

# 软件工程

## 第5讲 结构化软件设计

贾西平

Email: jiexp@126.com

1

1

## 课程主要内容

### 面向过程的软件工程

可行性研究

需求分析

结构化软件设计

软件编码

软件测试

### 面向对象的软件工程

面向对象概述

面向对象分析

面向对象设计

动态建模

面向对象测试

### 软件工程项目管理

软件度量

项目计划

风险管理

质量保证

2

## 提纲

- 相关概念
- 体系结构设计任务工具和原则
- 面向数据流的体系结构设计方法
- 详细设计的任务、工具和原则
- 详细设计的方法
- 面向数据结构的设计方法

09:09

3

3

## 提纲

- **相关概念**
- 体系结构设计的任务、工具和原则
- 面向数据流的体系结构设计方法
- 详细设计的任务、工具和原则
- 面向数据流的详细设计的方法
- 面向数据结构的设计方法

09:09

4

4

## 相关概念

模块

模块化

模块独立性

抽象

信息隐蔽

09:09

5

5

## 模块

- **概念**：具有相对独立性的，由数据说明、执行语句等程序对象构成的集合。
- **具体表现**：函数、子程序、过程等。
- **四个特征**：输入/输出(接口)、功能、内部数据和程序代码。
  - **输入/输出**：实现模块与其他模块间的数据传送。
  - **功能**：模块所完成的工作。
  - **内部数据**：仅在模块内部使用的局部量。
  - **程序代码**：描述实现模块功能的具体方法和步骤。
- **外部特征**：输入/输出、功能
- **内部特征**：内部数据、程序代码

09:09

6

6

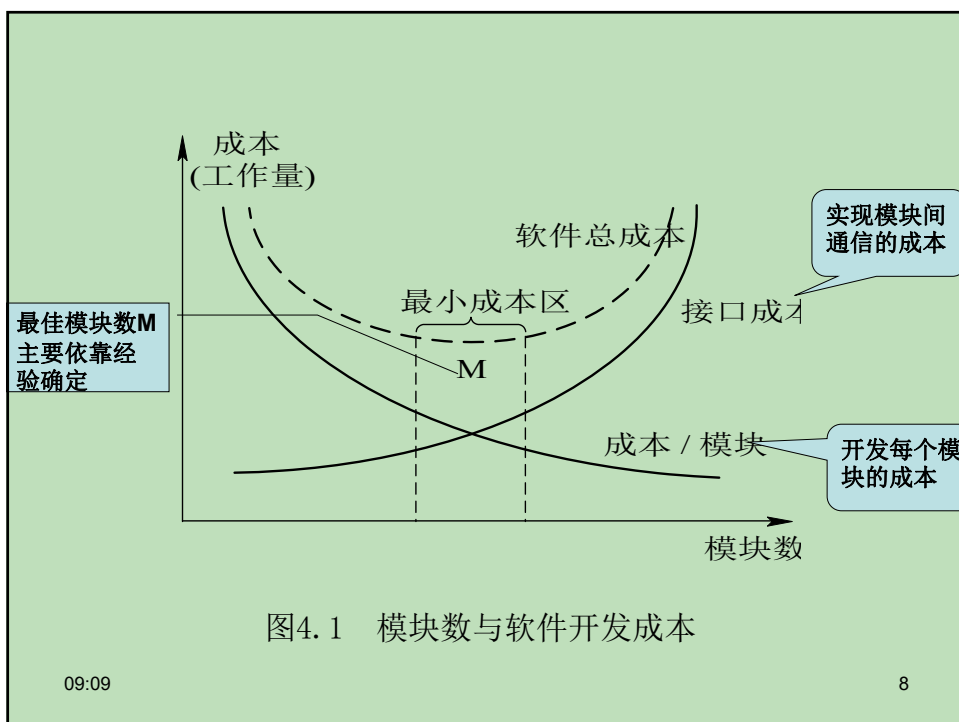
## 模块化

- **概念**：将整个程序划分为若干个模块，每个模块用于实现一个特定的功能。
- **理论依据**  
将复杂问题分解成若干个小问题，各个击破，工作量小于直接解决复杂问题所需的工作量。
- **模块化的好处**
  - 降低软件开发的难度、减少工作量
  - 使程序结构清晰，增加易读性和易修改性
  - 有利于提高代码的可重用性及团队合作开发大型软件的可行性。

09:09

7

7



09:09

8

8

## 模块独立性

- 耦合性
- 内聚性

09:09

9

9

## 耦合性

- 用于度量**软件结构内部**不同模块间联系的紧密程度。
- 模块间联系越紧密，耦合性越高，模块的独立性越低。
- **决定因素**：模块接口的复杂程度、调用模块的方式以及通过模块接口的数据。

09:09

10

10

## 耦合性分类

- **数据耦合**：两个模块间仅通过模块参数交换信息，且交换的信息全部为简单数据。
- **公共耦合**：两个或多个模块通过引用公共数据相互联系。
- **控制耦合**：模块之间交换的信息中包含有控制信息。
- **内容耦合**：一个模块对另一个模块中的内容进行了直接引用甚至修改，或通过非正常入口进入另一模块内部，或一个模块具有多个入口，或两个模块共享一部分代码。

09:09

11

11

## 耦合性分类

弱 
→
 强

数据  
耦合

公共  
耦合

控制  
耦合

内容  
耦合

09:09

12

12

## 耦合性分类（参考）

- **非直接耦合**：同级模块相互之间没有信息传递。
- **数据耦合**：调用下属模块时，交换的都是简单变量。
- **特征耦合**：调用下属模块时，交换的是数据结构。
- **控制耦合**：模块间传递的信息是作为控制信息的开关值或标志量。
- **外部耦合**：允许一组模块访问同一个全局变量。
- **公共耦合**：允许一组模块访问同一个全局性的数据结构。如：共享的通信区、公共的内存区域、任何存储介质文件、物理设备等。
- **内容耦合**：一个模块可直接调用另一个模块中的数据，或者直接转移到另一个模块中去，或者一个模块有多个入口。

09:09

13

13

## 内聚性

- 用于度量**模块内部**各个组成元素之间相互结合的紧密程度。
- 模块中组成元素结合的越紧密，模块的内聚性越高，独立性越高。
- 模块的高内聚性往往意味着模块间的低耦合性。
- 软件设计时应将更多的注意力集中在**提高模块的内聚性上**。

09:09

14

14

## 内聚性分类

- **偶然内聚**：模块内的各个任务在功能上没有实质性联系，因“偶然”因素组合在一起。
- **逻辑内聚**：由若干个逻辑功能相似的任务组成，通过模块外引入的一个开关量选择其一执行。
- **时间内聚**：模块内的各个任务由相同的执行时间联系在一起。例如，初始化模块。
- **过程内聚**：模块内的各个任务必须按照某一特定次序执行。
- **通信内聚**：模块内部的各个任务靠公用数据联系在一起。
- **顺序内聚**：模块内的各个任务是顺序执行的。
- **功能内聚**：模块各个成分结合在一起，完成一个特定的功能。

15

15

## 内聚性分类

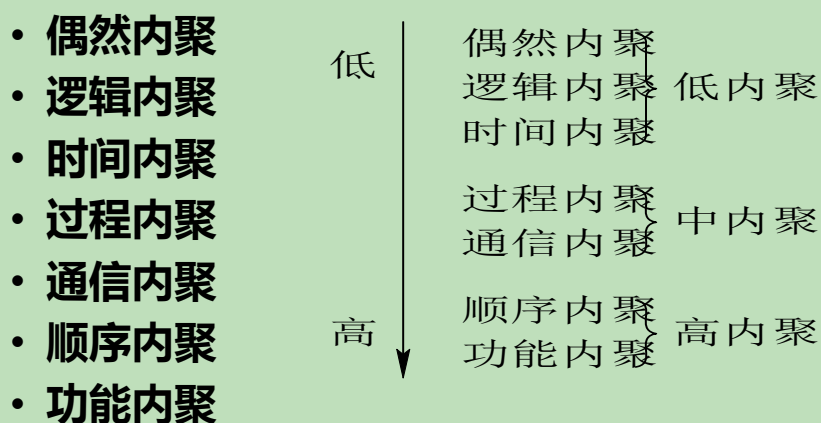


图4.2 内聚性的排列

09:09

16

16



## 抽象

将现实世界中**具有共性**的一类事物的**相似的、本质的**方面集中**概括**起来，而**暂时忽略**它们之间的**细节差异**。

09:09

17

17

## 信息隐蔽

- **概念**：模块将自身的内部信息向其他模块隐藏起来，以避免其他模块不恰当的访问和修改。
- **目的**：提高模块的独立性，减少将一个模块中的错误扩散到其他模块的机会。
- **实质**：模块之间的信息传递只允许通过合法的调用接口来实现

09:09

18

18

## 提纲

- 相关概念
- **体系结构设计任务、工具和原则**
- 面向数据流的体系结构设计方法
- 详细设计的任务、工具和原则
- 面向数据流的详细设计的方法
- 面向数据结构的设计方法

09:09

19

19

## 体系结构设计概述

- <https://www.bilibili.com/video/BV1Q741157ve?p=64>
- <https://www.bilibili.com/video/BV1Q741157ve?p=65>

09:09

20

20

## 体系结构设计的任务

软件整体结构设计

数据结构和数据库设计

系统可靠性、安全性设计

编写文档，参加复审

09:09

21

21

## 软件整体结构设计

将系统按功能划分为模块

确定模块之间的调用关系及其接口

对划分的结果进行优化和调整

09:09

22

22

## 数据结构和数据库设计

- 对数据字典加以细化，从计算机技术实现的角度出发，**确定软件涉及的文件系统及各种数据的结构。**
- 将独立于数据库实现的概念模型与具体的数据库管理系统的特征相结合，**建立数据库的逻辑结构**
- **确定数据库的模式、子模式及对数据库进行规范和优化等。**

09:09

23

23

## 系统可靠性、安全性设计

- **可靠性设计**  
保证程序及其文档具有较高的**正确性**和**容错性**，对可能出现的错误易于**修改**和**维护**。
- **安全性设计**  
增强系统的**自我防护能力**和**运行的稳定性**，防止系统遭受到有意或无意地入侵和破坏，保证系统在安全的环境下正常工作。

09:09

24

24

## 编写文档

### 体系结构设计阶段应交付的文档：

- (1) **体系结构设计说明书**：给出系统总体结构设计的结果，为系统的详细设计提供基础。
- (2) **用户手册**：根据体系结构设计成果，对需求分析阶段编写的用户手册进行补充和修改。
- (3) **测试计划**：明确测试中应采用的策略、方案、预期的测试结果及测试的进度安排。
- (4) **数据库设计说明书**：给出目标系统中数据库管理系统的选择及逻辑结构等设计结果

09:09

25

25

## 体系结构设计的工具

- **HIPO图**
- **结构图**

09:09

26

26

## HIPO图

- **HIPO**: Hierarchy Plus Input/Processing/Output
- **中文全名**：层次图加输入/处理/输出图
- **实质**：在描述软件总体模块结构的层次图(H图)的基础上，加入了用于描述每个模块输入/输出数据和处理功能的IPO图。

