# 通识

1. MIME:互联网标准，扩展了电子邮件标准，使其能支持，与安全无关。与电子邮件相关的是S/MIME安全多用途互联网邮件扩展协议。
2. PGP（pretty Good Privacy, 优良保密协议），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。
3. 公开加密（非对称加密），RSA、ElGamal等。DSA：数据签名。DES是典型的私钥加密，属于对称加密。
4. Kerberos系统采用发送时间戳方案防止重放攻击。
5. 数据流图 系统建模 外部实体（Data Flow Diagram, DFD由数据流、加工、数据存储和外部实体4个要素构成。
6. X卧底软件是一种安装在手机里的监控软件
7. DMA(Direct Memory Access， DMA)是指数据在主存和I/O设备间直接块传送。
8. 循环冗余校验CRC校验方法中，采用模2法构造校验位。
9. 震网（Stuxnet），一种蠕虫病毒，专门定向攻击能源设施。
10. 著作权自作品完成之时就受到著作权保护了。
11. 流水线执行时间 = 流水线建立时间 + (n - 1) \* 流水线周期时间
12. 外部实体一般为组织机构、人员、第三方系统
13. 内聚类型：过程内聚、时间内聚、顺序内聚、逻辑内聚
14. 记号流，词法分析的输出是记号流，也就是语法分析的输入。
15. 源程序，词法分析的任务就是将源程序的字符串转换成单词符号序列。
16. 分析树，如果语法没错误，语法分析后就能正确构建出其语法树。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。
17. 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。
18. 最短移臂调度算法，即优先响应距离较近的磁道申请。柱面号即磁道号。
19. CMM根据软件过程的不同成熟度划分为5个等级，其中，1级被认为是成熟度最低的，5级被认为是成熟度最高的。
20. 需求分析阶段，就需要对验收测试、系统测试设计相关测试，撰写相关测试设计文档。
21. 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来说，软件维护工作要更复杂。
22. 采用面向对象方法进行系统开发，多对多联系需要单独转换为一个关系模式，也需要重新建类。
23. 软件实体(类、模块、函数等)应该是可以扩展但不能修改的，这属于开发封闭原则。
24. 运行时结合是动态绑定，编译时结合是静态绑定。

# 流水线执行时间

流水线执行时间=流水线建立时间+（n-1）\*流水线周期

# 系统可靠性分析

系统可靠性分析：串联： R = R1\*R2\*…\*Rn

串联系统失效率：λ=λ1+λ2+λ3+……+λn

并联：R = 1 – (1-R1)\*(1-R2)\*…(1-Rn)

并联系统失效率：λ= 1/((1/1\*λ1)+(1/2\*λ2)+(1/3\*λ3)+...+(1/n\*λn))

MTBF，即[平均故障间隔时间](https://baike.baidu.com/item/%E5%B9%B3%E5%9D%87%E6%95%85%E9%9A%9C%E9%97%B4%E9%9A%94%E6%97%B6%E9%97%B4/5928609)，英文全称是“Mean Time Between Failure”

