2.计算机软件

2.1 软件基本概念

1、**计算机系统**由**硬件系统**和**软件系统**两部分构成。

（1）**硬件系统:运算器、控制器、存储器、输入设备、输出设备**等五大部件

（2）**软件系统：系统软件和应用软件**。

①、系统软件：操作系统、数据库管理系统、程序设计语言处理系统等

②、应用软件：文字处理软件、通信软件、图像处理软件等

2 软件：

（1）软件=程序+数据+相关文档。

（2）程序就是要计算机做什么或完成什么功能的一组指令（语句），这些指令能够被计算机理解并执行的命令。

（3）数据：就是计算机执行程序过程中的输入数据和输出数据。

（4）相关文档就是程序开发、维护和操作有关的资料，不可缺少。

（5）程序的特性：

①、不可见性：是无形的，不能被人们直接观察、欣赏和评价。

②、适用性：可以适应一类应用问题的需要。

③、依附性：依附于特定的硬件、网络和其他软件。

④、复杂性：规模越来越大，人力和物力消耗越来越高。

⑤、无磨损性：功能和性能一般不会发生变化。

⑥、易复制性：可以非常容易且无失真地进行复制。

⑦、不断演化：有生命周期。

⑧、有限责任：具有有限保证。

⑨、脆弱性：容易遭受黑客、病毒、信息盗用等损害。

（6）程序与数据的关系：

①、程序所处理的对象和处理后得到的结果统称为数据（分别称为输入数据和输出数据）

②、程序需要有合理输入数据，才会产生有意义的输出数据

③、给同一个程序输入不同的数据，可以得到不同的输出数据

④、给不同的程序输入相同的数据，可以得到不同的输出数据

（7）软件的特点

①、软件是一种**逻辑**实体，具有抽象性；

②、软件的生产与硬件不同，它没有明显的制作过程；

③、软件在运行、使用期间不存在磨损、老化问题；

④、软件的开发、运行对计算机系统具有依赖性，受计算机系统的限制，这导致了软件移植的问题；

⑤、**软件复杂性高（固有有）**，成本昂贵；

⑥、软件开发涉及诸多的社会因素。

3 软件分类：

（1）按软件用途和功能作用分：

①、应用软件：专门用于帮助最终用户解决各种具体应用问题的软件。

可以分为**定制应用软件**和**通用应用软件**。

通用软件类型	软件举例
文字处理类	**记事本 Notepad（纯文本）、写字板 Wordpad、微软 Word、金山 WPS 文字、永中 Office、Acrobat Reader 等**
文字输入类	**智能 ABC、微软拼音、搜狗拼音、极品五笔、拼音加加、谷歌拼音**
电子表格类	**微软 Excel、WPS 表格（金山公司，国产）等。**
幻灯片演示类	**微软 PowerPoint、 WPS 演示（金山公司，国产）等**
网页制作类	FrontPage、Dreamweaver、Web Page Maker、ASP Maker
网页浏览类	微软 IE、傲游 Maxthon、腾讯 TT、Mozilla Firefox
搜索引擎类	Baidu、Google、北大天网、搜狐 Sogou、新浪爱问、网易有道、雅虎易搜
杀毒软件类	瑞星、诺顿、卡巴斯基、金山毒霸、江民 KV、趋势科技
软件防火墙类	瑞星个人防火墙、江民防火墙、360 防火墙、天网防火墙、风云防火墙
图像处理类	**Paint、Photo Editor、PhotoImpact、ACDSee32、Adobe Photoshop**
矢量绘图软件	**Corel Draw、Illustrator、Freehand、Microsoft Visio、Adobe Flash、3D Max**
数字视频编辑软件	**Adobe Premiere、Windows Movie Maker、Video Edit Magic、PowerDirector**
数字声音编辑软件	**CoolEdit、GoldWave、Adobe Audition、Nero wave Editor**
媒体播放软件	**暴风影音、Windows Media Player、RealPlayer、QuickTime、千千静听、Winamp**
网络聊天类	**腾讯 QQ、微软 MSN、网易 POPO、新浪 UC、移动飞信、阿里旺旺**
解/压缩类	WinRAR、WinZip、7-Zip、ChinaZip、Zipghost
下载工具类	迅雷 Thunder、网际快车 FlashGet、比特彗星 BitComet、电驴 emule、腾讯旋风
邮件收发类	Outlook Express、Foxmail、DreamMail、Koomail、U-Mail
网络电视类	PPlive、PPStream、QQLive、UUsee、TVKoo、CCTV Box、迅雷看看
FTP 工具类	Serv-U FTP、CuteFTP、TurboFTP、ChinaFTP、LeapFTP

②、系统软件：系统软件是给用户使用计算机提供方便、为应用软件的运行和开发提供服务与支持、使计算机安全可靠高效地运行的必不可少的软件

软件类型	**软件举例**
操作系统（OS）	**Windows、Unix、Linux、安卓、HarmonyOS（鸿蒙 OS）等**
基本输入/输出系统	**BIOS**
程序开发工具与环境	**编译器、解释器、汇编程序（汇编器）、开发工具与平台**
数据库管理系统(DBMS)	**MySQL、Oracle、Access、DB2、SQL Server、Sybase、VFP、PostgreSQL、NoSQL（大数据中常用：BigTable、Redis、MongoDB、Neo4J、Hazelcast）**
实用程序	**磁盘清理程序、备份程序、杀毒软件、防火墙等**

（2）**按知识产权分：**

①、商品软件：得先购买后使用，受到版权和许可证保护。
例如：Windows 操作系统等
②、共享软件（试用软件）：具有版权，可免费试用一段时间，允许拷贝和散发（但不可修改），试用期满后需交费才能继续使用
例如：微软 Visio 等
③、自由软件（开源软件）：用户可共享，并允许随意拷贝、修改其源代码，允许销售和自由传播。
例如：Linux(CentOS)、Linux(Ubuntu)等
④、免费软件：无需付费即可获得的软件。例如 PDF 阅读器、Flash 播放器、QQ、微信等
自由软件很多是免费软件；免费软件不全是自由软件

4 软件版本
（1）版权授予软件作者(版权所有者)享有下列权利：拷贝、发布、修改、署名、出售、等
（2）软件许可证也称为“许可证协议”，它规定了计算机软件使用方式的法律合同，软件使用有哪些额外的限制，有哪些额外的权利等
（3）许可证的类型：单用户许可证、多用户许可证、并发用户许可证、定点许可证等。
（4）用户购买了一个软件后，仅仅得到了该软件的使用权，并没有它的版权。
（5）通过购买多用户许可证，允许将一个软件安装在若干台计算机上使用，或者允许所安装的一份软件同时被若干用户使用。

2.2 操作系统

1、操作系统（OS）
（1）未安装操作系统的计算机（裸机），是无法使用的。
（2）操作系统是用于执行各种具有共性和基础性操作的软件（应用软件、程序语言处理程序、支持工具等都是运行在 OS 之上），是用于控制、管理和分配计算机所有资源的，是现在计算机运行配置中不可或缺系统软件。
（3）操作系统是计算机硬件与用户之间的接口，也可以认为它是计算机软件与硬件之间的接口。
（4）当操作系统存在安全隐患啊时，可能会威胁到软件或硬件的安全运行。所以操作系统的安全性是计算机系统安全性的的核心。
2、操作系统的启动过程：
加电自检程序 POST、引导装入程序（自举）BOOT、引导程序、运行操作系统
3、操作系统组成：
（1）操作系统内核（kernel）：内核是操作系统最基本的部分。调度 cpu 资源,管理进程和内存等。
①、微软公司的 NT 内核：Windows 10、Windows XP、Windows 7 等操作系统
②、Linux 内核（自由软件）：GNU（GNU's Not Unix，以 GPL 方式发布）/Linux （edHat、Ubuntu、CentOS）、安卓（Android）等操作系统
③、Darwin 内核（类 Unix 系统）：iOS 操作系统
④、鸿蒙微内核：鸿蒙操作系统
（2）其他配套软件：用户界面（GUI）、常用的应用程序（如日历、计算器、资源管理器、浏览器等）、实用程序(任务管理器、磁盘清理程序、杀毒软件、防火墙等)、支撑软件（如应用框架、编译器、程序库等）。
4、操作系统的三个主要作用：
（1）为运行 APP（应用程序）管理和分配软/硬件资源
（2）为用户提供友善的人机界面（图形用户界面）
（3）为开发 APP（应用程序）提供高效率的平台

5、**操作系统主要功能**

（1）多任务与处理器管理（进程管理）：CPU 资源分配。

（2）文件管理：对外存的管理，主要是创建（或保存）文件而分配存储空间，为删除文件而回收空间，对空闲空间进行分配 。通常采用文件目录来组织和管理外存中的信息。

（3）存储管理：对内存的管理，虚拟内存=物理内存+硬盘的虚拟接口。

（4）设备管理：对所有硬件设备管理，驱动程序是操作系统和硬件之间的接口。

6、多任务与处理器管理

（1）前台任务：能接受用户输入（击键或单击鼠标）的窗口只能有一个，称为活动窗口，它所对应的任务称为前台任务。前台任务只能有 1 个。

（2）并发多任务：允许计算机同时执行多个任务，任务是并发执行的，无论是前台任务，还是后台任务，都可以分配到 CPU 资源。

（3）多任务实现的原理是：操作系统将 CPU 时间划分为许多小时间片，轮流为多个程序服务。多个任务宏观上同时运行，微观上轮流运行。多任务处理并不一定需要多核 CPU 的硬件支撑。