09:09

27

27

## H图

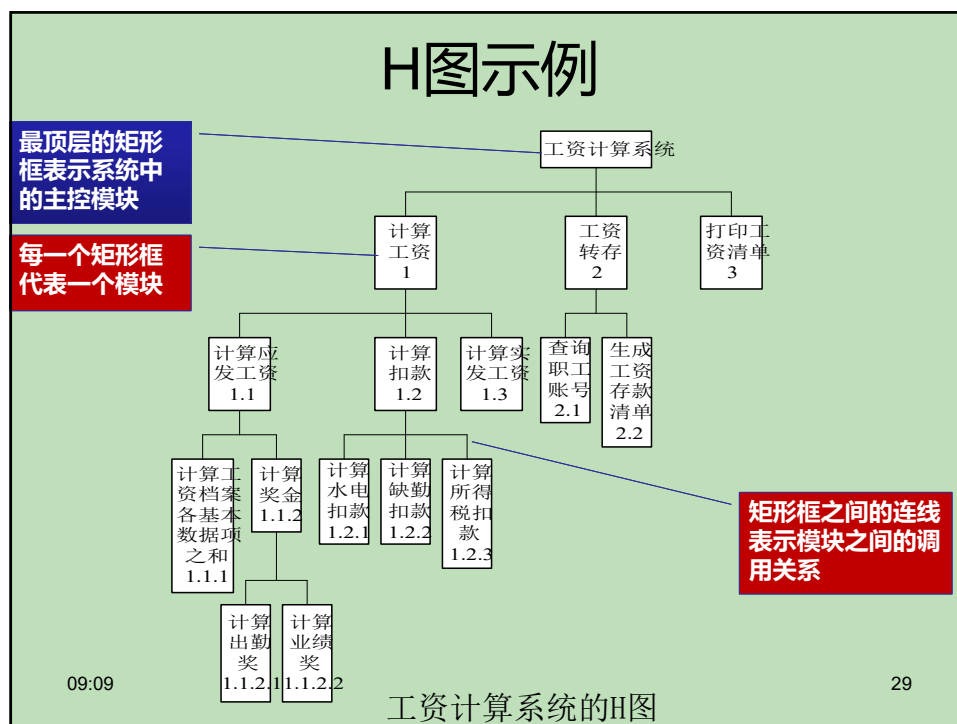
在体系结构设计过程中描  
绘软件的层次结构

适用于自顶向下进行分解  
的软件结构设计方法

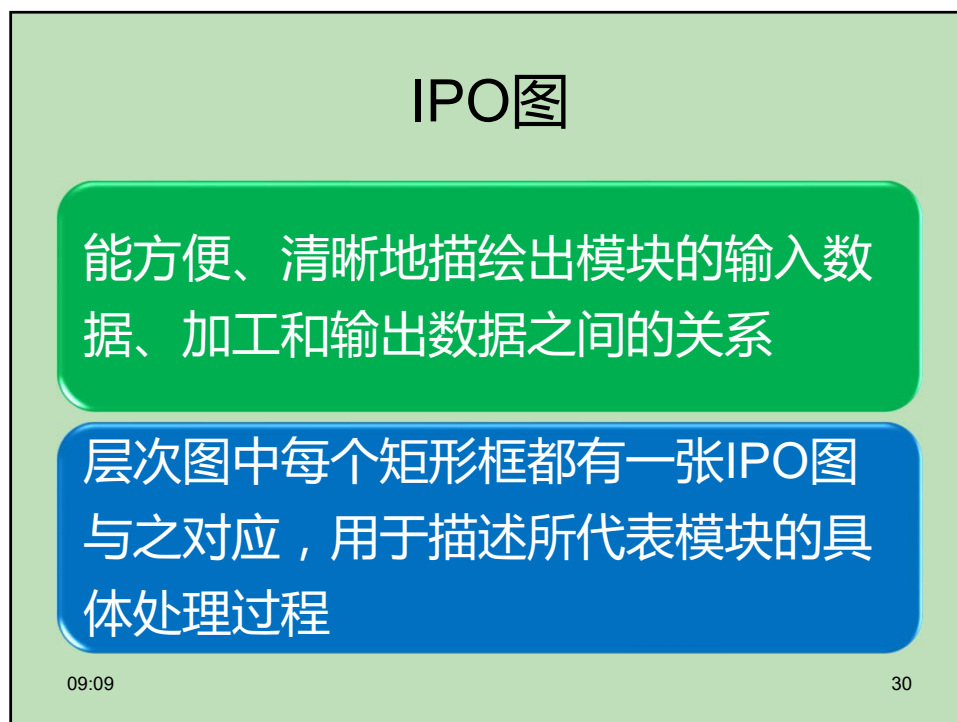
09:09

28

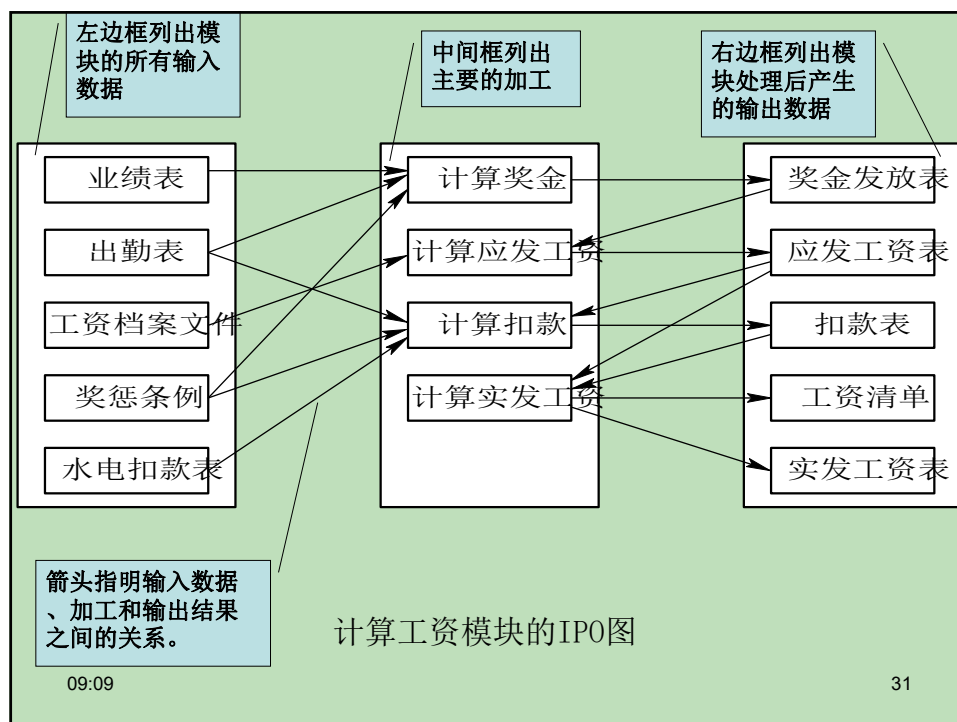
28



29



30







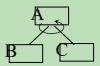
31



32



### 结构图中的基本符号

符 号	含 义
	用于表示模块，方框中标明模块的名称
	用于描述模块之间的调用关系
	用于表示模块调用过程中传递的信息，箭头上标明信息的名称；箭头尾部为空心圆表示传递的信息是数据，若为实心圆则表示传递的是控制信息
	表示模块A选择调用模块B或模块C
	表示模块A循环调用模块B和模块C

09:09

33

33

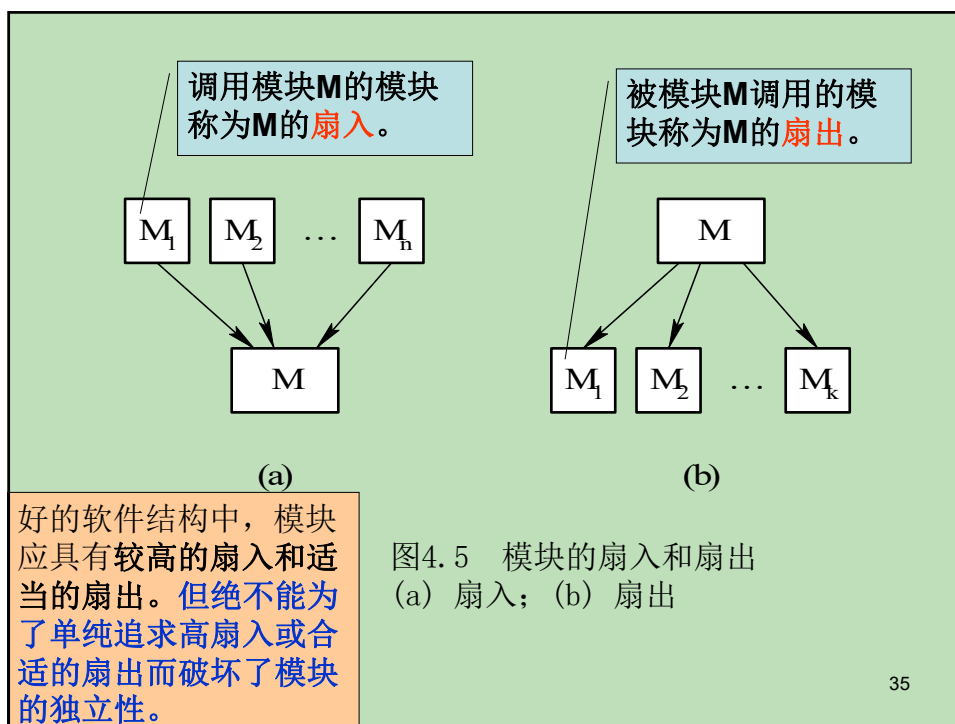
## 体系结构设计的原则

- (1) 降低模块的耦合性，提高模块的内聚性
- (2) 保持适中的模块规模
- (3) 模块应具有高扇入和适当的扇出
- (4) 软件结构中的深度和宽度不宜过大
- (5) 模块的作用域应处于其控制域范围之内
- (6) 尽量降低模块的接口复杂度

