# 第1章 软件与软件工程的概念

**1.1 举出你所知道的应用软件的例子。**

办公软件、游戏软件、财务软件、银行软件、人事管理软件、工资管理软件、学籍管理软件等。

**1.2 认为“软件就是程序，软件开发就是编程序。”这种观点是否正确？为什么？**

认为“软件就是程序，软件开发就是编程序。”这种观点是错误的。

首先，软件是计算机系统中与硬件相互依存的另一部分，它是包括程序，数据及其相关文档的完整集合，程序只是软件的组成部分之一；其次，在软件开发中，编程只是软件开发过程的一个阶段。

**1.3 如果将软件开发比作高楼大厦的建造，可以将软件的设计比作什么？**

可以将软件的设计比作建筑设计，软件设计的成果相当于建筑设计的设计图纸。

**1.4 什么是软件危机？它有哪些典型表现？为什么会出现软件危机？**

**软件危机：** 软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。

**典型表现：**

1. 对软件开发成本和进度的估计常常很不准确。
2. 用户对“已完成的”软件系统不满意的现象经常发生。
3. 软件产品的质量往往靠不住。
4. 软件常常是不可维护的。
5. 软件通常没有适当的文档资料。
6. 软件成本在计算机系统总成本中所占的比例逐年上升。
7. 软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势。

**产生软件危机的原因：** 除了软件本身的特点，其原因主要有以下几个方面：

(1) 缺乏软件开发的经验和有关软件开发数据的积累，使得开发工作计划很难制定。

(2) 软件人员与用户的交流存在障碍，使得获取的需求不充分或存在错误。

(3) 软件开发过程不规范。如，没有真正了解用户的需求就开始编程序。

(4) 随着软件规模的增大，其复杂性往往会呈指数级升高。需要很多人分工协作，不仅涉及技术问题，更重要的是必须有科学严格的管理。

(5) 缺少有效的软件评测手段，提交给用户的软件的质量不能完全保证。

**1.5 什么是软件工程？**

软件工程是指导计算机软件开发和维护的工程学科。采用工程的概念、原理、技术和方法来开发和维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以经济地开发出高质量的软件并有效地维护它，这就是软件工程。

**1.6 简述软件生存期由哪些主要的阶段组成，每一阶段的主要任务是什么？**

1. 问题定义与可行性研究：问题定义必须回答的关键问题是：“要解决的问题是什么”。可行性研究要回答的关键问题是：“在成本和时间的限制条件下能否解决问题？是否值得做？”。
2. 需求分析：这个阶段的任务仍然不是具体地解决客户的问题，而是准确地回答“目标系统必须做什么”这个问题。
3. 软件设计：本阶段要回答的关键问题是“目标系统如何做？”为此，必须在设计阶段中制定设计方案，把已确定的各项需求转换成相应的软件体系结构。结构中的每一组成部分都是意义明确的构件，此即所谓概要设计。进而具体描述每个构件所要完成的工作，为源程序编写打下基础，此即所谓详细设计。
4. 程序编码和单元测试：本阶段要解决的问题是“正确地实现已做的设计”，为此，需要选择合适的编程语言，把软件设计转换成计算机可以接受的程序代码，并对程序结构中的各个模块进行单元测试，然后运用调试的手段排除测试中发现的错误。要求编写出的程序应当是结构良好、清晰易读的，且与设计相一致的。
5. 集成测试和系统测试：集成测试的任务是将已测试过的模块按设计规定的顺序组装起来，在组装的过程中检查程序连接的问题。系统测试的任务是根据需求规格说明的要求，对必须实现的各项需求，逐项进行确认，判定已开发的软件是否符合用户需求，能否交付用户使用。为了更有效地发现系统中的问题，通常这个阶段的工作由开发人员、用户之外的第三者承担。
6. 运行维护：已交付的软件投入正式使用，便进入运行维护阶段。这一阶段可能持续若干年。软件在运行中可能由于多方面的原因，需要对它进行维护。通常有四种类型的维护：改正性维护、适应性维护、完善性维护和预防性维护。

**1.7 常见的软件生存期模型主要有哪些？每种模型的优缺点是什么？**

常见的软件生存期模型主要有瀑布模型、快速原型模型、增量模型、螺旋模型、喷泉模型和统一过程。

1. **瀑布模型**

**优点：**可强迫开发人员采用规范化的方法；严格地规定了每个阶段必须提交的文档；要求每个阶段交出的所有产品都必须经过质量保证小组的仔细验证。

**缺点：**由于瀑布模型几乎完全依赖于书面的规格说明，很可能导致最终开发出的软件产品不能真正满足用户的需要；用户往往需要等待很长时间才能看到可以运行的程序；适应需求变更的能力比较差。

**适用范围：**瀑布模型只适用于项目开始时需求已确定的情况。

**(2) 快速原型模型**

**优点：**有助于满足用户的真实需求；原型系统已经通过与用户的交互而得到验证，据此产生的规格说明文档能够正确地描述用户需求。

**缺点：**要求开发人员快速建立原型。

**适用范围：**适用于需求不明确的软件项目。

**(3) 增量模型**

**优点：**

* 能在较短时间内向用户提交可完成一些基本功能的产品，即从第一个构件交付之日起，用户就能做一些有用的工作。
* 逐步增加产品的功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给用户组织带来的冲击。
* 项目失败的风险较低，虽然在某些增量构件中可能遇到一些问题，但其他增量构件将能够成功地交付给客户。
* 优先级最高的服务首先交付，然后再将其他增量构件逐次集成进来。因此，最重要的系统服务将接受最多的测试。

**缺点：**

* 在把每个新的增量构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品。
* 软件体系结构必须是开放的，即向现有产品中加入新构件的过程必须简单、方便。

**适用范围：**适用于工期紧张、功能可以划分、比较复杂的软件项目。软件工程师必须有较高的技术水平，能够设计出开放的软件体系结构。

**(4) 螺旋模型**

**优点：**

* 对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标；
* 减少了过多测试或测试不足所带来的风险；
* 在螺旋模型中维护只是模型的另一个周期，在维护和开发之间并没有本质区别。