（4）多任务的目的：提高 CPU 的利用率，提高用户的工作效率。

（6）**现在流行的 Windows、Linux、Android、IOS、HarmonyOS 都支持多任务处理**

7、**文件管理**:

（1）**文件的概念。 文件是存储在外存储器中的一组相关信息的集合。**

（2）文件名的命名原则。

①、文件名的组成：主文件名.扩展名。主文件名不可以省略，扩展名可以省略。

②、主文件名中不可以出现"、、?、*、\、/、、<、>。

③、不可以使用系统保留的关键字 CON、PRN。

④、文件名最多有 255 个字符（中文或西文字符）。

⑤、Windows 中文件名不区分大小写。

⑥、文件名中可以使用空格，文件名开头不保留空格。

⑦、应用程序扩展名：.exe(Windows 中)、.app（iOS）、.apk（安卓）

⑧、常见文件扩展名：

文件类型	文件扩展名	文件类型	文件扩展名
文件文件	.txt	声音文件	.wav,.min,.voc,mp3
Word 文件	.doc、.docx	图形、图像文件	.bmp,.pcx,.tif,.wmf,jpg,.png.,git
Excel 文件	.els.elsx	动画/视频文件	.flc,.fli,avi,mp4,.flv.mpg
PPT 文件	.ppt,pptx	网页文件	.html,.htm

（3）文件的组成。

①、文件由文件说明信息和文件内容两部分组成。

②、文件说明信息存放在文件的目录中。

③、文件的内容存放在磁盘的数据区中。

（4）Windows 文件目录采用多层次树状结构。

（5）**操作系统的文件管理是指在外存中为创建文件或保存文件而分配外存空间，除文件而回收空间，并对空闲空间进行管理。**

8、**存储管理**

（1）内存储器空间划分为 2 个部分：系统区和用户区，用户区用来存放正在运行的应用程序

（2）**操作系统一般都采用虚拟存储器技术（也称虚拟内存技术，简称虚存）进行存储管理。**

（3）虚拟存储器 =物理内存 + 硬盘上的虚拟内存。

（4）虚拟存储器大小受到外存空间及 CPU 地址表示范围的限制。

（5）在 Windows 中，用户可设置虚拟内存位于哪个硬盘逻辑盘上及其容量。每个程序的虚存空间最大可达到 4GB。

（6）内存页面调度算法（LRU，最近最少使用）：在每次调换时，找到最近最少使用的那个页面调出内存。（7）在 Windows 操作系统上虚拟内存文件位于系统盘的系统根目录下的 pagefile.sys。

9、**设备管理**

（1）操作系统中的"设备管理"负责对用户或者应用程序的 IO 操作进行统一管理

（2）**设备管理通过驱动程序屏蔽和抽象了各种物理 I/O 设备的硬件操作细节。驱序程序是操作系统和硬件之间的接口。**

10、常用操作系统：

（1） **Windows。**

①、Windows 桌面版是单用户、多任务操作系统。 如 Windows XP，Windows 10 等

②、Windows 服务器版是多用户、多任务操作系统，如 Windows 2003 Sever。

（2）**UNX 操作系统。**

①、美国贝尔实验室开发的一种通用多用户交互式分时操作系统。

②、可移植性好，90%以上的代码用 C 语言编写。

③、网络中的服务器大部分安装 UNI。UNIX 也可以用于 PC、智能手机。

（3）**Linux 操作系统。**

①、Linux 是多用户、多任务、分时操作系统。

②、**Linux 是"自由软件"，其源代码向大众开放。**

③、Linux 系统最初是由一名芬兰大学生开发的，不是美国的研究机构。

(4) **HarmonyOS（鸿蒙 OS）**

①、HarmonyOS 是多用户、多任务、分布式操作系统。

②、HarmonyOS 可以适用于手机、车机、智能电视等设备（物联网）

③、HarmonyOS 是国产操作系统。

2.3 程序设计语言

1、**程序设计语言：主要用于描述算法，也称算法语言；**

2、**程序设计语言分类：机器语言（低级语言，计算机可以直接识别执行）、汇编语言（低级语言）、高级语言（如 C,C#等）**

3、**机器语言**

（1）机器语言就是计算机的指令系统，指令是使用二进制编码表示的。

（2）机器语言使用二进制代码编写，可以被计算机直接识别并执行，不需要翻译

（3）机器语言编写的程序记不住、难理解、编程效率低、不易维护，但是执行效率高

（4）机器语言对机器（硬件）有依赖性，移值性比较差。

（5）现在已经不直接用机器语言编写程序

（6）举例：在 MIPS 计算机上求最大公约数（GCD)的机器程序（16 进制表示）

```
27bdffd0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483fffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```

3、**汇编语言**

（1）汇编语言是用助记符来表示机器指令中的操作符号与操作数。

（2）汇编语言是面向机器的程序设计语言（对机器仍有依赖性），属于低级语言

（3）汇编语言程序操作数直接使用十进制表示，程序相对容易理解

（4）举例：

汇编语言程序示例

```
将 383 传送至 AX 寄存器
将 545 传送至 BX 寄存器
将 BX 内容加 AX 内容结果在 BX 中
将 1055 传送到 AX 寄存器
将 AX 内容减 BX 内容结果在 AX 中
```

4、**高级语言**

（1）高级语言（按近自然语言）容易理解、记忆和使用，降低了编程难度。

（2）高级语言克服汇编语言的缺陷，提高编程和维护效率

（3）高级语言程序移值性好，对硬件不太依赖

（4）高级语言对使用的符号、词汇、语法和语义等各种语言成分都有严格的规定

（5）举例：C 语言举例

```c
//返回类型为void，表示不返回任何值
void checkPrimeNumber(){
    int n, i, flag = 0;
    printf("输入一个正整数: ");
    scanf("%d",&n);

