“十二五”普通高等教育本科国家级规划教材

北京高等教育精品教材

21世纪软件工程专业规划教材

软件工程导论（第6版）

第6章　详细设计

张海藩，牟永敏编著　　清华大学出版社

第6章　详细设计

根本目标：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不是具体地编写程序，而是要设计出程序的“蓝图”。

详细设计的结果基本上决定了最终的程序代码的质量。

主要内容

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

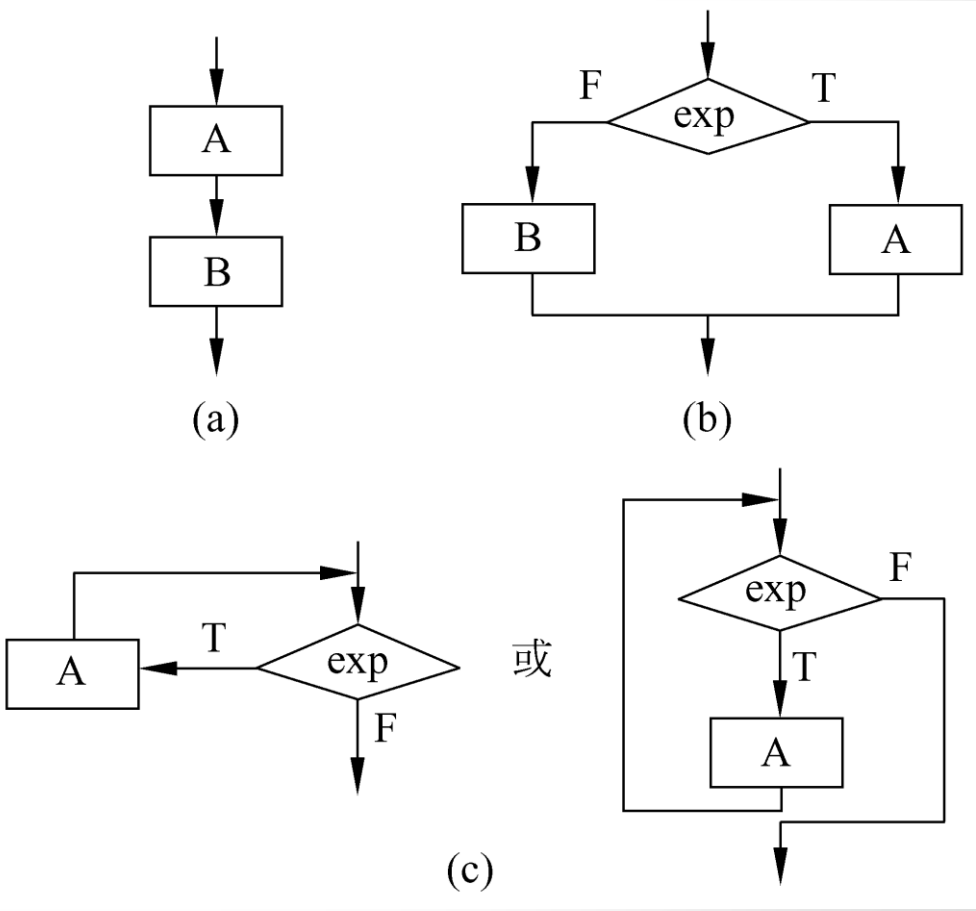
6.4 面向数据结构的设计方法

6.5 程序复杂程度的定量度量

主要内容

6.1 结构程序设计

- 1965年结构程序设计的概念最早由E.W.Dijkstra提出：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Bohm和Jacopini证明了只用“顺序”、“选择”和“循环”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

实际上用顺序结构和循环结构(又称DO-WHILE结构)完全可以实现选择结构(又称IF-THEN-ELSE结构），因此，理论上最基本的控制结构只有两种。

- 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种基本控制结构写程序。学界认识到不是简单地去掉GO TO 语句的问题，而是要创立一种新的程序设计思想、方法和风格。
- 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，补充了结构程序设计的规则。

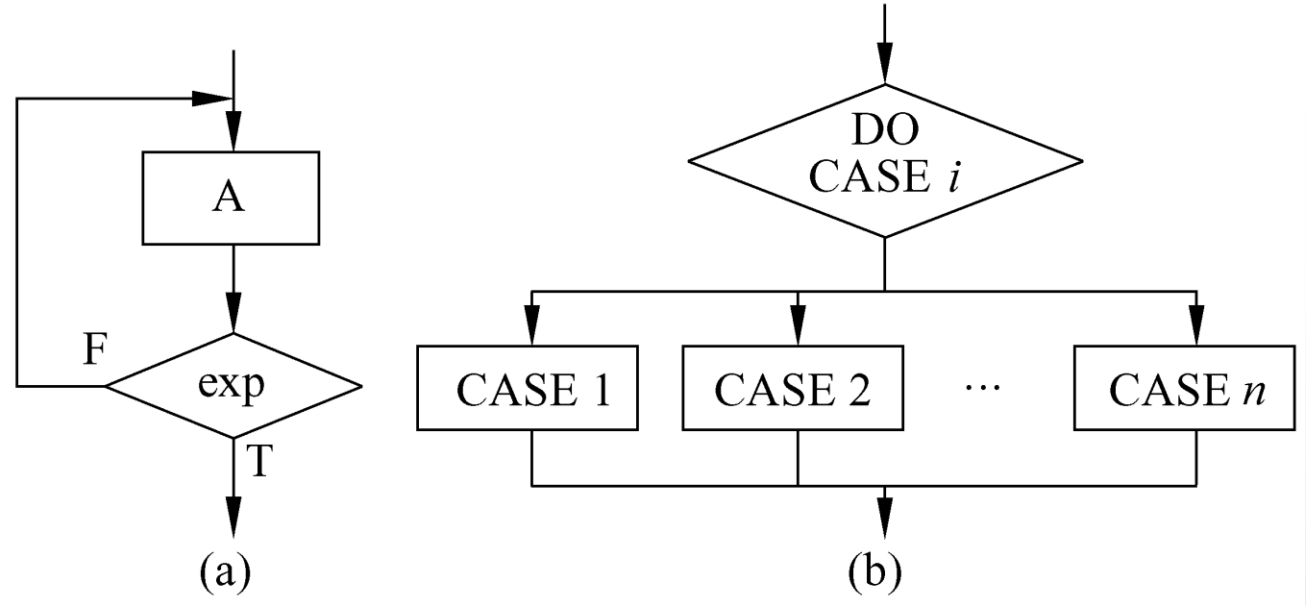
6.1 结构程序设计

结构程序设计经典定义：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。

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

从理论上说只用上述3种基本控制结构就可以实现任何单入口单出口的程序，但是为了实际使用方便起见，常常还允许使用DO-UNTIL和DO-CASE两种控制结构



6.1 结构程序设计

如果只允许使用顺序、IF-THEN-ELSE型分支和DO-WHILE型循环这3种基本控制结构，则称为经典的结构程序设计；

如果除了上述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，则称为扩展的结构程序设计；

如果再允许使用LEAVE(或BREAK)结构，则称为修正的结构程序设计。

主要内容

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

① 系统响应时间。

② 用户帮助设施。

③ 出错信息处理。

④ 命令交互。

6.2 人机界面设计

①系统响应时间。

系统响应时间指从用户完成某个控制动作(例如，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，分别是长度和易变性。

