# 易混淆知识点

# 一 计算机组成与体系结构

## 易混淆点1：原、反、补码的运算

1、原码：最高位是符号位，其余低位表示数值的绝对值（0表示正数，1表示负数）。

2、反码：正数的反码与原码相同，负数的反码是其绝对值按位取反（符号位不变）。

3、补码：正数的补码与原码相同，负数的补码是其反码末位加1（符号位不变）。

4、移码：补码的符号位按位取反。

## 易混淆点2：寻址方式的对比

1、立即寻址方式：操作数直接在指令中，灵活性差，但速度最快。

2、直接寻址方式：指令中存放的是操作数的地址，。

3、间接寻址方式：指令中存放了一个地址，这个地址对应的内容是操作数的地址。

4、寄存器寻址方式：操作数存放在寄存器中，指令指定寄存器号。

5、寄存器间接寻址方式：寄存器内存放的是操作数的地址。

## 易混淆点3：数据传输方式

1、程序控制（查询）方式：分为无条件传送和程序查询方式两种。方法简单，硬件开销小，但I/O能力不高，严重影响CPU的利用率（不可与CPU并行）。

2、程序中断方式：与程序控制方式相比，中断方式因为CPU无需等待而提高了传输请求的响应速度（可与CPU并行）。

3、DMA方式：DMA方式是为了在主存与外设之间实现高速、批量数据交换而设置的，DMA方式比程序控制方式与中断方式都高效（可与CPU并行）。

## 易混淆点4：可靠性、可用性、可维护性

1、可靠性可以用MTTF/（1+MTTF）来度量。

2、可用性可以用MTBF/（1+MTBF）来度量。

3、可维护性可以用MTTR/（1+MTTR）来度量。

## 易混淆点5：RISC和CISC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **指令系统类型** | **指令** | **寻址方式** | **实现方式** | **其他** |
| CISC（复杂） | 数量多，使用频率差别大，可变长格式 | 支持多种 | 微程序控制技术（微码） | 研制周期长，编译子程序库小 |
| RISC（精简） | 数量少，使用频率接近，定长格式，大部分为单周期指令，操作寄存器，只有Load/Store操作内存 | 支持方式少 | 增加了通用寄存器，硬布线逻辑控制为主，适合采用流水线 | 优化编译，有效支持高级语言，编译子程序库大 |

# 二 数据结构

## 易混淆点1：顺序存储与链式存储

|  |  |  |  |
| --- | --- | --- | --- |
| **性能类别** | **具体项目** | **顺序存储** | **链式存储** |
| 空间性能 | 存储密度 | =1，更优 | <1 |
| 容量分配 | 事先确定 | 动态变化，更优 |
| 时间性能 | 查找运算 | O（n） | O（n） |
| 读运算 | O（1），更优 | O（n），最好情况为1，最坏情况为n |
| 插入运算 | O（n）,最好情况为0，最坏情况为n | O（1）,更优 |
| 删除运算 | O（n） | O（1），更优 |

## 易混淆点2：空串与空格串

1、空串：长度为零，不包含任何字符。

2、空格串：由一个或多个空格组成的串。虽然空格是一个空白字符，但它也是一个字符，在计算串长度时要将其计算在内。

## 易混淆点3：子串和子序列

1、子串：由串中任意长度的连续字符构成的序列称为子串。含有子串的串称为主串。子串在主串中的位置是指子串首次出现时，该子串的第一个字符在主串中的位置。空串是任意串的子串。

2、子序列：一个串的“子序列”是将这个串中的一些字符提取出来得到一个新串，并且不改变它们的相对位置关系。

子串要求连续，而子序列要求不改变相对位置即可，例如：ABC的子串为AB，BC，而子序列可以为AC。

## 易混淆点4：树的遍历

1、前序遍历（根左右）

2、中序遍历（左根右）

3、后序遍历（左右根）

4、层次遍历（从上到下，从左到右）

## 易混淆点5：图的遍历—深度优先和广度优先

1、深度优先：从图中一个未访问的顶点开始，沿着一条路一直走到底，然后回退到上一个节点，再从另一条路开始走到底。

2、广度优先：从图的一个未遍历的节点出发，先遍历这个节点的相邻节点，再依次遍历每个相邻节点的相邻节点。

# 三 算法基础

## 易混淆点1：各类排序算法对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **类别** | **排序方法** | **时间复杂度** | | **空间复杂度** | **稳定性** |
| **平均情况** | **特殊情况** | **辅助** |
| 插入排序 | 直接插入 | O（n2） | 基本有序最优O（n） | O（1） | 稳定 |
| Shell排序 | O（n1.3） | - | O（1） | 不稳定 |
| 选择排序 | 直接选择 | O（n2） | - | O（1） | 不稳定 |
| 堆排序 | O（nlog2n） | - | O（1） | 不稳定 |
| 交换排序 | 冒泡排序 | O（n2） | - | O（1） | 稳定 |
| 快速排序 | O（nlog2n） | 基本有序最差O（n2） | O（log2n） | 不稳定 |
| 归并排序 | | O（nlog2n） | - | O（n） | 稳定 |
| 基数排序 | | O(d（n+rd）) | - | O（rd） | 稳定 |

