中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

软件工程

主要内容

第一章 软件工程学概述
第二章 可行性研究
第三章 需求分析
第四章 总体设计
第五章 详细设计
第六章 编码与测试
第七章 软件维护
第八章 面向对象方法学
第九章 面向对象分析设计与实现
第十章 软件项目管理

第五章　详细设计

第一节　结构程序设计

第二节　人机界面设计

第三节　过程设计的工具

第四节　向数据结构的设计方法

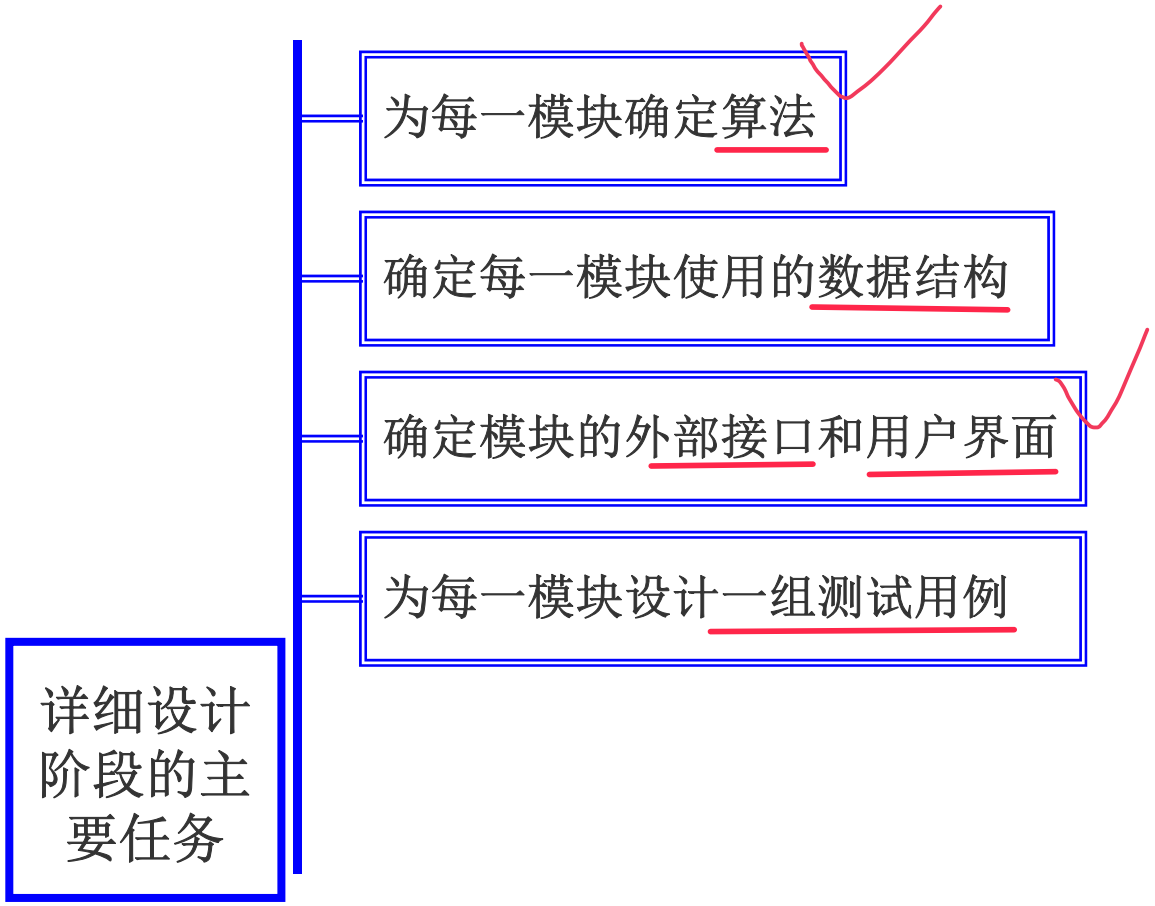
第五节　程序复杂程度的定量度量

第五章　详细设计

- 详细设计阶段的根本目标是确定如何具体实现所要求的系统。即，给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直接翻译成用某种程序设计语言书写的程序。
- 详细设计的结果基本上决定了最终程序代码的质量。

---- 确定如何具体实现所要求的系统。

详细设计阶段的目的与任务：



第五章　详细设计

第一节　结构程序设计

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。

第二节　人机界面设计

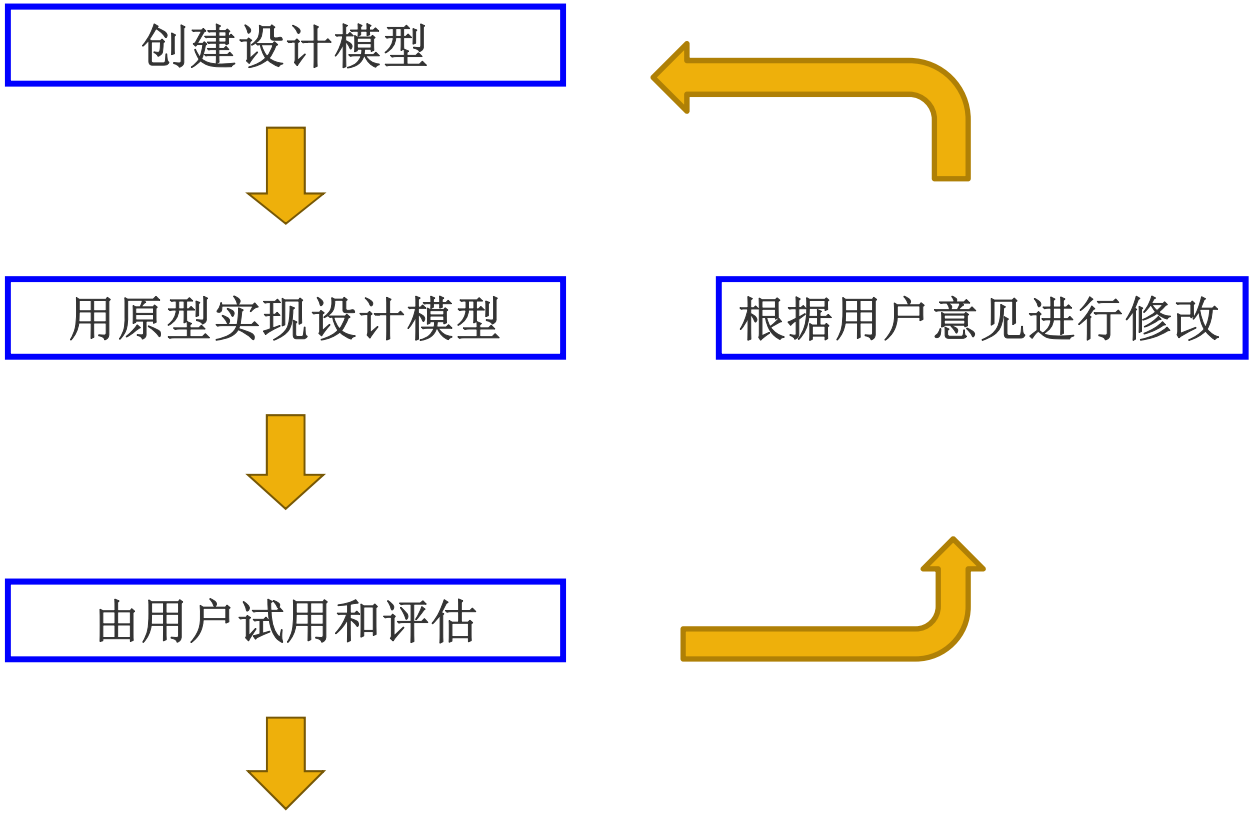
人机界面的设计质量，直接影响用户对软件产品的评价，从而影响软件产品的竞争力和寿命，因此，必须对人机界面设计给予足够重视。

一、人机界面设计问题

1. 系统响应时间（长度、易变性）
   - 系统响应时间过长，用户会感到紧张和沮丧；
   - 易变性是指系统响应时间相对于平均响应时间的偏差。
2. 用户帮助设施（集成、附加）
3. 出错信息处理（出错、警告）
4. 命令交互

二、人机界面设计过程

用户界面设计是一个迭代的过程。



三、人机界面设计指南

1. 一般交互指南

- 保持一致性
- 提供有意义的反馈
- 在执行有较大破坏性的动作之前要求用户确认
- 允许取消绝大多数操作
- 减少在两次操作之间必须记忆的信息量
- 提高对话、移动和思考的效率
- 允许犯错误。系统应该能保护自己不受严重错误的破坏
- 按功能对动作分类，并据此设计屏幕布局
- 提供对用户工作内容敏感的帮助设施
- 用简单动词或动词短语作为命令名

2. 信息显示指南

- 只显示与当前工作内容有关的信息。
- 不要用数据淹没用户，应该用便于用户迅速吸取信息的方式来表示数据。例如，可以用图形或图表来取代庞大的表格。
- 使用一致的标记、标准的缩写和可预知的颜色。
- 允许用户保持可视化的语境。
- 产生有意义的出错信息。
- 使用大小写、缩进和文本分组以帮助理解。
- 使用窗口分隔不同类型的信息。
- 使用“模拟”显示方式表示信息，使信息更容易被用户提取。
- 高效率地使用显示屏。