1）长度：时间过长，用户就会感到紧张，过短，加快用户操作节奏，可能会犯错误

2）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) 在用户与系统交互期间，是否在任何时候都能获得关于系统任何功能的帮助信息?有两种选择：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?有3种选择：帮助菜单，特殊功能键和HELP命令。

(3) 怎样显示帮助信息?有3种选择：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的交互方式中?有两种选择：屏幕上的返回按钮和功能键。

(5) 怎样组织帮助信息?有3种选择：平面结构，信息的层次结构和超文本结构。

6.2 人机界面设计

③ 出错信息处理。

出错信息和警告信息，是出现问题时交互式系统给出的“坏消息”。一般说来，交互式系统给出的出错信息或警告信息，具有下述属性。

(1) 用用户可以理解的术语描述问题。

(2) 提供有助于从错误中恢复的建设性意见。

(3) 指出错误可能导致哪些负面后果(例如，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。

(4) 伴随着听觉上或视觉上的提示

(5) 不能带有指责色彩，不能责怪用户。

6.2 人机界面设计

④ 命令交互。

许多高级用户仍然偏爱面向命令行的交互方式

在提供命令交互方式时，必须考虑下列设计问题。

(1) 是否每个菜单选项都有对应的命令?

(2) 采用何种命令形式?有3种选择：控制序列(例如，Ctrl+P)，功能键和输入命令。

(3) 学习和记忆命令的难度有多大?忘记了命令怎么办?

(4) 用户是否可以定制或缩写命令?

在越来越多的应用软件中，人机界面设计者都提供了“命令宏机制”。

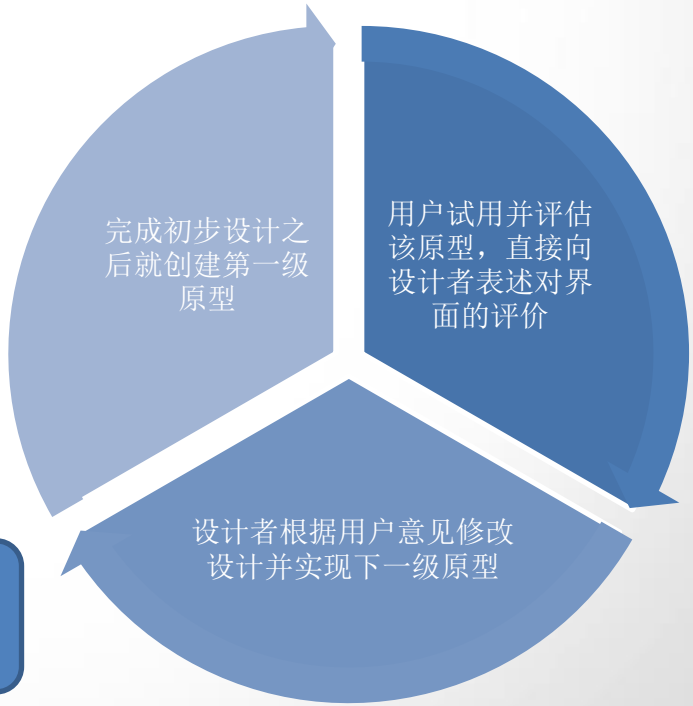
在理想的情况下，所有应用软件都有一致的命令使用方法。

6.2 人机界面设计

6.2.2.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。

建立起用户界面的原型，就必须对它进行评估，评估可以是非正式的也可以使正式的。



用户界面的评估周期

6.2 人机界面设计

创建了用户界面的设计模型之后，可以运用下述评估标准对设计进行早期复审。

(1) 系统及其界面的规格说明书的长度和复杂程度，预示了用户学习使用该系统所需要的工作量。

(2) 命令或动作的数量、命令的平均参数个数或动作中单个操作的个数，预示了系统的交互时间和总体效率。

(3) 设计模型中包含的动作、命令和系统状态的数量，预示了用户学习使用该系统时需要记忆的内容的多少。

(4) 界面风格、帮助设施和出错处理协议，预示了界面的复杂程度及用户接受该界面的程度。

6.2 人机界面设计

6.2.3 人机界面设计指南

① **一般交互指南：**涉及信息显示、数据输入和系统整体控制

(1)保持一致性。
(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。
(5)减少在两次操作之间必须记忆的信息量。
(6)提高对话、移动和思考的效率。
(7)允许犯错误。
(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
(10)用简单动词或动词短语作为命令名。

6.2 人机界面设计

② **信息显示指南：**多种不同方式“显示”信息：用文字、图形和声音；按位置、移动和大小；使用颜色、分辨率和省略。

(1)只显示与当前工作内容有关的信息。
(2)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。
(3)使用一致的标记、标准的缩写和可预知的颜色。
(4)允许用户保持可视化的语境。
(5)产生有意义的出错信息。
(6)使用大小写、缩进和文本分组以帮助理解。
(7)使用窗口分隔不同类型的信息。
(8)使用“模拟”显示方式表示信息，以使信息更容易被用户提取。
(9) 高效率地使用显示屏。

