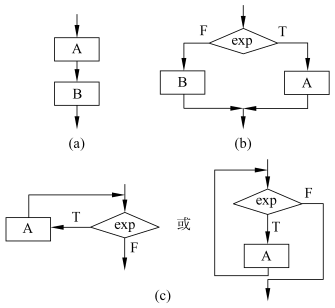
软件工程
第二部分 结构化方法学
详细设计

5.1 详细设计概述

- 详细设计的根本目标是确定应该怎样具体地实现所要求的系统。
- 详细设计阶段的任务是要设计出程序的“蓝图”，以后程序员将根据这个蓝图写出实际的程序代码。因此，详细设计的结果基本上决定了最终的程序代码的质量。
- 过程设计和人机界面设计是详细设计阶段的主要任务，而结构设计、接口设计、数据设计在总体设计阶段应基本完成。

5.2 结构化程序设计

- 结构化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Corrado Bohm和Giuseppe Jacopini证明了，只用“顺序”、“选择”和“循环” 3种基本的控制结构就能实现任何单入口单出口的程序。
- Bohm和Jacopini的证明给结构程序设计技术奠定了理论基础。理论上最基本的控制结构只有两种（顺序和循环）。



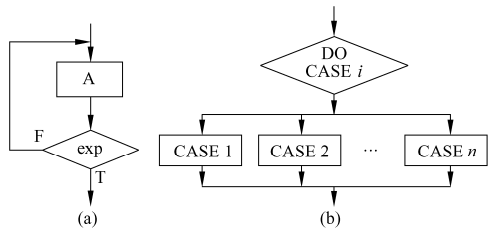
3种基本的控制结构

- 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。争论导致了新的程序设计思想、方法和风格的出现。
- 1971年IBM公司在“纽约时报信息库管理系统”和 “美国宇航局空间实验室飞行模拟系统”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，从而补充了结构程序设计的规则。

- 结构化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。”
- 更全面的结构化程序设计的定义

  “结构化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。”



其他常用的控制结构

5.3 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。
- 近年来，人机界面在系统中所占的比例越来越大，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(如游戏软件)。
- 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。

- 应注重的几个设计问题:
  - ✓系统响应时间：系统响应时间不宜过长或过短。
  - ✓用户帮助设施：必须提供充分的帮助信息。
  - ✓出错信息处理：出现问题时交互式系统必须给出适当的出错信息和警告信息。这些信息应该易于理解、有助于用户从错误中恢复。
  - ✓命令交互：用户既可以从菜单中选择软件功能，也可以通过命令行序列调用软件功能。

- 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。可以采用用户界面工具或用户界面开发系统设计与实现用户界面。
  - ✓ 用户界面设计的评估标准：
    - ➢ 用户界面的规格说明书的长度和复杂程度
    - ➢ 命令或动作的数量、命令的平均参数个数
    - ➢ 界面风格、帮助设施和出错处理协议等

- 设计指南：一般性指南
  - ✓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
  - ✓允许犯错误
  - ✓提高对话、移动和思考效率
  - ✓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

- 设计指南：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
  - ✓ 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
  - ✓ 高效率地使用显示屏

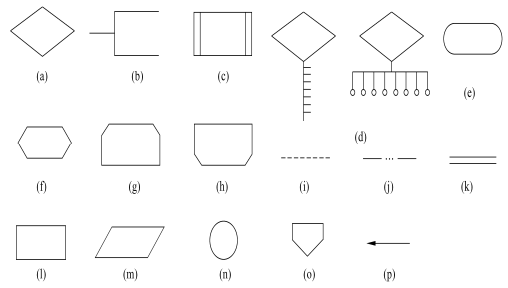
- 设计指南：数据输入指南
  - ✓尽量减少用户的输入动作
  - ✓保持信息显示和数据输入之间的一致性
  - ✓允许用户自定义输入
  - ✓交互应该是灵活的
  - ✓使在当前动作语境中不适用的命令不起作用
  - ✓让用户控制交互流
  - ✓对所有输入动作都提供帮助
  - ✓消除冗余的输入

