第一章

1、软件产品的特性是什么？

产品特性：

⑴是一种逻辑产品，与物质产品有很大的区别。

⑵软件产品的生产主要是研制，生产成本主要在开发和研制，开发研制完成后，通过复制就产生了大量软件产品。

⑶软件产品不会用坏，不存在磨损，消耗。

⑷生产主要是脑力劳动，还未完全摆脱手工开发方式，大部分产品是"定做"的

⑸开发软件的费用不断增加，致使生产成本相当昂贵。

2、软件生产有几个阶段？各有何特征？

⑴程序设计时代：这个阶段生产方式是个体劳动，使用的生产工具是机器语言，汇编语言。

⑵程序系统时代：这个阶段生产方式是小集团合作生产，使用的生产工具是高级语言，开发方法仍依靠个人技巧，但开始提出结构化方法。

⑶软件工程时代：这个阶段生产方式是工程化的生产，使用数据库、开发工具、开发环境、网络、分布式、面向对象技术来开发软件。

3、什么是软件危机？产生原因是什么？

软件开发技术的进步未能满足发展的要求。在软件开发中遇到的问题找不到解决的办法，问题积累起来，形态尖锐的矛盾，导致了软件危机。

产生原因：

⑴ 软件规模越来越大，结构越来越复杂。⑵ 软件开发管理困难而复杂。⑶ 软件包开发费用不断增加。

⑷ 软件开发技术落后。⑸ 生产方式落后，仍采用手工方式。⑹ 开发工具落后，生产率提高缓慢。

4、什么是软件工程？它目标和内容是什么？

软件工程就是用科学的知识程和技术原理来定义，开发，维护软件的一门学科。

软件工程目标：付出较低开发成本；达到要求的功能；取得较好的性能；开发的软件易于移植；只需较低的维护费用；能按时完成开发任务，及时交付使用；开发的软件可靠性高。

软件工程内容：研究内容包括开发技术和开发管理两个方面。

开发技术主要研究：软件开发方法，开发过程，开发工具和环境。

开发管理主要研究：软件管理学，软件经济学，软件心理学。

5、软件工程面临的问题是什么？

软件工程需要解决的问题：软件的费用，可靠性，可维护性，软件生产率和软件的重用。

6、什么是软件生存周期？它有哪几个活动？

软件生存周期：一个软件从提出开发要求开始直到该软件报废为止的整个时期。

包括：可行性分析和项目开发计划，需求分析，概要设计，详细设计，编码，测试，维护。

7、什么是软件生存周期模型？有哪些主要模型？

生存周期模型：描述软件开发过程中各种活动如何执行的模型。对软件开发提供强有力的支持，为开发过程中的活动提供统一的政策保证，为参与开发的人员提供帮助和指导，是软件生存周期模型化技术的基础，也是建立软件开发环境的核心。

主要有：瀑布模型，增量模型，螺旋模型，喷泉模型，基于知识的模型，变换模型。

8、什么是软件开发方法？有哪些主要方法？
使用早已定义好的技术集及符号表示习惯来组织软件生产的过程。通过使用成功的软件开发方法，在规定的投资和时间内，开发出符合用户需求的高质量的软件。软件开发方法是克服软件危机的重要方面之一，对软件工程及软件包产业的发展起了不可估量的作用。
主要有：结构化方法，JACKSON方法，维也纳开发方法（VDM），面向对象开发方法。

第二章

4.成本--效益分析可用哪些指标进行度量？
⑴.投资回收率:通常把建立系统若干年后能取得的收益折算成现在的价值和开发系统所需的费用进行比较得出投资回收率。
⑵.投资回收期：就是使累计的经济效益等于最初的投资费用所需的时间。
⑶.纯收入：整个生存周期之内的累计经济效益（折成现在值）与投资之差。

5.项目开发计划有哪些内容?
⑴.项目概述：说明项目的各项主要工作；说明软件的功能、性能；为完成项目应具有的条件；用户及合同承包者承担的工作完成的期限及其他条件限制；应交付的程序名称；所使用的语言及程储形式；应交付的文档。
⑵.实施计划：说明任务的划分，各任务责任人，项目开发进度，项目的预算，各阶段的费用支出，各阶段应完成的任务，用图表说明每项任务的开始和完成时间。
⑶.人员组织及分工：所需人员类型、数量、组成结构。
⑷.交付期限：最后完工日期。

第三章
1.什么是需求分析？需求分析阶段的基本任务是什么？
需求分析：开发人员准确地理解用户的要求，进行细致的调查分析，将用户非形式的需求陈述转化为完整的需求定义，再由需求定义转换到相应的需求规格说明的过程。
基本任务:
⑴问题识别：双方确定对问题的综合需求，这些需求包括功能需求，性能需求，环境需求，用户界面需求。
⑵分析与综合，导出软件的逻辑模型
⑶编写文档：包括编写"需求规格说明书","初步用户使用手册","确认测试计划","修改完善软件开发计划"

2.什么是结构分析方法？该方法使用什么描述工具？
结构化分析：简称SA，面向数据流进行数据分析的方法。采用自顶向下逐层分解的分析策略。顶层抽象地描述整个系统，底层具体地画出系统工程的每个细节。中间层则是从抽象到具体的过渡。使用数据流图，数据字典，作为描述工具，使用结构化语言，判定表，判定树描述加工逻辑。

3.结构化分析方法通过哪些步骤来实现？
⑴了解当前系统的工作流程，获得当前系统的物理模型。⑵抽象出当前系统的逻辑模型。⑶建立目标系统的逻辑模型。⑷作进一步补充和优化。

4.什么是数据流图？其作用是什么？其中的基本符号各表示什么含义？
数据流图（DFD）：以图形的方式描述数据在系统中流动和处理的过程。只反映系统必须完成的逻辑功能，是一种功能模型。

