做一个面试达人

美国编程师程序员面试宝典,年薪十万美金不是梦

作者: 顾颖琼 博士

联系 email: guyingqiong@hotmail.com

请用微信扫描此二维码,支持公共号,或者
加"dryingqionggu",或者搜索公共号 "顾颖琼博士说天下"

友情参与: 李大明 (华尔街资深金融分析师)

警告！

本书旨在帮助对电脑编程有兴趣的有为青年通过面试，拿到丰厚薪水，不保证该有为青年被录取后的工作表现.

作者简介:

顾颖琼，祖籍上海，1978 年出生。24 岁赴美求学，2007 年获 得圣母大学数学物理博士学位，现为电脑资深工程师。他研究生院所研究的核潮汐模型第一次将量子物理中的转动和振动有机的结合起来 。顾颖琼博士是美国电气和电子工程师资深会员，Trinity college 理学院院长。

顾颖琼博士先后在微软公司、Ebay 公司担任资深职位。 他善于发现问题，解决问题的风格为人称道。他擅长面向对象的语言设计,以及和工程师以及管理者进行多层次的交流,

顾颖琼博士非常关心热爱电脑编程的有为青年的成长，致力于帮助他们成为国际化的人才， 作者本人在美国面过超过百家的高科技公司, Microsoft,Google , Facebook, Twitter, Yahoo, Goldman sachs etc.

近千次的技术电话面试，几百次的程序员面对面面试给了他丰富的面试经验. 在这本书中,顾颖琼将从面试者和考官的角度分析怎么样成功通过面试,把握先机.

友情作者简介:

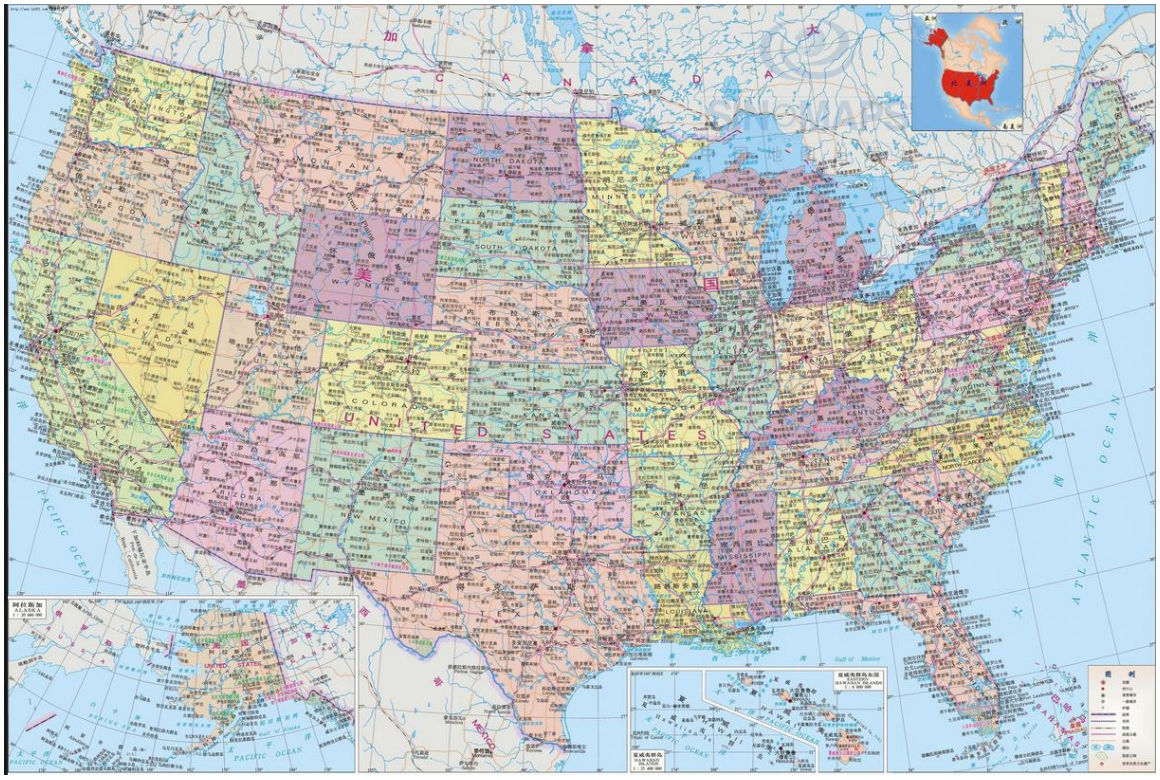
李大明，祖籍湖北武汉, 现为华尔街资深金融分析师,友情为本书奉献佳作

<<物理学博士捡破烂的心得>> .

本文更适合比较通用技能的编程师程序员职位，如果你有别人没有你的技能，恭喜你， 请直接跳至最后一页观赏面试小故事.

目录:

(1.1)公司位置分析



美国东西两个海岸聚集了绝大多数的高科技公司,两个海岸的面试风格以及公司中编程师程序员的地位也迥然不同.

东海岸的公司，编程师程序员多属于技术支持的地位，因为公司更多地侧重于金融服务等行业，面试官对求职者多侧重于实际工作环境，以及对具体操作细节的考察. 面试着装多需要正装.

西海岸因为有着软件业摇篮的传统，多数公司开创者是电脑本业出身.编程师程序员有着比较高的地位, 面试着装比较随意. 面试官对求职者多考察编程策略,以及软件设计等比较抽象的支持.这也是软件企业本身特点决定的.

对于初学者来说，如果具有一定的数学功底，西海岸的面试会更对面试者的胃口.

西海岸的面试风格更多地收到了微软的影响,本书将在随后的章节仔细分析这种风格.

(1.2)公司录取原则

在此，笔者希望读者记住三句话: 1)有人的地方就有江湖 2) 三分在天，七分在人，若尽人事，无怨无悔. 3)此处不留爷，自有留爷处.

很多人在面完编程师程序员职位之后，自我感觉相当完美，论据就是解决了所有的问题. 但用人单位的反馈往往不尽人意,如此反复再三，求职者容易陷入一种茫然无措的境界.

其实从用人单位或者 hiring manager (需要雇佣人的组头)来看，他/她需要一个 能解决问题，需求匹配，容易相处的人.

解决问题和需求匹配是一个比较硬性的标准，你知道什么，不知道什么，这些都是比较固定的.

容易相处往往是一个比较软性，比较主观的标准. 这也就为什么很容易在某一个组中聚集某一个种族或者特性的人原因.

微软面试的风格:

简历在微软官方网站
http://careers.microsoft.com/careers/en/us/home.aspx?popupflag=False&wsi=1&wa=wsignin1.0&popupflag=False

在线提交后, 你的简历会被电子过滤，如果你有对应位置所需关键字，你的简历会进入一个简历栈.

比如说，微软的一个位置想要候选人需要 SQL 的能力,但是你的简历中没有这个关键字，你的简历会被自动过滤掉.

Hiring manager 会在你的简历进入栈后浏览你的简历，这个过程通常只有几秒，如果他看见了他想要的一些经历和经验，他会转发你的简历给人力资源部，要求电话面试.

如果你通过了电话面试,人力资源部将会给你安排 Onsite 面试.

微软的 Onsite 会先安排三位面试官考察候选人的不同方面，每一个人的意见会在两个面试之间进行传递，在你进入下一个面试官的房间的时候，他已经知道前面人对你的评价.

三位面试官之后，如果有至少两位说 雇佣，那么这个候选人会被送去见更多的面试官,通常最后一个人是 AA,即为做决定的人.

微软的面试风格好处在于，如果三位面试官已经做出不雇佣的决定，候选人可以不需要见到后来的面试官.对双方的时间都有所节约，不宜之处也显而易见，前一任面试官的意见往往会影响下一任面试官对你的看法，一旦第一个表现不好，很容易造成恶性循环.

微软面试风格的改进 – google 面试风格

基于微软面试风格的缺陷,其余公司做了相应的改进.

Google 采用另外一种方式来录取 General hire 的工程师, http://www.google.com/about/jobs/，求职者的简历会被一个委员会的人审查. 然后对非本地候选人通常有两轮电话面试,在 Onsite 面试中，会有四到六人的面试官进行各方面考察。每一个面试官的意见会被分开发送到人力资源处, 分数会有强力推荐雇佣，一般雇佣，不推荐雇佣, 强力不推荐几档，和微软的不同，某一个人的不雇佣决定不会对最后的委员会决定作出关键性影响.雇佣委员会并没有直接见到候选人.好处在于雇佣与否的决定会更客观的基于面试者的技术能力上.

高盛(Goldman Sachs)采用另外一种完全不同的风格来面试候选人，候选人会见到很多人，每一个人往往只有半个小时，任何一个人说不雇佣，候选人就会被带走，结束当天的面试.

高盛(Goldman Sachs)做过一个内部调查，即使是最出色的员工，他也只会有 70%的工作时间在工作.怎么利用好剩下的 30%,变成了一个有趣的课题，高盛(Goldman Sachs)会要求员工尽量面试尽可能多的候选人,有句笑话说，高盛甚至连来华尔街的一条狗都会面试一下. 如此面试也是对面试官的一种压力，当他看见外面还有那么多优质候选人渴望进入高盛的时候，也是对面试官的一种压力，可以变相的促使员工更努力工作.

美国任何一个正式的公司对于面试官都有培训的过程,面试官的面试问题都必要符合公司要求。

现在给大家看看 google 的一个内部培训面试官材料 (**无修改全英文原始版本**)

Interviewing Guidelines

Our Goal as Interviewers

做一个面试达人　　美国编程师面试宝典,年薪十万美金不是梦

To motivate the candidate to share thoughts and information that will qualify or disqualify them for the position.

Behavior Based Interviewing

Behavior based interviewing focuses on experiences, behaviors, knowledge skills and abilities that are job related. It is based on the belief that past behavior and performance predicts future behavior and performance.

Examples of Behavior Based Interview Questions:

- Please give me an example of when you…
- Tell me about a time when you…
- Describe a situation where…

Closed and Leading Questions:

It is important to avoid closed and leading questions as they produce single word answers and can lead the candidate to the desired (although not necessarily accurate) response.

Interview Tips

- The interviewer should talk 20% of the time and listen 80% of the time
- Do your best to avoid the natural tendency to judge a candidate within the first five minutes of meeting them, but do trust your instincts
- Try to make the interview a conversation rather than a cross-examination
- Ensure all candidates have a positive experience and are excited about joining the company
- Be prepared!! Bring your list of questions, the candidate's resume and note taking materials
- Be on time and stick to the schedule
- The interviewer should run the interview. Do your best to keep the interview on track.
- Save enough time in the interview for the candidate to ask questions
- Do not discuss compensation or answer questions in this area. Please direct those questions to recruiting.

Legal vs. Illegal Interviewing

A candidate should only be asked questions that are job related. Before asking the question the interviewer should determine whether this information is really necessary in order to judge the candidate's qualifications, skill level, cultural fit and overall competence for the job.

Topics to Avoid:

- Race/Color
- National Origin
- Religion
- Marital Status
- Pregnancy
- Sex (gender)
- Age
- Sexual Orientation
- Genetic Information
- Disability
- Veteran Status

(2.1) 简历电脑初选

请注意，一般一份简历，用人单位初次只花五秒钟左右审查，如果看不见相应关键字，简历会被删除.

如同前面所讨论，你的简历一般会有电脑筛选，人力资源二选，Hiring Manager 三选. 举例子来说,摩根斯坦利招收实习生，因为收到太多的电子简历,所以他们设计了学校关键字以及特定方式，来淘汰大部分的简历候选人,这时候运气也是很重要的.

如果你有内部关系认识 Hiring Manager,可以直接提交简历给 Hiring Manager,也往往有奇效.

女性在寻找编程师程序员工作时，往往有特别的优势. 著名的金融分析师 Li daming 曾经面试了一位俄罗斯金发美女，他特别激动地发现对方居然还能解答微分方程 $y'=x$.美女被雇佣自然毫无疑问了.

(2.1) 简历修改

我修改并且审查过很多简历，最后觉得特别有用的是：

2.1.1 Summary 总结部分

在您的住址个人目标后面，加上一个 Summary (总结)，把公司的工作要求和自己的经验相结合，基本上所有的总结句子都在 提醒公司：hey,您需要的这个技能，我有直接相关工作经验。如果对方需要的某种技能你目前没有，你可以强调一些很强的学习能力和有一定的自学该知识的经验之类的。非常有用。

2.1.2. Education 教育背景

这是一把双刃剑，过高或者过低的教育背景都会让对方产生一种不够匹配的感觉.

Hiring manager 往往在招收人之前就有一个头脑中的设想，我想要一个什么样的人，这点往往会影响最初的简历筛选，如果候选人有博士和硕士学位，可以根据对方要求放不同学位上去.

Li daming 提出的一个建议很有用，我觉得非常适用于 多数人。就是在写毕业学校以及专业的时候，不要把时间写上去。大家以前有一些工作的经验，包括在学校工作，其实是很符合公司工作要求的。但 人力资源一看到 毕业时间就在 一段时间之前，就会假定你是新鲜毕业的人，根本不会往下看了。 在面试的时候大家一般会比较仔细的看简历。然后您就可以解释您的具体工作内容.

2.1.3. Working Experience 工作经验

如果有全职工作经验的话，您当然要写进去。 写经历的时候多用一些 buzz word。 因为过滤你简历的人或软件经常不知道你说的和工作要求其实是一回事。还有，写工作经历的时候多写一些实际的结果，improvement，有数字或百分比的更好。公司都喜欢有工作激情和有良好结果的人。

2.1.4.简历长度

长度不是问题，但是要言之有物，有侧重点.简历一定要千锤百炼，尽量找 同行业的资深人士帮着看和改。特别是做过面试官的朋友。 花钱找人修改不一定会有用， 不一定懂你的专业 .

面试编程师程序员时，需要阅读大量的书，但是更重要的是热爱编程，喜欢思考，以及要有把自己能力表现出的手段和方法.

对于其他一些技能，比如说 likeMS SQL,XML, HTML5,database design, COM,ASP,ADO,Java, VB,VBScript etc 挑自己知道的往简历上写，浅尝则止的也行，只要保证在 面试前再突击一下就行， 语言本身很难询问的更深.

(3) 电话面试

电话面试的优点:节约雇主的成本;减少繁琐成本.利于初步筛选候选人.缺点也很明显，更多地依赖于语言的表达，和受限于环境，一些面试者想表达的很难表达出来.

(3.1)电话面试常见题目

因为电话面试的限制，原来越多的电脑公司引入了 live meeting, good docs 进行在线编程，尽量减少纯语言可能带来的信息损失,尽管如此，依然有很多公司只是做电话面试. 东部公司依然侧重于电话，而西部高科技公司越来越多的引入了在线编程.

在一个长度通常为 45 分钟到 60 分钟的电话面试中, 面试官通常会有 5 分钟自我介绍,10 分钟询问简历内容,30 分钟技术测试.

在一个纯技术的电话面试中，怎么样才能迅速的和面试官进行无障碍的交流,准确理解面试官的意图是一个极大的挑战.

以下几点是美国面试官面试编程师程序员是常问的:

1)编程:候选人应该用正确的语法，在规定的时候内写一些简单的代码 ，用 C，C++或 Java 或者任何候选人感觉合适，面试官可接收的语言。
2）面向对象的设计: 面试官会提出一些需要被设计的对象, 来测试候选人的基本的面向对象的概念，并能拿出一个方案进行可行性 。
3）脚本语言和正则表达式： 候选人应该有能力来用一些成熟的脚本语言解决一些现存的问题。
4） 数据结构。候选人必须证明自己拥有最常见的数据结构的基本知识。
5）多线程, 过程, 比特和字节: 候选人必须能够表现出坚实的计算机基础知识。

6) 为什么我们公司或者我们项目 以及其他非技术问题? 候选人应该表达出对自己的激情和向往.

编程常见问题:

Write a function to reverse a string.
Write function to compute fibonacci number.
Write a function to reverse a string.

Write a function to find a string length.
Determine whether an integer is a palindrome.
Determine whether a string is a palindrome.

Implement regular expression matching with support for '.' and '*'.

Merge two sorted linked lists and return it as a new list.

Write a function to determine if a given target is in the array.

Given a binary tree, determine if it is height-balanced.

Find the contiguous subarray within an array (containing at least one number) which has the largest sum.

Given an unsorted array of integers, find the length of the longest consecutive elements sequence.

Implement wildcard pattern matching
Given a string, find the length of the longest substring without repeating characters.

Implement `atoi` to convert a string to an integer.

Given a sorted array, remove the duplicates in place such that each element appear only *once* and return the new length.
Given a collection of numbers that might contain duplicates, return all possible unique permutations.
Given a sorted linked list, delete all duplicates such that each element appear only *once*
Given a binary tree, determine if it is a valid binary search tree (BST).
Given a binary tree, return the *inorder* traversal of its nodes' values iteratively without using recursion.

Given inorder and postorder traversal of a tree, construct the binary tree.

Output all prime numbers up to a specified integer n.

Given a binary tree, find the lowest common ancestor of two given nodes in the tree.

Given preorder and inorder traversal of a tree, construct the binary tree.

还有大量类似题目，笔者将在以后的章节仔细分析如何解答此类问题.

面向对象的常见问题

Design a deck of cards that can be used for card game applications.
Design a parking lot.
Create a class design to represent a system.
Design an OO representation for given object.

脚本和正则表达式的常见问题

How do you take a single line of input from the user in a shell script?

Write a script to convert all DOS style backslashes to UNIX style slashes in a list of files.
Write a regular expression (or sed script) to replace all occurrences of the letter 'f', followed by any number of characters, followed by the letter 'a', followed by one or more numeric characters, followed by the letter 'n', and replace what's found with the string "UNIX".
What happens to a child process that dies and has no parent process to wait for it and what's bad about this?
How do you create a swapfile?1. How do you take a single line of input from the user in a shell script?

数据结构常见问题以及解答(**基础书本知识,知识来源 google**)

1) What is data structure?

Data structure refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with data structure, we not only focus on one piece of

data, but rather different set of data and how they can relate to one another in an organized manner.

2) Differentiate file structure from storage structure.

Basically, the key difference is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structure.

3) When is a binary search best applied?

A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is search starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

4) What is a linked list?

A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link of data storage.

5) How do you reference all the elements in a one-dimension array?

To do this, an indexed loop is used, such that the counter runs from 0 to the array size minus one. In this manner, we are able to reference all the elements in sequence by using the loop counter as the array subscript.

6) In what areas do data structures applied?

Data structure is important in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

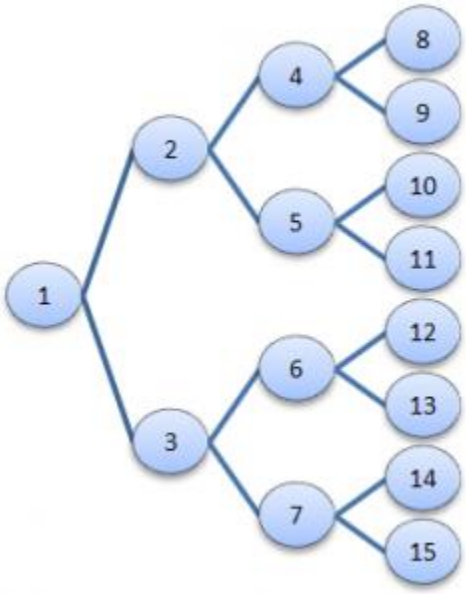
7) What is LIFO?

LIFO is short for Last In First Out, and refers to how data is accessed, stored and retrieved. Using this scheme, data that was stored last , should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

8) What is a queue?

A queue is a data structure that can simulates a list or stream of data. In this structure, new elements are inserted at one end and existing elements are removed from the other end.

9) What are binary trees?



A binary tree is one type of data structure that has two nodes, a left node and a right node. In programming, binary trees are actually an extension of the linked list structures.

10) Which data structure is applied when dealing with a recursive function?

Recursion, which is basically a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

11) What is a stack?

A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

12) Explain Binary Search Tree

A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

13) What are multidimensional arrays?

Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using a single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structure?

It actually depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?

Aside from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?

FIFO is short for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?

An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

18) What is merge sort?

Merge sort takes a divide-and-conquer approach to sorting data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continuous until you have one single sorted list.

19) Differentiate NULL and VOID.

Null is actually a value, whereas Void is a data type identifier. A variable that is given a Null value simply indicates an empty value. Void is used to identify pointers as having no initial size.

20) What is the primary advantage of a linked list?

A linked list is a very ideal data structure because it can be modified easily. This means that modifying a linked list works regardless of how many elements are in the list.

21) What is the difference between a PUSH and a POP?

Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being "pushed" into the stack. On the other hand, a pop denotes data retrieval, and in particular refers to the topmost data being accessed.

22) What is a linear search?

A linear search refers to the way a target key is being searched in a sequential data structure. Using this method, each element in the list is checked and compared against the target key, and is repeated until found or if the end of the list has been reached.

23) How does variable declaration affect memory allocation?

The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

24) What is the advantage of the heap over a stack?

Basically, the heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, memory of the heap can at times be slower when compared to that stack.

25) What is a postfix expression?

A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

26) What is Data abstraction?

Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

27) How do you insert a new item in a binary search tree?

Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

28) How does a selection sort work for an array?

The selection sort is a fairly intuitive sorting algorithm,, though not necessarily efficient. To perform this, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position. The smallest element remaining in the subarray is then located next with subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

29) How do signed and unsigned numbers affect memory?

In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (unsigned 8 bit number has a range 0-255, while 8-bit signed number has a range -128 to +127.

30) What is the minimum number of nodes that a binary tree can have?

A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

31) What are dynamic data structures?

Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

32) In what data structures are pointers applied?

Pointers that are used in linked list have various applications in data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

33) Do all declaration statements result in a fixed reservation in memory?

Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

34) What are ARRAYs?

When dealing with arrays, data is stored and retrieved using an index that actually refers to the element number in the data sequence. This means that data can be accessed in any order. In programming, an array is declared as a variable having a number of indexed elements.

35) What is the minimum number of queues needed when implementing a priority queue?

The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is intended for actual storage of data.

36) Which sorting algorithm is considered the fastest?

There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

37) Differentiate STACK from ARRAY.

Data that is stored in a stack follows a LIFO pattern. This means that data access follows a sequence wherein the last data to be stored will the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

38) Give a basic algorithm for searching a binary search tree.

1. if the tree is empty, then the target is not in the tree, end search
2. if the tree is not empty, the target is in the tree
3. check if the target is in the root item
4. if target is not in the root item, check if target is smaller than the root's value
5. if target is smaller than the root's value, search the left subtree
6. else, search the right subtree

39) What is a dequeue?

A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

40) What is a bubble sort and how do you perform it?

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values "bubble" to the top of the list, while the larger value sinks to the bottom.

41) What are the parts of a linked list?

A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes, with each node being linked in a sequential manner.

42) How does selection sort work?

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

43) What is a graph?

A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs, and are used to connect nodes where data can be stored and retrieved.

44) Differentiate linear from non linear data structure.

Linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks and queues. On the other hand, non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of non linear data structure include trees and graphs.

45) What is an AVL tree?

An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

46) What are doubly linked lists?

Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and other one that links to the previous node.

47) What is Huffman's algorithm?

Huffman's algorithm is associated in creating extended binary trees that has minimum weighted path lengths from the given weights. It makes use of a table that contains frequency of occurrence for each data element.

48) What is Fibonacci search?

Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can greatly reduce the time needed in order to reach the target element.

49) Briefly explain recursive algorithm.

Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

50) How do you search for a target key in a linked list?

To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.

非技术类型的常见问题:

What are you looking for in your next job? What is important to you?

更多的技术挑战;你们公司的技术和项目;对个人最重要的是高质量的产品;和家庭生活;

What is your greatest weakness?

对于这个这个陷阱问题，可以用如下几种方式: 1.分析职位所需技能，然后诚实的告知面试官希望更多地在你公司获取某种体验.2.转化话题为上一个工作中，你有什么提高的，从而告知面试官你怎么从过去的经验获取进步.3.尽量把负面的转化成正面,比如你可以说，你总是对项目的完成很有紧迫感.

样本答案:

1. 我在工作在每一个项目的时候，我更喜欢在最后期限之前就完成
2. 我可能不擅长组织多个项目，但是我用一个时间管理系统极大的帮助和改善了自己的这个问题
3. 我以前总是在最后时刻才去规划下一周的事情，我现在习惯了提前规划
4. 我过于追求完美.
5. 我过去总是习惯于一次只完成一个项目，现在我尝试和习惯了在一个时间内完成多个项目.

What is your greatest strength?