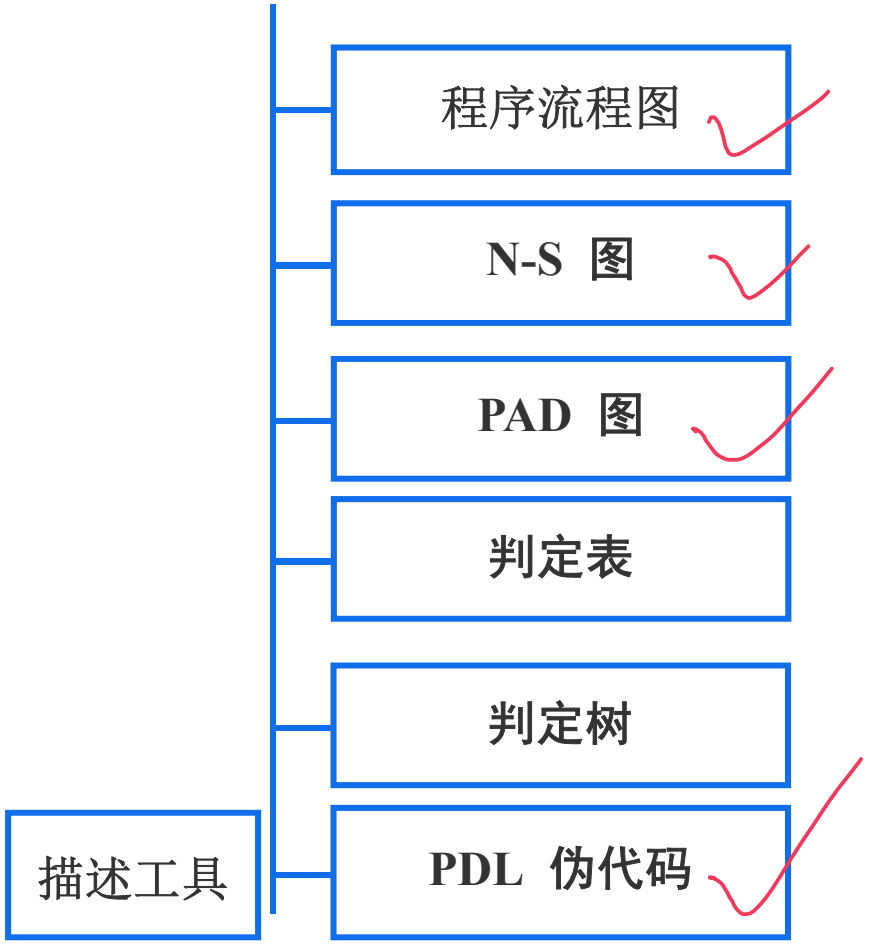
3. 数据输入指南

- 尽量减少用户的输入动作。
- 保持信息显示和数据输入之间的一致性。
- 允许用户自定义输入。
- 交互应该是灵活的，并且可调整成用户最喜欢的输入方式。
- 使在当前动作语境中不适用的命令不起作用。
- 让用户控制交互流。
- 对所有输入动作都提供帮助。
- 消除冗余的输入。

第三节　过程设计的工具

描述程序处理过程的工具称为过程设计工具，它们可以分为图形、表格和语言3类。

不论是哪类工具，对它们的基本要求都是能提供对设计的无歧义的描述。



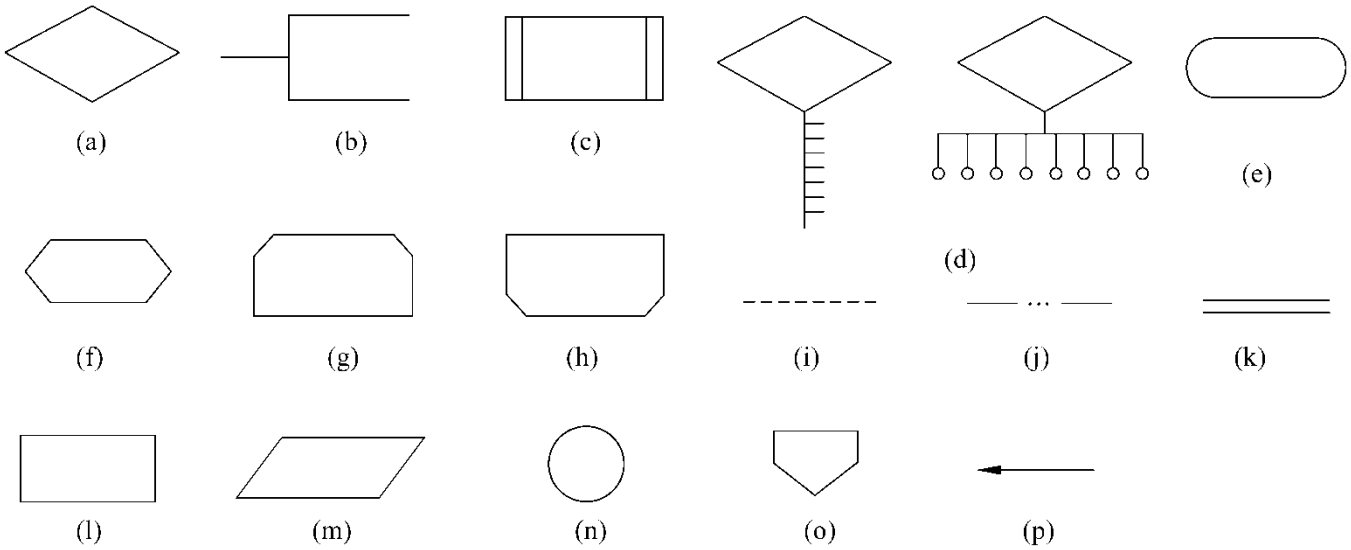
数据流图：数据在各个部位之间流动，箭头是数据

系统流程图：

一、程序流程图：详细描述程序执行过程、控制过程

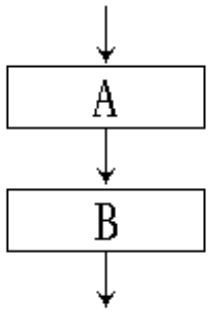
箭头：控制信息

1. 程序流程图中使用的符号

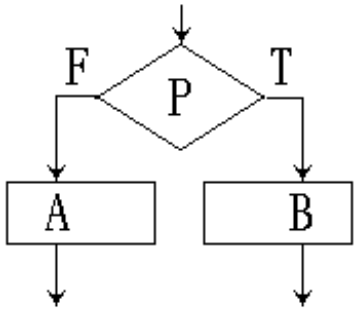


(a) 选择（分支）；(b) 注释；(c) 预先定义的处理；(d) 多分支；(e) 开始或停止；(f) 准备；(g) 循环上界限；(h)循环下界限；(i) 虚线；(j) 省略符；(k) 并行方式；(l) 处理；(m) 输入输出；(n) 连接；(o) 换页连接；(p) 控制流