MTBF=1/λ

*[并联后系统的故障率不再是常数]*

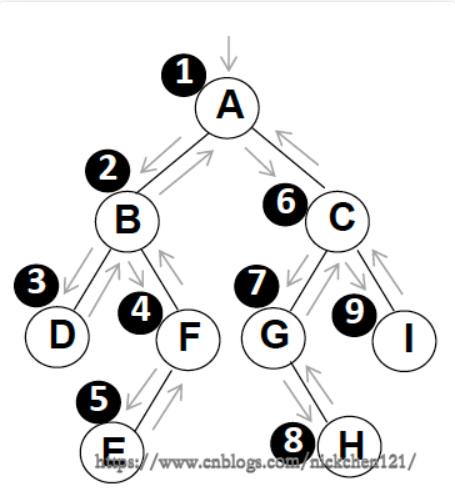
# 二叉树遍历运算



1. 先序遍历

遍历过程为：

1. 访问根结点
2. 先序遍历其左子树；
3. 先序遍历其右子树。

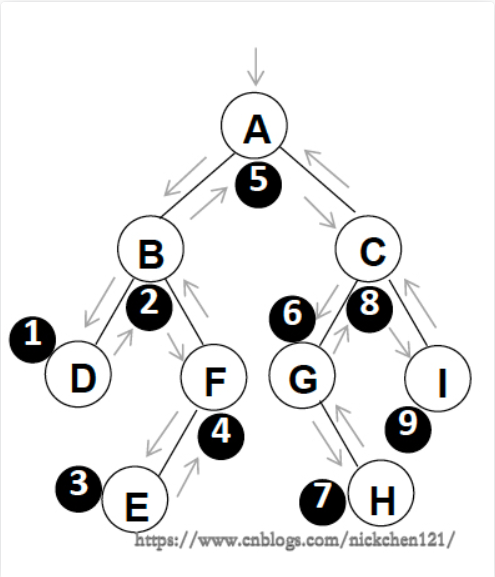


先序遍历：A （B D F E）（C G H I）

1. 中序遍历

遍历过程为：

1. 中序遍历其左子树；
2. 访问根节点；
3. 中序遍历其右子树。

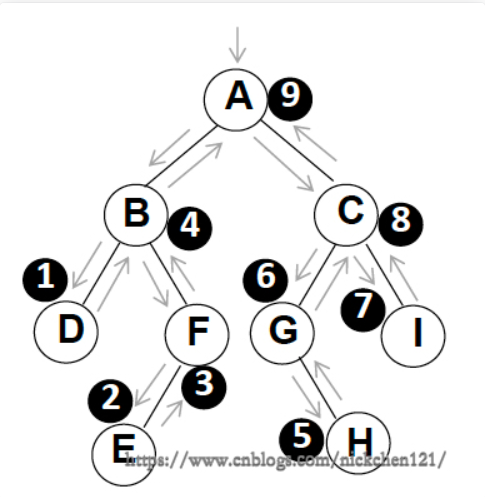


中序遍历：（D B E F）A（G H C I）

1. 后序遍历

遍历过程为：

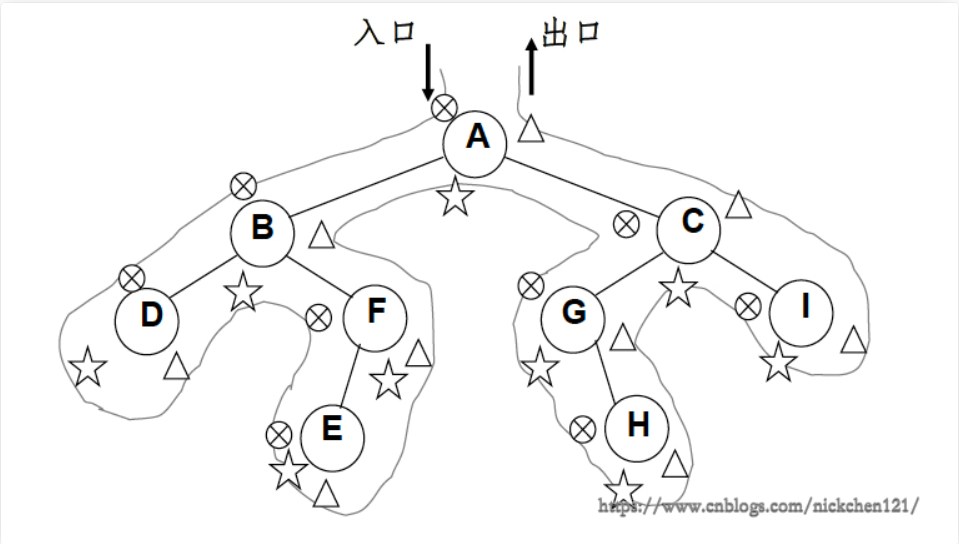
1. 后序遍历其左子树；
2. 后序遍历其右子树；
3. 访问根结点。



后序遍历：（D E F B）（H G I C）A

**小结：**先序、中序和后序遍历过程：遍历过程中经过结点的路线一样，只是访问各结点的时机不同。

图中在从入口到出口的曲线上用×、☆、△三种符号分别标记出了先序、中序和后序访问各结点的时刻。



# 函数参数

函数调用过程中，数据传递的方向是从实参到形参。传递方式时，传递数值，只要确定的即可，可以是常量、变量或表达式等。而采用传址传递方式时，传递的是地址，因此实参必须有地址。