5.画数据流图应注意什么事项？

命名： 不能使用缺乏具体含义的名字，加工名应能反映出处理的功能。

画数据流而不是控制流。 数据流名称只能是名词或名词短语，整个图中不反映加工的执行顺序。

一般不画物质流。

每个加工至少有一个输入数据流和一个输出数据流，反映出此加工数据的来源与加工的结果 。

编号： 某个加工分解成加一张数据流图时，上层图为父图，下层图为子图。子图应编号子图上的所有加工也应编号，子图的编号应与父图的编号相对应。

父图与子图的平衡：子图的输入输出 数据流同父图相应加工的输入输出数据流必须一致

局部数据存储：当某慨数据流图中的数据存储不是父图中相应加工的外部接口，而只是本图中某些加工之间的数据接口，则称这些数据存储为局部数据存储。

注意数据流图的易理解性。

9.简述SA方法的优缺点。

SA方法是软件需求分析中公认的，有成效的，技术成熟，使用广泛的一种结构化分析方法。较适用于开发数据处理类型软件的需求分析。利用图形等半形式化工具表达需求，简明，易读，也易于使用。但也存在一些弱点，表现如下：

⑴SA方法仅是一个静态模型，没有反映处理的顺序，即控制流程。

⑵SA方法使用DFD在分析与描述"数据要求"方面是有局限的，只有与数据库技术中的实体联系图（ER图）结合起来，才能较完整地描述用户对系统的需求。

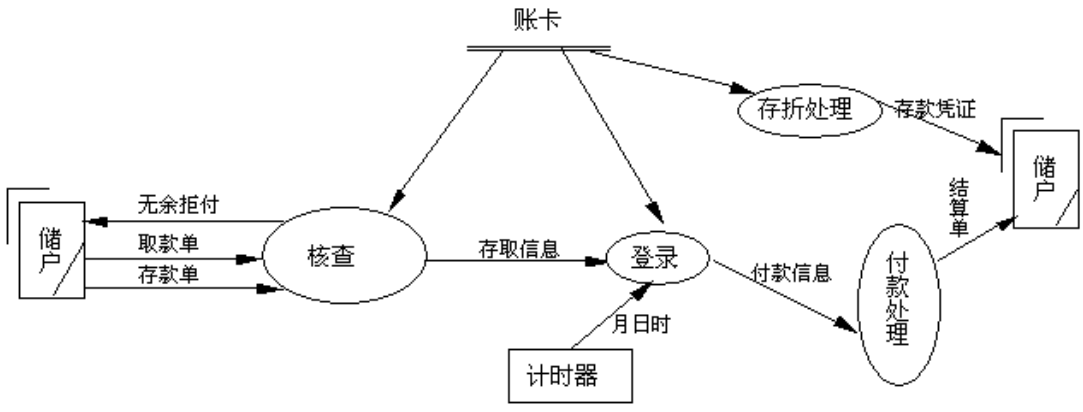
⑶DFD不适合描述人机界面系统的要求，一些人机交互较频繁的软件系统。

⑷SA方法要与形式化方法结合起来，才能更精确地描述软件需求。

⑸要借助需求分析工具，提高需求分析的质量及效率。

10.某银行的计算机储蓄系统功能是：将储户的存户填写的存款单或存款单输入系统，如果是存款，系统记录存款人姓名、住址、存款类型、存款日期、利率等信息，并打印出存款单给储户；如果是取款，系统计算清单给储户。请用DFD描绘该功能的需求。

答：



储蓄系统的 DFD 图

11.某图书管理系统有以下功能：

⑴借书：输入读者借书证。系统首先检查借书证是否有效，若有效，对于第一次借书的读者，在借书证上建

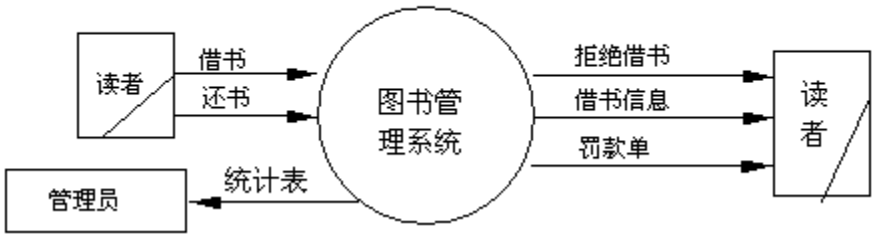
立档案。否则，查阅借书文件，检查该读者所借图书是否超过10本，若已达10，拒借，末达10本，办理借书（检查库存，修改库存目录并将读者借书情况录入借书文件。）

⑵还书：从借书文件中读出与读者有关的记录，查阅所借日期，如超期（3个月）作罚款处理。否则，修改库存目录与借书文件。

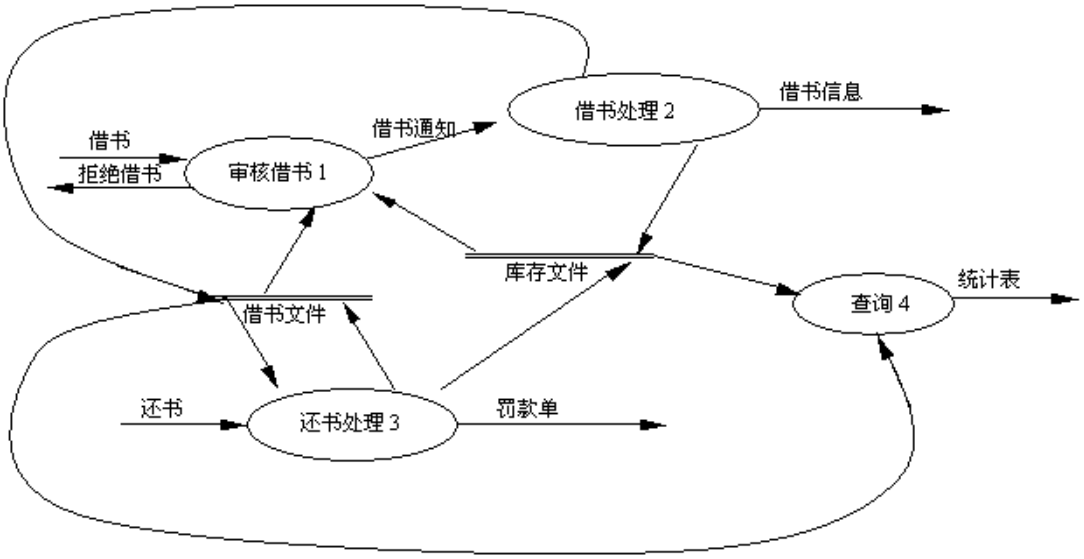
⑶查询：通过借书文件，库存目录文件查询读者情况、图书借阅及库存情况，打印统计表。

