# DMA（Direct Memory Access）

直接内存存取，是所有现代电脑的特色，它允许不同速度的硬件装置来沟通，而不需要依赖于cpu的大量终端负载。否则，cpu需要从来源把每一片段的资料复制到暂存器，然后把他们再次写回到新的地方。在这个时间中，cpu对于其他的工作来说就无法使用。

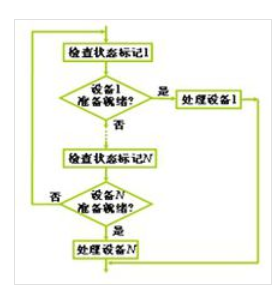
# 程序查询方式

程序查询方式是一种程序直接控制方式，这是主机与外设间进行信息交换的最简单的方式，输入和输出完全是通过CPU执行程序来完成的。

一旦某一外设被选中并启动后，主机将查询这个外设额某些状态，看其是否准备就绪？若外设未准备就绪，主机将再次查询；若外设已准备就绪，则执行一次I/O操作。

这种控制方式很简单，但外设和主机不能同时工作，各外设之间也不能同时工作，系统效率很低，因此，仅适用于外设的数目不多，对I/O处理的实时要求不怎么高，CPU的操作任务比较单一，并不很忙的情况。

多个设备工作流程如下：



# 程序中断方式

程序中断是指计算机执行现行程序的过程中，出现某些急需处理的异常情况和特殊请求，CPU暂时终止现行程序，而转去对随机发生的更紧迫的事件进行处理，在处理完毕后，CPU将自动返回原来的程序继续执行。

当主机启动外设后，无需等待查询，而是继续执行原来的程序，外设在做好输入输出准备时，向主机放出中断请求，主机借到请求后就暂时终止原来执行的程序，转去执行中断服务程序对外部请求进行处理，在中断处理完毕后返回原来的程序继续执行。显然，程序中断不仅适用于外部设备的输入输出操作，也适用于对外界发生的随机事件的处理。

程序中断在信息交换方式中处于最重要的地位，它不仅允许主机和外设同时并行工作，并且允许一台主机管理多台外设，使他们同时工作。但是完成一次程序中断还需要许多辅助操作，当外设数目较多时，中断请求过分频繁可能使CPU应接不暇；另外，对于一些高速外设，由于信息交换是成批的，如果处理不及时，可能会造成信息丢失，因此，它主要适用于中、低速外设。

程序中断与调用子程序的区别：

子程序的执行是由程序员实现安排好的，而中断服务程序的执行则是由随机中断事件引起的；

子程序的执行受到主程序或上层子程序的控制，而中断服务程序一般与被中断的现行程序毫无关系；

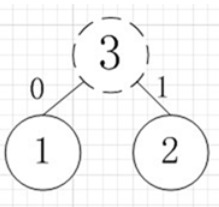
不存在同时调用多个子程序的情况，而有可能发生多个外设同时请求CPU为自己服务的情况。

# 哈夫曼编码（霍夫曼编码）Huffman Coding

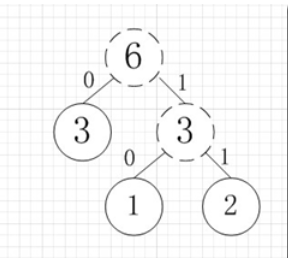
哈夫曼编码（huffman coding），又称霍夫曼编码，是一种编码方式，可变长编码（VLC）的一种。Huffman于1952年提出的一种编码方法，该方法完全依据字符出现概率来构造异字头的平均长度最短的码字，有时称之为最佳编码，一般就叫做Huffman编码）。

哈夫曼编码，主要目的是根据使用频率来最大化节省字符（编码）的存储空间。

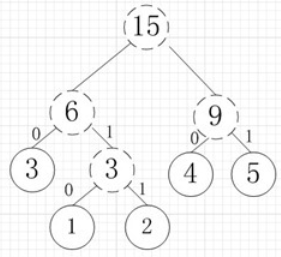
简易的理解就是，假如我有ABCDE五个字符，出现的频率（即权值）分别为：5，4，3，2，1，那么我们第一步先取两个最小权值作为左右子树构造一个新树，即取1，2构成新树，其节点为1+2=3，如图：



