



PERSONAL INFORMATION



★ 软考题库 - 微信搜一搜

Q 软考达人

7/27/2017

考试科目: 中级 软件设计师

上午 9:00 — 11:30 综合知识

下午 2:00 — 4:30 应用技术

考试要求

1. 掌握计算机内的数据表示、算术和逻辑运算方法;
2. 掌握相关的应用数学及离散数学基础知识;
3. 掌握计算机体系结构以及各主要部件的性能和基本工作原理;
4. 掌握操作系统、程序设计语言的基础知识, 了解编译程序的基本知识;
5. 熟练掌握常用数据结构 and 常用算法;
6. 熟练数据库、网络和多媒体的基础知识;
7. 掌握 C 程序设计语言, 以及 C++、Java 中的一种程序设计语言;
8. 熟悉软件工程、软件过程改进和软件开发项目管理的常识;
9. 了解常用信息技术标准、安全性, 以及有关法律、法规的基本知识;
10. 了解信息化、计算机应用的基础知识;
11. 正确阅读和理解计算机领域的简单英文资料。

学习资料

1. 《软件设计师教程(第5版)》
2. 《知识点练习题》、《章节练习题》、《冲刺练习题》, 近5年10套真题
3. 错题集

总分 75分

第一章 计算机组成与体系结构

常考选择题 6分左右

内容提要

- 1. 数据的表示 ★★★★★
- 2. 运算器与控制器 ★★
- 3. Flynn分类法 ★★
- 4. CISC和RISC ★★
- 5. 存储系统 ★★
- 6. 流水线技术 ★★
- 7. 总线系统 ★
- 8. 可靠性 ★
- 9. 校验码 ★★★★★

一、数据的表示

机器数 0, 1

1. 按权展开法: R进制转十进制

例: 二进制 $10100.01 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^{-2}$

七进制 $604.01 = 6 \times 7^2 + 4 \times 7^0 + 1 \times 7^{-2}$ 权

2. 短除法: 十进制转R进制

例: 94转二进制为 (1011110)

2	94	
2	47	0 ↑
2	23	1
2	11	1
2	5	1
2	2	1
2	1	0
	0	1

1. 在微机系统中, BIOS(基本输入输出系统)保存在主板上的ROM。

2. 在计算机中, M补码(表示数据位), 表示范围是 $-2^M \sim 2^M - 1$, 其中最小值是人为定义的。

例: $(-128)_{补} = 10000000$

3. 若有奇数数据出错, 则可以检测出该错并加以纠正 可以

3. 二进制与八进制、十六进制相互转换

10	001	110
2	1	6

1000	1110
8	E

十进制	DX, H.
八进制	0.

4. 码制

原码 反码 补码 移码 符号位: 0正1负

正数的原码, 反码, 补码 均相同。

负数反码: 除符号位按位取反, 补码: 反码+1, 移码: 补码符号位取反

数值表示范围

定点整数 —— 小数点在最后 } 小数点不占位数
 定点小数 □ —— 符号位

码制	定点整数	定点小数	n 为机器字长
原码	$-(2^{n-1}-1) \sim +(2^{n-1}-1)$	$-(1-2^{-(n-1)}) \sim +(1-2^{-(n-1)})$	
反码	$-(2^{n-1}-1) \sim +(2^{n-1}-1)$	$-(1-2^{-(n-1)}) \sim +(1-2^{-(n-1)})$	
补码	$-2^{n-1} \sim +(2^{n-1}-1)$	$-1 \sim +(1-2^{-(n-1)})$	
移码	$-2^{n-1} \sim +(2^{n-1}-1)$	$-1 \sim +(1-2^{-(n-1)})$	

↓ 少-0

三、浮点运算

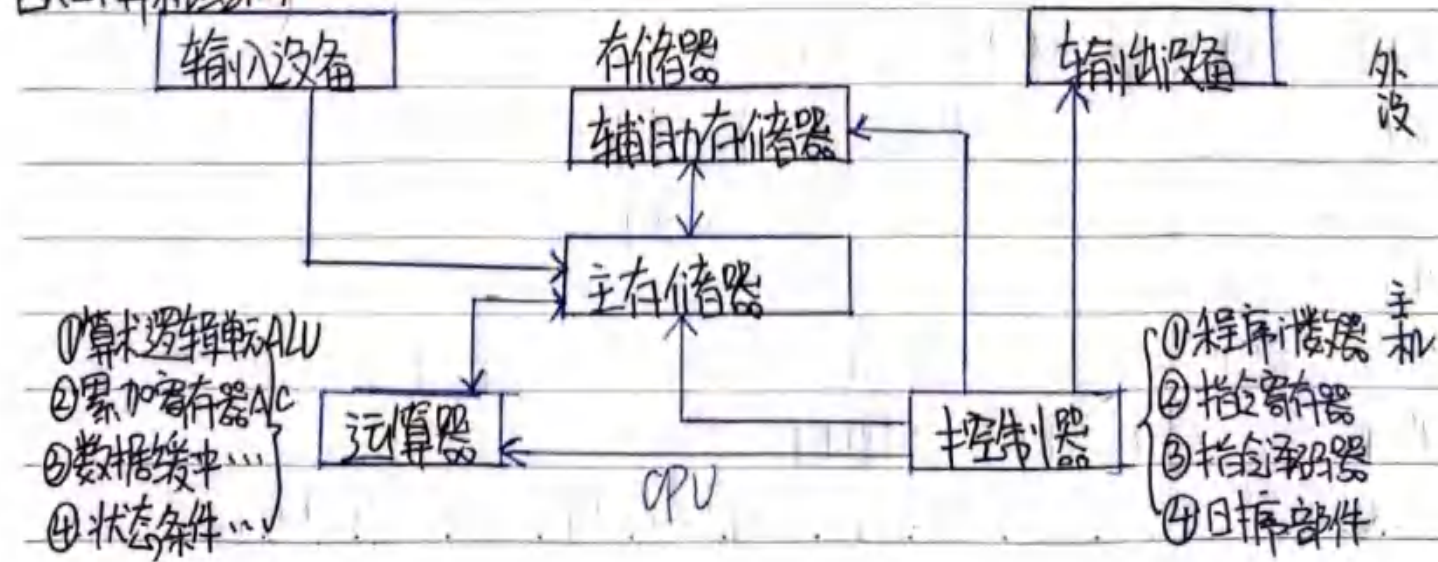
1. 浮点数表示: $N = \text{尾数} \times \text{基数}^{\text{指数}}$ 例: 3.14×10^3

2. 运算过程: 对阶 → 尾数计算 → 结果格式化

小阶对大阶 算术右移

3. 特点:
- ① 一般尾数用补码, 阶码用移码
 - ② 阶码的位数决定数的表示范围, 位数越多范围越大
 - ③ 尾数的位数决定数的有效精度, 位数越多精度越高
 - ④ 对阶时, 小阶对大阶
 - ⑤ 对阶时, 采用算术右移实现
- | | | | |
|----|----|----|----|
| 阶码 | 阶码 | 数符 | 数码 |
|----|----|----|----|

四、计算机结构



运算器

- ① 算术逻辑单元 ALU: 数据的算术运算和逻辑运算
- ② 累加寄存器 AC: 通用寄存器, 为 ALU 提供一个工作区, 用于暂存数据
- ③ 数据缓冲寄存器 DR: 写内存时, 暂存指令或数据
- ④ 状态条件寄存器 PSW: 存状态标志, 与控制标志, (译: 也有将其归为控制器的)

控制器

- ① 程序计数器 PC: 存储下一条要执行指令的地址
- ② 指令寄存器 IR: 存储即将执行的指令, 包括操作码, 地址码
- ③ 指令译码器 ID: 对指令的操作码字段进行分析解释
- ④ 时序部件: 提供时序控制信号

五计算机体系结构分类 — Flynn

✓ 体系结构类型	结构	关键特性	✓ 代表
单指令流单数据流 SISD	控制部分: 一个 处理器: 一个 主存模块: 一个		单处理器系统 - CPU
单指令流多数据流 SIMD	控制部分: 一个 处理器: 多个 主存模块: 多个	各处理器以同步的形式执行同一条指令	并行处理机 阵列处理 超级向量处理机
多指令流单数据流 MISD	控制部分: 多个 处理器: 一个 主存模块: 多个	被证明不可能, 至少是不实际	目前没有, 有文献称流水线计算机为此类
多指令流多数据流 MIMD	控制部分: 多个 处理器: 多个 主存模块: 多个	能够真正实现作业任务, 指令等各级全面并行	多处理机系统 多计算机

} 按不同结构材料

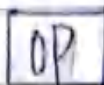
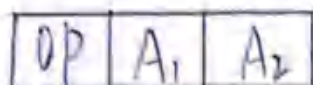
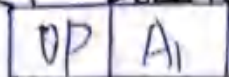
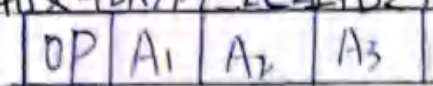
指令的基本概念

一条指令就是机器语言的一个语句, 它是一组有意义的二进制代码, 基本格式如下:

操作码字段 地址码字段

操作码部分指出了计算机要执行什么性质的操作, 如加、减、取数、存数等。

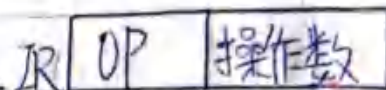
地址码字段需要包含各操作数的地址及操作结果的存放地址等,从地址结构的
角度可以分为三地址指令,二地址指令,一地址指令、零地址指令。



寻址方式

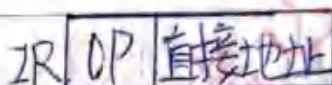
立即寻址方式

特点:操作数直接在指令中,速度快,灵活性差



直接寻址方式

特点:指令中存放的是操作数的地址

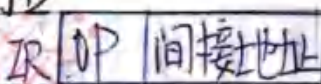


存

操作数

间接寻址方式

特点:指令中存放一个地址,这个地址对应的内容
是操作数的地址。

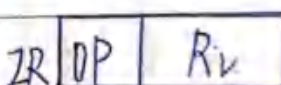


操作数地址

操作数

寄存器寻址方式

特点:寄存器存放操作数

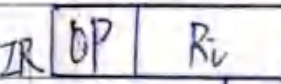


R_n

操作数

寄存器间接寻址方式

特点:寄存器内存放的是操作数的地址



R_n

操作数地址

操作数

八 CISC与RISC

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)	不开闭周期长
RISC (精简)	数量少,使用频率近, 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 控制内主适合采用 流水线	优化编译,有效 支持高级语言

寻址方式单一

~~CISC与RISC比较, 分哪些维度?~~

指令数量, 指令使用频率, 寻址法, 寄存器, 流水线支持, 高级语言支持



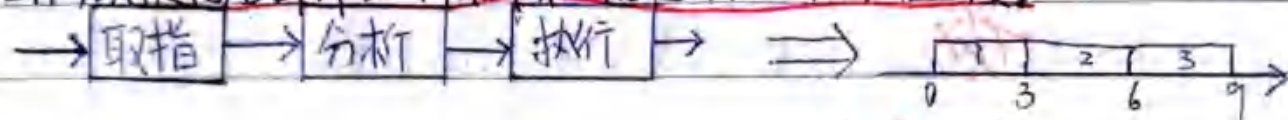
CISC: 复杂, 指令数量多, 频率差别大, 多寻址

RISC: 精简, 指令数量少, 操作寄存器, 单周期, 少寻址, 多通用寄存器, 流水线

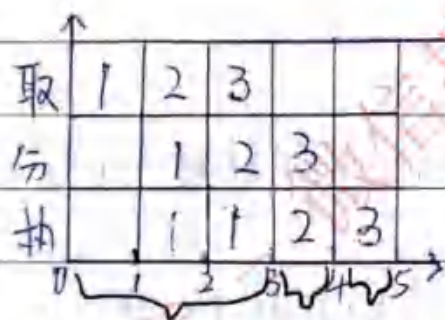
九、流水线 - 概念

相关参数计算: 流水线执行时间计算, 流水线吞吐率, 流水线加速比, 流水线效率

流水线是指在程序执行时, 多条指令重叠进行的一种准并行处理实现技术, 各种部件同时处理是针对不同指令而言的, 它们可同时为多条指令的不同部分进行工作, 以提高各部件的利用率和指令的平均执行速度。



↓ 优化



流水线周期为执行时间最长的一段: 取, 分, 执

~~流水线计算公式:~~

1. 指令执行时间 + (指令条数 - 1) * 流水线周期

① 理论公式: $(t_1 + t_2 + \dots + t_k) + (n-1) * \Delta t$

② 实践公式: $k * \Delta t + (n-1) * \Delta t$ → 流水线建立时间

例: 一条指令的执行过程可分解为取指, 分析, 执行三步, 在取指时间 t 取值 = $3\Delta t$,

分析 = $2\Delta t$, 执行 = $4\Delta t$, 若按串行方式执行, 则10条指令全部执行完需 $(90\Delta t)$;

若按流水线的3步执行, 流水线周期为 $(4\Delta t)$, 则10条指令全部执行完需 $(44\Delta t)$;

超标量流水线 — 增加流水线 度

$45\Delta t$

流水线吞吐率计算

定义: 是指单位时间内流水线所完成的任务数量或输出的结果数量, 公式如下:

指令条数

TP = 流水线执行时间

流水线最大吞吐率:

$$TP_{max} = \lim_{n \rightarrow \infty} \frac{n}{(k+n-1)\Delta t} = \frac{1}{\Delta t} \rightarrow \text{周期}$$

十层次化存储结构

速度快 ↑ CPU : 寄存器, 最快, 容量小, 成本高

(容量小)



Cache : 按内容存取



内存(主存) 分两类: 随机存取存储器(RAM), 只读存储器(ROM)

速度慢 ↓



(容量大)

外存(辅存) 硬盘, 光盘, U盘等

十一 Cache - 一根线

在计算机的存储系统体系中, Cache是访问速度最快的层次(若有寄存器, 则寄存器最快).

使用Cache改善系统性能的依据是程序的局部性原理 {时间 空间 防抖动}

如果以h代表对Cache的访问命中率, t_1 表示Cache的周期时间, t_2 表示主存周期时间, 以读操作为例, 使用“Cache + 主存存储器”的系统的平均周期为 t_3 , 则:

$$t_3 = h \times t_1 + (1-h) \times t_2$$

其中, $(1-h)$ 又称失效率(未命中率)

Cache - 映像

① 直接相联映像: 硬件电路较简单, 但冲突率很高。

② 全相联映像: 电路难于设计和实现, 只适用于小容量Cache, 冲突率较低。

③ 组相联映像: 直接相联与全相联的折中, 组间直接相联, 组内全相联。

地址映像是将主存与Cache的存储空间分成若干大小相同的块(或称为块)。

例: 某机的主存容量为1GB, 划分为2048页, 每页512KB;

Cache容量为8MB, 划分为16页, 每页512KB。

十二 主存一编址与计算

存储单元

字长

按字编址: 存储体的存储单元是字存储单元, 即最小寻址单位是一个字。

按字节编址: 存储体的存储单元是字节存储单元, 即最小寻址单位是一个字节。

B

1B = 8位

根据存储器所要求的容量和选定的存储芯片的容量, 就可以计算出所需芯片的总数, 即

$$\star \text{总片数} = \text{总容量} / \text{每片的容量}$$

例: 若内存地址区间为 4000H ~ 43FFH, 每个存储单元可存储 16 位二进制数, 该内存区域用 4 片存储器芯片构成, 则构成该内存所用的存储芯片的容量是多少?

$$43FFH - 4000H + 1 = 4001H = 2^{10}$$

$$\frac{2^{10} \times 16\text{bit}}{4} = 256 \times 16\text{bit}$$

4

十三 总线

一条总线同时只允许一个设备发送, 但允许多个设备接收。

总线的分类: (总线结构)

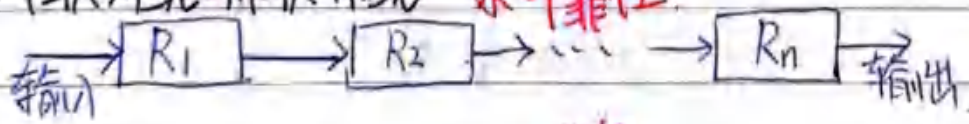
数据总线: 在 CPU 和 RAM 之间来回传递需要处理或是需要存储的数据。

地址总线: 用来指定在 RAM 之中存储的数据的地址。

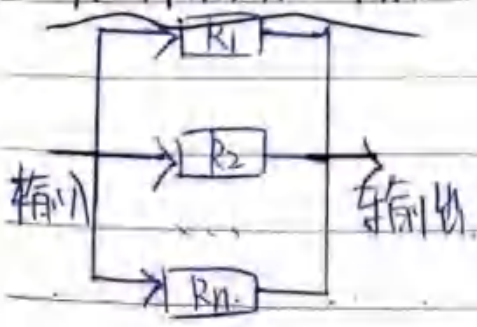
控制总线: 将微处理器控制单元的信号, 传送到周边设备, 一般常见的为 USB Bus 和 1394 Bus。

十四 串联系统与并联系统

求可靠性



$$R = R_1 \times R_2 \times \dots \times R_n$$



可靠性

$$\lambda = \lambda_1 + \lambda_2 + \dots + \lambda_n$$

$$R = 1 - (1 - R_1) \times (1 - R_2) \times \dots \times (1 - R_n)$$

$$u = \frac{1}{\frac{1}{\lambda} \sum_{j=1}^n \lambda_j}$$



十五 N模混合系统



$$R \times (1 - (1-R)^3) \times (1 - (1-R)^2)$$

十六、校验码 - 校验码基础知识

码距: 任何一种编码由许多码字构成, 任意两个码字之间最少变化的二进制位数就称为数据校验码的码距。

例: 用4位二进制表示16种状态, 则有16个不同的码字, 此时码距为1, 如0000与0001。

校验码 - 奇偶校验: 只能检错且只能检查1位, 不能纠错

编码方法: 由若干位有效信息 (如一个字节), 再加上一个二进制位 (校验位) 组成校验码。

奇校验: 整个校验码 (有效信息位和校验位) 中 "1" 的个数为奇数。

偶校验: 整个校验码 (有效信息位和校验位) 中 "1" 的个数为偶数。

校验码 - 循环校验码 [CRC] 可检错, 不能纠错

模2除法

校验码 - 海明校验码: 可检错, 可纠错

① 校验位数 $2^r \geq m+r+1$

(冗余位)

