定数据结构困难。

第6章 详细设计

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

6.4 程序复杂程度的定量度量

程序的复杂程度定量度量的意义：

- 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。

介绍定量度量的二个著名方法

- 6.4.1　McCabe方法：McCabe根据图论定义了“循环数”来得到一种软件复杂性度量方法，即著名的McCabe循环复杂度（Cyclomatic Complexity）。
- 6.4.2　Halstead方法：Halstead度是基于程序源代码。Halstead指出估计工作量，或者程序员工作时间，可以用运算符，运算元或语法数的函数来表示。

6.4.1 McCabe方法

McCabe方法概要

- McCabe方法根据程序控制流的复杂程度定量地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”。

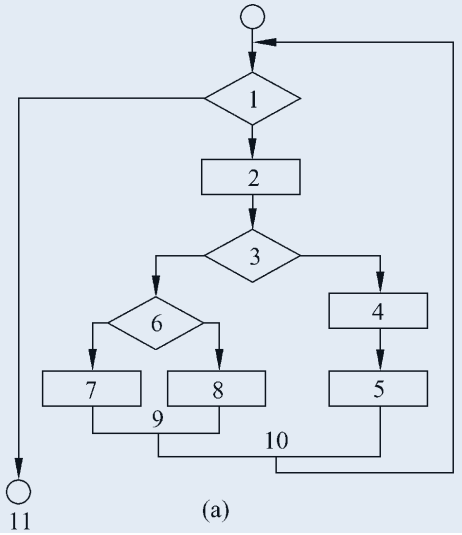
主要内容

- 流图概念及变换
- 计算环形复杂度的方法
- 环形复杂度的用途

6.4.1 McCabe方法——流图

流图

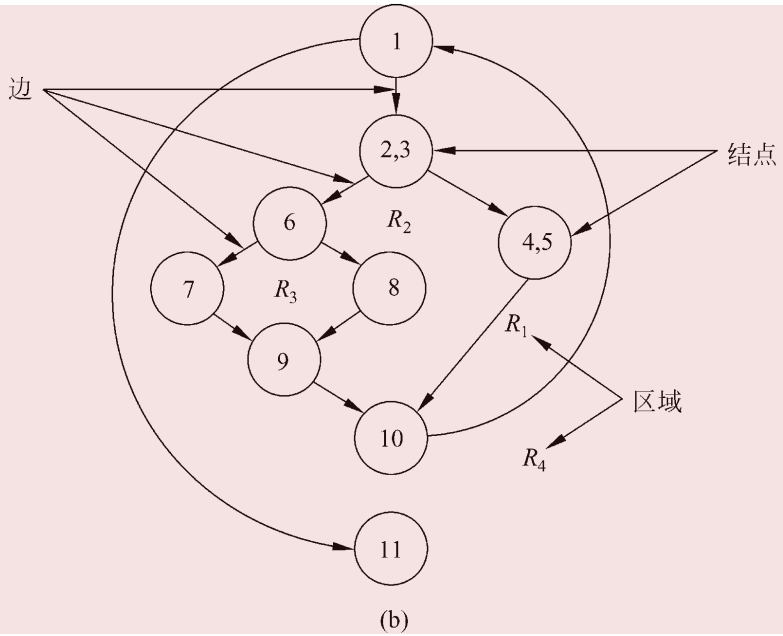
- “流图” 也称为“程序图”。
- 流图是退化了的程序流程图，仅仅描述了程序的“控制流”，完全不表示程序中的具体操作和分支或循环的具体判定条件。



(a)

流图的构成

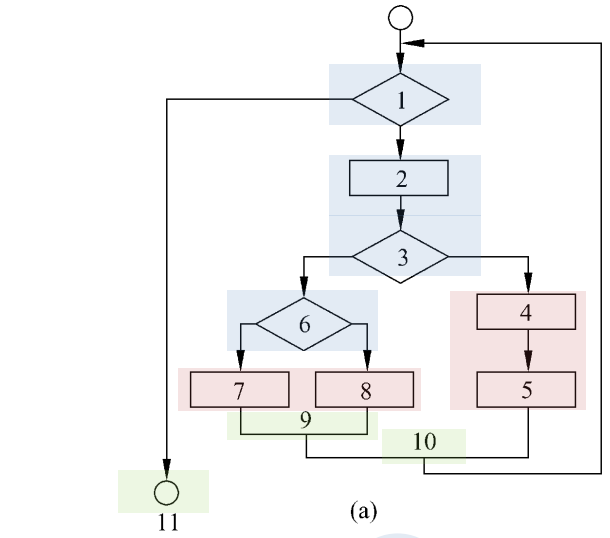
- 结点：流图中的圆点，表示一个元判定，程序中连续的操作和接着的一个判定可映射成一个结点。
- 边：流图中的箭头线，表示控制流，每条比边必须终止于某个结点。
- 区域：由流图的边所围成的封闭区域，所有区域以外的开区域（这样的区域只有一个）也是流图的一个区域。