解：

顶层图



0 层图



1 层图

图 1

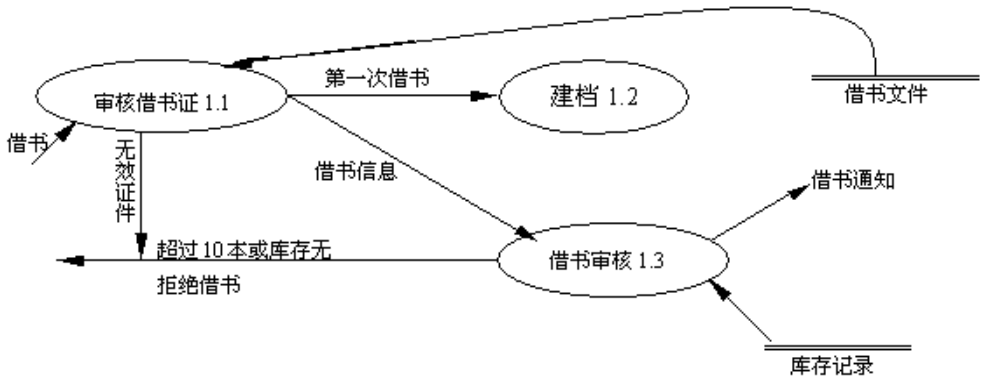


图 2

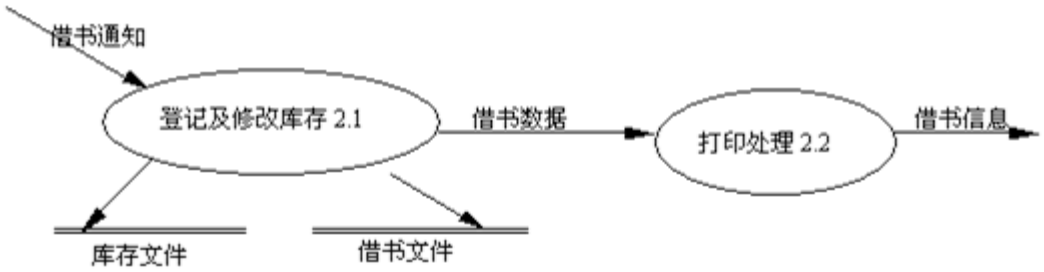


图 3

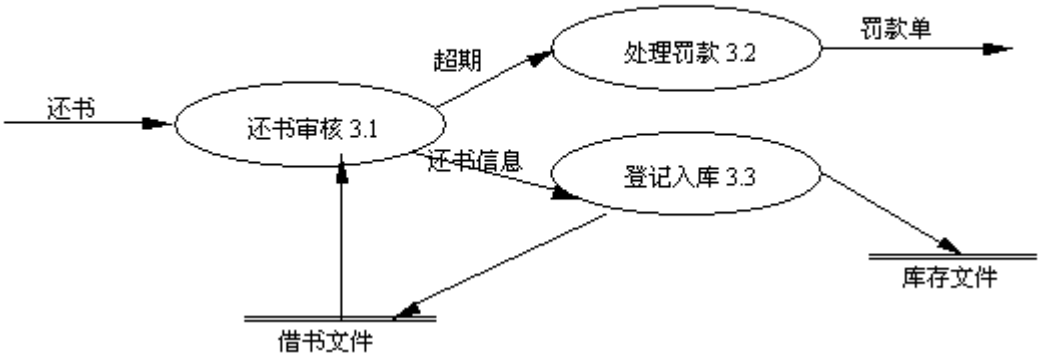
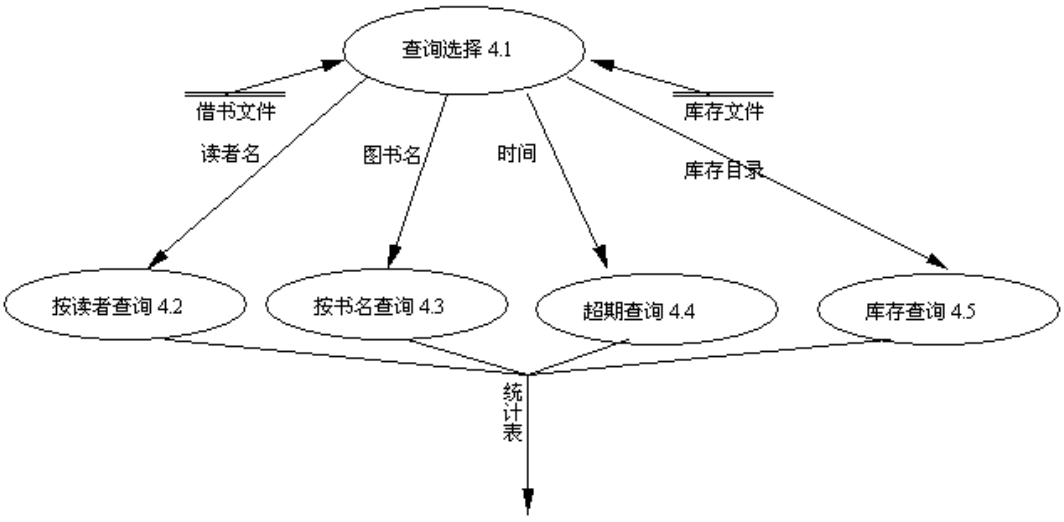