6.2 人机界面设计

③ **数据输入指南**：用户的大部分时间用在选择命令、输入数据和向系统提供输入。

(1)尽量减少用户的输入动作。

(2)保持信息显示和数据输入之间的一致性。

(3)允许用户自定义输入。

(4)交互应该是灵活的，并且可调整成用户最喜欢的输入方式。

(5)使在当前动作语境中不适用的命令不起作用。

(6)让用户控制交互流。

(7)对所有输入动作都提供帮助。

(8)消除冗余的输入。

(9) 高效率地使用显示屏。

主要内容

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

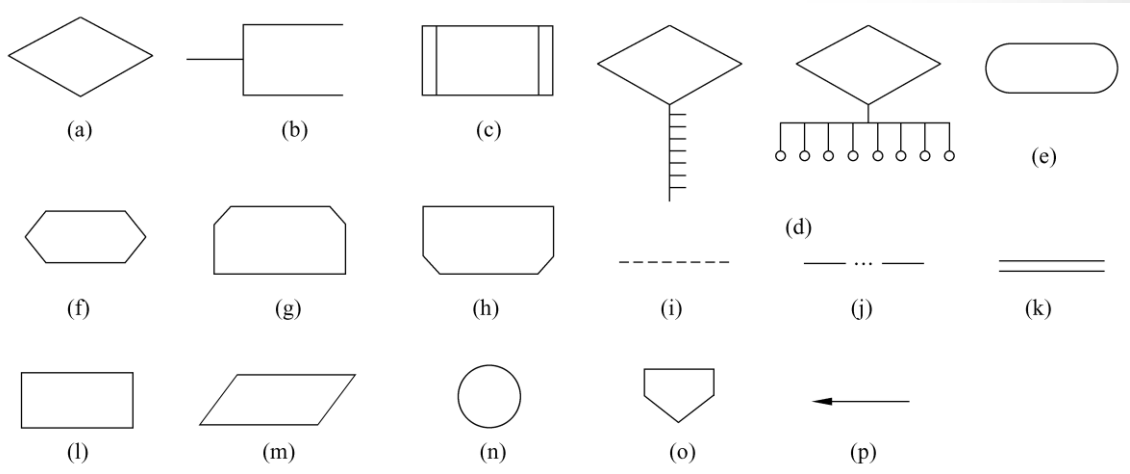
6.4 面向数据结构的设计方法

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

程序流程图又称为程序框图，它是使用最广泛的描述过程设计的方法。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