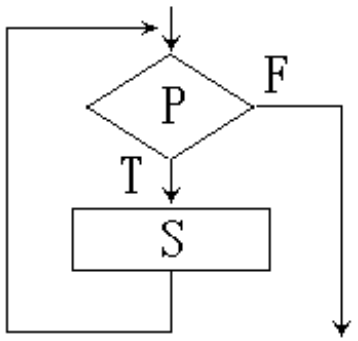
2. 程序流程图使用五种基本控制结构



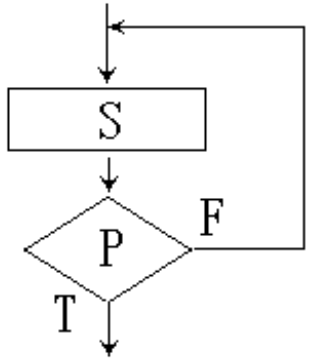
① 顺序型



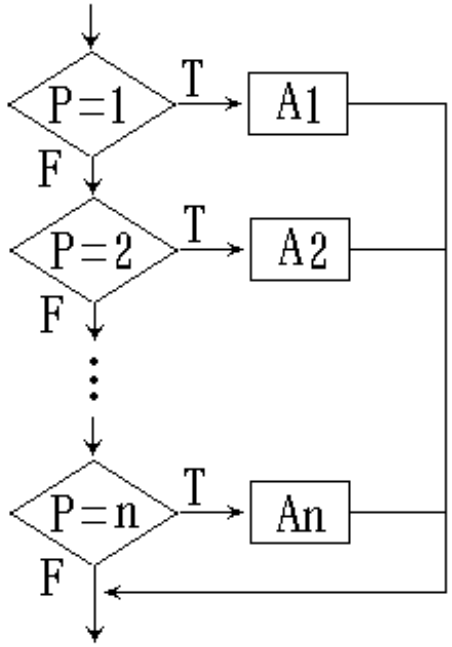
② 选择型



③ 先判定型循环
（DO-WHILE）

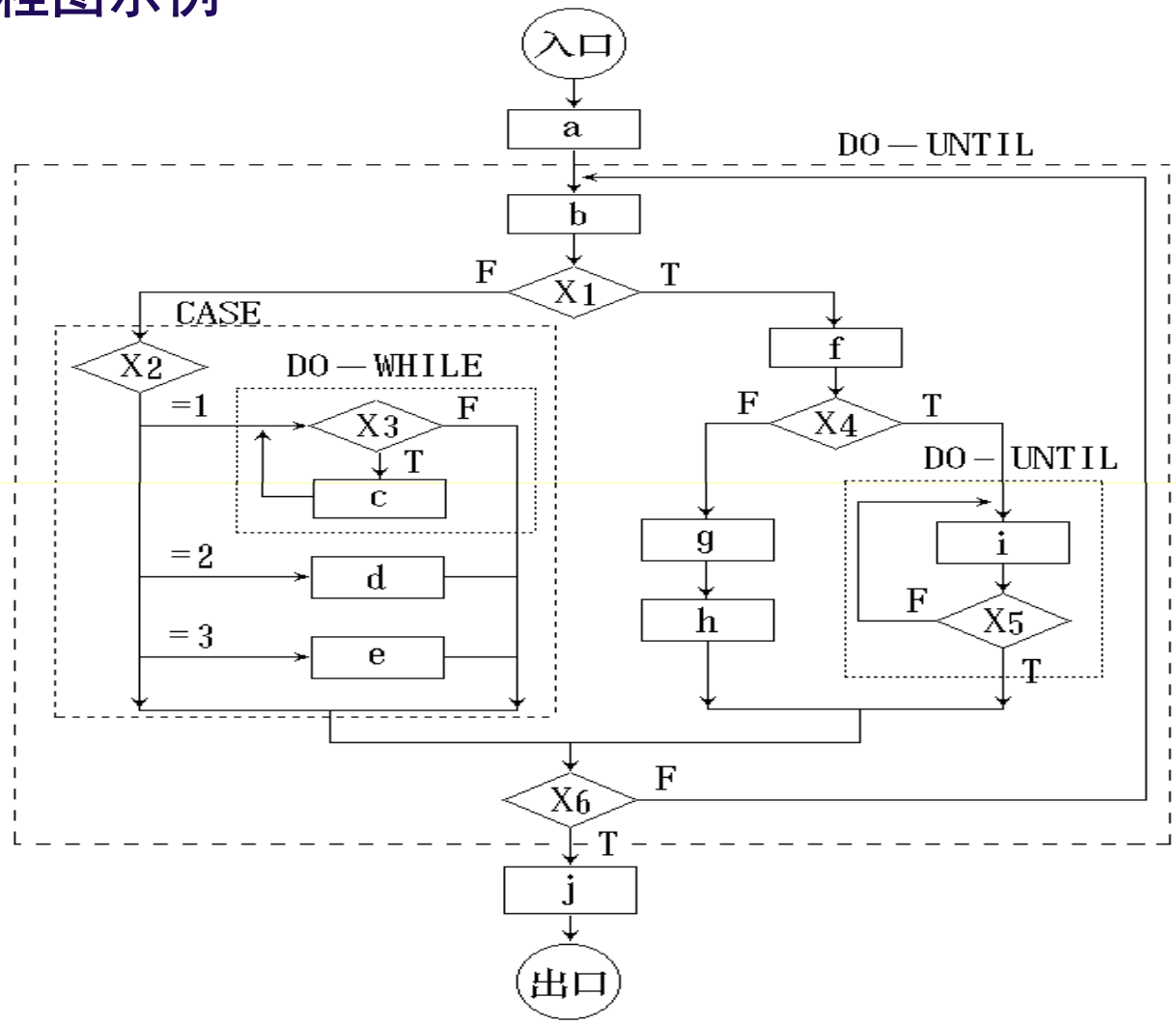


④ 后判定型循环
（DO-UNTIL）



⑤ 多情况选择型
（CASE型）

3. 程序流程图示例


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

4. 特点

（1）主要优点：

对控制流程的描绘很直观，便于初学者掌握。

（2）主要缺点如下：

- 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。
- 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
- 程序流程图不易表示数据结构。