1. 技术能力
2. 合作能力
3. 项目时间管理能力

Describe a typical work week.

1. 计划新的项目
2. 总结上一周得失
3. 分析可能问题和障碍
4. 和组员以及管理者交流
5. 项目研发

How would you describe the pace at which you work?

1. 充满挑战
2. 快速的进展
3. 准确的分析

How do you handle stress and pressure?

1. 散步
2. 深度分析压力来源
3. 和同事交流

What motivates you?

1. 热爱编程
2. 热爱项目
3. 对公司有爱

Tell me about yourself.

1. 有技术
2. 善于合作
3. 敢于承担责任

Questions about your career goals.

1. 成为该领域的专家
2. 帮助公司在这方面有所提高
3. 减少花费增加效率

What type of work environment do you prefer?

1. 有活力
2. 有挑战
3. 和谐相处

(3.2)电话面试技巧

想要在电话面试中表现出色，以下有一些好建议：

Do:

你应该这么做：

1.Prepare yourself for the call as you would for a face-to-face interview.

你怎么准备“面对面”面试的，就怎么准备电话面试。

2.Choose a quiet place to take the call, with no risk of interruption or background noise.

找一个安静的地方接受电话面试，确保不会被打断，也不要有背景杂音。

3.Pay as much attention to listening as to speaking.

听清楚问题和好好回答问题一样重要。

4.Call from a landline to minimise the chance of interference or lost connection.

用固定电话进行电话面试，以避免发生信号中断或信号干扰。

5.Take things slowly –there is no need to rush.

慢慢来，没有什么事需要你急着去做的。

Don't:

你不应该这么做：

1.Do something else while talking on the phone –give the call your full attention.

边打电话边做其他事情——你应该专心致志地进行电话面试。

2.Fail to take the interview seriously –it's your opportunity to make a good impression.

不把电话面试当回事 ——这是你给招聘人留下好印象的机会。

3.Forget to listen and respond to the interviewer's questions.

忘记认真聆听提问，也不记得回答面试官的问题。

4.Portray a poor telephone manner –remember, you are trying to impress.

不注重电话礼仪——记住，你要想着给面试官留下好印象。

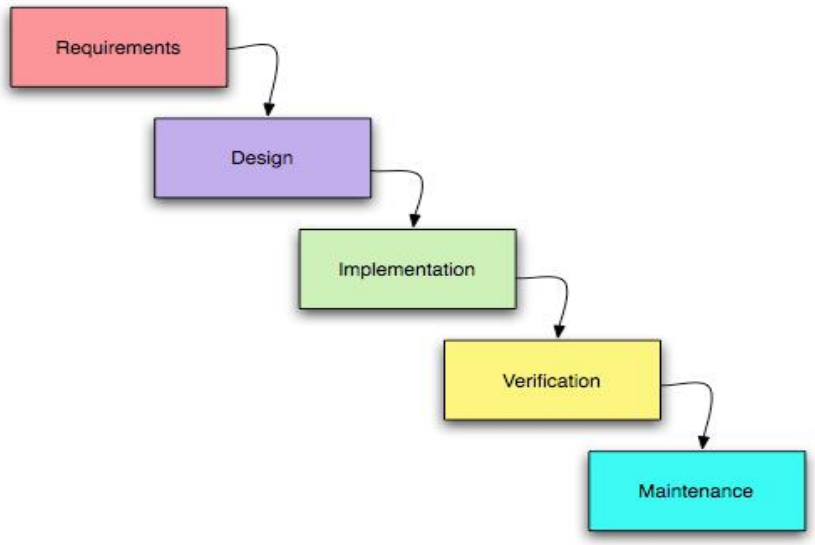
5.Exhibit a lack of enthusiasm –the interviewer expects to hear that you want this job.

(4.1) 面试意图分析

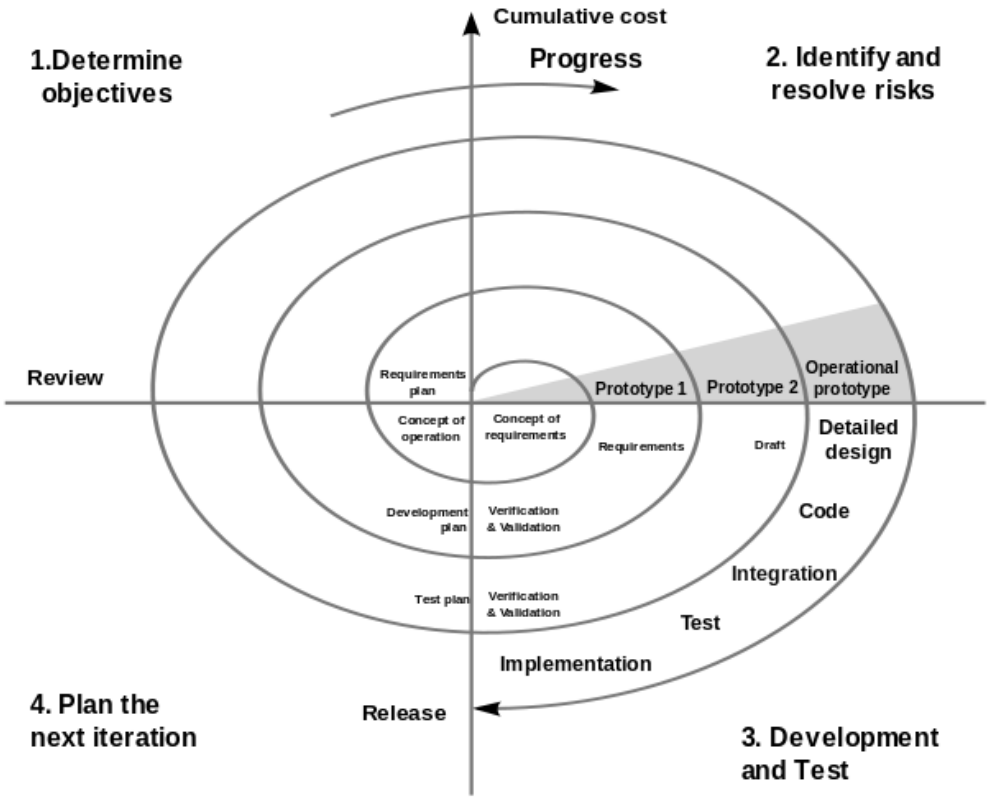
首先，面试官的不同背景，会导致面试官在面试过程中会有不同的侧重，面试官也不是完人,所以面试官也会有知识漏洞，怎么样在这样的面试过程中，能够保持通畅的交流和更准确全面的表达自我，以及在对方可能有多位候选人的情况，怎么样回答出问题，回答好问题，并且能回答出对方没有想到的精彩答案，这一切都是在于你要对编程有更深层次的理解.

首先你需要了解现代软件业的基本研发过程(几种基本研发类型):

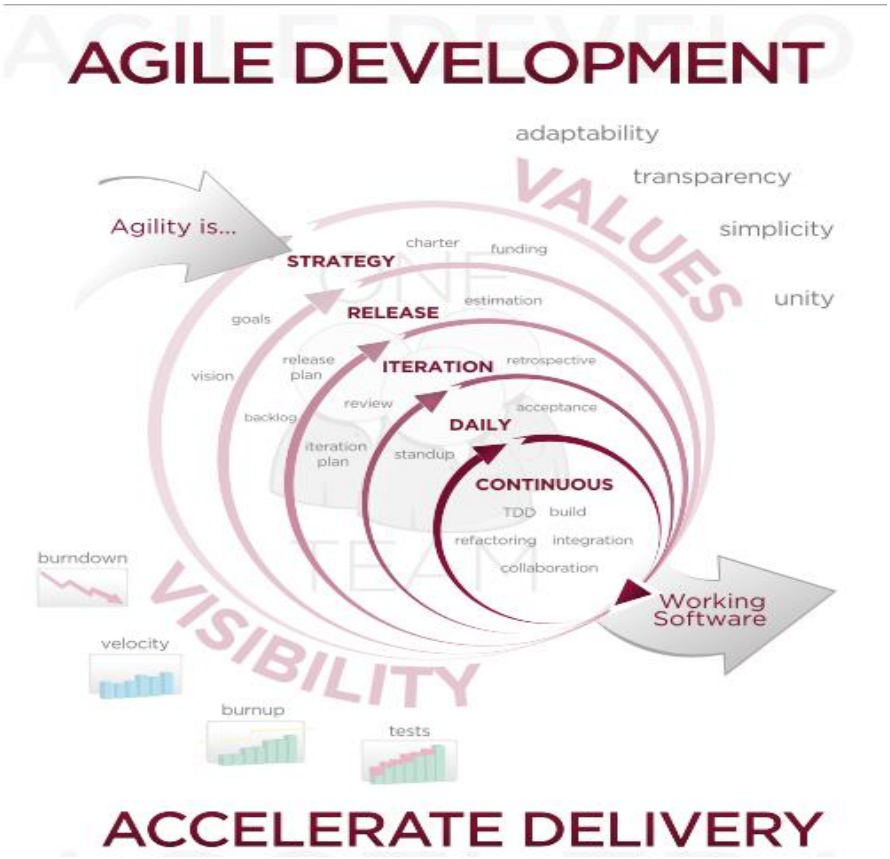
water fall model



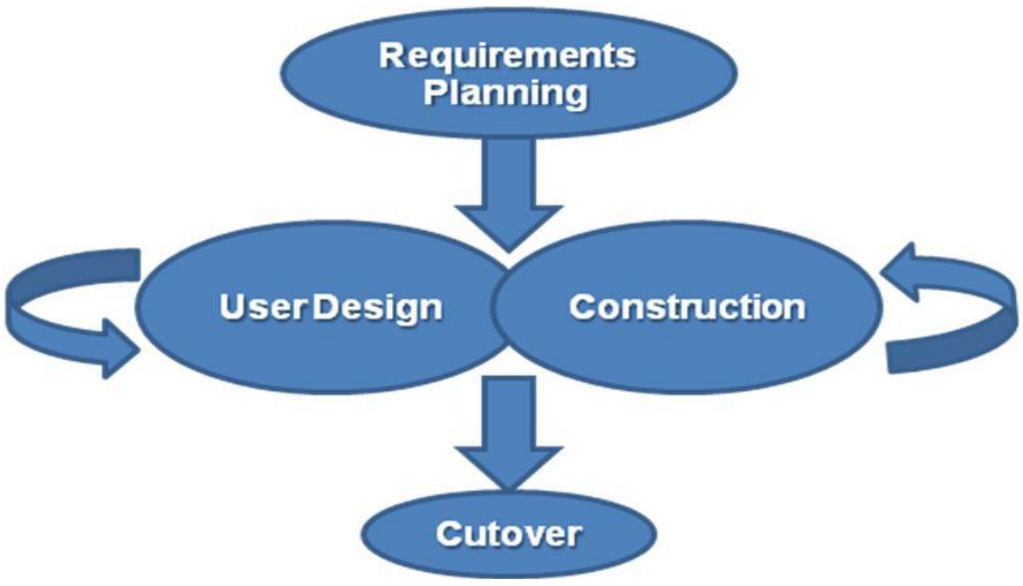
spiral model



Agile development

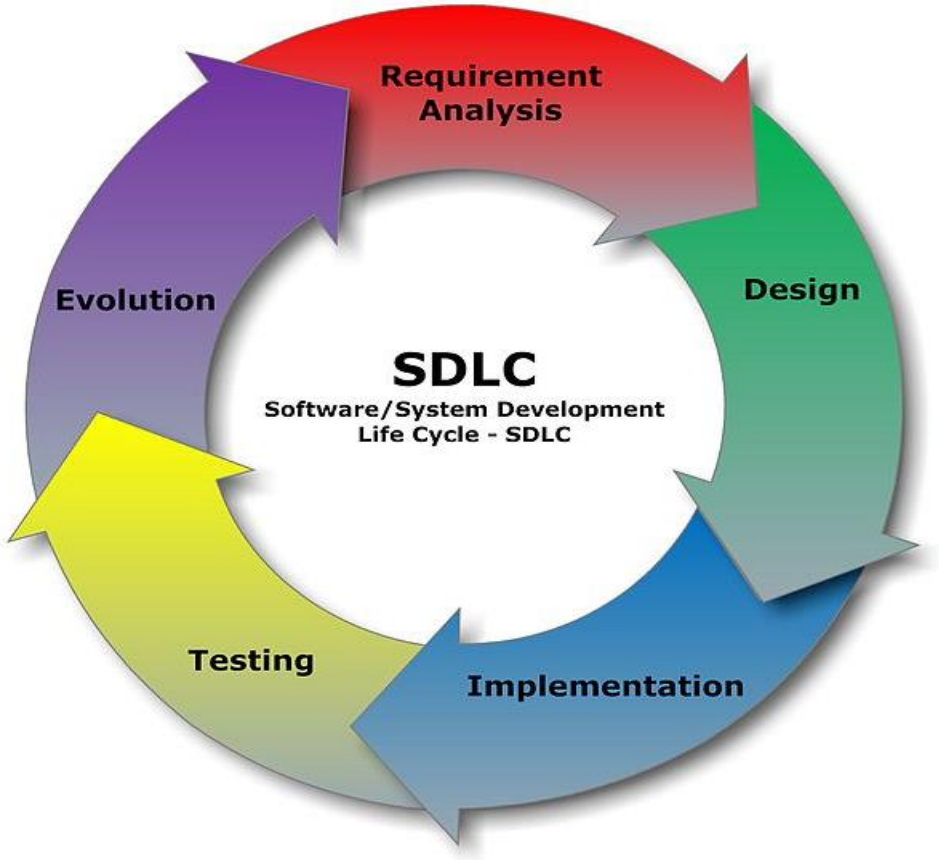


Rapid application development



各大高科技公司以及下面的项目组因为侧重点不同，他们会引入了不同的研发模式，作为候选人，应该首先了解公司的产品结构以及为什么会采取这种研发模式.甚至在微软,更多地小组采用了 Agile 的研发模式.

在对产品研发过程有过了解之后，你可以清楚地了解你所面试的角色,编程师在整个研发过程中的位置和作用.



在候选人通过了前 20 分钟简历讨论，以及个人介绍，和对公司产品的理解后，我们来到了编程的环节,下一个章节，我们将高强度讨论如何写出好的程序,以及洞察面试官的意图.

(4.2)常见题目分析

在这个章节中，笔者将详细列举不同题目，进行高强度分析.

与其余面试书不同的之处，笔者用详细的问答来解释这个题目的意义, 以及最佳的回答方式.

每一道题目不管难易如何，都需要跟从软件研发的顺序来解决.

1. **Given a binary tree, find its maximum height.（The maximum height of a binary tree is defined as the number of nodes along the path from the root node to the deepest leaf node. Note that the maximum height of an empty tree is 0.）**

作为候选人，第一要点，必须仔细聆听对方题目的要求 (研发过程中的 PM feature design 阶段)
根据题目，你需要询问以下问题 (假设你是和 PM 讨论，搞清楚产品需求)，和在纸上,白板上或者电脑上, 写下如下 code.
二叉树的定义:

```
struct BTree{
      int data;
      Tree* Right;
      Tree* Left;
     };
```

二叉树的高度定义:
The maximum height of a binary tree is defined as the number of nodes along the path from the root node to the deepest leaf node. Note that the maximum height of an empty tree is 0.

对方对编程语言和风格有何要求
c/c++ or java ?

举出几个边缘例子，对方怎么处理

空头指针，如果二叉树有循环指针，怎么处理

写一个方程 signature ,寻求是否同意

讨论可用的解答, 可以画出一个二叉树,然后手动的走一遍，看如何用机器语言实现

如果意识到自己可以用递归的方法解决， 什么是我的解决方案 (Dev 的 implementation 阶段)

自己方法的时间和空间的复杂度问题 O(n)

清晰地注释

/*

Find the Maximum Height (Depth) of a Binary Tree
1

/\ return 2, null head point return 0;

2 3

*/

```
int maxHeight(BTree *p) {
      //protection code, validate input
      if (!p) return 0;
      int left_height = maxHeight(p->Left);
      int right_height = maxHeight(p->Right);
      return (left_height > right_height) ? left_height + 1 : right_height + 1;
}
```

在写好程序之后，递交之前，先运行一个到两个测试案例,这是软件研发中 Unit testing 的部分.

测试部分就是一个小的 mini-test plan

询问测试所在的 level,和应用范围, white box, black box etc..

测试案例的意义在于用最少的测试案例覆盖最多的测试可能性(类似于 code coverage)

在给出测试案例时,需要按照不同的测试类型分类

同时列举 实例按照该模板 输入 ,输出，期待结果, 成功/失败/警告

1. unit testing
   1.1 空指针
   1.2 一个节点
   1.3 两个节点（左节点，或者右节点）
   1.4 三个节点(左右平衡分部或者分布在一边)
   1.5 大量节点
   1.6 节点中有回环
2. performance testing
   有时间和空间的需求对比
3. security testing
   潜在的多线程安全考虑
4. smoke testing
5. etc.

然后你可以进阶写下如下代码:

```
int maxHeight(BTree *p) {
    lock(this.write()){
      if (!p) return 0;
      int left_height = maxHeight(p->Left);
      int right_height = maxHeight(p->Right);
      return (left_height > right_height) ? left_height + 1 : right_height + 1;
     }
}
```

请读者告诉我为什么这么写

整个题目的分析过程，准确的涵盖了软件研发过程中每一步的需求,以及充分满足了对方所需要的信息量以及交流的重要性.
充分体现出了候选人的知识技能和交流能力以及对整个软件研发流程的理解.

2. **How To Reverse a single Linked List**

这个问题是非常简单和非常经典的，当对方问你这个问题的时候，你是否知道以下知识？

- Linked lists are building blocks for many other data structures like stacks and queues.
- Linked lists are a sequence of nodes containing data fields and pointers to the next node (or) next node and previous nodes based on its type.
- Linked lists permit addition/ removal of node in constant time.
- Unlike arrays the order of linked list elements need not be contiguous in memory.
- Unlike arrays there is no upper limit on the amount of memory reserved.
- A singly linked list is the simplest of linked lists.
- Each node of a singly linked list has data elements and a single link (pointer) that points to the next node of the list (or) NULL if it is the last node of the list.
- Addition/ Deletion of a node to the singly linked list involves the creation/ deletion of the node and adjusting the node pointers accordingly.
- A singly linked list could be represented as below:-

在你开始讨论你怎么写代码的时候，准备好对方可能会问你这些问题.

写下最简单的结构体

```
struct node {
    int x;
    node *next;
};
```

询问对方对方程式 signature 有何要求

编程风格有何限制

画出一个 list,然后手动走一遍

思考几分钟....

提出一到几个方案, 各个方案的空间，时间性能要求.

对几个特例的讨论，如果 list 内部有循环如何处理 (返回 error code 还是抛出 exception),你要对你说的每一个可能性，都了解为什么要这样做.

假设你有方法一:

/*

You need to point to the current node (the one you're handling), the previous node (so you can point back to it), and the next node (so you can move on to it after you've disrupted your next pointer).

*/

```
    node *reverse(node *head)
{
    node *previous = NULL;

    while (head != NULL) {
        // Keep next node since we trash
        // the next pointer.
        node *next = head->next;

        // Switch the next pointer
        // to point backwards.
        head->next = previous;

        // Move both pointers forward.
        previous = head;
        head = next;
    }

    return previous;
}
```

在写好程序之后，递交之前，先运行一个到两个测试案例,这是软件研发中 Unit testing 的部分.

测试案例的意义在于用最少的测试案例覆盖最多的测试可能性(类似于 code coverage)

在给出测试案例时,需要按照不同的测试类型分类

6. unit testing
   6.1 空节点
   6.2 一个节点
   6.3 两个节点
   6.4 三个节点
   6.5 大量节点
   6.6 节点中有回环
7. performance testing

有时间和空间的需求对比

8. security testing
   潜在的多线程安全考虑
9. smoke testing
10. etc.

3. **Implement `atoi` to convert a string to an integer.**
   这道经典题目的挑战之处在于怎么处理多种特别例子，如果有非法字符，如果大小越界这一些都需要在面试中和面试官进行讨论

```
int atoi(const char *str) {
     long result = 0;
     bool sign = true;
     for(;*str && *str==' ';++str);
     if(*str=='-'){
         sign=false;
         ++str;
     }
       else if(*str=='+') ++str;
       while(*str){
       if(isdigit(*str)){
            result=result*10+*str-'0';
           if(sign&&result>INT_MAX) return INT_MAX;
           else if(!sign&&-result<INT_MIN) return INT_MIN;
           str++;
     }
     else break;
   }
   if(sign) return result;
   else return -result;
}
```

4. **Insert a node into a sorted singly linked list, and return the pointer to the new linked list**

   这道题目测试了候选者对基本数据结构以及对应操作的理解，请问读者您还能对这个 code 用什么提高?

```
Node *SortedInsert(Node *head, Node *newNode )
{
    if(!newNode) return head;

    Node *Parent = NULL;
    Node *Current = head;

    while (Current && Current->data < newNode->data)
    {
        Parent = Current;
        Current = Current->next;
    }
```

```
        if(!Parent)
        {
                Parent = newNode;
                Parent->next = Current;

                return Parent;
        }

        Parent->next = newNode;
        newNode->next = Current;

        return head;
}
```

5. Given a string, returns the word with the maximum count

作者给出一个 c++版本的样本，读者思考过，如果字符串过长，超出 Map 的可接受范围，我们还有更好的解答吗?

```
// CountWord.cpp : Defines the entry point for the console application.


#include "stdafx.h"
#include <map>
#include <string>
#include <algorithm>
#include <ctype.h>

typedef std::map<std::string, int> WordMap;

// Given a string, returns the word with the maximum count
std::string CountWord(std::string String, int& MaxCount)
{
   char *delim = " .?!;,\t\r\n";
   WordMap wordmap;
   WordMap::iterator it;

   // Transform all characters to lower case
   std::transform(String.begin(), String.end(), String.begin(), tolower);

   // Make a copy of the string, as strtok would replace all delimiters to 0
   char* Str = strdup(String.c_str());

   // Find the first word
   char *Token = strtok(Str, delim);
```

```
   while(Token)
   {
      it = wordmap.find(Token);

      if(it == wordmap.end())
      {
         // Set the count of the word to 1 if not found in map
         wordmap[Token] = 1;
      }else
      {
         // Increase the count of the word by 1 if found in map
         it->second++;
      }

      Token = strtok(NULL, delim);
   }

   std::string MaxWord;

   it = wordmap.begin();

   MaxCount = 0;

   // Traverse the map and find the word with the maximum count
   while(it != wordmap.end())
   {
      std::string word = it->first;
      int count = it->second;

      if(it->second > MaxCount)
      {
         MaxWord = it->first;
         MaxCount = it->second;
      }

      it++;
   }

   free(Str);
   return MaxWord;
}

int _tmain(int argc, _TCHAR* argv[])
{
   char Str[] = "abc, ah ha! This is jack. Jack is back\n. Jack is doing interview. \tJack wants  it so badly.
Can you give me a   position? abc a This this FTE position?....ah ha!..\n";

   printf("Input string is:\n");
```

```
printf("================================================================================
==============\n");
   printf("%s\n", Str);

printf("================================================================================
==============\n");

   int MaxCount;
   std::string MaxWord = CountWord(Str, MaxCount);

   printf("The word with most count is: %s (%d)\n", MaxWord.c_str(), MaxCount);

   return 0;
}
```

6. Design Patterns 设计范例 (资源来自面向对象设计的网站)

在大家大脑有点疲劳的时候，给大家看看几种非常流行常用的 设计范例 以及讨论一下软件设计的原则

软件设计原则是什么？

软件的设计原则是指一组指导原则，可以帮助我们避免一个糟糕的设计。 一个不好的设计应避免的 3 个重要的特点：

绝对刚性 - 这是很难改变太多，因为每一个变化会影响系统的其他部分。
脆弱性 - 当你做出改变，意外的系统中断。
固定性 - 这是很难在另一个应用程序中重复使用，因为它不能从当前的应用程序获得释放。

设计开放封闭的原则 ，如类，模块和功能的软件实体应当对扩展开放，而不是封闭的修改。以确保当你需要扩展自己的行为，你没有改变类，但把它扩大。同样的原则也适用于模块，包库。如果你有一个库，其中包含了一组类的原因有很多，你会喜欢来扩展它在不改变已经写的代码（向后兼容，回归测试）。这就是为什么我们必须确保我们的模块遵循开放封闭原则。

