# 软件工程：第五章 详细设计

## 导学目标

- 了解详细设计的**目的、任务、原则**等
- 了解结构程序设计的三种**基本结构**、**分类**、**优点**等
- 了解人机界面设计经常遇到的问题以及设计过程
- **掌握过程设计工具的画法**：程序流程图、盒图，PAD图、判定表、判定树
- **掌握面向数据结构设计之Jackson图及Jackson方法**
- **掌握程序复杂度度量的McCabe方法**

### 详细设计

- 目的：确定应该怎样具体地实现所要求的系统
- 任务：设计出程序的蓝图
- 原则：逻辑模块的描述简明易懂；采用结构化程序设计方法，降低程序的复杂度，提高程序的可阅读性、可测试性和可维护性

## 第一节 结构程序设计

1965年，E.W.Dijkstra提出在高级语言中取消goto语句

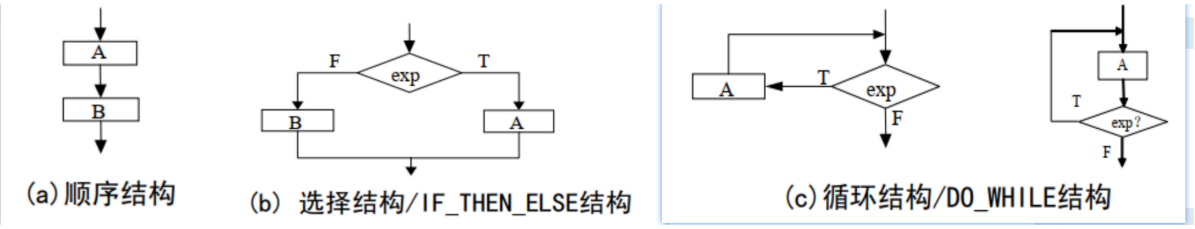
1966年，Bohm和Jacopini提出三种基本结构——**顺序、选择、循环**

**定义**

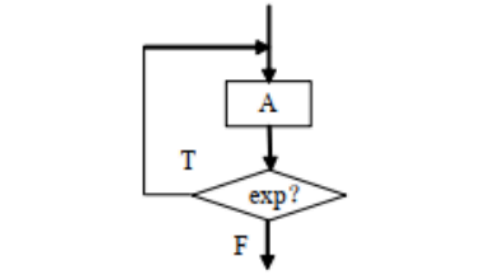
- 是一种程序设计**技术**，采用**自顶向下**，**逐步求精**的设计方法和**单入口单出口**的程序结构

用顺序结构和循环结构可以实现选择结构，因此理论上**最基本的控制结构只有两种**

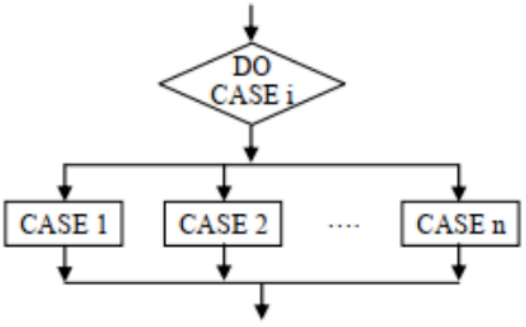
**三种基本的控制结构**




**其他常用的控制结构**




DO_UNTIL型循环结构




多分支结构

**结构程序设计的优点**

- 符合人们解决复杂问题的规律
- 先整体，后细节
- 不使用goto，易理解、易沟通
- 结构清晰、易于修改

**程序结构设计分类**

- 经典结构程序设计：顺序、IF-THEN-ELSE、DO-WHILE
- 扩展结构程序设计：DO-CASE、DO-UNTIL
- 修正结构程序设计：LEAVE、BREAK

# 第二节 人机界面设计

**人机界面设计的重要性**

- 人机界面的设计质量，直接影响用户对软件产品的评价

## 5.2.1 人机界面的设计问题

- **系统响应时间**
  - 指用户完成某个控制动作，到软件给出预期的响应
  - 两个属性：
    - 长度：不能过长也不能过短，适中
    - 易变性：系统响应时间相对于平均响应时间的偏差
- **用户帮助措施**
  - 常见的用户帮助措施有：集成和附加
    - 集成：从一开始就设计在软件里面
    - 附加：在系统建成后添加到软件中
  - 帮助措施必须解决下述问题：
    1. 是否是任何时候?
    2. 怎样请求帮助?
    3. 怎样显示帮助信息?
    4. 怎样返回到正常的交互方式中?
    5. 怎样组织帮助信息?
- **出错信息处理**
  - 是出现问题时，交互式系统给出的“坏消息”
  - 交互式系统给出的出错信息或者是警告信息，有以下属性：
    1. 信息应该用用户可以理解的术语描述问题
    2. 信息应该提供有助于从错误中恢复的建设性意见
    3. 信息应该指出错误可能导致的哪些负面后果
    4. 信息应该伴随着听觉或者视觉上的提示
    5. 信息不能带有指责色彩
- **命令交互**