二、盒图

1. 盒图

出于要有一种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，又称为N-S图。它有下述特点：

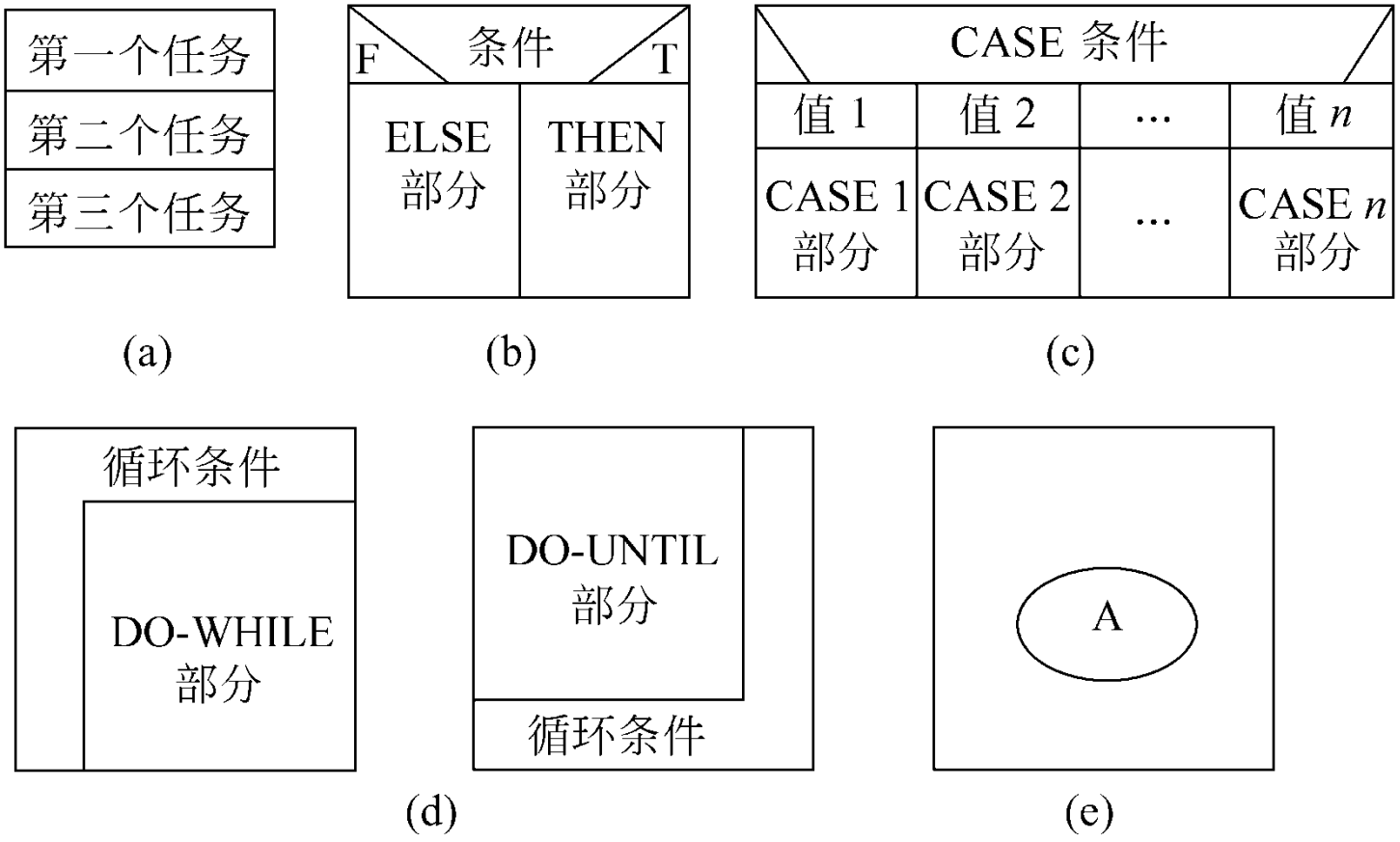
(1) 功能域(即一个特定控制结构的作用域)明确，可以从盒图上一眼就看出来；

(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；

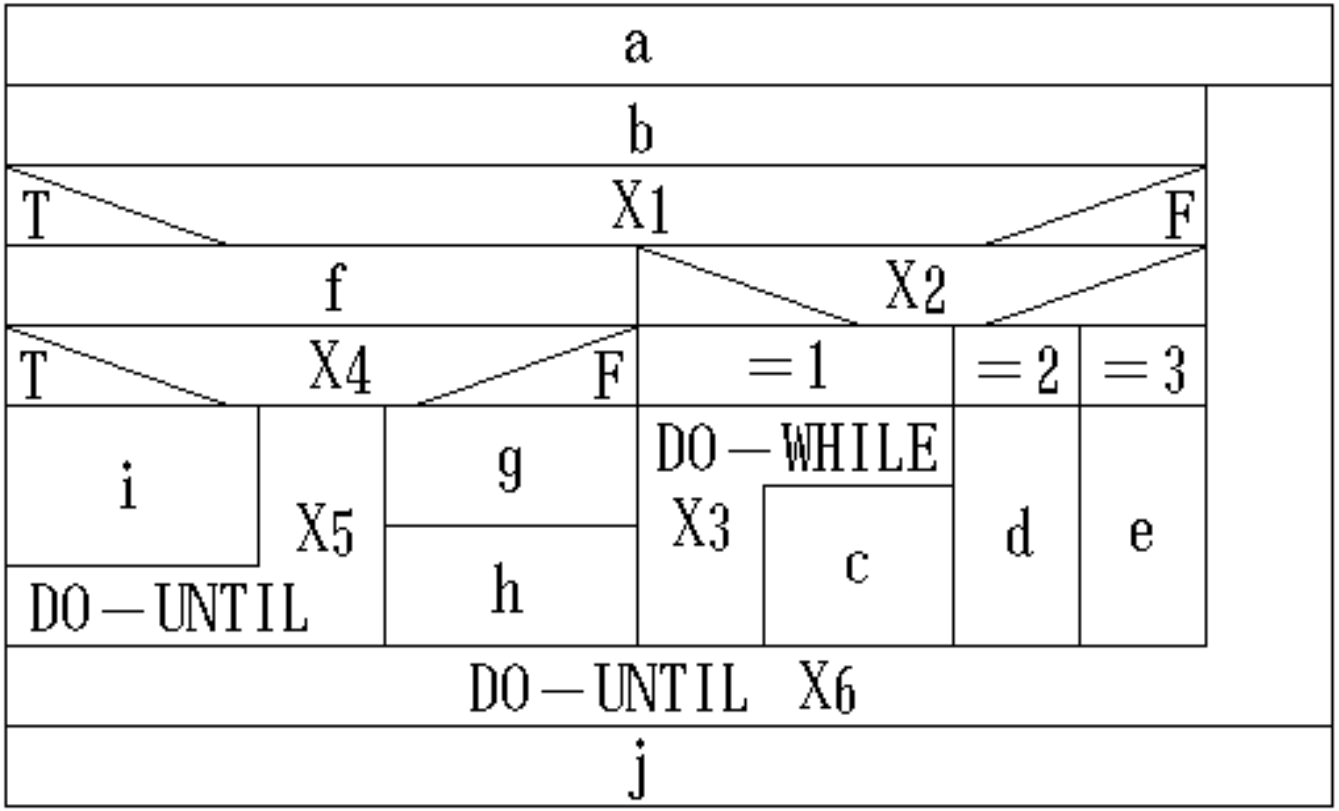
(4) 很容易表现嵌套关系，也可以表示模块的层次结构。

2. 盒图使用的五种基本控制结构



(a) 顺序；(b) 选择；(c) CASE多分支；(d) 循环；(e) 调用子程序A

3. 盒图示例


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

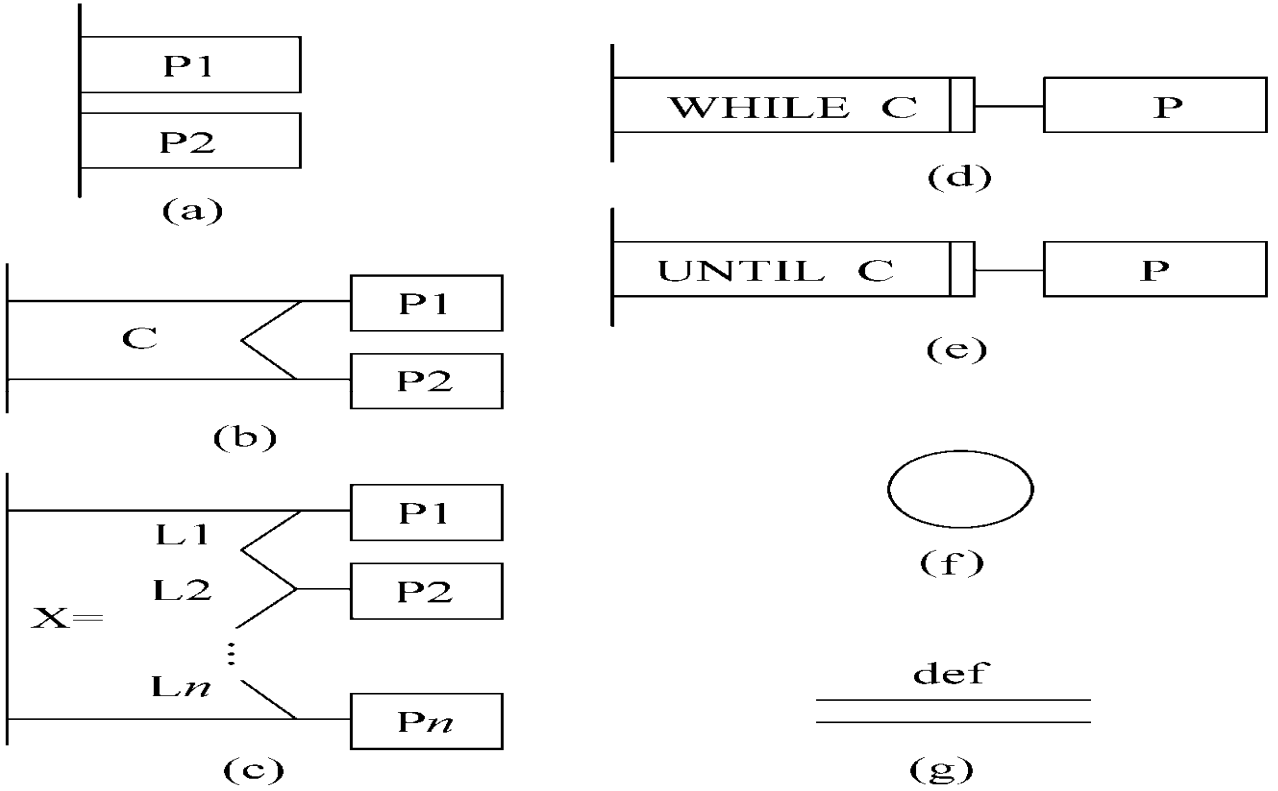
三、PAD图（Problem Analysis Diagram）

1.PAD图

它用二维树形结构的图来表示程序的控制流，将这种图翻译成程序代码比较容易。

它即克服了传统的流程图不能清晰表现程序结构的缺点，又不像N-S图那样受到把全部程序约束在一个方框内的限制。

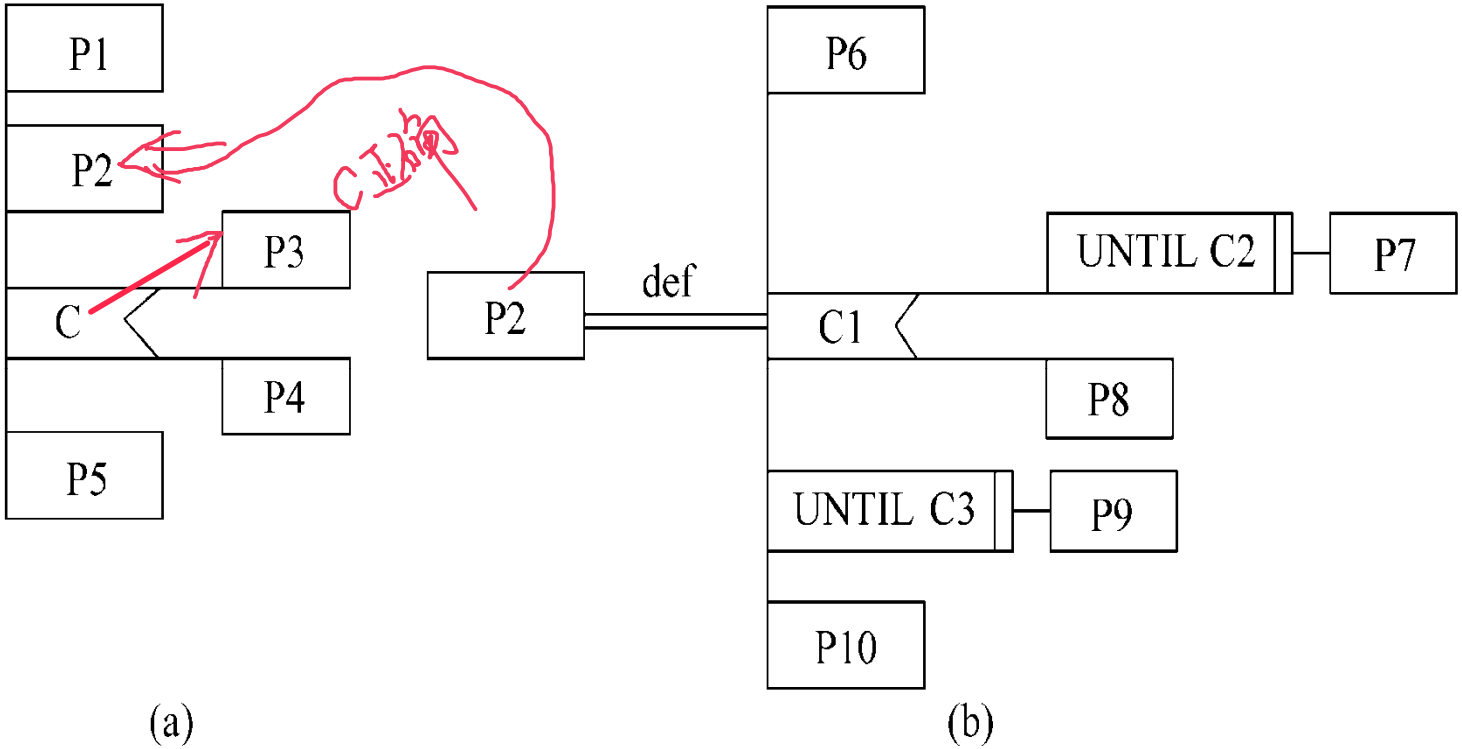
2. PAD图的基本符号



(a) 顺序；(b) 选择；(c) CASE多分支；(d) WHILE型循环；(e) UNTIL型循；

(f) 语句标号；(g) 定义

3. 使用PAD图提供的定义功能来逐步求精的例子



(a) 初始的PAD图；(b) 使用def符号细化处理框P2

4. PAD图主要优点

（1）使用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清晰，执行逻辑明确

- 最左面竖线是程序主线，程序每增加一个层次，图形向右扩展一条竖线；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。