5.4 过程设计工具

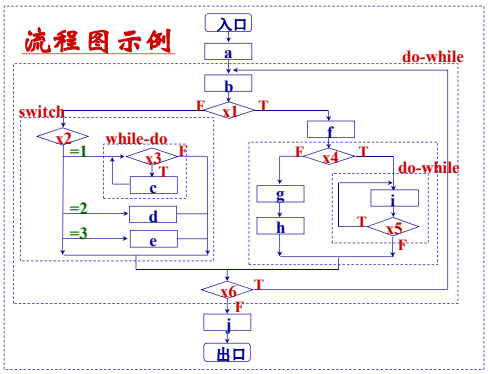
- 在过程设计阶段，要决定各个模块的实现算法，并精确地表达这些算法。描述程序处理过程的工具称为过程设计工具。
- 过程设计工具分为如下三类：
  - ✓ 图形工具：程序流程图，N-S ，PAD
  - ✓ 表格工具：判定表, 判定树
  - ✓ 语言工具： PDL

1、程序流程图

- 程序流程图又称为程序框图，程序控制流程图。历史最悠久、使用最广泛的描述过程设计的方法，用得最混乱的一种方法。
  - ✓ 主要优点:对控制流程的描绘很直观，便于初学者掌握。
  - ✓ 主要缺点：不是逐步求精的好工具、不易表示数据结构、对转移控制缺乏约束。

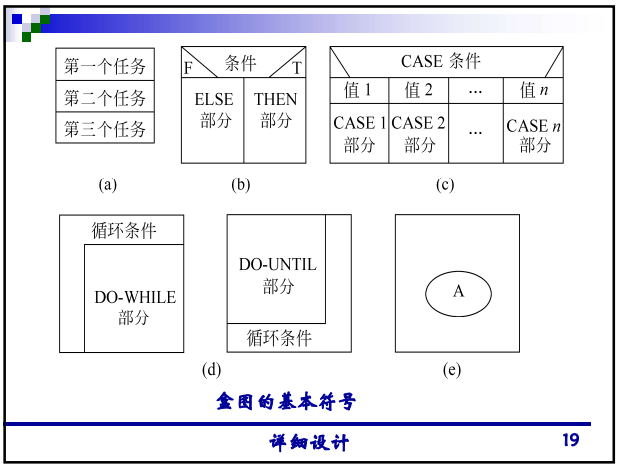


程序流程图中使用的符号

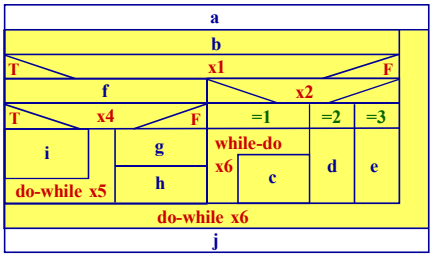


2、N-S 图

- N-S图又称盒图，是 Nassi和Shneiderman出于要有一种不允许违背结构化程序设计精神的图形工具的考虑提出的。
- N-S图的主要特点:
  - ✓ 功能域明确，可以从盒图上一眼就看出来
  - ✓ 很容易确定局部和全程数据的作用域
  - ✓ 很容易表现嵌套关系和模块的层次结构
  - ✓ 不可能任意转移控制