图 4



1. 什么是软件概要设计？该阶段的基本任务是什么？

把一个软件需求转换为软件表示时，首先设计出软件总的体系结构。称为概要设计或结构设计。

基本任务：

⑴设计软件系统结构，具体为：

采用某种设计方法，将一个复杂的系统按功能划分为模块。

确定每个模块的功能。

确定模块之间的调用关系。

确定模块之间的接口（模块之间传递的信息）

评价模块的结构质量

⑵进行数据结构及数据库的设计
⑶编写概要设计的文档
主要内容有：概要设计说明书，数据库说明书（DBMS的简介，概念模型，逻辑设计，结果）用户手册，修订测试计划（测试的策略、方法、步骤）
⑷评审：是否完整地实现了规定的功能、性能要求。设计方案是否可行。关键的处理及内部接口定义的正确性，有效性，各部分的一致性。

2．软件设计的基本原理包括哪些内容？
⑴模块化：模块是软件的组成部分，是具有独立功能且可命名的一段程序，所有模块组成
整体，可以满足问题的要求。模块化即按照一定的原则，将软件划分成若干个模块，每个模块完成一个特定的功能，然后把这些模块按照某种方法组装成一个软件系统。（可降低复杂度、减少工作量）
模块具有以下几种特性：
接口：模块的输入输出。
功能：指模块实现什么功能。
逻辑：描述模块内部如何实现要求及所需的数据。
状态：该模块的运行环境，模块间调用与被调用关系。
⑵抽象：认识复杂现象过程所使用权的工具，只考虑事物本质的共同特性，忽略细节和其它因素。通过抽象确定组成软件的过程实体。
⑶信息隐蔽：将模块实现自身功能的细节与数据"隐蔽"起来。模块间仅交换为完成系统功能所必须的信息。
⑷模块独立性：每个模块只完成系统要求的独立的子功能。

3．衡量模块独立性的两个标准是什么？各表示什么含义？
内聚和耦合
内聚：又称为块内联系，指模块内部各成分之间相互关联的程度，以高内聚为设计目标。
耦合：也称块间联系，模块之间相互联系程度的度量，联系越紧密，耦合性越强，独立性越差，以低耦合为设计目标。

4．模块的耦合性有哪几种？各表示什么含义？
⑴内容耦合：一个模块直接操作或修改另一模块的数据，或者不通过正常入口直接转入 另一模块
⑵公共耦合：两个或多个模块通过共同引用一个全局数据环境相互作用
⑶控制耦合：模块之间通过传递控制信息相互作用
⑷标记耦合：两个模块之间通过传递公共指针或地址相互作用的耦合
⑸数据耦合：模块之间通过传递数据交换信息
⑹无耦合：模块间无任何关系，独立工作

5．模块的内聚性有哪几种？各表示什么含义？
⑴偶然内聚：一个模块各个成分之间毫无关系
⑵逻辑内聚：将几个逻辑上相关的功能放在同一个模块中
⑶时间内聚：一个模块完成的功能在同一时间执行
⑷过程内聚：一个模块内部的处理成分是相关的，而且必须以特定的次序执行
⑸通信内聚：一个模块的所有成分都集中在同一个数据结构上
⑹顺序内聚：一个模块的各个成分同一个功能密切相关，而且一个成分的输出，作为另外一个成分的输入
⑺功能内聚：模块内的所有成分属于一个整体，完成单一的功能。（内聚最高）

6．什么是软件结构？简述软件结构设计的优化准则。
⑴改进软件结构，提高模块独立性 首先设计出软件初始结构，评价该结构，通过模块分解或合并，力求降低耦合提高内聚。
⑵模块的规模应该适中
⑶模块结构的深度、宽度、扇出和扇入应适中
深度：软件结构中控制的层数
宽度：软件结构中同一层次上最大模块总数
扇入：某一模块有多少直接调用它的上级模块数目（越大越好）
扇出：一个模块直接控制（调用）下级模块的数目。（越少越好，3，4个为宜，不超过9个，
"顶层扇出较高，中间扇出较少，底层模块高扇入多"
⑷一个模块的作用域（范围），应处在这个模块的控制域（范围）之内
模块的作用域：受该模块内一个判定影响的所有模块的集合
模块的控制域：这个模块本身以及所有直接或间接从属于它的模块的集合

第5章
1、详细设计的基本任务是什么，有哪几种描述方法?
答：详细设计的基本任务包括：
1、为每个模块进行详细的算法设计 2、为模块内的数据结构进行设计 3、对数据库进行物理设计
4、其他设计 5、编写详细设计说明书 6、评审
详细设计的描述方法有图形、表格和语言，其中图形常用结构化程序流程图、盒图和PAD(问题分析图)为描述工具，语言常用过程设计语言(PDL)来作为工具。

2、结构化程序设计的基本要点是什么?
答:主要有三个：
1、采用自顶向下、逐步求精的程序设计方法
2、使用三种基本控制结构构造程序。任何程序都可以由顺序、选择、重复(循环)三种基本控制结构构造，这三种基本结构的共同点是单入口、单出口。
3、主程序员组的组织形式。

3、简述Jackson方法的设计步骤。
答：Jackson方法的设计步骤如下：
(1)分析并确定输入数据和输出数据的逻辑结构，并用Jackson结构图表示这些数据结构。
(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)按一定的规则由输入、输出的数据结构导出程序结构。
(4)列出基本操作与条件，并把它们分配到程序结构图的适当位置。
(5)用伪码写出程序。

第6章
1、程序语言有哪些共同特征?
答：程序语言的共同特征(似为特性更符合题意)是指程序设计语言的语言特性，包括心理特性、工程特性和技术特性三个方面。语言的心理特性对人机通信的质量有主要的影响；语言的工程特性对软件开发成功一否有重要的影响，此外，语言的技术特性也会影响软件设计的质量。

2、在项目开发时选择程序设计语言通常考虑哪些因素?