    for(i=2; i <= n/2; ++i){
        if(n%i == 0){
            flag = 1;
        }
    }
    if (flag == 1)
        printf("%d 不是质数。", n);
    else
        printf("%d 是个质数。", n);
}
```

5、**程序设计语言基本三要素：**

（1）语法：表示构成语言的各个记号之间的组合规律

（2）语义：表示程序的各个语法成分含义。语义是程序语言中按语法规则构成的各个语法成分的含义，可分为静态语义和动态语义。

（3）语用：程序与使用者之间关系，与运行环境和编译环境有很大关系。

6、**程序设计语言的四大成分**：

（1）**数据成分**：程序中所涉及的数据部分（变量、数组、结构体等即，数据名、数据类型、数据结构等）

（2）**运算成分**：程序中所包括的运算部分（算术运算、逻辑运算、关系运算等）

（3）**控制成分**：程序中控制构造（顺序结构、选择结构、循环结构等）

（4）**传输成分**：程序中数据的传输（数据交换[赋值语句]、拷贝（strcpy、memcpy）、I/O 操作(输入输出库函数调用)、函数传参与返回等）

7、**程序的控制结构**：

（1）**顺序控制结构**：最基本、最普通的结构形式，按照程序中的语句行的先后顺序逐条执行。

（2）**选择控制结构**：单分支选择（if）、双分支选择(if…else)、多分支持（if…elseif…/switch…case）

（3）**循环控制结构**：for、do-while(最少被执行一次)、while

8、结构化程序设计

（1）**结构化程序设计**是面向过程的程序设计方法，**按照模块划分原则以提高程序可读性和易维护性、可调性和可扩充性为目标的一种程序设计方法**。

（2）**结构化程序设计**主张**使用三种基本控制结构**(顺序、选择分支、循环)来嵌套连接成具有复杂层次的“结构化程序“，**使用得程序易读、易修改**。

（3）**结构化程序设计四条原则：自顶向下；逐步求精；模块化；限制使用 goto 语句。**

9、面向对象程序设计

（1）面向对象的程序设计以对象为核心

（2）**面向对象程序方法的三个重要特征：封装性，继承性和多态性。**

①、封装性：对象的封装实现了信息的隐藏

②、继承性：是使用已有的类定义作为基础建立新类的定义技术，广义指能够直接获得已有的性质和特征，而不必重复定义他们。继承分单继承（C#,Java）和多重继承(C++)。单继承指一个类只允许有一个父类，即类等级为树形结构；多重继承指一个类允许有多个父类。

③、多态性: 是指同样的消息被不同的对象接受时可导致完全不同的行动的现象。

（3）面向对象程序设计的优点：

①、人类习惯的思维方法一致；

②、稳定性好；

③、可重用性好；

④、易于开发大型软件产品；

⑤、可维护性好。

（4）**对象的三要素**：**属性、事件、方法**。

（5）消息及其组成：

①、消息：是一个实例与另一个实例之间传递的信息。对象间的通信靠消息传递。它请求对象执行某一处理或回答某一要求的信息，它统一了数据流和控制流。

②、消息的组成包括：接收消息的对象的名称；消息标识符，也称消息名；零个或多个参数。

（6）**属性，类和实例**：

①、属性：即对象所包含的信息，它在设计对象时确定，一般只能通过执行对象的操作来改变。

②、类：是具有相似属性与操作的一组对象。类是关于对象性质的描述。

③、实例：类是对象的抽象，对象是其对应类的一个实例。

10、程序设计语言处理

（1）源程序：使用高级语言和汇编语言编写的程序称为源程序。

（2）目标程序：源程序—>翻译程序—>目标程序

（3）翻译程序：将某一种语言翻译成另一种语言

①、汇编程序：汇编语言的翻译程序，属于系统软件

②、**高级语言的翻译程序：解释程序和编译程序，也属于系统软件**

11、**解释程序**

（1）解释：解释器直接解释并且执行源语言程序,不产生目标程序(相当于“口译”)

（2）解释程序.按源程序中语句的执行顺序，逐条翻译并立即执行相应的功能的处理顺序，解释不产生目标程序。

（3）算法简单，灵活，便于查错，占用内存少，运行效率低。

（4）适用于调试程序、交互程序等。

（5）**解释型语言：BASIC、Visual Basic、VBScript、Java、JavaScript、PHP、Python 等。**

12、**编译程序**

（1）编译：把源程序编译为机器语言目标程序后，再由计算机运行。（相当于“笔译”）

（2）编译程序：将程序从高级语言转换到机器语言或汇编语言的翻译程序。
（3）编译产生目标程序，目标程序可反复被执行，运行效率高。
（4）适用于翻译规模大、结构复杂、运行时间长的大型应用程序。
（5）编译程序的处理过程：编辑 ->编译 ->链接->运行
（6）编译型语言：C、C++、C#等。
13、**常用程序设计语言**
（1）FORTRAN：面向过程程序设计言，用于数值计算。
（2）Java：跨平台，适用于网络，面向对象程序设计语言，（单继承）。
（3）C 面向过程,多用于系统、底层软件的开发等，linux 内核是 C 写的。
（4）C++;面向对象（多继承）。
（5）C#;面向对象（单继承）。
（7）VB:面向对象
（8）BASIC 面向过程或结构化程序设计语言。
（9）人工智能语言主要有 LISP、Prolog、Smalltalk、C++等。
（10）脚本语言：VBA、JavaScript、VBScript、ActionScript、PHP 等

2.4 算法和数据结构

1、算法
（1）**程序=算法+数据结构**
（2）**算法是指解决问题的一种方法或一个过程**。是若干指令的有穷序列。
（3）算法的设计一般采用由粗到细、由抽象到具体的逐步求精的方法。
（4）**算法的必要满足条件（有些教材称为五大特性，最后一条拆为无输入和至少一输出）：**
①、确定性：算法的每一条指令必须有确切的定义，无二义性。
②、有穷性：是指算法必须在有限的时间内做完，即算法必须能在执行有限个步骤之后终止。
③、可行性：算法中的描述在计算机中能执行，且在有穷时间内完成。
④、至少有一个（1 个或多个）输出，但是可以没有（0 个或多个）输入
（5）算法的组成要素：一个算法由**数据对象的运算和操作**以及其**控制结构**这两部分组成。
（6）算法的基本运算和操作：**算术运算，逻辑运算，关系运算，数据传输**等。
