* MIME它是一个互联网标准，扩展了电子邮件标准，使其能够支持，与安全无关。
* 与安全电子邮件相关的是S/MIME安全多用途互联网邮件扩展协议。
* SSL和HTTPS涉及到邮件传输过程的安全, HTTPS是HTTP与SSL的结合体，为传输层以上层次数据加密。
* SSH是建立在应用层和传输层基础上的安全协议，SSH是目前较可靠，专为远程登录会话和其他网络服务提供安全性的协议
* TCP是可靠的传输层协议
* SMTP邮件传输协议。
* PGP（全称：Pretty Good Privacy，优良保密协议），是一套用于信息加密、验证的应用程序，可用于加密电子邮件内容。
* 外部实体一般包括 组织机构、人员、第三方系统
* 如果一个模块内部的处理成分是相关的，而且这些处理必须以特定的次序执行，则称为过程内聚。
* 如果一个模块完成的功能必须在同一时间内执行（如系统初始化），但这些功能只是因为时间因素关联在一起，则称为时间内聚。
* 如果一个模块的各个成分和同一个功能密切相关，而且一个成分的输出作为另一个成分的输入，则称为顺序内聚。
* 几个逻辑上相关的功能被放在同一模块中，则称为逻辑内聚。
* 编译器的工作过程划分为词法分析、语法分析、语义分析，中间代码生成，代码优化和目标代码生成

记号流，词法分析的输出是记号流，也就是语法分析的输入。

字符流，在Java中，根据处理的数据单位不同，分为字节流和字符流。字符流是由字符组成的

源程序，词法分析的任务是把源程序的字符串转换成单词符号序列。

分析树，如果没有语法错误，语法分析后就能正确的构造出其语法树。

括号不匹配是典型的语法错误，会在语法分析阶段检测出来。

* I/O软件隐藏了I/O操作实现的细节。I/O软件向用户提供的是逻辑接口
* **磁盘调度**管理中，先进行移臂调度寻找磁道，再进行旋转调度寻找扇区。
* **易用性**的自特性包括易理解性、易学性、易操作性
* **可靠性质量**属性包括:成熟性、容错性和易恢复性。
* **可维护性**：可理解性、可测试性、可修改性
* MVC采用关注点分离的方针，将可视化界面呈现（View）、UI处理逻辑（Controller）和业务逻辑（Model）三者分离出来，将它们之间的依赖降至最低。解耦之后，有利于代码重用、提高系统的可维护性，也可提高系统的开发效率，但是由于分层之后，调用也需要分层调用，所以不能提高系统的运行效率
* **软件维护**是软件生命周期最长的一段，相对于软件开发任务而言，软件维护工作要更加复杂
* 进行面向对象系统设计时，软件实体（类、模块、函数等）应该是可以扩展但不可修改的，这属于（**开放封闭** ）设计原则
* 运行时结合是动态绑定，编译时结合是静态绑定
* **事物的特性**：

1. 原子性：事务是原子的，要么做，要么都不做。

2. 一致性：事务执行的结果必须保证数据库从一个一致性状态变到另一个一致性状态。

3. 隔离性：事务相互隔离。当多个事务并发执行时，任一事务的更新操作直到其成功提交的整个过程，对其它事物都是不可见的。

4. 持久性：一旦事务成功提交，即使数据库崩溃，其对数据库的更新操作也永久有效。**软件著作权**：

（1）发表权，即决定软件是否公之于众的权利；

（2）署名权，即表明开发者身份，在软件上署名的权利；

（3）修改权，即对软件进行增补、删节，或者改变指令、语句顺序的权利；

（4）复制权，即将软件制作一份或者多份的权利；

（5）发行权，即以出售或者赠与方式向公众提供软件的原件或者复制件的权利；

（6）出租权，即有偿许可他人临时使用软件的权利，但是软件不是出租的主要标的的除外；

（7）信息网络传播权，即以有线或者无线方式向公众提供软件，使公众可以在其个人选定的时间和地点获得软件的权利；

（8）**翻译权**：~~将原作品从一种自然语言文字转换成另一种自然语言文字的权利~~；**对于软件产品来说**，将原软件从一种程序设计语言转换成另一种程序设计语言