6.3 过程设计的工具

总的趋势是越来越多的人不再使用程序流程图了。

程序流程图的主要缺点如下。

(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。

6.3 过程设计的工具

6.3.2 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。

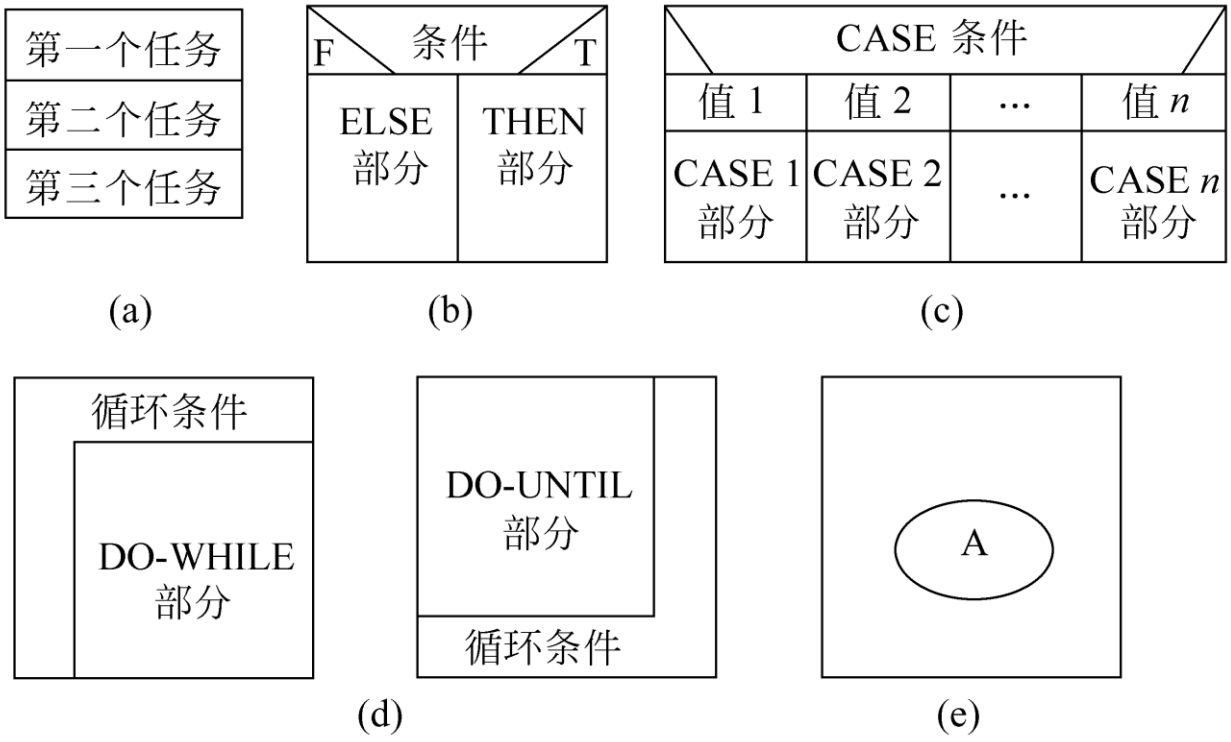
(3) 很容易确定局部和全程数据的作用域。

(4) 很容易表现嵌套关系，也可以表示模块的层次结构。

6.3 过程设计的工具

图给出了结构化控制结构的盒图表示，也给出了调用子程序的盒图表示方法。其中

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



6.3 过程设计的工具

6.3.3 PAD图

PAD是问题分析图(problem analysis diagram)的英文缩写，用二维树形结构的图来表示程序的控制流。基本符号

(a) 顺序(先执行P1后执行P2)；

(b) 选择(IF C THEN P1 ELSE P2)；

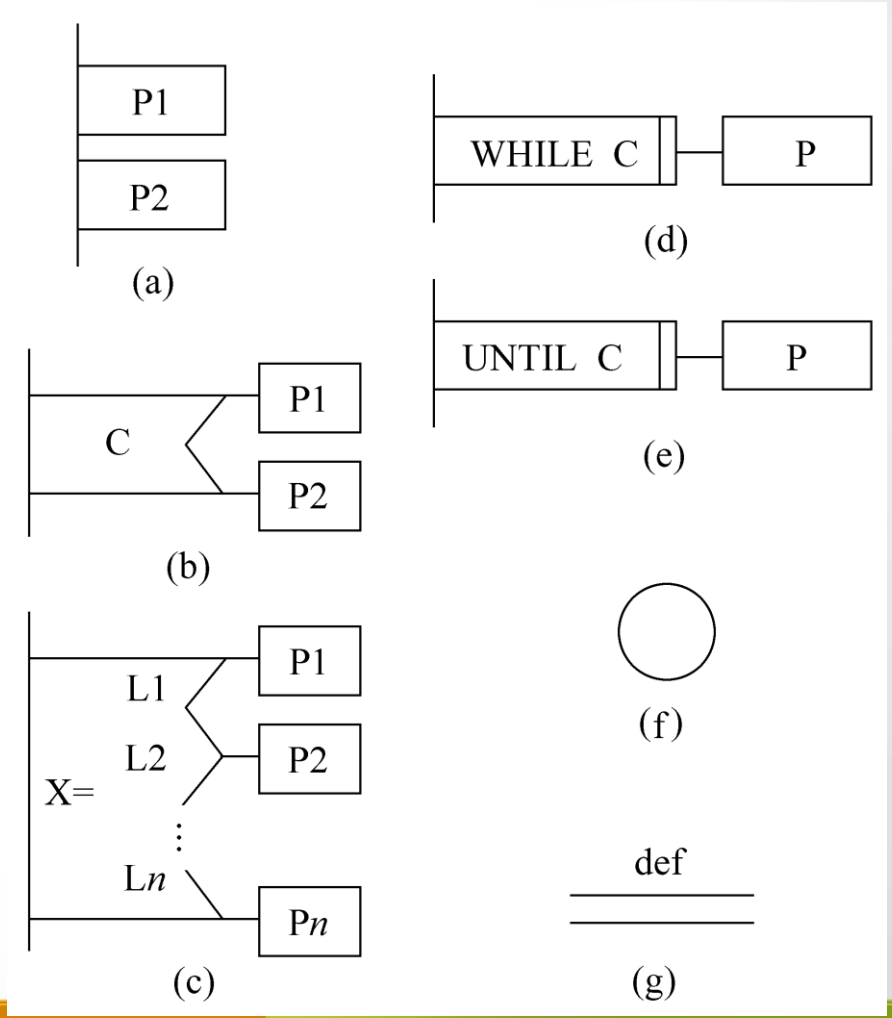
(c) CASE型多分支；

(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



6.3 过程设计的工具

PAD图的主要优点如下：

(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图表现程序逻辑，易读、易懂、易记。

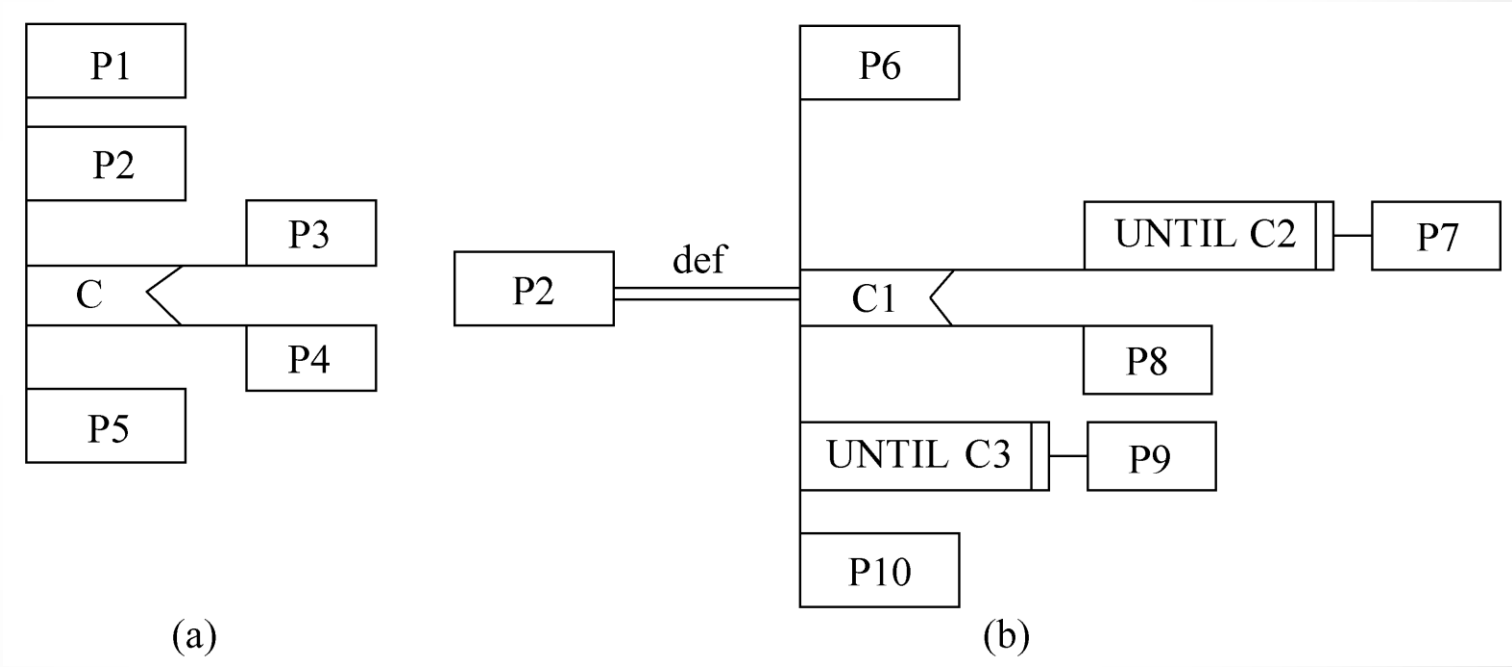
(4) 容易将PAD图转换成高级语言源程序，这种转换可 用软件工具自动完成，从而可省去人工编码的工作，有利于提高软件可靠性和软件生产率。

(5) 即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、逐步求精方法的使用。

6.3 过程设计的工具

例子



(a) 初始的PAD图； (b) 使用def符号细化处理框P2

6.3 过程设计的工具

6.3.4 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。

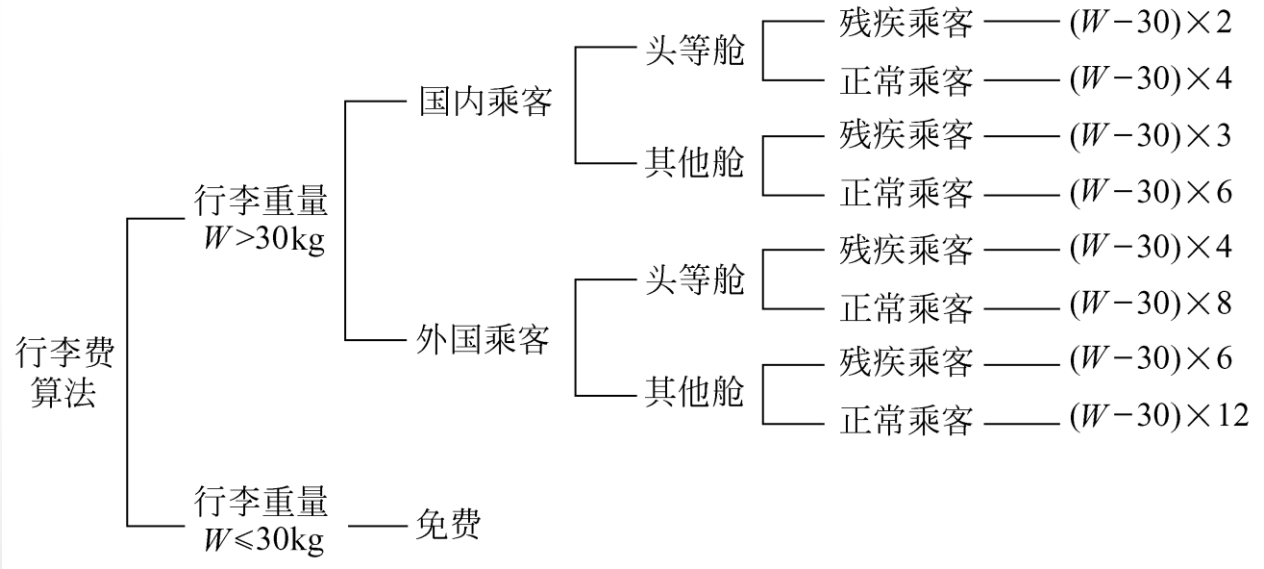
6.3 过程设计的工具

	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≤30kg	T	F	F	F	F	F	F	F	F
免费	X								
(W-30)X2				X					
(W-30)X3					X				
(W-30)X4		X						X	
(W-30)X6			X						X
(W-30)X8						X			
(W-30)X12							X		