我们需要可以保证使用抽象类和具体类实现他们的行为。这将执行具体的类，抽象类的扩展，而不是改变他们。这是某些特殊情况下的模板模式和策略模式。

高层模块不应该依赖于底层模块。双方应该依赖于抽象的定义。
抽象不应该依赖于细节。细节应该依赖于抽象。

我们应该从低级别的模块分离高层次模块，引进的高层次类和低层次的类之间的抽象层。 而不是

写我们的抽象的细节，我们应该写的基础上抽象细节。

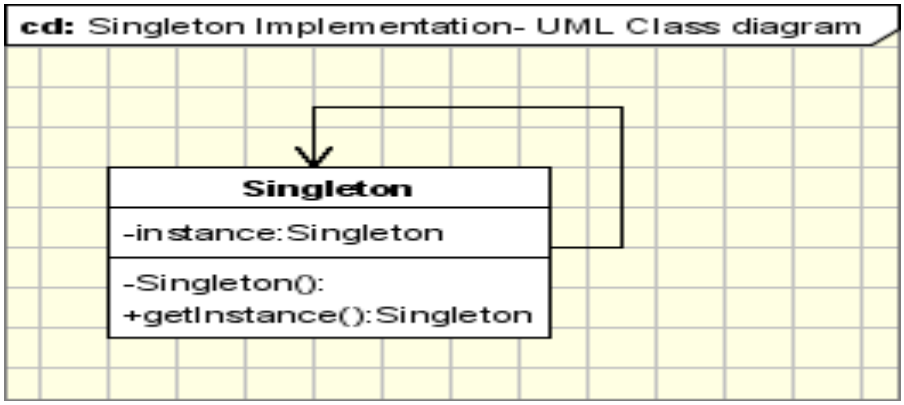
在传统的方式，当一个软件模块（类，框架），需要一些其他的模块，它初始化，并保持它的直接引用。这将使得 2 个模块的紧耦合。为了去耦所述第一模块将提供一个钩子（属性，参数）和一个外部模块控制依赖将注入参照第二个。客户端不应该被迫依赖他们不使用的接口。

设计原则告诉我们如何照顾我们写我们的接口。当我们写我们的接口中，我们应该注意添加唯一的方法，应该是有。如果我们添加的方法不应该有实现该接口的类，以及实现这些方法。例如，如果我们创建了一个叫工人的接口，并添加一个方法，午休时间，所有的工人将要实现它。
作为一个结论接口包含的方法， 我们应该避免使用它们。

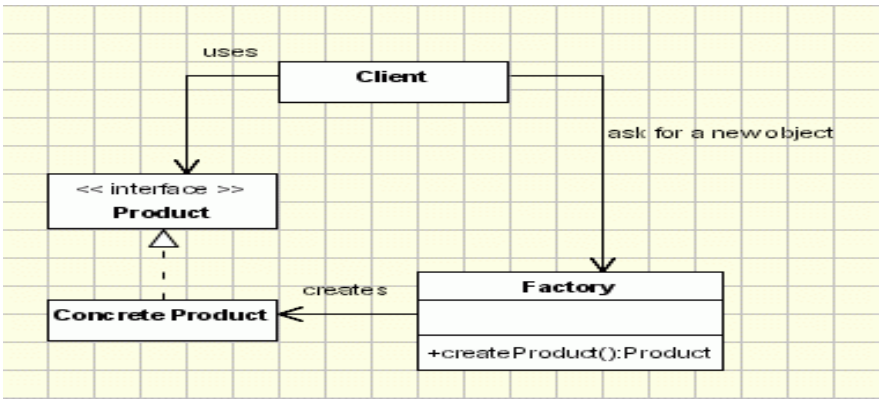
设计的类只应该有一个理由去改变。在这种情况下，责任被认为是改变的原因之一。这条原则指出，如果我们改变一类有 2 个原因，我们有两班分裂的功能。每一个类只处理一个责任和未来，如果我们需要做一个改变，我们要让它在课堂上处理。当我们需要做出改变的一类具有更多的职责的变化可能会影响其他功能的类。

我们看一下接下来几张图表深刻理解一下几种基本的设计范例.

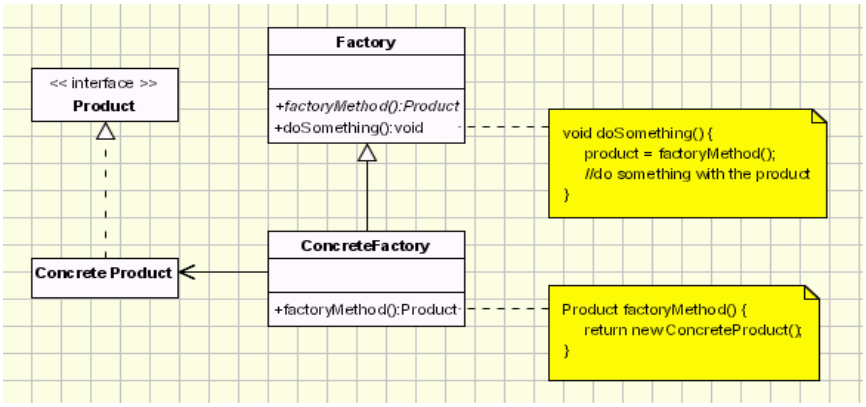
Singleton - Ensure that only one instance of a class is created and Provide a global access point to the object.



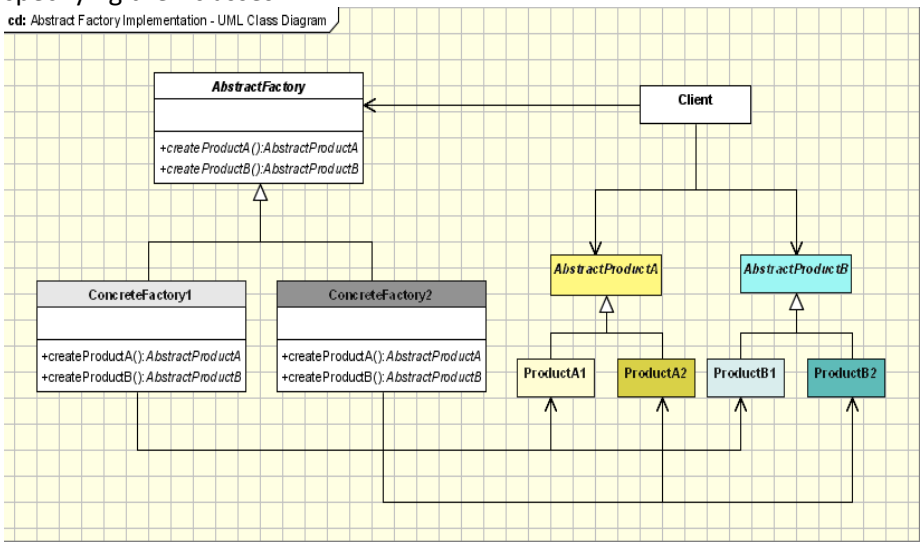
Factory(Simplified version of Factory Method) - Creates objects without exposing the instantiation logic to the client and Refers to the newly created object through a common interface.



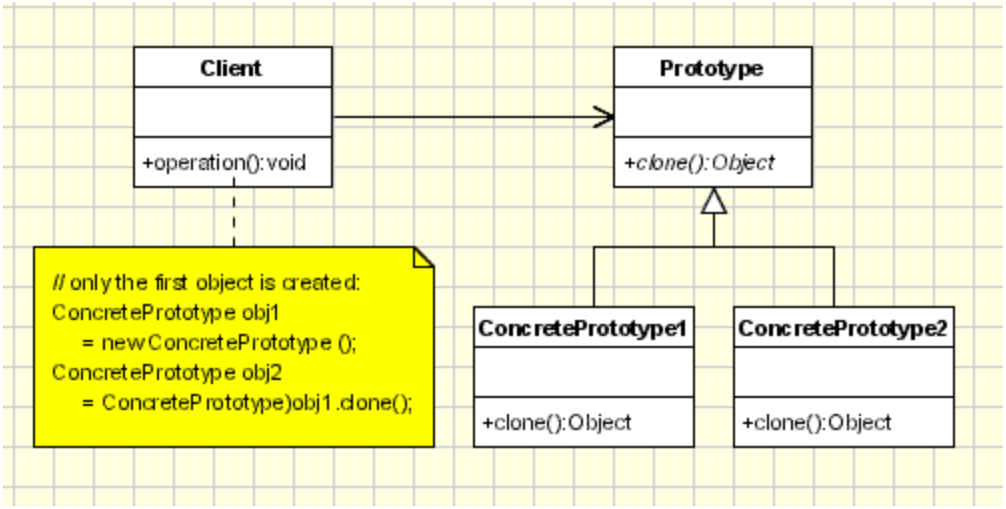
Factory Method - Defines an interface for creating objects, but let subclasses to decide which class to instantiate and Refers to the newly created object through a common interface.



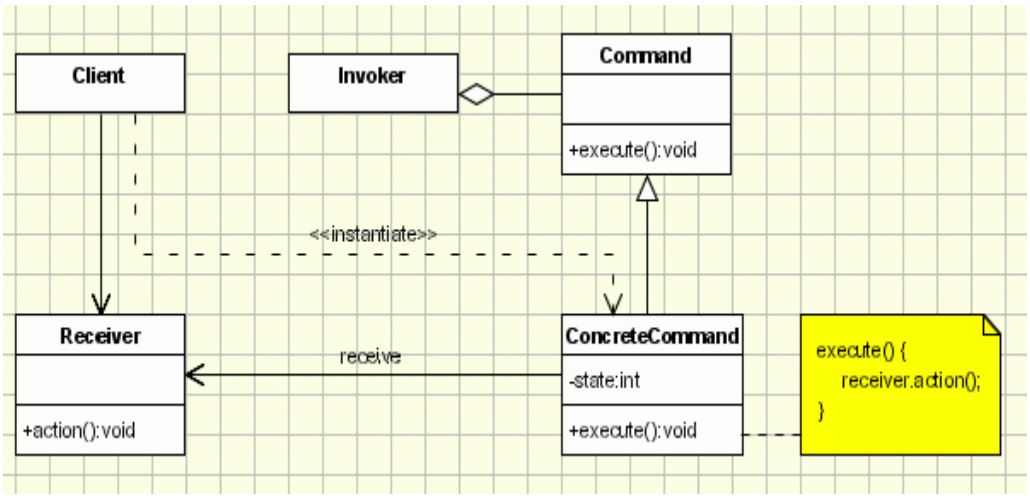
Abstract Factory - Offers the interface for creating a family of related objects, without explicitly specifying their classes.



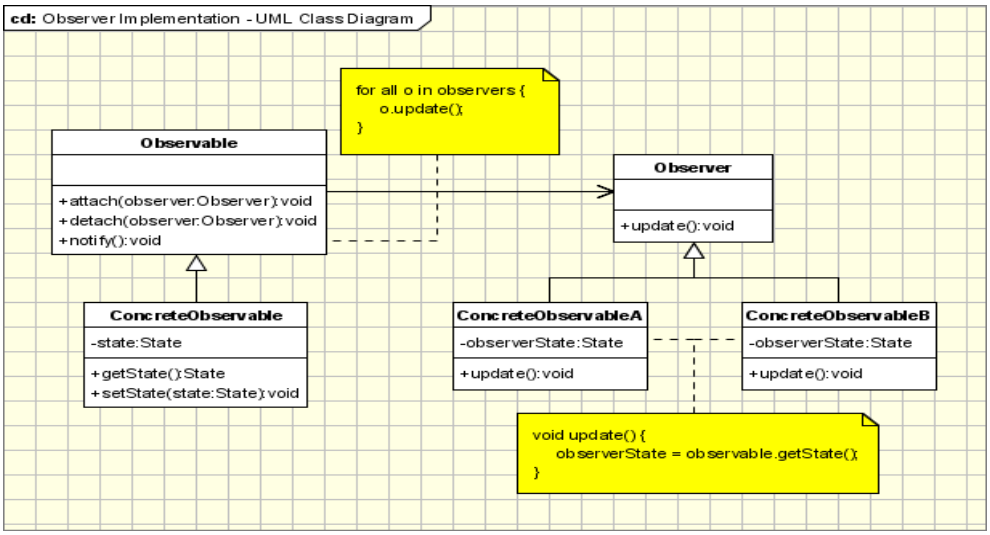
Prototype - Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.



Command - Encapsulate a request in an object, Allows the parameterization of clients with different requests and Allows saving the requests in a queue.



Observer - Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.



7. how to design a parking lot

那我们来牛刀小试，做做这道 amazon 的经典设计题目吧.请读者在阅读解答后自行补充读者认为更重要的部分.
我们遵循的停车场设计原则中的四个步骤：

功能:

停车位有三种类型：摩托车，汽车，残疾人车。摩托车和汽车只能停在指定的停车位，残疾人车可以停在自己的公园或普通汽车的。

任何车只能停到它的类型在停车场的空间。可用的停车位，每个类型都保持在停车场的对象。

每个车位的权限系统：要么支付到停车场或免费停车。

对象以及对应的类

停车场一个类
停车空间是一类
停车场是具有有限数量的停车空间的对象

没有必要非得为停车空间每种类型的的子类，因为它们共享通用停车空间相同的操作。被确定的停车空间对象中的成员变量的类型。

汽车类和它的子类 --实现每种类型的车辆。

接口权限是一个通用的支付系统。允许实施支付 给停车场或者免费停车.

需要一个类成员记录停车记录的开始和结束时间。

权限及责任

停车场把保持空置车位的阵列，每一种类型的空间。需要记录占用的资料在每个停车对象中。

定义类的车辆提供公共停车操作的方法 ，每个子类中实现接口，它的类型将车停入停车位的车辆。

停车场 有 访问和修改的操作，它是用来作为存储，以维持各类型的停车位。 是负责停车位的分配和回收。适用于这两种方法来获取停车费的.

读者可以参看网上一个设计代码具体实现的示例.

```
public interface Permission{
   float getFee(Calendar start, Calendar end);
}

class ParkingLot{

   private List&lt;ParkingSpace&gt; freeRegularSpace;
   private List&lt;ParkingSpace&gt; freeHandicappedSpace;
   private List&lt;ParkingSpace&gt; freeCompactSpace;
   public ParkingLot();

   public ParkingSpace allocateFreeSpace(ParkingSpaceType pspaceType)
   {
      //get a ParkingSpace from the corresponding free list
      pspace.setStart();
      return pspace;
   }

   public float reclainFreeSpace(ParkingSpace pspace){

      pspace.setEnd();

      //return free space to the list
      return pspace.getFee(perm);
   }

   private Permission perm;
}

class ParkingSpace
{
   enum ParkingSpaceType
   {
      MOTORBIKE, CAR, HANDICAPPED;
   }

   private int id;
   private ParkingSpaceType pspaceType;
   private ParkingMeter meter;

   public ParkingSpace(int id, ParkingSpaceType pspaceType)
   {
```

```
      super();
      this.id = id;
      this.pspaceType = pspaceType;
   }

   public int getId()
   {
      return id;
   }

   public ParkingSpaceType getpspaceType()
   {
      return pspaceType;
   }

   private class ParkingMeter{
      public GregorianCalendar start;
      public GregorianCalendar end;
   }

   public void setStart()
   {
      meter.start = new GregorianCalendar();
   }
   public void setEnd()
   {
      meter.end = new GregorianCalendar();
   }

   public float getFee(Permision perm)
   {
      return perm.getFee(meter.start, meter.end);
   }
}

public abstract class Vehicle{
   public boolean park(ParkingLot pLot);
   public boolean unpark(ParkingLot pLot){
      if(pspace != null){
         pLot.reclaimFreeSpace(pspace);
         return true;
      }
      return false;
   };
   private ParkingSpace pspace;
}

public class Motorbike extends Vehicle{
```

```
  public boolean park(ParkingLot pLot){
    if((pspace=pLot.allocateFreeSpace(ParkingSpaceType.MOTORBIKE))==null){
      return false;
    }
    return true;
  }
}

public class Cars extends Vehicle{
  public boolean park(ParkingLot pLot){
    if((pspace=pLot.allocateFreeSpace(ParkingSpaceType.CAR))==null){
      return false;
    }
    return true;
  }
}

public class HandicappedCars extends Vehicle{
  public boolean park(ParkingLot pLot){
    if((pspace=pLot.allocateFreeSpace(ParkingSpaceType.HANDICAPPED))==null &&
(pspace=pLot.getfreeSpace(ParkingSpaceType.CAR))==null){
      return false;
    }
    return true;
  }
}
```

8. **大数据量，海量数据 处理方法总结（出处：http://bbs.xjtu.edu.cn ;phylips@bmy）**

大数据量的问题是很多面试笔试中经常出现的问题，比如 google ,amazon 这样的一些涉及到海量数据的公司经常会问到。

下面的方法是对海量数据的处理方法进行了一个一般性的总结，当然这些方法可能并不能完全覆盖所有的问题，但是这样的一些方法也基本可以处理绝大多数遇到的问题。下面的一些问题基本直接来源于公司的面试笔试题目，方法不一定最优，如果读者有更好的处理方法，欢迎讨论。

1.Bloom filter

适用范围：可以用来实现数据字典，进行数据的判重，或者集合求交集

基本原理及要点：
对于原理来说很简单，位数组+k 个独立 hash 函数。将 hash 函数对应的值的位数组置 1，查找时如果发现所有 hash 函数对应位都是 1 说明存在，很明显这个过程并不保证查找的

结果是 100%正确的。同时也不支持删除一个已经插入的关键字，因为该关键字对应的位会牵动到其他的关键字。所以一个简单的改进就是 counting Bloom filter，用一个 counter 数组代替位数组，就可以支持删除了。

还有一个比较重要的问题，如何根据输入元素个数 n，确定位数组 m 的大小及 hash 函数个数。当 hash 函数个数 k=(ln2)*(m/n)时错误率最小。在错误率不大于 E 的情况下，m 至少要等于 n*lg(1/E)才能表示任意 n 个元素的集合。但 m 还应该更大些，因为还要保证 bit 数组里至少一半为 0，则 m 应该>=nlg(1/E)*lge 大概就是 nlg(1/E)1.44 倍(lg 表示以 2 为底的对数)。

举个例子我们假设错误率为 0.01，则此时 m 应大概是 n 的 13 倍。这样 k 大概是 8 个。

注意这里 m 与 n 的单位不同，m 是 bit 为单位，而 n 则是以元素个数为单位(准确的说是不同元素的个数)。通常单个元素的长度都是有很多 bit 的。所以使用 bloom filter 内存上通常都是节省的。

扩展：
Bloom filter 将集合中的元素映射到位数组中，用 k（k 为哈希函数个数）个映射位是否全 1 表示元素在不在这个集合中。Counting bloom filter（CBF）将位数组中的每一位扩展为一个 counter，从而支持了元素的删除操作。Spectral Bloom Filter（SBF）将其与集合元素的出现次数关联。SBF 采用 counter 中的最小值来近似表示元素的出现频率。

问题实例：给你 A,B 两个文件，各存放 50 亿条 URL，每条 URL 占用 64 字节，内存限制是 4G，让你找出 A,B 文件共同的 URL。如果是三个乃至 n 个文件呢？

根据这个问题我们来计算下内存的占用，4G=2^32 大概是 40 亿*8 大概是 340 亿，n=50 亿，如果按出错率 0.01 算需要的大概是 650 亿个 bit。现在可用的是 340 亿，相差并不多，这样可能会使出错率上升些。另外如果这些 urlip 是一一对应的，就可以转换成 ip，则大大简单了。

2.Hashing

适用范围：快速查找，删除的基本数据结构，通常需要总数据量可以放入内存

基本原理及要点：
hash 函数选择，针对字符串，整数，排列，具体相应的 hash 方法。
碰撞处理，一种是 open hashing，也称为拉链法；另一种就是 closed hashing，也称开地址法，opened addressing。

扩展：

d-left hashing 中的 d 是多个的意思，我们先简化这个问题，看一看 2-left hashing。2-left hashing 指的是将一个哈希表分成长度相等的两半，分别叫做 T1 和 T2，给 T1 和 T2 分别配备一个哈希函数，h1 和 h2。在存储一个新的 key 时，同时用两个哈希函数进行计算，得出两个地址 h1[key]和 h2[key]。这时需要检查 T1 中的 h1[key]位置和 T2 中的 h2[key]位置，哪一个位置已经存储的（有碰撞的）key 比较多，然后将新 key 存储在负载少的位置。如果两边一样多，比如两个位置都为空或者都存储了一个 key，就把新 key 存储在左边的 T1 子表中，2-left 也由此而来。在查找一个 key 时，必须进行两次 hash，同时查找两个位置。

问题实例：
1).海量日志数据，提取出某日访问百度次数最多的那个 IP。

IP 的数目还是有限的，最多 2^32 个，所以可以考虑使用 hash 将 ip 直接存入内存，然后进行统计。

3.bit-map

适用范围：可进行数据的快速查找，判重，删除，一般来说数据范围是 int 的 10 倍以下

基本原理及要点：使用 bit 数组来表示某些元素是否存在，比如 8 位电话号码

扩展：bloom filter 可以看做是对 bit-map 的扩展

问题实例：

1)已知某个文件内包含一些电话号码，每个号码为 8 位数字，统计不同号码的个数。

8 位最多 99 999 999，大概需要 99m 个 bit，大概 10 几 m 字节的内存即可。

2)2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

将 bit-map 扩展一下，用 2bit 表示一个数即可，0 表示未出现，1 表示出现一次，2 表示出现 2 次及以上。或者我们不用 2bit 来进行表示，我们用两个 bit-map 即可模拟实现这个 2bit-map。

4.堆

适用范围：海量数据前 n 大，并且 n 比较小，堆可以放入内存

基本原理及要点：最大堆求前 n 小，最小堆求前 n 大。方法，比如求前 n 小，我们比较当

前元素与最大堆里的最大元素，如果它小于最大元素，则应该替换那个最大元素。这样最后得到的 n 个元素就是最小的 n 个。适合大数据量，求前 n 小，n 的大小比较小的情况，这样可以扫描一遍即可得到所有的前 n 元素，效率很高。

扩展：双堆，一个最大堆与一个最小堆结合，可以用来维护中位数。

问题实例：
1)100w 个数中找最大的前 100 个数。

用一个 100 个元素大小的最小堆即可。

5.双层桶划分

适用范围：第 k 大，中位数，不重复或重复的数字

基本原理及要点：因为元素范围很大，不能利用直接寻址表，所以通过多次划分，逐步确定范围，然后最后在一个可以接受的范围内进行。可以通过多次缩小，双层只是一个例子。

扩展：

问题实例：
1).2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

有点像鸽巢原理，整数个数为 2^32,也就是，我们可以将这 2^32 个数，划分为 2^8 个区域(比如用单个文件代表一个区域)，然后将数据分离到不同的区域，然后不同的区域在利用 bitmap 就可以直接解决了。也就是说只要有足够的磁盘空间，就可以很方便的解决。

2).5 亿个 int 找它们的中位数。

这个例子比上面那个更明显。首先我们将 int 划分为 2^16 个区域，然后读取数据统计落到各个区域里的数的个数，之后我们根据统计结果就可以判断中位数落到那个区域，同时知道这个区域中的第几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域中的那些数就可以了。

实际上，如果不是 int 是 int64，我们可以经过 3 次这样的划分即可降低到可以接受的程度。即可以先将 int64 分成 2^24 个区域，然后确定区域的第几大数，在将该区域分成 2^20 个子区域，然后确定是子区域的第几大数，然后子区域里的数的个数只有 2^20，就可以直接利用 direct addr table 进行统计了。

6.数据库索引

适用范围：大数据量的增删改查

基本原理及要点：利用数据的设计实现方法，对海量数据的增删改查进行处理。
扩展：
问题实例：

7.倒排索引(Inverted index)

适用范围：搜索引擎，关键字查询

基本原理及要点：为何叫倒排索引？一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。

以英文为例，下面是要被索引的文本：
T0 = "it is what it is"
T1 = "what is it"
T2 = "it is a banana"
我们就能得到下面的反向文件索引：
"a": {2}
"banana": {2}
"is": {0, 1, 2}
"it": {0, 1, 2}
"what": {0, 1}
检索的条件"what", "is" 和 "it" 将对应集合的交集。

正向索引开发出来用来存储每个文档的单词的列表。正向索引的查询往往满足每个文档有序频繁的全文查询和每个单词在校验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。

扩展：

问题实例：文档检索系统，查询那些文件包含了某单词，比如常见的学术论文的关键字搜索。

8.外排序

适用范围：大数据的排序，去重

基本原理及要点：外排序的归并方法，置换选择 败者树原理，最优归并树

扩展：

问题实例：
1).有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 个字节，内存限制大小是 1M。返回频数最高的 100 个词。

这个数据具有很明显的特点，词的大小为 16 个字节，但是内存只有 1m 做 hash 有些不够，所以可以用来排序。内存可以当输入缓冲区使用。

9.trie 树

适用范围：数据量大，重复多，但是数据种类小可以放入内存

基本原理及要点：实现方式，节点孩子的表示方式

扩展：压缩实现。

问题实例：
1).有 10 个文件，每个文件 1G， 每个文件的每一行都存放的是用户的 query，每个文件的 query 都可能重复。要你按照 query 的频度排序 。

2).1000 万字符串，其中有些是相同的(重复),需要把重复的全部去掉，保留没有重复的字符串。请问怎么设计和实现？

3).寻找热门查询：查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超过 3 百万个，每个不超过 255 字节。

10.分布式处理 mapreduce

适用范围：数据量大，但是数据种类小可以放入内存

基本原理及要点：将数据交给不同的机器去处理，数据划分，结果归约。

扩展：

问题实例：

1).The canonical example application of MapReduce is a process to count the appearances of

each different word in a set of documents:

```
void map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
    EmitIntermediate(w, 1);

void reduce(String word, Iterator partialCounts):
  // key: a word
  // values: a list of aggregated partial counts
  int result = 0;
  for each v in partialCounts:
    result += ParseInt(v);
  Emit(result);
```

Here, each document is split in words, and each word is counted initially with a "1" value by

the Map function, using the word as the result key. The framework puts together all the pairs

with the same key and feeds them to the same call to Reduce, thus this function just needs to

sum all of its input values to find the total appearances of that word.