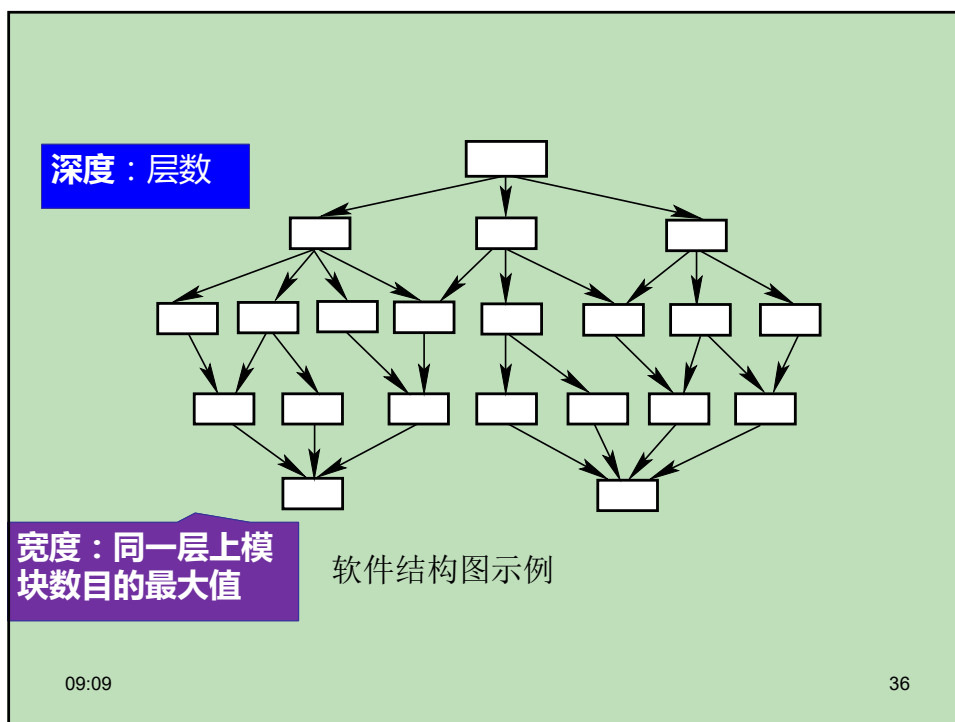
09:09

34

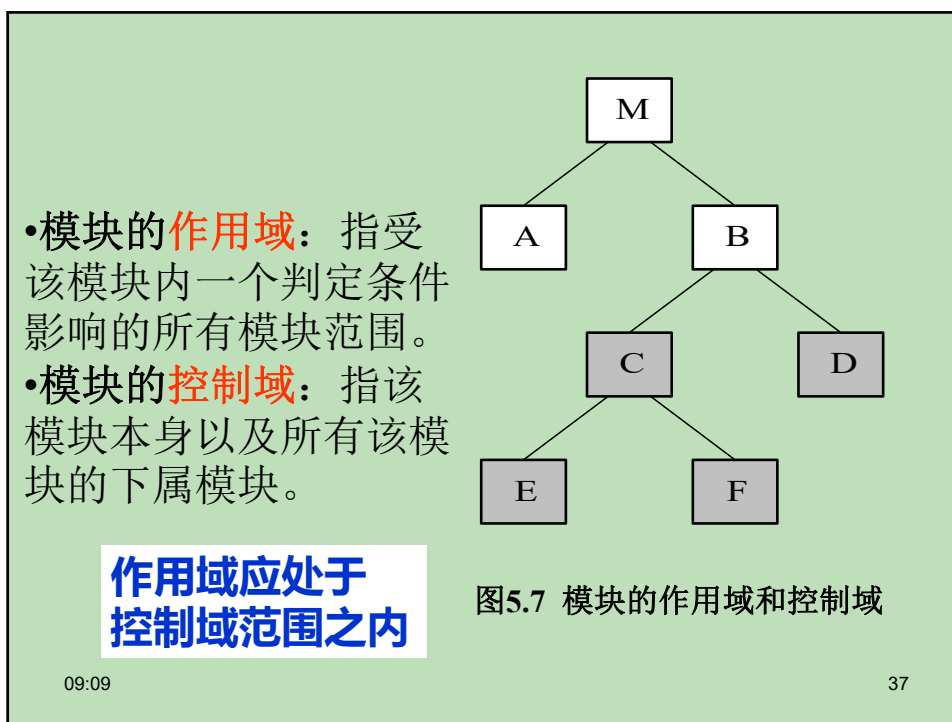
34



35



36



37

## 体系结构设计说明书

体系结构设计说明书的主要内容：

- (1) **引言**：用于说明编写本说明书的目的、背景，定义所用到的术语和缩略语，以及列出文档中所引用的参考资料等。
- (2) **总体设计**：用于说明软件的需求规定、运行环境要求、处理流程及软件体系结构等。
- (3) **运行设计**：用于说明软件的运行模块组合、运行控制方式及运行时间等。
- (4) **模块设计**：用于说明软件中各模块的功能、性能及接口等。
- (5) **数据设计**：用于说明软件系统所涉及的数据对象的逻辑数据结构的设计。
- (6) **出错处理设计**：用于说明软件系统可能出现的各种错误及可采取的处理措施。

09:09

38

38

## 提纲

- 相关概念
- 体系结构设计任务工具和原则
- **面向数据流的体系结构设计方法**
- 详细设计的任务、工具和原则
- 详细设计的方法
- 面向数据结构的设计方法

09:09

39

39

## 面向数据流的体系结构设计

- **任务**  
将数据流图转换成设计阶段所需的软件结构
- **数据流图类型**
  - 变换型数据流图
  - 事务型数据流图

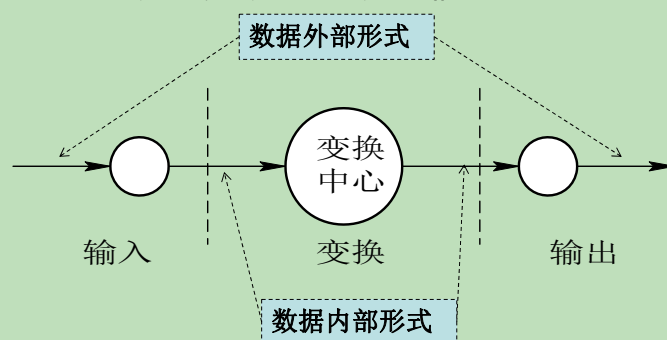
09:09

40

40

## 变换型数据流

- **结构特点**：由(逻辑)输入、变换中心和(逻辑)输出三部分组成。
- 反映了一个**顺序结构**的加工过程
- **描述的加工过程**  
输入、转换、变换、转换、输出



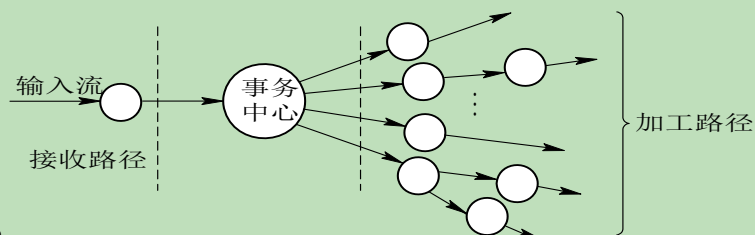
09:09

41

41

## 事务型数据流图

- **结构特点**：输入流经过“事务中心”的加工被分离为多个发散输出流，形成多个平行的加工处理路径
- **描述的加工过程**
  - 外部数据进入系统，被送往事务中心；
  - 事务中心接收输入数据，分析确定其类型；
  - 根据输入数据类型为其选择一条加工路径。

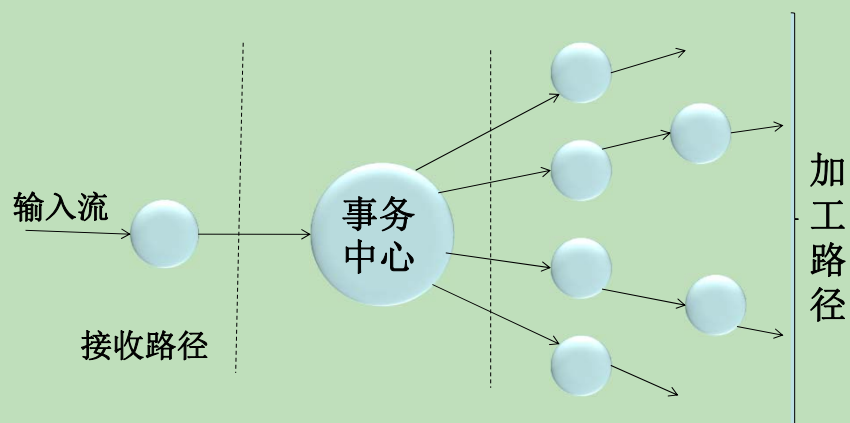


09:09

42

42

## 事务型数据流图



09:09

43

43

## 混合型数据流（补充）

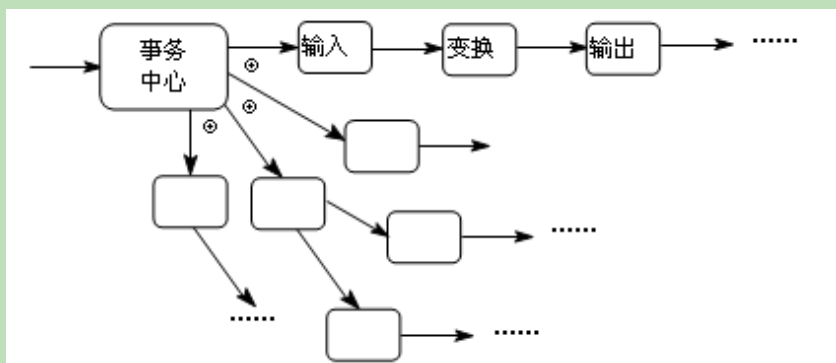
- 在一个大型系统的DFD中，变换流和事务流往往会同时出现。
- 例如，在一个事务型的DFD中，分支动作路径上的信息流也可能会体现出变换流的特征。
- 事务流和变换流组合出现，就是**混合型数据流**，简称**混合流**。

09:09

44

44

## 混合型数据流（补充）

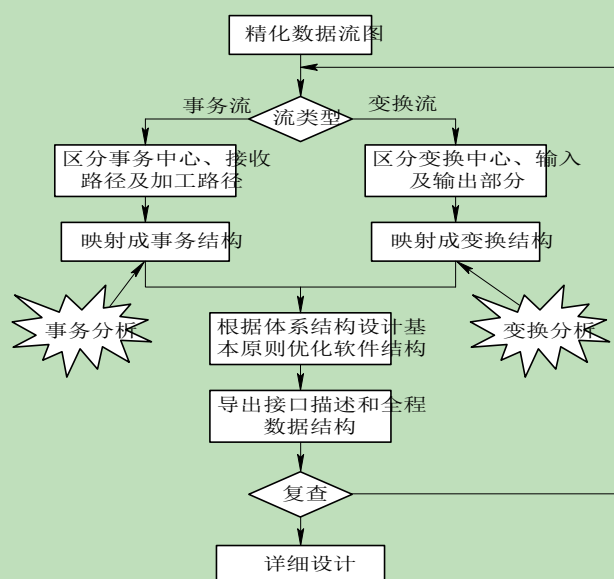


09:09

45

45

## 面向数据流的体系结构设计过程



09:09

46

46

# 变换分析设计

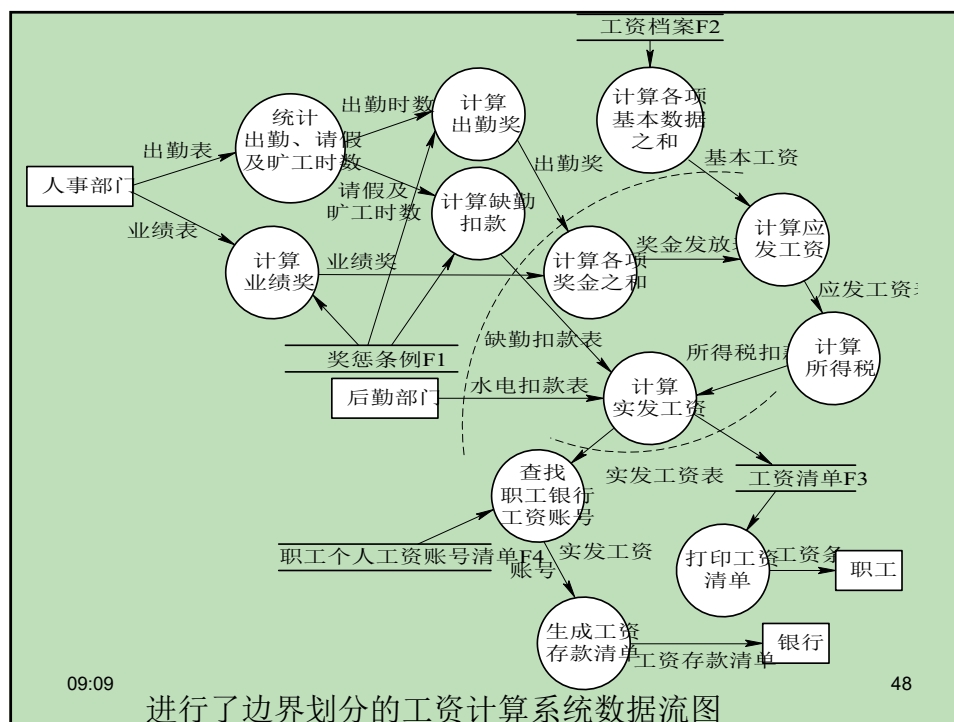
## 变换分析建立软件结构的步骤

### (1) 划分边界，区分系统的输入、变换中心和输出部分

- 从数据流图的物理输入端出发，沿着数据流方向逐步向系统内部移动，直至遇到不能被看作是系统输入的数据流为止，则此数据流之前的部分即为**系统的输入**；
- 从数据流图的物理输出端出发，逆着数据流方向逐步向系统内部移动，直至遇到不能被看作是系统输出的数据流为止，则该数据流之后的部分即为**系统的输出**；
- 夹在输入和输出之间的部分就是系统的**变换中心**。