（7）算法的基本控制结构包含：**顺序结构、选择结构、循环结构**。
（7）算法的基本设计方法：列举法、归纳法、递推、递归、减半递推技术。
2、**算法与程序的区别**
（1）算法是解决问题的方法、步骤
（2）程序是算法的具体代码实现
（3）算法是程序设计的核心，算法的好坏直接决定了程序的效率
（4）程序中的语句必须是机器可执行的，算法中的操作无此限制。
（5）算法与程序是相应的，但不是一一对应。
3、**算法的表示（描述工具）**
（1）自然语言：算法可以用自然语言来描述，缺点是不够准确。
（2）流程图
（3）程序设计语言
（4）伪代码（PDL）：介于自然语言和程序设计语言之间的文字和符号表达工具。
4、**评价算法的优劣应考虑的主要因素（算法的分析）**

算法分析是指对一个算法的运行时间和占用空间做定量的分析，一般计算出相应的数量级，常用时间复杂度和空间复杂度表示。**分析算法的目的就是要降低算法的时间复杂度和空间复杂度，提高算法的执行效率**。

（1）**算法的复杂度**：时间复杂度和空间复杂度

（2）**正确性**:算法的正确性是评价一个算法优劣的最重要的标准。

（3）**可读性（易理解）**:算法的可读性是指一个算法可供人们阅读的容易程度，算法是否容易理解。

（4）**健壮性**:健壮性是指一个算法对不合理数据输入的反应能力和处理能力，也称为容错性。

5、**算法的复杂度**

（1）算法的复杂度：算法效率的度量

（2）算法复杂度分类：**时间复杂度和空间复杂度**

①、**时间复杂度：指执行算法所需要的计算工作量。**通常，一个算法所用的时间包括编译时间和运行时间。度量方法是执行算法所需要的基本运算次数。

②、**空间复杂度：指执行这个算法所需要的内存空间。**包括算法程序所占的空间，输入的初始数据所占的空间，算法执行过程中所需的额外空间。

（3）**空间复杂度和时间复杂度并不相关**。

（4）查找算法最坏比较次数

①、顺序查找：只能用顺序查找的情况，**线性无序表；有序线性链表**。

长度为 n 的线性表最坏查找次数为：n 次；

最大值或最小值的比较次数为 n-1 次

②、二分法查找：二分法只适用于**顺序存储的有序表**。最坏比较次数为：$\log_2^n$ 次。

查找算法的复杂度表

查找	平均时间复杂度	查找条件	算法描述
顺序查找	O(n)	无序或有序队列	按顺序比较每个元素，直到找到关键字为止
二分查找（折半查找）	O(log2n)	有序数组	查找过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。 如果在某一步骤数组为空，则代表找不到。

（5）排序算法最坏比较次数

排序方式	最坏比较次数
冒泡排序	n(n-1)/2
快速排序	n(n-1)/2
简单插入排序	n(n-1)/2
简单选择排序	n(n-1)/2
希尔排序	$O(n^{1.5})$(第二小)
堆排序	$O(n\log_2 n)$(最小)

排序算复杂度表

排序法	最差时间分析	平均时间复杂度	稳定度	空间复杂度
冒泡排序	O(n2)	O(n2)	稳定	O(1)
插入排序	O(n2)	O(n2)	稳定	O(1)
选择排序	O(n2)	O(n2)	稳定	O(1)
二叉树排序	O(n2)	O(n*log2n)	不一顶	O(n)
快速排序	O(n2)	O(n*log2n)	不稳定	O(log2n)~O(n)
堆排序	O(n*log2n)	O(n*log2n)	不稳定	O(1)
希尔排序	O(n2)	O(log2n)--O(n2)	不稳定	O(1)

5、数据结构基本概念

（1）数据：数据是客观事物的符号表示，是能输入到计算机中并被计算程序识别和处理的符号的总称，如文档，声音，视频等。

（2）数据元素：数据元素是数据的基本单位。

（3）数据对象：数据对象是性质相同的数据元素的集合。

（4）**数据结构：是指由某一数据对象中所有数据成员之间的关系组成的集合。**

（5）研究数据结构一般包括三方面：**数据的逻辑结构、数据的存储结构，定义在逻辑结构和存储结构上的运算。**

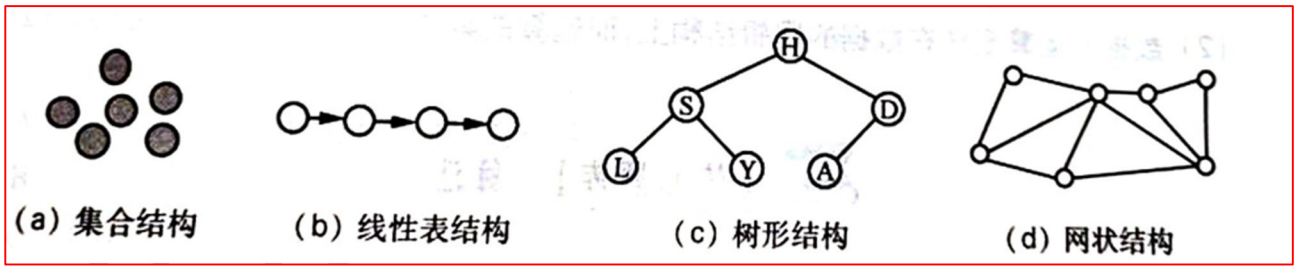
6、数据结构的逻辑结构

（1）**数据的逻辑结构是对数据元素之间的逻辑关系的描述，与数据的存储无关，是面向问题的，是独立于计算机的。它包括数据对象和数据对象之间的关系。**

要素：D,R；D 为数据元素的集合，R 是 D 上关系的集合

D={父亲，儿子，女儿}，R={{父亲，儿子}，{父亲，女儿}}

（2）常见的逻辑结构：



① 集合结构：数据中的元素同属于一个集合。

② 线性表结构：数据元素按先后次序连接。

③ 树形结构：数据元素间存在一对多的关系。

④ 网状结构：数据元素有各种各样的复杂连接，数据元素间存在多对多的关系。

7、数据结构的物理结构

（1）**数据的存储结构也称为数据的物理结构，是数据在计算机中的存放的方式，是面向计算机的，它包括数据元素的存储方式和关系的存储方式。**

（2）**常见的存储结构：顺序，链式，索引等**

8、**数据结构**

（1）图形表示：

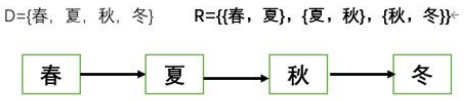
①、数据元素表示：

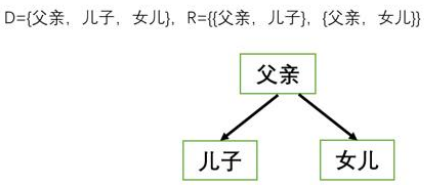
②、R 关系的二元组：一条有向线段从前件指向后件 

1）没有前件的结点称为“根结点”

2）没有后件的结点称为“叶子结点”

3）除了根结点和叶子结点的其他结点称为“内部结点”

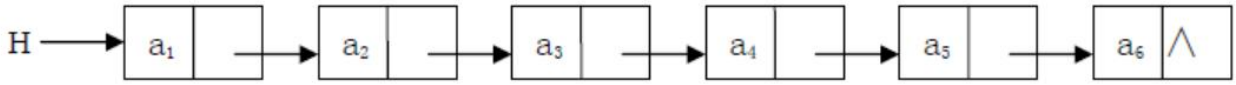




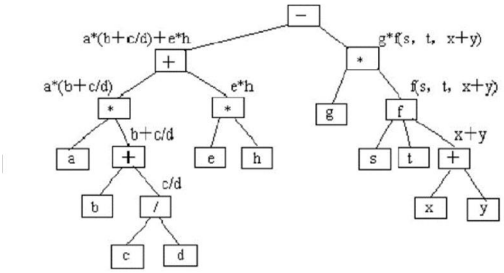
（2）线性结构：栈、队列、双向链表等

①、有且只有一个根结点

②、每个结点最多有一个前件，也最多有一个后件



（3）非线性结构：不满足线性结构条件的数据结构。如树、二叉树为非线性结构。



9、线性表及其顺序存储结构

（1）线性表：由一组数据元素构成，数据元素的位置只取决于自己的序号，元素之间的相对位置是线性的。

（2）在复杂线性表中，由若干项数据元素组成的数据元素称为记录；由多个记录构成的线性表称为文件。

（3）非空线性表的结构特征：

①、有且只有一个根结点 a1，它无前件

②、有且只有一个终端结点 an，它无后件；

③、除根结点与终端结点外，其他所有结点有且只有一个前件，也有且只有一个后件。

（4）结点个数 n 称为线性表的长度，当 n=0 时，称为空表