2).海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10。

3).一共有 N 个机器，每个机器上有 N 个数。每个机器最多存 O(N)个数并对它们操作。如何找到 N^2 个数的中数(median)？

经典问题分析

上千万 or 亿数据（有重复），统计其中出现次数最多的前 N 个数据,分两种情况：可一次读入内存，不可一次读入。

可用思路：trie 树+堆，数据库索引，划分子集分别统计，hash，分布式计算，近似统计，外排序

所谓的是否能一次读入内存，实际上应该指去除重复后的数据量。如果去重后数据可以放入内存，我们可以为数据建立字典，比如通过 map，hashmap，trie，然后直接进行统计即可。当然在更新每条数据的出现次数的时候，我们可以利用一个堆来维护出现次数最多的前 N 个数据，当然这样导致维护次数增加，不如完全统计后在求前 N 大效率高。

如果数据无法放入内存。一方面我们可以考虑上面的字典方法能否被改进以适应这种情形，可以做的改变就是将字典存放到硬盘上，而不是内存，这可以参考数据库的存储方法。

当然还有更好的方法，就是可以采用分布式计算，基本上就是 map-reduce 过程，首先可以根据数据值或者把数据 hash(md5)后的值，将数据按照范围划分到不同的机子，最好可以让数据划分后可以一次读入内存，这样不同的机子负责处理各种的数值范围，实际上就是 map。得到结果后，各个机子只需拿出各自的出现次数最多的前 N 个数据，然后汇总，选出所有的数据中出现次数最多的前 N 个数据，这实际上就是 reduce 过程。

实际上可能想直接将数据均分到不同的机子上进行处理，这样是无法得到正确的解的。因为一个数据可能被均分到不同的机子上，而另一个则可能完全聚集到一个机子上，同时还可能存在具有相同数目的数据。比如我们要找出现次数最多的前 100 个，我们将 1000 万的数据分布到 10 台机器上，找到每台出现次数最多的前 100 个，归并之后这样不能保证找到真正的第 100 个，因为比如出现次数最多的第 100 个可能有 1 万个，但是它被分到了 10 台机子，这样在每台上只有 1 千个，假设这些机子排名在 1000 个之前的那些都是单独分布在一台机子上的，比如有 1001 个，这样本来具有 1 万个的这个就会被淘汰，即使我们让每台机子选出出现次数最多的 1000 个再归并，仍然会出错，因为可能存在大量个数为 1001 个的发生聚集。因此不能将数据随便均分到不同机子上，而是要根据 hash 后的值将它们映射到不同的机子上处理，让不同的机器处理一个数值范围。

而外排序的方法会消耗大量的 IO，效率不会很高。而上面的分布式方法，也可以用于单机版本，也就是将总的数据根据值的范围，划分成多个不同的子文件，然后逐个处理。处理完毕之后再对这些单词的及其出现频率进行一个归并。实际上就可以利用一个外排序的归并过程。

另外还可以考虑近似计算，也就是我们可以通过结合自然语言属性，只将那些真正实际中出现最多的那些词作为一个字典，使得这个规模可以放入内存。

参考文献：

http://blog.csdn.net/jiaomeng/archive/2007/03/08/1523940.aspx d-Left Hashing

http://blog.csdn.net/jiaomeng/archive/2007/01/27/1495500.aspx
http://en.wikipedia.org/wiki/Bloom_filter
http://hi.baidu.com/xdzhang_china/blog/item/2847777e83fb0202293 应用 Bloom Filter 的几个小技巧
http://zh.wikipedia.org/wiki/%E5%80%92%E6%8E%92%E7%B4%A2%E5%BC%

9. **How to design a latest recently used cache?**
这是做网络搜索项目公司的一个热门经典题目,你怎么样设计一个数据结构来储存和调用你最近访问过的历史记录.

这道题目的意义在于，考察了候选人在面对比较复杂的问题是，能不能灵活应用多种数据结构的组合来达到最好的设计目的.

下面的代码是一个简单地 Python 以及 c++ 版本的代码，读者能够看出这个设计的 insertion, deletion, and search 方法的时间复杂度吗? O(?).

```
class LRU_Cache(object):

    def __init__(self, original_function, maxsize=1000):
        self.original_function = original_function
        self.maxsize = maxsize
        self.mapping = {}

        PREV, NEXT, KEY, VALUE = 0, 1, 2, 3
        self.head = [None, None, None, None]         # oldest
        self.tail = [self.head, None, None, None]    # newest
        self.head[NEXT] = self.tail

    def __call__(self, *key):
        PREV, NEXT, KEY, VALUE = 0, 1, 2, 3
        mapping, head, tail = self.mapping, self.head, self.tail
        sentinel = object()

        link = mapping.get(key, sentinel)
        if link is sentinel:
            value = self.original_function(*key)
            if len(mapping) >= self.maxsize:
                oldest = head[NEXT]
                next_oldest = oldest[NEXT]
                head[NEXT] = next_oldest
                next_oldest[PREV] = head
                del mapping[oldest[KEY]]
            last = tail[PREV]
            link = [last, tail, key, value]
            mapping[key] = last[NEXT] = tail[PREV] = link
        else:
            link_prev, link_next, key, value = link
            link_prev[NEXT] = link_next
            link_next[PREV] = link_prev
            last = tail[PREV]
```

```
            last[NEXT] = tail[PREV] = link
            link[PREV] = last
            link[NEXT] = tail
        return value

if __name__ == '__main__':
    p = LRU_Cache(ord, maxsize=3)
    for c in 'abcdecaeaa':
        print(c, p(c))
```

c++版本

```
#include <list>
#include <unordered_map>
#include <assert.h>

using namespace std;

template <class KEY_T, class VAL_T> class LRUCache{
private:
        list< pair<KEY_T,VAL_T> > item_list;
        unordered_map<KEY_T, decltype(item_list.begin()) > item_map;
        size_t cache_size;
private:
        void clean(void){
                while(item_map.size()>cache_size){
                        auto last_it = item_list.end(); last_it --;
                        item_map.erase(last_it->first);
                        item_list.pop_back();
                }
        };
public:
        LRUCache(int cache_size_):cache_size(cache_size_){
                ;
        };

        void put(const KEY_T &key, const VAL_T &val){
                auto it = item_map.find(key);
                if(it != item_map.end()){
                        item_list.erase(it->second);
                        item_map.erase(it);
                }
                item_list.push_front(make_pair(key,val));
                item_map.insert(make_pair(key, item_list.begin()));
                clean();
        };
        bool exist(const KEY_T &key){
                return (item_map.count(key)>0);
        };
        VAL_T get(const KEY_T &key){
                assert(exist(key));
                auto it = item_map.find(key);
                item_list.splice(item_list.begin(), item_list, it->second);
                return it->second->second;
        };
```

```
};
```

10. Given preorder and inorder traversal of a tree, construct the binary tree.

下面先给出对于某个节点的前序、中序、后序遍历的定义：

前序遍历：

先访问该节点，然后在递归遍历其左子树，然后再访问其右子树

中序遍历：

先递归遍历其左子树，然后访问该节点，然后再递归遍历其右子树

后序遍历：

先递归遍历其左子树，然后递归遍历其右子树，然后在访问该节点。

提示：
解决 这个问题的一个好方法是从后往前思考。这个问题的解决方法是先绘制二叉树，然后列出它的前序，中序遍历。由于大多数二进制树的问题，你要递归的解决这个问题。

关于重复：
在这个解决方案中，我们将假设在二叉树中，不允许重复。为什么呢？

考虑下面的情况：

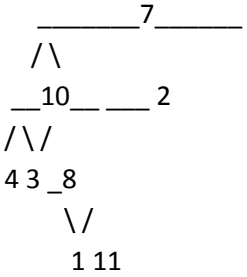
前序= [7,7]
中序= [7，7]

我们可以构造以下的二叉树树 ，这两者都是非常有效的解决方案。

```
 7
/or\
7
```

显然，就不会有歧义，在构造树时候如果被允许重复。

解决方案：
让我们来看看这个例子树。



上面的二叉树的前序和中序遍历是：

前序= [7，10，4，3，1，2，8，11]
中序= [4，10 3，1，7，11，8，2]

观察这个问题的关键是树的根目录中序遍历的第一个元素一致。这必须是真实的，因为在序遍历遍历根节点前儿童。根节点的值似乎是 7 从上述二进制树。

我们不难发现，作为第 4 个指数中序序列中出现。（请注意，我们假设在树中，不允许重复，所以就没有歧义）。前序遍历，访问左子树的第一，然后根节点的右子树。因此，7 剩下的所有元素都必须是在左子树和正确的所有元素必须是在右子树。

从上述的观察，我们看到了一个明确的递归模式。在创建根节点（7），我们构建其左，右子树的中序遍历[4，10，3，1]和[11，8，2]。我们还需要相应的遍历中可以找到类似的方式。如果你还记得，前序遍历根节点的顺序依次左子树和右子树。因此，左和右子树的后序遍历必须是[10，4，3，1]和[2，8，11]分别。自左而右子树二叉树自己的权利，我们可以递归解决！

我们离开了我们如何寻找根值的索引中序序列中的一些细节。一个简单的线性搜索如何呢？如果我们假设构造的二进制树总是平衡的，那么我们就可以保证运行的时间复杂度为 O（N 日志 N），其中 N 为节点的数目。然而，这不是一定的情况下，所构造的二进制树可以具有最坏的复杂度为 O（N 2）的左/右，偏斜。

一个更有效的方法是通过采用高效的查找机制，例如哈希表，以消除搜索。通过哈希及其相应的索引中序序列中的一个元素的值，在固定的时间，我们可以做的查找。现在，我们只需要 O（N）的时间来建构的树，这在理论上是最有效的方式。

仅供说明用途，下面的代码使用一个简单的数组为查表，这是限制的只有 0 到 255 的元素。你应该能够很容易地扩展它使用一个哈希表。

```
const int MAX = 256;
```

```
// a fast lookup for inorder's element -> index
// binary tree's element must be in the range of [0, MAX-1]
int mapIndex[MAX];
void mapToIndices(int inorder[], int n) {
   for (int i = 0; i < n; i++) {
      assert(0 <= inorder[i] && inorder[i] <= MAX-1);
      mapIndex[inorder[i]] = i;
   }
}

Node *buildInorderPreorder(int in[], int pre[], int n, int offset) {
   assert(n >= 0);
   if (n == 0) return NULL;
   int rootVal = pre[0];
   int i = mapIndex[rootVal]-offset;  // the divider's index
   Node *root = new Node(rootVal);
   root->left = buildInorderPreorder(in, pre+1, i, offset);
   root->right = buildInorderPreorder(in+i+1, pre+i+1, n-i-1, offset+i+1);
   return root;
}
```

现在，如果我们的中序和后序遍历，我们可以构建二叉树的吗？

答案是肯定的，使用类似的方法如上。

进一步的思考：

如果我们给定的序和后序遍历，我们可以构建二叉树的吗？为什么或者为什么不呢？
由于前序，中序和后序遍历，你怎么能确认这些遍历是指相同的二进制树吗？

11. 怎么判断单向链表中没有环

读者您能想出多少种方法那?
您能从时间和空间复杂度上分析你的方法的优劣吗?
以下代码是最经典的一个方式，您能写出你的方法吗?

Solution:
The best solution runs in O(N) time and uses O(1) space. It uses two pointers (one slow pointer and one fast pointer). The slow pointer advances one node at a time, while the fast pointer traverses twice as fast. If the list has loop in it, eventually the fast and slow pointer will meet at

the same node. On the other hand, if the loop has no loop, the fast pointer will reach the end of list before the slow pointer does.

```
bool hasLoop(Node *head) {
  Node *slow = head, *fast = head;
  while (slow && fast && fast->next) {
    slow = slow->next;
    fast = fast->next->next;
    if (slow == fast)
      return true;
  }
  return false;
}
```

12. 一些基本的网络问题知识汇总

1. What are 10Base2, 10Base5 and 10BaseT Ethernet LANs

   10Base2—An Ethernet term meaning a maximum transfer rate of 10 Megabits per second that uses baseband
   signaling, with a contiguous cable segment length of 100 meters and a maximum of 2 segments.
   10Base5—An Ethernet term meaning a maximum transfer rate of 10 Megabits per second that uses baseband
   signaling, with 5 continuous segments not exceeding 100 meters per segment.
   10BaseT—An Ethernet term meaning a maximum transfer rate of 10 Megabits per second that uses baseband
   signaling and twisted pair cabling.

   2. What is the difference between an unspecified passive open and a fully specified passive open
   An unspecified passive open has the server waiting for a connection request from a client. A fully specified passive open has the server waiting for a connection from a specific client.

   3. Explain the function of Transmission Control Block
   A TCB is a complex data structure that contains a considerable amount of information about each connection.

   4. What is a Management Information Base (MIB)
   A Management Information Base is part of every SNMP-managed device. Each SNMP agent has the MIB database that contains information about the device's status, its performance, connections, and configuration. The MIB is queried by SNMP.

   5. What is anonymous FTP and why would you use it
   Anonymous FTP enables users to connect to a host without using a valid login and password. Usually, anonymous FTP uses a login called anonymous or guest, with the password usually

requesting the user's ID for tracking purposes only. Anonymous FTP is used to enable a large number of users to access files on the host without having to go to the trouble of setting up logins for them all. Anonymous FTP systems usually have strict controls over the areas an anonymous user can access.

6. What is a pseudo tty
A pseudo tty or false terminal enables external machines to connect through Telnet or rlogin. Without a pseudo tty, no connection can take place.

7. Which layer of the 7 layer model provides services to the Application layer over the Session layer connection?
Presentation.

8. What does the Mount protocol do ?
The Mount protocol returns a file handle and the name of the file system in which a requested file resides. The message is sent to the client from the server after reception of a client's request.

9. What is External Data Representation
External Data Representation is a method of encoding data within an RPC message, used to ensure that the data is not system-dependent.

10. Which OSI Reference Layer controls application to application communication?
Session

11. BOOTP helps a diskless workstation boot. How does it get a message to the network looking for its IP address and the location of its operating system boot files ?
BOOTP sends a UDP message with a subnetwork broadcast address and waits for a reply from a server that gives it the IP address. The same message might contain the name of the machine that has the boot files on it. If the boot image location is not specified, the workstation sends another UDP message to query the server.

12. What is a DNS resource record
A resource record is an entry in a name server's database. There are several types of resource records used, including name-to-address resolution information. Resource records are maintained as ASCII files.

13. What protocol is used by DNS name servers
DNS uses UDP for communication between servers. It is a better choice than TCP because of the improved speed a connectionless protocol offers. Of course, transmission reliability suffers with UDP.

14. What is the difference between interior and exterior neighbor gateways
Interior gateways connect LANs of one organization, whereas exterior gateways connect the organization to the outside world.

15. What is the HELLO protocol used for
The HELLO protocol uses time instead of distance to determine optimal routing. It is an alternative to the Routing Information Protocol.

16. What are the advantages and disadvantages of the three types of routing tables
The three types of routing tables are fixed, dynamic, and fixed central. The fixed table must be manually modified every
time there is a change. A dynamic table changes its information based on network traffic, reducing the amount of manual maintenance. A fixed central table lets a manager
modify only one table, which is then read by other devices. The fixed central table reduces the need to update each machine's table, as with the fixed table. Usually a
dynamic table causes the fewest problems for a network administrator, although the table's contents can change without the administrator being aware of the change
.
17. What is a characteristic of Store and Forward switches?
They read the entire frame and check CRC before forwarding.

18. What is source route
It is a sequence of IP addresses identifying the route a datagram must follow. A source route may optionally be included in an IP datagram header.

19. What is RIP (Routing Information Protocol)
It is a simple protocol used to exchange information between the routers.

20. What is SLIP (Serial Line Interface Protocol)
It is a very simple protocol used for transmission of IP datagrams across a serial line.

21. What is Proxy ARP
It is using a router to answer ARP requests. This will be done when the originating host believes that a destination is local, when in fact is lies beyond router.

22. What is OSPF
It is an Internet routing protocol that scales well, can route traffic along multiple paths, and uses knowledge of an Internet's topology to make accurate routing decisions.

23. What is Kerberos

It is an authentication service developed at the Massachusetts Institute of Technology. Kerberos uses encryption to prevent intruders from discovering passwords and gaining unauthorized access to files.

24. What is a Multi-homed Host
It is a host that has a multiple network interfaces and that requires multiple IP addresses is called as a Multi-homed Host.

25. What is NVT (Network Virtual Terminal)
It is a set of rules defining a very simple virtual terminal interaction. The NVT is used in the start of a Telnet session.

26. What is Gateway-to-Gateway protocol
It is a protocol formerly used to exchange routing information between Internet core routers.

27. What is BGP (Border Gateway Protocol)
It is a protocol used to advertise the set of networks that can be reached with in an autonomous system. BGP enables this information to be shared with the autonomous system. This is newer than EGP (Exterior Gateway Protocol).

28. What is autonomous system
It is a collection of routers under the control of a single administrative authority and that uses a common Interior Gateway Protocol.

29. What is EGP (Exterior Gateway Protocol)
It is the protocol the routers in neighboring autonomous systems use to identify the set of networks that can be reached within or via each autonomous system.

30. What is IGP (Interior Gateway Protocol)
It is any routing protocol used within an autonomous system.

31. What is Mail Gateway
It is a system that performs a protocol translation between different electronic mail delivery protocols.

32. What is wide-mouth frog
Wide-mouth frog is the simplest known key distribution center (KDC) authentication protocol.

33. What are Digrams and Trigrams
The most common two letter combinations are called as digrams. e.g. th, in, er, re and an.

The most common three letter combinations are called as trigrams. e.g. the, ing, and, and ion.

34. What is silly window syndrome
It is a problem that can ruin TCP performance. This problem occurs when data are passed to the sending TCP entity in large blocks, but an interactive application on the receiving side reads 1 byte at a time.

35. What is region
When hierarchical routing is used, the routers are divided into what we call regions, with each router knowing all the details about how to route packets to destinations within its own region, but knowing nothing about the internal structure of other regions.

36. What is multicast routing
Sending a message to a group is called multicasting, and its routing algorithm is called multicast routing.

37. What is traffic shaping
One of the main causes of congestion is that traffic is often busy. If hosts could be made to transmit at a uniform rate,congestion would be less common. Another open loop method to help manage congestion is forcing the packet to be transmitted at a more predictable rate. This is called traffic shaping.

38. What is packet filter
Packet filter is a standard router equipped with some extra functionality. The extra functionality allows every incoming or outgoing packet to be inspected. Packets meeting some criterion are forwarded normally. Those that fail the test are dropped.

39. What is virtual path
Along any transmission path from a given source to a given destination, a group of virtual circuits can be grouped together into what is called path.

40. What is virtual channel
Virtual channel is normally a connection from one source to one destination, although multicast connections are also permitted. The other name for virtual channel is virtual circuit.

41. What is logical link control
One of two sublayers of the data link layer of OSI reference model, as defined by the IEEE 802 standard. This sublayer is responsible for maintaining the link between computers when they are sending data across the physical network connection.

42. Why should you care about the OSI Reference Model
It provides a framework for discussing network operations and design.

43. What is the difference between routable and non- routable protocols
Routable protocols can work with a router and can be used to build large networks. Non-Routable protocols are designed to work on small, local networks and cannot be used with a router.

44. What is MAU
In token Ring , hub is called Multistation Access Unit(MAU).

45. Explain 5-4-3 rule
In a Ethernet network, between any two points on the network, there can be no more than five network segments or four repeaters, and of those five segments only three of segments can be populated.

46. What is the difference between TFTP and FTP application layer protocols
The Trivial File Transfer Protocol (TFTP) allows a local host to obtain files from a remote host but does not provide reliability or security. It uses the fundamental packet delivery services offered by UDP. The File Transfer Protocol (FTP) is the standard mechanism provided by TCP / IP for copying a file from one host to another. It uses the services offered by TCP and so is reliable and secure. It establishes two connections (virtual circuits) between the hosts, one for data transfer and another for control information.

47. What is the range of addresses in the classes of internet addresses
Class A 0.0.0.0 - 127.255.255.255
Class B 128.0.0.0 - 191.255.255.255
Class C 192.0.0.0 - 223.255.255.255
Class D 224.0.0.0 - 239.255.255.255
Class E 240.0.0.0 - 247.255.255.255

48. What is the minimum and maximum length of the header in the TCP segment and IP datagram
The header should have a minimum length of 20 bytes and can have a maximum length of 60 bytes.

49. What is difference between ARP and RARP
The address resolution protocol (ARP) is used to associate the 32 bit IP address with the 48 bit physical address, used by a host or a router to find the physical address of another host on its network by sending a ARP query packet that includes the IP address of the receiver. The reverse address resolution protocol (RARP) allows a host to discover its

Internet address when it knows only its physical address.

50. What is ICMP
ICMP is Internet Control Message Protocol, a network layer protocol of the TCP/IP suite used by hosts and gateways to send notification of datagram problems back to the sender. It uses the echo test / reply to test whether a destination is reachable and responding. It also handles both control and error messages.

51. What are the data units at different layers of the TCP / IP protocol suite
The data unit created at the application layer is called a message, at the transport layer the data unit created is called either a segment or an user datagram, at the network layer the data unit created is called the datagram, at the data link layer the datagram is encapsulated in to a frame and finally transmitted as signals along the transmission media.

52. What is Project 802
It is a project started by IEEE to set standards that enable intercommunication between equipment from a variety of
manufacturers. It is a way for specifying functions of the physical layer, the data link layer and to some extent the network layer to allow for interconnectivity of major LAN protocols.
It consists of the following:
802.1 is an internetworking standard for compatibility of different LANs and MANs across protocols.
802.2 Logical link control (LLC) is the upper sublayer of the data link layer which is non-architecture-specific, that is
remains the same for all IEEE-defined LANs.
Media access control (MAC) is the lower sublayer of the data link layer that contains some distinct modules each
carrying proprietary information specific to the LAN product
being used. The modules are Ethernet LAN (802.3), Token ring LAN (802.4), Token bus LAN (802.5).
802.6 is distributed queue dual bus (DQDB) designed to be used in MANs.

53. What is Bandwidth
Every line has an upper limit and a lower limit on the frequency of signals it can carry. This limited range is called the
bandwidth.

54. Difference between bit rate and baud rate.
Bit rate is the number of bits transmitted during one second whereas baud rate refers to the number of signal units per

second that are required to represent those bits.
baud rate = bit rate / N where N is no-of-bits represented by each signal shift.

55. What is MAC address
The address for a device as it is identified at the Media Access Control (MAC) layer in the network architecture. MAC
address is usually stored in ROM on the network adapter card and is unique.

56. What is attenuation
The degeneration of a signal over distance on a network cable is called attenuation.

57. What is cladding
A layer of a glass surrounding the center fiber of glass inside a fiber-optic cable.

58. What is RAID
A method for providing fault tolerance by using multiple hard disk drives.

59. What is NETBIOS and NETBEUI
NETBIOS is a programming interface that allows I/O requests to be sent to and received from a remote computer and it
hides the networking hardware from applications.
NETBEUI is NetBIOS extended user interface. A transport protocol designed by microsoft and IBM for the use on small
subnets.