**缺点：**螺旋模型是风险驱动的，因此要求软件开发人员必须具有丰富的风险评估经验和这方面的专门知识，否则将出现真正的风险：当项目实际上正在走向灾难时，开发人员可能还以为一切正常。

**适用范围：**适用于内部开发的大型软件（开发周期长、比较容易受到社会因素影响的软件项目），软件开发人员具有丰富的风险评估知识和经验。

**(5) 喷泉模型**

**优点**：在开发过程中使用统一的概念和方法，各阶段之间没有明显的界限，容易实现各个开发过程的多次反复迭代，达到了认识的逐步深化，降低了出错的可能性。

**缺点**：由于各个阶段之间的界线不明确，容易造成软件开发过程的无序。

**适用范围：**适用于面向对象的软件开发。

**(6) 统一过程**

**优点：**提高了团队生产力，在迭代的开发过程、需求管理、基于组件的体系结构、可视化软件建模、验证软件质量及控制软件变更等方面，针对所有关键的开发活动为每个开发成员提供了必要的准则、模板和工具指导，并确保全体成员共享相同的知识基础。它建立了简洁和清晰的过程结构，为开发过程提供了较大的通用性。

**缺点**：RUP只是一个开发过程，并没有涵盖软件过程的全部内容，例如它缺少关于软件运行和支持等方面的内容；此外，它没有支持多项目的开发结构，这在一定程度上降低了在开发组织内大范围实现重用的可能性。

**适用范围：**适用于基于构件的软件开发。

# 第2章 软件工程方法与工具

**2.1 软件工程的三种基本要素是什么，各自的作用是什么？**

软件工程的三种基本要素是方法、工具和过程。

(1) 软件工程方法（method）为建造软件提供技术上的解决方法（“如何做”）。目前使用得最广泛的方法是传统方法（结构化方法）和面向对象方法。

(2) 工具为方法的运用提供自动的或半自动的软件支撑环境。

(3) 过程是为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

**2.2 简述传统方法和面向对象方法的特点。**

(1) 传统方法的特点

传统方法也称为生命周期方法或结构化范型。它采用结构化技术来完成软件开发的各项任务。这种方法学把软件生命周期的全过程依次划分为若干个阶段，然后顺序地逐步完成每个阶段的任务。每一个阶段的开始和结束都有严格的标准，对于任何两个相邻的阶段而言，前一个阶段的结束标准就是后一阶段的开始标准。

传统方法的主要缺点是在适应需求变化方面不够灵活，另外，结构化方法要么面向行为，要么面向数据，缺乏使两者有机结合的机制。

（2）面向对象方法的特点

面向对象方法把数据和行为看成同等重要，是将数据和对数据的操作紧密地结合起来的方法，这也是面向对象方法与传统方法的重要区别。

面向对象方法的出发点和基本原则，是尽量模拟人类习惯的思维方式，使开发软件的方法和过程尽可能接近人类认识问题和解决问题的方法与过程，从而使描述问题的问题空间与其解空间在结构上尽可能一致。对于大型、复杂及交互性比较强的系统，使用面向对象方法更有优势。

**2.3 形式化方法的特点是什么？**

形式化方法的主要特点是：

(1) 软件需求规格说明被细化为用数学记号表达的详细的形式化规格说明；

(2) 设计、实现和单元测试等开发过程由一个变换开发过程代替。通过一系列变换将形式的规格说明细化成为程序。

**2.5 安装Rational Rose2000/2002/2003，并练习基本操作。**

略。

# 第3章 软件需求获取与结构化分析方法

**3.1 为什么结构化分析要叫“结构化”？有其他可替代的术语吗？**

“结构化”一词应来源于“结构化程序设计”，先有“结构化程序设计”的思想，后有结构化设计及结构化分析。有时称传统的分析方法，指的就是结构化分析方法。

**3.2顶层数据流图或称环境图的作用是什么？**

顶层数据流图（或称环境图）仅包括一个数据处理过程，也就是要开发的目标系统。其作用如下：

(1) 确定系统在其环境中的位置，与系统有联系的外部实体（包括硬件、软件、组织机构及人）有哪些。

(2) 通过确定系统的输入和输出与外部实体的关系确定系统的边界，也就是要确定哪些功能或处理属于系统范围之内，哪些属于系统范围之外，需要由其他系统处理或人工处理。

**3.3 在对数据流图进行分解时需要注意哪些问题？**

在对数据流图进行分解时，需要注意以下两个问题：

(1) 当对数据流图分层细化时必须保持信息连续性，也就是说，当把一个处理分解为一系列处理时，分解前和分解后的输入/输出数据流必须相同。

(2) 注意分层细化时对编号的处理方法。

**3.4 银行存款业务如下：客户到银行柜台存款时，首先填写存款单（包括帐号、姓名、存款金额、存款类型），如果还没有开户，则需要先开户，填写开户单（包括姓名、身份证号、地址、电话、是否留密码）。填写完存款单或开户单后，交给营业员，营业员将存款单或开户单信息输入计算机，系统记录开户信息或存款单信息，如果是开户并选择留密码，则需要客户输入密码。最后印出开户单或存款单给用户。请画出银行存款业务的分层数据流图（至少画出两层）。**