数据位

异或运算

第二章

操作系统

628分

内容提要

进程管理

进程的同步与互斥 ★★

前趋图 ★★

信号量与PV操作 ★★

死锁及银行家算法 ★★

存储管理

段页式存储 ★★

页面置换算法 ★

文件管理

绝对路径与相对路径 ★★

索引文件 ★★

位示图 ★★

作业管理

设备管理 - 虚设备与SPooling技术 ★

一、进程管理 — 进程的概念

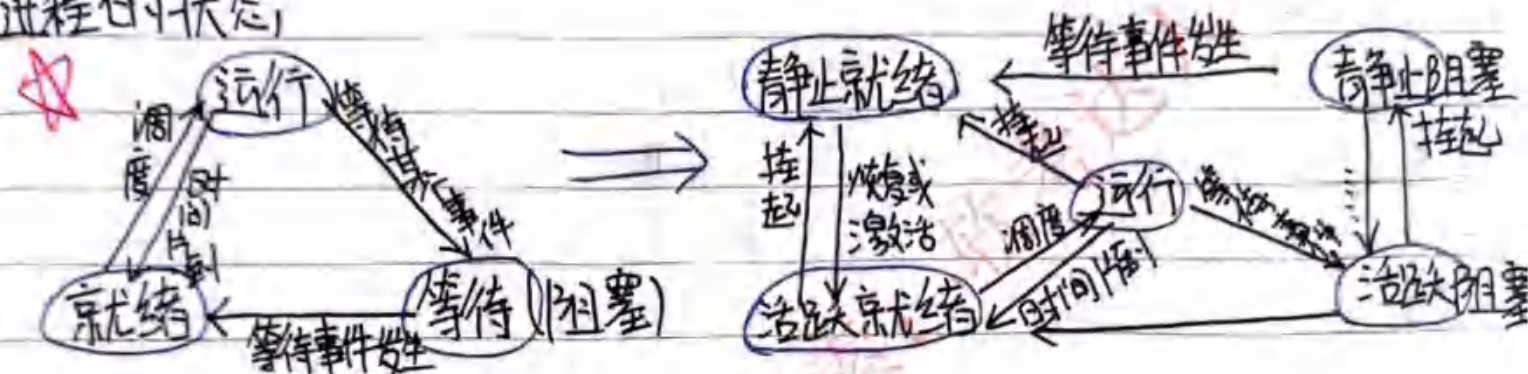
进程是程序在一个数据集上运行的过程，它是系统进行资源分配和调度的一个独立单位。它由程序块、进程控制块 (PCB) 和数据块三部分组成。

进程与程序的区别: **选择题**

进程: 动态的, 是程序的一次执行过程, 没有程序就没有进程。

程序: 静态的, 完成某特定功能的一系列程序语句的集合, 只要不被破坏, 就永远存在。

进程的状态



进程的同步与互斥

直接制约关系

间接制约关系

临界资源

同步: 速度差异, 在应情
况下停下等待

互斥 (争夺临界资源)

PV操作 — **P** 申请资源, **V** 释放资源 (唤醒)

临界资源: 进行处理。

↓ 诸进程间需要互斥方式对其进行共享的资源, 如打印机

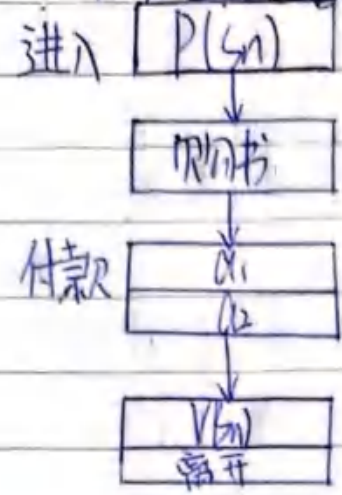
临界区: 每个进程中访问临界资源的那段代码

信号量: 衡量PV操作最重要的变量, 可正可负。

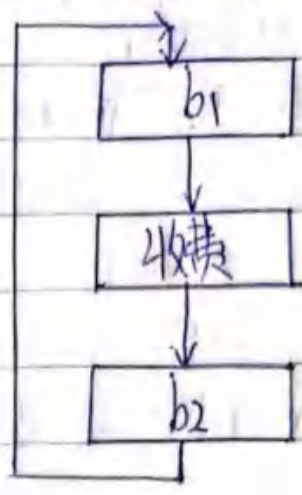
↓ 临界资源总数

例: 某书店有一个收银员, 该书店最多允许 n 个购书者进入, 将收银员和购书者看作不同的进程, 其工作流程如下图所示, 利用PV操作实现该过程, 设置信号量 S_1, S_2 和 S_n 初值分别为 0, 0, n , 则图中 a_1, a_2 应填入 $V(S_1), P(S_2)$, b 和 b_2 应填入 $P(S_1), V(S_2)$ 。

购物者进程 $i=1, 2, \dots, n$

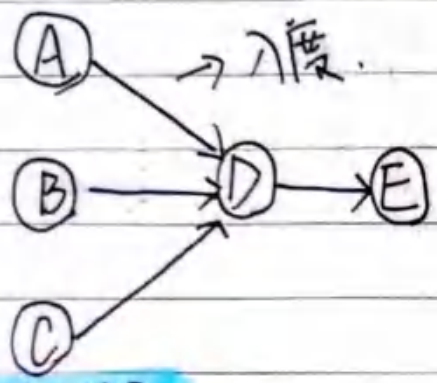


收银员进程



PV操作应用

前趋图



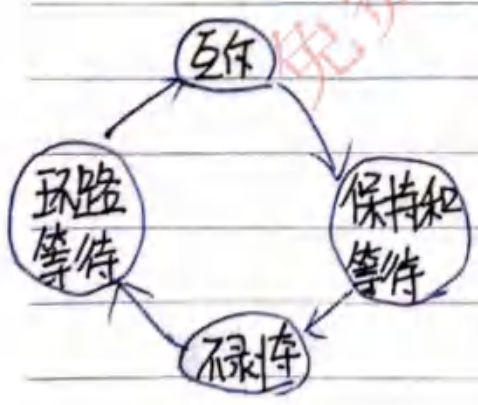
进程D:

寻找入口

$P(Sa)$
 $P(Sb)$
 $P(Sc)$
 $V(Sd)$
 $P(Se);$

死锁问题

进程管理是操作系统的核心，但如果设计不当，就会出现死锁的问题。如果一个进程在等待一件不可能发生的事，则进程就死锁了。而如果一个或多个进程产生死锁，就会造成系统死锁。



死锁的预防

打破四大条件

死锁的避免

有序资源分配法

银行家算法

✓ 银行家算法：分配资源的原则

当一个进程对资源的最大需求量 不超过 系统中的资源时可以接纳该进程。

★进程可以为其请求资源,但请求的总数不能超过最大需求量。

★当系统现有的资源不能满足进程所需资源数时,对进程的请求可以推迟分配,但总能使进程在有限时间里得到资源。

★ 存储管理

页式存储组织

页式存储:将程序与内存均划分为同样大小的块,以块为单位将程序调入内存。

用户程序	页表	物理块号, 又称页帧号	内存
0页	页号	块号	0
1页	0	2	1
2页	1	3	2
...	2	6	3
n页	...	...	4

逻辑地址 = 页号 + 页内地址

物理地址 = 页帧号 + 页内地址

优点: 利用率高, 碎片小, 分配及管理简单。

缺点: 增加了系统开销, 可能产生抖动现象。

段式存储组织

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, 段的长度可以不同。

优点: 多道程序共享内存, 各段程序修改互不影响。

缺点: 内存利用率低, 内存碎片浪费大。

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

优点: 空间浪费小, 存储共享容易, 能动态连接。

缺点: 由于管理软件的增加, 复杂性和开销也随之增加, 需要的软硬件及占用的内存也有所增加, 使执行速度大大下降。

页面置换算法

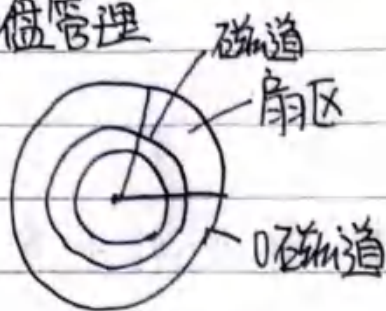
① 最优算法 — 理想

② 随机算法 — 随意

③ 先进先出: 有可能产生“抖动”

④最近最少使用：局部性原理 { 时间局部性
空间局部性

✓ 磁盘管理



每个磁道存储内容相同

圈 ↓ ↑

磁头 → 磁道 → 扇区找内容

存取时间 = 寻道时间 + 等待时间

↓

等待读写扇区转到磁头下方所用的时间

磁头移动到扇区所用的时间

✓ 磁盘调度算法

先到先服务 (FCFS)

最短寻道时间优先 (SSTF)

扫描算法 (SCAN) — 电梯算法

循环扫描 (CSCAN) 算法

读取磁盘数据时间计算

分三部分：①找磁道的时间

②找块(扇区)的时间，即旋转延迟时间

③传输时间

作业管理

1. 作业状态与作业管理

2. 作业调度算法

先来先服务法

时间片轮转法

短作业优先法

最高优先权优先法

高响应比优先法

响应比 = $\frac{\text{作业等待时间}}{\text{执行时间}}$

文件管理



索引
结点
(BT)

0
1
2
3
:
9
10
11
12

0~9 → 物理块

直接索引

10 → 一级间接索引 → 分成 1KB 每块用 4KB $\frac{1K}{4} = 256 \rightarrow 1K$

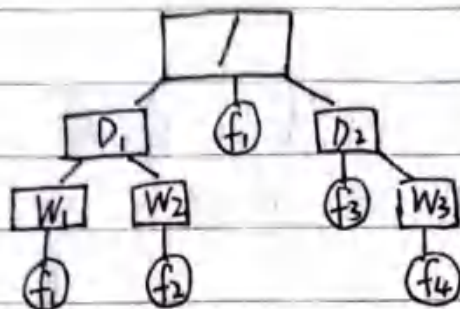
11 → 二级间接索引 64M

256K

12 → 三级间接索引 16G

编号从 0 开始 0~10239

树型目录结构



★ 考查: 相对路径 / 绝对路径

假设当前文件在 D1 下, 找 f2

W2/f2 /D1/W2/f2

空闲存储空间的管理

位示图法: 定票系统 (形象、生动)

设备管理

数据传输控制方式

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

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。

DMA 方式: DMA 方式是为了在主机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。与 CPU 无关

通道方式

I/O 处理机

设备与 SPOOLING 技术

SPOOLING 是关于慢速字符设备如何与计算机主机交换信息的一种技术, 通常称为“假脱机技术”, 通过磁盘实现。

第三章

数据库系统

6~8分

课程内容提要

- 数据库模式(★★)
- ER模型(★★★★)
- 关系代数(★★★)
- 规范化理论(★★★★★)
- SQL语言(★★★)
- 并发控制(★★)
- 数据库完整性约束(★)

采用三级结构/两级映像的数据库体系结构，如果对数据库的一张表创建聚簇索引，改变的是数据库的内模式。

2. DISTINCT: 避免重复统计导致错误结果。

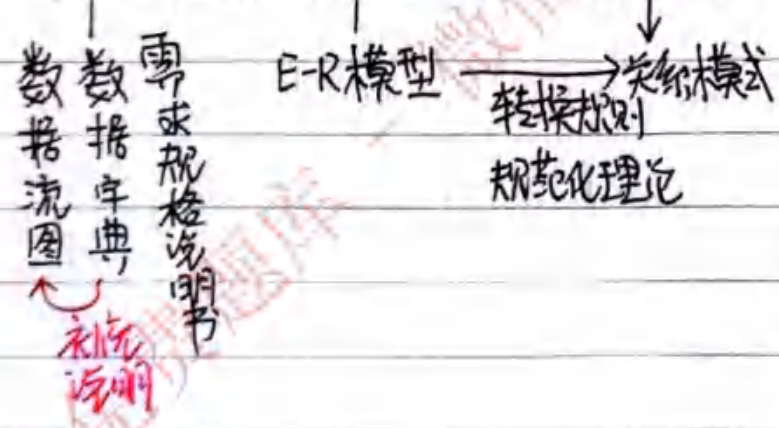
一、三级模式 - 两层映射

模式: 外模式(视图级/用户模式)、模式(逻辑模式/表级)、内模式(文件级)有储

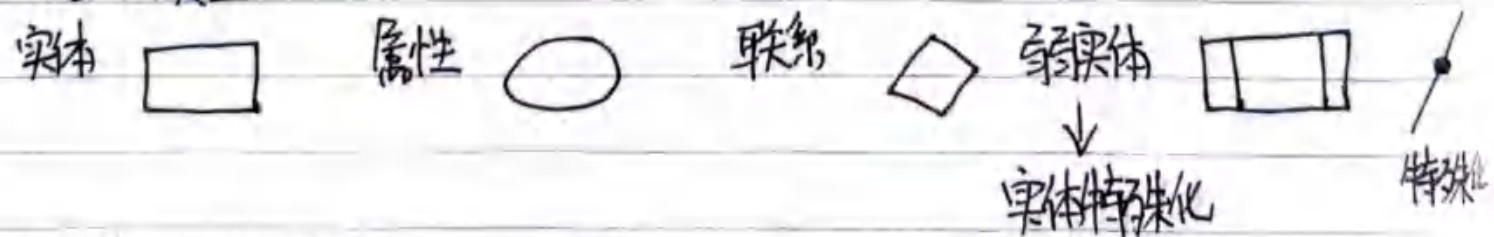
映射: 外模式与模式映射, 模式与内模式映射 $\Rightarrow$ 使数据库具有独立性
 数据独立性
 物理独立性

二、数据库设计过程

四阶段: 需求分析、概念结构设计、逻辑结构设计、物理设计



三、E-R模型



1:1 联系, 1:n 联系, m:n 联系

一个实体型转换为一个关系模式

联系转关系模式: 1:1 联系: 可将联系合并到任意一端的实体关系模式中。

1:n 联系: 可将联系合并到 n 端实体关系模式中。

m:n联系: 联系必须单独转换为关系模式。

三个以上实体间的一个多元联系。

四关系代数

并、交、差、笛卡尔积, 投影、选择、联接、自然连接

↓ ↓ ↓

列 行 去掉相同属性

五规范化理论

关系模式R (学生姓名、选修课程名、任课教师名、任课教师地址)

数据冗余

修改异常

插入异常

删除异常

部分函数依赖

传递函数依赖

→ 判断几范式

候选键: 唯一标识元组, 且无冗余, 可以是一个或多个 → 主属性

主键: 任意一个, 只能有一个, 从候选键中选出。

外键: 其它关系的主键

图解法求候选键

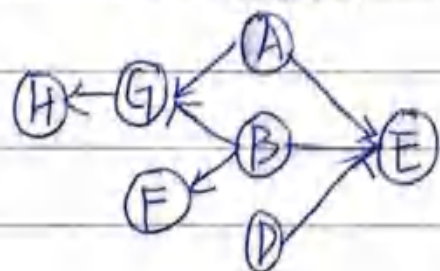
1. 将关系的函数依赖关系, 用“有向图”的方式表示

2. 找出入度为0的属性, 尝试遍历有向图, 若能正常遍历即为关系模式候选键。

3. 若入度为0的属性集不能遍历图中所有结点, 则需要尝试性的将一些中间结点并入入度为0的属性集中, 直到该集合能遍历所有结点, 集合为候选键。

↓ 有进有出

例1. 关系模式P(A, B, C, D, E, F, G, H, I, J) 满足下列函数依赖: $FD \Rightarrow (ABD \rightarrow E, AB \rightarrow G, B \rightarrow F, C \rightarrow J, CJ \rightarrow I, G \rightarrow H)$, 求候选码?



A, B, C, D 组合

① A, B, D, C 均为入度为0

② A, B, D 走完上面, C 走完下面



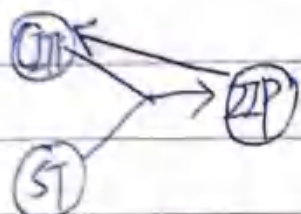
例2. 关系(A, B, C)满足函数依赖: $F(B \rightarrow C, B \rightarrow A, A \rightarrow BC)$, 关系R的候选关键字为 A和B



主属性与非主属性

定义: 组成候选码的属性就是主属性, 其它的的就是非主属性

例. 关系模式 $SZ(CITY, ST, ZIP)$, 其属性组上的函数依赖集为: $F = \{(CITY, ST) \rightarrow ZIP, ZIP \rightarrow CITY\}$



$(ST, CITY) \rightarrow ST, CITY, ZIP$ 为主属性
 $(ST, ZIP) \rightarrow ST, CITY, ZIP$ 为主属性

范式 $1NF > 2NF > 3NF > BCNF$

第一范式(1NF): 所有属性均为原子值, 即每个属性不可再分。

第二范式(2NF): 去掉部分函数依赖

第三范式(3NF): 去掉传递依赖

BC范式(BCNF): 每个依赖的决定因素必定包含某个候选码
逐步优化, 以解决 $\rightarrow$ 消除主属性对候选键的部分和传递依赖
插入异常, 删除异常, 数据冗余

模式分解

保持函数依赖分解

无损分解与有损分解

有损: 不能还原, 无损: 可以还原

无损联接分解: 指将一个关系模式分解成若干关系模式后, 通过自然连接和投影, 等运算仍能还原到原来的关系模式。

无损分解(表格法)

无损分解(公式法)

定理: 如果R的分解为 $\rho = \{R_1, R_2\}$, F 为R所满足的函数依赖集合, 分解 ρ 具有

无损联接性的充分必要条件是:

$$R_1 \cap R_2 \rightarrow (R_1 - R_2) \text{ 或 } R_1 \cap R_2 \rightarrow (R_2 - R_1)$$

例: 设 $R=ABC$, $F=\{A \rightarrow B\}$, 则分解 $P_1 = \{R_1(AB), R_2(AC)\}$ 与分解 $P_2 = \{R_1(AB), R_2(BC)\}$ 是否都为无损分解?