60. What is redirector
Redirector is software that intercepts file or prints I/O requests and translates them into network requests. This comes under presentation layer.

61. What is Beaconing
The process that allows a network to self-repair networks problems. The stations on the network notify the other stations on the ring when they are not receiving the transmissions. Beaconing is used in Token ring and FDDI networks.

62. What is terminal emulation, in which layer it comes
Telnet is also called as terminal emulation. It belongs to application layer.

63. What is frame relay, in which layer it comes
Frame relay is a packet switching technology. It will operate in the data link layer.

64. What do you meant by "triple X" in Networks

The function of PAD (Packet Assembler Disassembler) is described in a document known as X.3. The standard protocol has been defined between the terminal and the PAD, called X.28; another standard protocol exists between hte PAD and the network, called X.29. Together, these three recommendations are often called "triple X"

65. What is SAP
Series of interface points that allow other computers to communicate with the other layers of network protocol stack.

66. What is subnet
A generic term for section of a large networks usually separated by a bridge or router.

67. What is Brouter
Hybrid devices that combine the features of both bridges and routers.

68. How Gateway is different from Routers
A gateway operates at the upper levels of the OSI model and translates information between two completely different network architectures or data formats.

69. What are the different type of networking / internetworking devices
Repeater:
Also called a regenerator, it is an electronic device that operates only at physical layer. It receives the signal in the
network before it becomes weak, regenerates the original bit pattern and puts the refreshed copy back in to the link.

Bridges:
These operate both in the physical and data link layers of LANs of same type. They divide a larger network in to smaller
segments. They contain logic that allow them to keep the traffic for each segment separate and thus are repeaters that relay a frame only the side of the segment containing the intended recipent and control congestion.

Routers:
They relay packets among multiple interconnected networks (i.e. LANs of different type). They operate in the physical,
data link and network layers. They contain software that enable them to determine which of the several possible paths is the best for a particular transmission.

Gateways:
They relay packets among networks that have different protocols (e.g. between a LAN and a

WAN). They accept a
packet formatted for one protocol and convert it to a packet formatted for another protocol before forwarding it. They operate in all seven layers of the OSI model.

70. What is mesh network
A network in which there are multiple network links between computers to provide multiple paths for data to travel.

71. What is passive topology
When the computers on the network simply listen and receive the signal, they are referred to as passive because they don't amplify the signal in any way. Example for passive topology - linear bus.

72. What are the important topologies for networks

BUS topology:
In this each computer is directly connected to primary network cable in a single line.
Advantages:
Inexpensive, easy to install, simple to understand, easy to extend.

STAR topology:
In this all computers are connected using a central hub.
Advantages:
Can be inexpensive, easy to install and reconfigure and easy to trouble shoot physical problems.

RING topology:
In this all computers are connected in loop.
Advantages:
All computers have equal access to network media, installation can be simple, and signal does not degrade as much as
in other topologies because each computer regenerates it.

73. What are major types of networks and explain
Server-based network
Peer-to-peer network
Peer-to-peer network, computers can act as both servers sharing resources and as clients using the resources.
Server-based networks provide centralized control of network resources and rely on server computers to provide
security and network administration

74. What is Protocol Data Unit
The data unit in the LLC level is called the protocol data unit (PDU). The PDU contains of four fields a destination
service access point (DSAP), a source service access point (SSAP), a control field and an information field. DSAP, SSAP are addresses used by the LLC to identify the protocol stacks on the receiving and sending machines that are generating and using the data. The control field specifies whether the PDU frame is a information frame (I -
frame) or a supervisory frame (S - frame) or a unnumbered frame (U - frame).

75. What is difference between baseband and broadband transmission
In a baseband transmission, the entire bandwidth of the cable is consumed by a single signal. In broadband transmission, signals are sent on multiple frequencies, allowing multiple signals to be sent simultaneously.

76. What are the possible ways of data exchange
(i) Simplex (ii) Half-duplex (iii) Full-duplex.

77. What are the types of Transmission media
Signals are usually transmitted over some transmission media that are broadly classified in to two categories.
Guided Media:
These are those that provide a conduit from one device to another that include twisted-pair, coaxial cable and fiber-optic
cable. A signal traveling along any of these media is directed and is contained by the physical limits of the medium. Twisted-pair and coaxial cable use metallic that accept and transport signals in the form of electrical current. Optical fiber is a glass or plastic cable that accepts and transports signals in the form of light.
Unguided Media:
This is the wireless media that transport electromagnetic waves without using a physical conductor. Signals are
broadcast either through air. This is done through radio
communication, satellite communication and cellular telephony.

78. Difference between the communication and transmission.
Transmission is a physical movement of information and concern issues like bit polarity, synchronization, clock etc.
Communication means the meaning full exchange of information between two communication media.

79.The Internet Control Message Protocol occurs at what layer of the seven layer model?
Network

80.Which protocol resolves an IP address to a MAC address?
ARP

81.MIDI and MPEG are examples of what layer of the OSI seven layer model?
Presentation

82.What is the protocol number for UDP?
17

83.Which protocol is used for booting diskless workstations?
RARP

84.Which layer is responsible for putting 1s and 0s into a logical group?
Physical

85.What does 'P' mean when running a Trace?
Protocol unreachable

86.UDP works at which layer of the DOD model?
Host to Host

87.What is the default encapsulation of Netware 3.12?
802.2

88.Ping uses which Internet layer protocol?
ICMP

89.Which switching technology can reduce the size of a broadcast domain?
VLAN

90.What is the first step in data encapsulation?
User information is converted into data.

91.What is the protocol number for TCP?
6

92.What do you use the Aux port for?

Modem

93.Repeaters work at which layer of the OSI model?
Physical

94.WAN stands for which of the following?
Wide Area Network

95.What ISDN protocol specifies concepts, terminology, and services?
I

96.LAN stands for which of the following?
Local Are Network

97.DHCP stands for
Dynamic Host Configuration Protocol

98.What does the acronym ARP stand for?
Address Resolution Protocol

99.Which layer is responsible for identifying and establishing the availability of the intended communication partner?
Application.

100.Which OSI layer provides mechanical, electrical, procedural for activating, maintaining physical link?
Physical

13. 一些基本 sql 语言和数据库的基本知识

What is RDBMS?
Relational Data Base Management Systems (RDBMS) are database management systems that maintain data records and indices in tables. Relationships may be created and maintained across and among the data and tables. In a relational database, relationships between data items are expressed by means of tables. Interdependencies among these tables are expressed by data values rather than by pointers. This allows a high degree of data independence. An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage.

What is normalization?
Database normalization is a data design and organization process applied to data structures based on rules that help build relational databases. In relational database design, the process of organizing data to minimize redundancy. Normalization usually involves dividing a database into two or more tables and defining relationships between the tables.

The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

What are different normalization forms?

1NF: Eliminate Repeating Groups
Make a separate table for each set of related attributes, and give each table a primary key. Each field contains at most one value from its attribute domain.
2NF: Eliminate Redundant Data
If an attribute depends on only part of a multi-valued key, remove it to a separate table.
3NF: Eliminate Columns Not Dependent On Key
If attributes do not contribute to a description of the key, remove them to a separate table. All attributes must be directly dependent on the primary key
BCNF: Boyce-Codd Normal Form
If there are non-trivial dependencies between candidate key attributes, separate them out into distinct tables.
4NF: Isolate Independent Multiple Relationships
No table may contain two or more 1:n or n:m relationships that are not directly related.
5NF: Isolate Semantically Related Multiple Relationships
There may be practical constrains on information that justify separating logically related many-to-many relationships.
ONF: Optimal Normal Form
A model limited to only simple (elemental) facts, as expressed in Object Role Model notation.
DKNF: Domain-Key Normal Form
A model free from all modification anomalies.

Remember, these normalization guidelines are cumulative. For a database to be in 3NF, it must first fulfill all the criteria of a 2NF and 1NF database.

What is Stored Procedure?
A stored procedure is a named group of SQL statements that have been previously created and stored in the server database. Stored procedures accept input parameters so that a single procedure can be used over the network by several clients using different input data. And when the procedure is modified, all clients automatically get the new version. Stored procedures reduce network traffic and improve performance. Stored procedures can be used to help ensure the integrity of the database.
e.g. sp_helpdb, sp_renamedb, sp_depends etc.

What is Trigger?
A trigger is a SQL procedure that initiates an action when an event (INSERT, DELETE or UPDATE) occurs. Triggers are stored in and managed by the DBMS.Triggers are used to maintain the referential integrity of data by changing the data in a systematic fashion. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers can be viewed as similar to

stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.
Nested Trigger: A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.

What is View?
A simple view can be thought of as a subset of a table. It can be used for retrieving data, as well as updating or deleting rows. Rows updated or deleted in the view are updated or deleted in the table the view was created with. It should also be noted that as data in the original table changes, so does data in the view, as views are the way to look at part of the original table. The results of using a view are not permanently stored in the database. The data accessed through a view is actually constructed using standard T-SQL select command and can come from one to many different base tables or even other views.

What is Index?
An index is a physical structure containing pointers to the data. Indices are created in an existing table to locate rows more quickly and efficiently. It is possible to create an index on one or more columns of a table, and each index is given a name. The users cannot see the indexes, they are just used to speed up queries. Effective indexes are one of the best ways to improve performance in a database application. A table scan happens when there is no index available to help a query. In a table scan SQL Server examines every row in the table to satisfy the query results. Table scans are sometimes unavoidable, but on large tables, scans have a terrific impact on performance.

Clustered indexes define the physical sorting of a database table's rows in the storage media. For this reason, each database table may have only one clustered index. *Non-clustered indexes* are created outside of the database table and contain a sorted list of references to the table itself.

What are cursors? Explain different types of cursors. What are the disadvantages of cursors? How can you avoid cursors?

Cursors allow row-by-row prcessing of the resultsets.

Types of cursors: Static, Dynamic, Forward-only, Keyset-driven. See books online for more information.

Disadvantages of cursors: Each time you fetch a row from the cursor, it results in a network roundtrip, where as a normal SELECT query makes only one rowundtrip, however large the resultset is. Cursors are also costly because they require more resources and temporary storage (results in more IO operations). Furthere, there are restrictions on the SELECT statements that can be used with some types of cursors.

Most of the times, set based operations can be used instead of cursors. Here is an example:

If you have to give a flat hike to your employees using the following criteria:

Salary between 30000 and 40000 -- 5000 hike
Salary between 40000 and 55000 -- 7000 hike
Salary between 55000 and 65000 -- 9000 hike

In this situation many developers tend to use a cursor, determine each employee's salary and update his salary according to the above formula. But the same can be achieved by multiple update statements or can be combined in a single UPDATE statement as shown below:

```
UPDATE tbl_emp SET salary =
CASE WHEN salary BETWEEN 30000 AND 40000 THEN salary + 5000
WHEN salary BETWEEN 40000 AND 55000 THEN salary + 7000
WHEN salary BETWEEN 55000 AND 65000 THEN salary + 10000
END
```

Another situation in which developers tend to use cursors: You need to call a stored procedure when a column in a particular row meets certain condition. You don't have to use cursors for this. This can be achieved using WHILE loop, as long as there is a unique key to identify each row. For examples of using WHILE loop for row by row processing, check out the 'My code library' section of my site or search for WHILE.

Write down the general syntax for a SELECT statements covering all the options.

Here's the basic syntax: (Also checkout SELECT in books online for advanced syntax).

```
SELECT select_list
[INTO new_table_]
FROM table_source
[WHERE search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression [ASC | DESC] ]
```

What is a join and explain different types of joins.

Joins are used in queries to explain how different tables are related. Joins also let you select data from a table depending upon data from another table.

Types of joins: INNER JOINs, OUTER JOINs, CROSS JOINs. OUTER JOINs are further classified as LEFT OUTER JOINS, RIGHT OUTER JOINS and FULL OUTER JOINS.

For more information see pages from books online titled: "Join Fundamentals" and "Using Joins".

Can you have a nested transaction?

Yes, very much. Check out BEGIN TRAN, COMMIT, ROLLBACK, SAVE TRAN and @@TRANCOUNT

What is an extended stored procedure? Can you instantiate a COM object by using T-SQL?

An extended stored procedure is a function within a DLL (written in a programming language like C, C++ using Open Data Services (ODS) API) that can be called from T-SQL, just the way we call normal stored procedures using the EXEC statement. See books online to learn how to create extended stored procedures and how to add them to SQL Server.

Yes, you can instantiate a COM (written in languages like VB, VC++) object from T-SQL by using sp_OACreate stored procedure. Also see books online for sp_OAMethod, sp_OAGetProperty, sp_OASetProperty, sp_OADestroy. For an example of creating a COM object in VB and calling it from T-SQL, see 'My code library' section of this site.

What is the system function to get the current user's user id?

USER_ID(). Also check out other system functions like USER_NAME(), SYSTEM_USER, SESSION_USER, CURRENT_USER, USER, SUSER_SID(), HOST_NAME().

What are triggers? How many triggers you can have on a table? How to invoke a trigger on demand?

Triggers are special kind of stored procedures that get executed automatically when an INSERT, UPDATE or DELETE operation takes place on a table.

In SQL Server 6.5 you could define only 3 triggers per table, one for INSERT, one for UPDATE and one for DELETE. From SQL Server 7.0 onwards, this restriction is gone, and you could create multiple triggers per each action. But in 7.0 there's no way to control the order in which the triggers fire. In SQL Server 2000 you could specify which trigger fires first or fires last using sp_settriggerorder

Triggers can't be invoked on demand. They get triggered only when an associated action (INSERT, UPDATE, DELETE) happens on the table on which they are defined.

Triggers are generally used to implement business rules, auditing. Triggers can also be used to extend the referential integrity checks, but wherever possible, use constraints for this purpose, instead of triggers, as constraints are much faster.

Till SQL Server 7.0, triggers fire only after the data modification operation happens. So in a way, they are

called post triggers. But in SQL Server 2000 you could create pre triggers also. Search SQL Server 2000 books online for INSTEAD OF triggers.

Also check out books online for 'inserted table', 'deleted table' and COLUMNS_UPDATED()

There is a trigger defined for INSERT operations on a table, in an OLTP system. The trigger is written to instantiate a COM object and pass the newly insterted rows to it for some custom processing. What do you think of this implementation? Can this be implemented better?

Instantiating COM objects is a time consuming process and since you are doing it from within a trigger, it slows down the data insertion process. Same is the case with sending emails from triggers. This scenario can be better implemented by logging all the necessary data into a separate table, and have a job which periodically checks this table and does the needful.

What is a self join? Explain it with an example.

Self join is just like any other join, except that two instances of the same table will be joined in the query. Here is an example: Employees table which contains rows for normal employees as well as managers. So, to find out the managers of all the employees, you need a self join.

```
CREATE TABLE emp
(
empid int,
mgrid int,
empname char(10)
)

INSERT emp SELECT 1,2,'Vyas'
INSERT emp SELECT 2,3,'Mohan'
INSERT emp SELECT 3,NULL,'Shobha'
INSERT emp SELECT 4,2,'Shridhar'
INSERT emp SELECT 5,2,'Sourabh'

SELECT t1.empname [Employee], t2.empname [Manager]
FROM emp t1, emp t2
WHERE t1.mgrid = t2.empid
```

Here's an advanced query using a LEFT OUTER JOIN that even returns the employees without managers (super bosses)

```
SELECT t1.empname [Employee], COALESCE(t2.empname, 'No manager') [Manager]
FROM emp t1
LEFT OUTER JOIN
emp t2
```

ON
t1.mgrid = t2.empid

Given an employee table, how would you find out the second highest salary?

This question is quite a popular question . “How to find Nth Highest Salary of Employee”.

For particular example of employee :

How to get 1st, 2nd, 3rd, 4th, nth topmost salary from an Employee table

The following solution is for getting 2nd highest salary from Employee table ,
SELECT TOP 1 salary
FROM (
SELECT DISTINCT TOP 2 salary
FROM employee
ORDER BY salary DESC) a
ORDER BY salary
You can change and use it for getting nth highest salary from Employee table as follows
SELECT TOP 1 salary
FROM (
SELECT DISTINCT TOP n salary
FROM employee
ORDER BY salary DESC) a
ORDER BY salary
where n > 1 (n is always greater than one)

14.Dynamic Programming 动态规划的一些经典题目

编程中最有趣的，最迷人也是常常让候选人最迷惑的问题类型就是动态规划的问题了.

在数学，计算机科学，经济学，动态规划的方法解决复杂的问题分解成简单的子。它是适用于重叠的子表现出的属性的问题[1]和最优子结构（下面描述）。适用时，需要少得多的时间比天真的方法。

动态规划背后的想法是很简单的。在一般情况下，解决给定的问题，我们需要解决问题的不同部分（子），那么子结合的解决方案，以达成全面的解决方案。通常情况下，这些子实际上是一样的。动态规划方法旨在解决每个子只有一次，从而减少了计算的次数：到一个给定的子问题的解决方案已被计算一次，它被存储或“备忘录化”：下一次需要相同的解决方案，它简单看了一下。这种方法是特别有用的，当子问题的重复的数目呈指数增长作为一个功能的输入的大小。

动态规划算法可用于优化（例如，寻找两点之间的最短路径，或多个矩阵相乘以最快的方式）。动态规划算法将检查所有可能的方法来解决这个问题，将挑选最好的解决方案。因此，我们大致可以认为，动态规划，作为一个智能的，蛮力的方法，使我们能够通过所有可能的解决方案要挑最好的一个。如果该范围的问题是这样的，通过所有可能的解决方案是可能的和足够快的，动态

规划寻找最优解的担保。的替代品很多，如使用贪心算法，挑选最好的选择“在路上”在任何可能的分支。虽然贪婪算法并不能保证最佳的解决方案，它的速度更快。幸运的是，一些贪婪算法（最小生成树）已被证明可以导致最佳的解决方案。

例如，让我们说，你必须从点 A 到 B 点，尽可能快，在一个给定的城市，上下班高峰期。的动态规划算法研究整个交通报告，寻找到的道路，你可能会采取所有可能的组合，才告诉你哪条路是最快的。当然，你可能会等待一段时间，直到算法结束，然后才可以开始驾驶。你将采取的路径将是最快的（假设没有发生任何变化的外部环境）。另一方面，贪婪算法将开始你的驾驶立即将选择的道路，在每一个路口，看起来最快的。正如你可以想像，这种策略可能不会导致以最快的速度到达的时间，因为你可能会采取一些“简单”的街道，然后发现自己无可救药地陷入了交通堵塞。

动态编程的另一种替代方法是 memoization 的，即记录的中间结果。在一个基本的递归的方式，许多问题都可以迎刃而解。递归的问题是，它不记得中间结果，所以他们必须计算一遍又一遍。一个简单的解决方案来存储中间结果。这种方法被称为 memoization 的（而不是“记忆”）。动态规划算法更复杂比基本 memoization 的算法，从而可以更快。

动态规划在查找有很多**重叠子问题**的情况的最优解时有效。它将问题重新组合成子问题。为了避免多次解决这些子问题，它们的结果都逐渐被计算并被保存，从简单的问题直到整个问题都被解决。因此，动态规划保存递归时的结果，因而不会在解决同样的问题时花费时间。

动态规划只能应用于有**最优子结构**的问题。最优子结构的意思是局部最优解能决定全局最优解(对有些问题这个要求并不能完全满足，故有时需要引入一定的近似)。简单地说，问题能够分解成子问题来解决。

步骤

1. 最优子结构性质。如果问题的最优解所包含的子问题的解也是最优的，我们就称该问题具有最优子结构性质（即满足最优化原理）。最优子结构性质为动态规划算法解决问题提供了重要线索。
2. 子问题重叠性质。子问题重叠性质是指在用递归算法自顶向下对问题进行求解时，每次产生的子问题并不总是新问题，有些子问题会被重复计算多次。动态 规划算法正是利用了这种子问题的重叠性质，对每一个子问题只计算一次，然后将其计算结果保存在一个表格中，当再次需要计算已经计算过的子问题时，只是在表 格中简单地查看一下结果，从而获得较高的效率。

实例

斐波那契数列(Fibonacci polynomial)

计算斐波那契数列(Fibonacci polynomial)的一个最基础的算法是，直接按照定义计算：

```
function fib(n)
  if n = 0 or n = 1
    return 1
```

```
    return fib(n − 1) + fib(n − 2)
```

当 n=5 时，fib(5)的计算过程如下:

1. fib(5)
2. fib(4) + fib(3)
3. (fib(3) + fib(2)) + (fib(2) + fib(1))
4. ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
5. (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

由上面可以看出，这种算法对于相似的子问题进行了重复的计算，因此不是一种高效的算法。实际上，该算法的运算时间是指数级增长的。 改进的方法是，我们可以通过保存已经算出的子问题的解来避免重复计算：

```
array map [0...n] = { 0 => 0, 1 => 1 }
fib( n )
  if ( map m does not contain key n)
     m[n] := fib(n − 1) + fib(n − 2)
  return m[n]
```

将前 n 个已经算出的前 n 个数保存在数组 map 中，这样在后面的计算中可以直接易用前面的结果，从而避免了重复计算。算法的运算时间变为 *O(n)*

背包问题

背包问题作为 NP 完全问题，暂时不存在多项式时间算法。动态规划属于背包问题求解最优解的可行方法之一。此外，求解背包问题最优解还有搜索法等，近似解还有贪心法等， 分数背包问题有最优贪心解等。 背包问题具有最优子结构和重叠子问题。动态规划一般用于求解背包问题中的整数背包问题（即每种物品所选的个数必须是整数）。 解整数背包问题： 设有 n 件物品，每件价值记为 Pi，每件体积记为 Vi，用一个最大容积为 Vmax 的背包，求装入物品的最大价值。 用一个数组 f[i,j]表示取 i 件商品填充一个容积为 j 的背包的最大价值，显然问题的解就是 f[n,Vmax].

f[i,j]=

```
    f[i-1,j] {j<Vi}
    max{f[i-1,j],f[i,j-Vi]+Pi} {j>=Vi}
    0 {i=0 OR j=0}
```

对于特例 01 背包问题（即每件物品最多放 1 件，否则不放入）的问题，状态转移方程：

f[i,j]=

```
    f[i-1,j] {j<Vi}
    max{f[i-1,j],f[i-1,j-Vi]+Pi} {j>=Vi}
    0 {i=0 OR j=0}
```

下面是一个 动态规划问题列表，你应该能够实现对这些问题的算法。
在 DAGS 的最短路径

DAG 是一个有向无环图。可以线性的 DAG 节点，如可以安排，让所有边缘线从左至右（拓扑排序）。寻找最短路径向无环图，有一个经典的动态规划解决方案。
最长递增子序列（LIS）

给定一个序列的数字 A1，A2，...，找到最长的递增子序列。序列是这些数字的任意子集，为了 a_i1，a_i2，...，a_ik 其中-1 <I1 <I2 <IK。
最长公共子序列（LCS）

给定两个序列 X =（X1，X2，...，XMI）和 Y =（Y1，Y2，...，yni），我们希望找到的最大长度 X 和 Y 的公共子序列

给定一个序列 X =（X1，X2，...，xmi 的）中，Z =（Z1，Z2，...，ZK）是一个子序列的 X，如果存在一个严格递增序列 I1，I2，...，IK 的指数，其中 X，使得，对于所有的 j = 1，2，...，K，我们有：xij 表示= ZJ。

给定两个序列 X 和 Y，我们说，序列 Z 是 X 和 Y 中的一个共同的子序列，如果 Z 是 X 和 Y 的一个子序列。
编辑距离

由于 N 个字的字典，和另一个字 W，找到在字典中的字是最接近 W 的措施的亲密度，如果给定的由施加在输入字匹配字典中的词的数量的操作。可能的操作：插入，删除和替换一个字母。
背包

在一次抢劫中，一个窃贼发现更多的战利品比他的预期，决定采取什么。他的包（或“背包”）将举行总重量的最 W 磅。有 n 个项目的挑选，重量 W1，。。。Wn 和美元值 V1。。。VN。什么是最有价值的项目组合，他可以放入他的包吗？

这个问题有两个版本：1）每个项目或 2）每个项目有一个无限量的。
链矩阵乘法

我们如何确定最佳的顺序，如果我们要计算 A1×A2×…×安，艾未未的矩阵尺寸 M0×M1，M1×M2。。。MN-1×亿元，分别是多少？
Floyd-Warshall 算法

所有对图中的最短路径。
旅行商问题

减少整体的销售员要访问每个城市一次，从他的家乡距离。鉴于两两城市之间的距离，什么是最好的顺序看望他们吗？
独立设置的树

最大的独立设置的树。
楼梯问题

一个人爬上了 n 个台阶。他在一个步骤中，，只能爬到一个或两个楼梯。有多少种方法的人爬的 n 楼梯吗？
岩石问题

一个孩子有 n 个石头。他可以在岩石中多层次，每一层都有比其基层岩石。孩子有哪些方法可以安排他的 N 岩吗？
最大子序列

鉴于一个数字序列 A =（A1，A2，...，），找到的最大数目的子序列。一个子序列是序列 S =（A_K，A_K +1，... A_K + P）元素的序列 A.
最大子矩阵

的最大子序列问题，找到的最大子矩阵的所有元素的总和。

14. Given a string S, find the longest palindromic substring in S.

这道问题是如此的经典，只所以面试官喜欢问这个问题, 秘密在于这个问题很容易暴露出候选人的问题所在.
提示：
首先，请确保您了解回文是什么意思。回文是一个字符串，它在两个方向上读取相同的。例如，“aba”是 palindome 的，那么“abc”则不是。

一个常见的错误：
有些人会想拿出一个快速的解决方案，很不幸，这是有缺陷的（但是可以很容易地更正）：

逆转字符串 S 成为 S'。寻找最长公共子串 S 和 S'之间，也必须成为 最长的回文子串。

这似乎工作，让我们来看看下面的一些例子。

例如，S =“CABA”，S'=“ABAC”。的最长公共子之间的 S 和 S'是“阿巴”，这是答案。

让的尝试另一个例子：S =的“abacdfgdcaba”，S'=“abacdgfdcaba”的。 S 和 S 之间的最长公共子“是”abacd 的“。显然，这是不是一个有效的回文。

我们可以看到的最长公共子方法失败时，存在反转的非回文子串中的其它部分 S.为了纠正这个副本，每次我们找到一个最长公共子候选人，我们检查的子字符串的索引反转子串的原指数相同。如果是的话，那么我们尝试更新迄今发现的最长的回文，如果没有，我们跳过这一点，并寻找下一个候选人。

这给了我们一个 O（N2）DP 解决方案，它使用 O（N2），空间（可提高使用 O（N）的空

间）。请点击这里阅读更多关于最长公共字符串。

暴力解决方案，O（N3）：
最明显的暴力解决方案是，选择所有可能的起始位置和结束位置的子串，并验证它是否是一个回文。一共有 C（N，2）这样子字符串（不包括琐碎的解决方案，其中一个字符本身就是一个回文）。

由于验证每个子 O（N）时间，运行时间复杂度为 O（N3）。

动态规划的解决方案，O（N2）的时间和 O（N2）空间：
为了提高蛮力解决方案从 DP 的方法，首先想到如何避免不必要的重复计算，验证回文。考虑“巴”的情况。如果我们已经知道，“BAB”是回文，很明显，“亚的斯亚贝巴”必须是一个回文，因为这两个左和右端的字母是相同的。

即正式下文：

定义 P [I，J]←真，当且仅当子串 ... Sj 为回文，否则为 false。

因此，

P〔i，j]←[P（第 i +1，j-1 的]和 Si = Sj 的）

的基本情况是：

```
P[ i, i ] ← true
P[ i, i+1 ] ← ( Si = Si+1 )
```

这将产生一个直接的 DP 解决方案，我们首先初始化一个和两个字母回文，和工作的方式的时候，发现所有三个字母回文，等...