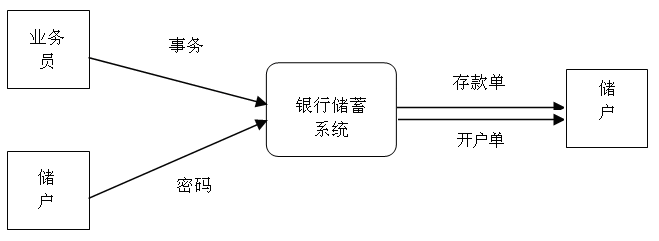
(1) 识别外部实体及输入输出数据流

外部实体：储户、业务员。

输入数据流：存款单，开户单，密码。可以将存款单和开户单抽象为事务。

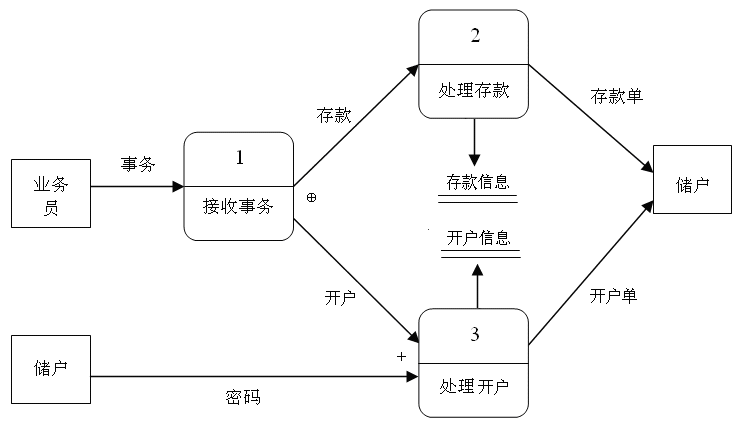
输出数据流：存款单，开户单。

(2) 顶层数据流图



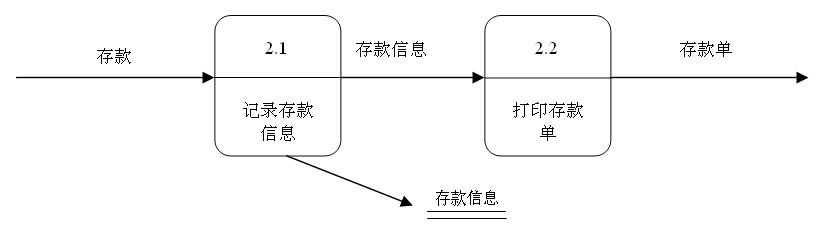
(3) 一层数据流图

对银行储蓄系统进行分解，从大的方面分解为接收事务、处理开户、处理存款三部分，得到一层数据流图。

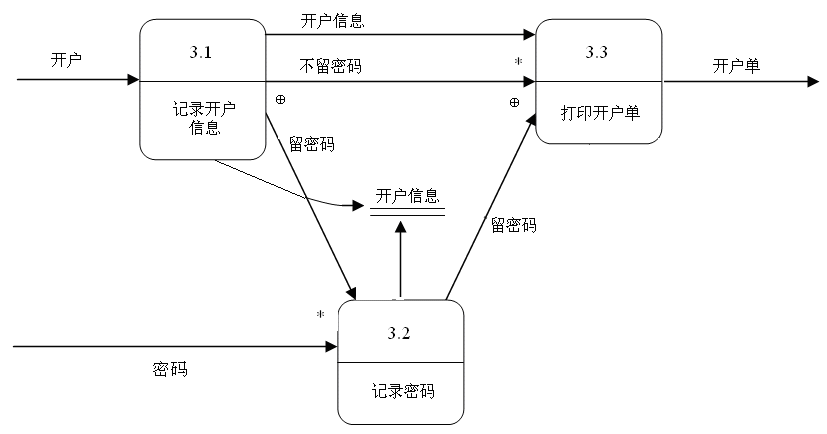


(4) 二层数据流图

对 “处理存款”及“处理开户”进行进一步分解，得到二层数据流图，即处理存款的数据流图和处理开户的数据流图。



处理存款的数据流图



处理开户的数据流图

**3.5 按照以下描述，画出ER图。**

**一本教材由许多章组成，每一章包含许多节、小结和习题组成，章和节都具有标题和序号属性。**

ER图如下：

教材

章

节

小结

习题

序号

标题

序号

标题

# 第4章 结构化设计方法

**4.1 当你“编写”程序时你设计软件吗？软件设计和编码有什么不同吗？**

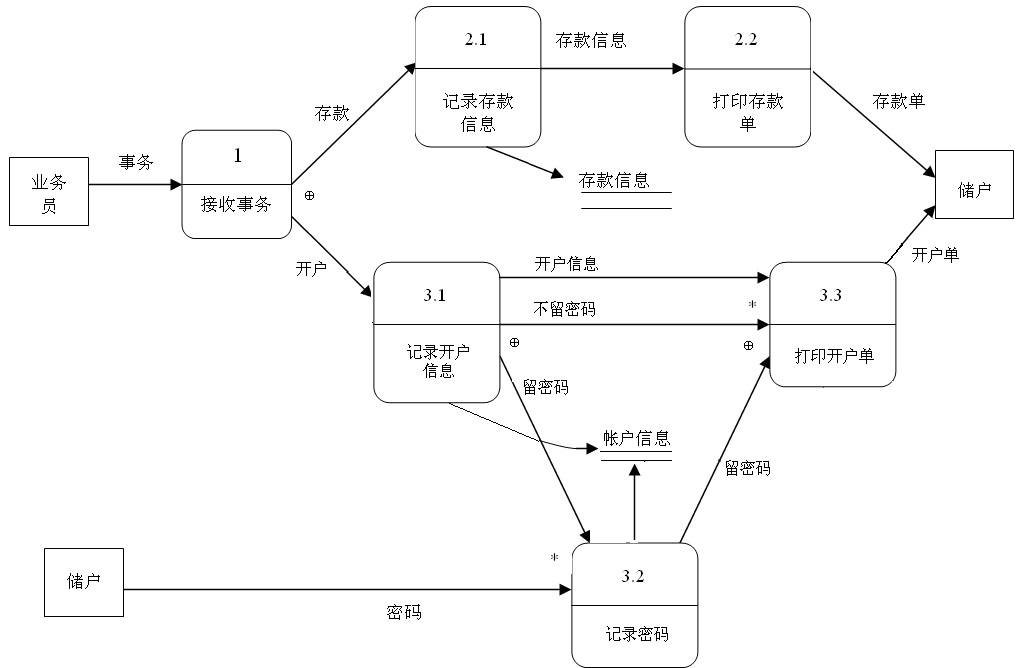
在“编写”程序时并没有设计软件。软件设计包括概要设计和详细设计，编码是将详细设计中的过程描述转换成用程序设计语言来描述。

**4.4 是否存在一种情况：复杂问题需要较少的工作去解决？这样的情况对模块化观点有什么影响？**

通过对复杂的问题进行合理分解，分解为若干个相对简单及独立的子问题，就可以用较少的工作去解决。这种情况能够较好地支持模块化的观点，每个子问题用单独的模块去解决，模块之间应该是高内聚、低耦合的，这样才能减少工作量，否则，虽然每个模块的工作简单了，但模块之间的联系很复杂，也增加了问题解决的难度和工作量。