（3）容易将PAD图转换成高级语言源程序，可用软件工具完成

（4）可用于表示程序逻辑，也可用于描绘数据结构

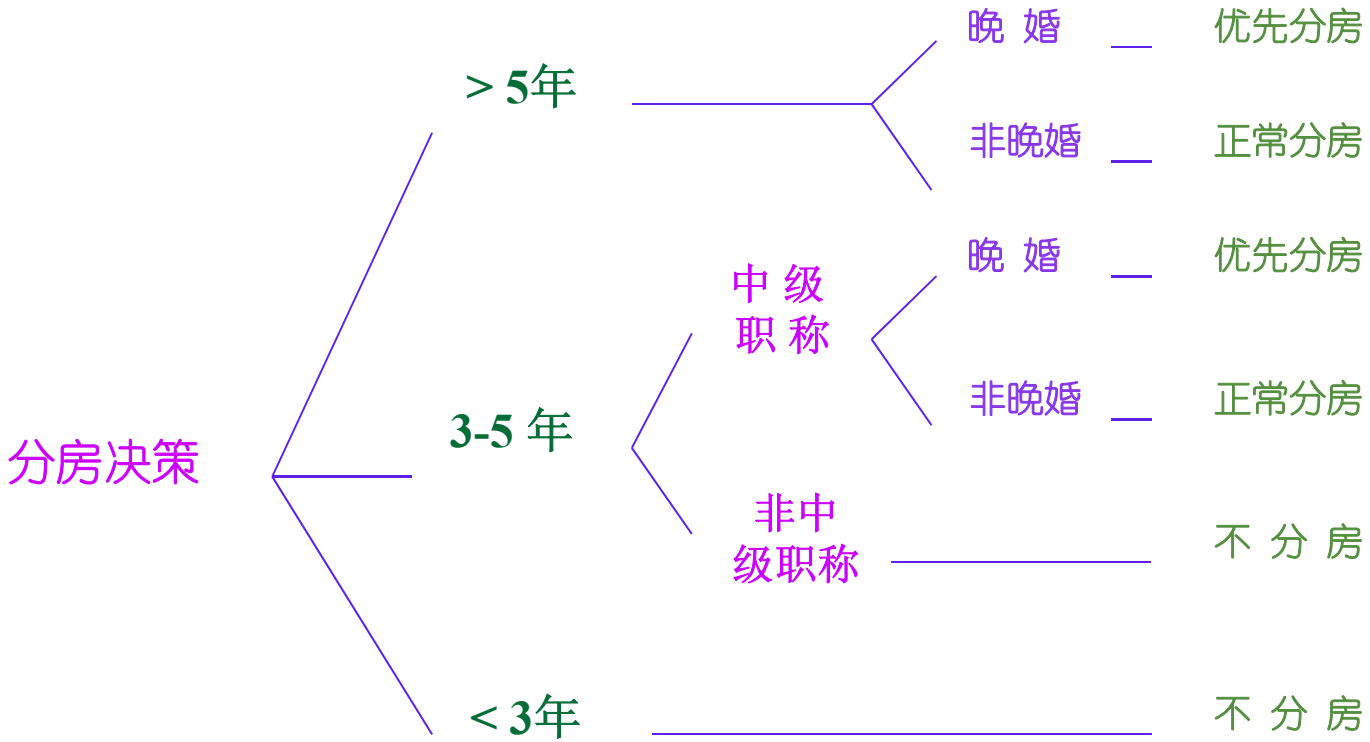
（5） PAD图的符号支持自顶向下、逐步求精方法的使用

- 使用def符号逐步增加细节，直至完成详细设计。

四、判定表

		1	2	3	4	5	6	说　明
条件	婚 龄	>5年		3-5年			<3年	(-) 表示任意 (Y) 条件满足 (N) 条件不满足 (*) 选中的决策
	中级职称	----------		Y		N	-----	
	晚 婚	Y	N	Y	N	----	-----	
决策	优先分房	*		*				
	正常分房		*		*			
	不分房					*	*	

五、判定树


分房决策
＞5年
3-5 年
＜3年
中 级
职 称
非中
级职称
晚 婚
非晚婚
晚 婚
非晚婚
优先分房
正常分房
优先分房
正常分房
不 分 房
不 分 房

六、PDL（Program Design Language）

1. PDL

- PDL是一种伪码，用正文的形式描述功能模块的算法设计和加工细节。
- PDL具有严格的关键字外语法，用于定义控制结构和数据结构；同时，表示实际操作和条件的内语法又是灵活、自由的，可使用自然语言的词汇。

2. 特点

- 优势：可以作为注释直接插在源程序中间；已有自动处理程序实现PDL向源程序的转换。
- 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，不如判定表清晰简单。

第四节　向数据结构的设计方法

一、基本思想

面向数据结构的设计是由英国人M. A. Jackson首先提出和倡导的，又称为Jackson方法(简称JSD)。

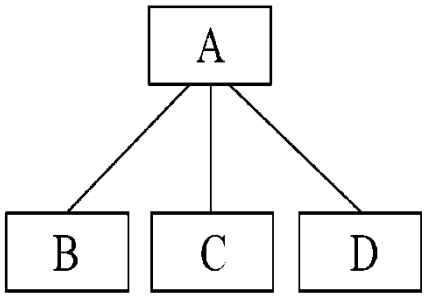
该方法的基本思想是使程序结构与问题结构相对应，即与数据结构相对应。因此它按输入输出以及内部存储信息的数据结构进行设计，把数据结构的描述变换为对程序结构的描述。

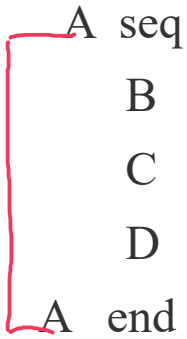
二、Jackson图

1. Jackson图

虽然程序中实际使用的数据结构种类繁多，但是它们的数据元素彼此间的逻辑关系却只有顺序、选择和重复3类，因此，逻辑数据结构也只有这3类。

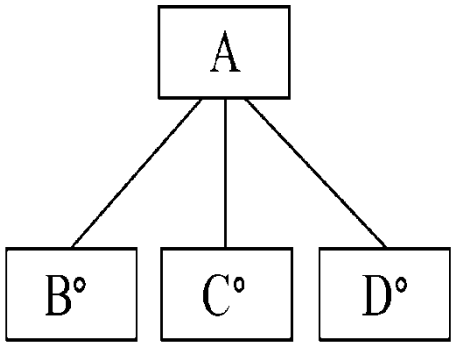
（1）顺序结构





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。

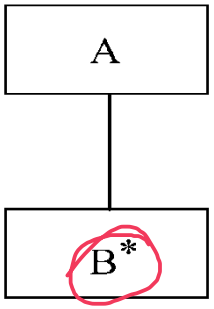
（2）分支结构




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

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



A iter until（或while）条件

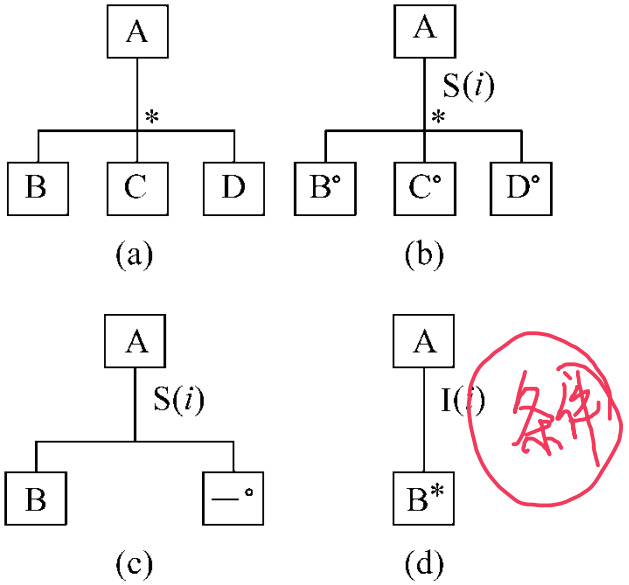
B

A end

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。

2. 改进的Jackson图

- 顺序结构，B、C、D中任一个都不能是选择出现和重复出现的数据元素；
- 加上判断条件
- 斜线改直线



三、Jackson方法

(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是重复的次序和次数都相同，才可能有对应关系）。

(3)用下列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：

- 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和在程序结构图的相应层次画一个处理框（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）。
- 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。
- 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。

（4）列出所有操作和条件（包括选择条件和重复结束条件），并把它们分配到程序结构图的适当位置。

（5）用伪码表示程序。

四、Jackson方法设计实例

问题陈述

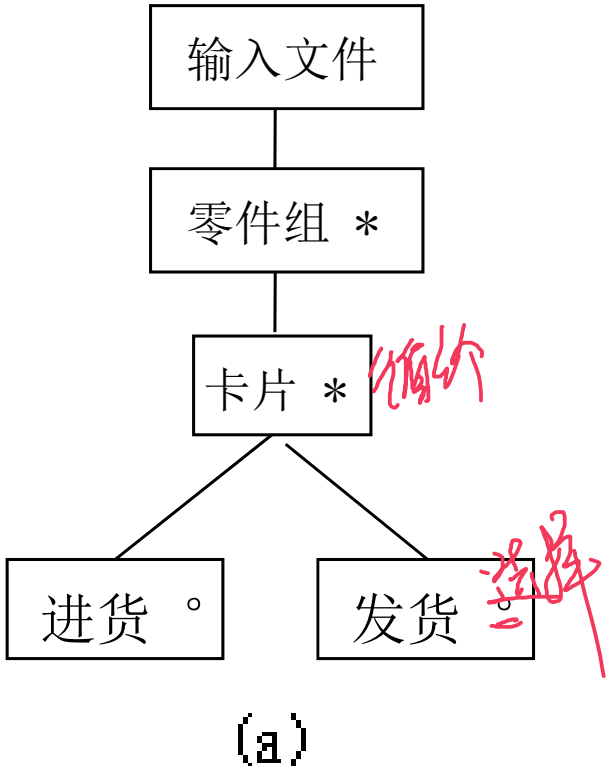
某仓库存放多种零件（如P1，P2，……），每个零件的每次进货、发货都有一张卡片作出记录，每月根据这样一叠卡片打印一张月报表。报表每行列出某种零件本月库存量的净变化。用JSD方法对该问题进行设计。

四、Jackson方法设计实例

(1)建立输入、输出数据结构

输入数据：根据问题陈述，同一种零件的进货、 发货状态不同，每月登记有若干张卡片。把同一种零件的卡片放在一起组成一组，所有的卡片组按零件名排序。所以输入数据是由许多零件组组成的文件，每个零件组有许多张卡片，每张卡片上记录着本零件进货或发货的信息。

输入数据结构的Jackson图如右图（a）。



(a)