47



进行了边界划分的工资计算系统数据流图

48



## 变换分析设计

### 变换分析建立软件结构的具体步骤

#### (2) 完成第一级分解，设计系统的上层模块

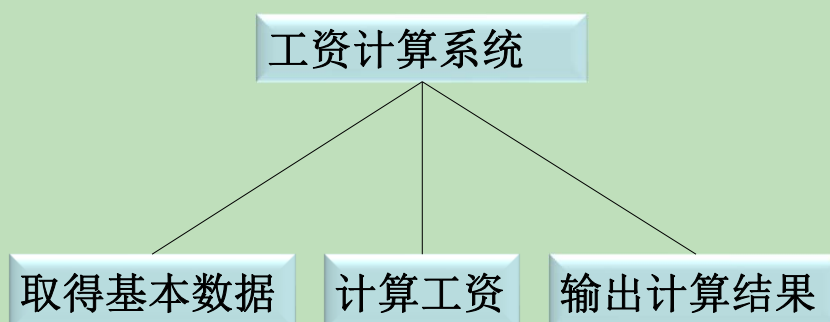
- 确定软件结构的顶层和第一层。
- 任何系统的顶层都只含一个用于控制的主模块。
- 变换型数据流图对应的软件结构的**第一层**一般由**输入**、**变换**和**输出**三种模块组成。

09:09

49

49

### 例：工资计算系统的一级分解



09:09

50

50

## 变换分析设计

### 变换分析建立软件结构的具体步骤

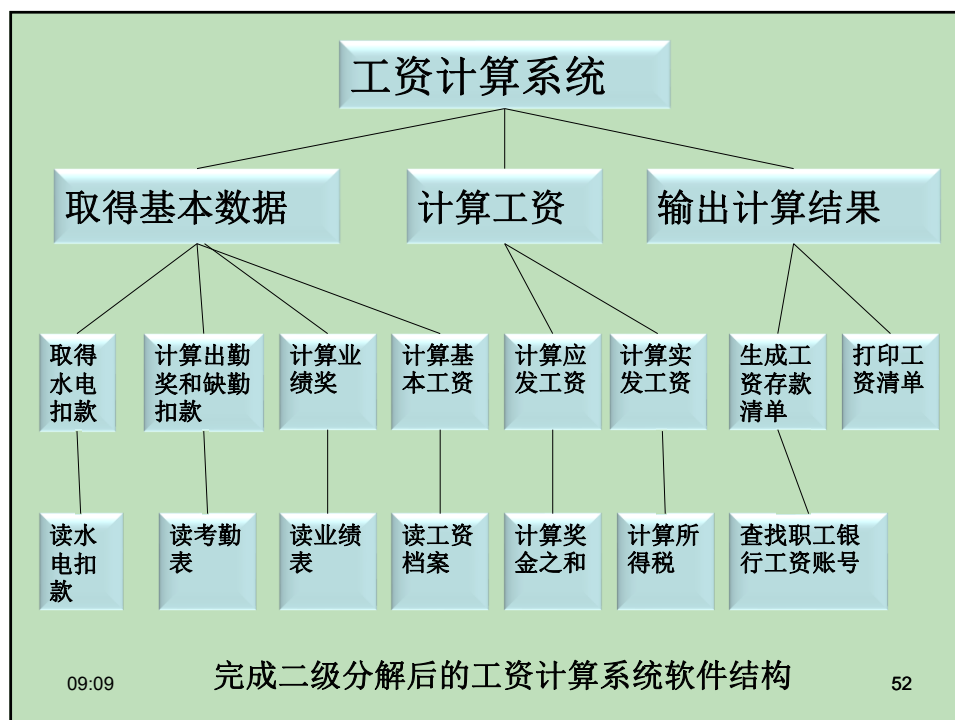
#### (3) 完成第二级分解，设计输入、变换中心和输出部分的中、下层模块

- 对上一步确定的软件结构进行逐层细化，为每一个输入、输出模块及变换模块设计下属模块。
- **输入模块**应包括用于**接收数据**和**转换数据**的两个下属模块；
- **输出模块**应包括用于**转换数据**和**传出数据**的两个下属模块；
- **变换模块**的分解没有固定的方法，一般应根据变换中心的组成情况及模块分解的原则来确定下属模块。

09:09

51

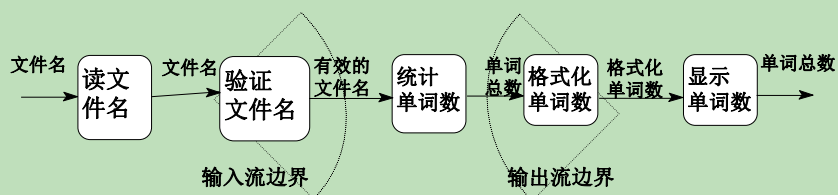
51



52

## 变换流设计

- 设计一个“统计输入文件中单词数目”程序

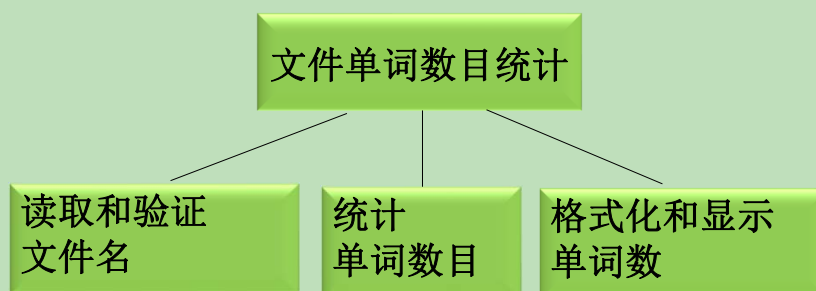


09:09

53

53

## 第一次分解

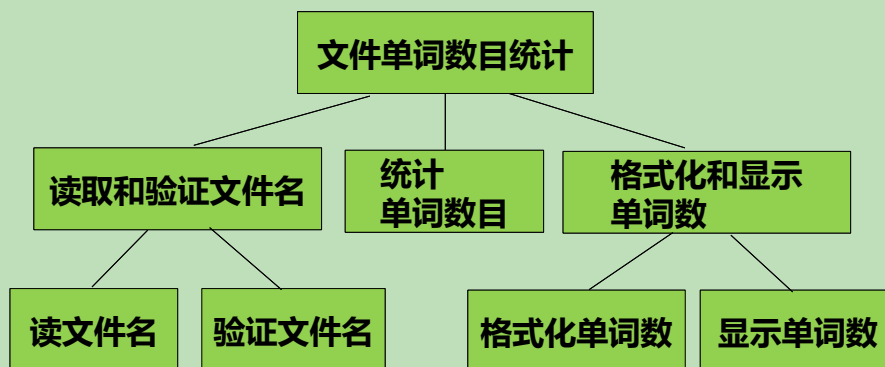


09:09

54

54

## 第二次分解



09:09

55

55

## 事务分析设计

事务分析设计方法生成软件结构的步骤：

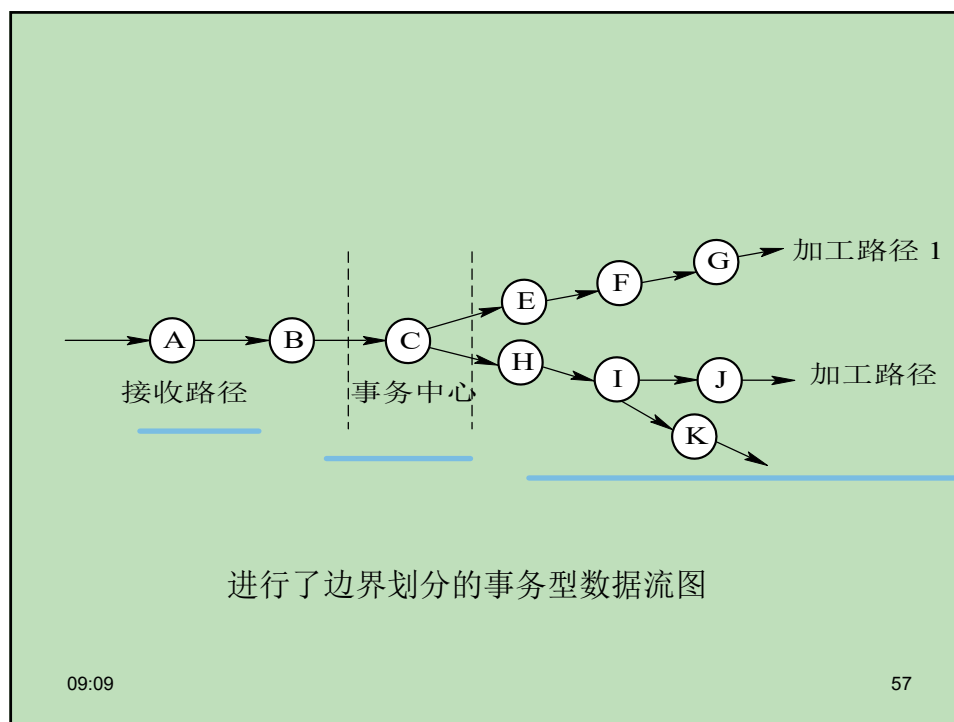
### (1) 划分边界，明确数据流图中的接收路径、事务中心和加工路径

- **事务中心**：位于多条加工路径的起点，经过事务中心的数据流被分解为多个发散的数据流。
- **接收路径**：向事务中心提供数据的路径。
- **加工路径**：从事务中心引出的所有路径。

09:09

56

56



57

## 事务分析设计

事务分析设计方法生成软件结构的步骤：

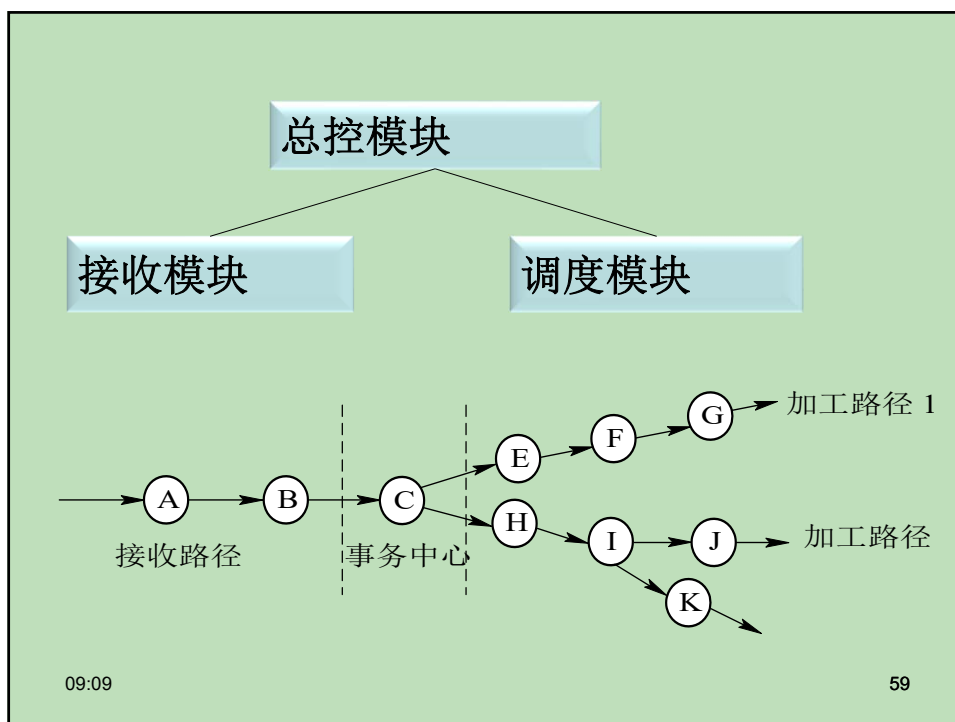
### (2) 建立事务型结构的上层模块

- 软件结构的顶层只有一个由事务中心映射得到的**总控模块**；
- 总控模块有两个下级模块，分别是由接收路径映射得到的**接收模块**和由全部加工路径映射得到的**调度模块**。
- **接收模块**负责接收系统处理所需的数据
- **调度模块**负责控制下层的所有加工模块。