- 提供命令交互时，需要考虑的问题
    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
    3. 学习和记忆命令的难度有多大
    4. 用户是否可以定制或缩写命令

### 5.2.2 人机界面的设计过程

- 用户界面设计是一个迭代的过程
    1. 创建设计模型
    2. 原型实现这个原型
    3. 用户适用和评估
    4. 根据用户意见进行修改
- 用户界面工具箱或用户界面开发系统
    - 用于界面设计和原型软件开发的软件工具
- 用户界面设计主要依靠设计者的经验，总结得出**设计指南**
    1. **一般交互指南**
        - 保持一致性
        - 提供有意义的反馈
        - 在执行有较大的破坏性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
        - 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措施
        - 用简单动作或者动作短语作为命令名
    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
        - 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
        - 使用模拟方式显示信息
        - 高效率的使用显示屏
    3. **数据输入指南**
        - 减少用户的输入动作
        - 保持信息显示与数据输入的一致性

- 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 过程设计工具

**详细设计使用的工具**

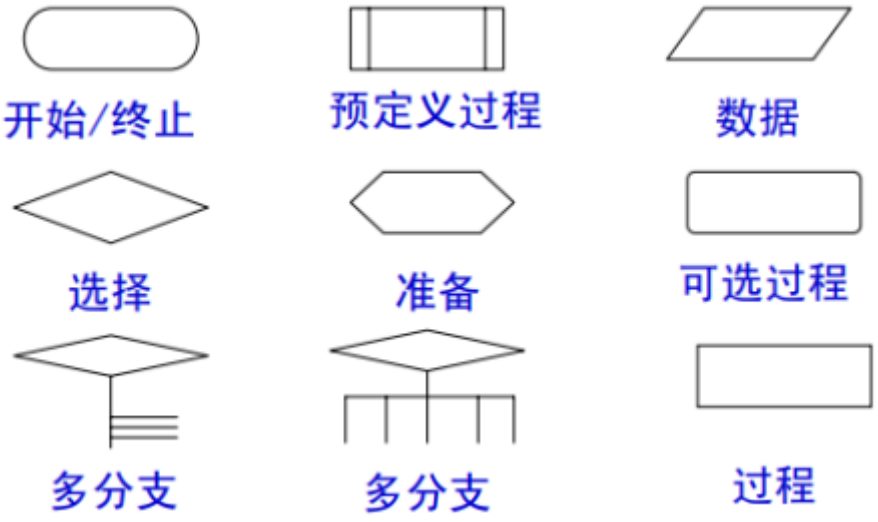
- 描述程序的处理过程
- 可以分为图形、表格和语言三类
- 能提供对设计的无歧义描述，指明控制流程、处理功能、数据组织和其他方面的实现细节