这给了我们一个运行时间复杂度为 O（N2），并使用 O（N2）的表空间来存储。

```
string longestPalindromeDP(string s) {
int n = s.length();
int longestBegin = 0;
int maxLen = 1;
bool table[1000][1000] = {false};
for (int i = 0; i < n; i++) {
   table[i][i] = true;
}
```

```
        for (int i = 0; i < n-1; i++) {
            if (s[i] == s[i+1]) {
                table[i][i+1] = true;
                longestBegin = i;
                maxLen = 2;
            }
        }
        for (int len = 3; len <= n; len++) {
            for (int i = 0; i < n-len+1; i++) {
                int j = i+len-1;
                if (s[i] == s[j] && table[i+1][j-1]) {
                    table[i][j] = true;
                    longestBegin = i;
                    maxLen = len;
                }
            }
        }
        return s.substr(longestBegin, maxLen);
    }
```

你能以上的空间复杂度进一步提高，以及如何？

一个简单的方法，O（N2）的时间和 O（1）空间：
事实上，我们可以解决这个问题的 O（N2）的时间，没有任何多余的空间。

我们观察到一个回文镜绕其中心。因此，一个回文可以从它的中心扩展，并且有只有 2N-1 这样的中心。

你可能会问，为什么有 2N-1，但不是 N 中心吗？原因是一个回文的中心可以在两个字母之间。这样的回文有偶数个字母（如“阿爸”），它的中心是两个的 b 的之间。

扩大回文以来围绕其中心可能需要 O（N）的时间，总的复杂度是 O（N2）。

```
string expandAroundCenter(string s, int c1, int c2) {
    int l = c1, r = c2;
    int n = s.length();
    while (l >= 0 && r <= n-1 && s[l] == s[r]) {
        l--;
        r++;
    }
    return s.substr(l+1, r-l-1);
}

string longestPalindromeSimple(string s) {
```

```
    int n = s.length();
    if (n == 0) return "";
    string longest = s.substr(0, 1);  // a single char itself is a palindrome
    for (int i = 0; i < n-1; i++) {
        string p1 = expandAroundCenter(s, i, i);
        if (p1.length() > longest.length())
            longest = p1;

        string p2 = expandAroundCenter(s, i, i+1);
        if (p2.length() > longest.length())
            longest = p2;
    }
    return longest;
}
```

15. 算法复杂度的讨论（来源 百度百科）

时间复杂度

算法复杂度分为时间复杂度和空间复杂度。其作用： 时间复杂度是度量算法执行的时间长短；而空间复杂度是度量算法所需存储空间的大小。任何算法运行所需要的时间几乎总是取决于他所处理的数据量，在这里我们主要说时间复杂度。对于一个给定计算机的算法程序，我们能画出运行时间的函数图。一个算法中的语句执行次数称为语句频度或时间频度。记为 T(n)。

1. 一般情况下，算法的基本操作重复执行的次数是模块 n 的某一个函数 f（n），因此，算法的时间复杂度记做：T（n）=O（f（n））

分析：随着模块 n 的增大，算法执行的时间的增长率和 f（n）的增长率成正比，所以 f（n）越小，算法的时间复杂度越低，算法的效率越高。

2. 在计算时间复杂度的时候，先找出算法的基本操作，然后根据相应的各语句确定它的执行次数，再找出 T（n）的同数量级（它的同数量级有以 下：1<Log2n <n <nLog2n <n 的平方<n 的三次方<2 的 n 次方<n！），找出后，f（n）=该数量级，若 T(n)/f(n)求极限可得到一常数 c，则时间 复杂度 T（n）=O（f（n））,例：

[java]

for（i=1;i<=n;++i）

```
1.  {
2.  for(j=1;j<=n;++j)
3.  {
4.  c[ i ][ j ]=0; //该步骤属于基本操作 执行次数：n 的平方 次
5.  for(k=1;k<=n;++k)
6.  c[ i ][ j ]+=a[ i ][ k ]*b[ k ][ j ]; //该步骤属于基本操作 执行次数：n 的三次方 次
7.  }
8.  }
```

则有 T（n）= n 的平方+n 的三次方，根据上面括号里的同数量级，我们可以确定 n 的三次方为 T（n）的同数量级,则有 f（n）= n 的三次方，然后根据 T（n）/f（n）求极限可得到常数 c。则该算法的 时间复杂度：T（n）=O（n^3） 注：n^3 即是 n 的 3 次方。

3.在 pascal 中比较容易理解，容易计算的方法是：看看有几重 for 循环，只有一重则时间复杂度为 O（n）,二重则为 O（n^2），依此类推，如果有二分则为 O(logn)，二分例如快速幂、二分查找，如果一个 for 循环套一个二分，那么时间复杂度则为 O(nlogn)。

按数量级递增排列，常见的时间复杂度有：

常数阶 O(1),对数阶 O(log2n),线性阶 O(n),

线性对数阶 O(nlog2n),平方阶 O(n^2)，立方阶 O(n^3),...，

k 次方阶 O(n^k), 指数阶 O(2^n) 。随着问题规模 n 的不断增大，上述时间复杂度不断增大，算法的执行效率越低。

根据定义，可以归纳出基本的计算步骤

1. 计算出基本操作的执行次数 T(n)

基本操作即算法中的每条语句（以;号作为分割），语句的执行次数也叫做语句的频度。在做算法分析时，一般默认为考虑最坏的情况。

2. 计算出 T(n)的数量级

求 T(n)的数量级，只要将 T(n)进行如下一些操作：

忽略常量、低次幂和最高次幂的系数，令 f(n)=T(n)的数量级。

3. 用大 O 来表示时间复杂度

当 n 趋近于无穷大时，如果 lim(T(n)/f(n))的值为不等于 0 的常数，则称 f(n)是 T(n)的同数量级函数。记作 T(n)=O(f(n))。

一个示例：

[java]

```
1.  int num1, num2;
2.  for(int i=0; i<n; i++){
3.      num1 += 1;
4.     for(int j=1; j<=n; j*=2){
5.         num2 += num1;
6.     }
7.  }
```

分析：

1.

语句 int num1, num2;的频度为 1；

语句 i=0;的频度为 1；

语句 i<n; i++; num1+=1; j=1; 的频度为 n；
语句 j<=n; j*=2; num2+=num1;的频度为 n*log2n；
T(n) = 2 + 4n + 3n*log2n

2.
忽略掉 T(n)中的常量、低次幂和最高次幂的系数
f(n) = n*log2n

3.
lim(T(n)/f(n)) = (2+4n+3n*log2n) / (n*log2n)
= 2*(1/n)*(1/log2n) + 4*(1/log2n) + 3
当 n 趋向于无穷大，1/n 趋向于 0，1/log2n 趋向于 0
所以极限等于 3。T(n) = O(n*log2n)

简化的计算步骤
再来分析一下，可以看出，决定算法复杂度的是执行次数最多的语句，这里是 num2 += num1，一般也是最内循环的语句。并且，通常将求解极限是否为常量也省略掉？
于是，以上步骤可以简化为：
1. 找到执行次数最多的语句
2. 计算语句执行次数的数量级
3. 用大 O 来表示结果

继续以上述算法为例，进行分析：
1.
执行次数最多的语句为 num2 += num1

2.
T(n) = n*log2n
f(n) = n*log2n

3.
// lim(T(n)/f(n)) = 1
T(n) = O(n*log2n)

二、插入排序算法的时间复杂度

现在研究一下插入排序算法的执行时间，按照习惯，输入长度 LEN 以下用 n 表示。设循环中各条语句的执行时间分别是 c1、c2、c3、c4、c5 这样五个常数：

[java] view plaincopy

```
1.  void insertion_sort(void)        执行时间
2.  {
3.      int i, j, key;
4.      for (j = 1; j < LEN; j++) {
5.          key = a[j];        c1
6.          i = j - 1;         c2
```

```
7.          while (i >= 0 && a[i] > key) {
8.              a[i+1] = a[i];     c3
9.              i--;          c4
10.         }
11.         a[i+1] = key;         c5
12.     }
13. }
```

显然外层 `for` 循环的执行次数是 n-1 次，假设内层的 `while` 循环执行 m 次，则总的执行时间粗略估计是(n-1)*(c1+c2+c5+m*(c3+c4))。当然，`for` 和 `while` 后面()括号中的赋值和条件判断的执行也需要时间，而我没有设一个常数来表示，这不影响我们的粗略估计。

这里有一个问题，m 不是个常数，也不取决于输入长度 n，而是取决于具体的输入数据。在最好情况下，数组 `a` 的原始数据已经排好序了，`while` 循环一次也不执行，总的执行时间是(c1+c2+c5)*n-(c1+c2+c5)，可以表示成 an+b 的形式，是 n 的线性函数（Linear Function）。那么在最坏情况（Worst Case）下又如何呢？所谓最坏情况是指数组 `a` 的原始数据正好是从大到小排好序的，请读者想一想为什么这是最坏情况，然后把上式中的 m 替换掉算一下执行时间是多少。

数组 `a` 的原始数据属于最好和最坏情况的都比较少见，如果原始数据是随机的，可称为平均情况（Average Case）。如果原始数据是随机的，那么每次循环将已排序的子序列 a[1..j-1]与新插入的元素 `key` 相比较，子序列中平均都有一半的元素比 `key` 大而另一半比 `key` 小，请读者把上式中的 m 替换掉算一下执行时间是多少。最后的结论应该是：在最坏情况和平均情况下，总的执行时间都可以表示成 an^2+bn+c 的形式，是 n 的二次函数（Quadratic Function）。

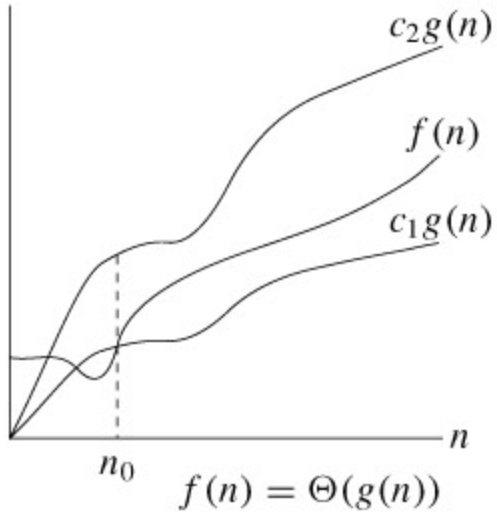
在分析算法的时间复杂度时，我们更关心最坏情况而不是最好情况，理由如下：

1. 最坏情况给出了算法执行时间的上界，我们可以确信，无论给什么输入，算法的执行时间都不会超过这个上界，这样为比较和分析提供了便利。
2. 对于某些算法，最坏情况是最常发生的情况，例如在数据库中查找某个信息的算法，最坏情况就是数据库中根本不存在该信息，都找遍了也没有，而某些应用场合经常要查找一个信息在数据库中存在不存在。
3. 虽然最坏情况是一种悲观估计，但是对于很多问题，平均情况和最坏情况的时间复杂度差不多，比如插入排序这个例子，平均情况和最坏情况的时间复杂度都是输入长度 n 的二次函数。

比较两个多项式 a_1n+b_1 和 $a_2n^2+b_2n+c_2$ 的值（n 取正整数）可以得出结论：n 的最高次指数是最主要的决定因素，常数项、低次幂项和系数都是次要的。比如 100n+1 和 n^2+1， 虽然后者的系数小，当 n 较小时前者的值较大，但是当 n>100 时，后者的值就远远大于前者了。如果同一个问题可以用两种算法解决，其中一种算法的时 间复杂度为线性函数，另一种算法的时间复杂度为二次函数，当问题的输入长度 n 足够大时，前者明显优于后者。因此我们可以用一种更粗略的方式表示算法的时间 复杂度，把系数和低次幂项都省去，线性函数记作 Θ(n)，二次函数记作 $\Theta(n^2)$。

Θ(g(n))表示和 g(n)同一量级的一类函数，例如所有的二次函数 f(n)都和 $g(n)=n^2$ 属于同一量级，都可以用 $\Theta(n^2)$来表示，甚至有些不是二次函数的也和 n^2 属于同一量级，例如 $2n^2+3\lg n$。“同一量级”这个概念可以用下图来说明（该图出自[算法导论]）：

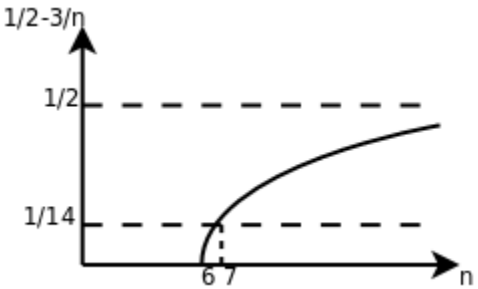
图 11.2. Θ-notation



如果可以找到两个正的常数 c_1 和 c_2，使得 n 足够大的时候（也就是 $n \geq n_0$ 的时候）f(n)总是夹在 $c_1g(n)$和 $c_2g(n)$之间，就说 f(n)和 g(n)是同一量级的，f(n)就可以用 Θ(g(n))来表示。

以二次函数为例，比如 $1/2n^2-3n$，要证明它是属于 $\Theta(n^2)$这个集合的，我们必须确定 c_1、c_2 和 n_0，这些常数不随 n 改变，并且当 $n \geq n_0$ 以后，$c_1n^2 \leq 1/2n^2-3n \leq c_2n^2$ 总是成立的。为此我们从不等式的每一边都除以 n^2，得到 $c_1 \leq 1/2-3/n \leq c_2$。见下图：

图 11.3. 1/2-3/n



这样就很容易看出来，无论 n 取多少，该函数一定小于 1/2，因此 $c_2=1/2$，当 n=6 时函数值为 0，n>6 时该函数都大于 0，可以取 $n_0=7$，$c_1=1/14$，这样当 $n \geq n_0$ 时都有 $1/2-3/n \geq c_1$。通过这个证明过程可以得出结论，当 n 足够大时任何 an^2+bn+c 都夹在 c_1n^2 和 c_2n^2 之间，相对于 n^2 项来说 bn+c 的影响可以忽略，a 可以通过选取合适的 c_1、c_2 来补偿。

几种常见的时间复杂度函数按数量级从小到大的顺序依次是：Θ(lgn)，Θ(sqrt(n))，Θ(n)，Θ(nlgn)，Θ(n^2)，Θ(n^3)，Θ(2^n)，Θ(n!)。其中，lgn 通常表示以 10 为底 n 的对数，但是对于 Θ-notation 来说，Θ(lgn)和 Θ($\log_2 n$) 并无区别（想一想这是为什么），在算法分析中 lgn 通常表示以 2 为底 n 的对数。可是什么算法的时间复杂度里会出现 lgn 呢？回顾插入排序的时间复杂度分 析，无非

是循环体的执行时间乘以循环次数，只有加和乘运算，怎么会出来 lg 呢？下一节归并排序的时间复杂度里面就有 lg，请读者留心 lg 运算是从哪出来 的。

除了 Θ-notation 之外，表示算法的时间复杂度常用的还有一种 Big-O notation。我们知道插入排序在最坏情况和平均情况下时间复杂度是 $\Theta(n^2)$，在最好情况下是 $\Theta(n)$，数量级比 $\Theta(n^2)$要小，那么总结起来在各种情况下插入排序的时间复杂度是 $O(n^2)$。Θ 的含义和“等于”类似，而大 O 的含义和“小于等于”类似。受内存管理机影响，指令的执行时间不一定是常数，但执行时间的上界（Upper Bound）肯定是常数，我们这里假设语句的执行时间是常数只是一个粗略估计。

三、常用的算法的时间复杂度和空间复杂度

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	$O(n^2)$	$O(n^2)$	稳定	O(1)
快速排序	$O(n^2)$	$O(n*\log_2 n)$	不稳定	$O(\log_2 n)$~O(n)
选择排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
二叉树排序	$O(n^2)$	$O(n*\log_2 n)$	不一顶	O(n)
插入排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
堆排序	$O(n*\log_2 n)$	$O(n*\log_2 n)$	不稳定	O(1)
希尔排序	O	O	不稳定	O(1)

(5)等待及自我分析

面试结束之时 ,就是 move on 和 move forward 的时候,在结束讨论之后，您应该准备一些很好的问题来询问面试官，来抓住最后机会进一步打动他.

What do you see ahead for your company in the next five years?

How do you see the future for this industry?

What do you consider to be your firm's most important assets?

What can you tell me about your new product or plans for growth?

How do you rate your competition?

The position's history
Asking about why the position is vacant can provide insight into the company and the potential for advancement. According to Annie Stevens and Greg Gostanian, managing partners at executive and career development firm ClearRock, good questions include:

What happened to the last person who held this job?

What were the major strengths and weaknesses of the last person who held this job?

What types of skills do you NOT already have onboard that you're looking to fill with a new hire?

The department
Asking about your department's workers and role in the company can help you understand more about the company's culture and hierarchy. Stanford suggests asking:

What is the overall structure of the company and how does your department fit the structure?

What are the career paths in this department?

What have been the department's successes in the last couple of years?

How do you view your group/division/department?

The job's responsibilities
To avoid any confusion later on, it pays to gain a solid understanding of the position. FGP International's Eddie Payne recommends inquiring:

What would you consider to be the most important aspects of this job?

What are the skills and attributes you value most for someone being hired for this position?

Where have successful employees previously in this position progressed to within the company?

Could you describe a typical day or week in this position? The typical client or customer I would be dealing with?

The expectations
To determine how and when you will evaluated, Payne recommend advises asking:

What are the most immediate challenges of the position that need to be addressed in the first three months?

What are the performance expectations of this position over the first 12 months?

在被面试官送出门后，你依然有机会进一步表现自己，这里有一份样板的感谢电子邮件，你可以使用一下.

Your address

Date

Interviewer
Title
Organization name
Organization address

Dear Interviewer (no first names!):

FIRST PARAGRAPH

Thank the person who invited you for the interview and also thank any other people who interviewed you.

Mention you enjoyed the visit and comment on some specific thing that interested or impressed you the most.

SECOND PARAGRAPH

Use this paragraph to reiterate your interest in the position and tell why you are qualified for this position.

Mention you want to be a part of their team/office/staff etc.

THIRD PARAGRAPH

If the interviewer mentioned a date when they will contact you, state what your next step will be (a phone call, etc.).

Sincerely,

(Your signature)

Your name typed

在你接收到对方的雇佣信后，你可以用一下网站来判断一下两地的生活水准差，如果您将搬到另外一个城市

http://www.bankrate.com/calculators/savings/moving-cost-of-living-calculator.aspx

下面这个网站，可以判断你的收入在当地同行业中的相对位置

www.glassdoor.com

下面有网上抄录的十条忠告，笔者觉得也很适合候选人的心态调整

刚刚走上社会的年轻人，充满了蓄势待发的豪情、青春的朝气、前卫的思想，梦想着丰富的待遇和轰轰烈烈的事业。可是，社会毕竟是一所包罗万象、喧嚣复杂的大学校，这里没有寒暑假，拒绝虚假和肤浅，更拒绝空想和庸碌，难以预告何时开课何时放学。

如何在涉世之初少走弯路，有一个好的开端，开始一番成功的事业？以下是一些先行者积累的 10 条有益的涉世忠告。好好地遵循、把握这些忠告和建议吧，比起所学的课堂课程来，它毫不逊色！

1. 买个闹钟，以便按时叫醒你。贪睡和不守时，都将成为你工作和事业上的

绊脚石，任何时候都一样。不仅要学会准时，更要学会提前。就如你坐车去某地，沿途的风景很美，你忍不住下车看一看，后来虽然你还是赶到了某地，却不是准时到达。“闹钟”只是一种简单的标志和提示，真正灵活、实用的时间，掌握在每个人的心中。

2. 如果你不喜欢现在的工作，要么辞职不干，要么就闭嘴不言。初出茅庐，往往眼高手低，心高气傲，大事做不了，小事不愿做。不要养成挑三拣四的习惯。不要雨天烦打伞，不带伞又怕淋雨，处处表现出不满的情绪。记住，不做则已，要做就要做好。

3. 每个人都有孤独的时候。要学会忍受孤独，这样才会成熟起来。年轻人嘻嘻哈哈、打打闹闹惯了，到了一个陌生的环境，面对形形色色的人和事，一下子不知所措起来，有时连一个可以倾心说话的地方也没有。这时，千万别浮躁，学会静心，学会忍受孤独。在孤独中思考，在思考中成熟，在成熟中升华。不要因为寂寞而乱了方寸，而去做无聊无益的事情，白白浪费了宝贵的时间。

4. 走运时要做好倒霉的准备。有一天，一只狐狸走到一个葡萄园外，看见里面水灵灵的葡萄垂涎欲滴。可是外面有栅栏挡着，无法进去。于是它一狠心绝食三日，减肥之后，终于钻进葡萄园内饱餐一顿。当它心满意足地想离开葡萄园时，发觉自己吃得太饱，怎么也钻不出栅栏了。相信任何人都不愿做这样的狐狸。退路同样重要。饱带干粮，晴带雨伞，点滴积累，水到渠成。有的东西今天似乎一文不值，但有朝一日也许就会身价百倍。

5. 不要像玻璃那样脆弱。有的人眼睛总盯着自己，所以长不高看不远；总是喜欢怨天尤人，也使别人无比厌烦。没有苦中苦，哪来甜中甜？不要像玻璃那样脆弱，而应像水晶一样透明，太阳一样辉煌，腊梅一样坚强。既然睁开眼睛享受风的清凉，就不要埋怨风中细小的沙粒。

6. 管住自己的嘴巴。不要谈论自己，更不要议论别人。谈论自己往往会自大虚伪，在名不副实中失去自己。议论别人往往陷入鸡毛蒜皮的是非口舌中纠缠不清。每天下班后和你的那些同事朋友喝酒聊天可不是件好事，因为，这中间往往会把议论同事、朋友当做话题。背后议论人总是不好的，尤其是议论别人的短处，这些会降低你的人格。

7. 机会从不会“失掉”，你失掉了，自有别人会得到。不要凡事在天，守株待兔，更不要寄希望于“机会”。机会只不过是相对于充分准备而又善于创造机会的人而言的。也许，你正为失去一个机会而懊悔、埋怨的时候，机会正被你对面那个同样的“倒霉鬼”给抓住了。没有机会，就要创造机会，有了机会，就要巧妙地抓住。

8. 若电话老是不响，你该打出去。很多时候，电话会给你带来意想不到的收获，它不是花瓶，仅仅成为一种摆设。交了新朋友，别忘了老朋友，朋友多了路好走。交际的一大诀窍就是主动。好的人缘好的口碑，往往助你的事业更上一个台阶。

9. 千万不要因为自己已经到了结婚年龄而草率结婚。想结婚，就要找一个能和你心心相印、相辅相携的伴侣。不要因为放纵和游戏而恋爱，不要因为恋爱而影响工作和事业，更不要因一桩草率而失败的婚姻而使人生受阻。感情用事往往会因小失大。

10. 写出你一生要做的事情，把单子放在皮夹里，经常拿出来看。人生要有目标，要有计划，要有提醒，要有紧迫感。一个又一个小目标串起来，就成了你一生的大目标。生活富足了，环境改善了，不要忘了皮夹里那张看似薄薄的单子。

当你加入美国公司之后，开始领取肥美的工资单的时候, 以下是十个小诀窍帮助你在公司中生存下去.