6.3 过程设计的工具

6.3.5 判定树

判定树是判定表的变种，它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。



6.3 过程设计的工具

6.3.6 过程设计语言

过程设计语言（PDL）也称为伪码。是用正文形式表示数据和处理过程的设计工具。

PDL有下述特点：

(1)关键字的固定语法，它提供了结构化控制结构、数据说明和模块化的特点。如，if…fi(或endif)等

(2) 自然语言的自由语法，它描述处理特点。

(3) 数据说明的手段。应该既包括简单的数据结构(例如纯量和数组)，又包括复杂的数据结构(例如，链表或层次的数据结构)。

(4) 模块定义和调用的技术，应该提供各种接口描述模式。

6.3 过程设计的工具

PDL有下述优点：

(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，很方便地完成PDL的书写和编辑工作。

(3) 已经有自动处理PDL的程序存在，而且可以自动由PDL生成程序代码。

PDL的缺点是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时，不如判定表清晰简单。

主要内容

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

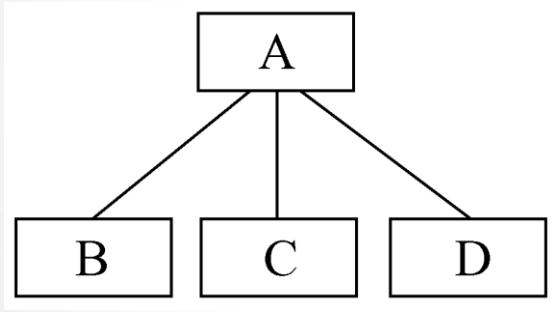
6.4 面向数据结构的设计方法

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

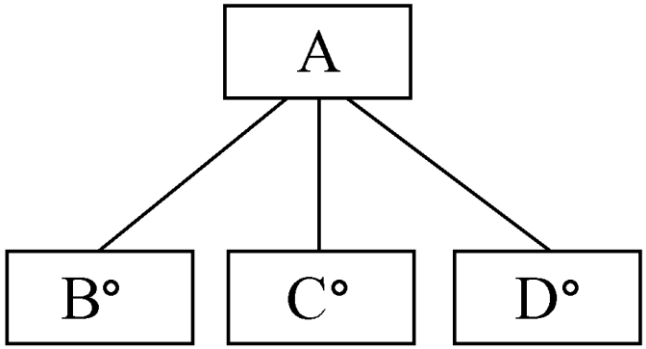
6.4.1 Jackson图

顺序结构



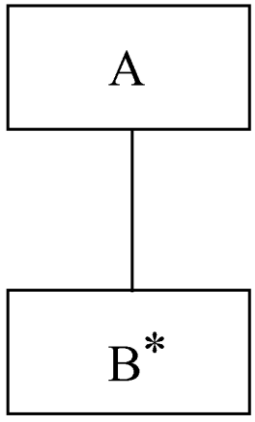
A由B、C、D 3个元素顺序组成(每个元素只出现一次，出现的次序依次是B、C和D)

选择结构



根据条件A是B或C或D中的某一个(注意，在B、C和D的右上角有小圆圈做标记)

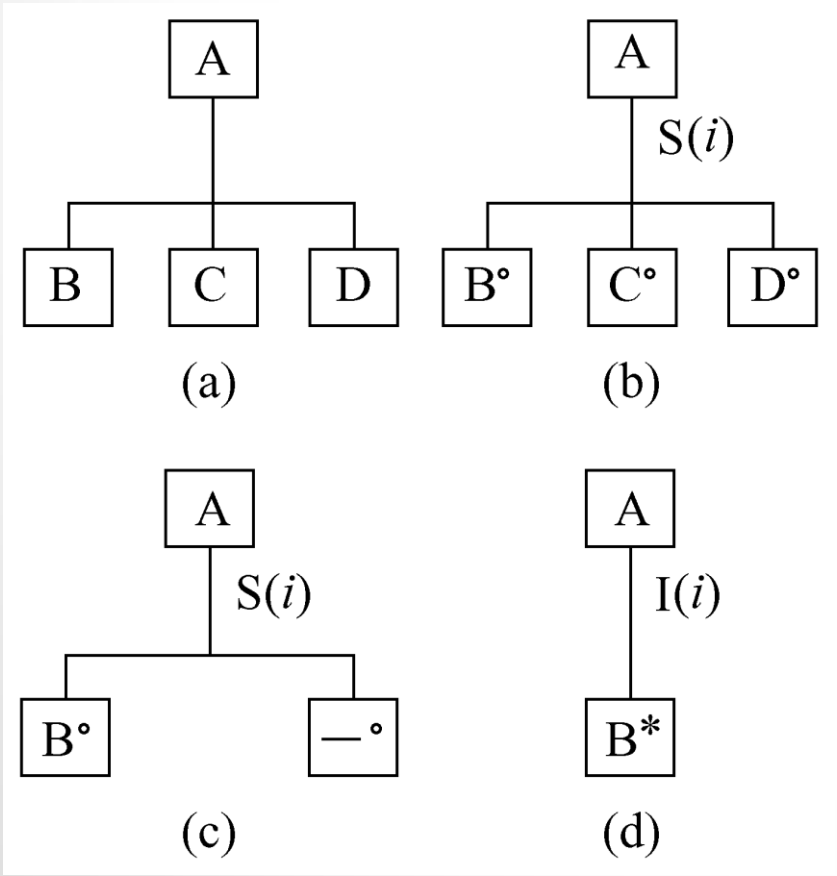
重复结构



A由B出现N次(N≥0)组成(注意，在B的右上角有星号标记)

6.4 面向数据结构的设计方法

6.4.2 改进的Jackson图



(a) 顺序结构，B、C、D中任一个都不能是选择出现或重复出现的数据元素(即不能是右上角有小圆圈或星号标记的元素)；

(b) 选择结构，S右面括号中的数字i是分支条件的编号；

(c) 可选结构，A或者是元素B或者不出现；

(d) 重复结构，循环结束条件的编号为i。

6.4 面向数据结构的设计方法

6.4.3 Jackson法

Jackson结构程序设计方法基本上由下述5个步骤组成。

(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。

6.4 面向数据结构的设计方法

(3) 用下述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。

① 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在程序结构图的相应层次画一个处理框

② 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。

③ 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。

6.4 面向数据结构的设计方法

(4) 列出所有操作和条件(包括分支条件和循环结束条件)，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方法中使用的伪码和Jackson图是完全对应的，下面是和3种基本结构对应的伪码。