**4.8 用面向数据流的方法设计第3章习题3.4所描述的银行存款业务的软件结构，并使用改进方法对模块结构进行精化。**

(1) 对第3章习题3.4给出的数据流图进行精化，确定其边界，如下图所示。



(2) 对上图按事务型数据流进行处理，完成第一级分解，得到顶层和一层模块结构图。

存款业务

输入数据

调度

输出数据

第一级分解后的结构图

(3) 完成第二级分解。对上图所示的“输入数据”、“输出数据”和“调度”模块进行分解，得到未经精化的输入结构、输出结构和事务结构。

输入数据

输入事务

输入密码

未经精化的输入结构

输出数据

打印存款单

打印开户单

未经精化的输出结构

调度

处理存款

处理开户

记录存款信息

记录开户信息

记录密码

未经精化的事务结构

将上面的三部分合在一起，得到初始的软件结构，如下图所示。

存款业务

输入数据

调度

输出数据

输入事务

输入密码

打印存款单

打印开户单

处理存款

处理开户

记录存款信息

记录开户信息

记录密码

初始软件结构图

(4) 对软件结构进行精化。

1) 由于调度模块下只有两种事务，因此，可以将调度模块合并到上级模块中，如图所示。

存款业务

输入数据

输出数据

输入事务

输入密码

打印存款单

打印开户单

处理存款

处理开户

记录存款信息

记录开户信息

记录密码

将调度模块合并到上级模块后的软件结构

2) “记录密码”模块的作用范围不在其控制范围之内（即“输入密码”模块不在“记录密码”模块的控制范围之内），需对其进行调整，如图所示。

存款业务

输入事务

输出数据

输入密码

打印存款单

打印开户单

处理存款

处理开户

记录存款信息

记录开户信息

记录密码

3) 提高模块独立性，对模块结构进行调整，如下图所示。

存款业务

输入事务

输入密码

打印存款单

打印开户单

处理存款

处理开户

记录存款信息

记录开户信息

记录密码

调整后的模块结构图

**4.9 将大的软件划分成模块有什么好处？是不是模块划分得越小越好？划分模块的依据是什么？**

将大的软件划分成独立命名且可独立访问的模块，不同的模块通常具有不同的功能或职责。这种方法有利于将复杂的问题简单化，是分而治之策略的具体表现。

尽管模块分解可以简化要解决的问题，但模块分解并不是越小越好。当模块数目增加时，每个模块的规模将减小，开发单个模块的成本确实减少了；但是，随着模块数目增加，模块之间关系的复杂程度也会增加，设计模块间接口所需要的工作量也将增加。

划分模块的依据是，模块只具有单一的功能且与其他模块没有太多的联系。

**4.11 结构化程序设计禁止使用goto语句吗？如果程序中使用了goto语句，是否就可以断定它是非结构化的？**

结构化程序设计并不禁止使用goto语句。如果程序中使用了goto语句，并不能断定它是非结构化的。

**4.12 对于给定的算法，如何判断它是否是结构化的？**

对于给定的算法，如果符合以下三条原则，就可以判断它是结构化的。

(1) 使用语言中的顺序、选择、重复等有限的基本控制结构表示程序逻辑。  
 (2) 选用的控制结构只准许有一个入口和一个出口。  
 (3) 程序语句组成容易识别的块（Block），每块只有一个入口和一个出口。

**4.13 对于图4-49所示的流程图，试分别用N-S图和PAD表示之。**

F

START

A

B

Q

END

T

F

T

P

图4-49 流程图

对应的N-S图如下： 对应的PAD如下：

while P

A

until !Q

B

while P

A

B

until !Q

**4.14 图4-50所示的流程图完成的功能是使用二分查找方法在table数组中找出值为item的数是否存在。**

**(1) 判断此算法是否是结构化的，说明理由。**

**(2) 若算法是非结构化的，设计一个等价的结构化算法，并用N-S图表示。**

F

F

F

T

F

F

table(i)>item

(finish-start)>1

table(start)=item

table(finish)=item

flag=0

flag=1

i＝(start＋finish)/2

table(i)=item

table(I)<item

finish=i-1

start=i+1

T

T

T

F

T

T

开始

结束

图4-50 二分查找算法的流程图

(1) 不是结构化的，最上面的循环有两个出口，最下面的分支有三个入口。

(2) 等价的结构化算法如下：

(FINISH-START)>1 && TABLE(I)!=ITEM

I＝(START＋FINISH)/2

TABLE(I)<ITEM

I＝(START＋FINISH)/2

T

F

START=I+1

FINISH=I-1

TABLE(START)==ITEM || TABLE(FINISH) ==ITEM || TABLE(I)==ITEM

T

F

FLAG=1

FLAG=0

或者

(FINISH-START)>1 && FLAG==0

I＝(START＋FINISH)/2

TABLE(I)==ITEM

FLAG＝0

T

F

FLAG=1

TABLE(I)<ITEM

T F

TABLE(START)==ITEM || TABLE(FINISH) ==ITEM

T

F

FLAG=1

START=I+1

FINISH=I-1

或者

FLAG＝0

(FINISH-START)>=0 && FLAG==0

I＝(START＋FINISH)/2

F

T

TABLE(I)==ITEM

TABLE(I)<ITEM

T F

FLAG=1

FINISH=I-1

START=I+1

**4.15 使用自顶向下、逐步细化方法设计算法，完成下列任务：产生一个10×10的二维随机整数方阵，先求出每一行的最大值和每一列的最小值；然后求10个最大值中的最小者，10个最小值中的最大者；最后求这两个数之差的平方。**

(1) 首先写出下面的程序框架：

main （） {  
 定义10×10的二维整数数组A，长度为10的一维数组B, C;

建立10×10的二维随机整数数组A；- - - - - - -- -- - - - - - - - - - - - 1  
 求A中每一行的最大值🡺数组B； - - - - - - - - -- - - - - - - - - - - - 2

求A中每一列的最小值🡺数组C； - - - - - - - - -- - - - - - - - - - - - 3

求数组B中的最小值🡺minOfB; - - - - - - - - -- - - - - - - - - - - - 4

求数组C中的最大值🡺maxOfC; - - - - - - - - -- - - - - - - - - - - - 5

(minOfB- maxOfC)^2 🡺result;