**详细设计工具的种类**

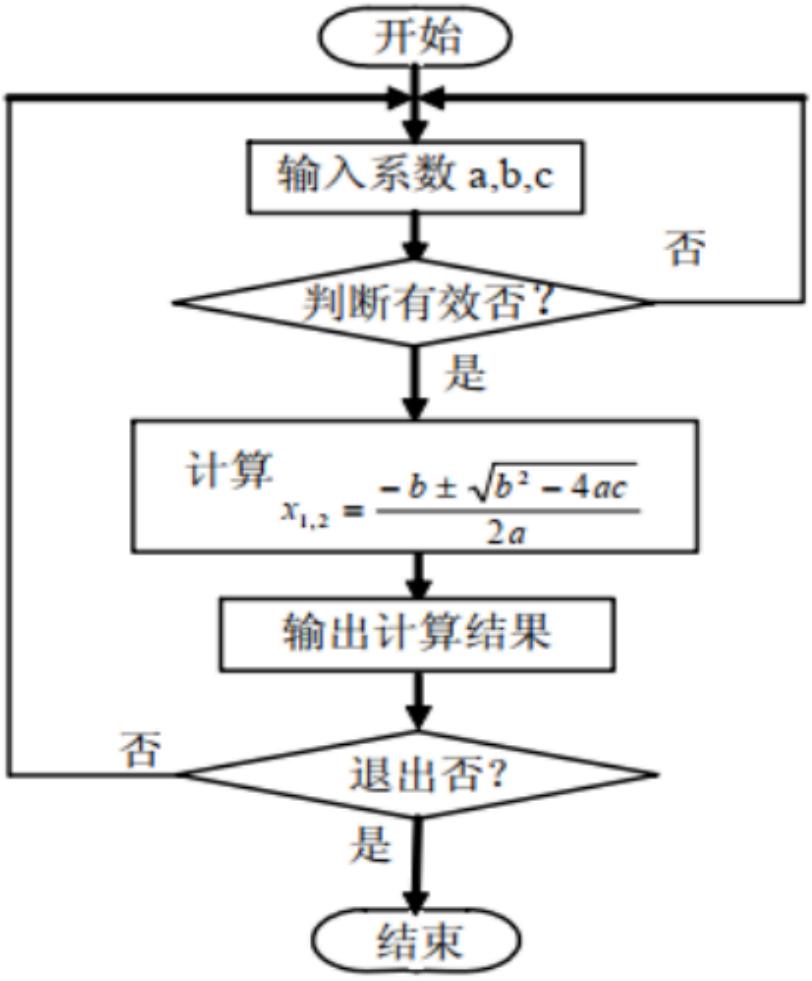
- 程序流程图
- 盒图（N-S图）
- PAD图
- 判定表、判定树
- 过程设计语言

## 5.3.1 程序流程图

- 程序框图，是20世纪70年代程序设计的主要工具，总的趋势是越来越多的人不再使用
- 程序流程图的主要缺点
  - 不是逐步求精的好工具
  - 箭头代表控制流，程序员不受任何约束
  - 不易表示数据结构
- 程序流程图中使用的符号