# 语言基础知识

编译正确的程序可以消除词法机语法错误，但不能完全消除语义错误。

语义错误分两类，静态语义错误可以通过编译程序检测发现，而动态语义错误需要到运行时才能发现。

# MVC

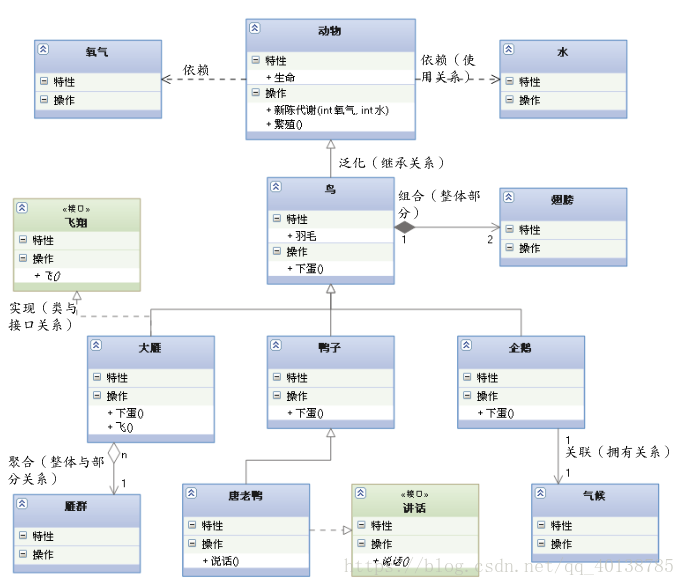
模型：应用问题域中包含的抽象领域知识。

视图：将应用问题域中的抽象领域知识呈现给用户的方法，一个模型可以用于多个视图。

控制器：用户界面对用户输入的响应方式。

# UML

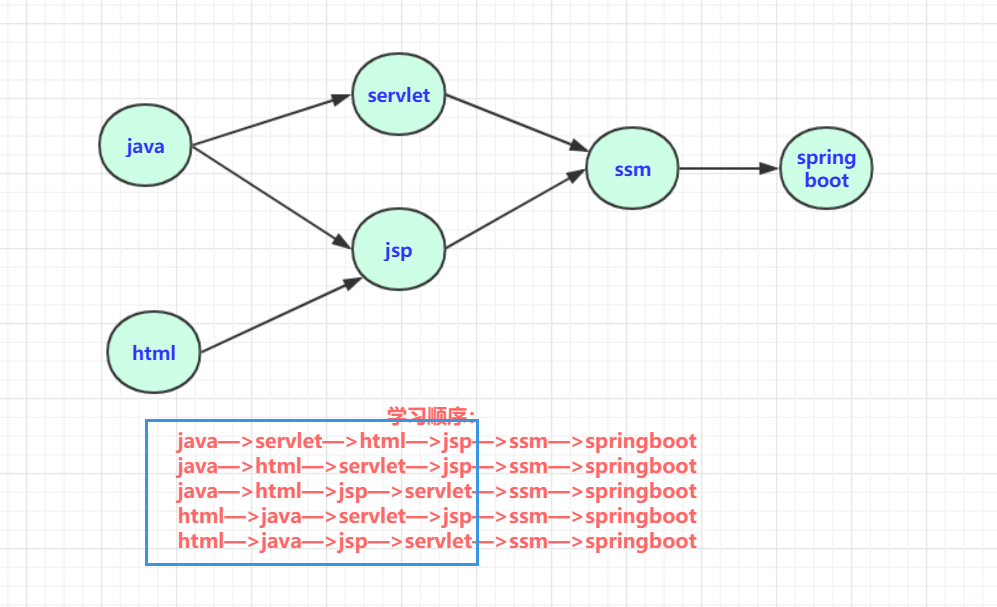
泛化（类与继承关系） = 实现（类与接口关系） > 组合（整体与部分的关系） > 聚合（整体与部分的关系） > 关联（拥有的关系） > 依赖（使用的关系）