↓

是

$$R_1 \cap R_2 = B$$

$$R_1 - R_2 = A$$

$$R_2 - R_1 = C$$

↓
 $R \cap R = A$

$$R_1 - R_2 = B$$

$$R_2 - R_1 = C$$

7. SQL语言—建表 (上午)

1. CREATE TABLE 建表
2. PRIMARY KEY 主键
3. NOT NULL 不为空
4. UNIQUE 唯一

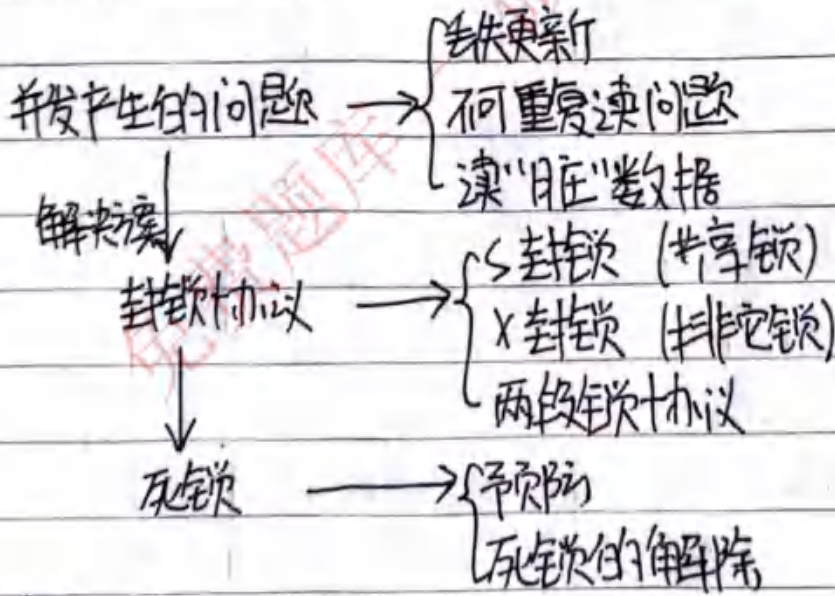
权限控制语句

赋予用户权限命令的简单格式:

GRANT 权限 ON 数据库对象 TO 用户
WITH GRANT OPTION

1. 并发控制

事务: 原子性、一致性、隔离性、持续性



9. 数据库完整性约束

- 实体完整性约束 → 主键 { 不为空, 唯一标识
- 参照完整性约束 → 外键 { 有 → 正确参照, 空
- 用户自定义完整性约束

触发器

第四章 计算机网络与信息交互

开放系统互连参考模型 (★★)

TCP/IP协议族 (★★★)

IP地址与子网划分 (★★★★)

网络规划与设计 (★)

3G与4G标准 (★)

HTML语言 (★)

一、OSI/RM七层模型

层次	名称	主要功能	主要设备与协议
7	应用层	实现具体的应用功能	POP3, FTP, HTTP
6	表示层	数据的格式与表达, 加密压缩	Telnet, SMTP
5	会话层	建立管理和终止会话	DHCP, TFTP, SNMP, DNS
✓ 4	传输层	端到端点的连接	TCP, UDP (不可靠) → 地址连接
✓ 3	网络层	分片和传输和路由选择	三层交换机, 路由器, ARP, RARP
✓ 2	数据链路层	传送以帧为单位的信	网桥, 交换机 (多端口网卡)
✓ 1	物理层	二进制传输, bit	中继器, 集线器 (多端口网卡)

保证信息不丢失

二、TCP/IP协议族

TCP/IP模型	OSI七层模型
应用层	应用层
	表示层
	会话层
传输层	传输层
网络层	网络层
网络接口层	数据链路/物理层

POP3 110	FTP 20/21	HTTP 80	Telnet 23	SMTP 25	NFS	DHCP 67	TFTP 69	SNMP 161	DNS 53
TCP					UDP				
IP ICMP IGMP					ARP RARP				
CSMA/CD					Token Ring				

POP3: 110端口, 由邮件收取, 有网就收。

SMTP: 25端口, 由邮件发送

载波监听/多路访问

HTTP	SNMP
TCP	UDP
IP	



超文本传输协议, 即使用SSL加密传输的HTTP!

FTP: 20数据端口/21控制端口, 文件传输协议 (基于TCP)

HTTP: 80端口, 超文本传输协议, 网页传输, 多为Web服务器使用, 但该协议不安全

DHCP: 67端口, IP地址自动分配

SNMP: 161端口, 简单网络管理协议, 用于网络管理, 该协议的报封装在

UDP: 53端口, 域名解析协议, 记录域名与IP的映射关系, UDP中传送。

TCP: 可靠的传输层协议

UDP: 不可靠的传输层协议

ICMP: 因特网控制协议, PING命令来自该协议

IGMP: 组播协议

ARP: 地址解析协议, IP地址转换为MAC地址

RARP: 反向地址解析协议, MAC地址转IP地址。

三. IP地址 (32位)

网络号 + 主机号. 有效: $2^{24} - 2$ → 全0/全1

解析协议

IP, ICMP, IGMP

网卡, PPTP, LTP, SUP, PPP

路由器

第 0	15	24	23	31位
0	127	128	191	
A类	网络号		主机号	
B类	10	128	191	
B类	网络号	24	主机号	
C类	110	192	255	
C类	网络号	28	主机号	
D类	1110			组播地址
E类	11110			保留为今后使用

四. 子网划分

1. 子网掩码: 网络地址全为1, 主机地址全为0

2. 将一个网络划分成多个子网 (取部分主机号当子网号)

3. 将多个网络合并成一个大的网络。

例: 将B类IP地址168.195.0.0划分成21个子网, 子网掩码为多少?

$2^4 < 21 < 2^5$ → 去主机号中借五位。

十进制

168.195.0.0

二进制

1010 1000

1100 0011

0000 0000

0000 0000

1111 1111

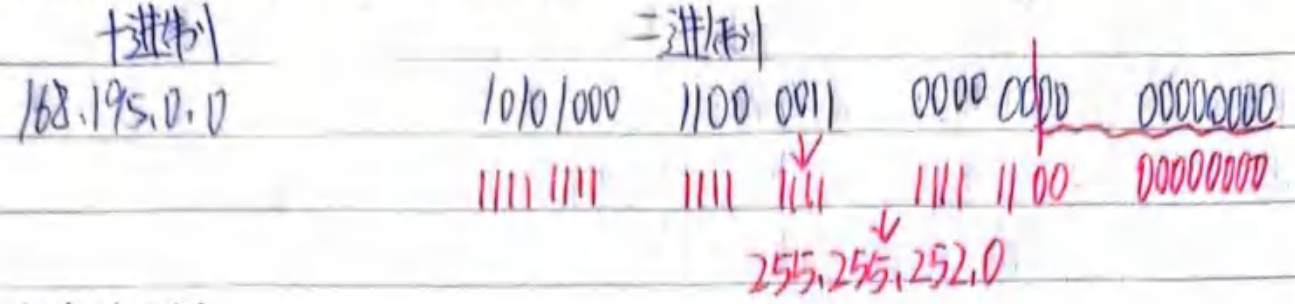
1111 1111

1111 0000

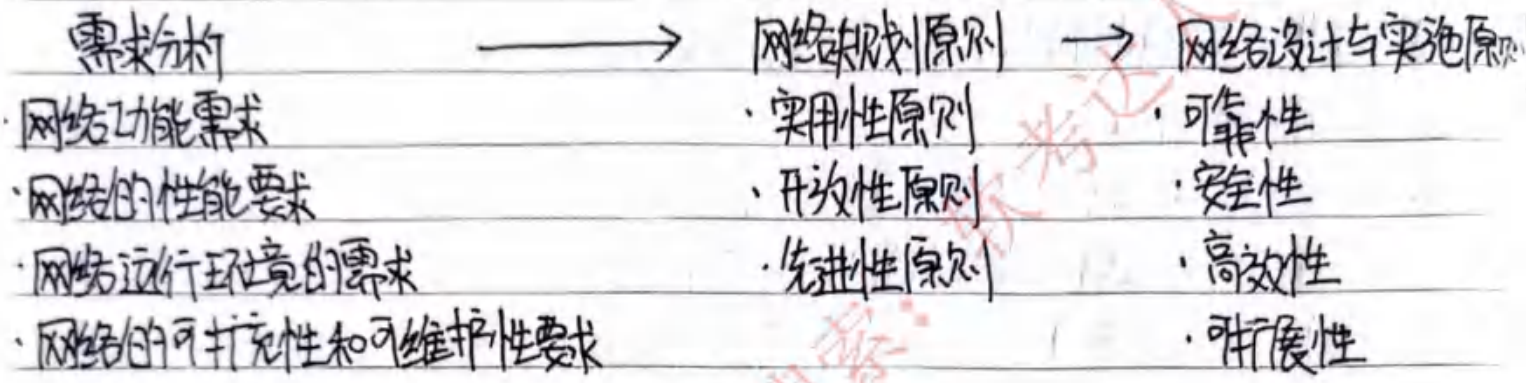
0000 0000

$2^7 < 100 < 2^8$

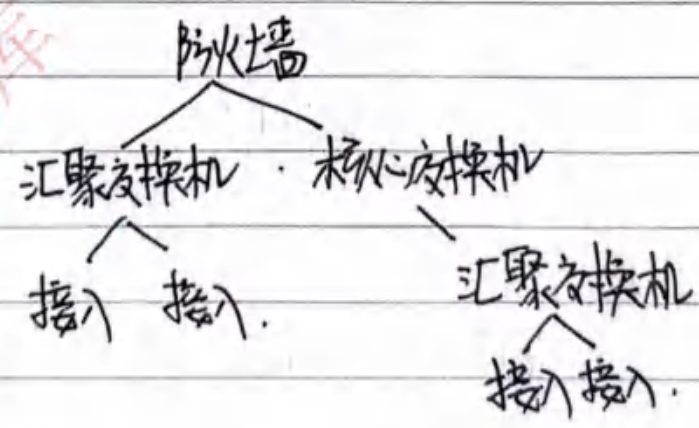
例2. 将B类IP地址168.195.0.0划分成若干子网, 每个子网内有主机100台, 子网掩码为1/10



五网络规划与设计



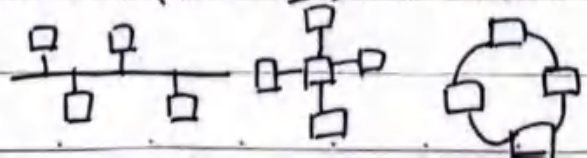
★ 核心层 汇聚层 接入层 → 层次化网络设计



二、计算机网络分类

按分布范围分: 局域网(LAN)、城域网(MAN)、广域网(WAN)、因特网

按拓扑结构分: 总线型、星型、环型



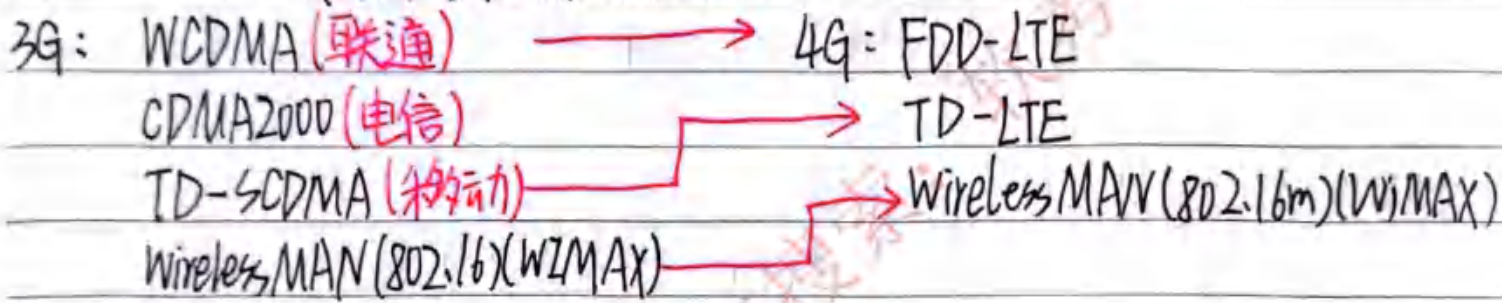


- 无线局域网 (WLAN, 802.11 Wi-Fi)
- 无线城域网 (WMAN, 802.16, WiMax)
- 无线广域网 (WWAN, 3G/4G)
- 无线个人网 (WPAN, 802.15, Bluetooth)

网络接入技术

- 有线接入:
- 公用交换电话网络 (PSTN)
 - 数字数据网 (DDN)
 - 综合业务数字网 (ISDN)
 - 非对称数字用户线路 (ADSL)
 - 网络传输光纤技术 (HFC)

- 无线接入:
- IEEE 802.11 (WiFi)
 - IEEE 802.15 (蓝牙 Bluetooth)
 - 红外 (IrDA)
 - WAPI



11. HTML

- | | |
|-----------------------|-----------------------|
| <a> 定锚 | 定义文字的字体、大小和颜色 |
| 加粗 | <form> 定义供用户输入的HTML表单 |
| <body> 主体 | <frame> 定义框架集的窗口或框架 |
| <button> 按钮 | <h1> 标题 |
| <center> 居中 | <hr> 水平线 |
| <col> 定义表格中一列或多个的列属性值 | |
| <html> 文档 | 定义强调文本 |
| 图像 | <table> 表格 |
| <p> 段落 | <td> 定义表格中的单元 |
| <script> 定义客户端脚本 | <tr> 行 |
| | <title> 标题 |

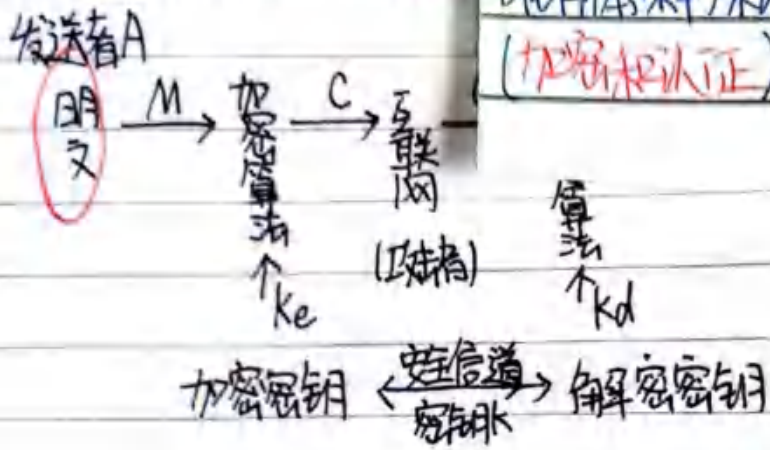
课程内容提要

- 对称加密与非对称加密 (☆☆)
- 信息摘要与数字签名 (☆☆)

数字签名的作用

- ① 接收者可验证消息来源的真实性
- ② 发送者无法否认发送过该消息
- ③ 接收者无法伪造或篡改消息
- X 不可验证接收者合法性
- 公钥体系中, 私钥用于(解密和签名), 公钥用于(加密和认证)。

- 数字证书(★★)
- 网络安全协议(★★)
- 防火墙技术与网络攻击(★★)
- 计算机病毒与木马(★★)
- 九对称加密技术 — 使用



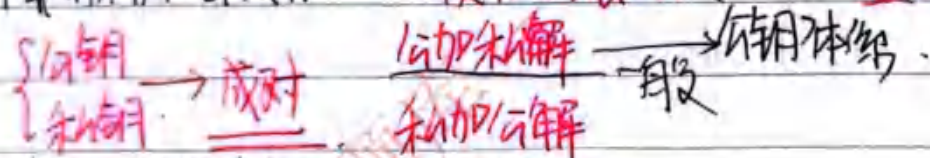
对称加密: $k_e = k_d$

特点: ① 加密强度不高, 但效率高 ② 密钥分发困难

常见对称密钥加密算法: DES, 3DES(三重DES), RC-5, IDEA算法

十非对称加密技术 — 使用不同密钥

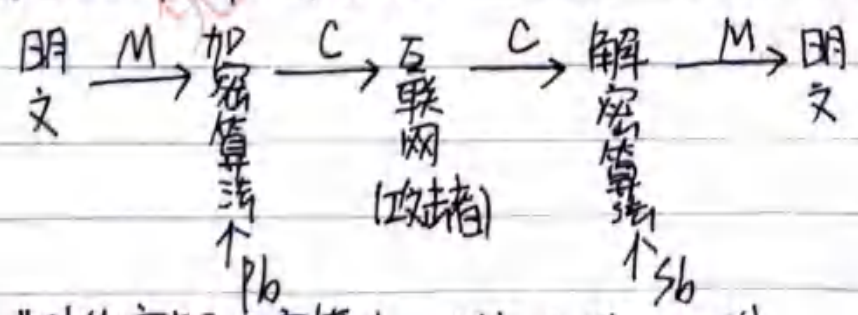
适用于大量的明文加密



特点: 加密速度慢, 但强度高

发送者A: 公钥Pa, 私钥Sa

接收者B: 公钥Pb, 私钥Sb



常见非对称密钥加密算法: RSA, ECC, DSA

一、数字签名

使用发送者的私钥Sa进行签名, 发送者的公钥Pa验证。

二、消息签名摘要

→ 不可逆

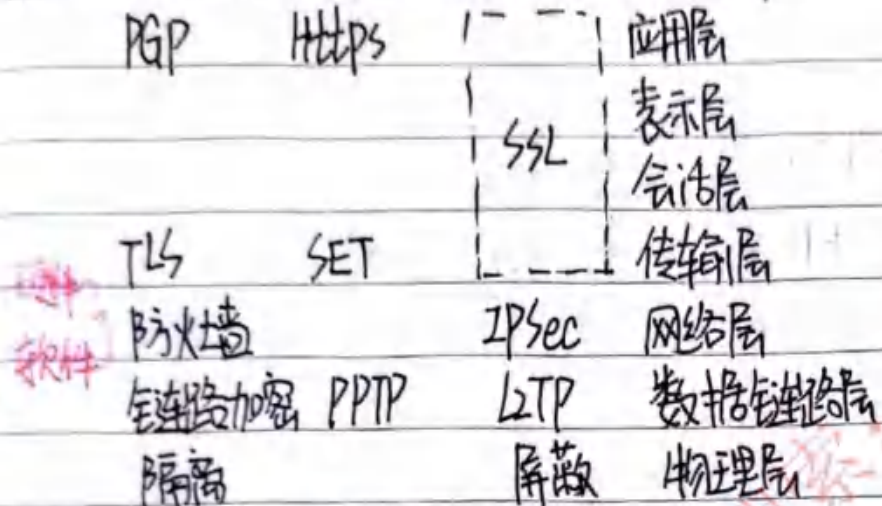
1. 数字摘要: 由单向散列函数加密成固定长度的散列值

2. 常用的消息摘要算法有 MD5, SHA等, 散列值分别为128和160位, 由于SHA通常采用的密钥长度较长, 因此安全性高于MD5.