顺序结构

A seq

B

C

D

A end

6.4 面向数据结构的设计方法

选择结构

A select cond1

B

A or cond2

C

A or cond3

D

A end

重复结构

A iter until(或while) cond

B

A end

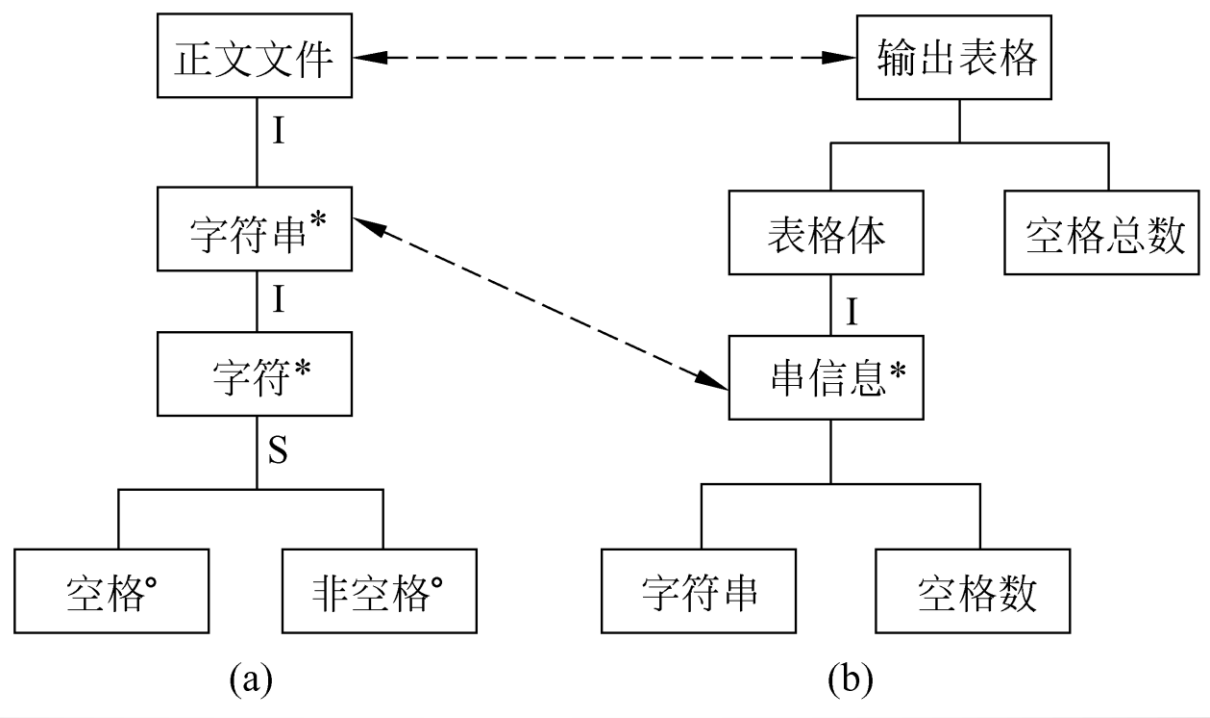
6.4 面向数据结构的设计方法

下面结合一个具体例子进一步说明Jackson结构程序设计方法。

例：一个正文文件由若干个记录组成，每个记录是一个字符串。要求统计每个记录中空格字符的个数，以及文件中空格字符的总个数。要求的输出数据格式是，每复制一行输入字符串之后，另起一行印出这个字符串中的空格数，最后印出文件中空格的总个数。