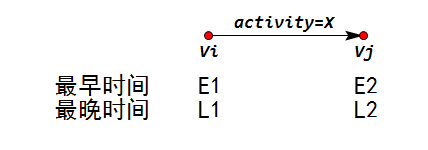
# UML建模之状态图

状态图重点在于描述对象的状态及其状态之间的转移，状态图的基本元素主要有：状态、转移、动作、自身转移、组合状态、进入节点、退出节点、历史状态、并发区域等，状态中的事件分为调用事件（Call）、变化事件（Change）、时间事件（Time）和信号事件(Singal)。最后以实例对状态对进行了分析。

# 拓扑排序

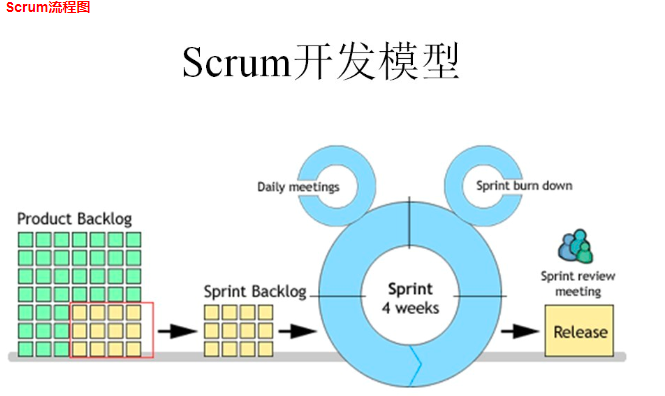


# 软件项目活动图



如果它是关键活动，则必然有**E1=L2-X**，否则它就不是关键活动

# 敏捷开发scrum开发模型



# 黑盒、白盒、灰盒测试

1. 黑盒测试

黑盒测试方法的主要特点是：执行测试的人员并不了解被测软件的内部结构和源代码。可以说，他们甚至不需要具备对于编程语言的深入了解，或出色的编程技能，便可开展测试。毕竟此类测试方法的目标并非深入研究代码，遍历软件内部，而是**直接与用户界面进行交互，测试其功能，并确保系统的每个输入与输出**，均符合既定的标准与要求。因此，黑盒测试也可以被称为**功能测试、或基于规范的测试**。

1. 白盒测试

与侧重于功能的黑盒测试相反，白盒测试方法的目标是对软件的内部结构、及其背后的逻辑进行分析。因此，**白盒测试有时也被称为结构测试、或逻辑驱动测试**。此类方法不但比较耗时，并且要求测试人员具有强大的编程能力，能够对所测软件进行全面了解，并且可以访问到所有的源代码、以及体系架构的相关文档。否则，测试人员将无法设计出适当的测试用例。

1. 灰盒测试

灰盒测试，是介于白盒测试与黑盒测试之间的一种测试，灰盒测试多用于**集成测试阶段**，不仅关注输出、输入的正确性，同时也关注程序内部的情况。灰盒测试不像白盒那样详细、完整，但又比黑盒测试更关注程序的内部逻辑，**常常是通过一些表征性的现象、事件、标志来判断内部的运行状态。**

# McCabe方法

环路复杂性V(G)计算方法：

V(G) = 流图中的区域数量(包括图外区域)

V(G) = 判定结点 + 1

V(G) = E – N + 2 （E:边数，N:结点数）