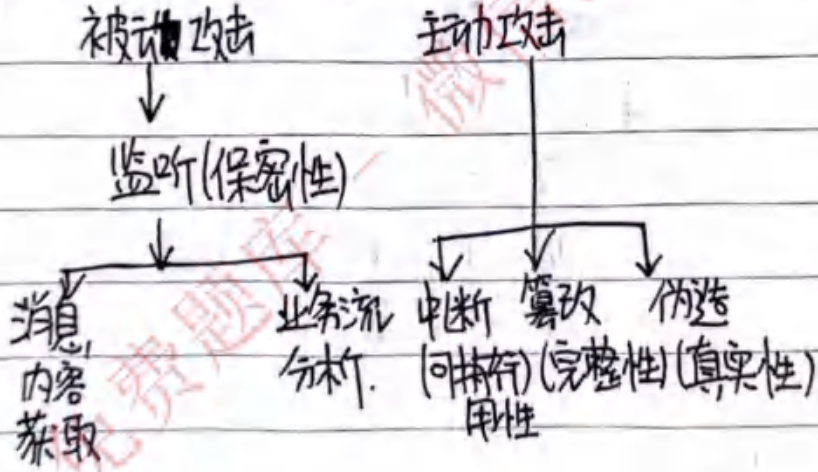
十三. PKI公钥体系

管理密钥, 注册, 更新, 注销. CA中心

十四. 网络安全 — 各个网络层次的安全保障

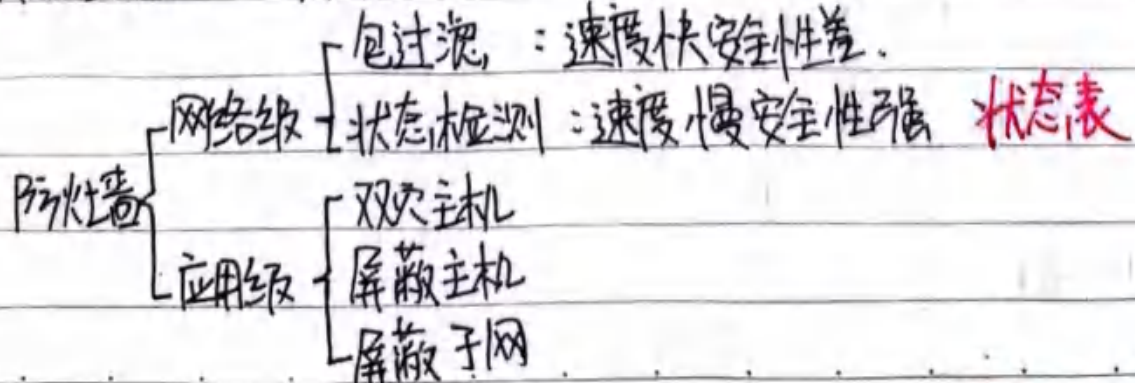


十五. 网络安全 — 主动攻击与被动攻击



十七. 网络安全 — DOS(拒绝服务)与DDOS.

十八. 网络安全 — 防火墙



信息安全防范体系

层次划分:

1. 物理环境的安全性
2. 操作系统的安全性
3. 网络的安全性
4. 应用的安全性
5. 管理的安全性

计算机病毒与木马

→ 直接破坏电脑

病毒: 编制或者在计算机程序中插入的破坏计算机功能或者破坏数据, 影响计算机使用并且能够自我复制的一组计算机指令或者程序代码。

木马: 计算机木马是一种后门程序, 常被黑客用作控制计算机的工具。

窃取信息

第五章 系统开发基础

课程内容提要

✓ 开发模型 (★★★★)

软件开发方法 (★)

需求分析 (★)

软件设计 (★★)

软件测试与维护

✓ 白盒测试用例 (★★★★)

· McCabe 复杂度计算 (★★)

✓ 软件维护类型 (★★★★)

软件质量保证 (★)

软件过程改进 (★★)

项目管理基础

✓ Gant 图与 PERT 图 (★★★★)

· 风险管理 (★★)

一、软件开发模型

瀑布模型



解决需求明确但项目
(难以适应变化的需求)

软件计划 { 可行性分析报告
软件计划书

需求分析 → 挖掘用户需求

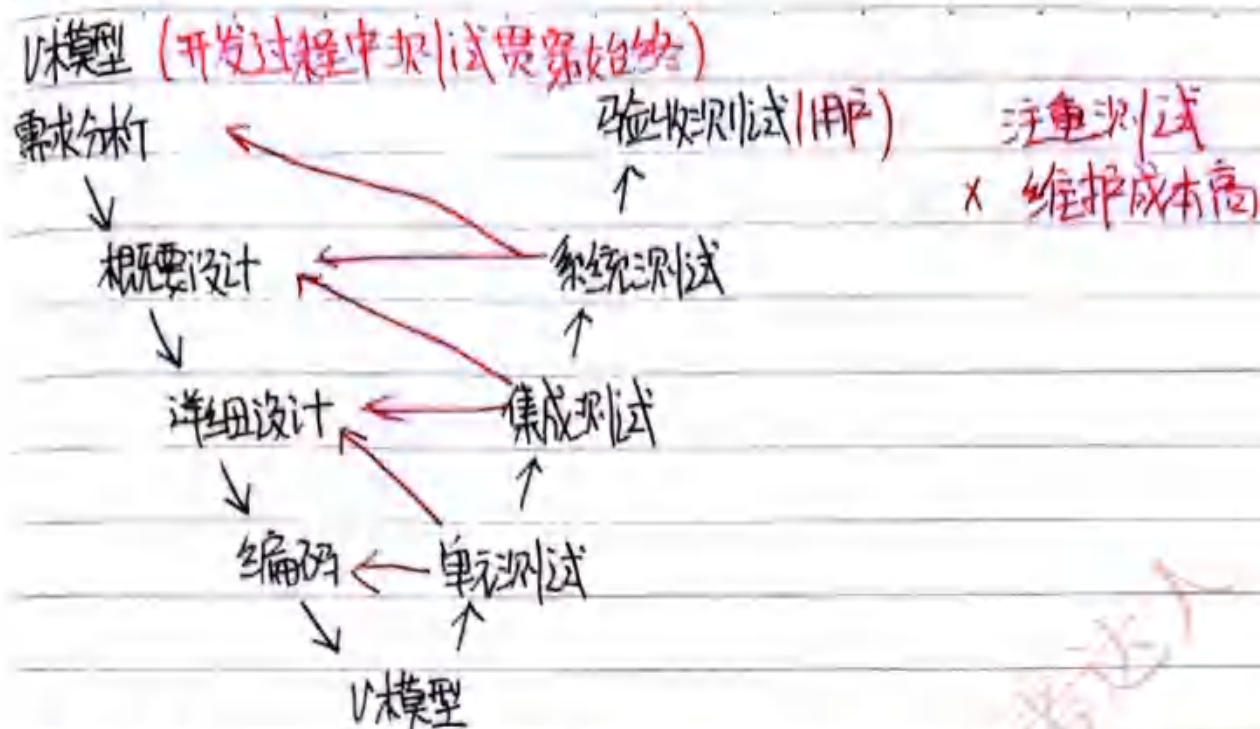
软件设计 → 概要设计, 详细设计

程序编码 → 产品

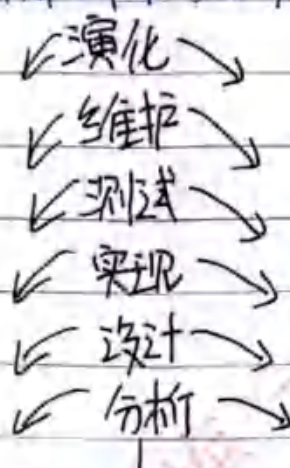
软件测试 → 用例, 报告

运行维护

不阶段
开发
需求
设计
编码
测试
维护



喷泉模型: 以用户需求为动力, 以对象为驱动, 适用于面向对象的开发。



面向对象
每阶段无明显届线

统一过程 (UP) 定义初始阶段、精化阶段、构建阶段、移交阶段和产生阶段, 每个阶段达到某里程碑时结束。初始阶段: 生命周期目标, 精化阶段 (生命周期架构), 构建阶段 (初始运行功能), 移交阶段 (产品发布)。

增量模型: 是一种能够快速构造可执行产品的好办法。可以快速构造初始产品。

原型化模型 (动态定义需求)

用户无法表述清楚, 沟通

演化模型 (又称增量模型)

开发前的小模型, 适用于

螺旋模型 → 结合瀑布模型和原型化模型

综合瀑布, 原型化模型的特点, 适合开发高风险项目。成品提高 维护解

统一过程 RUP 风险驱动, 可降低过多测试或测试不足带来的风险, 包含

用例驱动, 以架构为中心, 迭代和增量, 分四个阶段: 初始、细化、交付、构建

初始: 确定项目的范围和边界, 识别系统的关键用例, 展示系统的候选

架构, 估计项目费用和时间, 评估项目风险。



ITIL 课程 → 开发模型

敏捷开发是一种以人为本、迭代、循序渐进的开发方法。常见的敏捷开发方法有极限编程法、水晶法、并列表、自适应软件开发法等。

水晶法强调经常交付，认为每一种不同的项目都需要一套不同的策略、约定和流程。

并列表的核心是迭代、增量交付，按照30天进行迭代开发交付可实际运行的软件。

自适应软件开发核心是一个非线性的、迭代阶段、预测、合作、敏捷编程、XP、并列表、敏捷开发等。

4大价值观 → 5大原则

沟通	快速反馈	计划游戏	结对编程
简单	简单性假设	小型发布(早期)	集体代码所有制
反馈	逐步修改	隐喻	持续集成
勇气	提倡更改	简单设计	每周工作40小时
	优质工作	测试驱动	现场客户
		重构	编码标准

关注需求分析和架构演进

确认新系统、培训、调整产品

自动化测试

适应性计划调整

结对编程

测试驱动开发

持续集成

重构

扩展性
可维护性

1. 敏捷开发方法

(1) XP (极限编程): 对费用控制严格的公司中使用。

(2) Cockburn 水晶系列: 用最少的纪律约束而能获得成功。

(3) 开放式源码: 程序开发人员在地理上分布很广。联机开发, 协同开发

(4) SCRUM (并列表): 明确定义了的可重复的方法过程, 为明确定义了的可重复的人员所用, 来解决明确定义了的可重复的问题。

(5) ASD 方法: 核心, 预测, 合作与学习

二. 软件开发方法

1. 结构化方法 面向数据流, 采用数据流图。

· 自顶向下, 逐步分解 接口设计的任务主要是描述软件与外部环境之间的交互关系, 软件内模块之间的调用关系。

2. 原型法

3. 面向对象方法 喷泉模型



1. 架构和算法的设计。
2. 体系结构设计: 定义软件系统各主要部件之间的交互
3. 接口设计(人机界面设计): 软件内部、软件和时间以及软件和人之间如何通信。
4. 数据设计: 基于E-R图确定软件涉及的文件和系统数据结构及数据库的表结构。

应用的工具

· 数据流图 (DFD)

· 数据字典 (DD) ^{补充说明}

· 判定表 (描述地下情况)

· 判定树 (决策树)

· 基本条件、条件

· 基本动作、动作下

需求分析有(评审)每阶段

↓ 多条要求

四. 软件设计 怎么做

独立性, 高内聚, 低耦合, 保证模块独立性

开发过程最重要环节

- 应用的工具
- IPO图 — 输入处理输出图
 - PDL — 程序描述语言 (伪代码)
 - PAD — 问题分析图
 - 程序流程图
 - N/S盒图

选择软件开发工具时, 应

考虑功能、易用性、

稳健性、硬件要求、

服务和技能。

内聚类型	描述
功能内聚	完成一个单一功能, 各部分协同工作, 缺一不可
顺序内聚	处理元素相关, 而且必须顺序执行
通信..	所有处理元素集中在一个数据结构或区域上
过程..	处理元素相关, 而且必须按特定的次序执行
时间..	所包含的任务必须在同一时间间隔内执行
逻辑..	完成逻辑上相关的一组任务
偶然..	完成一组没有关系或松散关系的任务

耦合类型	描述
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通过主模块控制
数据耦合	一组模块借助参数表传递简单数据类型数据 值传递
标记耦合	一组模块通过参数表传递记录信息, (数据结构) 结构体
控制 ..	模块之间传递的信息, 中包含控制模块内部逻辑的信息
外部 ..	一组模块都访问同一个全局简单变量 而不是通过参数或通过全局变量
公共 ..	多个模块都访问一个公共数据环境 (全局数据结构, 共享通信)
内容 ..	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至另一个模块的内容; 两个模块有一部分程序代码重叠 一个模块有另一个入口

五软件测试

题:

1. 已知模块 A 给模块 B 传递数据结构 X, 则这两个模块的耦合类型为 (标记耦合)。

2. 白盒测试技术中, 路径覆盖法往往能比语句覆盖法发现更多的错误。

(运行) 动态测试
静态测试

条件覆盖 ..

路径 .. (覆盖所有可能的路径)

等价类划分

确定有效和无效等价类

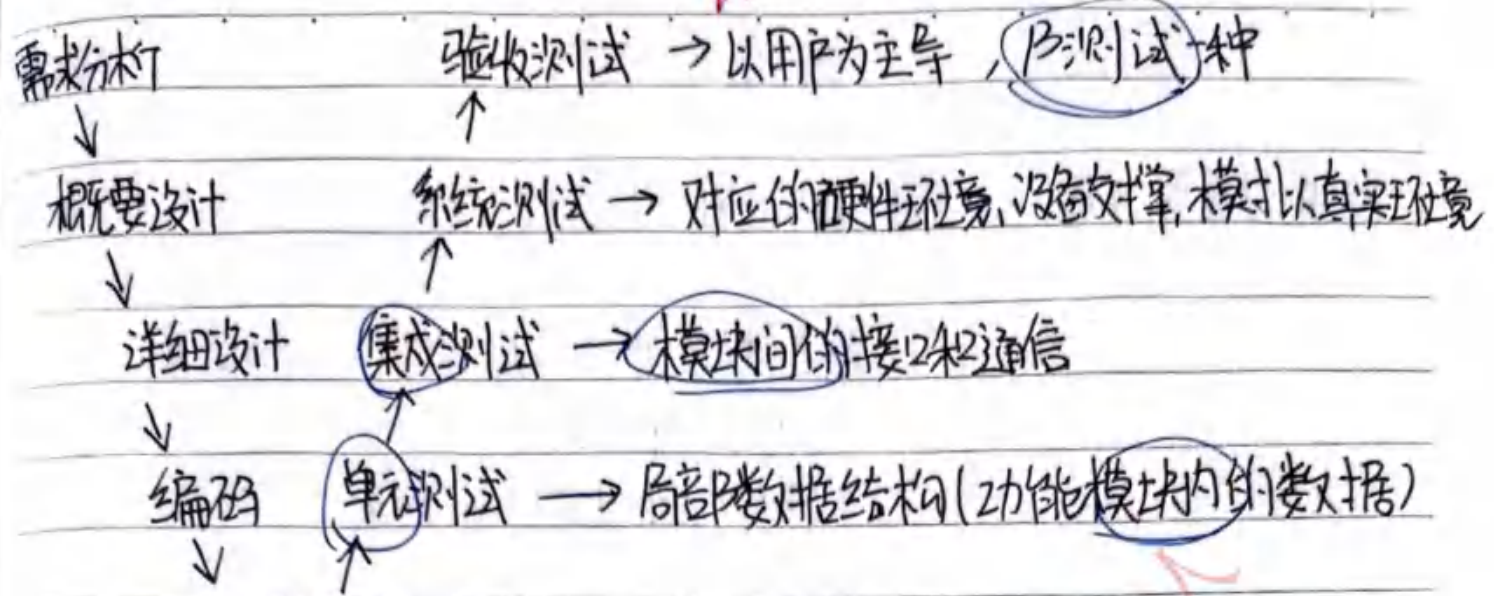
设计用例尽可能多的覆盖有效类

设计用例覆盖一个无效类

边界值分析

处理边界情况时最容易出错

选取的测试数据应该恰好等于、稍小于或稍大于边界值



V模型

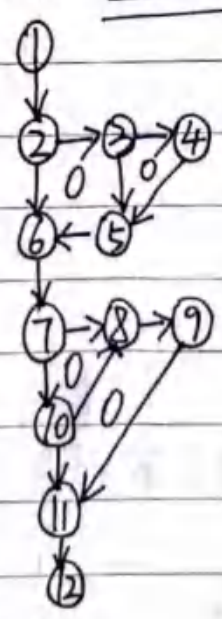
- 回归测试: 原本测试出问题维修后 二次测试
- 负载测试 → 性能测试 → 一定条件下所能承受的 最大负载
- 压力测试 → 达到时效条件下所能承受的压力