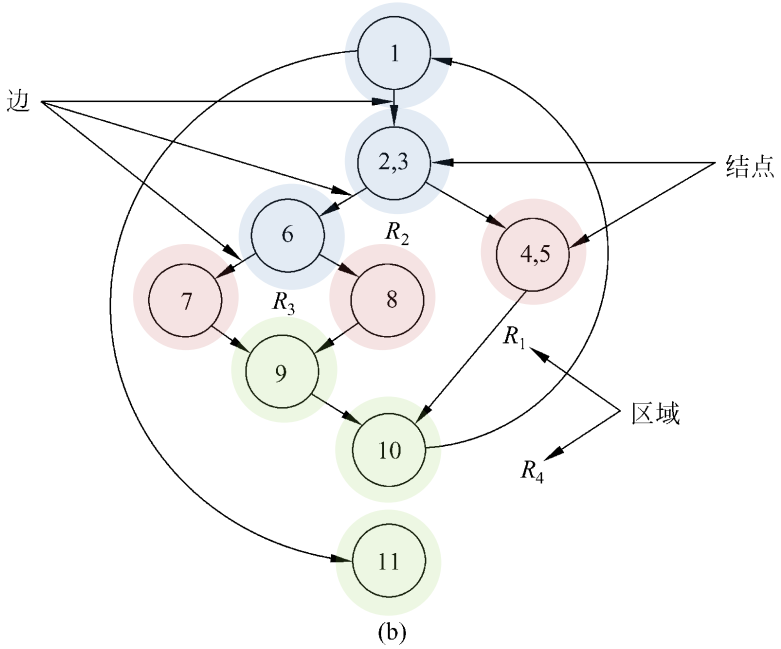
(b)

6.4.1 McCabe方法——程序流程图映射成流图

1. 从程序（或程序段）的入口开始，按程序流程顺序处理；
2. 每个判定设置一个结点(1, 3, 6)；
3. 将两个判定之间的所有连续操作忽略不计，或合并到后续的第一个判定结点(2, 3)；
4. 后续没有判定的操作（或连续操作合并）各设置一个结点(4和5，7, 8)；
5. 每个结构的出口汇合点各设置一个结点（9, 10，11）；
6. 按照程序流程图的逻辑关系用边（箭头线）连接起来。



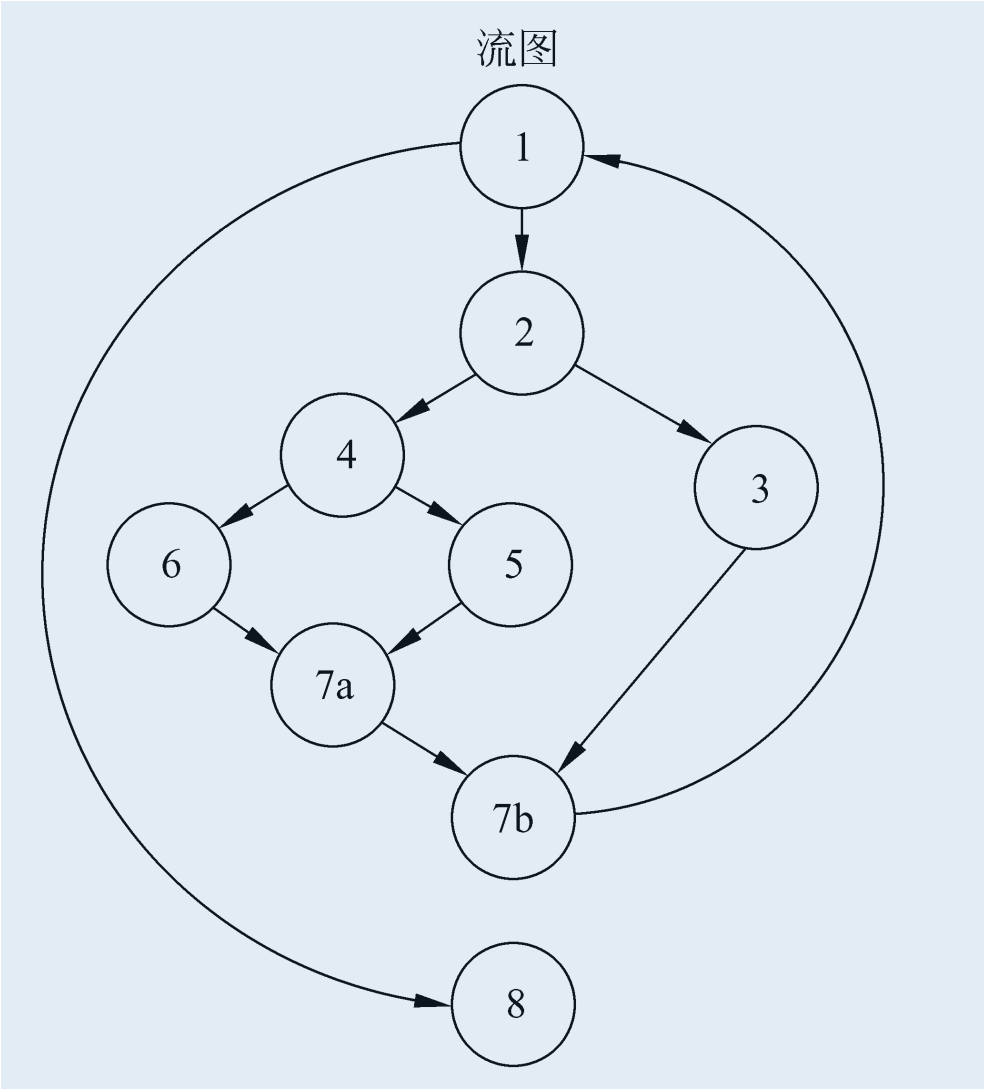
(a)