答：通常要考虑的因素有：

项目的应用领域、软件开发的方法、软件执行的环境、算法和数据结构的复杂性以及软件开发人员的知识等。

5、什么是程序设计风格?为了具有良好的设计风格，应注意哪些方面的问题?

答：程序设计风格是指一个人编制程序时所表现出来的特点、习惯、逻辑思路等。

要形成良好的程序设计风格，应从源程序文档化、数据说明、语句构造、输入输出和追求效率几个方面加以注意。

第7章

1、软件测试的目的是什么?软件测试中，应注意哪些原则?

答：软件测试的目的是为了发现软件的错误。

软件测试中应注意的原则有：

(1)测试用例应由输入数据和预期的输出数据两部分组成。这样便 于对照检查，做到有的放矢。

(2)测试且例不仅选用合理输入数据，还要选择不合理的输入数据。这样能更多地发现错误，提高程序的可靠性。对于不合理的输入数据，程序应拒绝接受，并给出相应的提示。

(3)除了检查程序是否做了它应该做的事，还应该检查程序是否做了它不应该做的事。

(4)应制定测试计划并严格执行，排除随意性。

(5)长期保留测试用例。

(6)对发现错误较多的程序段，应进行更深入的测试。

(7)程序员应避免测试自己的程序。测试是一种"挑剔性"的行为，心理状态是测试自己程序的障碍。

2、什么是白盒测试法?有哪些覆盖标准?试对他们的检错能力进行比较?

答：白盒法测试法把测试对象看作一个打开的盒子，测试人员须了解程序内部结构和处理过程，以检查处理过程的细节为基础，对程序中尽可能多的逻辑路径进行测试，检验内部控制结构和数据结构是否有错，实际的运行状态与预期的状态是否一致。

白盒法有下列几种覆盖标准：

语句覆盖　判定覆盖　条件覆盖　判定/条件覆盖　条件组合覆盖　路径覆盖

从左到右的覆盖标准其检错能力也从弱到强，其中条件组合发现错误的能力较强，凡满足其标准的测试用例，也必然满足前四种覆盖标准。在实际的逻辑测试中，一般以条件组合覆盖为主设计测试用例，然后再补充部分用例来达到路径覆盖的测试标准。

3、什么是黑盒测试法?采用黑盒技术测试用例有哪几种方法?这些方法各有什么特点?

答：黑盒测试法把被测试对象看成是一相黑盒子，测试人员完全不考虑程序的内部结构和处理过程，只在软件接口处进行测试，依据需求规格说明书，检查程序是否满足功能要求。

采用黑盒技术测试用例的方法有：等价类的划分、边界值分析、错误推测和因果图。

等价类的划分，是将输入数据按有效的或无效的(也称合理的或不合理的)划分成若干个等价类，测试每个等价类的代表值就等于对该类其他值的测试。这样就把漫无边迹的随机测试改为有针对性的等价类测试，用少量有代表性的例子代替大量测试目的相同的例子，能有效地提高测试效率。但这个方法的缺点是没有注意选择某些高效的、能够发现更多错误的测试用例。

边界值分析法一般与等价类划分结合起来。但它不是从一个等价类中任选一个例子做代表，而是将测试边界情况作为重点目标，选取正好等于、刚刚大于和刚刚小于边界值的测试数据。(边界情况是指输入等价类和输入等价类边界上的情况。)这种方法可以查出更多的错误，因为在程序中往往在处理边界情况时易发生错误。错误推测法是在测试程序时，人们根据经验或直觉推测程序中可能存在的错误，从而有针对性地编写检查这

些错误的测试用例。

因果图能够有效地检测输入条件的各种组合可能会引起的错误。它的基本原理是通过画因果图，把用自然语言描述的功能说明转换为判定表，最后为判定表的每一列设计一个测试用例。

这几种方法都不能提供一组完整的测试用例，在实际测试中应把各种方法结合起来使用。

综合策略：就是联合使用上述几种测试方法，尽可能多地发现程序中的错误。

4、软件测试要经过哪些步骤?这些测试与软件开发各阶段之间有什么关系?

答：软件测试要经过的步骤是：单元测试→集成测试→确认测试→系统测试。

单元测试对源程序中每一个程序单元进行测试，检查各个模块是否正确实现规定的功能，从而发现模块在编码中或算法中的错误。该阶段涉及编码和详细设计文档。

集成测试是为了检查与设计相关的软件体系结构的有关问题，也就是检查概要设计是否合理有效。

确认测试主要是检查已实现的软件是否满足需求规格说明书中确定了的各种需求。

系统测试是把已确认的软件与其他系统元素(如硬件、其他支持软件、数据、人工等)结合在一起进行测试。以确定软件是否可以支付使用。

5、单元测试有哪些内容?测试中采用什么方法?

答：单元测试主要针对模块的以下五个基本特征进行测试：

(1)模块接口　(2)局部数据结构　(3)重要的执行路径　(4)错误处理　(5)边界条件

测试的方法是为被测试模块编写驱动模块和桩模块来实现被测试单元的可运行。通过驱动模块来模拟被测试模块的上级调用模块，以上级模块调用被测模块的格式驱动被测模块，接收被测模块的测试结构并输出。桩模块则用来代替被测试模块所调用的模块。它的作用是返回被测模块所需的信息。

6、什么是集成测试?非渐增式测试与渐增式测试有什么区别? 渐增式测试如何组装模块?