（9）应当由软件著作权人享有的其他权利。

软件著作权在**软件创作完成**后即可获得，就是常说的进行软著登记，以起到类似公证的作用。著作权可以使你在别人对你的软件盗版时，采取保护措施，防止别人盗版

* **专利权**：是以技术方案的形式申请的，就是你软件流程图的内容。授权后，保护的是软件的构思，他人采用该构思，就可能构成侵权。因此软件专利的保护力度比软件著作权要大的多，能保护软件最核心的东西
* **数据流图**是结构化分析的工具，结构化方法就是采用自顶向下逐层分解的思想进行分析建模的。随着分解层次的增加，抽象的级别也越来越低，即越来越接近问题的解。数据流图建模应遵循:自顶向下、从抽象到具体的原则。
* **结构图**由模块、调用、数据、控制信息和转接符号5种基本符号组成。
* **程序设计语言**的基本成分包括数据、运算、控制和传输等。
* **程序设计语言**的控制成分包括顺序、选择和循环3种结构。
* **词法分析**阶段依据语言的词法规则，对源程序进行逐个字符地扫描，从中识别出一个个"单词符号，主要是针对词汇的检查。
* **语法分析**是在词法分析的基础上，根据语言的语法规则将单词符号序列分解成各类语法单位，如"表达式""语句"和“程序"等。语法规则就是各类语法单位的构成规则，主要是针对结构的检查。
* **语义分析**阶段分析各语法结构的含义，检查源程序是否包含语义错误，主要针对句子含义的检查。
* **符号表**在编译程序工作的过程中需要不断收集、记录和使用源程序中一些语法符号的类型和特征等相关信息。这些信息一般以表格形式存储于系统中。
* **代理**(Proxy）模式通过提供与对象相同的接口来控制对这个对象的访问，以使得在

确实需要这个对象时才对它进行创建和初始化。

* **组合** （Composite）模式描述了如何构造一个类层次式结构。
* **装饰模式**主要的目的是在无法生成子类的情况下给一个对象动态地增加新的职责;
* **享元设计**模式是共享大量细粒度的对象;
* **适配器**设计模式则是将己有的接口转换为系统希望的接口形式。
* 对象的状态包括这个对象的所有属性（通常是静态的）以及每个属性当前的值(通常是动态的）
* 行为是对象根据它的状态改变和消息传递所采取的行动和所作出的反应
* 模块化设计要求高内聚、低耦合；

在模块划分时需要遵循如下原则:

1. 模块的大小要适中
2. 模块的扇入扇出要合理
3. 深度和宽度适当

* 耦合性

非直接耦合:两个模块之间没有直接关系，它们之间的联系完全是通过主模块的控制和调用来实现的。

数据耦合:一个模块访问另一个模块时，彼此之间是通过简单数据参数(不是控制参数、公共数据结构或外部变量)来交换输入、输出信息的。

标记耦合︰一组模块通过参数表传递记录信息，就是标记耦合。

控制耦合:如果一个模块通过传送开关、标志、名字等控制信息，明显地控制选择另一模块的功能，就是控制耦合。

外部耦合:一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息，则称之为外部耦合。

公共耦合:若一组模块都访问同一个公共数据环境，则它们之间的耦合就称为公共耦合。

内容耦合:如果发生下列情形，两个模块之间就发生了内容耦合。

(1）一个模块直接访问另一个模块的内部数据;

(2)一个模块不通过正常入口转到另一模块内部;

(3）两个模块有一部分程序代码重叠(只可能出现在汇编语言中) ;