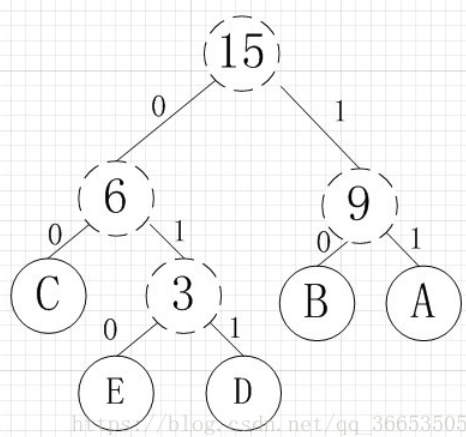
虚线为新生节点，第二步再把新生成的权值为3的节点放到剩下的集合中，所以集合变成{5，4，3，3}，在根据第一步，取最小的两个权值构成新树，如图：



再依次建立哈夫曼树，如下图：



其中各个权值替换对应的字符即为下图：



所以各字符对应的编码为：A->11,B->10,C->00,D->011,E->010

霍夫曼编码是一种无前缀编码。解码时不会混淆。其主要应用在数据压缩，加密解密等场合。

如果考虑到进一步节省存储空间，就应该将出现概率大（占比多）的字符用尽量少的0-1进行编码，也就是更靠近根（节点少），这也就是最优二叉树-哈夫曼树

# 邮件服务

25端口（SMTP）：25端口为SMTP（Simple Mail Transfer Protocol，简单邮件传输协议）服务所开放的，是用于发送邮件。如今绝大多数邮件服务器都使用该协议。当你给别人发送邮件时，你的机器的某个动态端口（大于1024）就会与邮件服务器的25号端口建立一个连接，你发送的邮件就会通过这个连接传送邮件到邮件服务器上，保存起来。

109端口（POP2）：109端口是为POP2（Post Office Protocol Version 2，邮局协议2）服务开放的，是用于接收邮件的。

110端口（POP3）：110端口是为POP3（Post Office Protocol Version 3，邮局协议3）服务开放的，是用于接收邮件的。

143端口（IMAP）：143端口是为IMAP（Internet Message Access Protocol）服务开放的，是用于接收邮件的。

目前POP3使用的比POP2广的多，POP2几乎被淘汰，也有某些服务器同时支持POP2和POP3协议。客户端可以使用POP3协议来访问服务端的邮件服务，如今ISP的绝大多数邮件服务器都是使用POP3协议（极少用POP2协议）。在使用邮件客户端程序的时候，会要求输入POP3服务器地址，默认情况下使用的就是110端口。当你使用邮件客户端（比如:Thunderbird,foxmail，MS Outlook Express以及各类 邮件精灵）登录时，你的机器就会自动用某一个动态端口（大于1024）连接邮件服务器的110端口，服务器就把别人发给你的邮件（之前保存在邮件服务器上），发送到你的机器，这样牛就可以看到新邮件了。

IMAP协议，和POP3协议一样是用来接收邮件的，但是他又他的特别和新颖之处，它是面向用户的，它和POP3协议的主要区别是：用户可以不把所有的邮件内容全部下载，而是只下载邮件标题和发件人等基本信息，用户可以由标题等基本信息，去决定是否下载邮件全文，用户可以通过客户端的浏览器直接对服务器上的邮件进行操作（比如：打开阅读全文、丢经垃圾箱 、永久删除、整理到某文件夹下、归档）。再简单来说就是：浏览器用的IMAP协议（143端口）来为你接收邮件以及让你很方便的操作服务器上的邮件。邮件客户端用的POP3协议（110端口）来为你接收邮件的全部信息和全文内容保存到你本地机器成为一个副本，你对邮件客户端上的副本邮件的任意操作都是在副本上，不干涉邮件服务器上为你保存的邮件原本。