(b)

6.4.1 McCabe方法——PDL翻译成流图

```
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
```



6.4.1 McCabe方法——包含复合条件的的流图

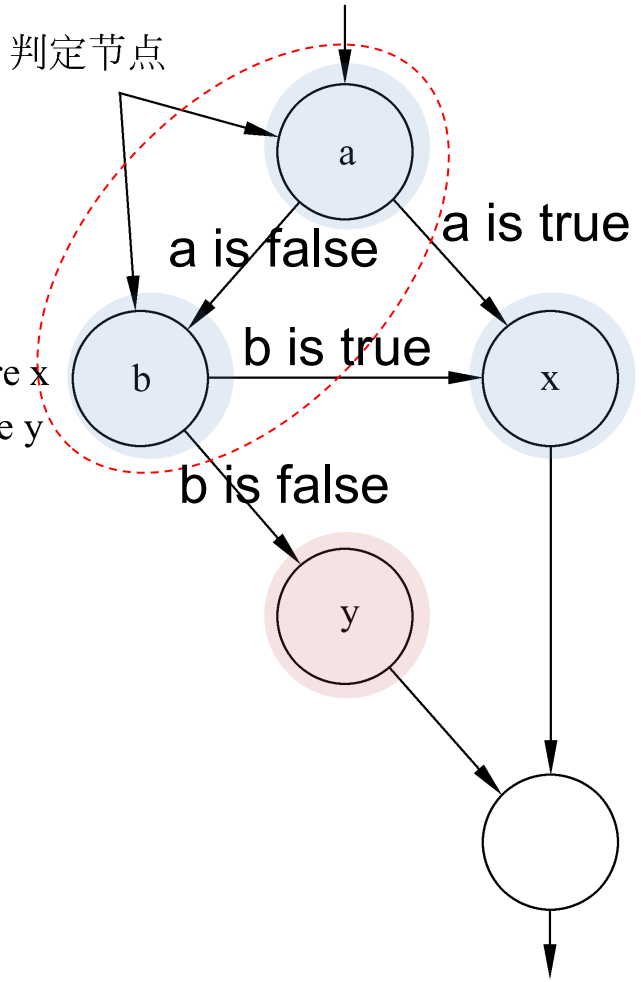
复合条件

- 指含有布尔运算符的条件：and、or、nand、nor等

复合条件的变换

- 当程序中包含复合条件时，将其拆分成若干个简单条件（元条件），每个简单条件对应一个结点（如图a、b）。
- 同一个复合条件中简单条件结点串联起来，且连接到同一操作结点。