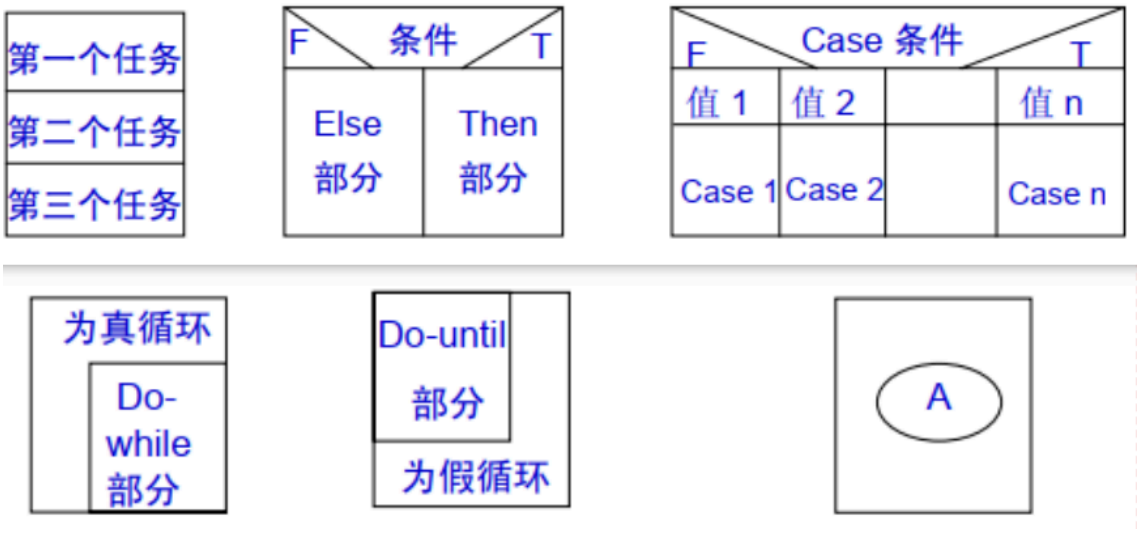



- 使用示例：求一元二次方程的解

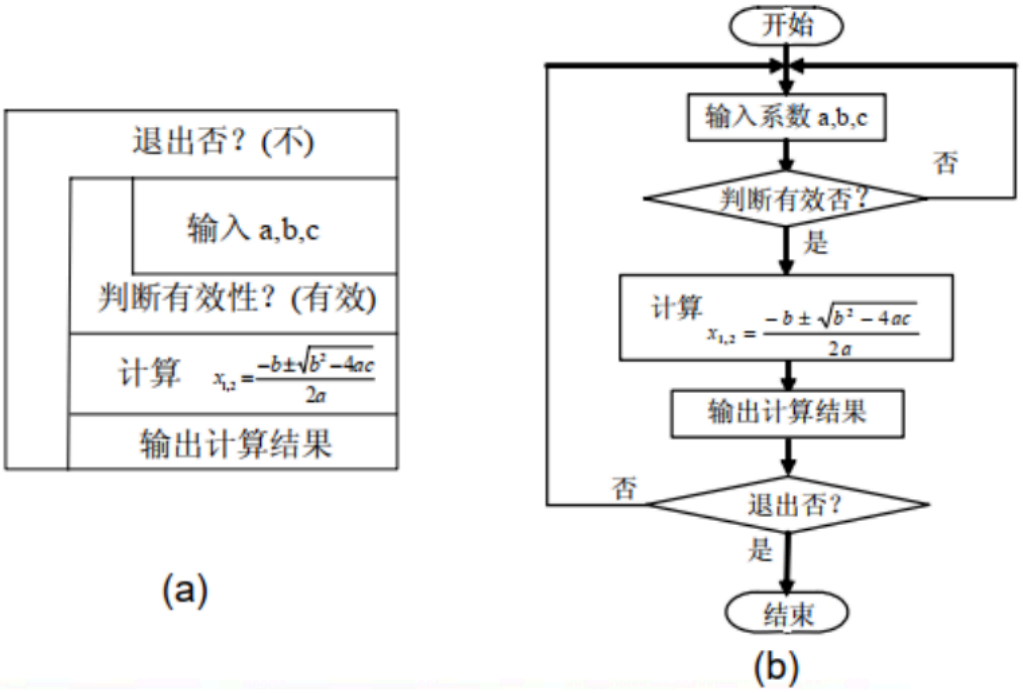



### 5.3.2 盒图（N-S图）

- 主要特点
  - 功能域明确
  - 不可能随意转移控制
  - 容易确定局部和全局数据的作用域
  - 很容易表现嵌套关系
- 基本符号




- 示例：求一元二次方程的解




### 5.3.3 PAD图

- 是问题分析图，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
- 主要特点
  - 必然是结构化程序
  - 程序结构十分清晰
  - 程序逻辑易读、易懂、易记
  - 容易将PAD图转换为高级语言源程序
  - 既可以表示程序逻辑，又可以表示数据结构
  - 支持自顶向下，逐步求精的方法
- 基本符号