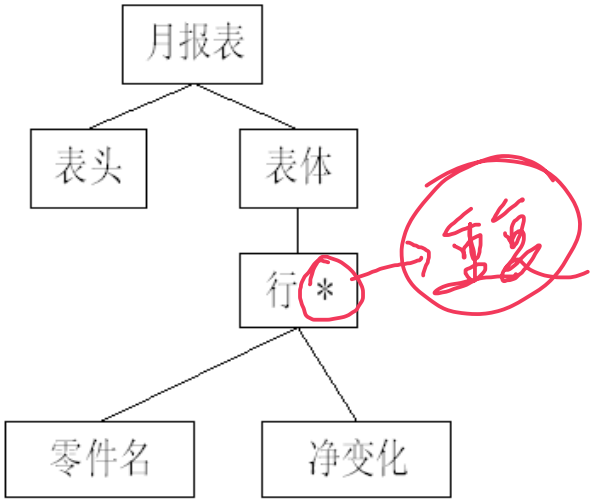
四、Jackson方法设计实例

输出数据：根据问题陈述，输出数据是一张如图(c）的月报表，它由表头和表体两部分组成，表体中有许多行，一个零件的净变化占一行。

其输出数据结构的Jackson图为图(b)。

月报表	
零件名	增加数量
P1	+200
P2	−100
P3	+1500
⋮	

(c)

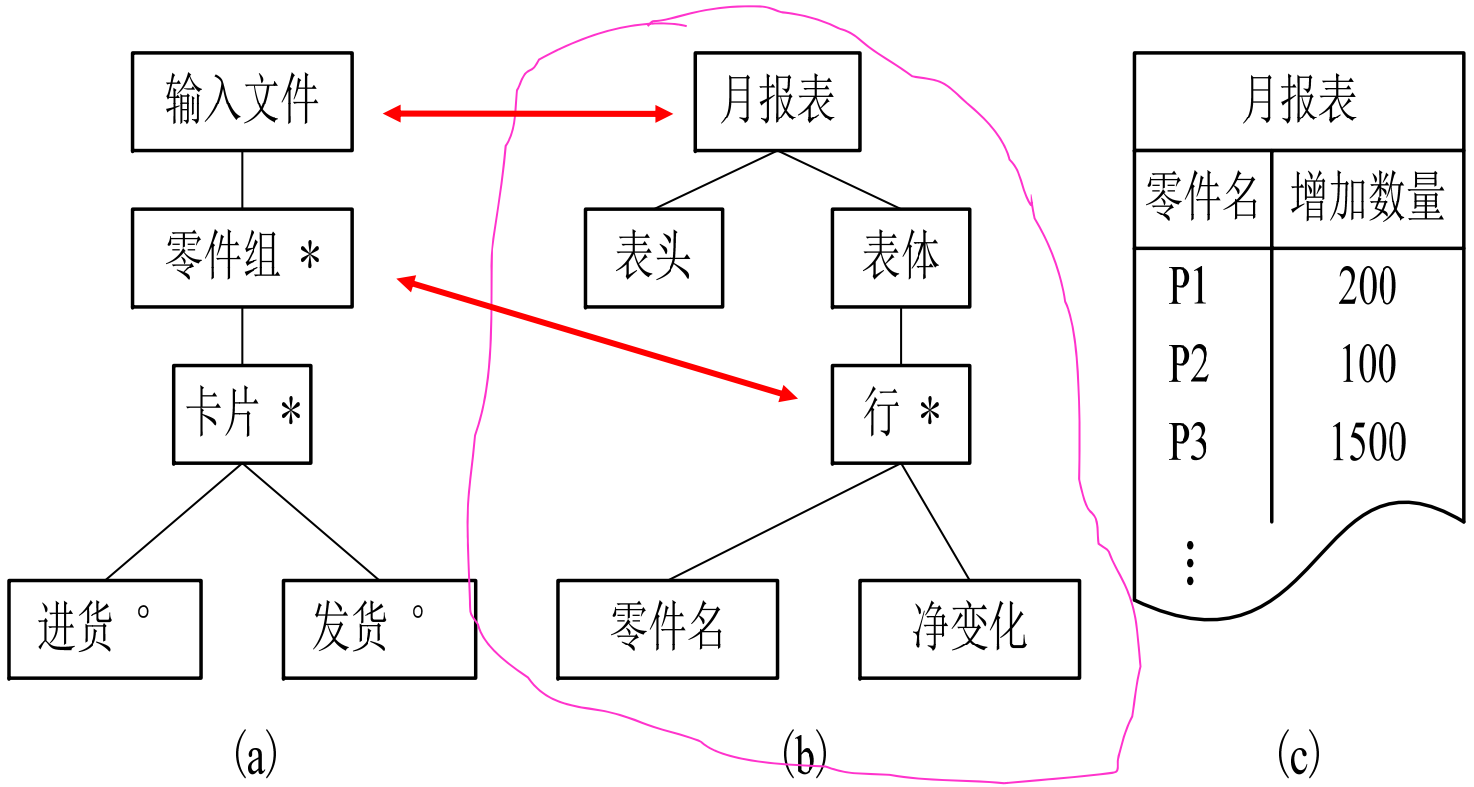


(b)

四、Jackson方法设计实例

(2)找出输入、输出数据结构中有对应关系的单元

- 月报表由输入文件产生，有直接的因果关系， 因此顶层的数据单元是对应的。
- 表体的每一行数据由输入文件的每一个“零件组”计算而来，行数与组数相同，且行的排列次序与组的排列次序一致，都按零件号排序，因此“零件组”与“行”两个单元对应。
- 以下再无对应的单元。