答：集成测试是指在单元测试的基础上，将所有模块按照设计要求组装成一个完整的系统进行的测试。

非渐增式测试是指首先对每个模块分别进行单元测试，再把所有模块组装成一个完整的系统进行的测试。而渐增式测试就是逐个把未经测试的模块组装到已经过测试的模块上去进行集成测试，每加入一个新模块进行一次集成测试，重复此过程直到程序组装完毕。渐增式测试有两种不同的组装方法：自顶向下和自底向上结合。

两者区别是：

(1)非渐增式方法把单元测试和集成测试分成两个不同的阶段，前一阶段完成模块的单元测试，后一阶段完成集成测试。而渐增式测试往往把单元测试和集成测试合在一起，同时完成。

(2)非渐增式需要更多的工作量，因为每个模块都需要驱动模块和桩模块，而渐增式利用已测试过的模块作为驱动模块或桩模块，因此工作量少。

(3)渐增式可以较早地发现接口之间的错误，非渐增式最后组装时才发现。

(4)渐增式有利于排错，发生错误往往和最近新加入的模块有关，而非渐增式发现接口错误推迟到最后，很难判断是哪一部分接口出错。

(5)渐增式比较彻底，已测试的模块和新的模块再测试。

(6)渐增式点用时间较多，但非渐增式所需更多的驱动模块和桩模块也占用一些时间。

(7)非渐增式开始可并行测试所有模块，能充分利用人力，对测试大型软件很有意义。

7、什么是确认测试?该阶段有哪些工作?

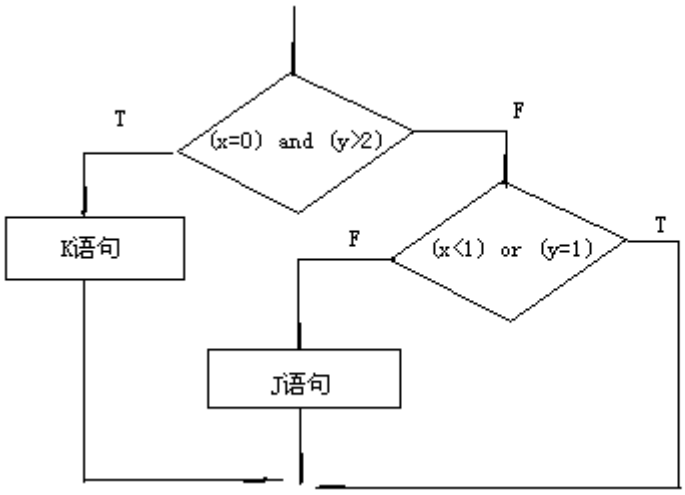
答：确认测试又称有效性测试。它的任务是检查软件的功能与性能是否与需求规格说明书中确定的指标相符合。因而需求说明是确认测试的基础。确认测试阶段有两项工作：进行确认测试与软件配置审查。

8、调试的目的是什么?调试有哪些技术手段?

答：调试则是在进行了成功的测试之后才开始的工作。调试的目的是确定错误的原因和位置，并改正错误，因此调试也称为纠错(Debug)。调试的技术手段有简单的调试方法、归纳法、演绎法和回溯法等。

9、将正确答案的编号填入题目空白处：

在白盒测试用例中，有语句覆盖、条件覆盖、判定覆盖、路径覆盖等，其中(A)是最强的覆盖准则。为了对下图所示的程序进行覆盖测试，必须适当地选取测试数据。若X，Y是两个变量，可供选择的测试数据组共有I，II，III，IV四组(如表中给出)，则实现语句覆盖至少应采用的测试数据组是(B)；实现条件覆盖至少应采用的测试数据组是(C)；实现路径覆盖至少应采用的测试数据组是(D)或(E)。



	X	Y
测试数据组I	0	3
测试数据组II	1	2
测试数据组III	-1	2
测试数据组IV	3	1

可供选择的答案：

A：(1)语句覆盖 (2)条件覆盖 (3)判定覆盖 (4)路径覆盖

B～E：(1)I和II组 (2)II和III级 (3)III和IV组 (4)I和IV组 (5)I、II和III组 (6)II，III和IV组

(7)I，III和IV组 (8)I、II和IV组

答：A：(4) B:(1) C:(4) D：(5) E：(8)

第8章

1、软件维护有哪些内容？

答：（1）校正性维护。在软件交付使用后，一些隐含的错误在某些特定的使用环境下会暴露出来。为了识别和纠正错误，修改软件性能上的缺陷，应进行确定和修改错误的过程，这个过程就称为校正性维护。

（2）适应性维护。为了使应用软件适应计算机硬件、软件环境及数据环境的不断发生的变化而修改软件的过程称为适应性维护。

（3）完善性维护。为增加软件功能、增强软件性能、提高软件运行效率而进行的维护活动称为完善性维护。

（4）预防性维护。为了提高软件的可维护性和可靠性而对软件进行的修改称为预防性维护。

2、 软件维护的特点是什么？

答：主要体现在三个方面：

（1） 非结构化维护和结构化维护。软件的开发过程对软件的维护有很大的影响。若不采用软件工程的方法开发软件，则软件只有程序而无文档，维护工作非常困难，这是一种非结构化的维护。若采用软件工程的方法开发软件，则各阶段都有相应的文档，容易进行维护工作，这是一种结构化的维护。

（2） 维护的困难性。软件维护的困难性是由于软件需求分析和开发方法的缺陷。软件生存周期中的开发阶段没有严格而又科学的管理和规划，就会引起软件运行时的维护困难。（3）

软件维护的费用。软件维护的费用在总费用中的比重是在不断增加的，这是软件维护有形的代价。另外还有无形的代价，即要占有更多的资源。软件维护费用增加的占有原因是软件维护的生产率非常低。

3、 软件维护的流程是什么？
答：软件维护的流程如下：知道维护申请报告、审查申请报告并批准、运行维护并做详细记录、复审。

4、 软件维护的副作用有哪些？
答：维护的副作用有以下三种：
（1） 编码副作用。在使用程序设计语言修改源代码时可能引入的错误。
（2） 数据副作用。在修改数据结构时，有可能造成软件设计与数据结构不匹配，因而导致软件错误。数据副作用是修改软件信息结构导致的结果。但它可以通过详细的设计文档加以控制。
（3） 文档副作用。如果对可执行软件的修改没有反映在文档中，就会产生文档副作用。

5、什么是软件可维护性？可维护性度量的特性是什么？
答：软件可维护性的定义：软件能够被理解、校正、适应及增强功能的容易程度。软件的可维护性可用以下七个质量特性来衡量，即可理解性、可测试性、可修改性、可靠性、可移植性、可使用性和效率。