五软件测试 — McCabe 复杂度 (环路)

计算有向图G的环路复杂度公式: $V(G) = m - n + 2$. 最少需要V(G)个测试用例

有向图G的环路数 = 边数 - 结点数 + 1

$V(G) = \text{封闭区域数} + 1$



六 软件维护

可维护性因素决定
软件维护类型

可理解性 文档

可测试性

可修改性 适应可扩展

改正性维护 (25%) 原系统有问题

适应性维护 (20%) 环境

预防性维护 (15%) 将来

完善性维护 (50%) 扩充功能

七软件工程国家标准 — 软件文档管理指南 — 按阅读对象分类

文档分类: 开发文档、产品文档、管理文档

开发文档: 可行性研究和项目任务书

需求规格说明

功能规格说明

设计规格说明 (包括程序和数据规格说明)

开发计划

软件集成和测试计划

质量保证计划、标准、进度

安全和测试信息

管理文档: 开发过程的每个阶段的进度

软件变更情况的记录

相对于开发的判定记录

职责定义

产品文档: (用户)

培训手册

参考手册和用户指南

软件支持手册

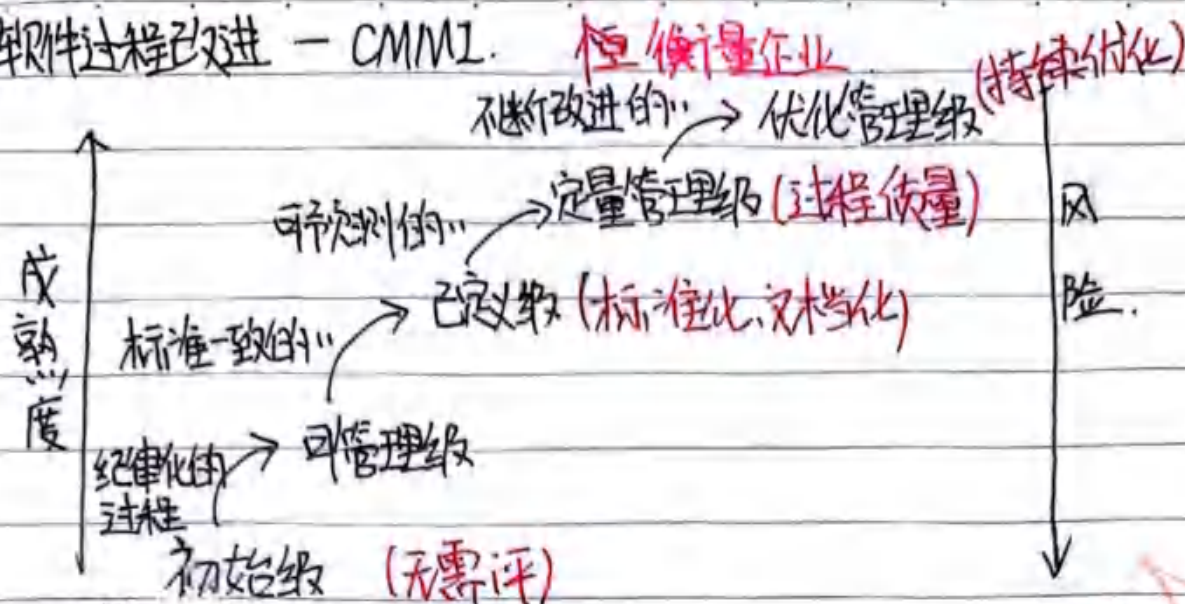
产品手册和信息

2. 软件质量保证: ISO/IEC 9126 软件质量模型

外部和内部质量

功能性	可靠性	易用性	效率	维护性	可移植性
适应性	成熟性	易理解性	时间特性	易析性	适应性
准确性	容错性	易学性	资源利用性	易改变性	易安装性
互操作性	易恢复性	易操作性	效率/保真性	稳定性	共存性
安全保密性	可靠性的依从性	可测试性	易测试性	易测试性	易替换性
功能性的依从性	...	...	...	...	...

3. 软件过程改进 - CMMI



1. 项目管理

十大知识领域

- ① 范围管理 ② 时间管理 ③ 成本管理 ④ 质量管理 ⑤ 人力资源管理
⑥ 沟通管理 ⑦ 风险管理 ⑧ 采购管理 ⑨ 整体管理 ⑩ 项目组织管理

九时间管理

1. 关键路径法 → 工期最长路径 (无活动时间)

关键路径法是在制订进度计划时使用的一种进度网络分析技术, 理论上的最早开始与完成日期, 最迟开始与完成日期, 确定任何资源限制。

总时差 (松弛时间): 在不延误总工期的前提下, 该活动的机动时间。

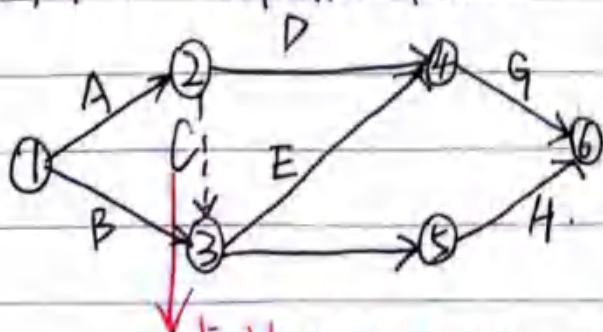
活动的总时差 = 该活动最迟完成时间 - 最早完成时间

= 该活动最迟开始时间 - 最早开始时间

2. 前导图法 (单代号网络图, PDM)

找出关键路径, 总工期, 总时差。

十活动排序 - 箭线图法 ADM



虚箭头 (虚工作), 不占时间, 不工作, 表示 2, 3 之间有联系。

十一-项目管理

风险曝光度：计算方法是风险出现的概率乘以风险可能造成的损失。

例：假设正在开发的软件项目可能有在一个未被发现的错误，而这个错误出现的概率是0.5%，给公司造成的损失将是1000000元，那么这个错误的风险曝光度为 $1000000 \times 0.5\% = 5000$ 元。

* 第六章 面向对象技术

上
12分

下
两到大题30分

课程内容提要

面向对象的基本概念, (☆☆☆☆)

↓
万物皆对象

UML

· 用例图 (☆☆☆☆) 理解系统做什么 即PM为

· 类图 (☆☆☆☆) 一个类中成员变量和成员函数有时可以分离被称为 (属性

· 顺序图 (☆☆) 和方法)。

· 通信图 (☆☆)

· 状态图 (☆☆)

· 活动图 (☆☆)

设计模式 (☆☆☆☆)

一、面向对象的基本概念 (选择)

1. 对象：属性(数据) + 方法(操作) + 对象ID (唯一标识)

2. 类 (实体类/控制类/边界类) 具有相同属性的集合。

3. 继承与泛化：复用机制

4. 封装：隐藏对象的属性和实现细节 仅对外公开接口

5. 多态：不同对象收到同样的消息，产生不同的结局

6. 接口：一种特殊的类，他只有方法定义没有

7. 重载：一个类可以有多个同名参数类型不同的方法

8. 模式类

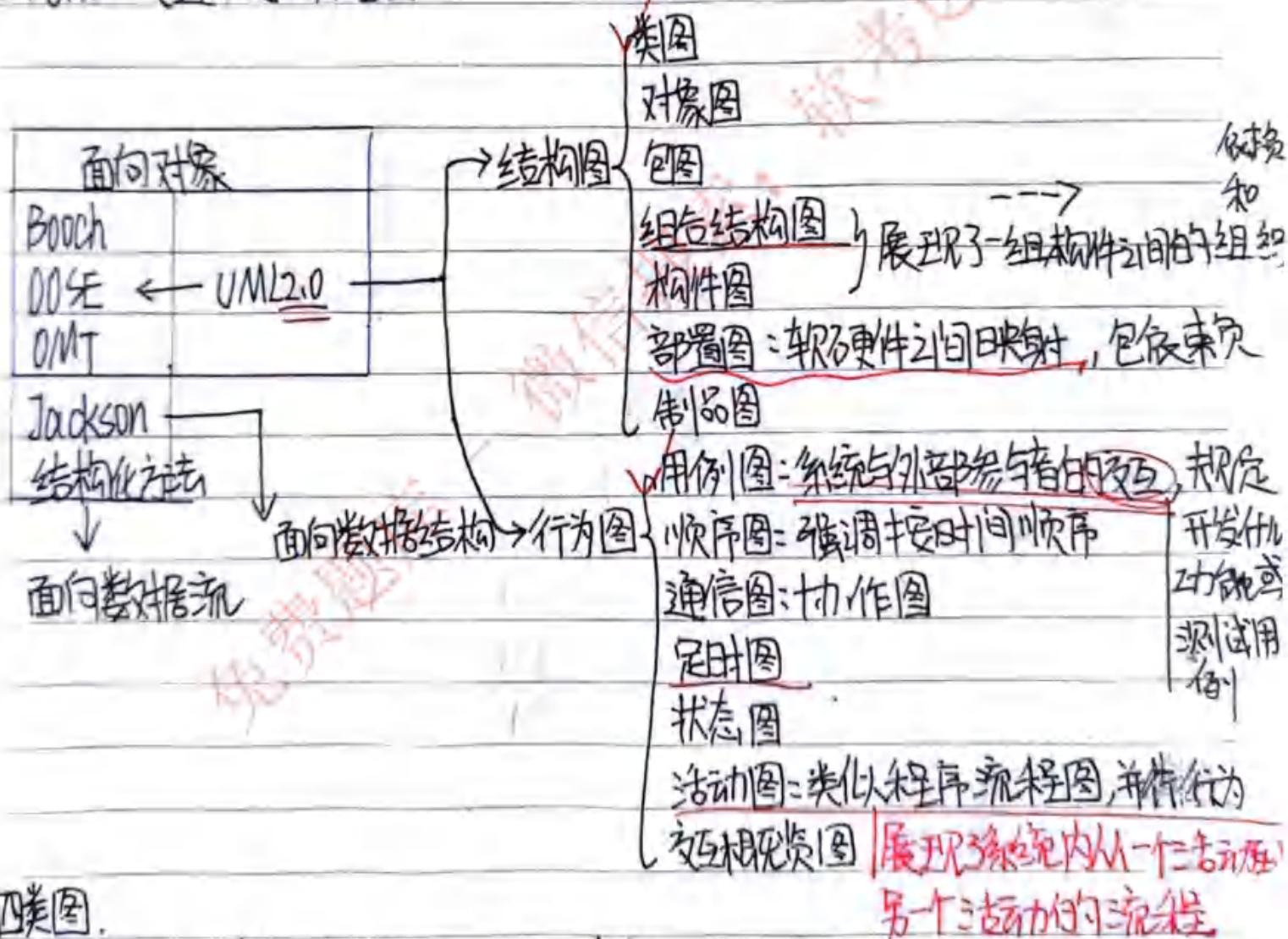
9. 消息和消息通信：消息是异步通信的

同步 异步

二、面向对象设计 7 大原则

1. 单一职责原则：设计目的单一的一类 **保持独立性**
2. 开放-封闭原则：对打展开放，对修改封闭
3. 李比替换原则：子类可以替换父类
4. 依赖倒置原则：要依赖于抽象，而不是具体实现 **针对接口编程，不要针对实现编程**
5. 接口隔离原则：使用多专门的接口比使用单一的差，接口要好
6. 组合重用原则：要尽量使用组合，而不是继承关系达到重用目的
7. 迪米特原则（最少知识法则）：一个对象应当对其他对象尽可能少的了解。

三、UML（通用建模语言）



四、类图

依赖关系：一个事物发生变化影响另一个事物

类A ----> 类B

泛化关系：特殊-一般关系

子类 ----> 父类

父子

继承

用于在已存在的类的基础上创建新类，
它是做什么的具体实现，具体式

关联关系: 描述了对象之间的连续。

类A —— 类B.

组合关系: 整体与部分生命周期相同 部——◆整 部分离开整体不能使用

聚合关系: 整体与部分生命周期不同 "——◇" 部分离开整体可使用

实现关系: 接口与类之间的关系 -----▷

五、设计模式的分类.

1. 创建型模式

工厂方法 (factory method) 模式

抽象工厂 (abstract factory) 模式

原型 (prototype) 模式

单例 (singleton) 模式

建造者 (builder) 模式

2. 结构型模式

适配器 (adapter) 模式

桥接 (bridge) 模式

组合 (composite) 模式

装饰 (decorator) 模式

外观 (facade) 模式

享元 (flyweight) 模式

代理 (proxy) 模式

3. 行为型模式

职责链 (chain of responsibility) 模式

* 英文

命令 (command) 模式

解释器 (interpreter) 模式

迭代器 (iterator) 模式

中介者 (mediator) 模式

备忘录 (memento) 模式

观察者 (observer) 模式

状态 (state) 模式

策略 (strategy) 模式

模板方法 (template method) 模式

访问者 (visitor) 模式

✓ 生成器 (Builder) 模式: 适用于当创建复杂对象时, 算法应该独立于该对象的

组成部分时, 装配式时.

答题技巧

① 找到一个接口, 指向一个接口

② 针对系统提供统一的高层接口.

③ 对系统进行封装, 对外保持一个接

创建性模式

设计模式名称	简要说明	适用场景
抽象工厂模式	提供一个接口,可以创建一系列相关或相互依赖的对象,而无须指定它们具体的类。	生产一系列对象
构造器模式	将一个复杂类的表示,与其构造相分离,使得相同的构建过程能够得出不同的表示。	复杂对象构造
工厂方法模式	定义一个创建对象的接口,但由子类决定需要实例化哪一个类。工厂方法使得子类实例化的过程推迟。	动态生产对象
原型模式	用原型实例指定创建对象的类型,并且通过拷贝这个原型来创建新的对象。	克隆对象
单例模式	保证一个类只有一个实例,并提供一个访问它的全局访问点。	单实例

结构性模式

设计模式名称	简要说明	适用场景
适配器模式 Adapter	将一个类的接口转换成用户希望得到的另一种接口,它使原本不相容的接口得以协同工作。	转换接口
桥接模式	将类的抽象部分和它的实现部分分离开,使它们可以独立地变化。	继承树对部分
组合模式	将对象组合成树型结构来表示"整体-部分"的层次结构,使得用户对单个对象和组合对象的使用具有一致性。	树型目录结构
装饰模式 动态,添加额外职责	动态地给一个对象添加一些额外的职责,它提供了用子类扩展功能的一个灵活的替代,比原生一个子类更加灵活。	附加职责
外观模式	定义一个高层接口,为子系统的一组接口提供一个一致的外观,从而简化了该子系统的接口。	对统一接口 Compiler
享元模式	提供支持大量细粒度对象共享的有效方法。	共享对象
代理模式	为其他对象提供一种代理以控制这个对象的访问。	

桥接模式不希望将抽象和它的实现部分之间有一个固定绑定关系。

行为性模式

设计模式名称	简要说明	速记关键字
职责链模式	通过给多个对象处理请求的机会,减少请求的发送者与接收者之间的耦合,将接收对象链接起来,在链中传递请求直到有一个对象处理这个请求。	传递职责 Handler
命令模式	将一个请求封装为一个对象,从而可用不同的请求对客户进行参数化,将要求排队或记录请求日志,实现对撤销的操作。	日志记录 可撤销 Command
解释器模式 语言+文法+解释器	给定一种语言,定义它的文法表示,并定义一个解释器该解释器用来根据文法表示来解释语言中的句子。	虚拟机的 机制 Interpreter
迭代器模式	提供一种方法来顺序访问一个聚合对象中的各个元素,而不需要暴露该对象的内部表示。	数据库数据集
中介者模式 中介对象封装对象交互	用一个中介对象来封装一系列的对象交互,封装对象不需要显式地相互调用,从而达到了低耦合,还可以独立地改变对象间的交互。	不直接引用 Mediator
备忘录模式	在不破坏封装性的前提下,捕获一个对象的内部状态,并在该对象之外保存这个状态,从而可以在以后将该对象恢复到原先保存的状态。	
观察者模式 引入其他对象来分布通信	策略模式 定义了一系列的算法,并将每个算法封装起来,而且使它们可以相互替换。策略模式让算法独立于使用它们的客户而变化。这可能需要不同情况使用不同的策略(算法),或者策略还可能在未来用其他方式来实现。	联动 状态成类 频繁切换
状态模式		
策略模式		
访问者模式	算法+封装+相互替换+独立	



创建型模式

工厂模式 (Factory Method)

意图: 定义一个创建对象的接口, 但由于类决定需要实例化工厂方法使得子类实例化的过程推迟。

抽象工厂模式 (Abstract Factory)

意图: 提供一个接口, 可以创建一系列相关或相互依赖的对象, 而无须指定它们的类。

原型模式 (Prototype)

意图: 用原型实例指定创建对象的类型, 并且通过拷贝这个原型来创建新的对象。

模板方法模式 (Strategy)

意图: 定义一个操作中的算法骨架, 而将一些步骤延迟到子类中, 使得子类可以不改变一个算法的结构即可以重新定义算法的某些特定步骤。

第5章 数据结构与算法基础

上午10分

下午大题

课程内容简介

数组与矩阵 (☆☆)

排序与查找 (☆☆☆☆)

线性表 (☆☆☆☆)

时间复杂度与空间复杂度 (☆☆☆☆)

广表 (☆☆)

算法基础及常见的算法 (☆☆☆☆)

树与二叉树 (☆☆☆☆)

图 (☆☆)

一、数组

数组类型	存储地址计算
一维数组 $a[n]$	$a[i]$ 的存储地址为 $a + i * len$
二维数组 $a[m][n]$	$a[i][j]$ 的存储地址 (按行存储) 为: $a + (i * n + j) * e$ $a[i][j]$ 的存储地址 (按列存储) 为: $a + (j * m + i) * e$