（5）线性表的顺序存储结构具有以下两个基本特点：

①、线性表中所有元素所占的存储空间是连续的；

②、线性表中各数据元素在存储空间中是按逻辑顺序依次存放的。

（6）线性表存储结构实现的举例：

①、数组：有序存储，占有一片连继地址，顺序存储结构的。

②、链表：无序存放的元素关联起来，链式存储结构的。

③、栈：先进先出(限定在一端)，链式结构或顺序存储结构。

（7）线性表的运算与操作

①、顺序表的运算：查找、插入、删除、合并、排序、复制、逆转等

②、原则：运算后仍操持线性表的特性

③、示例：

10、**栈**

（1）栈是一种特殊的线性表，只允许在表的一端进行插入和删除的线性表；插入，删除的一端为栈顶，另一端为栈底；当表中没有元素时为空栈。

（2）栈是一种后进先出（或先进后出 Last In First Out）的线性表。栈具有记忆功能。

（3）**栈的存储结构：**

①、顺序存储结构：用一组地址连续的存储单元即一维数组来存储；

②、链式存储：用线性链表来存储；

（4）栈的基本运算

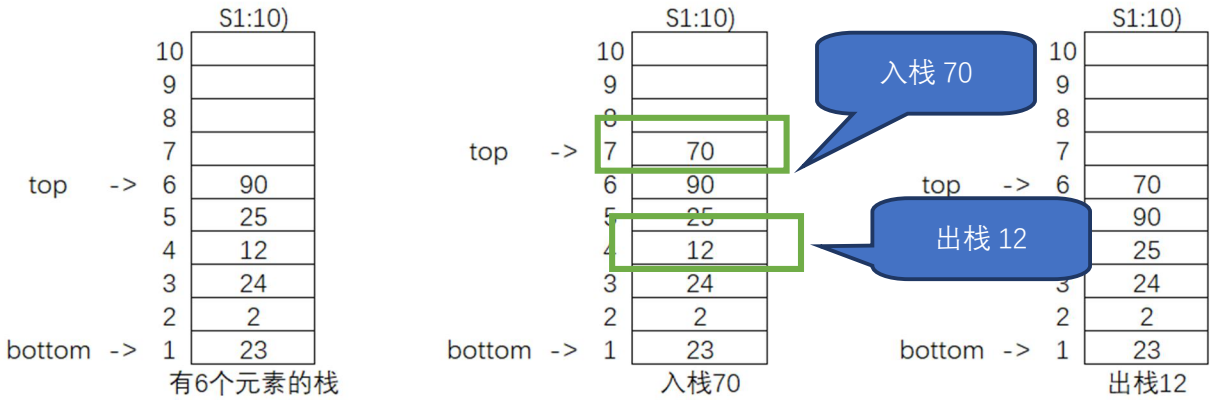
①、**入栈运算**，在栈顶位置插入元素；

②、**退栈运**算，删除元素(取出栈顶元素并赋给一个指定的变量)；

③、**读栈顶元素**，将栈顶元素赋给一个指定的变量，此时指针无变化。

（5）运算举例：

top=0，表示栈空，top=m 表示栈满



12、**队列**

（1）队列是一种特殊的线性表，只允许在表的一端插入，在另一端删除，允许插入的一端是队尾（rear），允许删除的一端为队头（front）；当表中没有元素是空队列；队列是一种先进先出的线性表。(FIFO)

（2）**队列的存储结构：**

①、顺序存储结构：一维数组

②、链式存储：线性链表；

（3）**队列的顺序存储结构一般采用循环队列的形式**。循环队列 s=0 表示队列为空；s=1 且 front=rear 表示队满。

（4）队列的基本运算

①、入队运算：从队尾插入一个元素；

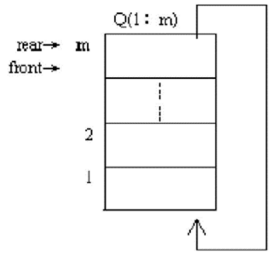
②、退队运算：从队头删除一个元素

③、取读队头元素

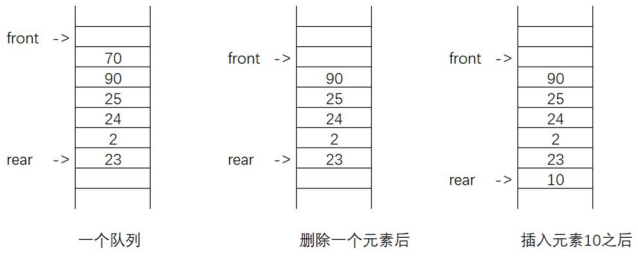
（5）计算循环队列的元素个数：“尾指针减头指针”，若为负数，再加其容量即可。

队列长度（元素个数）=（m+rear-front)%m

m 为总长度。



（6）运行举例：



13、**线性链表**

（1）**线性链表是线性表的链式存储结构，数据结构中的每一个结点对应于一个存储单元，这种存储单元称为存储结点，简称结点。**结点由两部分组成：

①、用于存储数据元素值，称为**数据域**；

②、用于存放指针，称为**指针域**，用于指向前一个或后一个结点。

（2）在链式存储结构中，存储数据结构的**存储空间可以不连续**，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致，而数据元素之间的逻辑关系是由指针域来确定的。链式存储方式既可以用于表示线性结构，也可用于表示非线性结构。

（3）线性单链表中，HEAD 称为头指针，HEAD=NULL（或 0）称为空表。

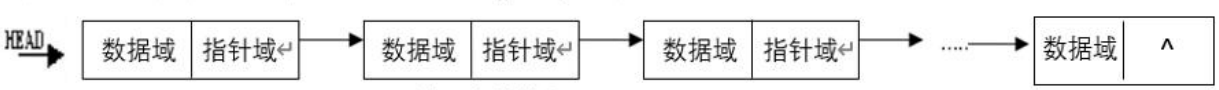


图 1 单链表的结构

（4）对线性链表中的每个结点设置**两个指针，一个称为左指针，用以指向其前件结点**；另一个称为**右指针，用以指向其后件结点**。这样的表称为**双向链表**。

双向链表有两个指针：**左指针（Llink）指向前件结点，右指针（Rlink）指向后件结点**。

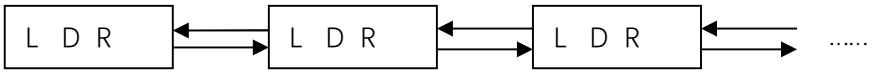


图 2 双链表的结构

（5）在线性链表中，其插入与删除的运算虽然比较方便，但还存在一个问题，在运算过程中对于空表和对第一个结点的处理必须单独考虑，使空表与非空表的运算不统一。为了克服线性链表的这个缺点，可以采用另一种链接方式，即循环链表。

循环链表具有以下两个特点：

①、在**链表中增加了一个表头结点**，其数据域为任意或者根据需要来设置，指针域指向线性表的第一个元素的结点，而**循环链表的头指针指向表头结点**；

②、循环链表中**最后一个结点的指针域不是空**，而是**指向表头结点**。即在循环链表中，所有结点的指针构成了一个环状链。

循环链表：**循环链表与单链表的不同的是它的最后一个结点的指针域存放的是指向第一个结点的指针而单链表存放的是空指针。**

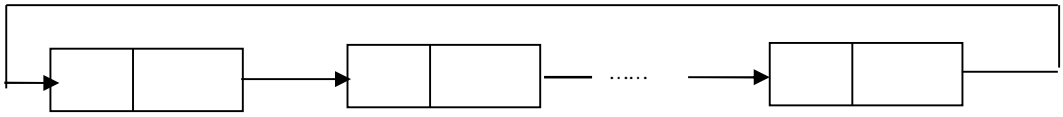
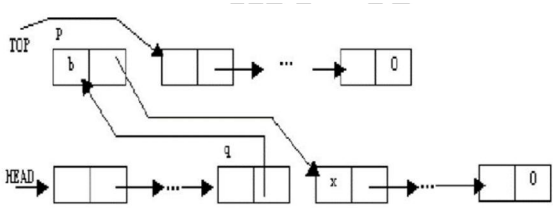


图 3 循环链表的结构

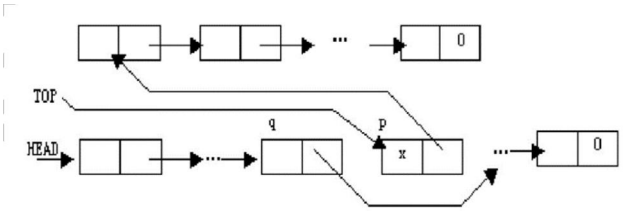
（6）线性链表的基本运算：**查找、插入、删除**。

（7）运行举例：

①、插入：



②、删除：



14、**树**

（1）**树是一种非线性结构，是 n 个结点的有限集。当 n=0 时为空树，n>0 时为非空树。**

①、结点的度：结点所拥有的子树的个数，数一下分叉个数，如结点 1 的有三个叉，即为 3.

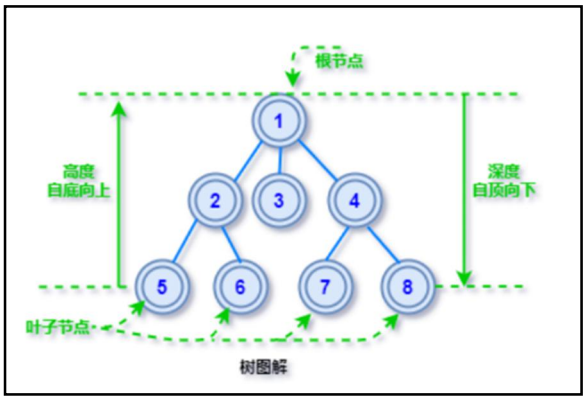
②、**叶子结点**：度为 0 的结点,即没有下一代无后继无分叉。如图 3，5，6，7，8 结点。

③、分支结点：除叶子结点以外的结点。

④、结点的层次：根结点在第一层，同一层上左右结点的子结点在下一层。

⑤、**树的深度**：所处层次最大的那个结点的层次。从根节点（结点 1）开始数有多少层，就是多少深度，如图上所示，有三层。即深度为 3。

⑥、**树的度**：树中所有结点的度的最大值。即 1 结点有三个子结点（2，3，4），且为最大，度为 3。



（2）**二叉树是一种特殊的树形结构，每个结点最多只有两棵子树，且有左右之分不能互换，因此，二叉树有五种不同的形态。**

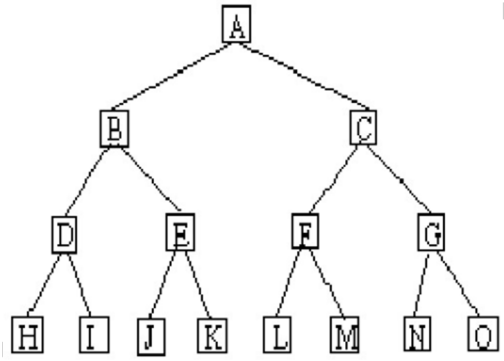
①、性质 1 在二叉树的第 k 层上，最多有 2k-1(k≥1）个结点。

②、性质 2 深度为 m 的二叉树最多有 2^m-1 个结点。

③、性质 3 在任意一棵二叉树中，度为 0 的结点（叶子结点）总是比度为 2 的结点多一个。