(4)一个模块有多个入口。

* 类可以分为三种:实体类、接口类(边界类)和控制类。实体类的对象表示现实世界中真实的实体，如人、物等。接口类(边界类)的对象为用户提供一种与系统合作交互的方式，分为人和系统两大类，其中人的接口可以是显示屏、窗口、Web窗体、对话框、菜单、列表框、其他显示控制、条形码、二维码或者用户与系统交互的其他方法。系统接口涉及到把数据发送到其他系统，或者从其他系统接收数据。控制类的对象用来控制活动流，充当协调者。
* 浮点数的表示分为阶和尾数两部分。两个浮点数相加时，需要先对阶，即将小阶向大阶对齐，同时将尾数右移n位
* 《中华人民共和国著作权法》和《计算机软件保护条例》是构成我国保护计算机软件著作权的两个基本法律文件。
* 《中华人民共和国著作权法》对著作权的保护期限作了如下规定:(1）著作权中的署名权、修改权、保护作品完整权的保护期不受限制。
* 软件维护工具主要有: 1、版本控制工具;2、文档分析工具;3、开发信息库工具;4、逆向工程工具;5、再工程工具;
* 瀑布模型是将软件生存周期各个活动规定为依线性顺序连接的若干阶段的模型，适合于软件需求很明确的软件项目，适用场景：需求明确、较大型系统、开发周期不紧张
* V模型是瀑布模型的一种演变模型，将测试和分析与设计关联进行，加强分析与设计的验证
* 原型模型是一种演化模型，通过快速构建可运行的原型系统，然后根据运行过程中获取的用户反馈进行改进

-原型模型不适宜大规模软件的开发

* 演化模型特别适用于对软件需求缺乏准确认识的情况，适用场景：需求不清、开发周期短的中小型系统
* 螺旋模型将瀑布模型和演化模型结合起来，加入了两种模型均忽略的风险分析
* 需求分析确定软件要完成的功能及非功能性要求
* 概要设计将需求转化为软件的模块划分，确定模块之间的调用关系
* 详细设计将模块进行细化，得到详细的数据结构和算法
* 编码根据详细设计进行代码的编写，得到可以运行的软件，并进行单元测试
* 需求不清晰且规模不太大时采用原型化方法最合适，而数据处理领域的不太复杂的软件，适于用结构化方法进行开发
* 软件项目计划的一个重要内容是安排进度，常用的方法有Gantt图和PERT图

- Gantt 图用水平条状图描述，它以日历为基准描述项目任务，可以清楚地表示任务的持续时间和任务之间的并行，但是不能清晰地描述各个任务之间的依赖关系

- PERT图是一种网络模型，描述一个项目的各任务之间的关系。可以明确表达任务之间的依赖关系，即哪些任务完成后才能开始另一些任务，以及如期完成整个工程的关键路径，但是不能清晰地描述各个任务之间的并行关系

* 硬盘容量分为格式化容量和非格式化容量两种：

格式化容量=面数x（磁道数/面）x（扇区数/道）x（字节数/扇区）

非格式化容量=面数x（磁道数/面）x内圆周长x最大位密度

* **海明校验码：2^r≥k+r+1** 其中r为校验位 ，k为信息位

海明校验码是一种可以纠正一位的差错码，是利用奇偶性来检错和纠正的校验码

海明码的码距必须大于1，码距为2可以检错，码距>=3可能纠错

海明码只能检测出2位错,纠1位错

* 奇偶校验码只能检错不能纠错
* 循环冗余校验码既可以检错，也纠错，码距为2
* 循环冗余校验码具有很强的检错能力
* 冗余技术又称储备技术，它是利用系统的并联模型来提高系统可靠性的一种手段。冗余附加件包括：独立设计的相同功能冗余备份程序的存储及调用；实现纠错检测及恢复的程序；为实现容错软件所需的固化程序。
* 原型模型不适宜大规模软件的开发。
* **父图与子图平衡**。父图中某加工的输入输出数据流必须与它的子图的输入输出数据流在数量和名字上相同。如果父图的一个输入（或输出）数据流对应于子图中几个输入（或输出）数据流，而子图中组成这些数据流的数据项全体正好是父图中的这一个数据流，那么它们仍然算是平衡的
* 在面向对象系统中，绑定是一个把过程调用和响应调用需要执行的代码加以结合的过程。在有些程序设计语言中，绑定是在编译时进行的，叫做**静态绑定**。在有些程序设 计语言中，绑定则是在运行时进行的，叫做**动态绑定**
* UML