## 易混淆点2：常见算法特征总结

|  |  |  |  |
| --- | --- | --- | --- |
| **算法名称** | **关键点** | **特征** | **典型问题** |
| 分治法 | 递归技术 | 把一个问题拆分成多个小模块的相同子问题，一般可用递归解决。 | 归并排序、快速排序、二分搜索 |
| 贪心法 | 一般用于求满意解，特殊情况可求最优解（部分背包） | 局部最优，但整体不见得最优。每步有明确的，既定的策略。 | 背包问题（如装箱）、多机调度、找零钱问题 |
| 动态规划法 | 最优子结构和递归式 | 划分子问题（最优子结构），并把子问题结果使用数组存储，利用查询子问题结果构造最终问题结果。 | 矩阵乘法、背包问题、LCS最长公共子序列 |
| 回溯法 | 探索和回退 | 系统的搜索一个问题的所有解或任一解。有试探和回退的过程。 | N皇后问题、迷宫、背包问题 |

# 四 程序设计语言与语言处理程序基础

## 易混淆点1：编译与解释

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **编译型语言** | **解释型语言** |
| **共同点** | | 高级程序语言 | |
| 有词法分析、语法分析、语义分析过程 | |
| **不同点** | 翻译程序 | 编译器 | 解释器 |
| 是否生成目标代码 | 生成目标代码 | 不会生成目标代码 |
| 目标程序能够直接执行 | 目标程序直接执行 | 边解释边执行 |
| 翻译程序是否参与执行 | 编译器不参与执行 | 解释器参与执行 |
| 执行效率 | 执行效率高 | 执行效率低 |
| 灵活性与可移植性 | 灵活性差，可移植性差 | 灵活性好，可移植性强 |

## 易混淆点2：传值和传址调用

|  |  |
| --- | --- |
| **传递方式** | **主要特点** |
| 传值调用 | 形参取的是实参的值，形参的改变不会导致调用点所传的实参的值发生改变。 |
| 传址调用（引用调用） | 形参取的是实参的地址，即相当于实参存储单元的地址引用，因此其值的改变同时就改变了实参的值。 |

# 五 软件工程

## 易混淆点1：内聚性

|  |  |
| --- | --- |
| **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 | 所有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） | 完成一组没有关系或松散关系的任务 |

## 易混淆点2：耦合性

|  |  |
| --- | --- |
| **耦合类型** | **描述** |
| 非直接耦合 | 两个模块直接没有直接关系，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 | 一组模块借助参数表传递简单数据 |
| 标记耦合 | 一组模块通过参数表传递记录信息（数据结构） |
| 控制耦合 | 模块之间传递的信息中包含用于控制模块内部逻辑的信息 |
| 外部耦合 | 一组模块都访问同一全局简单变量，而且不是通过参数表传递该全局变量的信息 |
| 公共耦合 | 多个模块都访问同一个公共数据环境 |
| 内容耦合 | 一个模块直接访问另一个模块的内部数据；一个模块不通过正常入口转到另一个模块的内部；两个模块有一部分程序代码重叠；一个模块有多个入口 |

## 易混淆点3：概要设计与详细设计

1、概要设计：是指整个体系结构的设计，模块与模块之间的设计，对应的是测试的集成测试。

2、详细设计：是模块内部的设计，对应是测试的单元测试。

## 易混淆点4：软件维护类型

1、改正性维护：是指改正正在系统开发阶段已发生而系统测试阶段尚未发现的错误（强调改正错误）。

2、适应性维护：适应性维护是指使应用软件适应信息技术变化和管理需求变化而进行的修改（强调适应环境）。

3、完善性维护：这是为扩充功能和改善性能而进行的修改，主要是指对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征（强调完善功能）。