例: 已知 25 行 5 列的二维数组 a 中的每个元素占两个字节, 求元素 $a[2][3]$ 按行优先存储的存储地址?

$$a + (2 * 5 + 3) * 2 = a + 26$$

代入法

二、稀疏矩阵

- 稀疏矩阵
- 上三角矩阵
- 下三角矩阵

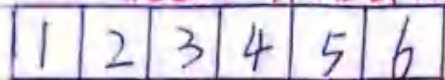
1. 若对线性表的最常用操作是访问任意指定序号的元素, 并在表尾加入和删除元素, 则适宜采用(顺序表)存储。

2. 在程序的执行过程中, 系统用(栈)实现嵌套调用(递归调用)函数的正确返回。

n

三、线性表

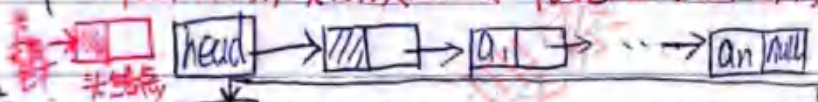
1. 顺序表



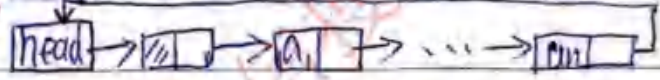
当加入新结点后, 头指针指向头结点, 链表为空时头结点仍存在
在链表中加入头结点后, 查找表中某一元素仍从头指针出发, 顺序找到目标元素或失败时找到表尾处, 时间复杂度与表长成正比

2. 链表

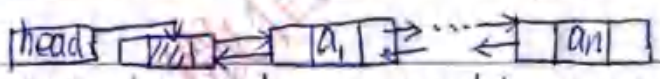
① 单链表



② 循环链表



③ 双向链表



3. 基本操作: 单链表删除, 插入结点; 双向链表删除, 插入结点。

4. 顺序存储与链式存储对比

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, 改变, 更优
时间性能	查找运算	$O(n/2)$	$O(n/2)$
	读运算	$O(1)$, 更优	$O([n+1]/2)$, 最好情况为1, 最坏为n
	插入运算	$O(n/2)$, 最好情况为0, 最坏情况为n	$O(1)$, 更优
	删除运算		$O(1)$, 更优

5. 队列与栈

队列: 先进先出

则入栈操作和出栈操作都不需要遍历

栈: 先进后出: 若用单链表作为栈的存储结构, 并用头指针作为栈顶指针

依次元素 a, b, c 的顺序进入栈, 请写出所有可能的出栈序列。

5种 cba, bca, bac, abc, acb

循环队列 队空条件: $head = tail$
队满: $(tail + 1) \% 总长 = head$

四 广义表

1. 广义表是 n 个表元素组成的有限序列, 是线性表。
2. 通常用递归的形式进行定义, 记做 $LS = (a_0, a_1, \dots, a_n)$
3. 注: 其中 LS 是表名, a_i 是表元素, 它可以是表 (称做子表), 也可以是数对元素 (称为原子), 其中 n 是广义表的长度 (也就是最外层包含的元素个数), $n=0$ 的广义表为空表; 而递归定义的重数就是广义表的深度, 直观的说, 就是定义中的括号的重数 (原子的深度为 0, 空表的深度为 1)

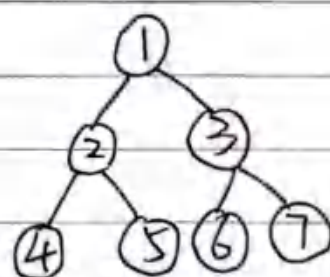
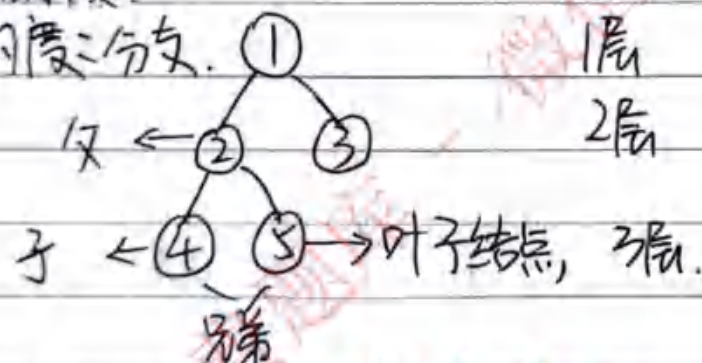
若有 $LS_1 = (a, (b, c), (d, e))$, 则 $head(LS_1) = a$; $tail(LS_1) = ((b, c), (d, e))$
 $head(head(tail(LS_1))) = b$

五 树与二叉树

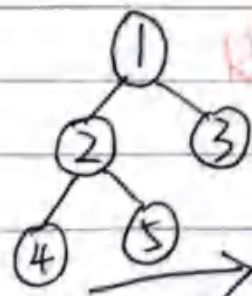
层数: 树的深度

结点的度

树的度: 分支



满二叉树 (所有结点的度为 2)



完全二叉树

二叉树的重要特性:

1. 在二叉树的第 i 层上最多有 2^{i-1} 个结点, ($i \geq 1$)
2. 深度为 k 的二叉树最多有 $2^k - 1$ 个结点, ($k \geq 1$)
3. 对任何一棵二叉树, 如果其叶子结点个数为 n_0 , 度为 2 的结点个数为 n_2 , 则有 $n_0 = n_2 + 1$
4. 如果对一棵有 n 个结点的完全二叉树的结点按层序编号 (从第 1 层到 $\log_2 n$ 层, 每层从左到右), 对任一结点 i ($1 \leq i \leq n$), 有:
 - ① 如果 $i=1$, 则结点无父结点, 是二叉树的根; 如果



① $i > 1$, 则父结点是 $\lfloor \frac{i}{2} \rfloor$;

② 如果 $2i > n$, 则结点 i 为叶子结点, 否则, 其左子结点是 $2i$;

③ 如果 $2i+1 > n$, 则结点 i 无右子叶点, 否则, 其右子结点是 $2i+1$.

二叉树遍历

① 前序遍历 (根左右)

② 中序遍历 (左根右)

③ 后序遍历 (左右根)

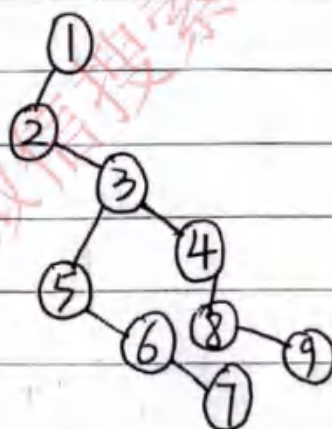
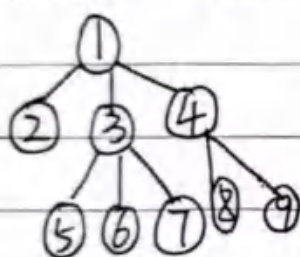
④ 层次遍历

双向构造二叉树

树转二叉树

① 子孩子结点, 一左子树对结点,

② 兄弟结点, 一右子孩子结点,



查找二叉树

① 左孩子小于根

② 右孩子大于根

插入结点:

① 若该关键字值结点已存在, 则不再插入;

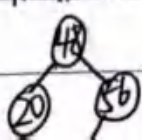
② 若查找二叉树为空树, 则以该结点为查找二叉树;

③ 将要插入点值与插入后结点值比较, 确定。

删除结点:

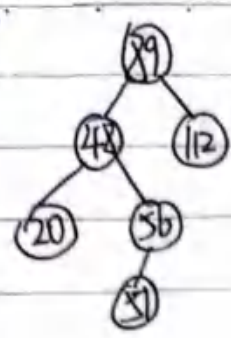
① 若待删除结点是叶子结点, 则直接删除;

② 若



删除 5b, 则 5a 代替 5b。

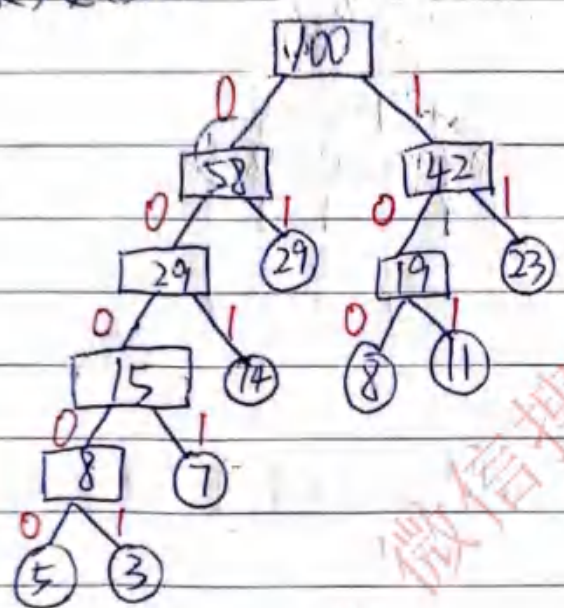
删除89, 则56代替89, 51代替56。



最优二叉树 (哈夫曼树) —— 一类带权路径长度最短的树。

例: 假设有组权值 5, 29, 7, 8, 14, 23, 3, 11 请尝试构造哈夫曼树。

① 找权值最少数组组合。



编码采用“左0右1”

带权路径长度 = $5 \times 5 + 29 \times 2 + 7 \times 4 + 8 \times 4 + 14 \times 3 + 23 \times 2 + 3 \times 5 + 11 \times 3 = 246$

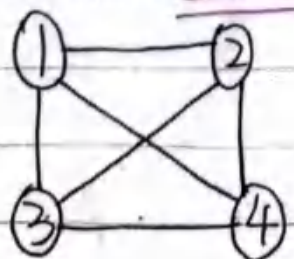
线索二叉树 (记住前/后继)

平衡二叉树

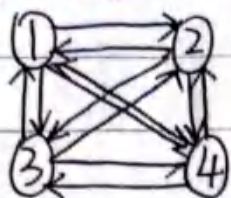
判断: 任意结点的左右子树深度相差不超过1
每结点的平衡度只能为-1, 0或1

六图

完全图: 在无向图中, 若每对顶点之间都有条边相连, 则称该图为完全图。



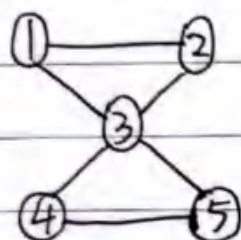
在有向图中,若每对顶点之间都有一对自向边相互连接,则称该图为完全图。



邻接矩阵

用一个 n 阶矩阵来存放图中各结点的关联信息,其矩阵元素 R_{ij} 定义为:

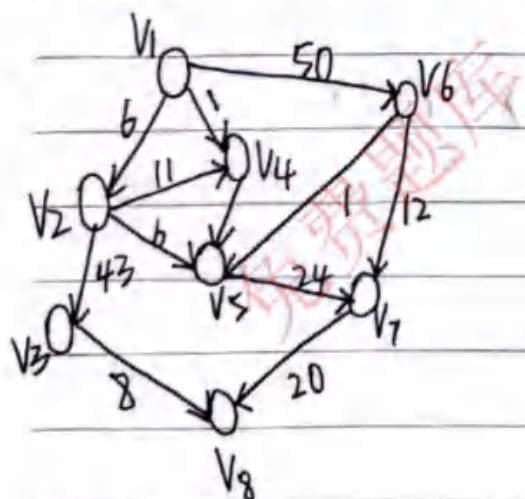
$$R_{ij} = \begin{cases} 1 & \text{若顶点 } i \text{ 到顶点 } j \text{ 有邻接边} \\ 0 & \text{若顶点 } i \text{ 到顶点 } j \text{ 无邻接边} \end{cases}$$



$$R_1 = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

邻接表

首先把每个顶点的邻接顶点,用链表示出来,然后用一个一维数组来顺序存储上面每链表的头指针。



V_1	2	6	4	11	6	50	1
V_2							
V_3							
V_4							
V_5							

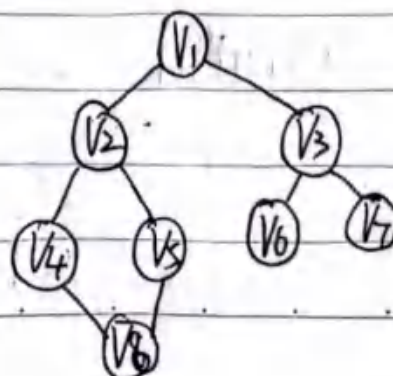
图的遍历

✓ 深度优先

$V_1 \rightarrow V_2 \rightarrow V_4 \rightarrow V_8 \rightarrow V_5 \rightarrow V_3 \rightarrow V_6 \rightarrow V_7$

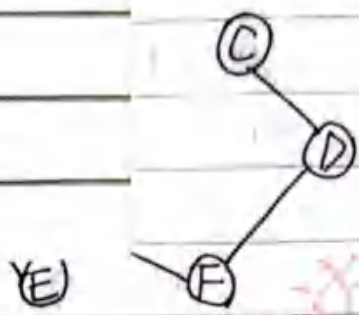
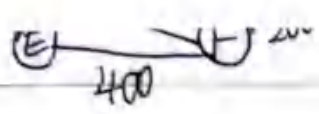
✓ 广度优先

$V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_6 \rightarrow V_7 \rightarrow V_8$



1. 果简单无向连通图G的顶点数为n, 则图G最少和最多分别有 $(n-1, n*(n-1)/2)$ 条边。
2. 无向连通图只保证每对结点间都有路径。它的邻接矩阵是对称矩阵。

向图称为顶点表示活动网络,



第8章 程序设计语言与语言处理程序基础 3-5分

- 编译与解释 (★★)
- 文法 (★★)
- ★ 正规式 (★★★★)
- 有限自动机 (★)
- 后缀表达式 (★★)
- ★ 传值与传址 (★★★★)
- 多种程序语言特点 (★★)

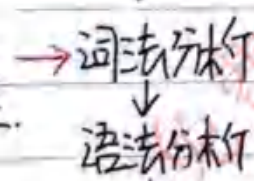
一、编译过程

语法分析阶段的输入是记号流。

源程序

词法错误: 非法字符, 关键字或标识符拼写错误。

解释型
编译型



正规式, 有限自动机
语法错误: 语法结构出错, if, end if 不匹配, 缺分号。括号不匹配

静态
动态

语义分析

语义错误: 死循环, 零除数, 其他逻辑错误。云编译

中间代码生成

常见的: 后缀式

编译: 直接生成目标代码, 在机器上执行。

代码优化

三地址码

而编译器不参与, 速度快。

目标代码生成

树

解释: 生成中间代码或等价形式, 程序执行需要参与执行, 并控制, 速度慢

目标程序

二、文法定义 (描述一种新的语言)

一个形式文法是一个有序四元组 $G = (V, T, S, P)$, 其中:

① V : 非终结符, 是语言组成部分, 不是最终结果, 可理解为占位符。

② T : 终结符, 是语言的组成部分, 是最终结果。 $V \cap T = \emptyset$

③ S : 起始符。是语言的开始符号

④ P : 产生式。用终结符替代非终结符的规则。形如 $\alpha \rightarrow \beta$

正则闭包: $A^+ = A^1 U A^2 U A^3 \dots U A^n U \dots$

闭包: $A^* = A^0 U A^+$ (在正则闭包的基础上, 加上 $A^0 = \{\epsilon\}$)。

例 $a^* = (a, aa, aaa, \dots, \epsilon)$, 而 $(ab)^* = (ab, abab, \dots, \epsilon)$

类型	别称	说明	对应自动机
0型	短语法	G 的任何产生式 $\alpha \rightarrow \beta$ 满足 α 属于 V 的正则闭包, 且至少含有一个非终结符, 而 β 属于 V 的闭包。	图灵机
1型	上下文有关文法	G 的任何产生式 $\alpha \rightarrow \beta$ 满足 $ \alpha \leq \beta $, 仅仅 $S \rightarrow \epsilon$ 例外, 但 S 不得出现在任何产生式右部	线性界限自动机
2型	上下文无关文法	G 的任何产生式为 $A \rightarrow \beta$, A 为非终结符, β 为 V 的闭包	非确定的下推自动机
3型	正规文法	G 的任何产生式为 $A \rightarrow \alpha B$ 或 $A \rightarrow \alpha$, α 属于非终结符的闭包, A, B 都属于非终结符。	有限自动机

三、语法推导树

一棵语法树应具有以下特征:

1. 每个结点都有一个标记, 此标记是 V 的一个符号。

2. 根的标记是 S 。

3. 若一结点 n 至少有一个它自己除的 M 的子树, 并且有标记 A , 则 A 肯定在 V_N 中;

4. 若如果结点 n 的直接子树, 从左到右的次序是结点 $n_1, n_2, \dots, n_k$, 其标记分别为 $A_1, A_2, \dots, A_k$ 那么 $A \rightarrow A_1 A_2 \dots A_k$ 一定是 P 中的一个产生式。

例. 文法 $G = (\{ab\}, \{S, A\}, S, P)$, 其中:

$S \rightarrow aAS|a;$

$A \rightarrow SbA|SS|ba$

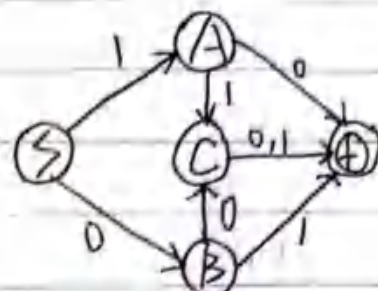
请构造句型 $aabAa$ 的推导树

$aAS = a(SbA)S = a(abA)S = aabAa$

四有限自动机

$$M = (S, \Sigma, \delta, s_0, Z)$$

- 1) S 是一个有限集，每个元素为一个状态，
- 2) Σ 是一个有穷字母表，每个元素为一个输入字符
- 3) δ 是转换函数，是一个单值对照，
- 4) s_0 ，属于 S ，是其唯一的初态，
- 5) Z 是一个终态集 (可空)