1.系统的研究。

如果你想适应， 你要观看你的同事是如何相互作用的，他们是如何工作的，以及他们如何发牢骚。学习和效仿这些细微之处是必不可少的，“领导风格，工作文化和惯用的字体类型，”

2。 盟友。

开始是你的上司和直接的队友。但是，不要忽视你的部门以外的同事和经理。丰富的

DeMatteo， 建议分别与同事和经理开会。要求在该公司自己的角色，以及如何最好的，你可以与他们一起工作。除了学习如何在公司经营中，您将学习如何赚钱 - 你需要的关键信息到 Excel 中的新角色。“新员工花时间在内部网络内的公司有更好的表现，晋升速度更快，可以期望有一个更强大的网络”

3。加入游戏。

如果你没有足够的工作，你的盘子上，提供一些关你的经理或队友的手。“你的老板看到你作为一个拼命三郎，和你的同事会更喜欢你。同样，当你的第一次会议，参加 马上加入战局。准备参加讨论（或至少是，加盟 的自我介绍，并表达你的兴奋）。要注意到，虽然，你 可能只是一个底层的工程师.

4。超出预期。

问问你的老板，她希望你来完成你的第一个 3 个月的时间。“什么是你的经理的目标和你的部门的高级管理人员的期望是什么？” “如何做你的工作适合部门的目标吗？你越了解你的角色是如何融入大局，更容易将强调正确的事情，你的表现。”你越早知道你的老板对你的期望，就可以更快地超越他们。

5。多问问题。

没有人会错，你问了很多问题。作为部门的新手，这是相当多的工作。“不要以为人们会认为你需要知道的一切， 我们大多数人的利益没有得到正式的培训计划。” “应该已经要求并不能带来灾难性的后果，不会很快被人遗忘的一个问题。”

6。承认错误。

自己粗心 - 立即。“不要等待，看看老板已经注意到，大多数老板都更细心的你可能会认为，” “不花时间试图找出如何摆脱困境，说相反，与你的老板找到一个可行的解决方案和实施它的时候了。这样做显示了诚信和能力来解决。

7。抓住细节。

喜欢看你有一切尽在掌握是成功的一半。因此，让你的最后期限。显示会议的时间和准备。当你说你要跟进，跟进。记住别人的名字，并学习正确发音。如果你有做笔记。拼写检查和校对你的 文字。按时清理您的工作区 。

8。努力的完美主义。

在把任何邮件发出前，仔细思考，是否能够更好的表达

9。和老板同事享用午餐。

每天午餐就是了一个机会，得到内幕消息的组织， 你的上级，建议餐饮，每月两次。 你会是第一个知道大的变化，未来的长矛和黄金机会，你可以跳出自我的小圈子。

10。做真实的自己。

很容易看那些已经成功地在你面前，并尝试模仿他们， 但请记住，你有你自己的一套独特的长处和短处，拥抱他们。如果你正努力成为你崇拜的的人，你将永远不会做正义的真正的优势。

虽然在调查中显示，电脑专业的学历和收入不成正比，但是你依然应该努力学习给自己创造更多地空间.

根据对六万五千名电脑工程师的调查，发现其中很多收入在五万到六万年薪的人根本没有大学学位。在这个行业中更看中的是技术程度和对新鲜事物的掌握能力。

当然学历也有一些作用，但差别不大。一般来讲，具有硕士学位者平均工资在六万四千元左右，学士学位在五万四千元，高中学历为四万九千元上下。

更大的问题是年资拉不开档次。调查发现，新参加工作者与已经工作十到二十年者工资差别很小。例如三十一岁到三十五岁者平均年薪为五万九千元，三十六岁到四十岁平均年薪为六万一千元，四十一岁到四十六岁者平均年薪为六万五千元。或许说明在这个行业中，因为技术发展太快，重要的是对新技术的应用，以往的经验并不为人看中

(6) 友情收录

1) 物理博士的一百次面试

物理博士的一百次面试
时间飞快，已经快到纽约一年了,在纽约找工作也快有 8 个月了
掐指算来，我考居然
已经有了一百个面试,phone screen + onsite
时光爆炸，让我也来回忆一把

做一个面试达人　　美国编程师面试宝典,年薪十万美金不是梦

1. 2007/2/13,在学校, Cxxx 公司的第一轮笔试,fail,不过好像学校里面不怎么有人过
2. 2007/3/4,在学校,Mxxxx,屁股再植公司,phone screen, fail,不能理解这个公司要我干什么
放弃就业市场的活动,写完论文,7 月底搬到纽约,8 月开始正式找工作
3.8/20 纽约,iXXX 电话面试,没有任何东西是贯通的，还被听见敲击键盘的呻吟,fail
4.9/5 纽约 Mxxx 模特公司电话面试,fail, 不准备自己出钱去 las vegas road show
5.9/12 纽约,one XXXX 模特公司 onsite,fail,皮肤不好
6.9/20 纽约 ,onsite, s XXX trading firm,成功，拿了一个要求做 trader 的 offer, 但是要求三个月无收入培训，谈判，错过电话,没有继续
7.9/24 纽约,错过摩根斯坦利的 HR 的电话，打过去，他们说明天给我电话,或者第三天, 再也没有打回来过,fail
8.9/26,纽约,onsite with Strategic Wealth XXXX,一个骗人公司,fail
9. 10/2 纽约,phone screen, GXXX website,fail,也不准备去
10.10/4 纽约深夜,phone screen,一个中国人开的公司，要招收 part tim,fail
11.10/10 纽约,onsite c++ paper test.Ixxxx company,失败，考试作弊被抓
12.10/15 纽约,onsite with further XXX ,结果发现原来是给一个欧洲人大的绿卡广告
13.10/16 纽约,phone screen ,PXXXX ,fail,没有说服 HM
14.10/17 纽约,onsite with FXXXX,靠，自己都说工作只适合高中生 fail
15.10/26,纽约 onsite with BXXXX,第一个大投行，fail，没有经济学的知识
16.10/31, 纽约,online test with cXXX ,pass.
17.11/1, 纽约,phone screen with MXXX, pass,大家都很开心,, 直接要 onsite
18,11/2 上午，phone with Q xxx 在芝加哥的 trading firm,韩国人，c++问的问题都不知道
19.11/2 下午,onsite with cXXX,fail,C++考试没有通过,韩国人，问的问题只知道一 half
20.11/8 深夜, phonescreen with DXXX bank,fail,三个很简单的 derivited 的问题，只对了一个
21.11/14, phone screen with B XXX hedge fund, fail,因为已经知道了题目，过于兴奋，被对方识破
22.11/15 . phone screen with 英国 GoldXXX ,fail,被对方击败,简单的 braintease, 其实几天前，我还做过
23.11/16 ,纽约,onsite with Mxxx,大家都很开心，以为搞定，后来告诉我没有钱了, kao!
24. 11/19 ,纽约, onsite with Wallstreet XXX,迟到半个小时，后来对方告诉我需要有经验的人.fail
25.11/29,和纽约 Goldxxx, phonescreen,中国兄弟识破我啥都不懂,fail
26.12/3,纽约，onsite with S xxx hedge fund,失败，口水太多，但是别人需要已经有工作经验的（这个鸟公司，我以后还会遇到!)
27.12/6. 纽约直接电话找到了某一个 Hedge fund 的 CTO,phone screen 后，他说没有适

合你的位置
28.12/11, 纽约,phone screen with Mxxx trading firm,part time,c++知识不够,
fail
29.12/14 ,纽约，onsite with XXX trading firm for a parttime c++ position,最
后关口失败了,因为说错了，总是偏离主题,fail
30.12/19,phone screen,with sholes 的公司，结果被中国 MD 的问题击败，C,c++,C#,
sql,math,finanice,braintease! fail,不过他们也不支持 H1B
31.12/28,phone screen with lehman 亚洲部，fail,被第一个中国兄弟击倒，第二个
印度人电话面试结束的时候都说我应该去哪里工作了，//sigh,中国人了我 1:20 分钟的
问题,是不是害怕招收一个懒人进去和自己一样不能干活？haha,我们还是需要更放松一
点,当然我 c++知识不过关!
32.1/2, 纽约。phone screen with Janet XXXX,45 分钟被我掌握主动，结果还是被击
败最后他指出了一道问题我没有做出来. fail!后来知道他们去年面世了，300 个人有一
个人也没有招收
33.1/3 纽约, phone screen with WXXX HF,失败，
34.1/4 phone screen with GXXX HF,失败
35.1/9, onsite with goldXXX,失败，被老妖怪赶出来了
36.1/7 onsite with cxxx,fail,迟到了半个小时!再说，他们需要的也不是我能做的
37.1/8,忘记那一个公司，来自佛罗里达,pphone screen, c++,database,失败
38.1/17 ,phone screen with sxxx again,中国姐姐在我的眼泪之下，让我 pass!
39.1/22 onsite with sxxx,见到了老板，依然失败，需要一个正在工作的由美国工作
经验的
40.1/24 phone screen with Fxxxx,pass,邀请 onsite
41.1/25 phone screen with Cred suisse,可惜啊，他们不是需要物理学的博士的,他
们需要一个 account manager,fail
42.1/25,onsite with a consult company,为 SAC 提供服务的,吹自己的 C #能力，被当
场击败,fail
43.1/28. onsite with fxxx,大家都很高兴，但是不知道为什么最后什么也没有发生估
计也是没有工作经验!
44.1/29 onsite with Xxxxx.我考大早上，跑过去，明显不需要我这样的人啊
45.1/30,onsite with Ixx,我靠，北印度人侮辱了，拿 java 的问题当成 c++问我，可惜
啊，我还没有抢到识破它的地步
46.1/XX ,phone screen with gold XXX again,之后，没有回应 after GoldXXX
Exploratory Interview
47.2/15, onsite with open XXXX,失败,c++ paper test 没有做好
48.2/6, phone screen with gxxx,fail,嫌弃我没有经济学知识
49.2/20, phone screen with brXXXXX,忘记了这个 phone screen,彻底没有印象了，到
底发生了什么？fail!
50.2/25 phonescreen with NY state 一家 php 网站公司，失败，要价太高!

51.2/26 phone screen with 另外一个网站公司，莫名其妙，博士资格太高,哈哈
52.2/28 online test with Fxxxx,失败,以后还会遇见他!!!
53.3/2 phone screen with AT& T ,一说 H1B,对方就挂电话了,fail
54.3/5, online testwith imes tech.pass
55.3/7 onsite with pxxx. HF,被普林斯顿的博士赶出来了
56.3/10, onsites with rimes,我靠，口水一大堆，明显什么都能做，还是被中国兄弟 fail 掉
57.3/11.phone screen with blueink,fail,超出他们的接受范围
58.3/9 phone screen with 一个纽约青年，他想自己做一个成人网站，需要网站帮手，我和她交谈了一下,fail,因为再也没有联系我,gaga,我其实只是想看看他的姑娘的质量
59.3/12,online test with Fxxx,就是前面说过的哪个公司,我换了名字,pass
60.3/12 早上,onsite with Knowledge Delivery Systems, Inc.妈的，被黑人助手击败，不知道说了我什么坏话,靠,fail
61.3/13 深夜,living meeting with micro xxx,失败了，因为 live meting 在物理博士的老电脑上不能用....,当然了也和水平没有过关有关系
62.3/14. onsite with Fxxx,没有通过，因为跑过去发现，我靠，还有一个 paper test ,失败了.不过我胡汉三还是会回来的
63.3/16, phone screen with micro XXX 的印度猎头商人,pass
64.3/17,深夜，take c++ online test with Bxxx capiytal 纽约，失败,他们需要很高的 c++的水平
65.3/18,深夜，phone screen,pass,被某一个 online college 聘请为科学院的院长，每一个月几百块钱
66.3/31,onsite with bilantic.com,一群流氓，妈的明显没有钱，支付工资，我要求 65K,他们只能出 40K-60K,还让我去 onsite,我靠,后来写信骂他们了
67.4/3 ,phone screen with 意大利黑手党!通过
66.4/4,换了一个名字 again,take the online test with fXXXX,通过
68.4/7 ,onsite with a new HF,没有回音，应该 fail 了， Caltech 的毕业生小小年纪，妈的没有说服他
69.4/5, phone screen with cyberlead,通过，主程序员喜欢我电话上交流的方式
70.4/8,又去了 fxxx,onsite, paper test 全部正确，而且自己也化了妆,进来一位中国兄弟，我考饲养怪气，被我 fail 掉了
71.忘记说,4/5 晚上,phone screen with edit.com,fail,不愿意做 pattime 来开始
72.忘记说,3/22, onsite wih 一个网站公司 for php developer,被 php 考题击败
73,忘记说 3/23,take doubleclick 的 c++,.net online test,失败
74.4/10 onsite with 意大利黑手党，我靠，这么热情，到现在还没有理我,fail，太多 bear 的 lay off 的人了
75.4/11 早上 onsite with cyberlead,老板说我不合格但是 head programmer 很喜欢我交流的方式，，老板问那个人说"如果物理学博士不会怎么办?"程序员说“我可以教他",我靠，不是遇见 gay 了吧.到现在没有联系我，估计 fail 掉了

76.4/12 下午 onsite with xxx 公司就在 wall street 上面，看来我的计算机知识还不能让我的中国同胞满意,fail
77.忘记说了,三月 X 号,phone with sxxx HF again,fail,也是因为目前没有什么工作
78.4/14,phone screnn with a new york small company for data analyst,失败，因为很奇怪电话上说说好好再也没有联系我
79.4/17,phone screen with HSBC,到现在还没有联系我,fail
80.4/16 phone sceen with microxxx 的印度猎头公司，pass,但是这家不愿意出钱给我买飞机票
81.12/XX/2007,忘记说，曾经 onsite with a trading firm ,结果需要自己掏钱成为 trader, 拒绝.
82-86.4/17 深夜，我成为了导演，我要拍一部关于华尔街的性爱电影，广告发布后，几位美女来应征，我靠,居然说不能 free shoot test,还有的都 45 碎了还要来献身，体会到公司招人的辛苦.我不怪你们了,Hiring manager 们
87. 4/18, phone screen with ITG xxx,fail,没有说服 HM,我明显 over qualified,在电话上，我哭了没有用.
88.忘记说,3/28 living meeting with micro xxx 的印度猎头公司 pass,
89.4/18 phone scren with education xxx,NJ,电话上，我说我只要 50 K,HM 立刻邀请我 onsite,我没有去因为要飞去西雅图
90.4/21,西雅图，上午,micro XX.印度人，中国人,失败,coding 题目做得不好
91.4/21,西雅图，下午,micro XX.印度人，印度人,没有见到中国 HM,失败,coding 题目做得不好
92.4/22,西雅图，上午,micro XX.中国人，小白,香港 HM(他很喜欢 twins,我问他，你恨成关系吗？他说我不恨，我觉得他做的很对)，失败,coding 题目做得不好
93.4/22,西雅图，下午,micro XX.中国人，俄国人,德国人失败,coding 题目做得不好？？我觉得做的不错啊，做的挺快，错误也少
94.4/23,西雅图，上午,micro XX.中国女人，小白,失败,coding 题目做得不好（？？），可能和我不敢与中国姐姐眼神交流有关，不是我的错，姐姐实在太….,也许我太敏感，也许她太敏感
95.4/23,西雅图，下午,micro XX.印度女人，小白，印度男人,成功,coding 题目做得很好，幸好，接受我的 offer 来的很早，不然我就开车死在西雅图的雪山上了，真他妈的郁闷
96.4/25, phone screen with M xxx,不错，但是需要接下来若干考试.kao,不去
97,4/25 phone screen with another HF, 也做出要接下来考试的样子，靠不去
98.4/25 晚上知道,马上要和 morgan xxx 上海 phone screen,娱乐一下吧
99,4/24 phonescreen with Parkcentral Capital & Perot Investments, 不会去这种鸟地方
100,都和接近 200 多个猎头或者电话或者 face to face 见面过了，知道在市场不好的情况下，大家的辛苦……我觉得我们还是生活在一个爱的社会里面!!

友情收录 2) **物理学博士捡破烂的心得（李大明友情奉献个人作品）**

捡垃圾日记（1）之序

兑完了易拉罐回来，看见门上夹着一张纸条。不用说，肯定是房东在催帐了。这个月的房租还欠着，房东已经几次**要叫**来抄家了，只是还没有付诸行动。看来说话不算数并不总是坏的。

找工作基本上没希望了。八个月了，面试了五家，结果惊人的一致。我读物理博士时的实验结果从来没有这么一致过。当初搬家到纽约，孤注一掷找所谓的矿工， 希望把 30 年的光阴来个大翻盘。当然，这样的雄心壮志现在已经完全让位给了捡易拉罐换钱付房租了。这还是最近才发现的一条生财之路。不过竞争对手也多，都 是说广东话的老头老太。

捡易拉罐也是技术活儿，不掌握诀窍也做不好。要在傍晚时分出动，太早了垃圾袋还没有扔出来，太晚了被老头老 太抢了先。对市场上的稀有事件也要关注，也就是随机过程里面的泊松分布，对 fairprice 的影响还是挺大的。seaport 举行露天音乐会就是这样的 稀有事件，往往伴随着易拉罐的大量出货。我现在要清点一下今天兑易拉罐的钱了.....

捡垃圾日记（2）之得而复失的 100 美金

和东边不一样，运河大道西边的易拉罐似乎是完全不可预测的，也就是马汀狗（martingale）的标准状态。

看来今天运气不好。我没捡到易拉罐，却注意到街边围了很多人，我拨开诡异兴奋的人群挤了进去。原来是有人设局赌博。三个矿泉水瓶盖，一个红豆子。猜豆子 在哪个瓶盖下面。赌注是惊人的 100 美金。赌客却不像是有钱人。设局的是一个肥胖的黑女人，貌似精明，对我大喊,"justpointit"。我不想参与 这样的赌局。

但我还是很快就看出了这个赌局并不是 1/3 的胜率。虽然黑女人手法很快，但是还是可以很容易跟踪到红豆子。而且红豆子在瓶盖下面竟然能透过一抹红色。

我指了一个瓶盖，揭开真有红豆子。我竟然赢了。黑女人没有丝毫犹豫的抽出百元大钞给我。我急忙捏住大钞，但是抽不出来。黑女人嚷道，"showmeyour100bucks"。我没钱，我连赢钱的资格都没有！

捡垃圾日记(3)之 derivatives

垃圾会有很多衍生物(derivatives)，天生就有的，不是华尔街的肥胖的 CEO 们空想出来的骗钱

的玩意。我就是垃圾衍生物，我捡垃圾，我从物理博士到一文不名。

傍晚匆匆赶到中国城的 Pellstreet。那里有个垃圾堆经常有易拉罐。鹿鸣春外面挤满了人等着进去，女侍者拿着牌在外面傲慢地叫着号。我的目光却如同极的磁铁一

样躲过鹿鸣春的玻璃窗。广东老头也知道这个易拉罐地点，所以我要快点过去。

路过德昌门口，看见一个长相很福建的干瘪老头蹲在他的自行车下面倒油桶。自行车上面夸张的挂满了餐馆用过的 32 磅的白色食用油桶，很多。这种油桶是不能换钱的，我知道。

老头慢慢悠悠的把油桶往另一个桶里倒。我注意到其实每个空桶底还剩一口黄黄的油。要倒出来却颇为不易。油是有粘性的，这是由于油分子侧链之间的引力引起 的。所以油会附在桶壁上，能流出来的或许只是一小滴。这个工作需要耐心。显然老头符合这个工作要求。他已经收集了小半桶油了，小半桶食用油，他可能会用这 个炒菜。

食用油也是垃圾的衍生物.....