SMTP，POP2，POP3，IMAP协议都是不安全的协议。因考虑到网络安全的因素，下面介绍基于SSL（Secure Sockets Layer安全套接层）协议的安全的邮件收发协议。你的邮件在传输过程中可能被网络黑客截取邮件内容，如果你的邮件机密性非常强，不想被收件人以外的任何人和任何黑客截取，或者是涉及国家机密安全的，等待。那么你的邮件就不该用上述的协议进行收发。

465端口（SMTPS）：465端口是为SMTPS（SMTP-over-SSL）协议服务开放的，这是SMTP协议基于SSL安全协议之上的一种变种协议，他继承了SSL非对称加密的高度安全可靠性，可防止邮件泄露。SMTPS的SMTP协议一样，也是用来发送邮件的，只是更安全些，防止邮件被黑客截取泄露，还可实现邮件发送者抗抵赖功能。防止发送者发送之后删除已发邮件，拒不承认发送过这样一份邮件。

995端口（POP3S）：995端口是为POP3S（POP3-over-SSL）协议服务开放的，这是POP3协议基于SSL安全协议之上的一种变种协议，它继承了SSL安全协议的非对称加密的高度安全可靠性，可防止邮件泄露。POP3S的POP3协议一样，也是用来发送邮件的，只是更安全些，防止邮件被黑客截取泄露，还可实现邮件发送者抗抵赖功能。防止发送者发送之后删除已发邮件，拒不承认发送过这样一份邮件。

993端口（IMAPS）：993端口是为IMAPS（IMAP-over-SSL）协议服务开放的，这是IMAP协议基于SSL安全协议之上的一种变种协议，它继承了SSL安全协议的非对称加密的高度安全可靠性，可防止邮件泄露。IMAPS的IMAP协议一样，也是用来发送邮件的，只是更安全些，防止邮件被黑客截取泄露，还可实现邮件发送者抗抵赖功能。防止发送者发送之后删除已发邮件，拒不承认发送过这样一份邮件。

# TCP和 UDP的优缺点

Tcp的优点：可靠、稳定，TCP的可靠体现在TCP传递数据之前会有三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传完后，还会断开连接用来节约系统资源。

TCP的缺点：慢、效率低、占用系统资源高，易被攻击。TCP在传输数据之前，要先建立连接，这会消耗时间，而且在数据传递时，确认机制、重传机制、拥塞控制机制等都会消耗大量的时间，而且要在每台设备上维护所有的传输连接，事实上，每个连接都会占用系统的CPU、内存等硬件资源。而且，因为TCP有确认机制、三次握手机制，这些也导致TCP容易被人利用，实现DOS、DDOS、CC等攻击。

UDP优点：快，比TCP稍安全，UDP没有TCP的握手、确认、窗口、重传、拥塞控制等机制，UDP是一个无状态的传输协议，所以他在传递数据时非常快。没有TCP的这些机制，UDP较TCP被攻击者利用的漏洞就要少一些。但UDP也是无法避免攻击的，比如：UDP Flood攻击…