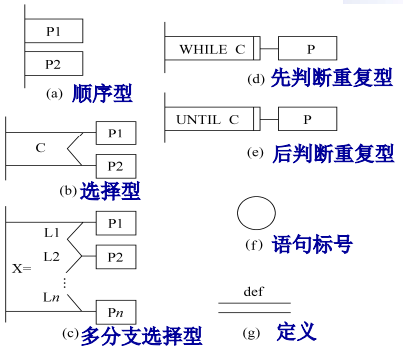
盒图的基本符号



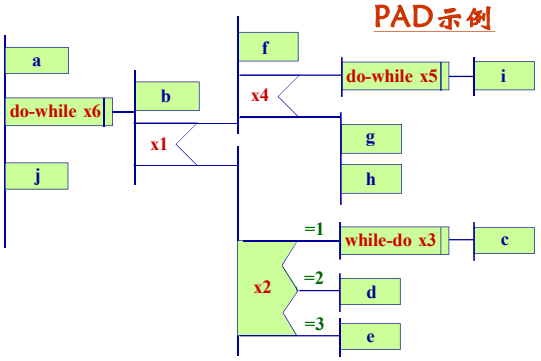
N-S图（盒图）例子

3、PAD

- PAD (Problem Analysis Diagram)又称问题分析图。1973年由日本日立公司发明，用二维树形结构的图来表示程序的控制流。
- PAD图的主要优点：
  - ✓将这种图翻译成程序代码比较容易
  - ✓用PAD符号所设计出来的程序必然是结构化程序
  - ✓支持逐步求精的使用
  - ✓描述的程序结构清晰
  - ✓可以表示程序逻辑，也可以用于描绘数据结构



PAD图的基本符号



4、判定表

- 判定表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：左上部列出所有条件，左下部是所有可能做的动作，右上部是表示各种条件组合的一个矩阵，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。
- 判定表能够简洁而又无歧义地描述处理规则。把判定表和布尔代数或卡诺图结合起来使用时，可以对判定表进行校验或化简。

表 6.1　用判定表表示计算行李费的算法

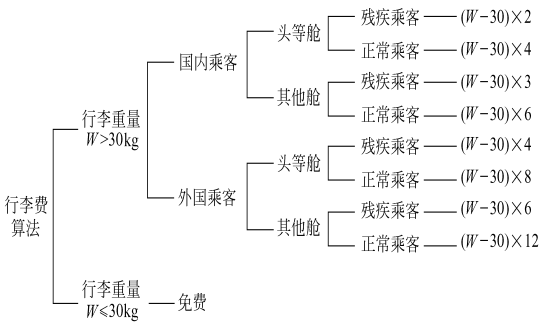
		规则								
		1	2	3	4	5	6	7	8	9
条件	国内乘客		T	T	T	T	F	F	F	F
	头等舱		T	F	T	F	T	F	T	F
	残疾乘客		F	F	T	T	F	F	T	T
	行李重量 $W\leqslant 30$kg	T	F	F	F	F	F	F	F	F
可能动作	免费	×								
	$(W-30)\times 2$				×					
	$(W-30)\times 3$					×				
	$(W-30)\times 4$		×						×	
	$(W-30)\times 6$			×						×
	$(W-30)\times 8$						×	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

条件组合

条件组合对应的动作

5、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。判定树是一种比较常用的系统分析和设计的工具。
- 判定树的优点: 形式简单, 表达直观，易于掌握和使用。
- 判定树的缺点：不够简洁，对应于同一判定表的判定树可能不唯一。



与表6.1等价的判定树

6、PDL

- PDL (Program Design Language)是一种用于描述功能模块的算法设计和加工细节的语言, 也称为设计程序用语言或伪码。
- 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；表示实际操作和条件的内语法，可使用自然语言的词汇。
- PDL的优点在于易于书写与编辑、易于与源程序融合，缺点是不够形象直观。

PDL例子: 拼词检查程序

```
PROCEDURE spellcheck
BEGIN
   --* split document into single words
   LOOP get next word
      add word to word list in sortorder
      EXIT  WHEN all words processed
   END  LOOP
   --* look up words in dictionary
   LOOP get word from word list
```



```
      IF word not in dictionary THEN
        --* display words not in dictionary
        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
```

- 为了区别关键字，规定关键字一律大写，其它单词一律小写。
- 内语法使用自然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。

5.5 面向数据结构的设计方法

- 为什么面向数据结构的设计方法?