4、预防性维护：为了改进应用软件的可靠性和可维护性，为了适应未来的软/硬件变化，应主动增加预防性的新功能（强调提前预防）。

# 六 面向对象技术

## 易混淆点1：UML图中关系

1、依赖关系：一个事物发生变化影响另一个事物，例如：y=x+1，我们就说y依赖于x，y的值随着x的值变化而变化。

2、泛化关系：父子关系，一般与特殊的关系，例如动物类和猫类。

3、关联关系：两者之间用链进行连接，一般表示两个类进行通信。

4、聚合和组合关系：强调部分与整体关系，前者强调生命周期不同，后者强调生命周期相同，比如：大雁和翅膀属于组合关系，大雁和雁群属于聚合关系。

5、实现关系：是接口和类的关系。

# 七 知识产权与标准化

## 易混淆点1：知识产权人确定

|  |  |  |  |
| --- | --- | --- | --- |
| **情况说明** | | **判断说明** | **归属** |
| 作品软件 | 委托开发 | 有合同约定著作权归属 | 委托方 |
| 合同中未约定著作权归属 | 创作方 |
| 合作开发 | 只进行组织、提供咨询意见、物质条件或者进行其他辅助工作 | 不享有著作权 |
| 共同创作的 | 共同享有，按人头比例  成果可分割的，可分开申请 |
| 商标 | | 谁先申请谁拥有（除知名商标的非法抢注），无法协商时，抽签决定 | |
| 同时申请，则根据谁先使用（需提供证据） | |
| 专利 | | 谁先申请谁拥有，同时申请则协商归属，但不能够同时驳回双方的专利申请 | |

# 八 操作系统

## 易混淆点1：页式存储、段式存储和段页式存储

1、页式存储：将程序与内存均划分为同样大小的块，以页为单位将程序调入内存。

2、段式存储：按用户作业中的自然段来划分逻辑空间，然后调入内存，段的长度可以不一样。

3、段页式存储：段式与页式的综合体。先分段，再分页。1个程序有若干个段，每个段中可以有若干页，每个页的大小相同，但每个段的大小不同。

# 九 数据库系统

## 易混淆点1：分布式数据透明性

1、分片透明：强调用户对如何进行分片不必关心。

2、复制透明：强调用户对复制情况不必关心。

3、位置透明：强调用户对操作数据位置放在何处不必关心。

4、逻辑透明：是最低层次透明性，强调用户对底层数据模型、操作语言不必关心。

## 易混淆点2：逻辑独立性和物理独立性

1、逻辑独立性：数据的逻辑结构发生变化后，用户程序也可以不修改。但是为了保证应用程序能够正确执行，需要修改外模式和概念模式之间的映像。

2、物理独立性：数据的物理结构发生改变时，应用程序不用改变。但是为了能够保证应用程序能够正确执行，需要修改概念模式和内模式之间的映像。

# 十 计算机网络

## 易混淆点1：TCP和UDP协议

|  |  |  |
| --- | --- | --- |
|  | **TCP** | **UDP** |
| 共同点 | 基于IP协议的传输层协议，可以端口寻址 | |
| 不同点 | 面向连接（连接管理）、三次握手、流量控制、差错校验和重传、IP数据报按序接收不丢失不重复、可靠性强、牺牲通信量、效率低 | 不可靠、无连接、错误检测功能弱，无拥塞控制、无流量控制，有助于提高传输的高速率性。  不对无序IP数据报重新排序、不负责重传、不消除重复IP数据报、不对已收到的数据报进行确认、不负责建立或终止连接，这些由UDP进行通信的应用程序进行处理。 |
| 相关协议 | HTTP、FTP、Telnet、POP3、SMTP | DNS、DHCP、TFTP、SNMP |

# 十一 信息安全

## 易混淆点1：对称加密和非对称加密

1、对称加密技术：Ke=Kd；加密解密共用一个密钥；

特点：加密强度不高，但效率高；密钥分发困难。

常见对称密钥（共享密钥）加密算法：DES、AES、3DES(三重DES)、RC-5、IDEA算法。

2、非对称加密技术：Ke ≠ Kd；密钥必须成对使用（公钥加密，相应的私钥解密）。

特点：加密速度慢，但强度高。

常见非对称密钥（公开密钥）加密算法： RSA、DSA、ECC。

## 易混淆点2：数字签名和数字加密

1、数字签名：使用发送方的密钥对，发送方用自己的私有密钥进行加密，接收方用发送方的公开密钥进行解密，是一对多的关系，任何拥有发送方公开密钥的人都可以验证数字签名的正确性。采用非对称密钥加密算法，保证发送信息的完整性、身份认证和不可否认性。

2、数字加密：使用接收方的密钥对，是多对一的关系，任何知道接收方公开密钥的人都可以向接收方发送加密信息，只有唯一拥有接收方私有密钥的人才能对信息解密。采用对称密钥加密算法和非对称密钥加密算法相结合的方法，保证发送信息的保密性。

# 十二 项目管理

## 易混淆点1：Gantt图和PERT图

1、Gantt图

优点：能够清晰描述每个任务从何时开始到何时结束，任务的进程情况以及各个任务之间的并行关系。

缺点：不能清晰反映出各个任务之间的依赖关系，难以确定整个项目的关键所在，也不能反映计划中有潜力的部分。

2、PERT图

优点：不仅给出了每个任务的开始时间、结束时间和完成任务所需的时间，还给出了任务之间的关系，即哪些任务完成之后才能开始另外的一些任务，以及如期完成整个工程的关键路径。图中的松弛时间则反映了某些任务是可以推迟其开始时间或延长其所需完成的时间。

缺点：不能反映任务之间的并行关系。