UDP的缺点：不可靠，不稳定，因为UDP没有TCP那些可靠的机制，在数据传递时，如果网络质量不好，就会很容易丢包。

# 网络攻击

网络攻击（Cyberattack，也译为赛博攻击）是指针对计算机信息系统、基础设施、计算机网络或个人计算机设备的，任何类型的进攻动作。

于计算机和计算机网络中，破坏、揭露、修改、使软件或服务失去功能、在没有得到授权的情况下偷取或访问任何一台计算机的数据，都会被视为于计算机和计算机网络中的攻击。

主动攻击：主动攻击会导致某些数据流的篡改和虚假数据流的产生。这类攻击可分为篡改、伪造消息数据和中断（拒绝服务）。

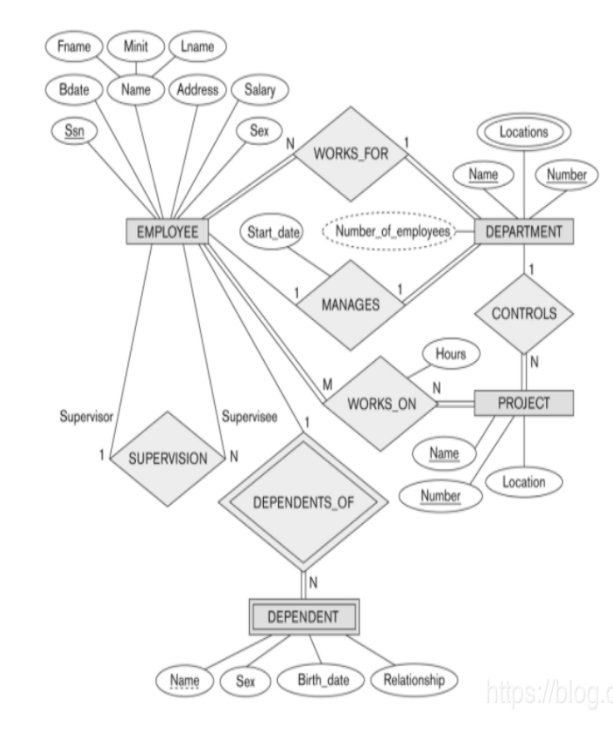
被动攻击：被动攻击中攻击者不对数据做任何修改，窃取/窃听是指在未经用户同意和认可的情况下攻击者获得了信息或相关数据。通常包括窃听、流量分析、破解弱加密的数据流等攻击方式。

# DFD（数据流图）

数据流图（Data Flow Diagram）；简称DFD，他从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变化过程，是结构化系统分析方法的主要表达工具及用于软件模型的一种图示方法。

# 实体关系图（ERD）

我们来看一个公司数据库的例子



实体关系模型，也称为实体关系（ER）图，是实体（将称为您的表）及相互关系的图形表示。数据库建模是创建数据模型的过程。

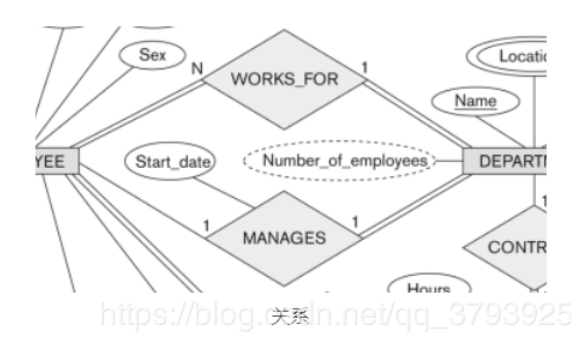
## 实体

一个简单的矩形代表一张表。



## 关系

它使用钻石形状勾勒出来



您可以绘制关系的类型，无论是使用1-M的一对多，还是使用M-N的多对多，或使用1-1的一对一。

连接具有关系的实体的线是单线还是双线引用另一个约束，称为存在依赖性约束（也称为“参与约束”）

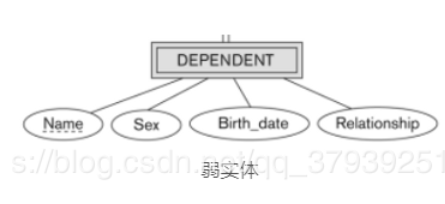
因此，如果一名员工必须为一个部门工作，这由双线勾画，并称为“全部或强制参与”。如果一名员工可能会或可能不会管理某个部门，则可以通过单行描述并称之为“部分或可选参与”。

## 递归关系

这是与同一实体的关系。例如，员工可以担任一个或多个员工的主管，而员工可以由员工监督（这是一对多的关系）。

## 弱实体

弱实体只是一个实体，他的存在依赖另一个实体。在缺少employee表的情况下，您无法在逻辑上拥有依赖（儿子、女儿…）他的轮廓与普通实体相同，但有双线。



# 需求分析阶段的任务

对现实世界要处理的对象（组织、部门、企业等）进行详细调查，在了解现行系统的概况、确定新系统功能的过程中，确定系统边界、收集支持系统目标的基础数据及处理方法。

# 逻辑设计阶段的任务

逻辑设计阶段的任务之一是对关系模式进一步的规范化处理。因为生成的初始关系模式并不能完全符合要求，会有数据冗余、更新异常存在，这就需要根据规范化理论对关系模式进行分解，以消除冗余和更新异常。不过有时根据处理要求，可能还需要增加部分冗余以满足处理要求。逻辑设计阶段的任务就需要做部分关系模式的处理，分解、合并或增加冗余属性，提高存储效率和处理效率。