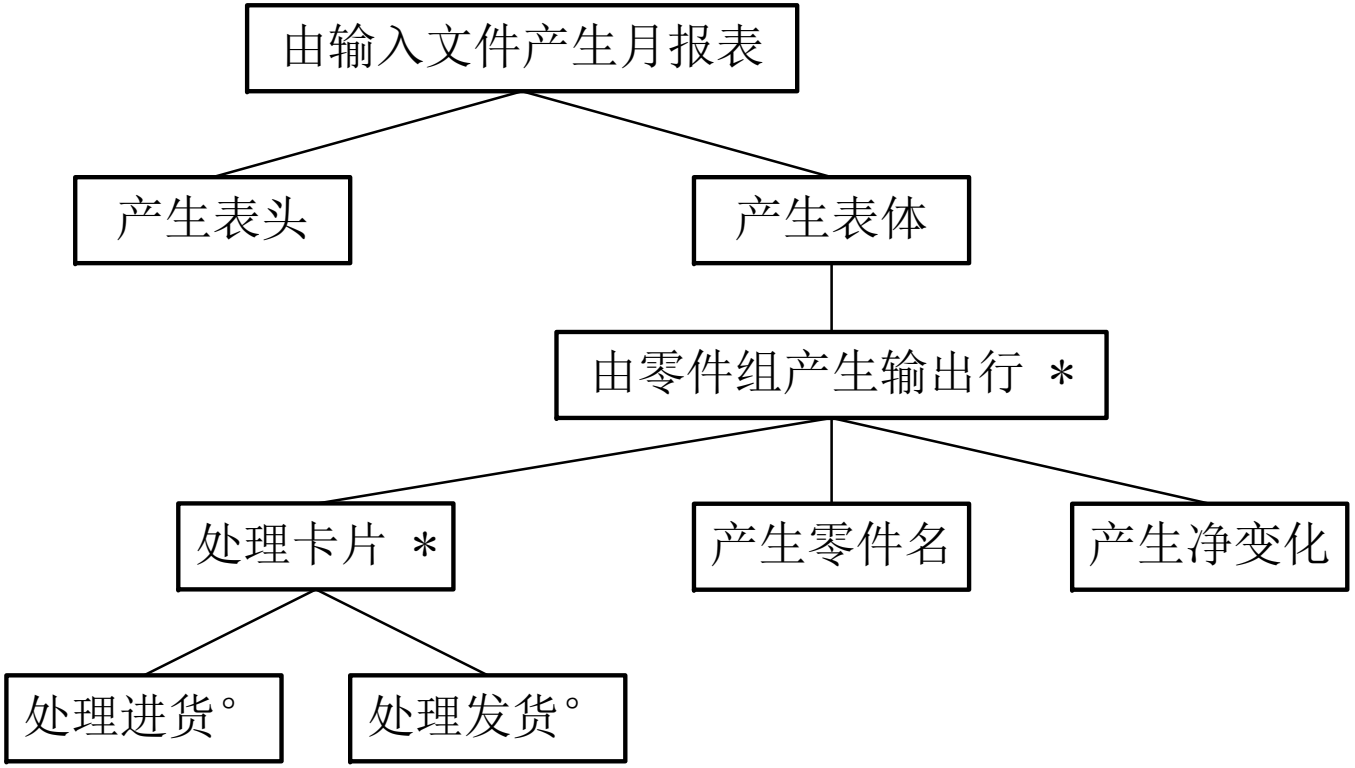
图：某仓库系统输入、 输出数据结构

（a）输入数据结构； （b）输出数据结构； （c）输出表

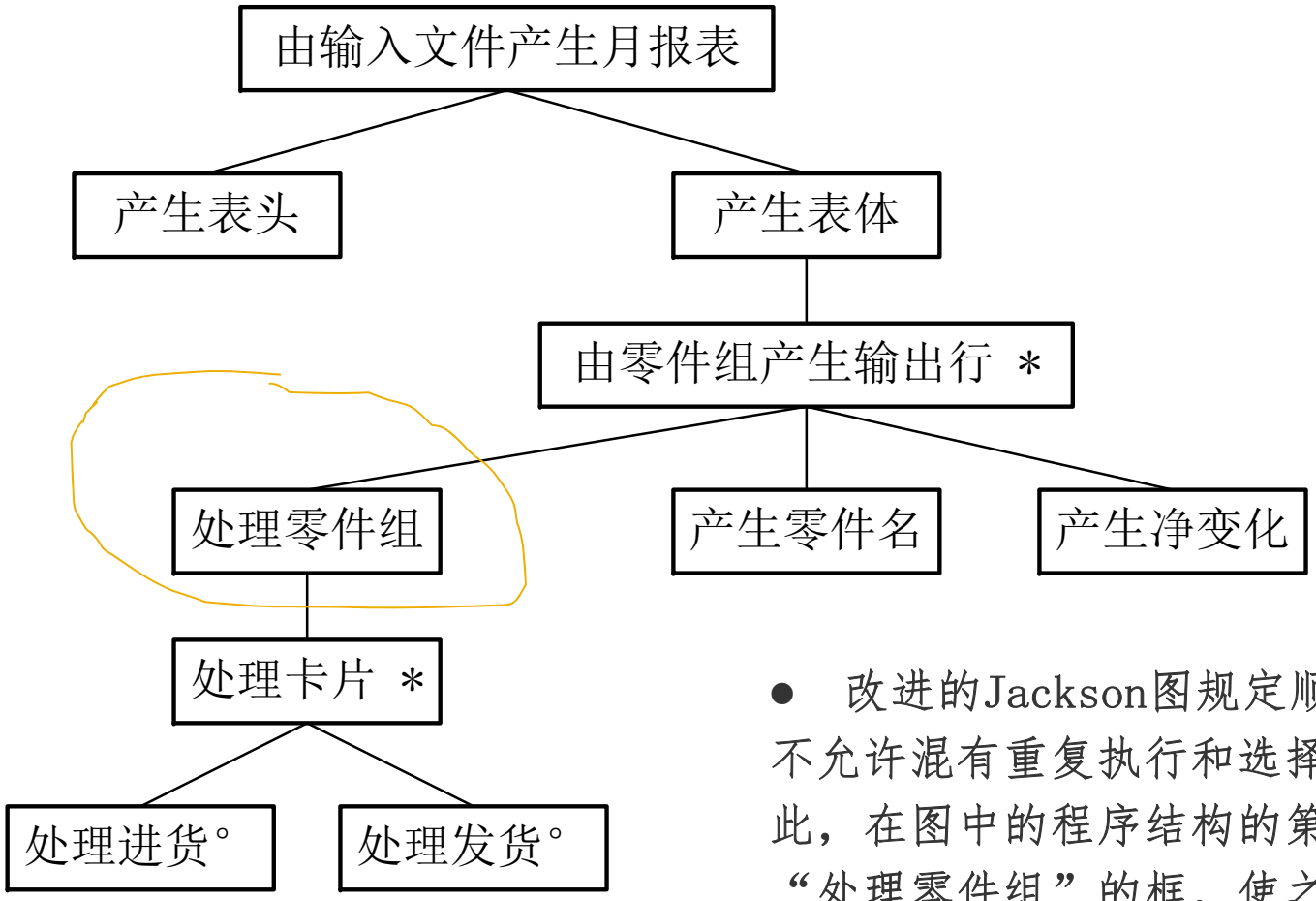
四、Jackson方法设计实例

(3)导出程序结构

- 找出对应关系后，根据以下规则导出程序结构：对于输入数据结构与输出数据结构中的数据单元，每对有对应关系的数据单元按照它们所在的层次，在程序结构图适当位置画一个处理框，无对应关系的数据单元，各画一个处理框。
- 根据以上规则，画出的程序结构图如图所示。



某仓库系统程序结构图



- 改进的Jackson图规定顺序执行的处理中不允许混有重复执行和选择执行的处理。因此，在图中的程序结构的第4层增加了一个“处理零件组”的框，使之符合该规定，同时也提高了结构图的易读性。

四、Jackson方法设计实例

(4)列出并分配操作与条件

为了对程序结构作补充，要列出求解问题的所有操作和条件，然后分配到程序结构图的适当位置， 就可得到完整的程序结构图。

- 本问题的基本操作列出如下：

A：停止　　B： 打开文件　　C：关闭文件　　D：打印字符行

E：读一张卡　　F：产生行结束符　G：累计进货量　　H：累计发货量

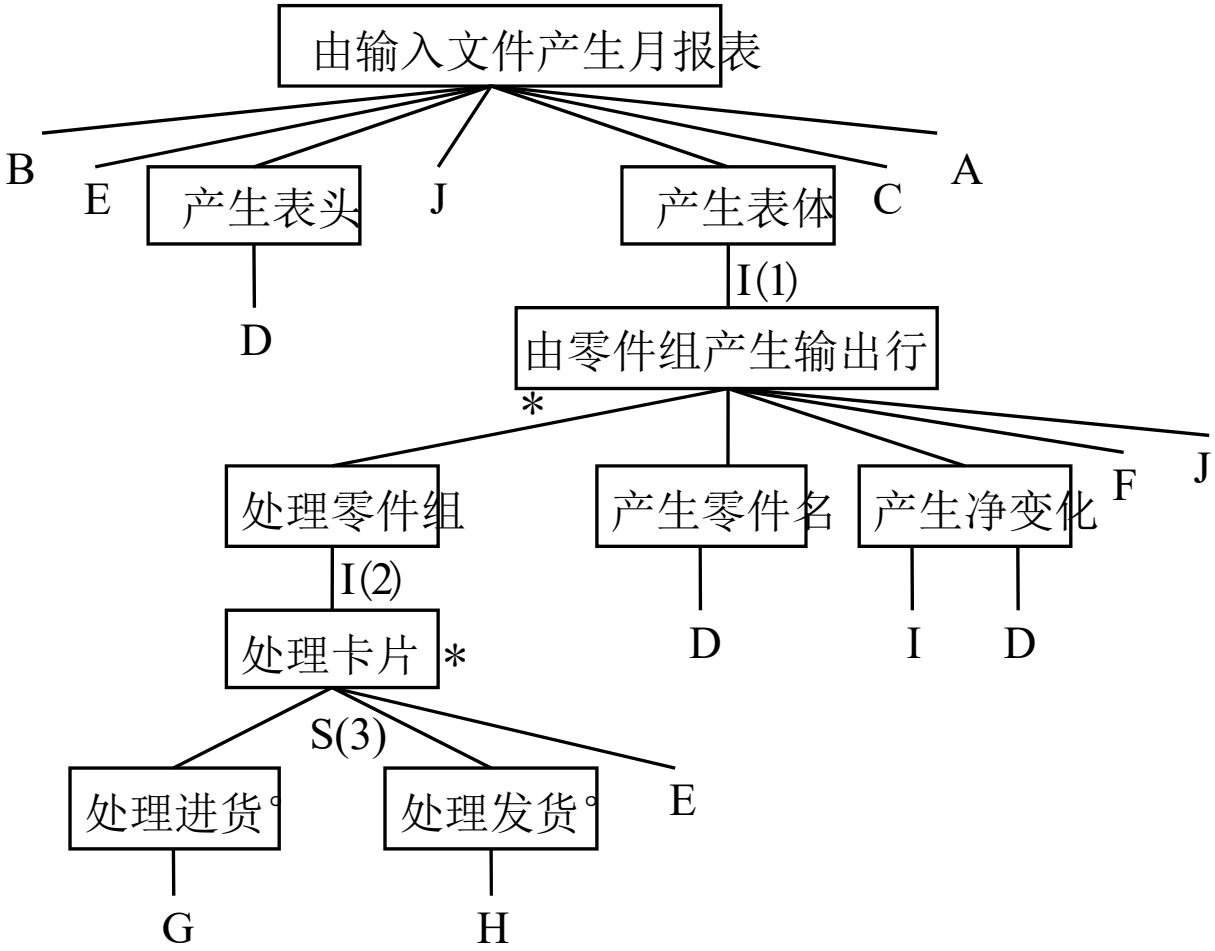
I：计算净变化　　J：置零件组开始标志

- 列出条件如下：

I（1）：输入文件未结束　I（2）：零件组未结束　　S（3）：进发货标志

将操作与条件分配到适当位置的程序结构图如下图所示。

分配操作后的程序结构图



B：打开文件
C：关闭文件
D：打印字符行
E：读一张卡
F：产生行结束符
G：累计进货量
H：累计发货量
I：计算净变化
J：置零件组开始标志
I(1)输入文件未结束
I(2)零件组未结束
S(3):进发货标志
A：停止

四、Jackson方法设计实例

在分配操作时注意：为了能获得重复和选择的条件，Jackson建议至少超前读一个记录，以便使得程序不论在什么时候判定，总有数据已经读入，并做好使用准备。因此在图中，将操作E（读一张卡）放在打开文件之后，同时在处理完一张卡片后再读一次。

(5)用伪码写出程序

Jackson方法中的伪码与Jackson所示的程序结构图完全对应，用伪码写出程序的过程，实际上就是自顶向下用这些伪码替换Jackson图中每个处理框的过程，每个处理框都看作是下层处理框及分配在上面的操作组成。

四、Jackson方法设计实例

```
产生月报表 seq
  打开文件
  读一张卡
  产生表头 seq
    打印字符行
  产生表头 end
  置零件组开始标志
  产生表体 iter while 输入文件未结束
    由零件组产生输出行 seq
      处理零件组 iter while 零件组未结束
        处理卡片 select 进货标志
          处理进货 seq
            累计进货量
          处理进货 end
        处理卡片 or 发货标志
          处理发货 seq
            累计发货量
          处理发货 end
        处理卡片 end
        读一张卡
      处理零件组 end
    产生零件名 seq
      打印字符行
    产生零件名 end
    产生净变化 seq
      计算净变化
      打印字符行
    产生净变化 end
    换行
    置零件组开始标志
  由零件组产生输出行end
  产生表体 end
  关闭文件
  停止
产生月报表 end
```

五、Jackson方法小结

- 优点：简单，适合于规模不大的系统，建立了问题的数据结构之后，可直接推导出相应的程序结构。
- 局限性：当输入数据结构与输出数据结构不相同且无对应关系时，难于应用。当数据结构发生变化时，程序结构也发生变化，一般是总体上用SD方法，局部范围用JSD方法。
- SD方法和JSD方法的异同