}  
(2) 对后面加标记的部分进行细化

main （） {  
 定义10×10的二维整数数组A，长度为10的一维数组B, C;

/\*建立10×10的二维随机整数数组A\*/ - - - - - -- -- - - - - - - - - - - - 1

for (i=0; i<=9; i++)

for (j=0; j<=9; j++)

产生随机整数🡺 A[i,j];  
 /\*求A中每一行的最大值🡺数组B\*/ - - - - - - - - -- - - - - - - - - - - - 2

for (i=0; i<=9; i++) {

求数组A第i行的最大值🡺B[i]; --------------------------------------2.1

}

/\*求A中每一列的最小值🡺数组C\*/ - - - - - - - - -- - - - - - - - - - - - 3

for (j=0; j<=9; j++) {

求数组A第j列的最小值🡺C[j]; --------------------------------------3.1

}

/\*求数组B中的最小值🡺minOfB\*/ - - - - - - - - -- - - - - - - - - - - - 4

minOfB = B[0];

for (i=1; i<=9; i++) {

if (minOfB < B[i]) {

minOfB = B[i];

}

}

/\*求数组C中的最大值🡺maxOfC\*/ - - - - - - - - -- - - - - - - - - - - - 5

maxOfC = C[0];

for ( i=1; i<=9; i++) {

if (maxOfC>C[i]) {

maxOfC = C[i];

}

}

(minOfB- maxOfC)^2 🡺result;

}  
(3) 下一步可以继续对2.1和3.1进行细化。具体略。

**4.16 设计算法完成下列任务：输入一段英文后，无论输入的文字都是大写，还是小写，或大小写任意混合，都能将其整理成除每个句子开头字母是大写外，其他都是小写的文字。**

假设在输入的文字中，两个单词间只允许是空格、,、.、?、!，则在输出的文字中，大写的情况有以下几种：

1. 整段文字的第一个字母是大写；
2. “.”后的第一个字母是大写；
3. “?”后的第一个字母是大写；
4. “!”后的第一个字母是大写；

设变量a存储输入的字符串，变量b存储结果字符串，变量c存储当前处理的字符，变量e存储c之前的字符（当c为第一个字符时除外）。算法如下：

输入字符串a, n = strlen(a)

如果a的第一个字符是字母，则转成大写。

b[0]=a[0]

for ( i=1; i<n; i++)

c = a[i], e=a[i-1]

c是字母

T F

e属于″.?!″

T F

c改大写 c改小写

b[i]=c

给b加字符串结束标志

输出字符串b

# 第5章 编码

**5.1 有人说程序编好后能上机运行就可以了，为什么还要讲究风格和可读性呢？你觉得对吗？为什么？**

对于学生的练习题和作业题，程序编好后能上机运行，一般认为任务就完成了。但对于实际开发的项目来说，任务并没有结束。编码人员编写完程序后，要对自己的程序进行单元测试，测试人员要进行集成测试和系统测试，提交用户使用后，还需要对软件进行持续的维护修改工作。在软件开发和维护过程中，都需要阅读程序。道理很简单，如果需要对某段程序进行修改，首先要读懂，阅读程序是软件开发和维护过程中的一个重要组成部分，而且读程序的时间比写程序的时间还要多。所以，程序的风格和可读性很重要，具有良好的程序风格和可读性的程序，有助于对程序的正确理解，可以大大减少阅读理解程序的时间，从而提高维护工作的效率。

**5.2 一般情况下，程序的效率和清晰性相比哪一个更重要？**

在计算机发展的早期，由于CPU效率低，内存小，程序在运行时容易出现运行时间太长或内存溢出问题。因此，人们在编写程序时，很注重程序的时间效率和空间效率，而不太注重程序的清晰性。

随着计算机硬件性能的飞速发展以及软件规模和复杂性的急剧增加，这种情况已经发生了根本的转变，程序的清晰性越来越受到重视，程序的清晰性不好会给测试、维护修改带来困难，这对于规模庞大和复杂的软件尤其明显。

软件运行的效率主要取决于软件的体系结构及算法，编码阶段虽然也有机会提高效率，但效果并不明显。对于效率没有特殊严格要求的系统来说，要将程序的清晰性放在第一位，在不影响清晰性的情况下，去改进效率。

**5.6 在一行内只写一条语句，并且采取适当的移行格式，使程序的逻辑和功能变得更加明确。许多程序设计语言允许在一行内写多个语句。但这种方式会使程序可读性变差。下面是一段排序程序，请对其编码风格进行改进，以增加其可读性。**

**for (i=1; i<=n-1; i++) for (j=1; j<=n-i; j++) if (a[j]>a[j+1]) { temp=a[j]; a[j]=a[j+1]; a[j+1]=temp;}**

改进后的代码如下：

for (i=1; i<=n-1; i++)

for (j=1; j<=n-i; j++)

if (a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

**5.7 请对下面代码的布局进行改进，使其符合规范其更容易理解。**

**for (i=1; i<=n-1; i++){**

**t=i;**

**for (j=i+1; j<=n; j++)**

**if (a[j]<a[t]) t=j;**

**if (t!=i) {**

**temp=a[t];**

**a[t]=a[i];**

**a[i]=temp**

**}**

**}**

改进后的代码如下：

for (i=1; i<=n-1; i++)

{

t=i;

for (j=i+1; j<=n; j++)

{

if (a[j]<a[t])

t=j;

}

if (t!=i)

{

temp=a[t];

a[t]=a[i];

a[i]=temp

}

}

**5.8使用你熟悉的语言编写出习题4.15的程序，注意编码风格。**

下面为使用C语言编写的习题4.15的程序：

#include "math.h"

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

/\* 求10×10的二维数组a的每一行的最大值存入一维数组b，每一列的最小值存入一维数组c，minOfB存储数组b的最小值，maxOfC存储数组c的最大值，result存储minOfB与maxOfC之差的平方。\*/

int main()

{

int a[10][10], b[10],c[10];

int i,j,minOfB, maxOfC, result;

/\*产生随机数种子 \*/

srand((unsigned int)time(NULL));

/\*建立10 ×10的二维随机整数数组a\*/

for(i=0;i<=9;i++)

for (j=0; j<=9; j++)

a[i][j]=rand()%100+1; //产生1-100之间的随机数

/\*求a中每一行的最大值🡺数组b\*/

for (i=0; i<=9; i++)