# 主属性定义

“包含在任何一个候选码中的属性叫做主属性（Prime Attribute），否则叫做非主属性（Nonprime attribute）。

# BIOS

Bios（基本输入输出徐通）是一个程序，是固化在主板上的ROM芯片里面的，它相当于硬件底层的一个操作系统，控制和保存着硬件的输入输出工作。

# 奇偶校验

奇校验：如果以二进制数据中的1的个数是奇数为依据，则是奇校验

偶校验：如果以二进制数据中的1的个数是偶数为依据，则是偶校验

如果传输过程中包括校验位在内的奇数个数据位发生改变，那么奇校验位将出错表示传输过程有错误发生，但是由于没有办法确定哪一位出错，所以不能进行错误校正。

# 浮点数对阶

两个浮点数对阶的时候要把阶码小的数的尾数右移n位，与阶码大的对齐。

# 上下文无关文法

形式语言理论中一种重要的变换文法，用来描述上下文无关语言，在乔姆斯基分层中称为2型文法。由于程序设计语言的语法基本上都是上下文无关文法，因此应用十分广泛。

# 超长指令字

VLIW（very long instruction word）超长指令字，一种非常长的指令组合，他把许多指令连在一起，增加了运算的速度。

# 解释方式和编译方式

解释程序也称为解释器，它或者直接解释执行源程序，或者将源程序翻译成某种中间表示形式后再加以执行。

编译程序（编译器）则是将源程序翻译成目标语言程序，然后在计算机上运行目标程序。

两种语言处理程序的根本区别是：在编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程，而在解释方式下，解释程序和源程序（或某种等价表示）要参与到程序的运行过程中，运行程序的控制权在解释程序。

解释器翻译源程序时不生成独立的目标程序，而编译器则将源程序翻译成独立的目标程序。

# 词法分析

词法分析是编译或解释用户源程序过程中唯一与源程序打交道的阶段，其主要功能是按顺序分析出源程序的记号。

# 主要的程序语言

Fortran语言：第一个高级程序设计语言，科学计算，执行效率高

Pascal语言：结构化程序设计语言，表达能力强，Delphi

C语言：通用、结构化程序设计语言，指针操作能力强，高效

Lisp语言：函数式程序语言，符号处理，人工智能

C++语言：C语言基础上增加了类机制，面向对象，高效

Java语言：面向对象，中间代码，跨平台，通用的程序设计语言

Python语言：面向对象，解释型程序设计语言，通用的脚本语言

PHP：服务端脚本语言，制作动态页面

Ruby：简单快捷，面向对象，脚本语言

Delphi：快速应用程序开发工具，可视化编程环境

Cobol：数据处理领域最为广泛的程序设计语言，高级编程语言

Prolog：逻辑式语言，建造专家系统、自然语言理解、智能知识库等

# 常用的中间代码

常用的中间代码的表达形式有语法树、后缀式、三地址代码。

# 指令集

1. 指令系统：RISC设计者把主要精力放在那些经常使用的指令上，尽量使他们具有简单高效的特色。对不经常用的功能，常通过组合指令来完成。而CISC计算机的指令系统比较丰富，有专用的指令来完成特定的功能。因此，处理特殊任务效率较高。
2. 存储器操作：RISC对存储器操作有限制，使控制简单化；而CISC机器的存储器操作指令多，操作直接。
3. 程序：RISC汇编语言程序一般需要较大的内存空间，实现特殊功能时程序复杂，不易设计；而CISC汇编语言程序编程相对简单，科学计算及复杂操作的程序设计相对容易，效率较高。
4. 设计周期：RISC微处理器结构简单，布局紧凑，设计周期短，且易于采用最新技术；CISC微处理器结构复杂设计周期长。
5. 应用范围：由于RISC指令系统的确定与特定应用领域有关，故RISC机器更适合于专用机；而CISC机器则更适合于通用机。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令系统类型 | 指令 | 寻址方式 | 实现方式 | 其他 |
| CISC | 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 |  |
| RISC | 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，采用流水线 | 优化编译，有效支持高级语言 |