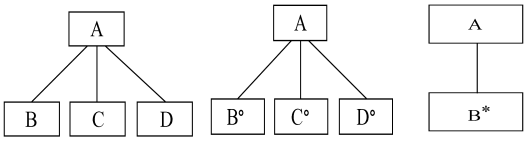
  在许多应用领域中信息都有清楚的层次结构，输入数据、内部存储的信息以及输出数据都可能有独特的结构。

  数据结构既影响程序的结构又影响程序的处理过程。例如：重复出现的数据通常由具有循环控制结构的程序来处理，选择数据要用带有分支控制结构的程序来处理。

- 面向数据结构的设计方法：
  - ✓Jackson方法和Warnier方法
  - ✓面向数据结构的设计方法的最终目标是得出对程序处理过程的描述。它并不明显地使用软件结构的概念，模块是设计过程的副产品，对于模块独立原理也没有给予应有的重视。这种方法最适合于在详细设计阶段使用。
  - ✓使用面向数据结构的设计方法，首先需要分析确定数据结构，并且用适当的工具清晰地描绘数据结构。

Jackson图

- Jackson图是对层次方框图的一种精华。刻画了数据元素彼此间的逻辑关系，提供了顺序、选择和重复三类逻辑数据结构。



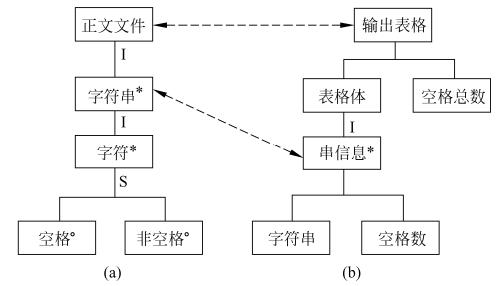
A由B、C、D3个元素顺序组成　　A是B或C或D中的某一个　　A由B出现N次(N≥0)组成

- Jackson图具有便于表示层次结构、可读性好、即可表示数据结构和也可表示程序结构等优点。
- Jackson图和层次图形式相似，含义不相：
  - ✓层次图表现的是调用关系，一个模块除了调用下级模块外，还完成其他操作；Jackson图表现的是组成关系，方框中包括的操作仅仅由它下层框中的那些操作组成。
  - ✓层次图中的方框通常代表一个模块；Jackson图即使在描绘程序结构时，一个方框也并不代表一个模块，通常一个方框只代表几个语句。

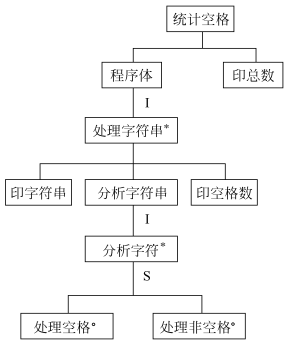
Jackson结构程序设计方法

- Jackson结构程序设计方法主要步骤：
  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。
  3. 用相应的规则从描绘数据结构的Jackson图导出描绘程序结构的Jackson图。
  4. 列出所有操作和条件，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。

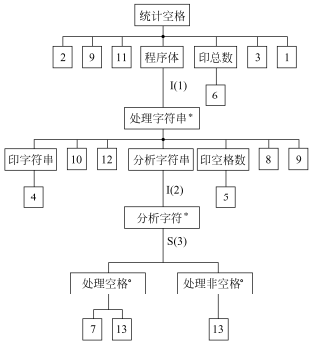
- Jackson结构程序设计方法在设计比较简单的数据处理系统时特别方便，当设计比较复杂的程序时常常遇到输入数据可能有错、条件不能预先测试、数据结构冲突等问题。
- 正文文件字符统计的例子。



输入数据结构Jackson图　　输出数据结构Jackson图



程序结构的Jackson图



把操作和条件分配后程序结构的Jackson图

5.6 程序复杂性度量

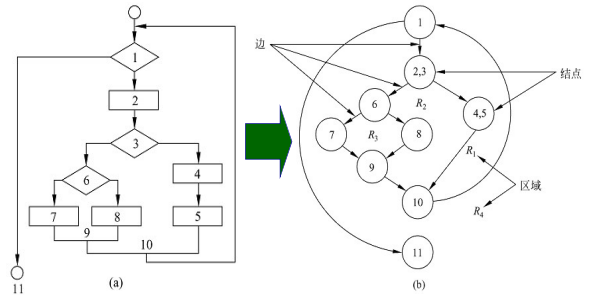
- 遵循软件设计的基本原理和概念，经过详细设计之后每个模块的内容都非常具体了。那么有没有办法评价设计出的模块质量呢？特别是定量地度量软件的性质？