09:09

58

58



59

## 事务分析设计

事务分析设计方法生成软件结构的步骤：

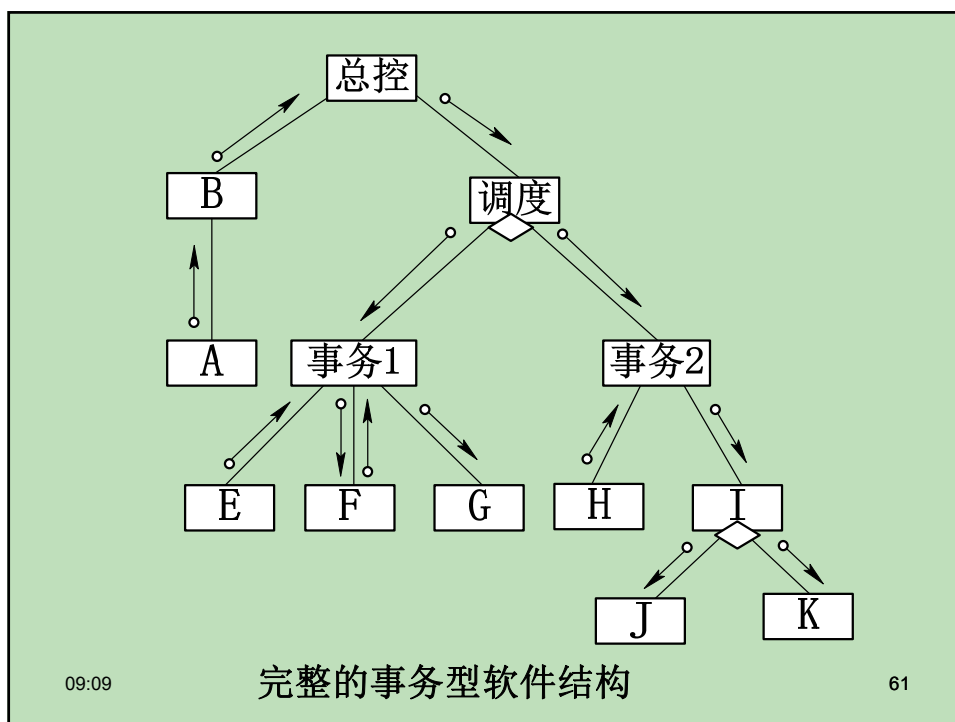
(3) **分解、细化接收路径和加工路径，得到事务型结构的下层模块。**

- 接收路径通常具有变换型特性，**分解方法同变换型结构输入模块的分解方法。**
- 加工路径的分解应按照每一条路径本身的结构特征，分别采用变换分析或事务分析方法进行分解。

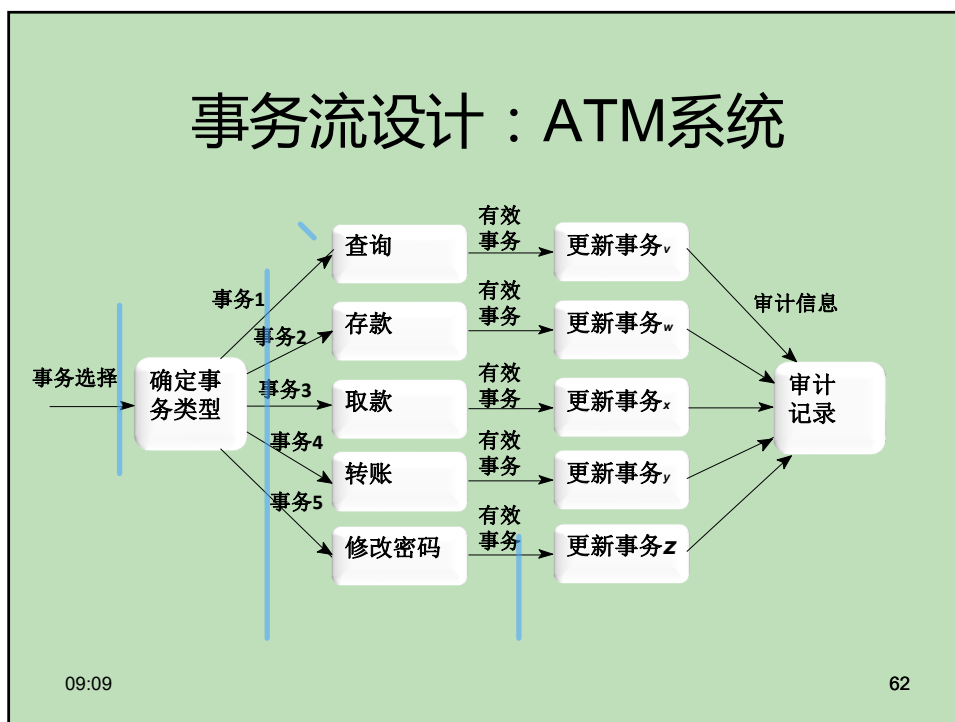
09:09

60

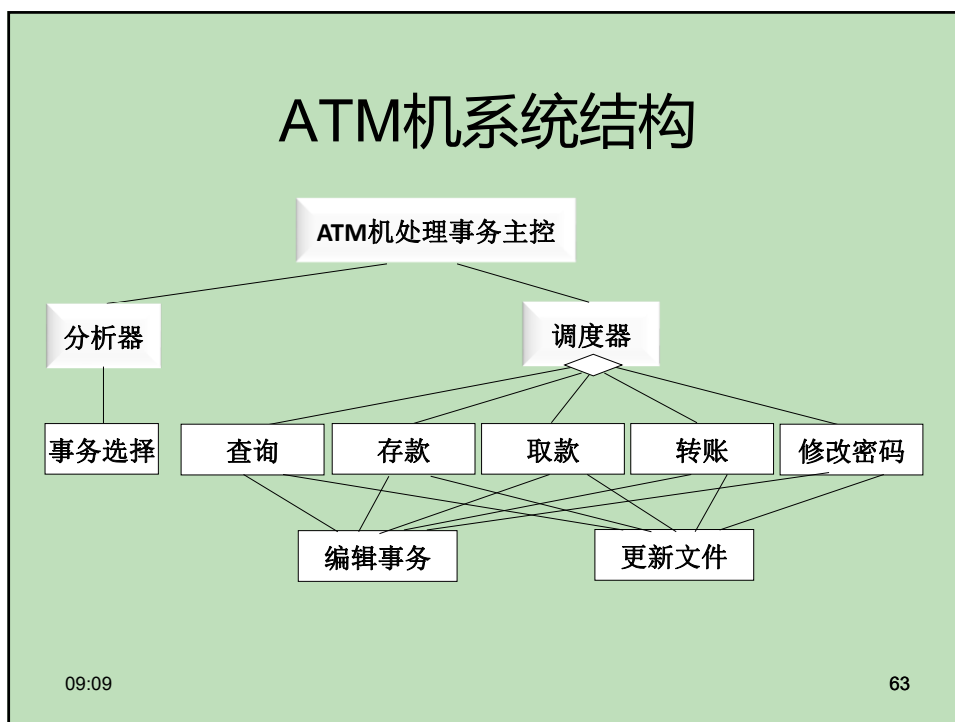
60



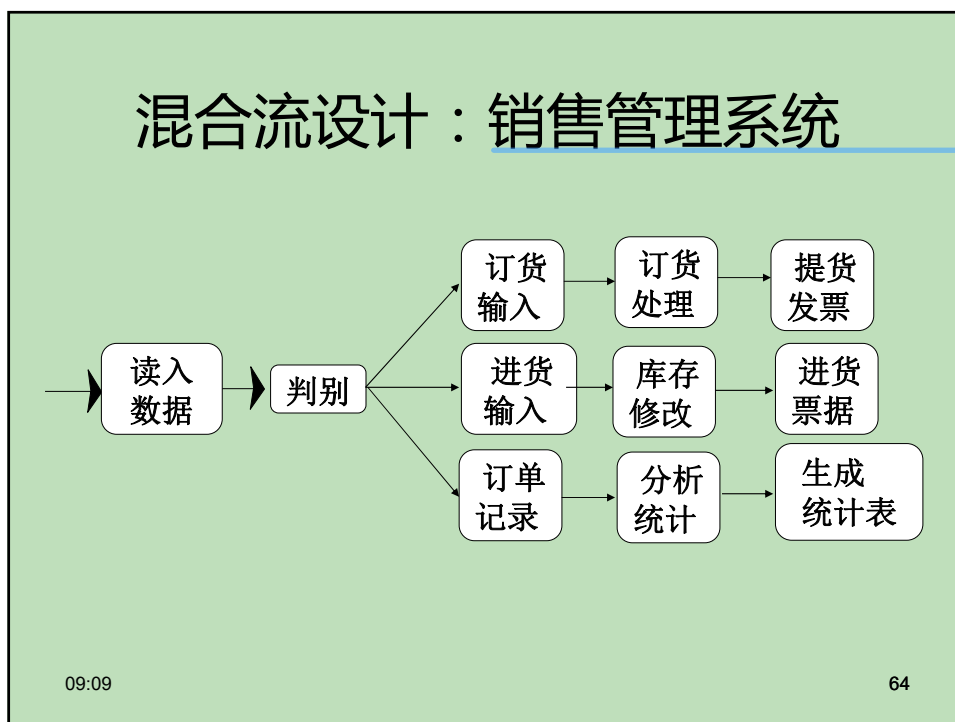
61



62



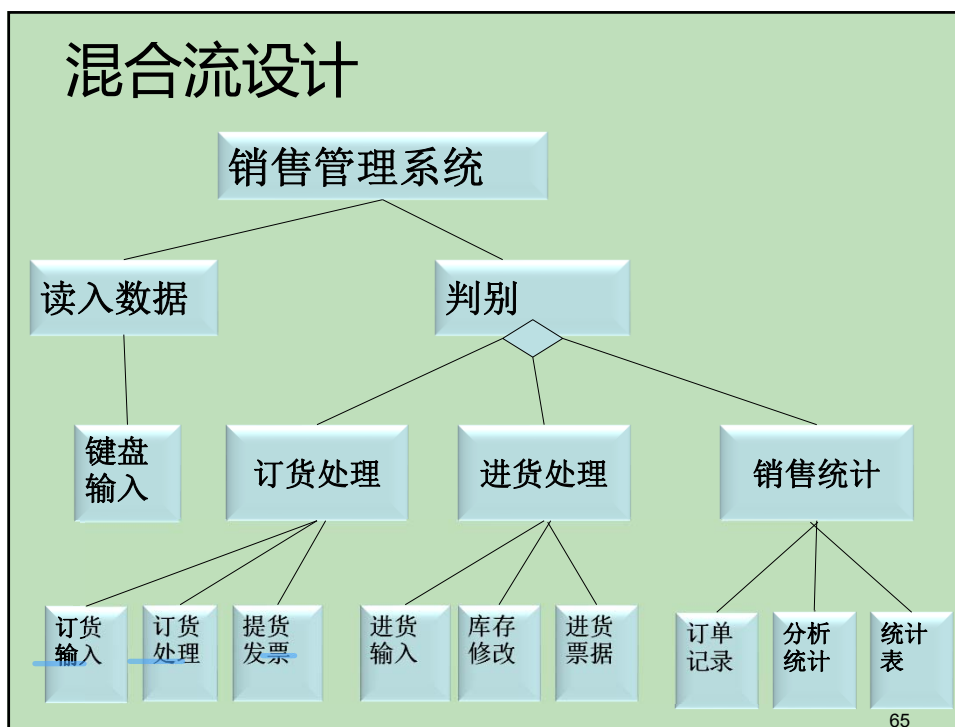
63



64



## 混合流设计



65

## 推荐视频学习

- <https://www.bilibili.com/video/BV1Q741157ve?p=66>
- <https://www.bilibili.com/video/BV1Q741157ve?p=67>
- <https://www.bilibili.com/video/BV1Q741157ve?p=68>
- <https://www.bilibili.com/video/BV1Q741157ve?p=69>
- <https://www.bilibili.com/video/BV1Q741157ve?p=70>
- <https://www.bilibili.com/video/BV1Q741157ve?p=71>