类图中的关系有

-依赖

-关联关系

-泛化关系

-实现

用例图

-扩展关系

-包含关系

-泛化关系

* 编译器与解释器

二者比较

1、编译器需要将源程序译成可执行的目标代码，并进行优化。

2、编译器效率更高。因为是整个源程序直接翻译完成。

3、最大区别：

解释器，程序运行时的控制权在解释器而不在用户程序；

编译器：运行时的控制权在用户程序。

* 栈的特点是后进先出，队列的特点是先进先出。 因此，入队序列与出队序列一定相同。在入栈序列一定的情况下，由于元素的出栈时机不同，会形成不同的出栈序列，入栈序列与出栈序列可以相同，也可以不同
* 若类A的方法中仅仅用了类B的对象，那么类A***依赖***于类B。
* 如果类A的部分是B的对象组成，并且类A控制类B的生命周期，那么类A与类B是***组合***关系。
* 编译过程中为变量分配存储单元所用的地址是逻辑地址，程序运行时再映射为物理地址。
* CMM（Capability Maturity Model）是**能力成熟度模型**，每一较低级别是达到较高级别的基础。其中五级是最高级

一级为初始级，过程无序，进度、预算、功能和质量等方面不可预测

二级为可重复级，达到该级的软件公司过程已制度化，有纪律，可重复；

三级为已定义级，即过程实现标准化；

四级称为已管理级，达到该级的软件公司已实现过程的定量化；

五级优化级，达到该级的软件公司过程可自发地不断改进，防止同类问题二次出现；

* 下午第一题：

***- 实体：人、物、外部系统***

- E：实体 D：数据 P：加工

- 数据流的一端必须是加工

- 写确实的数据时，一定要注意数据流的起点或终点是不是有一个加工，如果两端都没有加工，那么就是找错了

- 保持父图与子图平衡。父图中某加工的输入输出数据流必须与它的子图的输入输出数据流在数量和名字上相同。如果父图的一个输入（或输出）数据流对应于子图中几个输入（或输出）数据流，而子图中组成这些数据流的数据项全体正好

是父图中的这一个数据流，那么它们仍然算是平衡的

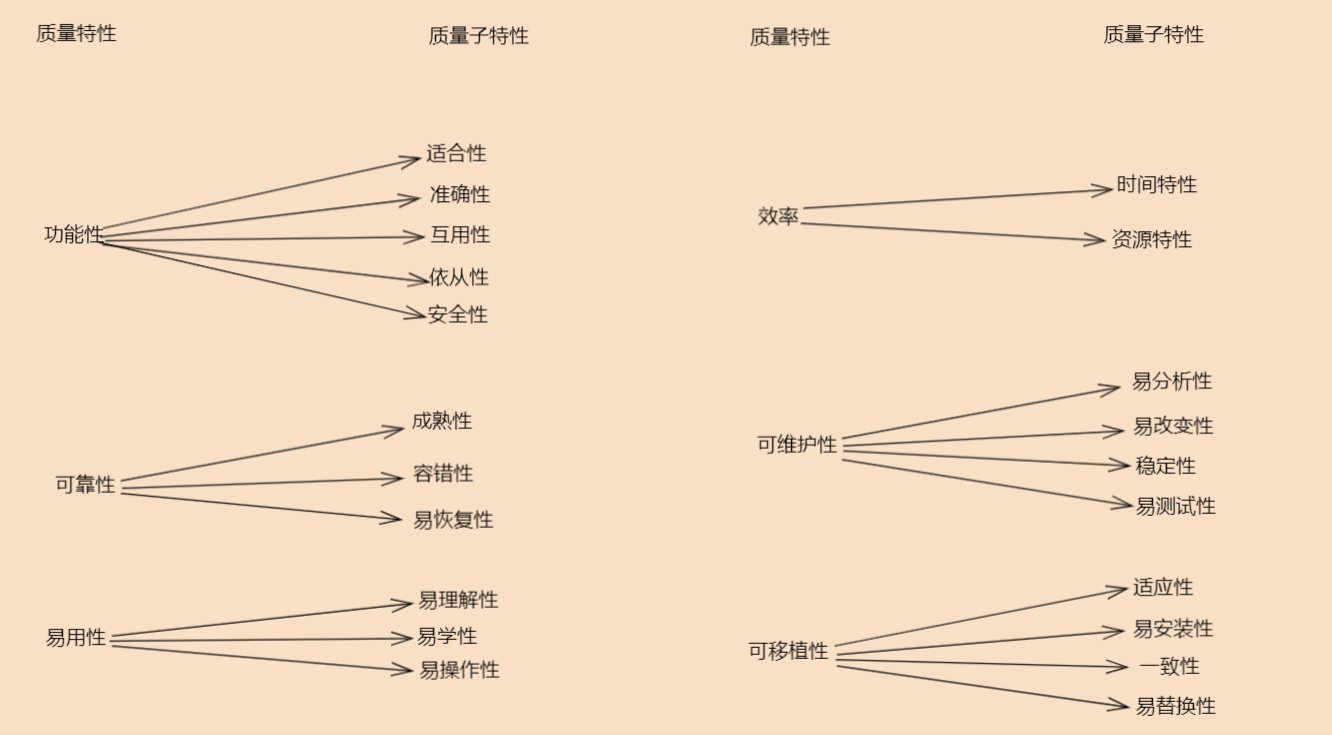
* 包含多态在许多语言中都存在，最常见的例子就是子类型化。
* 过载多态是同一个名字在不同的上下文中代表不同的含义。
* 著作权中修改权、署名权、保护作品完整权都是永久保护的。
* 线程共享的内容包括：进程代码段、进程的公有数据（利用这些共享的数据，线程很容易的实现相互之间的通讯）、进程打开的文件描述符、信号的处理器、进程的当前目录、进程用户ID与进程组ID 。线程独有的内容包括：线程ID、寄存器组的值、线程的堆栈、错误返回码、线程的信号屏蔽码。