④、性质 4 具有 n 个结点的二叉树，其深度不小于[log2n]+1,其中[log2n]表示为 log2n 的整数部分。

（3）**满二叉树：除最后一层外，每一层上的所有结点都有两个子结点。**

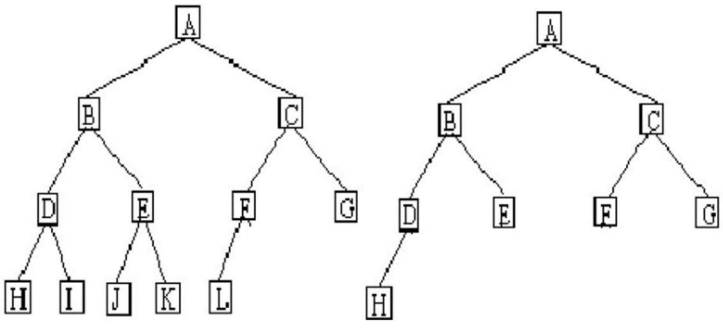


深度为 4 的满二叉树

（4）**完全二叉树:除最后一层外，每一层上的结点数均达到最大值；在最后一层上只缺少右边(连续)的若干结点。**

①、性质 1 具有 n 个结点的完全二叉树的深度为[log2n]+1。

②、性质 2 完全二叉树中度为 1 的结点数为 0 或 1。

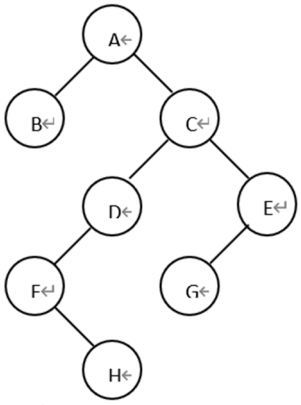


深度为 4 的完全二叉树

（4）**满二叉树是完全二叉树，而完全二叉树一般不是满二叉树。**

（5）**二叉树遍历：前序、中序和后序**

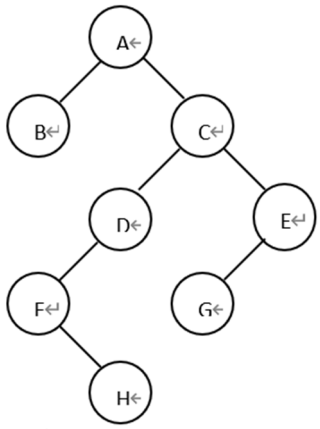
①、**前序遍历**：先访问根结点、然后遍历左子树，最后遍历右子树；并且，在遍历左、右子树时，仍然先访问根结点，然后遍历左子树，最后遍历右子树。（根-左-右）



前序遍历结果：

ABCDFHEG

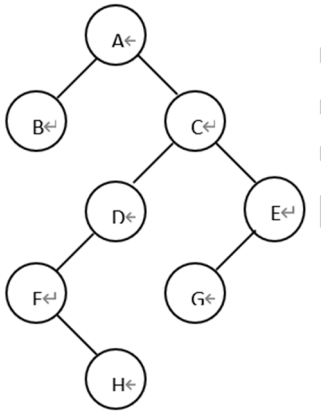
②、**中序遍历**：先遍历左子树、然后访问根结点，最后遍历右子树；并且，在遍历左、右子树时，仍然先遍历左子树，然后访问根结点，最后遍历右子树。（左-根-右）



中序遍历结果：

BAFHDCGE

③、**后序遍历**：先遍历左子树、然后遍历右子树，最后访问根结点；并且，在遍历左、右子树时，仍然先遍历左子树，然后遍历右子树，最后访问根结点。（左-右-根）



后序遍历结果：

BHFDGECA

2.5 软件工程基础

1、软件危机

软件危机泛指在计算机软件的开发和维护过程中遇到的一系列严重的问题，集中表现在成本，质量，

生产效率等几个方面。

2、软件工程

（1）软件工程是指采用工程的概念、原理、技术和方法指导软件的开发与维护。

（2）软件工程的主要思想强调在软件开发过程中需要应用工程化原则。

（3）软件工程的核心思想是把软件当作一个工程产品来处理。

3、**软件工程三要素：方法、工具、过程。**

记忆口诀：阮三要（软件工程三大要素）一要芳芳（方法）为爱情，二要工作（工具）为事业，三要过日子（过程）为生活。

（1）方法：方法是完成软件工程项目的技术手段；为软件开发提供"如何做"的技术。

（2）工具：工具支持软件的开发、管理、文档生成；提供的自动的或半自动的软件工程的支撑环境。

（3）过程：过程支持软件开发的各个环节的控制、管理；获得高质量的软件所需要完成的一系列任务的框架。

4、软件工程的目标（七大目标）：

记忆口诀：阮七目（可移植性）靠（可靠性）维（可维护性）修（可修改性）重（可重用）新追（可追踪性）你（可理解性）

（1）**可修改性**：允许对系统进行修改而不增加原系统的复杂性。它支持软件的调试和维护，是一个难以达到的目标。

（2）**可靠性**：能防止因概念、设计和结构等方面的不完善造成的软件系统失效，具有挽回因操作不当造成软件系统失效的能力。

（3）**可理解性**：系统具有清晰的结构，能直接反映问题的需求。可理解性有助于控制系统软件复杂性，并支持软件的维护、移植或重用。

（4）**可维护性**：软件交付使用后，能够对它进行修改，以改正潜伏的错误，改进性能和其它属性，使软件产品适应环境的变化等。软件维护费用在软件开发费用中占有很大的比重。可维护性是软件工程中一项十分重要的目标。

（5）**可重用性**：把概念或功能相对独立的一个或一组相关模块定义为一个软部件。可组装在系统的任何位置，降低工作量。

（6）**可移植性**：软件从一个计算机系统或环境搬到另一个计算机系统或环境的难易程度。

（7）**可追踪性**：根据软件需求对软件设计、程序进行正向追踪，或根据软件设计、程序对软件需求的逆向追踪的能力。

4、软件工程原则（八大原则）：

记忆口诀：阮八原（软件工程八大原则）居（局部化）然相信（信息隐藏）一（一致性）碗（完备性）馍（模块化）会因为缺（确定性）一克盐（可验证性）变臭（抽像）了。

（1）**抽象**：采用分层次抽象，自顶向下、逐层细化的办法控制软件开发过程的复杂性。

（2）**模块化**：模块化有助于信息隐蔽和抽象，有助于表示复杂的系统。

模块化的独立性：**耦合度和内聚度是衡量软件模块独立性的主要标准**，软件设计应尽量做到**高内聚，低耦合**（**高内聚低耦合是判断软件设计好坏的标准**）。

①、**内聚**：模块内部的紧密程度，模块内部联系越紧，模块与模块之间的耦合就会越低。

②、**耦合**：模块与模块之间的联系或关联程度。

（3）**信息隐蔽**：将模块设计成"黑箱"，实现的细节隐藏在模块内部，不让模块的使用者直接访问，这就是信息封装。

（4）**局部化**：保证模块之间具有松散的耦合，模块内部具有较强的内聚，这有助于控制解的复杂性。

（5）**确定性**：软件开发过程中所有概念的表达应是确定的、无歧义性的、规范的。

（6）**一致性**：整个软件系统使用一致的概念、符号和术语。

（7）**完备性**：软件系统不丢失任何重要成分，可以完全实现系统所要求功能的程度。

（8）**可验证性**：易于检查、测试、评审，确保系统的正确性。

5、软件工程过程

（1）软件工程过程是把软件转化为输出的一组彼此相关的资源活动

（2）软件工程过程的四个基本活动：

记忆口诀：阮四基（软件工程的过程中四个基本活动）

一基小猪佩奇计划找 1P（Plan，软件规格说明）,一起看 2D(Do，软件开发)动画,发现没有安全的 3C(Check，软件确认)电子产品，最后决定去 4A(Action，软件演进)级景区玩，忽然两人互相看看了对方的胸，原来他们就是景区啊。

①、**P(plan)—软件规格说明**。规定软件的功能及其运行时的限制。

②、**D(do)—软件开发**。产生满足规格说明的软件。

③、**C(check)—软件确认**。确认软件能够满足客户提出的要求。

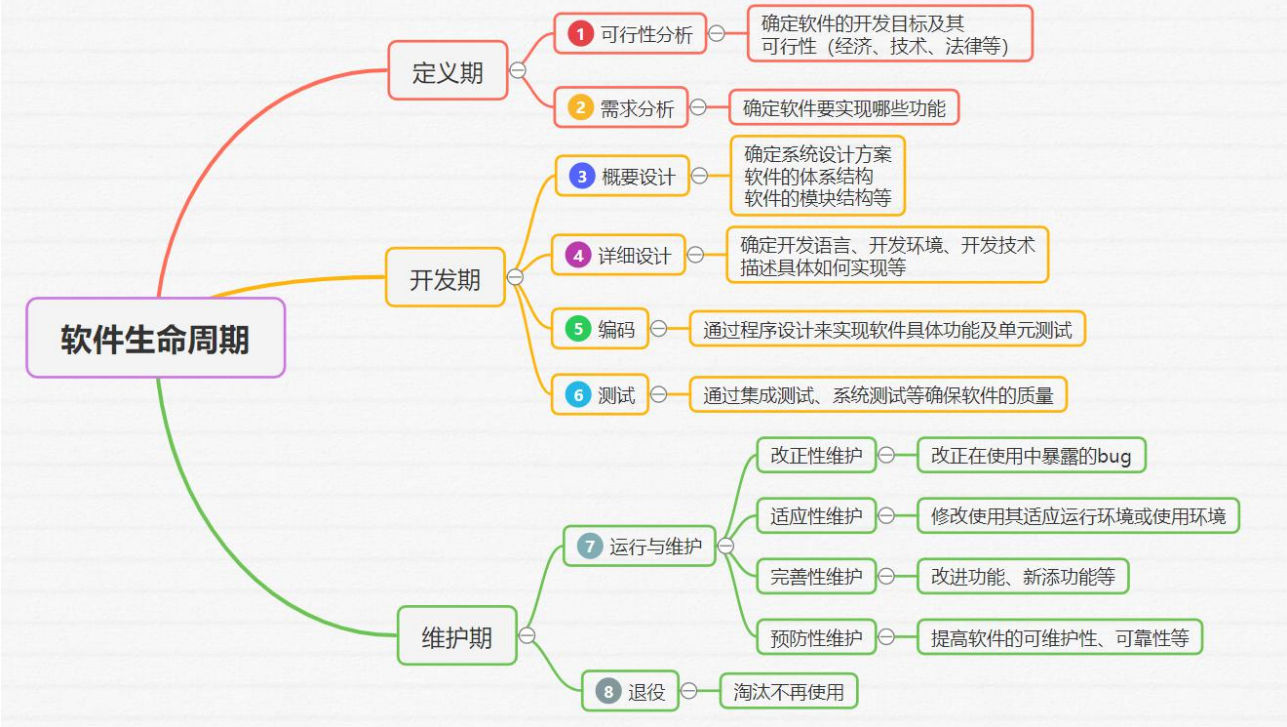
④、**A(action)--软件演进**。为满足客户的变更要求，软件必须在使用的过程中演进。

6、软件工程模型与**软件生命周期**

（1）软件工程模型：也称软件开发模型，它是软件开发全部过程、活动和任务的结构框架。

（2）典型模型：瀑布模型、增量模型、原型模型、喷泉模型、V 模型等