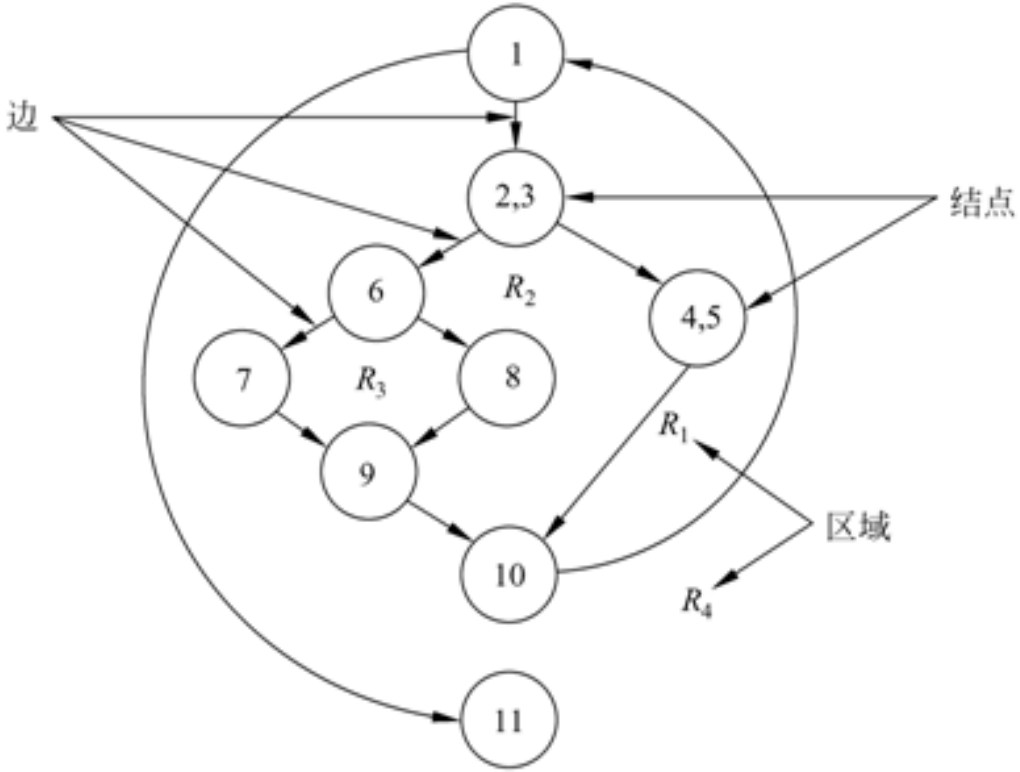
```
⋮
IF a OR b
   then procedure x
   else procedure y
ENDIF
⋮
```



6.4.2 McCabe方法—— 计算环形复杂度的方法

可采用下面任何一种方法计算“流图的环形复杂度”：

1. 使用区域数计算
   - V(G) = D
   - 其中D为区域数
2. 使用边与结点数计算：
   - V(G)=E-N+2
   - 其中：E为边数，N为结点数
3. 使用判定结点数计算：
   - V(G)=P+1
   - 其中：p为判定结点数



例：4个区域: R1, R2, R3, R4

V(G)=E-N+2=11-9+2=4

V(G)=P+1=3+1=4

6.4.2 McCabe方法——环形复杂度的用途

预测程序的复杂度

- 程序的“环形复杂度”取决于程序控制流的复杂程度，即反映了程序控制结构（选择结构、循环结构）的复杂程度 。
- 因此，环形复杂度可用于预测程序的测试难度和预测软件可靠性。

帮助控制程序模块的规模

- V(G)高的程序结构复杂，难于实现，容易出错。
- 因此，环形复杂度V(G)可用于帮组控制模块规模，参考的模块规模为V(G)≤10 。

6.4.2 Halstead方法

Halstead方法概要

- 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。

程序长度N

- N1——程序中运算符出现的总次数

  N2——程序中操作数出现的总次数
- 则程序的总长度 N=N1+N2

预测长度H(详细设计完成之后)

- n1 ——程序中不同运算符（包括关键字）出现的总次数

  n2 ——程序中不同操作数（包括变量和常量）出现的总次数
- 则程序的预测总长度　H=n1 $\log_2$ n1 + n2 $\log_2$ n2

程序中错误数E的预测

- E=N $\log_2$（n1 + n2 ）/ 3000

Halstead方法的评价

可操作性强

- Halstead度量方法以程序中出现的运算符(Operator)和操作数(Operand)为计数对象，以它们的出现次数作为计数目标来测算程序容量和工作量。 该方法可操作性强，易于使用、易于计算。

应用性强（软件规模、开发、测试、错误度量）

- Halstead度量方法不仅仅度量了程序长度，还描述了程序的最小实现和实际实现之间的关系，并据此阐释程序语言的等级高低。还可预测程序中的错误数。

计算结果与实测贴近

- 根据试验统计， Halstead度量方法的计算结果（程序长度、错误数）与实际非常接近。（误差在8%以内）

6.6小结

详细设计的关键任务

- 在总体设计的基础上，进一步确定软件怎样具体实现。
- 主要任务包括人机界面设计、模块过程设计。

人机界面设计（指导原则）

- 重视4个人机界面设计问题：系统响应时间、用户帮助设施、出错信息处理、命令交互方式。
- 领会3类人机界面设计指南：一般交互、信息显示、数据输入。

过程设计（工具）

- 掌握结构化设计工具（图形、表格、语言）：程序流程图、盒图、PAD图、判定表、判定树、PDL、Jackson方法。

程序复杂性度量（方法）

- McCabe方法：基于程序流中“循环数”的环形复杂度。
- Halstead方法：基于程序中运算符和运算数的程序长度。