09:09

66

66

## 提纲

- 相关概念
- 体系结构设计任务、工具和原则
- 面向数据流的体系结构设计方法
- **详细设计的任务、工具和原则**
- 面向数据流的详细设计的方法
- 面向数据结构的设计方法

09:09

67

67

## 详细设计

### **详细设计的任务：**

- (1) 确定每个模块的具体算法
- (2) 确定每个模块的内部数据结构及数据库的物理结构
- (3) 确定模块接口的具体细节
- (4) 为每个模块设计一组测试用例
- (5) 编写文档，参加复审

09:09

68

68

## 详细设计的工具

程序流程图

N-S图

PAD图

PDL语言

09:09

69

69

## 程序流程图

- 能有效描述问题求解过程中的程序逻辑结构
- **优点**：对程序的控制流程描述直观、清晰，使用灵活，便于阅读和掌握
- **缺点**：
  - (1) 可以随心所欲地使用流程线，容易造成程序控制结构的混乱
  - (2) 难以描述逐步求精的过程，容易导致程序员过早考虑程序的控制流程，而忽略程序全局结构的设计。
  - (3) 难以表示系统中的数据结构

09:09

70

70

## 程序流程图常用符号

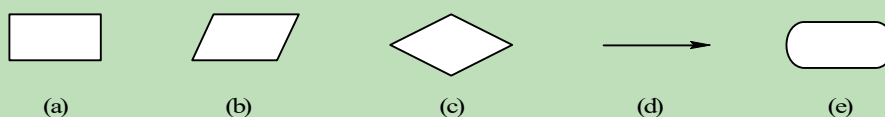


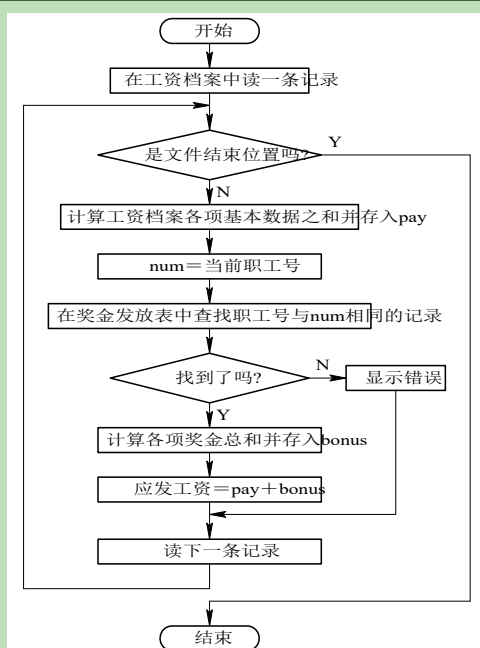
图5.17 程序流程图中的基本符号

(a) 一般处理框；(b) 输入/输出框；(c) 判断框；(d) 流程线；(e) 起止框

09:09

71

71



09:09

计算应发工资模块程序流程图

72

72

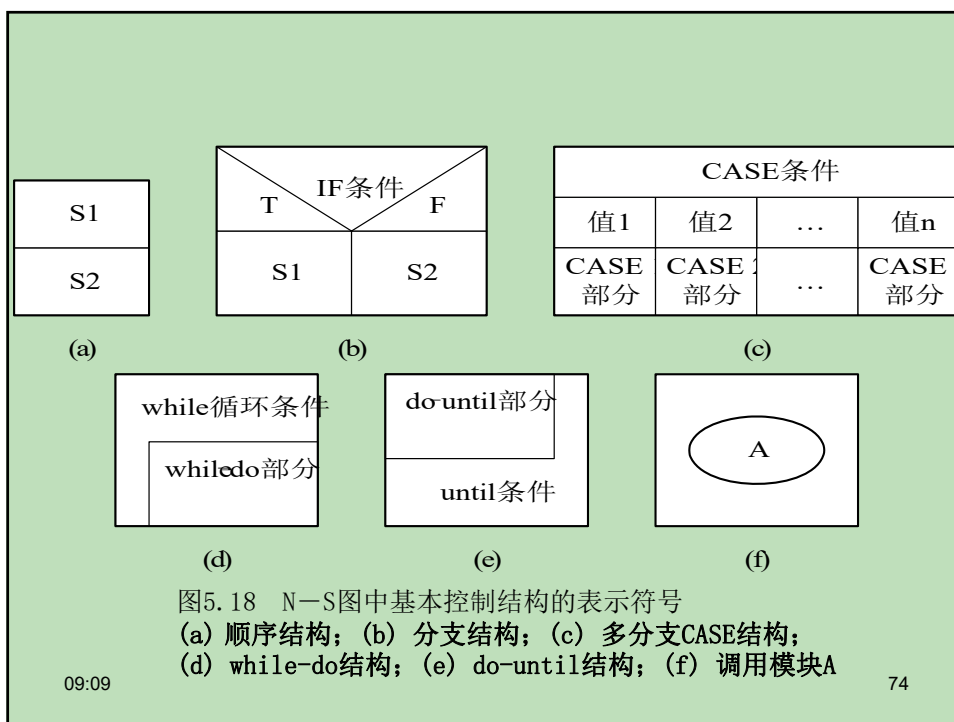
## N-S图

- 又称为盒图
- 所有的程序结构均使用矩形框表示
- **优点：**
  - 能清晰地表达结构中的**嵌套**及模块的**层次关系**
  - 有利于培养软件设计人员的良好设计风格。
- **缺点：**
  - 当所描述的程序嵌套层次较多时，N - S图的内层方框会越画越小，不仅影响可读性而且不易修改。