（3）软件生命周期（三个时期八个阶段）：定义期（可行性分析、需求分析）、开发期（概要设计、详细设计、编码、测试）、维护期（运行与维护、退役）。如下图所示。



①、**瀑布模型**：顺序清晰,一般**需求比较明确**，面向过程(结构化)开发。

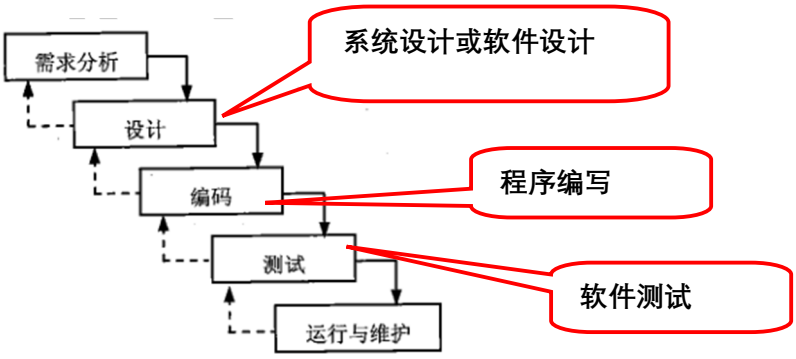


图 5-2 瀑布模型

②、**增量模型**：**增量模型是将整个模型分为多个子开发阶段的开发模型，其中每个开发阶段的相应测试阶段都是实践**。对所开发的领域比较熟悉而且已有原型系统，进行**已有产品升级或新版本开发**等时候比较适合。

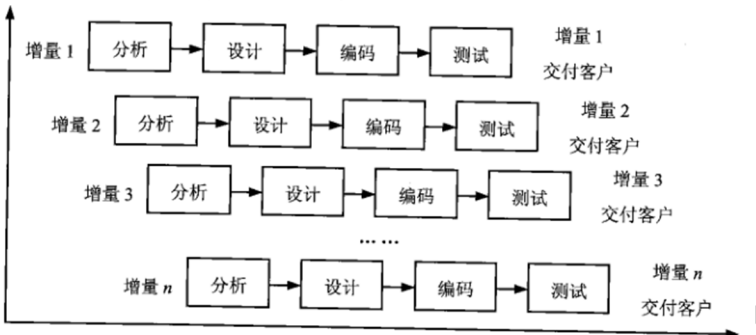


图 5-4　增量模型

③、**原型模型**：原型模型采用逐步求精的方法完善原型，使得原型能够“快速”开发，避免了像瀑布模型一样在冗长的开发过程中难以**对用户的反馈作出快速的响应**。**原型模型适用于那些不能预先确切定义需求的软件系统的开发。**

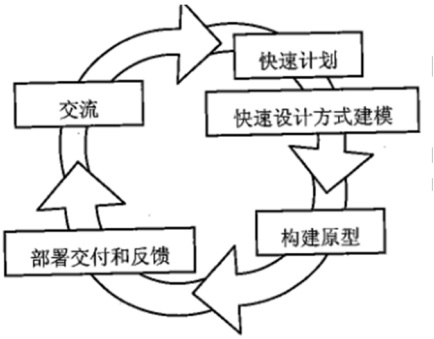


图 5-5　原型模型

④、**喷泉模型**：喷泉模型是一种**以用户需求为动力，以对象为驱动的模型**，主要用于描述面向对象的软件开发过程。适应于**面向对象的软件开发过程**。

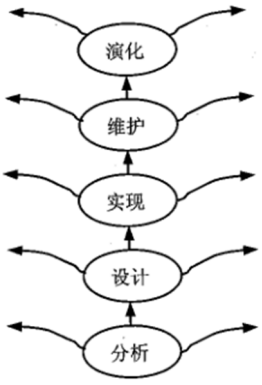
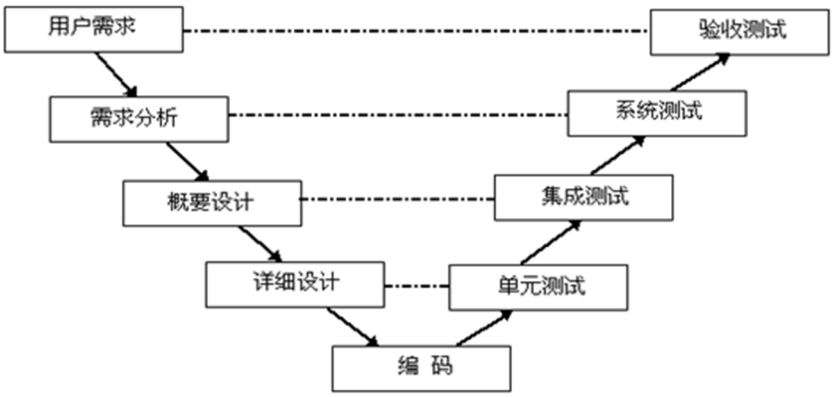


图 5-7 喷泉模型

⑤、**V 模型**：**瀑布模型的变种，它反映了测试活动与分析和设计的关系 。模式是一种传统软件开发模型，一般适用于一些传统信息系统应用的开发。**



7、需求分析概述

（1）**需求分析阶段的工作：需求获取，需求分析，编写需求规格说明书，需求评审。**

（2）结构化需求分析方法：

①、面向数据结构的 Jackson 方法（ISD）

②、面向数据流的结构化分析方法（SA）

③、面向数据结构的结构化数据系统开发方法（DSSD）

（3）**面向对象的分析方法（OOD）：抽象，信息隐蔽，模块化，局部化，确定性，一致性，完备性，可验证性。**

（4）**软件需求规格说明书（SRS，Software Requirement Specification）是需求分析阶段得出的最主要的文档。**软件需求规格说明书的特点：有正确性、无歧义性、完整性、可验证性、一致性、可理解性、可修改性和可追踪性。其中最重要的是无歧义性。

8、结构化设计

（1）**结构化方法包括结构化分析方法，结构化设计方法，结构化编程方法。**

（2）**结构化方法中，软件功能分解属于总体设计阶段。**

（3）结构化分析方法是面向数据流自顶而下逐步求精进行需求分析的方法。

（4）结构化分析方法在软件需求分析阶段的应用。

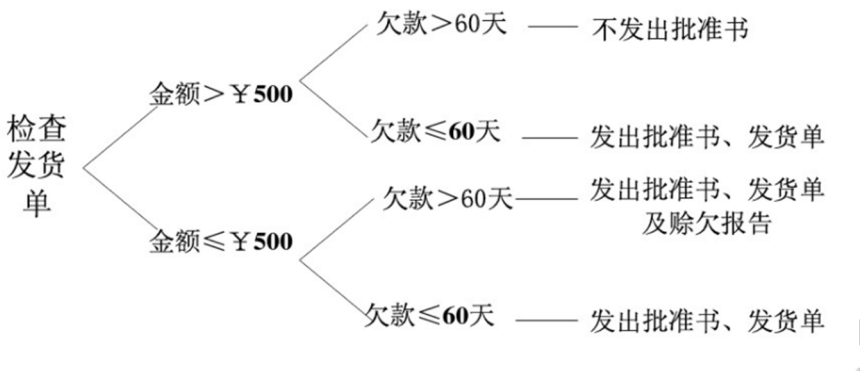
（5）结构化分析的常用工具：

①、**数据流图(DFD)**：箭头、圆或椭圆、双横、方框

图形	说明
⬭	加工（转换）。输入数据经加工变换产生输出。
→	数据流。先江头方向传送数据的通道，一般在旁边标注数据流名。
═	存储文件（数据源）。表示处理过程中存放各种数据的文件。
▭	源，潭。表示系统和环境接口，属系统之外的实体。

②、**数据字典(DD)**：数据流、数据流分量、数据存储、处理

③、**判定树（决策树）**：



④、**判定表**:

		1	2	3	4
条件	发货单金额	＞￥500	＞￥500	≤ ￥500	≤ ￥500
	赊欠情况	＞60天	≤60天	＞60天	≤60天
操作	不发出批准书	√			
	发出批准书		√	√	√
	发出发货单		√	√	√
	发出赊欠报告			√	

9、**结构化设计**

（1）**概要设计(总体设计)：将软件需求转化为软件体系结构、确定系统级接口、全局数据结构或数据库模式；**

①、任务：划分出组成系统的物理元素、设计软件的结构;

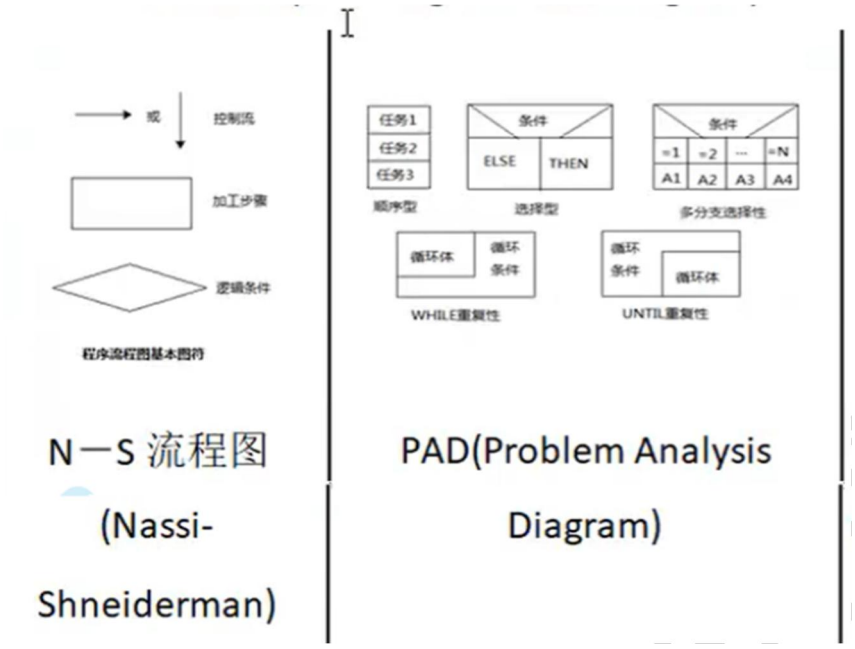
②、工具：**结构图（程序结构图）、面向数据流设计**。

③、产物：**概要设计说明书**

（2）**详细设计：确立每个模块的实现算法和局部数据结构，用适当方法表示算法和数据结构的细节。**

①、任务：**详细设计的主要任务是设计每个模块的实现算法、所需的局部数据结构。**

②、工具：**图形工具（程序流程图、N-S 图、PAD 图）、表格工具（判定表）、语言工具（PDL—过程设计语言）**



③、产物：**详细设计说明书**

10、**面向对象分析与设计**

（1）**面向对象分析（OOA）：确定需求或者业务的角度，按照面向对象的思想来分析业务.**

（2）面向对象分析五个活动：认定对象、组织对象、描述对象的相互作用、确定对象的操作、定义对象的内部信息。

①、认定对象：在应用领域中，按自然存在的实体确立对象。

②、组织对象：分析对象间的关系，抽象出类，简化关联对象，建议层次结构等

③、对象间的相互作用：描述各对象在系统中的关系，得出对象的界面描述。

④、基本对象的操作：对象的操作可以是简单的（创建，增加和删除等）;也有复杂的操作（多个对象信息的连接）；确定对象操作后，进行定义对象的内部，包括内部数据信息、信息存储方法、继承关系以及可能生成的实例数等属性。

（3）**面向对象设计:设计阶段考虑与实现有关的因素，对 OOA 模型进行调整并补充与实现有关的部分，形成 OOD 模型。**包括四个部件，即人机交互部件、问题域部件、任务管理部件和数据管理部件。

①、设计人机交互部件

②、设计问题域部件

③、设计任务管理部件

④、设计数据管理部件

（4）**面向对象分析与设计工具：UML**

①、UML(Unified Modeling Language): 用于系统的可视化建模语言，是一种用于软件蓝图的标准语言，可用于详细描述的语言，是一种构造语言，也是 一种文档化语言。

②、作用：为软件系统建立可视化模型；为软件系统建立构件；为软件系统建立文档。

③、主要的模型：功能模型（用例图）、对象模型（对象图、类图）、动态模型（序列图、活动图、状态图）。

④、UML 由模型元素（类、对象、消息等）、图（元素集图形表示）、视图（系统的抽像表示）和通用机制（注释、模型元素的语义等）等几个部组成。

⑤、UML 图：用例图、类图、对象图、序列图、协作图、状态图、活动图、部署图、构件图。