|  |  |
| --- | --- |
| 内聚类型 | 描述 |
| 功能内聚 | 完成一个单一功能，各个部分协同工作，缺一不可 |
| 顺序内聚 | 处理元素相关，而且必须顺序执行 |
| 通信内聚 | 所处理元素集中在一个数据结构的区域上 |
| 过程内聚 | 处理元素相关，而且必须按特定的次序执行 |
| 瞬时内聚（时间内聚） | 所包含的任务必须在同一时间间隔内执行 |
| 逻辑内聚 | 完成逻辑上相关的一组任务 |
| 偶然内聚（巧合内聚） | 完成一组没有或关系松散的任务 |

* **软件可靠性**：MTTF/(1+MTTF) MTTF：平均无故障时间
* **软件可维护性**：1/(1+MTTR)
* ***路径覆盖***要使所有情况都走一下
* ***判定覆盖***使得程序中每个判断的取真分支和取假分支至少运行一次。
* 共同封闭原则：包中的所有类对于同一种性质的变化应该是共同封闭的。一个变化若对一个封闭的包产生影响，则将对该包中的所有类产生影响，而对于其他包则不造成任何影响。面向对象设计的原则之一。
* 共同重用原则：面向对象编程术语，指一个包中的所有类应该是共同重用的。如果重用了包中的一个类，那么也就相当于重用了包中的所有类。
* 开放-封闭原则：对扩展开放，对修改封闭。
* 接口隔离原则：使用多个专门的接口比使用单一的总接口要好
* +-0编码相同的是补码和移码
* 浮点数所能表示的数值范围主要由阶码来决定，所表示的数值精度由尾数决定
* 浮点数的**规格化**就是将尾数的绝对值限定在区间[0.5,1]
* 编译器和解释器
* 编译方式：词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成
* 解释方式：词法分析、语法分析、语义分析
* 编译器和解释器都不能省略 词法分析、语法分析、语义分析且顺序不可交换
* 编译器方式中间代码生成、代码优化不是必要的可以优化，可省略
* 确定有限自动机和非确定有限自动机都是词法分析的工具
* 语法分析方法有多种，根据产生**语法树**的方向，可分为自底向上和自顶向下两类。
* 自顶向下分析法：递归下降分析法和预测分析法。
* 自底向上分析法：算符优先分析法和LR分析法
* 软件风险一般包括不确定性和损失两个特性
* C/C++都是编译型语言，编译型语言处理过程为：预处理-编译-链接
* 对称密钥（私钥、私有密钥加密）算法：DES、3DES、AES、RC4、RC-5、IDEA
* 非对称密钥（公钥、公开密钥加密）算法：RSA、、DSA ECC