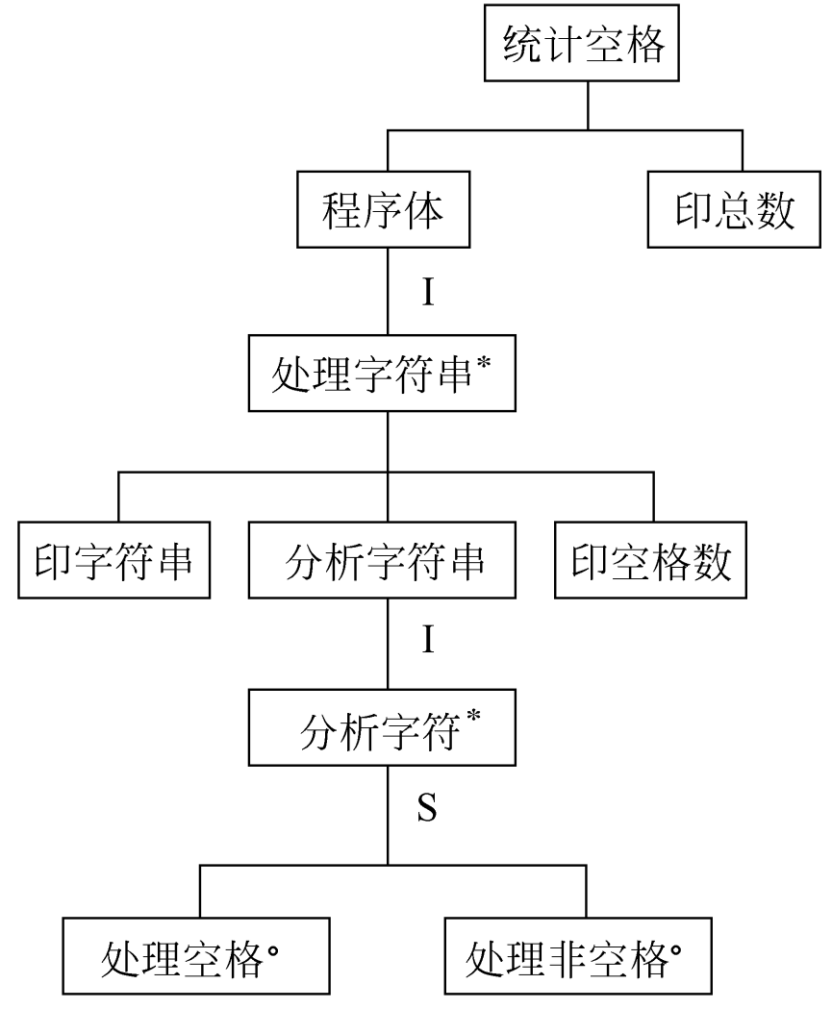
6.4 面向数据结构的设计方法

输入和输出数据的结构很容易确定，用Jackson图描绘的输入输出数据结构



6.4 面向数据结构的设计方法

导出描绘程序结构的Jackson图



6.4 面向数据结构的设计方法

统计空格个数需要的全部操作和条件如下：

(1) 停止

(2) 打开文件

(3) 关闭文件

(4) 印出字符串

(5) 印出空格数目

(6) 印出空格总数

(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

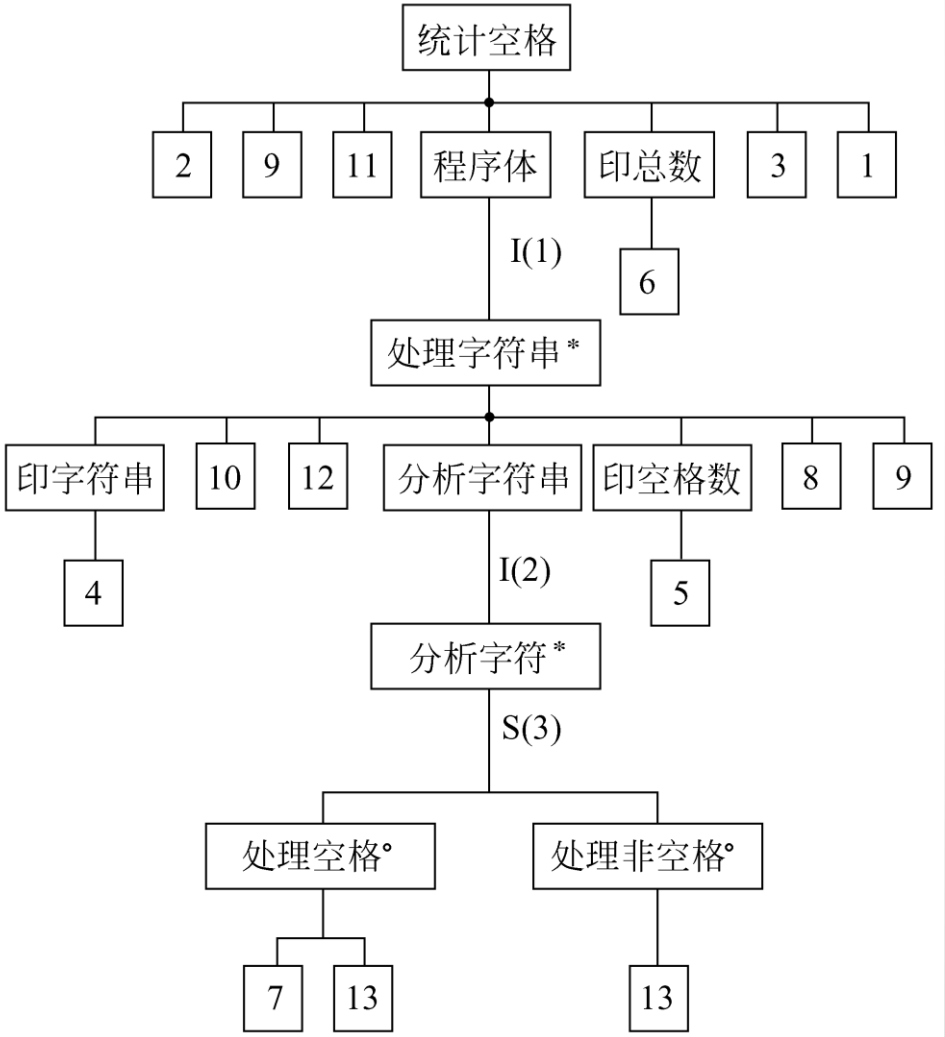
I(1) 文件结束

I(2) 字符串结束

S(3) 字符是空格

6.4 面向数据结构的设计方法

经过简单分析不难把这些操作和条件分配到程序结构图的适当位置：



6.4 面向数据结构的设计方法

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
pointer：=1
分析字符串iter until字符串结束
分析字符select字符是空格
处理空格Seq
sum：=sum+1
pointer：=pointer+1
处理空格end
分析字符or字符不是空格
处理非空格seq
pointer：=pointer+1
处理非空格end
分析字符end
分析字符串end
印空格数seq
印出空格数目
印空格数end
totalsum：=totalsum+sum
读入字符串
处理字符串end
程序体end
印总数seq
印出空格总数
印总数end
关闭文件
停止
统计空格end