{ /\*求数组a第i行的最大值🡺b[i] \*/

b[i]=a[i][0];

for (j=1; j<=9; j++)

if (a[i][j]>b[i]) b[i] = a[i][j];

}

/\*求a中每一列的最小值🡺数组c\*/

for (j=0; j<=9; j++)

{ /\*求数组a第j列的最小值🡺c[j]; \*/

c[j]=a[0][j];

for (i=1; i<=9; i++)

if (a[i][j]<c[j]) c[j] = a[i][j];

}

/\*求数组b中的最小值🡺minOfB\*/

minOfB = b[0];

for (i=1; i<=9; i++)

if (minOfB > b[i]) minOfB = b[i];

/\*求数组c中的最大值🡺maxOfC\*/

maxOfC = c[0];

for (i=1; i<=9; i++)

if (maxOfC<c[i]) maxOfC = c[i];

result = pow(minOfB- maxOfC, 2);

printf("结果是：%d",result);

return(result);

}

**5.9使用你熟悉的语言编写出习题4.16的程序，注意编码风格。**

下面为使用C语言编写的习题4.16的程序：

#include "stdio.h"

#include "string.h"

#include "ctype.h"

/\*将一段英文输入变量a（假设两个单词间只允许是空格、,、.、?、!），将其整理成每个句子开头字母是大写、其他都是小写的字符串存储在变量b中。变量c存储当前处理的字符，变量e存储c之前字符。\*/

void main()

{ char a[1000], b[1000], c,e;

int i,n;

gets(a);

n=strlen(a);

//将整段的第一个字母转成大写。

if(isalpha(a[0]))

{ a[0]=toupper(a[0]);

b[0]=a[0];

}

//处理第一个字符之后的每个字符，将.?!后面的字母改为大写，其他改为小写。

for (i=1; i<n; i++)

{ c=a[i];

e=a[i-1];

if (isalpha(c))

{ if(strchr(".?!",e))

c=toupper(c);

else

c=tolower(c);

}

b[i]=c;

}

b[n]='\0';

puts(b);

}

# 第6章 软件测试方法

**6.5 假设汽车的车牌号可由车主人在规定范围内自选，若其规定为：**

**(1) 车牌上应有7个字符；**

**(2) 为首的字符限定为汉字“京”；**

**(3) 第2个字符可任选一字母（A~Z）；**

**(4) 第3~7个字符可选任意数字。**

**请为相关的处理程序采用等价类划分方法设计等价类表及相应的测试用例。**

**等价类表**

|  |  |  |
| --- | --- | --- |
| **输入数据** | **有效等价类** | **无效等价类** |
| 字符个数 | 7个（1） | 0~6个（2），>7个（3） |
| 首字符 | “京”（4） | 非“京”（5） |
| 第2个字符 | 字母（6） | 非字母（7） |
| 第3~7个字符 | 数字（8） | 非数字（9） |

测试用例如下：

1）京H87774 }（1），（4），（6），（8）

2）京H8777 }（2）

3）京H877745 }（3）

4）冀H87774 }（5）

5）京987774 }（7）

6）京BA7774 }（9）

**6.6 比较α测试与β测试，说明其异同。**

α测试与β测试都是集成测试之后所进行的测试，都属于系统测试。所不同的是，α测试是在开发环境中所做的测试，主要由开发方人员进行。而β测试则是在用户实际工作环境中所做的测试，不受开发人员的干预。

补充题：设计下列伪码程序的语句覆盖和路径覆盖测试用例：

START

INPUT(A,B,C)

A>5

X=10

X=1

B>10

Y=20

Y=2

Y

N

Y

N

C>15

Z=30

Z=3

Y

N

PRINT(X,Y,Z)

INPUT(A,B,C)

IF A>5

THEN X=10

ELSE X=1

END IF

IF B>10

THEN Y=20

ELSE Y=2

END IF

IF C>15

THEN Z=30

ELSE Z=3

END IF

PRINT(X,Y,Z)

STOP

设输入数据为<A，B，C>，输出数据为<X，Y，Z>。

语句覆盖测试用例：【<6,12,20>，<10,20,30>】, 【<4,8,10>，<1,2,3>】

路径覆盖测试用例：

【<6,12,20>，<10,20,30>】

【<6,12,10>，<10,20,3>】

【<6,8,20>，<10,2,30>】

【<6,8,10>，<10,2,3>】

【<4,12,20>，<1,20,30>】

【<4,12,10>，<1,20,3>】

【<4,8,20>，<1,2,30>】

【<4,8,10>，<1,2,3>】

# 第7章 统一建模语言UML概述

**7.2 UML中有哪些关系？解释类图中聚合和关联的相同点和不同点。**

在UML中，常见的关系有泛化、聚合、关联、依赖和实现。

在类图中，关联是一种结构关系，它描述了两个或多个类的实例之间的连接关系。聚合描述了整体和部分之间的结构关系。按照关联关系的定义，聚合属于一种特殊的关联关系，只不过这种关联关系具有明确的整体-部分含义而已。从耦合度的角度看，聚合关系要强于关联关系。

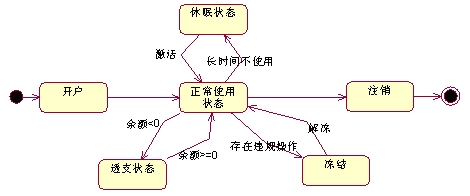
**7.4 讨论顺序图与协作图的关系，何时用顺序图建模优于协作图建模，以及相反的情况。**

顺序图和协作图都能描述对象间的交互关系，但两者的侧重点不同，顺序图着重表现交互的时间顺序，协作图则着重表现交互对象的静态链接关系。但即使他们各有侧重，但从语义上来说是等价的，可从一种图自动转换为另一种图。