$S \rightarrow F$
10
01
1110, 11
0010, 11

五正规式

正规式是描述程序语言单词的表达式，对于字母 Σ ，其上的正规式及其表示的正规集可以递归定义如下：

- ① ε 是一个正规式，它表示集合 $L(\varepsilon) = \{\varepsilon\}$
- ② 若 a 是 Σ 上的字符，则 a 是一个正规式，它所表示的正规集 $L(a) = \{a\}$ 。
- ③ 若正规式 r 和 s 分别表示正规集 $L(r) = L(s)$ ，则

$r|s$ 是正规式，表示集合 $L(r) \cup L(s)$;

$r \cdot s$ 是正规式，表示集合 $L(r)L(s)$;

r^* 是正规式，表示集合 $(L(r))^*$;

(r) 是正规式，表示集合 $L(r)$ 。

反由有限次地使用上述三个步骤定义的表达式是 Σ 上的正规式。由此可见，正规式要么为空，要么由字母、或、连接、闭包运算符组成。其中闭包运算符“*”具有最高的优先级，连接运算符具有次高优先级，或运算符“|”具有最低优先级。

六数据类型与程序控制结构

常见的数据类型：数字数据类型、布尔类型、字符类型、枚举类型、指针类型。

程序控制结构主要有：顺序结构、选择结构、循环结构。

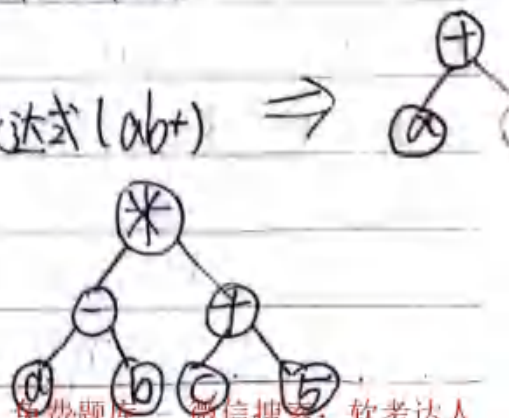
七程序语言基础

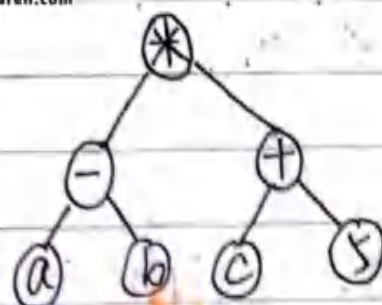
表达式：前缀表达式 $(+ab)$ 、中缀表达式 $(a+b)$ 、后缀表达式 $(ab+)$ $\Rightarrow$

例表达式 $(a-b) * (c+d)$ 的后缀式是

方法一： $((a-b) * (c+d))$ 即 $ab-cd+*$

方法二：





ab-c5+*

八函数调用
传值与传址

```

int  FunctionExample  int x, float y
↑      ↑              ↑
返回值类型  过程名    形参列表
{ ... ← 过程体
}
return 0;  返回
    
```

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

引用(传址)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也就改变了实参的值。

九. 程序语言基础 - 各种程序语言特点

1. Fortran 语言(科学计算, 执行效率高)

2. Pascal 语言(为教学而开发的, 表达能力强, Delphi)

3. C语言(指针操作能力强, 高效)

4. Lisp 语言(函数式程序语言, 符号处理、人工智能)

5. C++ 语言(面向对象, 高效)

6. Java 语言(面向对象, 中间代码, 跨平台)

7. C# 语言(面向对象, 中间代码, .Net)

8. Prolog 语言(逻辑推理, 简洁性, 表达能力, 数据库和专家系统)

9. Python 语言(解释型, 面向对象, 脚本语言)

10. PHP: 适合用来进行动态网页处理

HTML: 静态网页

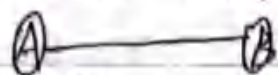
UML — 统一建模语言

1. 类图: 以图形化的方式描述系统中对象及其关联关系。

专注于系统的静态视图。

包括多重性、关联关系、泛化/特化关系以及聚合关系。

↓ 双向关系, 即关联的对象双方都能与对方通信



展现了一组对象、接口、协作和它们之间的关系。

UML模型图的构成

相性事物: UML模型的静态部分, 描述概念或物理元素。

类	具有相同属性相同操作相同关系相异的对象的描述
接口	描述元素的外部观行为, 即服务集合的定义说明
协作	描述了一组事物间的相互作用的集合
用例	代表一个系统或系统的一部分行为, 是一组动作序列的集合。
构件	系统中物理存在, 可替换的部件
节点	运行中存在的物理元素

关系: 把事物紧密联系在一起

依赖	两个事物间的语义关系, 其中一个事物(独立事物)发生变化, 会影响到另一个事物(依赖事物)的语义。
关联	一种结构关系, 指明一个事物的对象与另一个事物的对象间的联系
泛化	一种特殊-一般的关系。继承
实现	类元之间的语义关系, 其中的一个类元指定了另一个类元保证执行的契约

图: 事物和关系的可视化表示。

用例图: 从用户角度描述系统功能, 是用户所能观察到的系统功能的模型图, 用例是系统中的一个功能单元。这些图在对系统行为进行组织, 建模。

类图: <1>描述系统中类的静态结构, 既定义系统中的类, 表示类之间的相互关系, 如关联、依赖、聚合等, 也包括类的内部结构(属性和操作)

<2>以类为中心类组织的, 类图中的其他元素或属性属于某个类或与类相关联

对一个复杂用例中的业务处理流程进行进一步建模的最佳工具是UML活动图。

下列关于类的静态成员的描述,不正确的是()

- A. 该类的对象共享其静态成员变量的值
- B. 静态成员变量可被该类的所有方法访问
- C. 该类的静态方法只能访问该类的静态成员变量
- D. 该类的静态数据成员变量的值不可修改

状态图

- ①用于描述一个对象在外部用例中的行为
- ②用于某些具有多个状态的对象而不是系统中大多数或全部对象
- ③用于用户界面或控制对象

序列图、通信图、互相视图、时序图被称为交互图。

使用类图情况的3种

- ①对系统的词汇进行建模。
- ②对简单协作进行建模。
- ③对逻辑数据库模式建模。

使用对象图的情况

- ①对象快照

选择候选类的原则,以及对候选类进行删除的原则

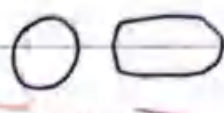
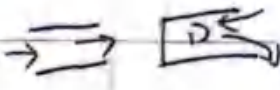
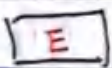
选择候选类时通常考虑的是问题域中自然存在的名词。

具有下列特征的候选类需要删除:含义相近(冗余)、含义不明确的对象、暗示实现的类、表示属性或特征、有动词含义的名词(表示行为和动作)。

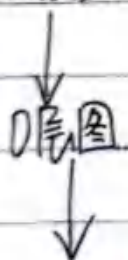
Composite 模式 (组合)

例: 以树形结构表示个体参与者和团体参与者之间的“整体-部分”关系,使得对单个对象和组合对象的使用具有一致性。

数据流图：结构化分析的一个重要模型。描述数据在系统中如何被传递或变换，以及描述如何对数据流进行变换的功能，用开功能建模。

元素	说明	图元
数据流	由一组固定成分的数据组成，表示数据的流向，每数据流通常有一个合适的名称，反映数据流的含义。	
加工	加工描述输入数据流到输出数据流之间的变换，也就是输入数据流做了什么处理后变成输出数据流。	
数据存储 (文件)	用来表示暂时存储的数据，每个文件都有名字。流向文件的数据流表示写文件，流出的表示读文件。	
外部实体	指存在于软件系统外的人员或组织。	

顶层图



逐步细化
父子平衡

DFD/4.1 4.2 4.3

数据字典 — 配合数据流图的使用

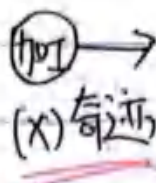
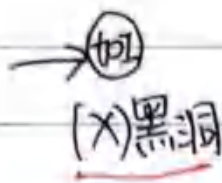
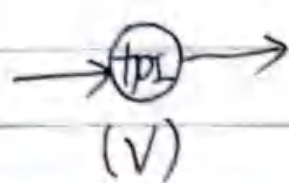
符号	含义	举例说明
=	被定为	机票 = 姓名 + 日期 + ...
+	与	$X = a + b$, X 由 a 和 b 组成
[... , ...]	或	$X = [a, b]$ $X = [a b]$
或 [... ...]		
$\{ \dots \}$	重复	$X = \{a\}$ 表示 X 由0个或多个 a 组成
(...)	可选	$X = (a)$ 表示 a 可在 X 中出现，也可以不出现



DFD 平衡原则

父图与子图之间的平衡

子图内平衡



数据分析结果表示了现实世界中实体的属性随时间相互关系。

- 加工
- ① 一个加工必须既有输入又有输出
 - ② 数据流只能与加工有关。即从加工流向加工，数据源流向加工或加工流向数据源

建模图1和图1-2时如何保持数据流图的平衡

图1-1(父图)中某加工的输入输出数据流必须与图1-2(子图)的输入输出数据流在数量和名字上相同；图1-1(或父图)中的一条输入(或输出)数据流对应于图1-2(子图)中几个输入(或输出)数据流，而图1-2(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条数据流。

数据流图设计原则

- ① 数据守恒原则
- ② 守恒加工原则
- ③ 对于每个加工，必须既有输入数据流，又有输出数据流。
- ④ 外部实体与外部实体之间不存在数据流
- ⑤ 外部实体与外部存储之间不存在数据流
- ⑥ 数据存储与数据存储之间不存在数据流
- ⑦ 父图与子图之间的平衡原则
- ⑧ 数据流与加工有关，且必须经过加工。

多媒体基础知识,

一、音频

乐器/人耳 20Hz-20KHz 声音的带宽 → 说话: 300-3400Hz.

A/D 转换: 采样 → 量化 → 编码

↓ ↓
模 数

↓
采样频率
↓
采样精度

采样频率应为声音最高频率的 1.5~2 倍. 可保证不失真,

常见的音频格式: WAVE, MP3, MIDI (乐器)

二、图像相关概念,

1. 亮度

2. 色周: 颜色的种类

3. 饱和度: 色彩的纯洁性, 即颜色的艳丽程度



设计模式的适用场合

观察者模式 (Observer)

意图:

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自动更新。

(1) 当一个抽象模型有两个方面, 其中一个方面依赖于另一个方面, 将这两者封装在独立的对象中以便它们可以各自独立地改变和复用。

(2) 当对一个对象的改变需要同时改变其他对象, 而不知道具体有多少对象有待改变时。

(3) 当一个对象必须通知其他对象, 而它又不能假设其他对象是谁, 且不希望这些对象耦合的。

Subject (目标) 知道它的观察者, 可以有任意多个观察者观察同一个目标;

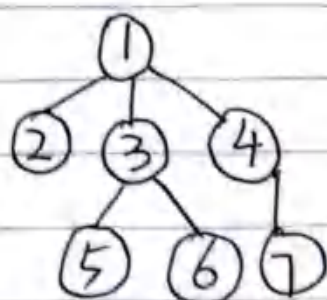
Observe (观察者) 为那些目标发生改变时需要获得通知的对象定义一个更新接口。

免费题库 - 微信搜索

树与二叉树的特性

树可进行先根遍历、后根遍历、层序遍历。

例



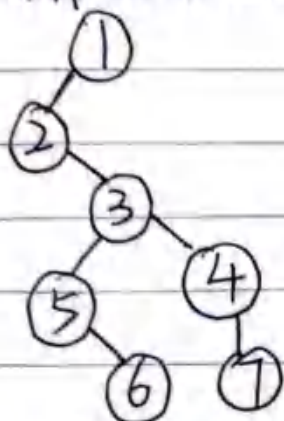
先根遍历: 1, 2, 3, 5, 6, 4, 7

后根遍历: 2, 5, 6, 3, 7, 4, 1

层序遍历: 1, 2, 3, 4, 5, 6, 7

二叉树: 先序遍历、后序遍历、层序遍历。

例



先序遍历: 1, 2, 3, 5, 4, 6, 7

后序遍历: 6, 5, 7, 4, 3, 2, 1

中序遍历: 2, 5, 6, 3, 7, 4, 1

层序遍历: 1, 2, 3, 5, 4, 6, 7

将树转换为二叉树后, 树的先根序列 = ^对对应二叉树的先序序列
树的后根序列 = 对应二叉树的中序序列

堆

定义: 一颗顺序存储的完全二叉树。

小根堆: $R_i \leq R_{i+1}$ 且 $R_i \leq R_{i+2}$ 。

大根堆: $R_i \geq R_{i+1}$ 且 $R_i \geq R_{i+2}$ 。

排序

类别	排序方法	时间复杂度		空间复杂度 辅助存储	稳定性
		平均情况	最坏情况		
插入排序	直接插入	$O(n^2)$	$O(n^2)$	$O(1)$	稳定
	shell排序	$O(n^{1.5})$	$O(n^2)$	$O(1)$	不稳定
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
	堆排序	$O(n \log_2 n)$	$O(n \log_2 n)$	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	$O(n^2)$	$O(1)$	稳定
	快速排序	$O(n \log_2 n)$	$O(n^2)$	$O(n \log_2 n)$	不稳定
归并排序		$O(n \log_2 n)$	$O(n \log_2 n)$	$O(n)$	稳定
基数排序		$O(d \cdot r \cdot n)$	$O(d \cdot r \cdot n)$	$O(r \cdot n)$	稳定

algorithm	算法.	implement	实施执行
plunge into	应用到, 跳入	outlining	概述.
construction	建筑构造构造	worthiness	值得.
usable	可用的, 适用的.		
redesigned	重新设计		
discard	丢掉.		
alternative	可供选择的事物		
version	版本		
throw away	扔掉		
pilot	实验型.		
the project workbook	项目工作手册		
separate	独立的		
internal	内部的		
crafted	规定的.		
haphazard	杂乱的		
external	外部的.		
internal	内部的		
distribution	发布.		
hardware	硬件		
software	软件.		
release plan	发布计划		
specification	规格.		
feasibility	可行性		
traceability	可追踪性.		
modifiability	可变性.		
correctness	正确.		
navigate	导航.		



计算机网络

1. 常见TCP/IP协议

DHCP协议的功能是自动分配IP地址。

FTP协议的作用是文件传输，并且是基于TCP的。

传输层的协议包括TCP和UDP。

OSPF：链路状态路由选择协议或最短路径优先协议。

2. 把CSS样式表与HTML网页关联，可使用的方法

① 在HTML对应的<head>标签内定义CSS样式

② 用<import>引入样式表文件

③ 用<link>标签链接网上可访问的CSS样式表文件

3. 多模光纤：成本低，宽芯线，聚光好，耗散大，低效，用于低速度，短距离的通信。

单模光纤：成本高，窄芯线，需要激光光源，耗散小，高效，用于高速度，长距离的通信。

4. 一个标准URL的格式：

协议://主机名.域名.域名后缀或IP地址(必需2号)/目录/文件名

5. 在ASP的内置对象中，能修改Cookie中的值的是response，还可以创建Cookie，而request可以访问Cookie中的参数。

6. 网络攻击有主动攻击和被动攻击两类。其中主动攻击是指通过一系列的方法，主动向被攻击对象实施破坏的一种攻击方式，例如重放攻击、IP地址欺骗、拒绝服务攻击等。

流量分析攻击是通过持续检测现有网络中的流量变化或变化趋势，而得到相应信息的一种被动攻击方式。

7. 无效的IP地址：169.254.X.X (Windows) } DHCP配置的内容
0.0.0.0 (Linux)

8. 测试网络连通性通常采用的命令是(ping)，可以很好地帮助我们分析和判定网络故障。

9. 使用的端口号在0-1023。

多媒体基础

1. PC机处理的音频信号主要是人耳听得到的音频信号, 它的频率范围是(20Hz-20kHz)

小于20Hz的声波信号为亚音信号(次音信号)

20Hz-20kHz的为音频信号.