主要内容

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

6.5 程序复杂程度的定量度量

定量度量程序复杂程度的方法很有价值：

a) 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，

b) 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；

c) 程序的定量的复杂程度可以作为模块规模的精确限度。

6.5 程序复杂程度的定量度量

6.5.1 McCabe方法

① 流图

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。

流图实质上是“退化了的”程序流程图，描绘程序的控制流程，不表现对数据的具体操作以及分支或循环的具体条件。

6.5 程序复杂程度的定量度量

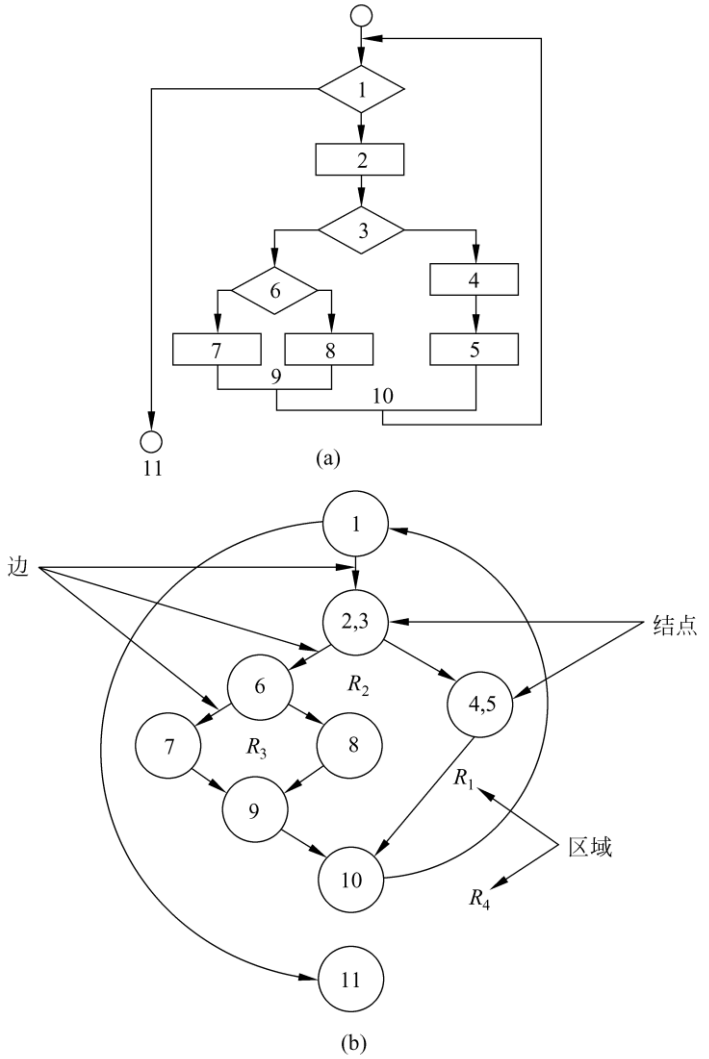
① 流图

一个圆代表一条或多条语句；

一个顺序结构可以合并一个结点；

流图中的箭头线称为边，代表控制流；

在流图中一条边必须终止于一个结点

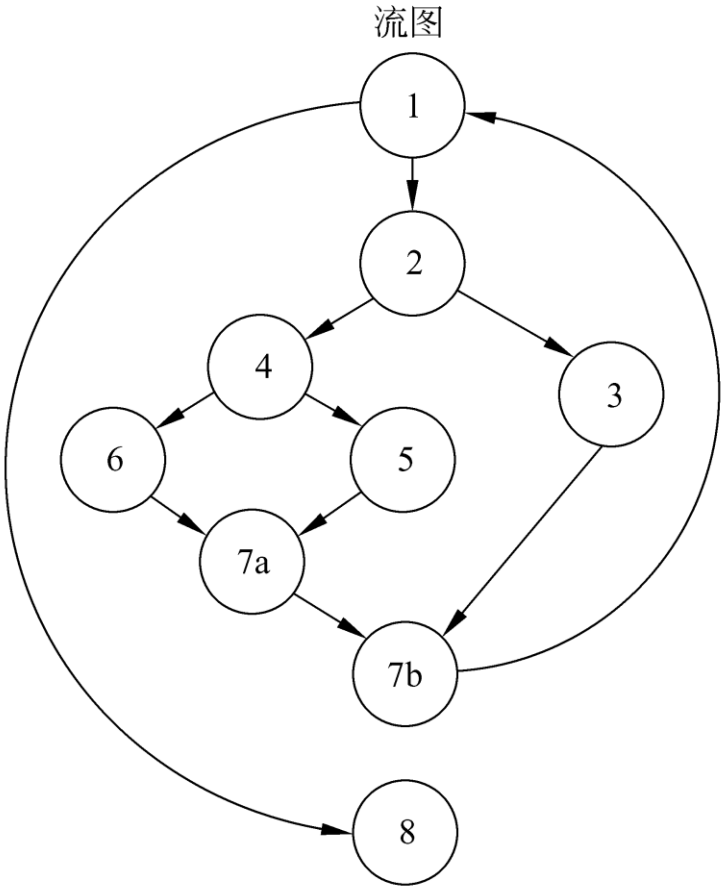


6.5 程序复杂程度的定量度量

① 流图

由PDL翻译成的流图

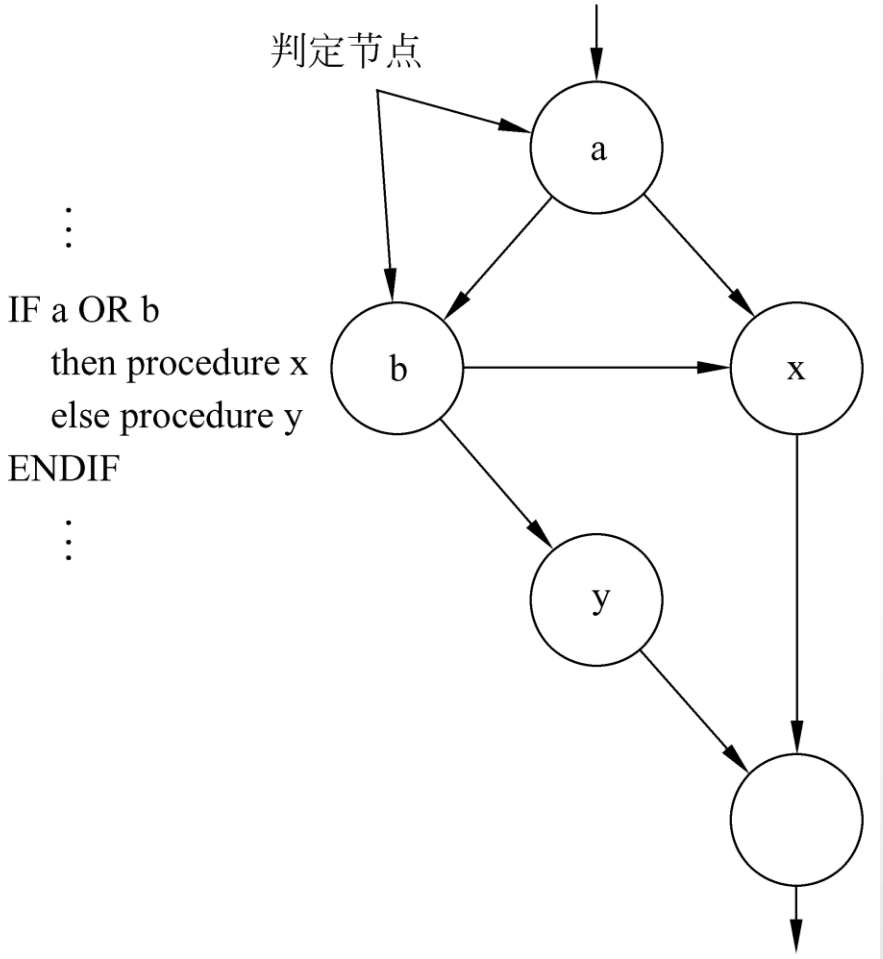
```
PDL
procedure:sort
1:    do while records remain
2:        read record;
          if record field 1=0
3:          then process record;
                store in buffer;
                incremert counter;
4:        elseif record field 2=0
5:          then reset counter;
6:          else process record;
                store in file;
7a:       endif
          endif
7b:   enddo
8:    end
```



6.5 程序复杂程度的定量度量

① 流图

复合条件，就是在条件中包含了一个或多个布尔运算符(逻辑OR，AND，NAND，NOR)



6.5 程序复杂程度的定量度量

② 计算环形复杂度的方法

(1) 流图中线性无关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。

图6.17流图的环形复杂度为4。

6.5 程序复杂程度的定量度量

③ 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)≤10为宜

6.5 程序复杂程度的定量度量

2. 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N1为程序中运算符出现的总次数，N2为操作数出现的总次数，程序长度N定义为：

$$N=N1+N2$$

程序中使用的不同运算符(包括关键字)的个数n_1，以及不同操作数(变量和常数)的个数n_2。Halstead给出预测程序长度的公式如下：

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

6.5 程序复杂程度的定量度量

多次验证都表明，预测的长度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：

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

本章小结

1.结构程序设计技术是进行详细设计的逻辑基础。

2.人机界面设计必须重视。

3.过程设计是详细设计阶段完成的主要工作。

4.在开发有清楚的层次结构时可采用面向数据结构的设计方法完成设计过程设计。

5.使用环形复杂度可以定量度量程序的复杂程度。

本章结束