⑥、UML 常见关系：UML 常见的六种关系图标：泛化（继承）、实现（接口）、关联（比如学生、课程和课程表）、聚合（比如汽车、引擎和轮胎）、组合（比如公司和部门）、依赖（比如现代人和计算机之间操作）

⑦、UML 是一种建模语言，不是一种方法，不包括过程的 概念，本身是独立于过程

11、**软件测试**

（1）**软件测试：为了发现错误而执行程序的过程，成功的测试是发现了至今尚未发现的错误的测试。**

（2）**测试的目的就是希望能以最少的人力和时间发现潜在的各种错误和缺陷。**

（3）一般软件测试分为：**单元测试、集成测试、系统测试和验收测试。**

①、**单元测试**：针对每个单元的测试，以确保每个模块能正常工作为目标。

②、**集成测试**：对已测试过的模块进行组装，进行集成测试，目的在于检验与软件设计相关的程序结构。

③、**系统测试**：检验软件产品能否与系统的其他部分（如硬件、数据库及操作人员）协调工作。

④、**验收（用户）测试**：检验软件产品质量的最后一道工序，主要突出用户的作用，同时软件开发人员也应有一定程度的参与。

（4）测试方法分类：

①从是否需要执行被测软件的角度分为静态测试和动态测试

- **静态测试包括代码检查、静态结构分析、代码质量度量。不实际运行软件，主要通过人工进行。**
- **动态测试是通过运行软件来检验软件中的动态行为和运行结果的正确性。**

②按功能分为白盒测试和黑盒测试

12、**白盒测试**

（1）白盒测试也称为结构测试或逻辑测试，是把程序看成装在一只透明的白盒子里，测试者完全了解程序的结构和处理过程。它根据程序的内部逻辑来设计测试用例，检查程序中的逻辑通路是否都按预定的要求正确地工作。

（2）基本原则：

- 保证所测模块中每一独立路径至少执行一次。
- 保证所测模块所有判断的每一分支至少执行一次。
- 保证所测模块每一循环都在边界条件和一般条件下至少各执行一次。
- 验证所有内部数据结构的有效性。
- 按照白盒测试的基本原则，“白盒”法是穷举路径测试。

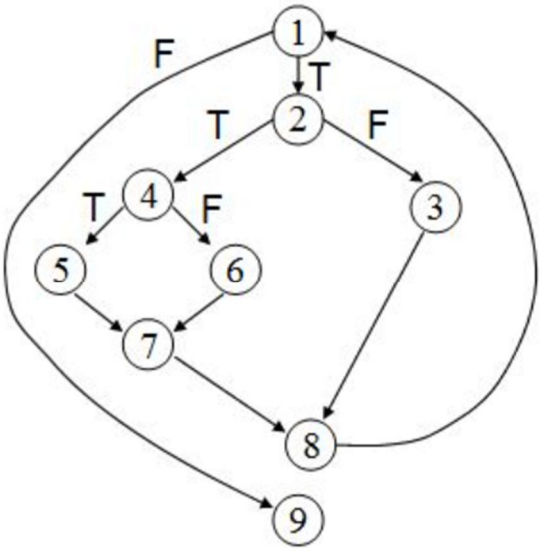
（3）**白盒测试的方法：逻辑覆盖，基本路经测试**

（4）逻辑覆盖：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖和路径覆盖发现错误的能力呈由弱至强的变化。

- 语句覆盖每条语句至少执行一次。
- 判定覆盖每个判定的每个分支至少执行一次。
- 条件覆盖每个判定的每个条件应取到各种可能的值。判定/条件覆盖同时满足判定覆盖条件覆盖。
- 条件组合覆盖每个判定中各条件的每一种组合至少出现一次。
- 路径覆盖使程序中每一条可能的路径至少执行一次。

（5）基本路经测试

白盒测试的一种常用方法是基本路径法，根据源代码构造程序流程图，转换为控制流程图，得到基本路径，进而为每条基本路径设计测试用例。基本路径法的一个关键步骤是识别出所有的基本路径。



CC=4
1,9
1,2,3,8,1
1,2,4,5,7,8,1
1,2,4,6,7,8,1

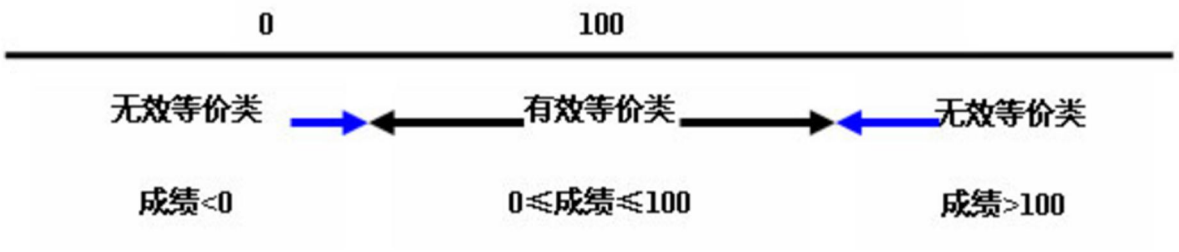
13、**黑盒测试**

（1）黑盒测试也称功能测试或数据驱动测试，是把程序看成一只黑盒子，测试者完全不了解，或不考虑程序的结构和处理过程。它根据规格说明书的功能来设计测试用例，检查程序的功能是否符合规格说明的要求。

（2）黑盒测试的方法：等价划分法，边界值分析法，错误推测法。

（3）等价划分法：这是一种典型的黑盒测试方法，它是将程序的所有可能的输入数据划分成若干部分(及若干等价类)，然后从每个等价类中选取数据作为测试用例。

划分等价类：有效等价类、无效等价类



（4）边界值分析法:它是对各种输入、输出范围的边界情况设计测试用例的方法。

边界检验类型：数字、字符、位置、重量、大小、速度、方位、尺寸、空间等。

例如：例如手机号的位置是 11 位

（5）错误推测法：人们可以靠经验和直觉推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的用例。

使用方法：极限值设计、特殊取值设计、特殊组网考虑、端到端用例考虑等。

14、程序调试

（1）**在对程序进行了成功的测试之后将进入程序调试（Debug，即排错）。程序的调试任务是诊断和改**

正程序中的错误。

（2）**程序调试的基本步骤：错误定位；修改设计和代码，以排除错误；进行回归测试，防止引进新的错误。**

（3）软件调试可分为静态调试和动态调试。

（4）主要的调试方法有：**强行排错法；回溯法；原因排除法，包括演绎法，归纳法和二分法。**

15、**软件测试与程序调试的区别：**

软件测试是尽可能多地发现软件中的错误，而程序调试先要发现软件的错误，然后借助于一定的调试工具去执行找出软件错误的具体位置。

软件测试贯穿整个软件生命期，调试主要在开发阶段。

16、信息系统开发方法的发展过程

年代	20 世纪 70 年代	20 世纪 80 年代	20 世纪 90 年代
程序设计方法	**SP 方法** JSP 方法		
软件工程方法	SADT 方法 JSD 方法	prototyping 方法	**OO 方法**
管理/需求分析	SRD 方法 BSP 方法	CSFs 方法	
自动化开发方法			CASE 方法

SP(Structured Program)为结构化程序方法，

JSP(Jackson Structured Program)为杰克逊结构程序方法，

JSD(Jackson System Development)为杰克逊系统开发方法，

SADT(Structured Analysis & Design Technology)为结构化系统分析与设计技术，

prototyping 为原型方法，

OO(Object Oriented)为面向对象的开发方法，

SRD(Structured Requirements Defination)为结构化需求定义方法，

BSP(Business Systems Planning)为商业系统规划法，

CSFs(Critical Success Fastors)为关键成功因子法，

CASE(Ccomputer Aided Software Engineering)为计算机辅助软件工程方法。

17、**软件的可靠性包括正确性和健壮性。软件的健壮性是非常重要的软件外部量度标准。软件设计的健壮与否直接反映了分析设计和编码人员的水平。**