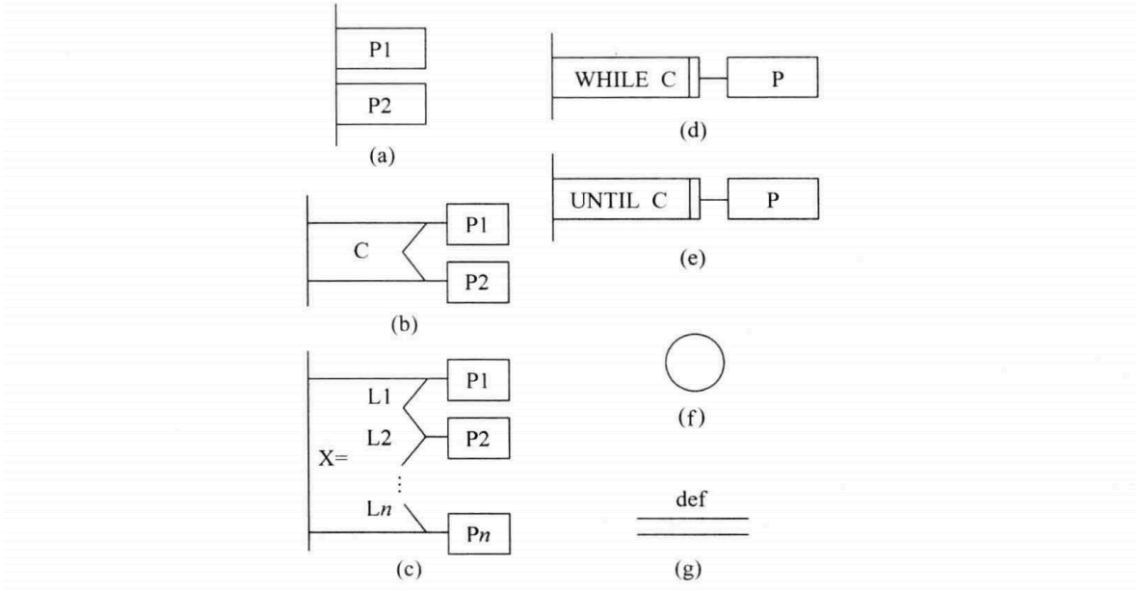



图 6.5　PAD 图的基本符号

(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

- 示例

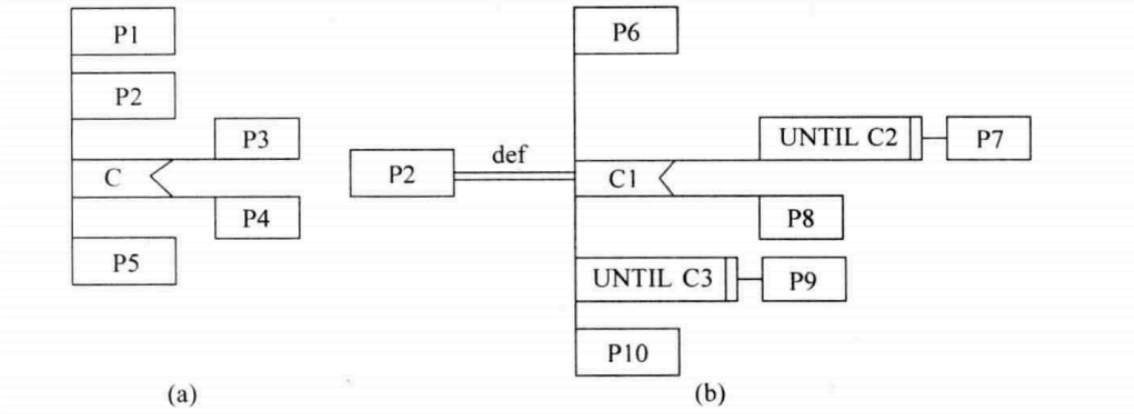
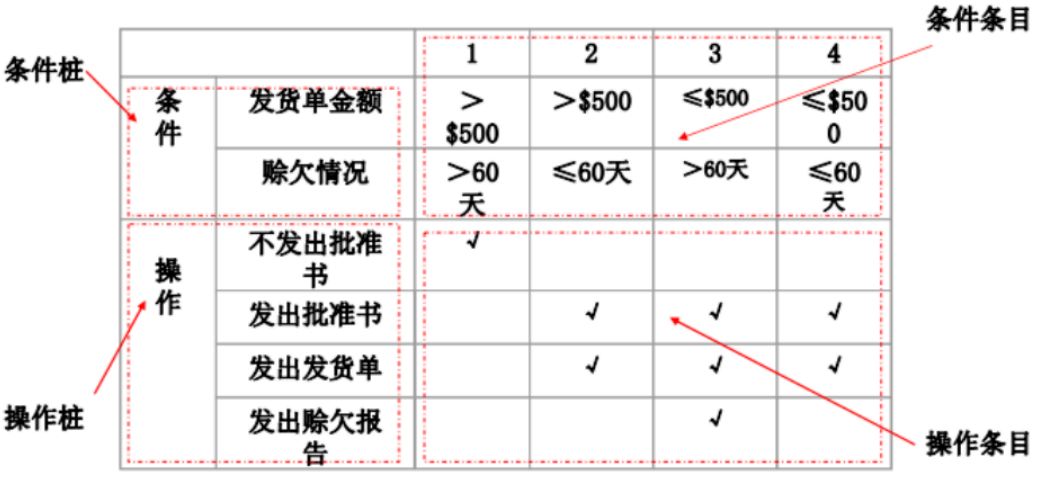