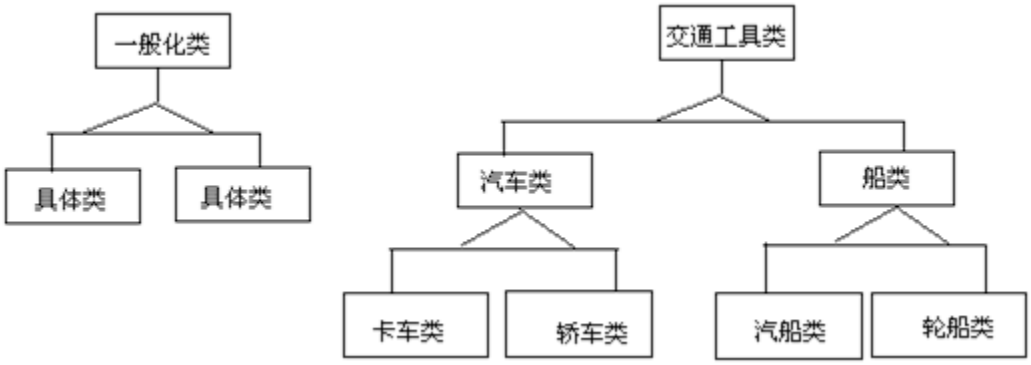
6、 提高可维护性的方法有哪些？
答：（1）建立明确的软件质量目标。 （2）使用先进的软件开发技术和工具。 （3）建立明确的质量保证。（4）选择可维护性的程序设计语言。 （5）改进程序的文档。

第10章
1、说明对象、类、类结构、消息的基本概念。
答：(1)对象：对象是人们要进行研究的任何事物，它不仅能表示具体的事物，还能表示抽象的规则、计划或事件。对象包括有形实体、作用、事件、性能说明等类型。对象具有状态和行为。一个对象用数据值来描述它的状态，对象的操作则用于改变状态，对象及其操作就是对象的行为。对象实现了数据和操作的结合，使数据和操作封装于对象的统一体中。对象内的数据具有自己的操作，从而可灵活地专门描述对象的独特行为，具有较强的独立性和自治性，其内部状态不受或很少受外界的影响，具有很好的模块化特点。为软件重用奠定了坚实的基础。
(2)类：具有相同或相似性质的对象的抽象就是类。因此，对象的抽象就是类，类的具体化就是对象，也可以说类的实例是对象。
(3)类结构：类与类之间的结构关系，包括一般-具体结构关系和整体-部分结构关系。
一般-具体关系结构称为分类结构，也可以说是"或"关系或"is a"关系，类的这种层次结构可用来描述现实世界中的一般化的抽象关系，通常越在上层的类越具有一般性和共性，越在下层的类越具体、越细化。
整体-部分结构称为组装结构，它们之间的关系是一种"与"关系，或者是"has a"关系。类的这种层次关系可用来描述现实世界中的类的组成的抽象关系。上层的类具有整体性，下层的类具有成员性。
在类的层次结构中，通常上层的类称为父类或超类，下层类称为子类。
(4)消息：对象之间进行通信的构造叫做消息。在对旬的操作中，当一个消息发送给某个对象时，消息包含接收对象去执行某种操作的信息。接收消息的对象经过解释，然后给予响应。这种通信机制称为消息传递。发送一条消息的格式是"对象名.方法名(参数)"。

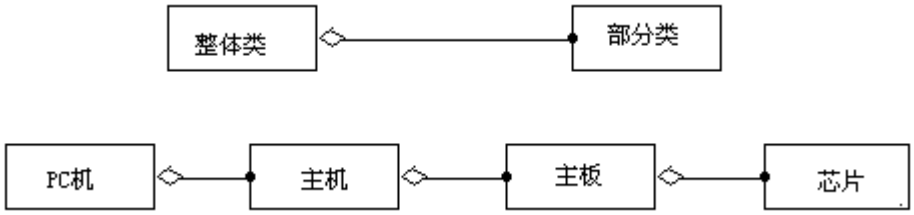
2、说明面向对象的特征和要素。
答：面向对象的特征是：
(1)对象唯一性。每个对象都有自身唯一的标识，在对象生存期中,其标识不变,不同的对象不能有不同的标识。

(2)分类性。是指将具有一致的数据结构(属性)和行为(操作)的对象抽象成类
(3)继承性。是子类自动共享父类数据结构和方法的机制，这是类之间的一种关系。
(4)多态性。是指相同的操作或函数过程可以作用于多种类型的对象上并获得不同的结果。
面向对象的要素是：
(1)抽象。 (2)封装性(信息隐蔽) (3)共享性。

3、说明对象模型的特征，举现实世界的例子，给出它的一般关系、聚集关系的描述。
答：象模型表示了静态的、结构化的系统数据性质，描述了系统的静态结构，它是从客观世界实体的对象关系角度来描述。表现了对象的相互关系。该模型的特征是用对象图来表现对象的结构、属性和操作，它是分析阶段三个模型的核心，也是其他两个模型的框架。
在对象模型中，定义了两种类的层次结构：一般化关系和聚集关系。
一般化关系是在保留对象差异的同时共享对象相似性的一种高度抽象方式，即"一般-具体"的关系。比如下图所示：



聚集关系是一种"整体-部分"关系。在这种关系中，有整体类和部分类之分。如下图所示：



第11章
1、软件质量与软件质量保证的含义是什么?
答：从实际应用来说，软件质量定义包括三个一致性：
(1)与所确定的功能和性能需求的一致性； (2)与所成文的开发标准的一致性；
(3)与所有专业开发的软件所期望的隐含特性的一致性。
软件质量保证是指确定、达到和维护所需要的软件质量而进行的所有有计划、有系统的管理活动。