高于20kHz的信号称为超音频率信号(超声波)

2. Xara 主要用于动画编辑和处理。

3. 数字语音的采样频率定义为8kHz, 这是因为(语音信号定义的频率最高值为4kHz) 如果取样速率大于模拟信号最高频率的2倍, 则可以用得到的样本恢复原来的模拟信号。

免费题库 - 微信搜索: 软考达人

程序设计语言基础

1. **错误管理**：一般程序中的错误可分为语法错误和语义错误(或逻辑错误)，语法错误是语句形式上的错误，语义错误是含义的错误(逻辑错误)。
2. **中间代码**：一种简单且含义明确的记号系统，与具体的机器无关，可以有若干种形式。可以将不同的高级程序语言编译成同一种中间代码。由于与具体机器无关，使用中间代码有利于进行与机器无关的优化处理，以提高编译程序的可靠性和移植性。常用的中间代码有后缀式、三元式、四元式和控制流图等形式。
3. XML(可扩展标记语言)是标准通用标记语言的子集，是一种用于标记电子文件使其具有结构性的标记语言。
4. Python是一种面向对象、解释型计算机程序设计语言。
5. Prolog是逻辑型程序设计语言。
6. 语言具有独立含义的最小语法单位是符号(单词)。如标识符、无符号常数与界限符。
7. **词法分析**：总是把构成源程序的字符串转换成单词符号序列。
8. **有限自动机**：一种识别装置的抽象概念，它能准确地识别正规集。
分类：确定的有限自动机(DFA)，不确定的有限自动机(NFA)
9. 对高级语言源程序进行编译过程中，有限自动机(DFA或NFA)是进行词法分析的工具。
10. PHP更适合作为进行动态网页处理。
11. Lisp一种基于谓词的函数式编程语言。
12. 脚本语言主要采用解释方式实现。
13. 在程序运行过程中，(将整数变量与浮点型变量相加)时涉及整型数据转换为浮点型数据的操作。



软件工程

1. RUP应用了角色、活动、制品和工作流4种重要的模型元素,其中角色表述“谁做”,制品表述“做什么”,活动表述“怎么做”,工作流表述“什么时候做”。

2. 自顶向下集成

优点:较早地验证了主要控制和判断点;按深度优点,可以首先实现和验证一个完整的软件功能;功能较早证实,带来信心;只需一个驱动,减少驱动程序开发的费用;支持故障隔离。

缺点:桩件开发量大;底层验证被推迟;底层组件测试不充分

★ 适用于产品控制结构比较清晰和稳定;高层接口变化较少;底层接口未定或经常可能被修改;产品的主要控制组件具有较大的技术风险,需要早被验证;希望早能看到产品的系统功能行为。

3. 自底向上集成

优点:对底层组件行为较早验证;工作最初可以并行集成,比自顶下的效率高;减少了桩的工作量;支持故障隔离。

缺点:驱动组件的开发工作量大;对高层的验证被推迟,设计上的错误不易被及时发现。

★ 适用于底层接口比较稳定;高层接口变化比较频繁;底层组件较早被完成。

4. 在基于Web的电子商务应用中,访问存储在数据库中的业务对象的常用方式之一是

(JDBC) Java Database Connectivity

5. 系统设计为系统制定蓝图,软件设计模型关注整个系统总体结构、代码设计、处理过程、数据结构和界面模型等。

4. 好的测试用例在设计测试用例时,一个好的无效等价类,应该只从一个角度违反规则。

5. 数据流图必须要保持父图与子图平衡,且每个加工必须要有输入数据流,输出数据流,而每条数据流至少有一端是加工,而且还要能保持数据守恒。

6. 决策树一种适合于描述加工中具有多个策略且每个策略和若干条件有关的逻辑功能的图形工具。

例:数据流图中某个加工的一组动作依赖于逻辑条件的取值,则用(决策树)

能够清楚地表示复杂的条件组合与应做的动作之间的对应关系。

7. 开发模型

初期阶段 → 细化阶段 → 构建阶段 → 测试阶段 → **产生阶段**

8. 面向对象开发方法有 Booch 方法, Goad 方法和 OMT 方法。

Jackson 方法是一种面向数据结构的数据开发方法。

9. **极限编程 (XP)** 是敏捷开发的典型方法之一, 是一种轻量级(敏捷)、高效、低风险、柔性、可预测的、科学的软件开发方法, 由价值观、原则、实践和行为 4 个部分组成。其中价值观为沟通、简单性、反馈和勇气。

10. **A** 不属于软件设计质量评审。 **用户满意度**

A 功能与模块之间的对应关系

B 软件规格说明是否符合用户的要求

C 软件是否具有可测试性

D 软件是否具有有良好的可靠性

11. 软件维护的内容一般包括 正确性维护、适应性维护、完善性维护、预防性维护。

12. 单元测试主要检查模块五个特征: 模块接口、局部数据结构、重要的执行路径、出错处理和边界条件。

13. 软件风险

① 特性: 不确定性、损失

② 救火和危机管理是对不适合但经常采用的软件风险管理策略。

③ 分类: 已知风险、救火风险

④ 人员工作和预算是在识别风险时需要识别的因素。

14. 风险管理

① 对风险排优先级是根据风险的曝光度来进行的, $\text{曝光度} = \text{风险的产生后果} \times \text{风险发生的概率}$

② 可以通过改变系统性能或功能需求来避免某些风险

③ 不可能去除所有风险, 但可以通过采取行动来降低或减轻风险

④ 在项目开发过程中, 需要定期地评估和管理风险。

数据结构与算法基础

1. 现需要对一个基本有序的数组进行排序。此时最适宜的算法为(插入)排序算法, 时间复杂度为($O(n^2)$)

2. 在串比较、求子串、串连接、串替换运算中, 除了串替换外, 其他运算都不会改变串中的内容, 因此, 在链表存储式下进行串替换最方便。

3. 将树转换为二叉树后:

树的先根序列 = 对应的二叉树的先序序列, 树的后根序列 = 对应的二叉树的中序序列

免费题库 - 微信搜索: 软考达人

信息安全

1. 传输经过SSL加密的网页所采用的协议是(HTTPS)
2. **熊猫烧香**是一种感染型的蠕虫病毒,它能感染系统中.exe, .v, .pif, .src, .html和.asp等文件,还能中止大量的反病毒软件进程并且会删除扩展名为gho的文件,该文件是一系统备份工具GHOST的备份文件,使用户的系统备份文件丢失。
被感染的用户系统中所有.exe可执行文件全部被改成熊猫举着三柱香的模样。
例:感染“熊猫烧香”病毒后的计算机不会出现(D)的情况。
A 执行文件图标变成熊猫烧香
B 用户信息被泄漏
C 系统运行变慢
D 破坏计算机主板
3. **数字证书**在PKI体制中,识别数字证书的颁发机构以及通过该机构核实证书的有效性,了解证书是否被篡改均通过一种机制——对数字证书做数字签名。数字签名将由CA机构使用自己的私钥进行。
4. 终端设备与远程站点之间建立安全连接的协议是SSH (Secure shell)
5. 为说明某一问题在学术论文中需要引用某些资料,但仅限于介绍、评论作品,只要构成自己作品的主要部分,可适当引用资料,而不必征得作者的同意,不需要向他支付报酬。
6. **防火墙的功能特性**: ①控制进出网络的数据包和数据流向 ②提供流量信息的日志和审计 ③隐藏内部IP以及网络结构细节
7. **数字签名** 先对正文产生信息摘要,之后使用发送者A的私钥对该信息摘要进行加密,这就完成签名。当接收者B收到签名的摘要以后,会对摘要使用发送者A的公钥进行解密(认证),若能认证,则表明信息是由A发送的。
A的私钥加密, A的公钥解密
8. 重放攻击:又称为重播攻击,回放攻击或新鲜性攻击。是指攻击者发送一个目的主机已接收过的包,来达到欺骗系统的目的,主要用于身份认证过程,破坏



认证的正确性。

kerberos 系统中可通过在报文中加入时间戳来防止重放攻击。

9. Macro 表示宏病毒：感染对象是使用某些程序创建的本体文档、数据库、电子表格等文件。 word.exe

软考达人
微信搜索：
免费题库

面向对象技术

1. 在面向对象软件开发过程中, 采用设计模式 (以复用成功的设计和体系结构)
2. 敏捷开发方法中, (水晶法 Crystal) 认为每一种不同的项目都需要一套不同的策略、约定和方法论。
3. 面向对象分析的第一步是 (确定问题域)
4. 多态分为参数多态, 包含多态, 重载多态, 和强制多态, 四种不同形式, 其中 (包含多态) 在许多语言中都有存在, 最常用的为子类型化。
5. 聚合对象是指一个对象 (包含其他对象)
6. 采用面向对象方法进行软件开发, 分析阶段架构师主要关注系统的行为, 即系统应该做什么。
7. 在对象或类模型中不包含 (D)
A 属性 B 操作 C 关联 D 对象或对象
8. 继承: 父类和子类之间共享类属性和方法的机制。
9. 覆盖: 子类的方法覆盖了基类的方法, 以实现不同的功能, 或者对父类的功能进行扩充。

免费题库

1. 把网络 117.15.32.0/23 划分为 117.15.32.0/27, 得到的子网是 (16) 个, 每个子网中可使用的地址是 (30)

题中将网络号为 23 的 IP 地址划分为网络号为 27 位的子网, 可以分为 $2^{(27-23)} = 16$ 个子网, 每个子网产生的主机地址是 $2^{32-27} - 2 = 30$ 个

知识点: 若要把 $xxx.xxx.xxx.0/m$ 划分为 $xxx.xxx.xxx.0/n$ ($m < n < 32$), 得到的子网个数为 $2^{(n-m)}$, 可用子网个数为 $2^{(n-m)}$, 每个子网中可使用的主机地址是 $2^{(32-n)} - 2$.

计算主机号时, 需除去全 0 (网络号) 和全 1 (广播) 这两个地址。

2. 虚拟存储体系由 (主存 - 辅存) 两级存储器构成。

3. 一个类中可以拥有多个名称相同而参数表 (参数类型或参数个数或参数类型顺序) 不同的方法, 称为 (方法重载)

4. 著作权, (署名权) 的保护期不受限制。

5. 以下的算法设计方法中, (动态规划) 以获取问题最优解为目标。

6. 类图: 描述类的构成以及类之间存在的静态关系的一种工具。

7. 采用 UML 进行软件建模过程中, 类图是系统的一种静态视图, 用 (聚合关系) 可明确表示两类事物之间存在的整体/部分形式的关联关系。

8. 若程序运行时系统报告除数为 0, 这属于 (语义) 错误。

9. 可维护性质量特性是指与软件维护的难易程度相关的一组软件属性, 包含了易分析性、稳定性、易测试性和易改变性 4 个子特性。

易分析性: 描述诊断缺陷或失效原因、判定待修改程度的难易程度的特性。

稳定性: 描述修改造成难以预料的结果的风险程度, 风险程度越低, 稳定性越好。

易测试性: 描述测试已修改软件的难易程度的特性。

易改变性: 描述测试及修改、纠错或适应环境变化的难易程度。

10. 在 UNIX 操作系统中, 把输入/输出设备看作是 (特殊文件)

12. 如果 DNS 服务器更新了某域名的 IP 地址, 造成客户端无法解析故障, 在客户端可以用两种方式解决此问题, 其中一种是 Window 命令行下执行 (ipconfig /flushdns)

13. 震网，指一种蠕虫病毒。它的复杂程度远超一般电脑黑客的能力。这种病毒于2010年6月首次被检测出来，是第一个专门定向攻击真实世界中基础（能源）设施的“蠕虫”病毒，比如核电站、水坝、国家电网。

14. 引导区病毒破坏引导盘、文件目录等

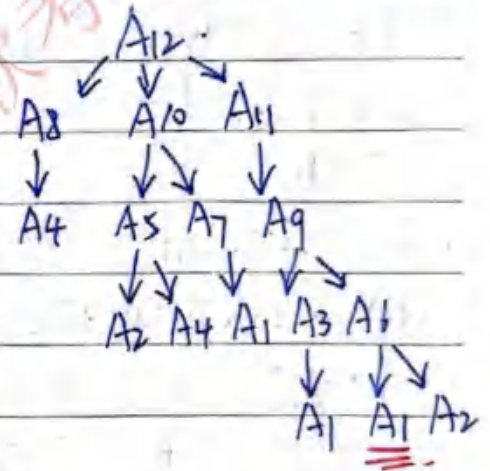
15. 宏病毒破坏 OFFICE 文件相关。

16. 木马的作用一般强调远程控制操作。

17. 某项目的活动持续时间及其依赖关系如下表所示，则完成该项目的最少时间为（55）天。

活动	持续时间(天)	依赖关系
A ₁	8	-
A ₂	15	-
A ₃	15	A ₁
A ₄	10	-
A ₅	10	A ₂ , A ₄
A ₆	5	A ₁ , A ₂
A ₇	20	A ₁
A ₈	25	A ₄
A ₉	15	A ₃ , A ₆
A ₁₀	15	A ₅ , A ₇
A ₁₁	7	A ₉
A ₁₂	10	A ₈ , A ₁₀ , A ₁₁

倒推法求关键路径 → 最大路径



$$10 + 7 + 15 + 15 + 8 = 55$$

操作系统

1. 某系统中1台打印机，N个进程共享打印机资源，每个进程要求3台。当N的取值不超过(5)时，系统不会发生死锁。

资源分配 $(3-1) \times N + 1 \leq 11$

$$N \leq 5$$

2. 某操作系统采用分页存储管理方式，下图给出了进程A和进程B的页表结构。如果物理页的大小为512字节，那么进程A逻辑地址为1111(十进制)的变量存放在(4)号物理内存中。假设进程A的逻辑页4与进程B的逻辑页5要共享物理页8，那么应该在进程A页表的逻辑页4和进程B页表的逻辑页5对应的物理页处分别填8。

进程A页表		进程B页表		物理页
逻辑页	物理页	逻辑页	物理页	
0	0	0	1	0
1	2	1	3	1
2	4	2	5	2
3	6	3	7	3
4		4	2	...
5		5		8
				9

$$(1111)_{10} = (1000101111)_2$$

2 → 4

3. SCAN调度算法也叫“电梯”算法，磁头固定从外向内转，后从内向外沿柱面运动。如此往复，遇到所请求的柱面时立即为其服务。

磁盘的读取过程是先垂直移动寻道——即查找对应磁道(柱面号)，然后再旋转寻道——即寻找扇区。(不考虑磁头号)

4. 进程的状态

进程控制块PCB的组织方式有①线性表式②索引表式③链接表式

①线性表式：不论进程的状态如何，将所有的PCB连续地存放在内存的系统区。这种方式适用于系统中进程数目不多的情况。

②索引表式：该方式是线性表式的改进，系统按照进程的状态分别建立就

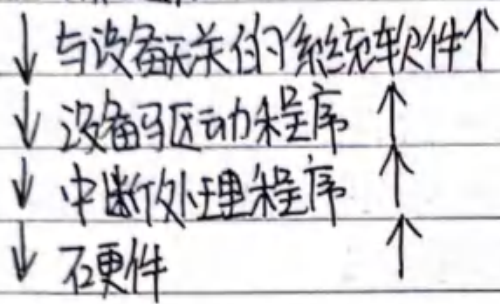
就绪队列、阻塞队列等。

③链接表式：系统按照进程的状态，将进程的PCB组成队列，从而形成就绪队列、阻塞队列、运行队列等

I/O管理软件

一般分为4个层次

I/O请求 → 用户进程



Spooling
I/O应答 产生I/O请求、格式化I/O、

映射、保护、分块、缓冲、分配
设置设备寄存器、检查状态、
执行I/O操作

免费题库 - 微信搜索：软考达人

计算机组成原理

1. 以下属于CPU中算术逻辑单元的部件是 (B) **指令计数器/单元 ACU**
A. 程序计数器 B. 加法器 C. 指令寄存器 D. 指令译码器
2. 假设磁盘块与缓冲区大小相同, 每个盘块读入缓冲区的时间为16us, 由缓冲区送至用户区的时间是5us, 在用户区内系统对每块数据的处理时间为1us。若用户需将大小为10个磁盘块D0C1文件逐块从磁盘读入缓冲区, 并送至用户区进行处理, 那么利用单缓冲区需要花费的时间为 (211) us; 采用双缓冲区需要花费的时间为 (166) us。

$$(21+1) + 9 \times 21 = 211$$

$$(16+5+1) + 9 \times 16 = 166$$

↑ 局限性原理 { 空间, 时间
3. 计算机的存储系统采用分级存储体系的理论依据是 (程序访问的局部性)。目前计算机中常用的三级存储体系是 (Cache、主存、辅存)。
4. 在程序执行过程中, Cache与主存的地址映像由 (硬件自动完成)。
5. 计算机运行过程中, 进行中断处理时需保存现场, 其目的是 (能正确返回到中断的程序继续执行)。
6. 若某计算机采用8位整数补码表示数对, 则运算 $(127+1)$ 将产生溢出。
范围 $-128 \sim 127$
7. 在计算机系统中常用补码表示和运算数据, 原因是采用补码可以 (简化计算机运算部件的设计)。
8. 两个同符号的数相加或异符号的数相减, 所得结果的符号位SF和进位标志CF进行 (异或) 运算为1时, 表示运算的结果产生溢出。
表达式: $VF = SF \oplus CF$
9. 组成内存存储器的芯片数量 = 内存存储器的容量 / 单个芯片的容量。
10. (C) 不属于按寻址方式划分的一类存储器。
A. 随机存储器 B. 顺序存储器 C. 相联存储器 D. 直接存储器
按存储器内容来存储和访问
11. 编写汇编语言程序时, 程序员可访问的是 **程序计数器 PC**。
12. VLIW 是超长指令字的缩写。

1. 数据的物理独立性和逻辑独立性分别是通过修改(模式与内模式之间的映像, 外模式与模式之间的映像)来完成的。
2. 采用级结构/两级映像的数据库体系结构, 如果对数据库的一张表创建聚簇索引, 改变的是数据库的(内模式)。
3. 测试严格按照计划进行, 避免测试的随意性。
4. 在数据库逻辑结构设计阶段, 需要(需求分析)阶段形成的(需求说明文档、数据字典和数据流程图)。
5. **需求分析**: 分析用户的需求, 包括数据、功能和性能需求, 得到数据流图、数据字典和需求说明书。
6. **概念设计**: 用数据模型明确地表示用户的数据需求。其反映了用户的现实工作环境, 与数据库的具体实现技术无关。
7. **逻辑设计**: 根据概念数据模型所具有的数据模型特性, 按照一定的转换规则和规范化理论, 把概念模型转换为逻辑数据模型, 如层次模型、网状模型、关系模型等。关系规范化是在逻辑设计阶段进行。
8. 在数据库逻辑设计阶段, 若实体中存在多值属性, 那么将E-R图转换为关系模式时, 将实体的码分解和每个多值属性独立构成一个关系模式, 得到的关系模式属于4NF。
9. **遗传算法**利用模拟生物进化的三个基本过程“繁殖(选择)→交叉(重组)→变异(突变)。
10. SQL提供可为关系和属性重新命名的机制, 这是通用具有“Old-name as new-name”形成的as子句来实现。As子句既可以出现在select子句, 也可以出现在from子句中。

数据结构

1. 采用单向循环链表存储的特点之一是从表中任意结点出发都能遍历整个链表便于元素的删除与插入。
2. 如需对表中的任意节点进行随机访问需采用顺序存储结构。
3. 白盒测试技术中, 路径覆盖法往往能比语句覆盖法发现更多的错误。
4. 连通无向网的最小生成树中, 顶点数恰好比边数多1。
5. (插入排序)在其最好情况下的算法时间复杂度为 $O(n)$ 。
6. 归并排序, 堆排序算法的时间复杂度为 $O(n \lg n)$ 。
7. 字符串的长度是指串中所含字符的个数。
8. 散列表(哈希)装填因子 α 表示哈希表的装满程度, 显然, α 越大发生冲突的可能性越大。



六 软考题库 - 微信搜一搜

Q 软考达人