图 6.6　使用 PAD 图提供的定义功能来逐步求精的例子

（a）初始的 PAD 图；（b）使用 def 符号细化处理框 P2

### 5.3.4 判定表

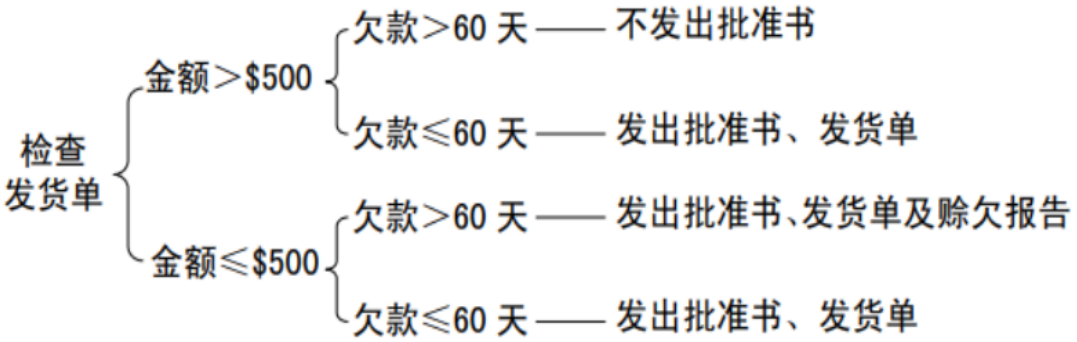
- 在某些数据处理中，某数据流图的加工需要依赖于**多个逻辑条件**的取值，也就是完成加工的一组动作是某一组条件的**取值组合**而引发的，这时候使用判定表来描述比较合适
- 当算法中包含多重嵌套条件的条件选择时，用程序流程图、盒图、PAD图、过程设计语言（PDL）都不易清楚地描述。然而判定表确能够清晰的表示**复杂的条件组合**与应做的动作之间的对应关系
- **判定表组成**
  - 左上部列出的是所有的条件
  - 左下部列出的是所有的操作
  - 右上部表示各种条件组合的一个矩阵
  - 右下部表示对应每种条件组合的操作
- 判定表示例：发货判定表




| | | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 条件 | 发货单金额 | >$500 | >$500 | ≤$500 | ≤$500 |
| | 赊欠情况 | >60天 | ≤60天 | >60天 | ≤60天 |
| 操作 | 不发出批准书 | √ | | | |
| | 发出批准书 | | √ | √ | √ |
| | 发出发货单 | | √ | √ | √ |
| | 发出赊欠报告 | | | √ | |

### 5.3.5 判定树

- 判定表虽然能清晰的表示复杂的条件组合与应做的动作之间的对应关系，但是其含义不是一眼就能看出来的，初次接触这种工具的人需要有个简短的学习过程
- 判定树是判定表的变种，它也能清晰的表示复杂的条件组合与应做的动作之间的对应关系。判定树的优点在于不需要任何说明，一眼就能看出其含义，易于理解和使用
- 判定树示例：发货判定树




### 5.3.6 过程设计语言（PDL）

- 也称伪码，是用正文的形式表示**数据处理**和**过程的设计工具**
- PDL特点
  - 具有关键字固定语法
  - 自然语言的描述
  - 数据说明的手段
  - 模块定义和调用的技术

### 5.3.7 过程设计工具的选择

- 不太复杂的判断逻辑，使用**判断树**会比较好
- 复杂的判断逻辑，使用**判定表**比较好
- 如果一个处理逻辑既包含了一般的顺序执行动作，又包含了判断或循环逻辑，则使用**结构化语言（PDL）**比较好