一般情况下，当需要强调消息传递的时间顺序时，采用顺序图；当表示涉及很多对象的模型时，协作图比顺序图更形象。顺序图在分析中更常用，而协作图则在设计中更常用。

**7.5 考虑银行系统中的帐户会有哪几种状态，画出帐户对象的状态图。**

账户的状态包括开户、正常使用、透支、休眠、冻结和注销。账户对象的状态图如下图所示。



# 第8章 面向对象分析

**8.1比较面向对象的分析方法和面向数据流的分析方法，阐述它们各自的特点。**

面向对象的分析方法使用用例模型来表示用户的功能需求，用例模型相当于功能模型，在对用例模型进行细化的过程中，也伴随着对功能的分解。但面向对象的分析方法并不是以功能分解为核心，在获取并分析用户的功能需求之后，重点以类和对象为核心，建立对象模型，交互模型也是围绕对象模型进行的。面向数据流的分析方法是从建立顶层数据流图（环境图）开始，之后逐层对加工进行分解。面向数据流的分析方法是以功能分解为核心的。

**8.2 面向对象分析需要建立的三个模型是什么？**

面向对象分析需要建立的3个模型是：由用例和场景表示的功能模型（用例模型）；用类和对象表示的静态模型（对象模型）；由状态图和顺序图表示的动态模型（交互模型）。

**8.3 用例建模的主要步骤是什么？**

用例建模的主要步骤如下：

1. 确定业务参与者──可以是与系统有交互的外部硬件、软件、组织、人等。
2. 确定业务需求用例──参与者需要系统提供的完整功能。
3. 创建用例图──标识参与者与用例之间、用例与用例之间的关系。

**8.5** **用例与用例之间的关系主要有哪两种？其区别是什么？**

用例之间的关系主要有《包含》（也称《使用》）和《扩展》。

《包含》表示一个用例所执行的功能中总是包括被包含用例的功能；《扩展》是指一个用例的执行可能需要由其他用例的功能来扩展，《扩展》联系可用于对期望或可选的行为建模，但其主要用途是使基本用例的功能不依赖于扩展用例。

**8.6** **对于复杂的系统，其对象模型应该由哪五个层次组成？**

对于复杂的系统，对象模型应该由下述5个层次组成：主题层(也称为范畴层)、类-对象层、结构层、属性层和服务层，如图所示。



对象模型的五个层次

**8.7 解释关联类的作用，在什么时候需要使用关联类？**

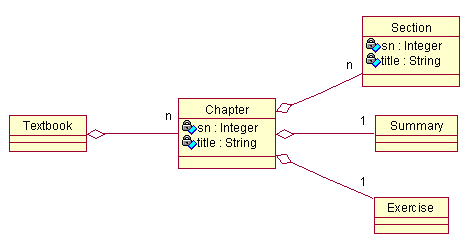
关联类的作用是描述关联的属性。

在关联关系比较简单的情况下，关联关系的语义用关联关系的名字来概括。但在某些情况下，需要对关联关系的语义做详细的定义、存储和访问，为此可以建立关联类（association class）。

**8.8 按照以下描述，画出UML类图：**

**一本教材由许多章组成，每一章包含许多节、小结和习题组成，章和节都具有标题和序号属性。**

UML类图如下：



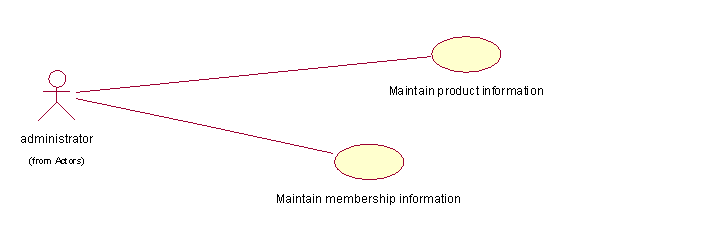
**8.9 考虑一个计算机超市，出售硬件、外设和软件。分析谁是这个系统的参与者？这个系统有哪些主要用例？画出用例图。**

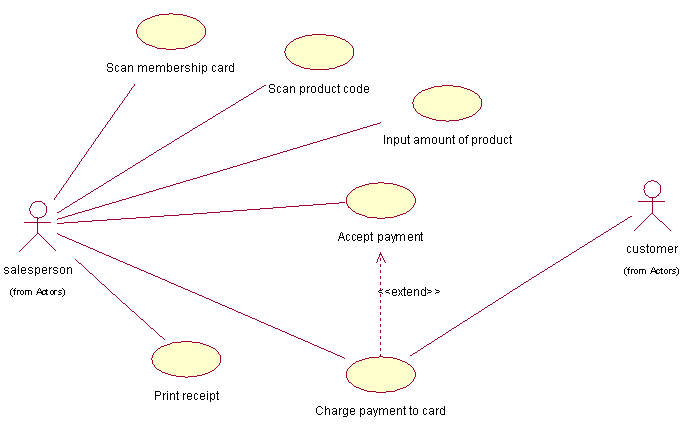
系统的参与者：系统管理员（administrator），售货员（salesperson），客户（customer）。

主要用例

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **参与者** | **用例** | **说明** |
| 1 | administrator | Maintain product information（维护商品信息） | 增加、修改、删除商品信息（商品编号、名称、型号、价格） |
| 2 | Maintain membership information（维护会员信息） | 增加、修改、删除会员信息（会员编号、姓名、身份证号、办卡日期） |
| 3 | salesperson | Scan membership card（扫描会员卡） | 在商品出售之前，系统通过扫描仪刷卡（会员卡）来确认顾客的会员身份 |
| 4 | Scan product code（扫描商品的条形码） | 通过扫描商品的条形码，自动识别商品的名称和价格 |
| 5 | Input amount of product（输入商品数量） | 对于已经扫描的条形码，手工输入购买的此件商品的数量 |
| 6 | salesperson，customer | Accept payment（接收现金付款）  Charge payment to card（使用银行卡付款） |  |
| 7 | salesperson | Print receipt（打印收据） |  |

用例图如下：





# 第10章 面向对象设计

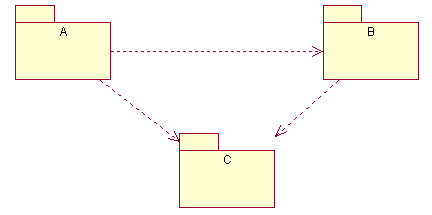
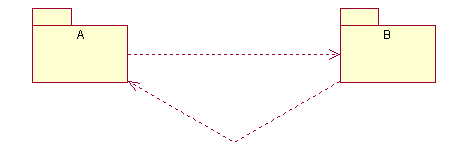
10.1 软件模块之间的依赖性可以从哪些角度和抽象层次进行分析？