- 定量度量程序复杂程度的意义：
  - ✓ 程序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。
- Macabe方法和Halstead方法是比较成熟的程序复杂程度定量度量方法。

1、McCabe方法

- 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，度量结果称为程序的环形复杂度。
- 环形复杂度取决于程序控制流的复杂程度。当程序内分支数或循环个数增加时，环形复杂度也随之增加，因此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。

- 流图实质上是“退化了的”程序流程图。用任何方法表示的过程设计结果，都可以翻译成流图。
- 由程序流程图翻译成流图的一些规则：
  - 流图中的圆表示结点，一个圆代表一条或多条语句。
  - 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。
  - 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。
  - 由边和结点围成的面积称为区域。
  - 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。

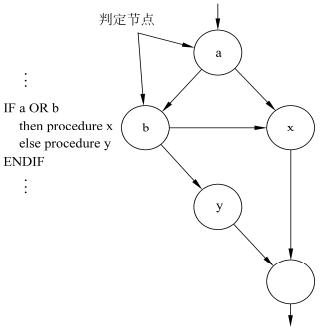


程序流程图转化为流图例子

- 过程设计中某些条件中包含了一个或多个布尔运算符。

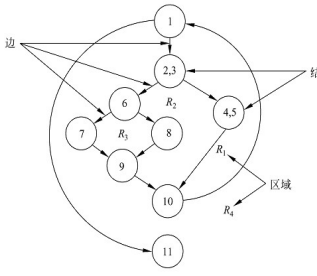
生成流图时，应该把复合条件分解为若干个简单条件，每个简单条件对应流图中一个结点。

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。



环形复杂度的计算方法

- 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。
  - ✓流图中的区域数等于环形复杂度。
  - ✓流图G的环形复杂度V(G)=E-N+2,其中，E是流图中边的条数，N是结点数。
  - ✓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。



区域: R1,R2,R3,R4

边数: 11

结点数:9

判定结点: 1, 2, 6

流图中的区域数 4

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

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

2、Halstead方法

- Halstead方法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。
  - ✓ 令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，程序长度N定义为：

    $N = N_1 + N_2$
  - ✓ 程序中不同运算符(包括关键字)的个数n_1，不同操作数的个数n_2。Halstead给出预测程序长度的公式如下：

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

- ✓ 实践表明：预测长度H与实际长度N非常接近。
- ✓ Halstead还给出了预测程序中包含错误的个数的公式如下：

  $E = N \log_2 (n_1+n_2) /3000$
- ✓ 部分实践表明：发现预测的错误数与实际错误数相比误差在8%之内。
- ✓ Halstead无需结构分析，简单易行，可预测错误率以及 可预测维护工作量。

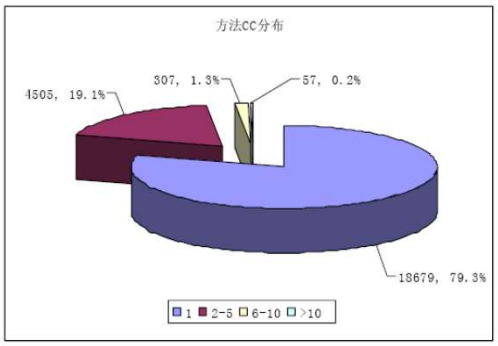


图 10、方法 CC 度量分布

小结

- 详细设计阶段的关键任务是确定系统实现的具体方案，设计出系统的蓝本，以便写出高质量的程序；为此，结构化程序设计技术是关键。
- 详细设计阶段的主要工作是过程设计和界面设计。界面设计是接口设计的一部分，应遵循相关原则与指南。Jackson方法是一种面向数据结构的程序设计技术，可以用于某些系统的过程设计。
- 过程设计工具分为图形、表格和语言，各有所长。
- Macabe方法和Halstead方法是两种比较成熟的程序复杂程度度量方法，在实践中有着广泛应用。