了解：

* Hash函数
* MD5、SHA摘要算法
* SHA-1安全散列算法
* 线性表
* 顺序存储（时间复杂度）：查O（1）、插O(n)、删O（n）
* 链式存储（时间复杂度），查、增、删， 复杂度都是O（n）
* 总结记忆：只有顺序表的查找时间复杂度时O(1),其他时间复杂度都是0(n)
* 存储密度顺序更优，容量分配链式更优
* 包含任意个空格字符的字符串成为**空白串**
* 对于循环队列：fear表示队列的对头位置，rear表示队尾元素的位置，M是队列容量，len表示队列的长度
* 求队头元素的指针的计算公式为：(rear-len+1+M)%M
* 队尾计算公式为指针为（fear+len-1+M）%M
* 求队列中元素个数公式为：(rear-fear+M)%M
* 栈和队列都是操作受限的线性表：栈仅在表尾插入和删除元素；队列仅在表头删除元素、在表尾插入元素
* 树
* 树的节点总数等于树中所有结点的度数之和加1
* 对于任何一棵二叉树，度为0的结点数等于度为2的结点数+1



* **静态查找表**有：顺序查找、折半（二分）查找、分块查找
* **动态查找表**有：二叉排序树、平衡二叉树、B\_树、哈希表
* 若一组记录的关键字的取值均在0到9（含）之间，则适宜采用**计数排序**算法
* 单元测试：
* 单元测试开始时，应该对通过所有被测模块的数据流进行测试。
* ①输入参数是否与形式参数是否一致。
* ②调用模块的实参与被调用模块形参是否一致。
* ③全局变量在各模块中的定义和用法是否一致
* ④外部输入、输出等等。
* 面向对象中，不同类的对象通过**消息**相互通信

设计模式

* **策略**（Strategy） 设计模式定义一系列算法，把它们一个个封装起来，并且使它们可相互替换。这一模式使得算法可独立于它的客户而变化。
* **抽象工厂**（Abstract Factory）模式提供一个创建一系列相关或相互依赖对象的接口，而无需指定他们具体的类。
* **观察者**（Observer）模式定义对象间的一种一对多的依赖关系，当一个对象的状恶发生改变时，所有依赖于它的对象都得到通知并被自动更新。
* **访问者**(Visitor）模式的主要意图是:表示一个作用于某对象结构中的各元素的操作。它允许在不改变各元素的类的前提下定义作用于这些元素的新操作。
* **桥接**（Bridge）模式的主要意图是:将抽象部分与其实现部分分离，使它们都可以独立地变化
* **状态**（State）模式是使得一个对象在其内部状态改变时通过调用另一个类中的方法改变其行为，使这个对象看起来如同修改了它的类。
* 按照设计模式的目的可以分为创建新模式、结构型模式以及行为型模式三大类
* 观察者模式、策略模式和访问者模式均为行为设计模式
* 桥接模式是结构型对象模式
* **数据字典**的条目包括：数据流、数据项、数据存储、基本加工
* **能力成熟度模型CMMI**的6个过程域能力等级（课本P247页）：
* CLO（未完成的):过程域未执行或未得到CL1中定义的所有目标。
* CL1(已执行的):其共性目标是过程将可标识的输入工作产品转换成可标识的输出工作产品，以实现支持过程域的特定目标。
* CL2（已管理的):其共性目标集中于已管理的过程的制度化。根据组织级政策规定过程的运作将使用哪个过程，项目遵循已文档化的计划和过程描述，所有正在工作的人都有权使用足够的资源，所有工作任务和工作产品都被监控、控制和评审。
* CL3(已定义级的):其共性目标集中于已定义的过程的制度化。过程是按照组织的剪裁指南从组织的标准过程集中剪裁得到的，还必须收集过程资产和过程的度量，并用于将来对过程的改进上。
* CL4(定量管理的):其共性目标集中于可定量管理的过程的制度化。使用测量和质量保证来控制和改进过程域,建立和使用关于质量和过程执行的定量目标作为管理准则。
* CL5（优化的):使用量化（统计学）手段改变和优化过程域，以对付客户要求的改变和持续改进计划中的过程域的功效。
* 软件设计一般包括两个阶段，即概要设计和详细设计。概要设计主要进行软件体系结构设计，逻辑数据结构设计，数据库设计和模块之间的接口设计。而详细设计主要进行模块内部的数据结构和算法的设计。
* **耦合**取决于各个模块之间接口的复杂程度、调用模块的方式以及通过接口的信息类型等
* 模块的内聚类型则与模块内各部分功能之间的关系有关。
* 目前常用的**软件调试**方法有如下几种(理解即可):
* (1）试探法。根据错误的特点，猜测问题的所在位置，利用在程序中设置输出语句，分析寄存器、存储器的内容等手段来获得错误的线索，一步步地试探和分析出错误所在。
* (2)回溯法。从发现错误症状的位置开始，人工沿着程序的控制流程往回跟踪代码，直到找出错误根源为止。
* (3)对分查找法。在该方法中，如果已经知道程序中的变量在若干位置的正确取值，可以在这些位置上给这些变量以正确值，观察程序运行输出结果，如果没有发现问题，则说明从赋予变量一个正确值到输出结果之间的程序没有错误，问题可能在除此之外的程序中。否则，错误就在所考察的这部分程序中，对含有错误的程序段再使用这种方法，直到把故障范围缩小到比较容易诊断为止。
* (4）归纳法。从测试所暴露的问题出发，收集所有正确或不正确的数据，分析它们之间的关系，提出假想的错误原因，用这些数据来证明或反驳，从而查出错误所在。
* (5）演绎法。根据测试结果，列出所有可能的错误原因。分析已有的数据，排除不可能和彼此矛盾的原因。对其余的原因，选择可能性最大的，利用已有的数据完善该假设，使假设更具体。用假设来解释所有的原始测试结果，如果能解释这一切，则假设得以证实，也就找出错误;否则，要么是假设不完备或不成立，要么有多个错误同时存在，需要重新分析，提出新的假设，直到发现错误为止。
* 对象的**状态**标识了该对象的所有属性以及每个属性的当前值
* Lisp是**函数式编程语言**，Prolog是**逻辑式程序语言**，Python支持过程式编程也支持面向对象编程，Java/C++是面向对象的编程语言。
* 交换机依据帧中的目的地址进行交换，属于链路层设备;
* 中继器对物理层传输的信号进行放大或再生，属于物理层设备;
* 路由器依据分组中的目的IP地址进行分组的转发，属于网络层设备;
* 网桥依据帧中目的地址进行交换，属于链路层设备。
* **软件详细设计**阶段的主要任务包括:对模块内的数据结构进行设计;对数据库进行物理设计;对每个模块进行详细的算法设计;