## 第四节 面向数据结构的设计方法

**定义**

- **面向数据流**的设计方法：根据**数据流**，确定软件结构
- **面向数据结构**的设计方法：根据**数据结构**，设计程序处理过程的方法

**目标**

- 得出对处理过程的描述，也就是在完成软件设计之后，使用面向数据结构设计方法来设计每个模块的处理过程

**方法**

- **Jackson**
- Warnire

### 5.4.1 Jackson图

**三种类型**

- 顺序结构：数据由一个或者多个元素组成，**每个元素按照确定次序出现一次**

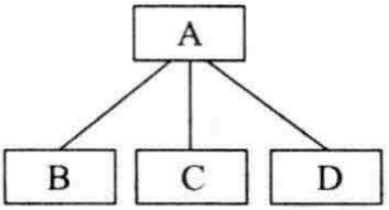



图 6.8　A 由 B、C、D 3 个元素顺序组成

- 选择结构：数据包含两个或多个元素，**按一定条件在这些元素中选择一个**

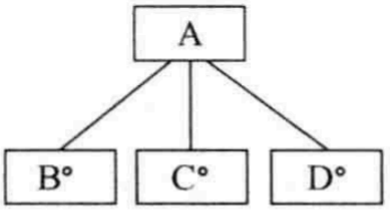



图 6.9　根据条件 A 是 B 或 C 或 D 中的某一个
（注意，在 B、C 和 D 的右上角有小圆圈做标记）

- 重复结构：根据条件，由一个数据元素**出现0次或者多次**构成

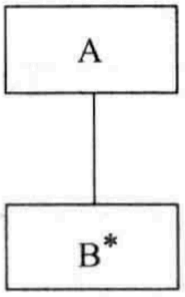


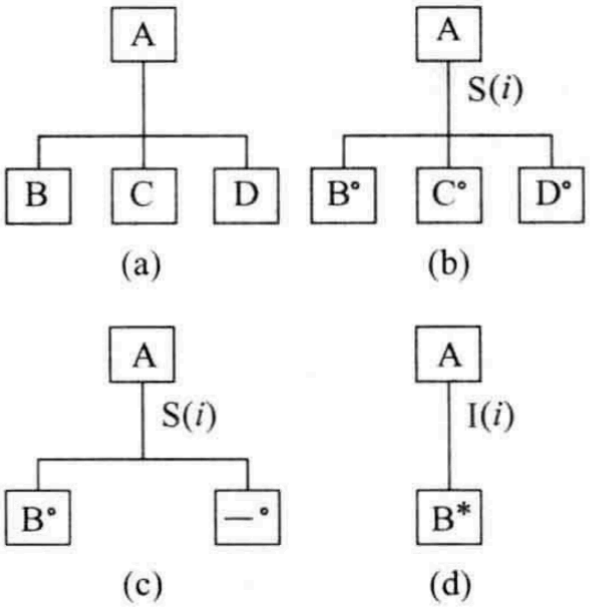

图 6.10　A 由 B 出现 $N$ 次（$N \geqslant 0$）组成
（注意，在 B 的右上角有星号标记）

**优点**

- 便于表示程序结构
- 形象直观、可读性好
- 既能表示数据结构，也能表示程序结构

**改进Jackson图**

- Jackson图的缺点
  - 表示选择或者重复结构时，**选择或者循环结束条件不能直接在图上表示出来**，影响了图的表达能力
- 改进策略
  - 框间连线由**斜线变为直线**，**增加**选择和重复结构**条件**
- 改进后结构图

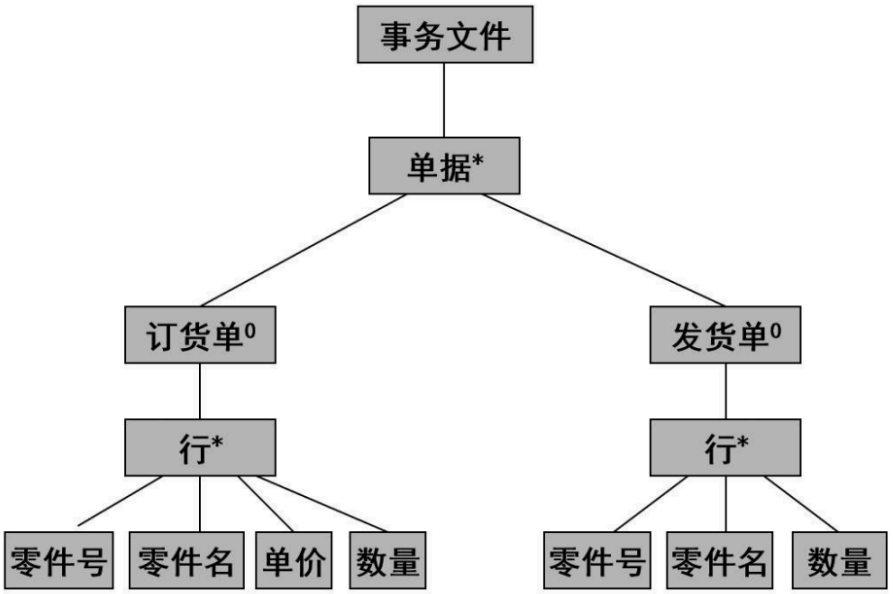



**Jackson方法步骤**

1. 确定输入数据和输出数据**逻辑结构**，用Jackson图表达
2. 确定输入结构和输出结构中有对应关系（因果）的单元
3. 描绘数据结构的Jackson图导出描绘程序结构Jackson图
4. 列出所有操作和条件，分配到Jackson图中
5. 用伪码表示