09:09

73

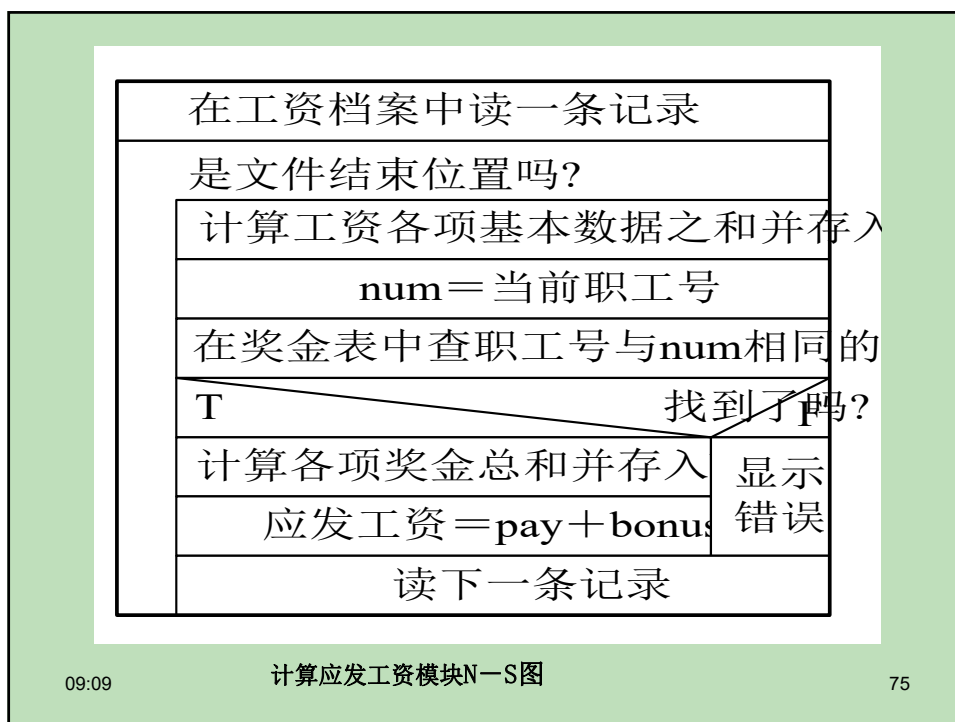
73



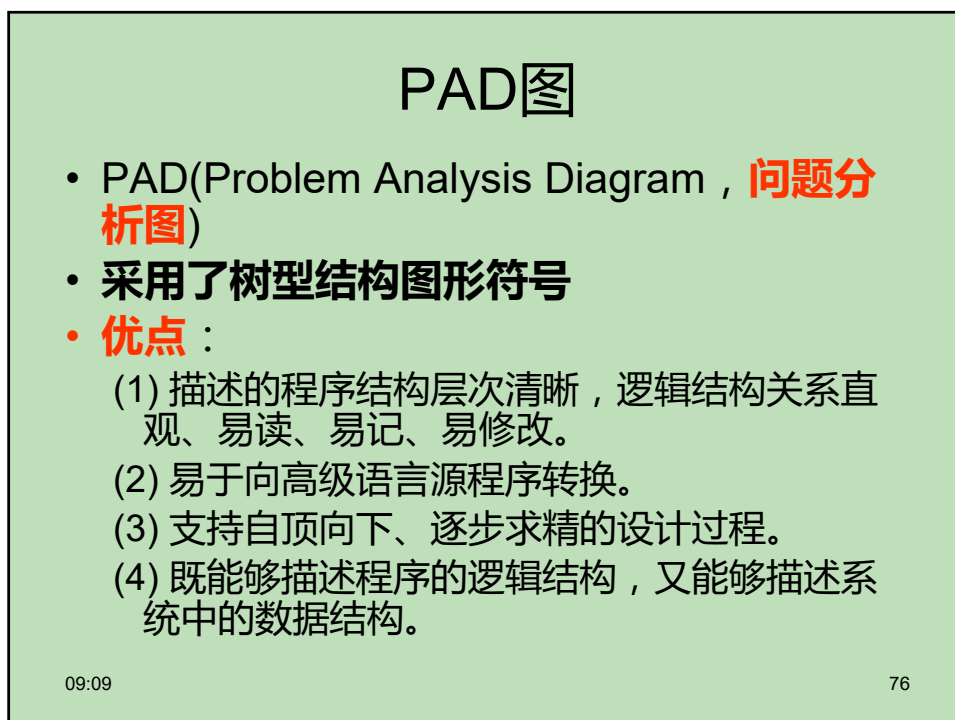
09:09

74

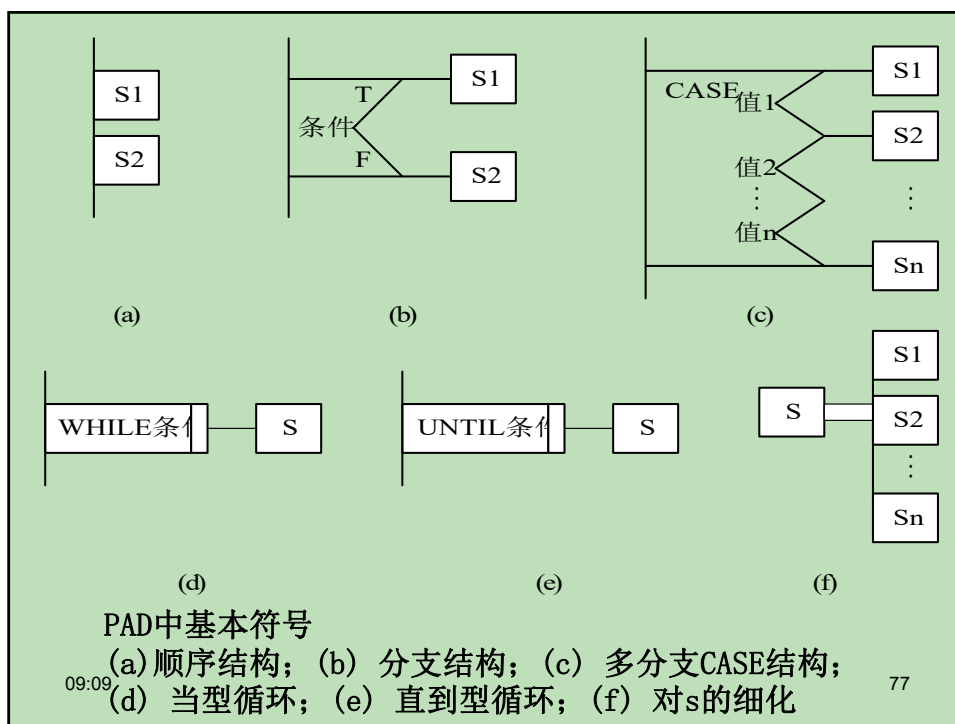
74



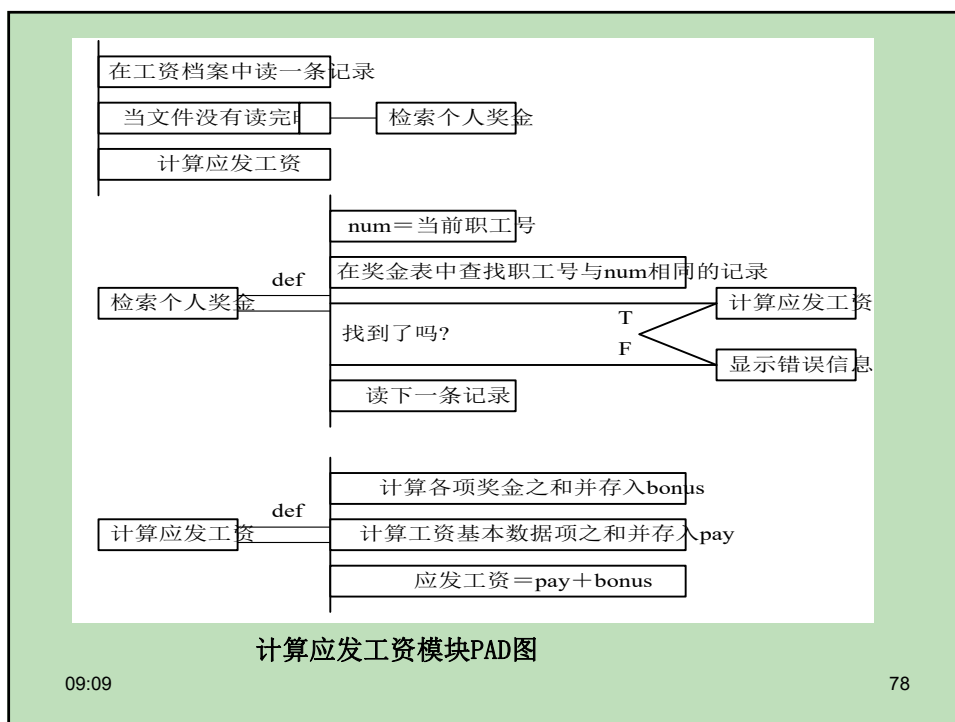
75



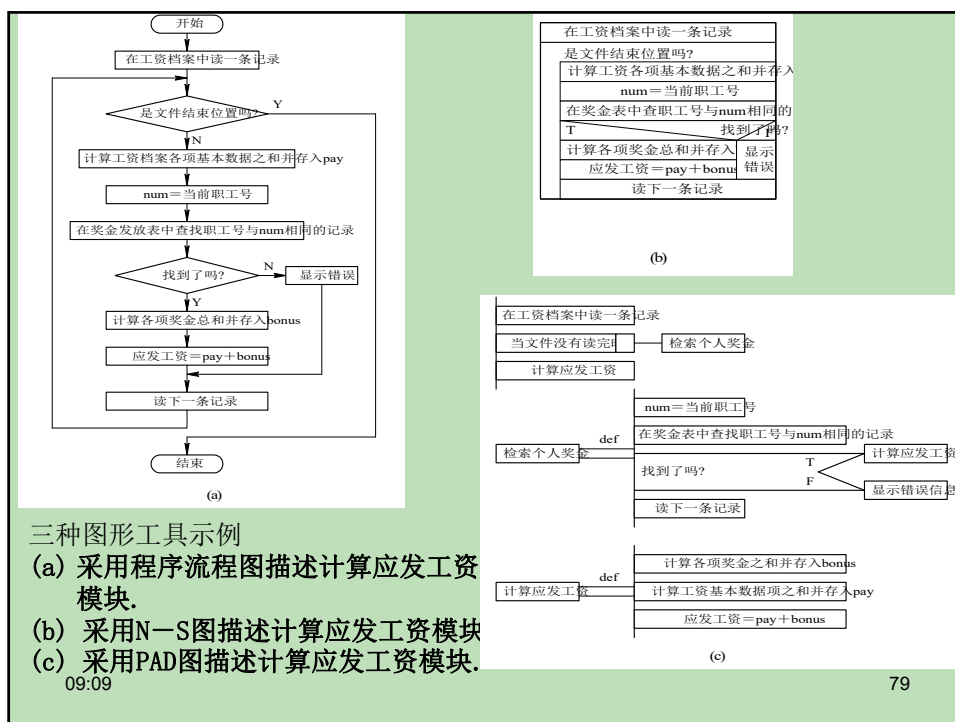
76



77



78



79

## PDL语言

- PDL (Process Design Language)语言，**过程设计语言**
- 用于描述程序算法和定义数据结构的伪代码
- 兼有自然语言和结构化程序设计语言语法的“混合型”语言。
- **PDL语言与结构化语言的主要区别在于：**
  - PDL语言表达的算法是编码的直接依据，其语法结构更加严格，处理过程描述更加具体详细。

09:09

80

80



## PDL语言的主要特点

- (1) **有严格的语法形式**，使程序结构、数据说明等更加清晰
- (2) **提供了数据说明机制**，可用于定义简单及复杂的数据结构
- (3) **提供了模块的定义和调用机制**，方便了程序模块化的表达

09:09

81

81

## PDL语言的主要定义语句

### (1) 数据定义：

DECLARE 属性 变量名， ...

**属性包括：**整型、实型、双精度型、字符型、指针、数组及结构等类型。

09:09

82

82

## PDL语言的主要定义语句

### (2) 模块定义:

PROCEDURE 模块名 (参数)

:

RETURN

END

09:09

83

83

## PDL语言的基本控制结构

### (1) 顺序结构:

顺序结构的语句序列采用自然语言进行描述。

语句序列S1

语句序列S2

语句序列Sn

09:09

84

84

## PDL语言的基本控制结构

### (2) 选择结构:

#### ① IF-ELSE结构

IF条件

语句序列S1

ELSE

语句序列S2

ENDIF

09:09

或:

IF条件

语句序列S

ENDIF

85

85

## PDL语言的基本控制结构

### ② 多分支IF结构

IF条件1

语句序列S1

ELSEIF条件2

语句序列S2

⋮

ELSE

语句序列Sn

ENDIF

09:09

86

86

## PDL语言的基本控制结构

### ③ CASE结构

```

CASE 表达式 OF
CASE 取值1
    语句序列S1
CASE 取值2
    语句序列S2
    ⋮
ELSE 语句序列Sn
ENDCASE

```

09:09

87

87

## PDL语言的基本控制结构

### (3) 循环结构:

#### ① FOR结构

```

FOR循环变量=初值 TO 终值
    循环体S
END FOR

```

#### ② WHILE结构

```

WHILE 条件
    循环体S
ENDWHILE

```

09:09

88

88

## PDL语言的基本控制结构

### ③ UNTIL结构

REPEAT

循环体S

UNTIL 条件

### 输入/输出语句

#### ① 输入语句：

GET (输入变量表)

#### ② 输出语句：

PUT (输出变量表)

### 模块调用语句

09:09

CALL 模块名(参数)

89

89

## 详细设计的原则

**(1) 将保证程序的清晰度放在首位**

**(2) 设计过程中应采用逐步细化的实现方法**

**(3) 选择适当的表达工具**

09:09

90

90

## 详细设计说明书

主要包括以下几方面的内容：

- (1) **引言**：用于说明编写本说明书的目的、背景，定义所用到的术语和缩略语，以及列出文档中所引用的参考资料等。
- (2) **总体设计**：用于给出软件系统的体系结构图。
- (3) **模块描述**：依次对各个模块进行详细的描述，主要包括模块的功能和性能，实现模块功能的算法，模块的输入及输出，模块接口的详细信息等。

09:09

91

91

## 提纲

- 相关概念
- 体系结构设计任务工具和原则
- 面向数据流的体系结构设计方法
- 详细设计的任务、工具和原则
- **面向数据流的详细设计的方法**
- 面向数据结构的设计方法