# 表示媒体

表示媒体指的是为了传输感觉媒体而认为研究出来的媒体，借助于此种媒体，能有效地存储感觉媒体或将感觉媒体从一个地方传送到另一个地方。如：语言编码、电报码、条形码等。

# 表现媒体

是指进行信息输入和输出的媒体，如：键盘、鼠标、扫描仪、话筒和摄像机等输入媒体以及显示器、打印机和扬声器等输出媒体。

# 感觉媒体

是指直接作用于人的感觉器官，使人产生直接感觉的媒体，如：引起听觉反应得声音，引起视觉反应得文本、图形和图像等。

# 存储媒体

存储媒体是指用于存储表示媒体得物理介质，如：硬盘、软盘、关盘和胶卷等。

# 传输媒体

是指用于传输表示媒体得物理介质，如：电缆和光缆等。

# 光盘存储技术

DVD-RAM和DVD-RW是DVD技术所支持的两种不同的可多次插除重写的DVD光盘 格式，CD-R指一次性可写（刻录）CD光盘，而CD-RW指多次可擦除、重写的CD光盘。

# 程序语义

分为静态语义和动态语义，其中静态语义分析方法是语法制导翻译，其基本思想是将语言结构的语义以属性的形式赋予代表此结构的文法符号，而属性的计算以语义规则的形式赋予文法的产生式。

# 色彩

亮度是指发光体（反光体）表面发光（反光）强弱的物理量。

色调指的是一幅画中画面色彩的总体倾向，是大的色彩效果。在大自然中，我们经常见到这样一种现象：不同颜色的物体或被笼罩在一片金色的阳光之中，或被笼罩在一片轻纱薄雾似的、淡蓝色的月色之中；或被秋天迷人的金黄色所笼罩；或被统一在冬季银白色的世界之中。这种在不同颜色的物体上，笼罩者某一种色彩，使不同颜色的物体都带有同一种色彩的倾向，这样的色彩现象就是色调。

饱和度是指色彩的鲜艳程度，也称色彩的纯度。

# 海明码

海明码是利用奇偶性来检错和纠错的校验方法，码距最小为2n+1。

# 主存与cache地址映射

全相联地址映射：主存的任意一块可以映射到cache中的任意一块。

直接相联映射：主存中的一块只能映像到cache的一个特定块中。

组相联映射：各区中的某一块只能存入缓存的同组号的空间内，但组内各块地址之间则可以任意存放。即从主存的组到cache的组之间采用直接映射，在两个对应的组内部采用全相联映射。

# 累加寄存器AC

通用寄存器，为ALU（算术逻辑单元）提供一个工作区，用来暂存数据。

# CPU

指令寄存器（IR）用来存放从内存中读取的指令；累加器（累加寄存器AC），通用寄存器，可用于传送和暂存数据，也可参与算术逻辑运算，并保存运算结果；程序计数器（PC）是用于存放下一条指令所在单元的地址，当执行一条指令时，首先需要根据PC中存放的指令地址，将指令由内存取到指令寄存器中，即 将程序计数器PC中的内容送到地址总线上，此过程称为“取指令”，与此同时，PC中的地址或自动加1或由转移指针给出下一条指令的地址，此后经过分析指令，执行指令，完成第一条指令的执行，而后根据PC取出第二条指令的地址，如此循环，执行每一条指令；状态寄存器（PSW）用于存放状态标志与控制标志如：中断标志，溢出标志等。