设计模式

* 按设计模式的目的可以分为三大类：
* 创建型模式，与对象的创建有关
* 结构型模式，处理类或对象的组合
* 行为型模式，对类或对象怎样交互和怎样分配职责进行描述
* **各种设计模型的意图（必背）：**

**创建型模式：**

* 抽象工厂：提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类
* 生成器：将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示
* 工厂方法：定义一个用于创建对象的接口，让子类决定实例化哪一个类
* 原型：用原型实例指定创建对象的种类，并且通过复制这些原型创建新的对象
* 单例:保证一个类仅有一个实例，并提供一个访问它的全局访问点

**结构型模式：**

* 适配器：将一个类的接口换成客户希望的另外一个接口
* 桥接：将抽象部分与实现部分分离，使它们都可以独立地变化
* 组合：将对象组合成树型结构以表示“部分-整体”的层次结构。
* 装饰：动态的给一个对象添加一些额外的职责
* 外观：为子系统的一组接口提供一个一致的界面
* 享元：运用共享技术有效的支持大量细粒度的对象
* 代理：为其他对象提供一种代理以控制对这个对象的访问

**行为型模式：**

* 责任链：使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系，将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它为止
* 命令：将一个请求封装为一个对象，从而使得可以用不同的请求对客户进行参数化；对请求排队或记录请求日志，以及支持可撤销的操作
* 解释器：给定一个语言，定义它的文法的一种表示，并定一个解释器，这个解释器使用该表示来解释语言中的句子
* 迭代器：提供一种方法顺序访问一个聚合对象中的各个元素，且不需要暴露该对象的内部表示
* 中介者:用一个中介对象来封装一系列的对象交互；
* 备忘录:在不破坏封装性的前提下捕获一个对象的内部状态，并在对象之外保存这个状态
* 观察者：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新
* 状态：允许一个对象在其内部状态改变时改变它的行为
* 模板方法：定义一个操作中的算法骨架，而将一些步骤延迟到子类中
* 访问者：表示一个作用于某对象结构中的各元素的操作。它允许在不改变各元素的类的前提下定义作用于这些元素的新操作
* IPv6地址由128位二进制表示