09:09

92

92

## 面向数据流的详细设计方法

### 关键技术：

#### (1) 设计过程中采用了自顶向下，逐步细分的方法

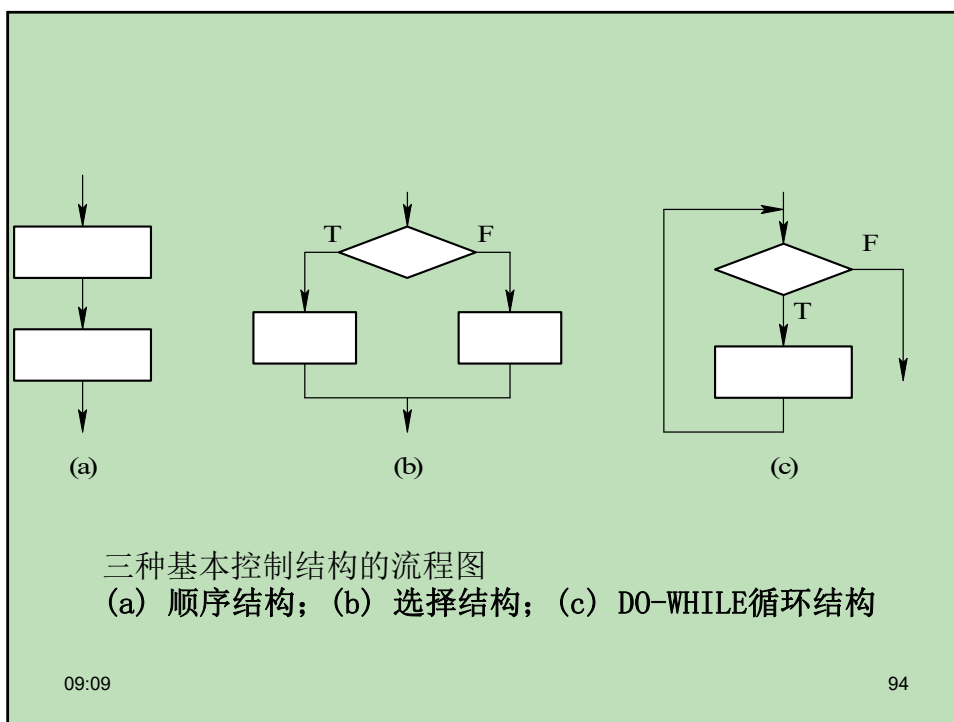
- 所有模块的实现都只采用单入口、单出口的三种基本控制结构

#### (2) 采用DO-UNTIL循环结构和多分支选择结构(DO-CASE)两种补充结构

09:09

93

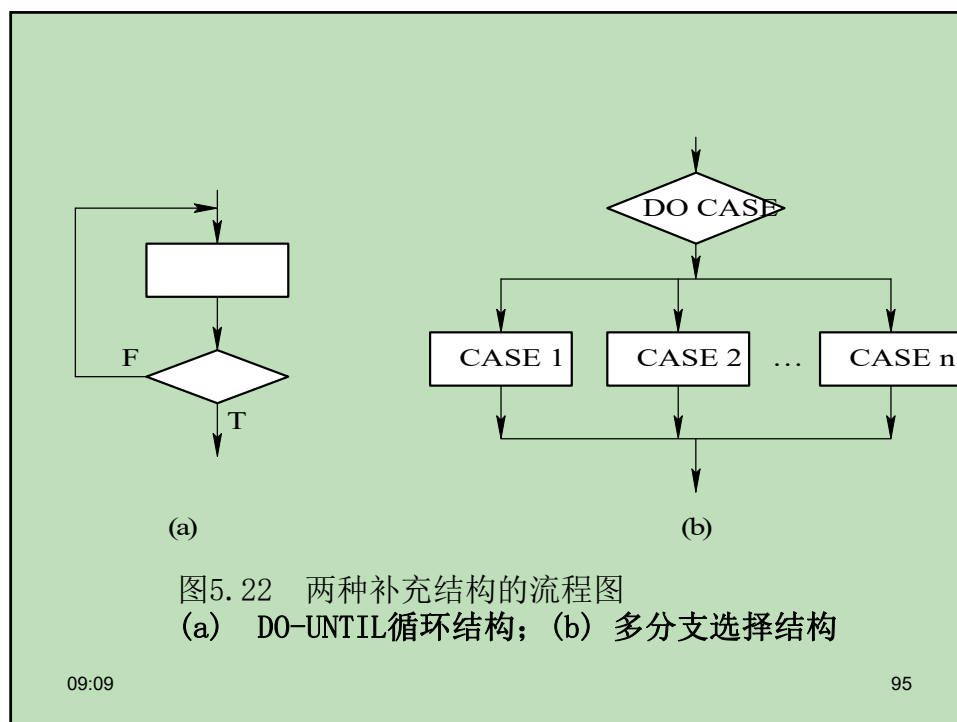
93



09:09

94

94



95

## 面向数据流的详细设计方法

- **优点：**
  - 自顶向下、逐步细分保证了程序的可靠性
  - 基本控制结构的使用则保证了程序的清晰易懂
- **缺点：**
  - 程序结构的清晰往往要以存储容量的增加和运行效率的降低为代价

09:09

96

96



## 提纲

- 相关概念
- 体系结构设计任务工具和原则
- 面向数据流的体系结构设计方法
- 详细设计的任务、工具和原则
- 面向数据流的详细设计的方法
- **面向数据结构的设计方法**

09:09

97

97

## 面向数据结构的设计方法

- **设计步骤：**
  - (1) 画出系统中输入、输出数据对应的数据结构图。
  - (2) 根据数据结构图，映射得到相应的程序结构图。
  - (3) 按照程序结构图，分析得到程序的详细过程性描述。

09:09

98

98

## Jackson图

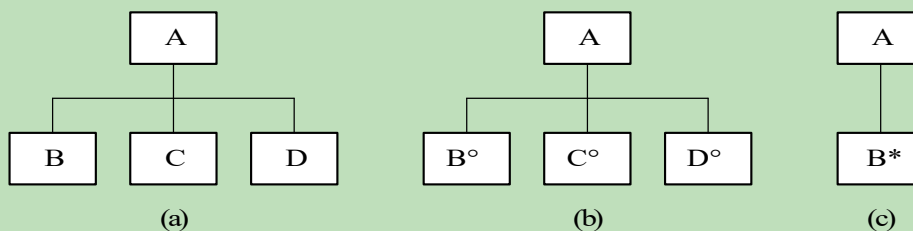
- Jackson图由**方框、连线及有特殊含义的一些标记**组成。
- 尽管数据结构种类繁多，但**数据元素之间的联系只有顺序、选择和循环三种**

09:09

99

99

- (a)图表示A由B、C和D三部分**顺序**组成；  
 (b)图表示A根据分支条件由B、C、D三个部分中选择一个，  
 “°”标记表示**选择**；  
 (c)图表示A由B重复若干次组成，“\*”标记表示**重复**。



三种基本结构在Jackson图中的表示符号  
 (a) 顺序结构；(b) 选择结构；(c)；循环结构

09:09

100

100

## Jackson方法

- 一种典型的面向数据结构的结构程序设计方法
- **设计目标**：从分析系统的数据结构出发，最后得出用Jackson伪代码表示的程序处理过程。
- **例**：  
假定某单位原来存在一个职工工资文件和一个职工档案文件，两个文件中的记录均按照职工编号升序排列且数目相等，现在要将这两个独立的文件合并为一个职工工资档案文件。采用Jackson方法设计，共分为如下四步进行：

09:09

101

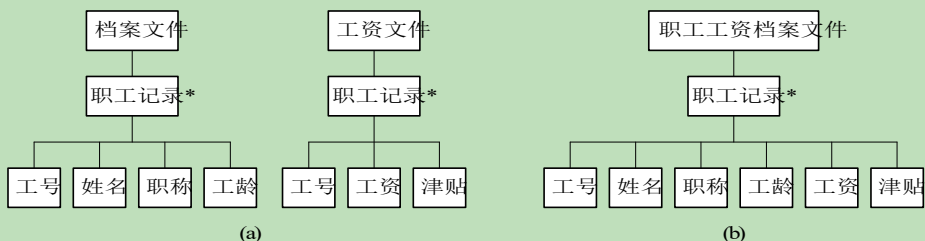
101

## Jackson方法

**(1) 分析问题，确定输入、输出数据的逻辑结构，并用Jackson图将其描述出来。**

**输入数据**：职工档案文件，职工工资文件

**输出数据**：职工工资档案文件



输入和输出数据结构

09:09 (a) 输入数据的数据结构；(b) 输出数据的数据结构

102

102

## Jackson方法

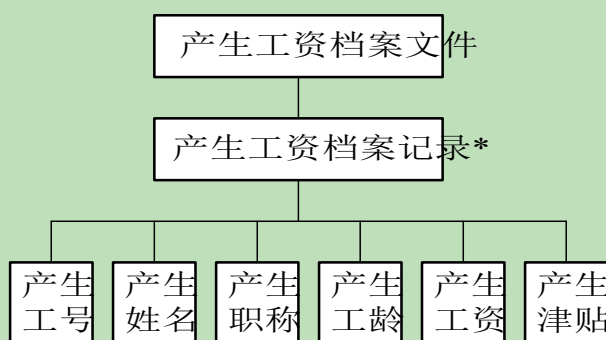
(2) 找出输入数据结构和输出数据结构中有对应关系的单元，并按下列规则导出描述程序结构的Jackson图。

- 为每对输入结构与输出结构中**有对应关系**的数据单元在程序结构图的相应层次画一个处理框。
- 为输入数据结构中剩余的每一个数据单元在程序结构图的相应层次画一个处理框。
- 为输出数据结构中剩余的每一个数据单元在程序结构图的相应层次画一个处理框。

09:09

103

103



生成工资档案文件的程序结构图

09:09

104

104

## Jackson方法

### (3) 列出完成结构图中各处理框功能的所有操作、分支及循环条件，并把它放到程序结构图上的适当位置。

例子中所涉及的基本操作和条件如下：

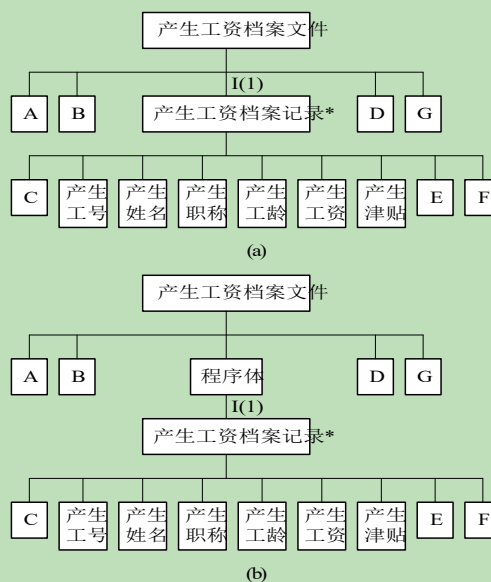
A. 打开输入文件；B. 新建工资档案文件；C. 读取输入文件中的一条记录；D. 关闭文件；E. 合并生成工资档案记录；F. 将工资档案记录写入文件；G. 终止；I(1). 输入文件未结束。

09:09

105

105

在图 (a)中的产生工资档案文件顺序结构中，混有**产生工资档案记录**这个循环结构。为了防止混淆，应将其改进为图(b)所示的程序结构图。



09:09 加入基本操作和条件后的程序结构图

(a) 改进前的程序结构图；(b) 改进后的程序结构图

106

106

## Jackson方法

### (4) 用Jackson伪代码写出程序的处理过程。

Jackson方法中使用的伪代码和Jackson图完全对应，以下是Jackson图中三种基本结构对应的伪代码表示。

- **顺序结构**：其中，seq和end是关键字。

```
A seq
    B
    C
    D
09:09 A end
```

107

107

## Jackson方法

- **选择结构**：其中，select、or和end是关键字；cond1、cond2和cond3分别是执行B、C或D的条件。

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

09:09

108

108

## Jackson方法

- **循环结构**：循环结构有until和while两种形式，其中，itel、until、while和end是关键字；cond是循环的条件。

```
A itel until(或while)cond
```

```
    B
```

```
A end
```

09:09

109

109

## Jackson方法

图4. 26 (b) 所示的程序结构图对应的伪代码表示如下：

```
产生工资档案文件seq
```

```
    打开两个输入文件
```

```
    新建工资档案文件
```

```
    程序体itel while 输入文件未结束
```

```
        产生工资档案记录seq
```

```
            从两个输入文件中各读取一条记录
```

```
            合并生成工资档案记录
```

```
            将生成的工资档案记录写入工资档案文件中
```

```
        产生工资档案记录end
```

```
    程序体end
```

```
    关闭所有文件
```

```
    终止
```

```
09:09 产生工资档案文件end
```

110

110

## 课程回顾

- 相关概念
- 体系结构设计任务工具和原则
- 面向数据流的体系结构设计方法
- 详细设计的任务、工具和原则
- 面向数据流的详细设计的方法
- 面向数据结构的设计方法

09:09

111

111

## 思考题

- 什么是模块？有哪些特征？
- 什么是模块化？
- 什么是信息隐蔽？
- 体系结构设计的原则有哪些？
- 数据流有哪些类型？如何区分？
- 变换流设计的步骤有哪些？
- 事务流设计的步骤有哪些？
- 详细设计的工具有哪些？

09:09

112

112



谢谢！

113

113