# 编译过程

一般分为5个阶段：词法分析、语法分析、语义分析、优化、目标代码生成。

1. 词法分析：也就是从左到右一个一个读入源程序，识别一个单词或符号，并进行归类
2. 语法分析：在词法分析的基础上，将单词序列分解成各类语法短语，如：程序、语句、表达式等
3. 语义分析：审查源程序是否由语义错误，当不符合语言规范的时候，程序就会报错
4. 代码优化：这个阶段是对前阶段的中间代码进行变换或改造，目的是使生成的目标代码更为高效，即节省时间和空间
5. 目标代码生成：也就是把优化后 的中间代码变化成指令代码或汇编代码

# 死循环

死循环错误属于典型的语义错误，但静态的语义错误可被编译器发现，到程序真正陷入死循环说明编译器并未发现，所以属于动态语义错误。

# 各种码制下带符号的数的表示范围

|  |  |  |
| --- | --- | --- |
| 码制 | 定点整数 | 定点小数 |
| 原码 | -((2^(n-1))-1)~(2^(n-1))-1 | -(1-2^(-(n-1)))~(1-2^(-(n-1))) |
| 反码 | -((2^(n-1))-1)~(2^(n-1))-1 | -(1-2^(-(n-1)))~(1-2^(-(n-1))) |
| 补码 | -2^(n-1)~(2^(n-1))+1 | -1~(1+2^(-(n-1))) |
| 移码 | -2^(n-1)~(2^(n-1))-1 | -1~(1+2^(-(n-1))) |

# 内存分配

一个由c/c++编译的程序占用内存分为以下几个部分：

1. 栈区（stack）：由编译器自动分配释放，存放函数的参数值 ，局部变量的值等。其操作方式类似于数据结构中的栈。
2. 堆区（heap）：一般由程序员分配释放，若程序员不释放，程序结束时可能由OS回收。注意它与数据结构中的堆是两回事，链表的数据空间必须采用堆存储分配策略。
3. 全局区（静态区）（static）：全局变量和静态变量的存储是放在这一块的。初始化的全局变量和静态变量在一块区域，未初始化的全局变量和未初始化的静态变量在相邻的另一块区域，程序结束后由系统释放。
4. 文字常量区：常量字符串就是放在这里的，程序结束后由系统释放。
5. 程序代码区：存放函数体的二进制代码。

# 存储器

SRAM静态的随机存储器：特点是工作速度快，只要电源不撤除，写入SRAM的信息就不会消失，不需要刷新电路，同时在读出时不破坏原来存放的信息，一经写入可多次读出，但集成度较低，功耗较大。SRAM一般用来作为计算机中的高速缓冲存储器（Cache）。

DRAM只能将数据保持很短的时间。为了保持数据，DRAM使用电容存储，所以必须隔一段时间刷新（refresh）一次，如果存储单元没有被刷新，存储的信息就会丢失。关机就会丢失数据。

# 遗传算法

遗传算法（Genetic Algorithm）是模拟达尔文生物进化论的自然选择仿遗传学机理的生物进化过程的计算模型，是一种 通过模拟自然进化过程搜索最优解的方法。遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，而一个种群则由经过基因（gene）编码的一定数目的个体组成。每个个体实际上是染色体带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，他决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。

遗传算法的基本运算过程如下：

1. 初始化
2. 个体评价
3. 选择运算
4. 交叉运算
5. 变异运算

# 队列和栈的应用

栈：表达式求值、括号匹配、递归

队列：打印队列

# 链表

单向链表仅设头指针时，在表尾插入节点时需要遍历整个链表，时间复杂度为O（n），仅设表尾指针时，在表尾插入节点的时间复杂度为O（1），但是不能访问除了尾节点之外的所有其他节点。

单向循环链表仅设头指针时，在表尾插入节点时需要遍历整个链表，时间复杂度为O（n），仅设表尾指针时，在表尾插入节点的时间复杂度为O（1），同时达到表头节点的时间复杂度为O（1）。

# 二叉树

二叉树采用二叉链表存储时空指针个数为节点数+1，采用 三叉链表存储时空指针个数为节点数+2。