2、影响软件质量的因素有哪些?
答：影响软件质量的因素分为可以直接度量的因素(如单位时间内千行代码中所产生的错误)和只能间接度量的因素(如可用性和可维护性)。

3、什么是软件质量保证策略?软件质量保证的主要任务是什么?
答：软件质量保证策略是指软件质量保证工作的过程和侧重点。
质量保证的主要任务包括以下几点：

(1)正确定义用户的要求。 (2)技术方法的应用。 (3)提高软件开发的工程能力。 (4)软件的复用。
(5)发挥每个开发者的能力。 (6)组织外部力量协作。 (7)排除无效劳动。 (8)提高计划和管理质量。

5、程序复杂性的度量方法有哪些?
软件复杂性的度量方法有：
1、代码行度量法：以源代码行数作为程序复杂性的度量。
2、McCabe度量法：一种基于程序控制流的复杂性度量方法。

6、什么是软件的可靠性?它们能否定量计算?
软件可靠性是指在给定的时间内，在规定的环境条件下系统完成所指定功能的概率。
衡量软件可靠性的两个常用指标是平均失效等待时间MTTF和平均失效间隔时间MTBF。就是说可以定量计算。

7、为什么要进行软件评审?软件设计质量评审与程序质量评审都有哪些内容?
答：因为软件生存期每个阶段的工作都有可能引入人为错误，如果某一阶段的错误不及时纠正，就会传播到开发的后结阶段，引出更多错误，因此，进行软件评审是必要的，评审可以揭露软件中的缺陷然后加以改正。
设计质量评审的对象是在需求分析阶段产生的软件需求规格说明、数据需求规格说明，在软件概要设计阶段产生的软件概要设计说明书等。主要内容有：
(1)评价软件的规格说明是否合乎用户的要求。 (2)评审可靠性。 (3)评审保密措施实现情况。
(4)评审操作特性实施情况。 (5)评审性能实现情况。
(6)评审软件是否具有可修改性、或扩充性、可互换性和可移植性。
(7)评审软件是否具有可测试性。 (8)评审软件是否具有复用性。
程序质量评审的重点在于软件本身的结构、与运行环境的接口、变更带来的影响而进行的评审活动。

8、说明容错软件的定义与容错的一般方法。
答：容错软件的定义有四种，指规定功能的软件，
(1)在一定程度上对自身错误的作用具有屏蔽能力的软件。
(2)在一定程度上能从错误状态自动恢复到正常状态的软件。
(3)在因错误而发生错误时，仍然能在一定程度上完成预期的功能的软件。
(4)在一定程度上具有容错能力的软件。
实现容错技术的主要手段是冗余。冗余通常分为四类：
(1)结构冗余，又分为静态、动态和混合冗余三种。 (2)信息冗余 (3)时间冗余 (4)冗余附加技术

第12章
1、软件工程管理包括哪些内容?
答：软件工程管理的具体内容包括对开发人员、组织机构、用户、文档资料等方面的管理。

2、软件项目计划中包括哪些内容?
答：软件项目计划内容包括： (1)范围 (2)资源 (3)进度安排 (4)成本估算 (5)培训计划

3、软件开发成本估算方法有哪几种?
答：软件开发成本估算的方法主要有： (1)自顶向下估算方法 (2)自底向上估算方法 (3)差别估算方法以及专家估算法、类推估算法、算式估算法等几类方法。

第十三章

2、请叙述软件开发环境的分类。

答：软件开发环境可按解决的问题、软件开发环境的演变趋向和集成化程度进行分类：

(1)按解决的问题可分为：程序设计环境、系统合成环境、项目管理环境三类。

(2)按软件开发环境的演变趋向可分为：以语言为中心的环境、工具箱环境和基于方法的环境三类。

(3)按集成化程度有第一代、第二代、第三代的开发环境。

3、何谓软件工具?通常包含哪几部分?

答：软件工具的定义是：可用来帮助和支持软件需求分析、软件开发、测试、维护、模拟、移植或管理而编制的计算机程序或软件。软件工具是一个程序系统。

软件工具通常由工具、工具接口和工具用户接口三部分构成。

4、当今软件工具发展有何特点?

答：软件工具的发展有以下特点：

(1)软件工具由单个工具向多个工具集成化方向发展。 (2)重视用户界面设计。

(3)不断地采用新理论和新技术。 (4)软件工具的商品化与软件产业的发展形成良性互动。

5、什么是CASE?CASE工具有哪些分类?

答： CASE是一组工具和方法的集合，可以辅助软件开发生命周期各阶段进行软件开发。CASE把软件开发技术、软件工具和软件开发方法集成到一个统一而一致的架中，并且吸收了CAD(计算机辅助设计)、软件工程、操作系统、数据库、网络和许多其他计算机领域的原理和技术。因而，CASE领域是一个应用、集成和综合的领域。

根据CASE系统对软件过程的支持范围，CASE可分为三类：

(1)支持单个过程任务的工具。 (2)支持某一过程所有活动或某些活动的工作台。

(3)环境支持软件过程大部分乃至所有活动，一般包括几个不同工作台的集合。

6、请叙述集成化CASE的五级模型。

答：集成化CASE的五级模型包括

(1)平台集成：工具运行在相同的硬件/操作系统平台上

(2)数据集成：工具使用共享数据模型来操作

(3)表示集成：工具使用相同的用户界面

(4)控制集成：工具激活后能控制其他操作

(5)过程集成：工具在一个过程模型和"过程机"的指导下使用

7、CASE工作台有哪些分类?

答：CASE工具台是一组工具集，支持像设计、实现或测试等特定的软件开发阶段。CASE工具组装成一个工具台后工具能协同工作，可提供比单一工具更好的支持。

CASE工作台包括：

(1)程序设计工作台。由支持程序设计的一组工具组成。

(2)分析和设计工作台。支持软件过程的分析和设计阶段。

(3)测试工作台。

(4)交叉开发工作台。这些工作台支持在一种机器上开发软件，而在其他的系统上运行所开发的软件。

(5)配置管理(CM)工作台。这些工作台支持配置管理。

(6)文档工作台。这些工具支持高质量文档的制作。

(7)项目管理工作台。支持项目管理活动。