## 5.4.2 考题分析

- 某仓库管理系统每天要处理大批单据的事务文件。单据分为订货单和发货单两种，每张单据由多行组成，订货单每行包括零件号、零件名、单价、数量等四个数据项，发货单每行包括零件号、零件名、数量等三个数据项，用Jackson结构图表示该事务文件的数据结构



# 第五节 程序复杂度度量

**目的**

- 定量的衡量软件模块的性质

**方法**

- McCabe
- Halstead

## 5.5.1 McCabe方法

- McCabe方法是根据**程序控制流**的复杂程度定量度量程序的复杂程度，这样度量出的结果叫**程序的环形复杂度**

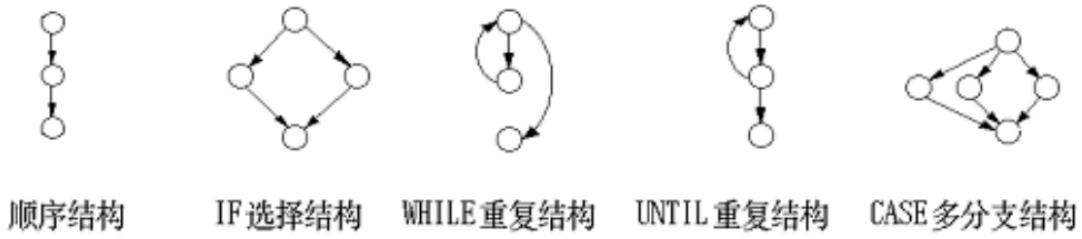
**相关概念**

> 这里使用流图

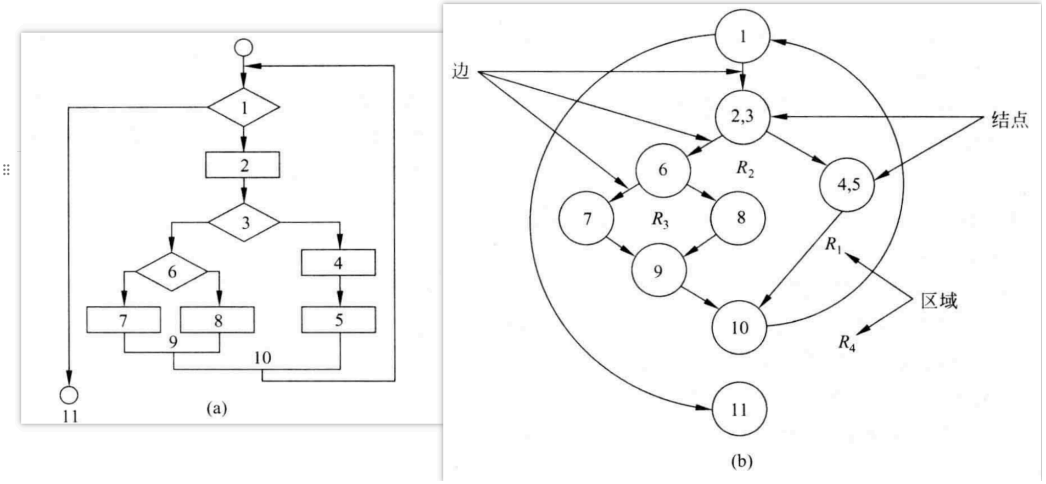
- 用圆表示结点点，一个圆代表一条或多条语句
  - 程序流程图中的一个顺序的处理框序列和一个菱形判定狂，可以映射成流图中的一个结点
- 流图中的箭头线称为边，它和程序流程图中的箭头线类似，代表控制流。
  - 在流图中一条边必须终止于一个结点,即使这个结点并不代表任何语句(实际上相当于一个空语句)。
- 由边和结点围成的面积称为区域，当计算区域数时应该包括图外部未被围起来的那个区域。

**步骤**

1. 根据过程设计结果画出相应流图描述程序控制流，基本图形符号如下图所示




   - 程序流程图映射成流图




2. 计算环形复杂度（三种方法）

- ```
  V（G）＝区域数

  V（G）＝E-N＋2          E为流图中边数，N为流图中节点数

  V（G）＝P＋1            P为判定点数
  ```
- 上图示例
  - V（G）=4（区域数）
  - V（G）=11（边数）-9（结点数）+2=4
  - V（G）=3（判定结点数）+1=4
  ```