在面向对象软件中，常见的软件模块有类、接口、包、构件。因此，分析软件模块之间的依赖性就可以从类、接口、包和构件的角度和层次进行分析。

**10.2 消除包之间循环依赖性的方法是什么？**

本质上，两个包之间的依赖性来自于两个包中类之间的依赖性。类之间的循环依赖性是个特别棘手的问题，好在大多数情况下可以通过重新设计避免循环依赖性。具体方法是增加新包。举例说明如下：

在下图中，包A依赖于包B，反过来包B又依赖包A，构成了循环依赖。具体解决办法是：将包B依赖的包A中的元素从包A中分离出来，组成包C，使得包B不再依赖包A，而是依赖包C。



**10.3 请给出构件的一般性定义，然后给出传统的软件工程环境及面向对象的软件工程环境中构件的定义，最后选择你熟悉的编程语言说明怎样定义一个构件。**

软件构件是一种组装单元，它具有规范的接口规格说明和显式的语境依赖。软件构件可以被独立部署，并由第三方任意组装。OMG UML规范中将构件定义为“系统中某一定型化的、可配置的和可替换的部件，该部件封装了实现并暴露一系列接口”。

在传统的软件工程环境中，一个构件就是程序的一个功能要素，程序由处理逻辑和实现处理逻辑所需的内部数据结构以及能够保证构件被调用和实现数据传递的接口构成。传统的构件也称为模块，是软件体系结构的一部分。

在面向对象的软件工程环境中，面向对象技术已达到了类级复用，而构件级复用则是比类级复用更高一级的复用，它是对一组类的组合进行封装（当然，在某些情况下，一个构件可能只包含一个单独的类），并代表完成一个或多个功能的特定服务，也为用户提供了多个接口。一个构件可以是一个编译的类，可以是一组编译的类，也可以是其他独立的部署单元，如一个文本文件、一个图片、一个数据文件、一个脚本等。

选择你熟悉的编程语言说明怎样定义一个构件。（略）

**10.4 典型的面向对象设计模型在逻辑上由哪几部分组成？对每一部分进行设计时所包含的主要内容是什么？**

典型的面向对象设计模型在逻辑上将系统划分为4个部分，分别是问题域部分、人机交互部分、任务管理部分及数据管理部分。

1. 问题域部分的设计

在面向对象设计过程中，可能要对面向对象分析所得出的问题域模型进行补充或调整。例如，调整需求、复用已有的类、把问题域类组合在一起、增添泛化类以建立类间的协议、调整继承的支持级别、改进性能等。

1. 人机交互部分的设计

人机交互界面的设计质量直接影响到用户对软件的使用。在设计阶段，必须根据需求把交互细节加入到用户界面设计中，包括人机交互所必需的实际显示和输入。

1. 任务管理部分的设计

任务管理主要包括任务的选择和调整。常见的任务有事件驱动型任务、时钟驱动型任务、优先任务、关键任务和协调任务等。

设计任务管理子系统时，需要确定各类任务，并将任务分配给适当的硬件或软件去执行。

1. 数据管理的设计

在采用面向对象方法进行软件开发时，数据的存储还是普遍使用关系数据库。在面向对象设计中，可以将UML类图看作是数据库的概念模型，一个类可以映射为一个表或多个表，此外，还要考虑类之间的关联关系、继承关系的映射。

**10.5 用面向对象方法设计网上购书系统的软件结构，网上购书系统的业务如下：**

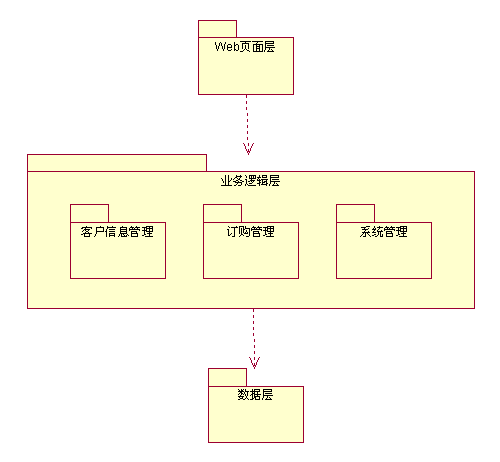
**某书店为方便客户通过Internet购买相关图书，开发一个“网上购书系统”，客户可以通过Web页面注册并登录“网上购书系统”，通过Web页面查看、选择图书，系统根据用户选择的图书单价、数量，系统自动生成订单，并计算总价格。**

**客户在提交订单之前，必须填写关于寄送地址和发票及付款方式等细节，一旦订单被提交，系统显示确认信息，并附上订单的详细信息。客户可以在线查询订单的状态。**

**系统管理人员查看客户的订单，验证客户的信用和付款方式，向仓库请求所购图书，打印发票并发货。**

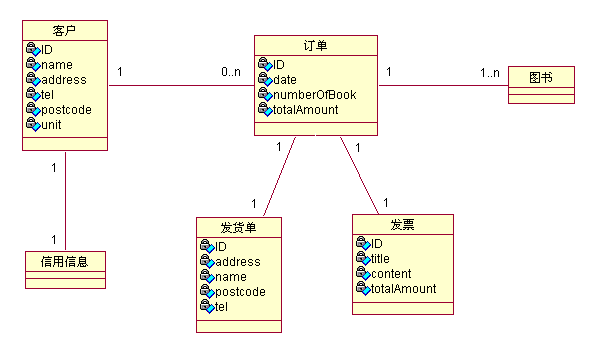
本系统从逻辑结构上分为三层，Web页面层，业务逻辑层和数据库层。Web页面层负责接收用户的请求，业务逻辑层在获知用户请求后，执行相应的业务逻辑，向数据库层发出数据请求，获得所需要的数据后，通知Web页面层将数据呈现给用户。

从业务的角度，又可以将系统划分为订购管理、客户信息管理和系统管理三个子系统（包）。系统的软件结构如图所示。

****

**10.6 对10.5中的网上购书系统设计其问题域的类图。**

问题域类包括客户、信用信息、订单、图书、发票及发货单。类图如下所示：

****