面向数据流的设计（SD方法）和面向数据结构的设计（JSD方法）的共同点都是数据信息驱动的，都试图将数据表示转换成软件表示；不同之处在于面向数据结构的设计不利用数据流图，而根据数据结构的表示来设计。

程序流程图：控制流

第五节　程序复杂程度的定量度量

一、代码行度量法

统计一个程序的源代码行数，以源程序行数作为程序复杂程度的量度，源程序代码行数越大认为越复杂。

源代码行数度量法基于两个前提：

- 程序复杂性随着程序规模的增加而均衡地增长；
- 控制程序规模的方法最好是采用分而治之的办法。将一个大程序分解成若干个简单的可理解的程序段。

二、McCabe度量法

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，这样度量出的结果称为程序的环形复杂度。

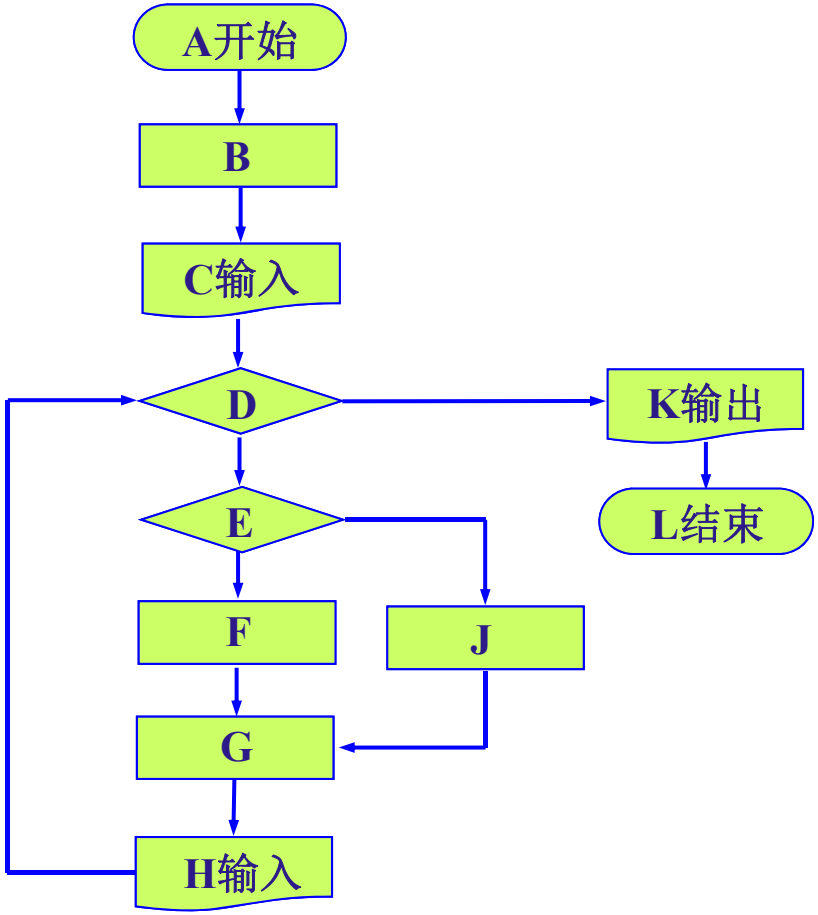
1. 流图

为了突出表示程序的控制流，通常使用流图（也称为程序图）。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。

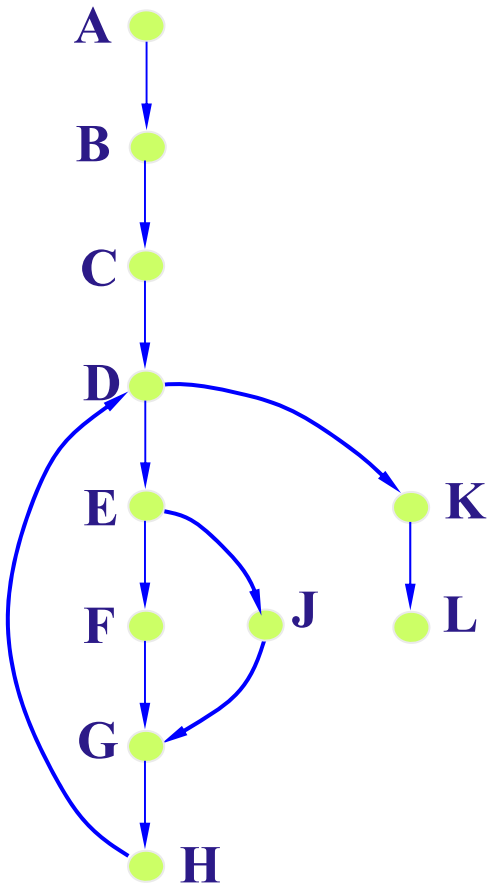
流图的表示：

- 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那个区域。

如下图举例说明把程序流程图映射成流图的方法。



程序流程图



流图

2. 使用流图计算环形复杂度的方法

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,其中，E是流图中有向边的条数，N是结点数。

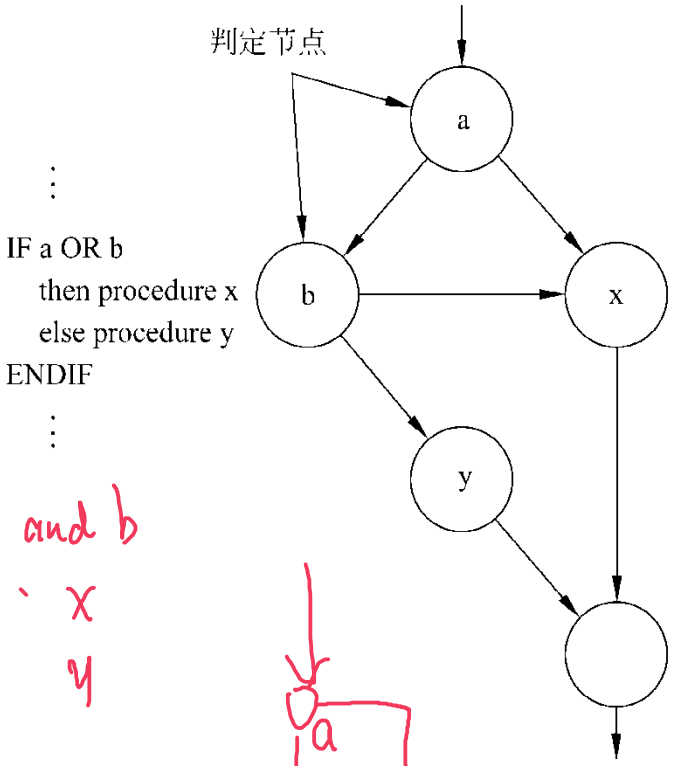
(3) 流图G的环形复杂度V(G)=P+1，其中，P是流图中判定结点的数目。

3. 应注意的问题

当过程设计中包含复合条件时，生成流图的方法稍微复杂一些。所谓复合条件，就是在条件中包含了一个或多个布尔运算符。

在这种情况下，应该把复合条件分解为若干个简单条件，每个简单条件对应流图中一个结点。包含条件的结点称为判定节点，从每个判定结点引出两条或多条边。

右图是由包含复合条件的PDL片断翻译成的流图。



由包含复合条件的PDL映射成的流图

4. 环形复杂度的用途

- 它是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。
- McCabe研究大量程序后发现，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，也就是说，V(G)=10是模块规模的一个更科学更精确的上限。

三、Halstead方法

采用程序中运算符和操作数的总数来度量程序复杂程度。

(1) 实际的 Halstead 长度:

N=N1+N2

N1：为运算符出现的总次数

N2：为操作数出现的总次数

- 在定义中，运算符包括：
  算术运算符、赋值符(=或:=)、逻辑运算符、分界符(，或；或:) 、关系运算符、括号运算符、子程序调用符、数组操作符、循环操作符等；
- 特别地，成对的运算符，例如
  begin…end、if…then…else、for…to、repeat …until、while…do、（…）等都当做单一运算符。
- 运算对象包括变量名和常数。

(2)预测的 Halstead 长度公式：

$$H = n1*\log_2 n1 + n2*\log_2 n2$$

n1：为运算符的数量（不同类型）

n2：为操作数的数量（不同类型）

H：为预测程序长度。

(3)实践验证表明，预测长度与实际长度非常接近。

示例程序

```
      SUBROUTINE SORT ( X, N )
      DIMENSION  X( N )
      IF ( N .LT. 2 ) RETURN
      DO  20  I=2, N
          DO  10  J=1, I
            IF ( X(I) .GE. X(J) ) GO TO 10
            SAVE = X(I)
            X(I) = X(J)
            X(J) = SAVE
10        CONTINUE
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SORT ( X, N )
      DIMENSION  X( N )
      IF ( N .LT. 2 ) RETURN
      DO  20  I=2, N
          DO  10  J=1, I
            IF ( X(I) .GE. X(J) ) GO TO 10
            SAVE = X(I)
            X(I) = X(J)
            X(J) = SAVE
10        CONTINUE
20    CONTINUE
      RETURN
      END
```

运算符	计数	运算对象	计数
可执行语句结束	7	X	6
数组下标	6	I	5
=	5	J	4
IF（ ）	2	N	2
DO	2	2	2
,	2	SAVE	2
程序结束	1	1	1
.LT.	1	$n2=7$	$N2=22$
.GE.	1		
GO TO 10	1		
$n1=10$	$N1=28$		

- $H = 10 * \log_2 10 + 7 * \log_2 7 = 52.87$
- $N = 28 + 22 = 50$


Thank
You