捡垃圾日记(4)之四维时空和虫洞

时空是四维的，你不需要是物理博士就能明白这一点。捡易拉罐也需要掐好时间和地点才能捡到。虫洞是连接两个四维时空的通道。当我通过虫洞时，原子会被从新排序，我就在虫洞的另一端获得了重生。但是我没法携带信息，所以我在虫洞的两端都失忆。

从 seaport 到孔子大厦，两个相距较远的易拉罐地点。但是扔垃圾的时间却很接近。我必须在 seaport 完工后迅速赶到孔子大厦。推着超市购物车，装满空易拉罐赶路，罐子们会发出非常吵杂的响声。

人在这种状态下会恍惚发呆。直到路过布鲁克林大桥的桥底，被桥上滴下来的冰冷水滴直穿后脖颈来个透心凉，才会从发呆的状态恍然醒悟过来。发呆的本质其实就是通过虫洞到了四维空间的另一端，返回后却完全失忆，只有时间的流逝证实了虫洞的真实存在。

捡垃圾日记(5)之无产阶级的援助

“东”就是贫穷。东方国家，东欧，东德。当然，东纽约是这里的贫民窟这毫不奇怪。不太冷的时候，我能够打开窗户看到后面的破败小街。有个破房子，窗户上钉着木板 子，墙上有油漆喷的大的立体感的字母。几个月前搬来一家墨西哥人。他们认识我。有时候看我走过来，会提出一大袋子空罐子给我，同时感谢我正在做一件很有意 义的工作。我其实没有工作。他们有。因为那个墨西哥男的前几天很兴奋的告诉我，他找到工作了。我问是什么工作，他说是 cut fish。要每天早上 4 点钟起床。我表示了祝贺。今天路过的时候，在他们门口捡了个空易拉罐，上面插着一张钱，是 20 美元。

捡垃圾日记(6)之秘笈不外传

捡易拉罐不纯粹是体力活。只要是人干的活，都是有诀窍的。

首先就是要发现新的易拉罐地点，淘汰枯竭的易拉罐地点。好喝啤酒的住户搬走以后，自然易拉罐的来源就枯竭了。这样的地点要排除。新住户搬来后，要观察是不是又是一个酒鬼。有的每天都有一两箱子罐子扔出来。那就是钱。

对纽约的大小公共聚会要留心。经常会扔很多很多空罐子。这样的聚会一般都是在几个固定地点，时间却不固定。还有就是罐子的牌子，不是每个罐子都能换到钱的。有的罐子纽约是不收的，

除非拿到缅因州去换钱。哦，对了，罐子回收机也各有不同，有的很挑剔罐子的牌子，有的很宽容。和人一样。

还有其他的诀窍我现在不能告诉你。

捡垃圾日记(7)之所罗门王的宝藏

所罗门王的宝藏是一部老电影。电影里，所罗门王囤积大量金银珠宝，宝藏所在地没人知道。我也不知道。但是我也有一个秘密地点，别人不知道。所罗门王也不 知道。我捡到大量的易拉罐来不及换钱，又没法拖回家，就会暂时存放在我的藏宝地点，上面搭上油毛毡。没有人会注意到下面是钱。第二天我会起个早床来把财宝 拖出来，还要避免被他人看见。这样会产生一些刺激兴奋感。我也许会画一张藏宝图，然后把它扯成两半，分给两个继承人。又或许会写一首诗，暗示藏宝地点，只 有聪明人才能看明白。但这有一定的危险性。福尔摩斯探案集里面有一篇“马斯格雷夫典礼”。管家和福尔

摩斯都是聪明人。管家其实更聪明，先一步看明白了那首诗而发现了宝藏的秘密，但却栽在女人手里。

其实我没有宝藏，我只有一些没人要的易拉罐。或许早就有人翻开过油毛毡，但是没人在意那些垃圾。

捡垃圾日记(8)之平地大道的农贸市场

东纽约的平地大道(Flatlands Ave)附近，所有的不像样的东西全在这里扎堆，像样的东西你一样也看不到。但有一点例外，这里有受人喜爱的廉价的农贸市场。说廉价其实并不准确。大家其实把它当成免费菜市场。傍晚 6 点钟光景，小贩会开始收摊。具体方式有两种：超低价甩卖，或者倒扣在地上踩烂。而最终结局都是在地上踩烂，因为超低价还是不 会有人买的，大家都等着捡

免费菜。

运气好的话能捡到整箱子的芹菜，我一般不需要这么多，就会招呼附近的看得顺眼的或黑或棕或灰的哥 们，一起分享。或者交换一下对方手中多余的菜。大家都兴高采烈，混乱肮脏的菜市场上弥漫着过节的气氛。当天色渐渐黑下来的时候，大铲车会过来把地上的垃圾 撮成一座小山，里面全是破烂的青椒茄子芹菜洋葱和板条箱之类。

大家都满载而归。我也是。

捡垃圾日记(9)之弗来明戈的纹身

有人在敲门，我犹豫着把门打开，幸好不是房东。是墨西哥邻居。我问他有什么事。他说要我帮个忙。我说行，是什么事。他说他的名字用中文怎么写。我询问了之后写在纸上给他。他很高兴，问我能不能用中文书法写给他。我觉得这个忙我能帮。

我大概花了 25 分钟，在一张旧报纸的空白处把他的中文名字描成楷书。然后小心的撕下来给他。我觉得他很感激我。几天之后，我路过墨西哥邻居家，他正在门口摆弄一把烂锯条。看见我，颇为自豪的展现左胳膊给我看。上面纹着四个楷书汉字“弗来明戈”。

捡垃圾日记(10)之带血的鱼头

弗来明戈凌晨 3 点来敲我的门的时候，我早已戴好帽子穿好大衣，还不忘在口袋里面塞上几个塑料袋子。弗来明戈带上水，我们就在夜幕中匆匆出发了。筋疲力竭 又冷又饿的时候，弗来明戈说到了。抬头发现是一处海滩，乱石嶙峋，长满青苔和贝壳。地上却有凌乱的脚印和拖拽的痕迹。岸边有一条废弃的木船，半船仓水。弗 来明戈踩着乱石跳过去说，今天果然有。我也看见岸边有很多血淋淋的鱼头，有的很大，有的还在动。很显然，渔船刚来过。弗来明戈拿出他的塑料袋，一个套在手 上当手套。另一个用来装鱼头。很快就捡了一大袋子鱼头，然后又加了几层袋子提着顺手。我也捡了更大的一袋子鱼头。

捡垃圾日记(11)之国际友人的问候

收获一大推车易拉罐的日子屈指可数，今天就是这样的一天。我已经在易拉罐回收机旁边工作了好一会儿了。我的后面还有一个黑人，他有一小袋子罐子。他在等我。回收机的设计是笨拙的。只有听到易拉罐被哗的一声压碎的时候，才能塞进下一个。

有时候塞进去还要转好几圈，又吐出来。换个方向塞进去，又可以了。如果几次都吐出来，我就把它扔进旁边的垃圾桶。表示这个不能换成钱。回收机是通过扫描 条码来认识罐子的。如果复印很多条码贴在所有的罐子上，那么应该都能换成钱。愚蠢的回收机!黑人似乎等的不耐烦了，开始左右晃动。课本里面讲过，多线程可 以减少等待时间。也就是说，我仍一个罐子，换黑人来扔一个，然后我再来。如此而已。我觉得是自欺欺人。我读过的课本大部分是胡说八道。我还有

半车子罐子。 黑人终于等不了了，大叫，“what the hell?!”

捡垃圾日记(12)之上帝的蓬蓬车

布鲁克林学院东边的东 42 街，那里有一座教堂。我去那儿不是因为我是上帝的羊。

我不进教堂，我只去教堂后面停车场。那里有一个黄色的铁皮箱子，一人多高。这是**捐东西的地方。也是我捡东西的地方。有几个篮球扔在地上。我挑了一个 手感好的。我还没想好去哪儿玩球。有一个烤面包机，我捡起来发现好像坏了，又放下。还有一双鞋子，看起来很脏。我是不屑于要的。还有几本书。有一本是 little woman。书脊的装订胶被掰成了两半。还有一箱子别人不要的玩具。都是一些式样非常古旧的汽车模型，和烫了卷发的男人画片。

大概是猫王时代的东西。都是一些旧东西。除了一辆蓬蓬车。那是一辆看起来比较新的蓬蓬车。小孩子用的。可以坐，也可以躺，前面有小餐台，下面有踏脚的。 上面还有帐篷可以撑起来挡太阳。我小的时候没有享用过这东西。那个时候的人们需要找关系到车间弄几根铁条，然后求人把它焊接起来。坐在上面大概像古代的囚 车。

爱因斯坦说上帝不掷骰子。上帝看来掷蓬蓬车。

捡垃圾日记(13)之 我的物理学垃圾

我捡垃圾。我也扔垃圾。今天晚上要把一些垃圾书清出来扔掉。量子力学讲义，两本，我能说出哪个公式在哪一页。还有两本很厚的习题书，手写的。有代表性的 题我都做过，考 qual 也有一题是这上面的。我把他们放进垃圾桶。近代数学讲义。一堆稀奇古怪的符号。学了不用，现在再翻开好像没学过。也进了垃圾桶。费 曼物理学讲义，罗哩罗嗦，不是我喜欢的书。然后看到了我的 paper，是一篇晶体衍射的文章，我的 30 年光阴就浓缩在这几页纸上。我用量子力学来研究这个 问题的。物理评论上有 copy，全世界到处都能找到这个 copy。所以我不需要保留这个。它也进了垃圾桶。我不停的扔，我的纸箱子很快见了底。我其实没有 什么可保留的。他们都是垃圾。

捡垃圾日记(14)之 30 岁的单车

弗来明戈的 儿子在家门口用锈铁丝在地上掏一个洞玩。我骑着单车在他面前转了几圈。我很满意。我花了一整天时间修好了这个车。捡的。我把锈死的链条拆下来在地上摔了很 久，直到水泥地上出现了一大片白印子。唯一的麻烦是那个密码锁。四位。我随机的试了几次，打不开。我把它调到零，一个一个的试。我需要试一万次。当我最终 试到四个九的时候，还是打不开。弗来明戈家门口有一箱破烂工具。我拿起一个拧歪了的螺丝刀，又放下。然后看到一把合不拢的钳子。我没有碰它。我不知道我在 找什么。我从四个五开始又试了一次。这次打开了。这不难，我知道它一定有一个答案。但是很多问题不一定存在答案。或者是混沌态，有无穷无尽的答案 也就是没有答案。

我又骑了一次，直到街头南边的那段废弃铁路。然后回来。我需要捡一个气筒，虽然我已经很满意了。弗来明戈的儿子已经走了。他可能回去吃饭去了。

捡垃圾日记(15)之哥伦布公园的精灵

我觉察到的时候，一个黑女人已经坐在石头凳子的另一侧了。她有一只竹篓，里面有木屑和彩蛋。她说她认识那个守门的，所以拿到了只有小孩子才能领到的篓子。

这个公园称作一小块空地更合适。一道铁丝网把公园分割成更小的两块。那边一块是草地，有个守门的只让小孩进去。大人在铁丝网外面看，或者闲聊。小孩在里 面跑，或者叫。黑女人接着说，彩蛋里面是巧克力，可以吃的。我说我知道。然后她又重复了一遍她认识那个守门的，所以拿到了只有小孩子才能领到的篓子。我看 了一下守门的人，点头表示同意她的说法。

我走的时候，小孩还在里面跑。我说的精灵是指那个守门的。如果你学过热力学，你知道我在说什么。

捡垃圾日记(16)之冰箱贴

我把一个冰箱贴剪坏了。我发现没有修补的可能了。这是我花了 40 美元邮购的。几个简单工具，一个冰箱贴样品，和一套原材料。按照样品做好后，邮寄回去。合格的话，会寄更多的原材料给我。他们会按照成品付钱。

但是，我把它剪坏了。

捡垃圾日记(17)之运河大道的奢侈品

运河大道东的奢侈品分两类：真的和假的。我无法鉴别真的，但是那些晃眼的金银首饰店显然不欢迎我。我只能挤在街边侃价的人群中，时不时摸起地摊上的假 表，用拇指擦亮表盘左看右看。我需要买一块手表看时间。和老板艰难的侃价到 5 块钱。然后我发现表带太长。老板拿出一套工具把表带去掉了几节。我求他把那几 节还给我。然后我给了钱准备走。我对今天的购物比较满意。我开始兴奋的摆弄这个表。我把分钟调到 0，发现时钟不能指到整点。转了几圈还是不行。我忙回去问 那个老板能不能修好。他接过去看了一下，摸出一叠钱，抽出 5 块给我，说，你走吧。

捡垃圾日记(18)之**

在 10 点还差 10 分钟的时候，我急忙拉上窗帘，好让我看起来不在家。

过了一会儿，有人敲门。然后他们开始敲我的窗户玻璃。我只好掀开窗帘。果然是他们，那两个穿着黑西装的年轻人，我前几天遇到的两个**的传道士。

我只好把他们让进来。他们问我那本摩门经读得怎么样了。我说我没时间。他们说其实不用花很多时间，每天只要抽出 30 分钟就可以了。我说我现在要谋生，没有 办法认真读那个书。然后他们开始胡扯创世纪以及耶稣到南美洲传道的神迹，以及强调这本经书的重要性。然后我给他们普及了一下天体物理的基本知识。然后他们 继续强调要先相信神，不要纠缠细节，神这样做肯定有他的道理的。我们的对话就像两条平行的铁轨，没有交点。

看来还是不学物理比较好。物理是可以明目的，但是看的太透了反而不好。就像亚当夏娃本来很快乐，吃了蛇果，反而看明白了很多事情，徒增烦恼而已。

捡垃圾日记(19)之我的同事

如果捡垃圾是一个工作的话，我也有几个同事。

一对老两口，每天很高兴的样子，说靠捡易拉罐，老家已经盖起了楼房，准备再干一阵子就回国了，再也不来了。反正身份早就黑了。

还有一个越南人，把我当假想敌。早上看见我，就随便指一个方向，说那边有，然后自己却往另一边跑。其实两边我都捡过了。

另一个老头，知识分子模样，看见我总是很不好意思的说，随便捡捡，是个早餐钱。我不知道我是不是应该告诉他我靠这个付房租。

捡垃圾日记(20)之尾声

这个系列发表以来，大量的读者朋友发来邮件表示关心。所以我觉得有必要交代一下故事的结局。

文章中的我，现在在华尔街工作也有些年头了，也早就搬家到了世外桃源般的罗斯福岛。这个系列之所以称作“日记”，而不是“故事”，因为它是一段真实的经历，而且还没有结束。我偶尔还会捡易拉罐，正像弗来明戈说的，这是一个很有意义的工作。

弗来明戈，现在升任华尔街一家快餐厅的烤鱼主厨。如果有读者朋友在华尔街地区，我们可以一起去吃烤鱼午餐。弗来明戈一定会亲自为我们烤鱼，顺便显摆一下他的中文名的纹身。

（7）各种技术快速入门收录（感谢各位计算机前辈总结）

有人认为，现在是 java 和.net 的时代，有谁还需要 C 以及汇编呢？孰不知，java 和.net 是建立在软件之上的，是为了垄断市场而建立起来的 体系，犹如挖好一个金壁辉煌的坑，请你往下跳，还自以为站在巨人的肩膀上，事实上成了坑底之蛙。要成为一个真正的程序员，并期望成为一个程序员高手，必须 从机器出发，从 cpu 到操作系统，再到软件体系，高手的境界就是悟道后的明镜灵台，软件设计出神入化，我就是程序，程序就是我。

旁观者李四说：此人大笨也！我用鼠标随便拖几个控件，就是一个 xxx 管理系统了，你用 C 语言怕是一年也写不出来吧！好吧，我要承认，讲这话的 都已经是 mS 的奴才了，别的我不了解，MFC 本身就是一个封闭的架构，从 MFC 入手学习，你只会形成一种封闭的思维模式，因为 MS 希望很多人只学会表面的 东西，不致成为高手，所以它大力推荐所谓的可视化的程序开发工具，也真有很多人愿意上他的当，最后真正迷失方向。说他坐不了程序吧，他也可以作，但是如果 程序复杂一点，出现问题时，问题出再哪里就搞不清楚了，反正是不清楚！

梁肇新，大牛啊，他说："我就搞不懂了，用鼠标怎么写程序呢？在我的公司里，高手的键盘响个不停，鼠标偶尔响一下，新手是鼠标响个不停，键盘偶尔响一下，他们的薪水相差的就不是一倍那么多了！"

C 语言是各大操作系统的基础，Unix、Linux、Windows 其内核都清一色是 C 语言开发的，(某些地方是和汇编语言混合开发的)，君 不见 WindowsAPI 都是 C 语言函数的接口？Unix/Linux 绝大多数应用都是 C 语言开发的；Windows 应用程序用纯 API 开发已然不多，大 多都是依靠某种 ApplicationFramework，比如所谓的 VC++，其实就是指 VCIDE+C++语言+MFC(现在重点已转向 ATL、 WTL)，但是 Windows 服务、网络、驱动程序等底层软件，还是 C 语言开发的。各种语言的编译器，包括 java 虚拟机，都是用 C 语言开发的。各种嵌入 式设备，如手机、PDA 也都是 C 语言开发的。

第一个要装进行囊和你一起前进是"规范的格式" 。所以说，规范的格式是入门的基础。那这个规范的格式包括什么呢？不少啊，要坚持才能做到！长标志符命名，代码缩进，一对大括号范围不超过一屏幕，等等。

第二个要装进行囊的是耐心，所谓工欲善其事，必先利其器。要想成功，没有一个相对平淡的过程是不可能的。这不仅仅指你在学习过程中要有耐心，要循 序渐进，而起也说的是另一个重要的方面：调式程序。调试是写程序过程中一个重要的方面，如果有人能一次写成程序，牛啊，而且是大牛，不光是大牛，还是老子 骑的那头青牛，凡人是做不道的！调试是每个程序必定经历的历程。

第一招，学什么呢？打狗棍法！呵呵

有一个伟人说过："重复权威是成熟的必经之路"，这是站在巨人的肩膀上的做法，习武之人首先要学的都是各种套路，比方说辟邪剑法，然后才能融会贯通，开宗流派；我们学习写程序也要这样来，这是一个捷径，帮你走得更远得捷径。

看书，看好书！书中所写，是前人数十年经验所写，看十本书，就相当于汲取了前人数十年的功力，那么你的内功也会迅速上升 1 甲子。：）书当然 要看好书，只有好书才营养丰富。要做到读书破万卷，编程如有神；枯燥的看书是很郁闷的，很容易变成化石！现在很多是书都是用源码说明问题的，源码就像是动 画、就像是幻灯片，把书中的招式一一演练给你看。 自己手工输入这一步不能省略，现在很多书有配套源码，很多同学或者成年人学习的时候都要小聪明，直接把源码复制过去编译运行，hoho,这是没有效果的。

TC 至少有一个好处，可以锻炼我们使用组合键的习惯，可以锻炼我们使用键盘编程的习惯。

一、要读就读好书，否则不如不读

Kernighan 和 Ritchie 的《TheCProgrammingLanguage》（中译名《C 程序设计语言》）堪称经典中的经典， 不过旧版的很多内容都已过时，和现在的标准 C 语言相去甚远，大家一定要看最新的版本，否则不如不看。另外，即使是最经典最权威的书，也没有办法面面俱到， 所以手边常备一本《C 语言参考手册》是十分必要的。《C 语言参考手册》就是《CReferenceManual》，是 C 语言标准的详细描述，包括绝大多数 C 标准库函数的细节，算得上是最好的标准 C 语言的工具书。顺便提一句，最新的《C 程序设计语言》是根据 C89 标准修订的，而《C 语言参考手册》描述的是 C99 标准，二者可能会有些出入，建议按照 C99 标准学习。还有一本《C 和指针》，写得也是相当地不错，英文名是《PointersonC》，特别地强调 指针的重要性，算是本书的一个特点吧。不过这本书并不十分适合初学者，如果你曾经学过 C 语言，有那么一些 C 语言的基础但又不是很扎实，那么你可以尝试一下 这本书。我相信，只要你理解了指针，C 语言便不再神秘。

如果你已经啃完了一本 C 语言教材，想要更进一步，那么有两本书你一定要看。首先是《CTrapsandPitfalls》（中译名《C 陷井 与缺陷》），很薄的一本小册子，内容非常非常地有趣。要注意一点，这本书是二十多年前写成的，里面提到的很多 C 语言的缺陷都已被改进，不过能够了解一些历 史也不是什么坏事。然后你可以挑战一下《ExpertCProgramming》（中译名《C 专家编程》），书如其名，这本书颇具难度，一旦你仔细读完并 能透彻理解，你便可以放心大胆地在简历上写"精通 C 语言"了。

切记一个原则，不要读自己目前还看不懂的书，那是浪费生命。如果你看不懂，那你一定是缺失了某些必需基础知识。此时，你要仔细分析自己需要补充哪些内容，然后再去书店寻找讲述的这些内容的书籍。把基础知识补充完毕再回头来学习，才会真正的事半功倍。

二、Unix/Linux 还是 Windows，这是个很大的问题

不同的编程环境会造就出不同思维的程序员。Windows 的程序员大多依赖集成开发环境，比如 VisualStudio，而 Unix 程序员更 加钟爱 Makefile 与控制台。显而易见，集成开发环境更容易上手，在 Windows 上学习 C 语言，只需要会按几个基本的 VisutalC++工具栏按钮就可以开始写 Hello,World!了，而在 Unix 下，你需要一些控制台操作的基本知识。有人也许认为 Unix 的环境更简洁，但习惯的力量是很大的 。

所以我建议初学者应该以 VisualC++6.0（不是 VisualC++.NET）或者 DevC++作为主要的学习环境，而且千万不要在 IDE 的使用技巧上过多纠缠，因为今后你一定要转向 Unix 环境的。VisualC++6.0 使用很方便，调试也很直观，但其默认的编译器对 C 标准的支持 并不好，而 DevC++使用 gcc 编译器，对 C99 的标准都支持良好。使用顺带提一下，很多大学的 C 语言课程还在使用 TurboC2.0 作为实验环境，这 是相当不可取的，原因其一是 TC2.0 对 C 标准几乎没有支持，其二是 TC2.0 编译得到的程序是 16 位的，这对今后理解 32 位的程序会造成极大的困扰（当 然,用 djgpp 之类的东西可以使 TC2.0 编译出 32 位程序，不过那过于复杂了）。

等你学完一本 C 语言的教材，你一定要转向 Unix 平台继续学习，几乎所有的 C 语言高级教程都是基于 Unix 平台的（比如《C 专家编程》）。转变的 过程是痛苦的，你需要面对的是各种纷繁复杂的命令，完全不同于 Windows 平台的思考方式，但是这种痛苦是值得的。Unix 与 C 是共生的，Unix 的思 考方式和习惯更加符合 C 语言的思考方式和习惯。在 Unix 下，你可以找到无数优秀的源代码供你尽情阅读，你可以方便地查看某个库函数的联机手册，还可以看 到最优秀的代码风格（说到代码风格，我会专门写一篇文章详细叙述）。

归结起来就是一句话：初学 C 语言，建议使用 Windows 系统和集成开发环境，在准备向"高手"方向努力时，请先转向 Unix 平台。

三、万事俱备，你就是东风

书已选定，环境配置完成，正所谓万事俱备，只欠你自己的努力了。请从书的前言开始，仔细地阅读手头的教材，很多人看书喜欢直接从第一章开始 看，这是错误的做法。前言是作者对整本书的大体介绍，作者一般会告诉你需要什么基础才能够顺利阅读本书，这可以帮助你检验自己的基础知识是否已经具备。看 完前言，还要浏览一下目录，了解一下书的整体结构，顺便给自己安排一下学习计划。

例子之后就是习题了，我建议初学者把所有的习题都独立做一遍。

也许你认为这样学习太慢，其实不然。学得细致就不用走回头路，等你学到后面才发现自己前面没搞清楚，那才是真的得不偿失。一般说来，整本书 读完，你应该完成数千行乃至上万行的代码，无论是原封不动照抄书上的，还是自己心血来潮写就的，都是今后继续学习的一笔财富。以我自己举例，阅读 《Windows 核心编程》时（我只阅读了 3/4 的内容），除了抄书上的代码，还自己写了很多例子，一共有 5574 行（用 unix 下的 wc 工具

统计），时 隔多日，我早已记不清 Windows 的系统编程了，但只要花几分钟翻出以前的代码看看，便会重新了然于胸。所谓好记性不如烂笔头，就是这个道理。

仔细读书、认真抄写源代码、独立完成习题外加更进一步的实验，最后将所有的代码留下，成为自己的经验和财富，绝对的辛苦，也绝对的事半功倍。当然，这种方式只适合学习需要精通的技术，如果不是学习 C 语言，你还要具体情况具体分析。

写到最后，还有非常非常重要的一点没有提及──代码风格，从最开始学习就必须强迫自己模仿最优秀的代码风格。因为代码风格太重要内容也太多，我会用专门的一篇文章来详细讨论，请大家关注《程序员之路──关于代码风格》。

在这里停一下，上面说的便是我学习的风格：总要经历一番波折，瞎折腾几下，然后才会有偶然间的明悟－开窍了！我甚至没见过几个人有我这么笨，现在 明白过来了，我总是一开始把事情想得过于复杂，造成狗咬刺猬的难堪的局面，然而竟然有意想不到的收获！但是不建议大家模仿这种风格，起码追女朋友的时候不 能这样，生活中很多机会在于接手的那一瞬间，失去就永远找不回来了。

现在回来，后来偶然间，我看懂了一行代码，是 print 语句，当时兴奋得要命，又仔细看了这个看了那个，一下子看懂了很多东西，像是在霎那间被什么给击中似的，立刻间醍醐灌顶。。。。

可是，还是那句话不破不立，不阻不行。就像我现在有了电脑，整天泡在互联网的海洋里，感受着快餐文化，好久都没静下心来好好想点东西了。

谈及 C 语言，我想凡是学过它的朋友都有这样一种感觉，那就是"让我欢喜让我忧。"欢喜的是，C 语言功能非常强大、应用广泛，一旦掌握了后，你 就可以理直气壮地对他人说"我是电脑高手！"，而且以后若是再自学其他语言就显得轻而易举了。忧虑的是，C 语言犹如"少林武功"一般博大精深，太难学了。 其实就笔者认为 C 语言并非是"difficult（困难）"的，只要你能理清思路，掌握它的精髓，那么自学 C 语言是一件非常容易且又其乐无穷的事。今天本 人就与大家一起谈谈如何学习 C 语言或者说学习 C 语言应从哪几方面着手。

就个人感触，无论学习哪门语言首先应该了解一下自己所学语言的背景，也可以说它的发展史。C 语言属于高级程序语言的一种，它的前身 是"ALGOL"。其创始人是布朗·W·卡尼汉和丹尼斯·M·利奇。C 语言问世时是带有很大的局限性，因为它只能用于 UNIX 系统上。然而随着科学技术的 进步，计算机工业的发展，C 语言逐渐脱离 UNIX。1987 年美国标准化协会制定了 C 语言的国际标准，简称"ANSIC"，从此以后它便成为一种广泛使用 的程序语言。C 语言的优点很多，主要的有如下四点：

1. 兼备高级语言与低级语言的优点，属于一种中间语言。
2. 它是一种结构化程序设计语言，非常适合结构化程序设计。
3. 有较丰富的数据类型、运算符以及函数供以选用。
4. 直接与内存打交道，使修改、编辑其他程序与文档变得轻松，简单。

怎样才能学好 c 语言？想尽快上手就得掌握计算机的特点，计算机的特点包括：

1. 计算机在问题的处理方式上要求全，将所有的可能都要告诉它。
2. 计算机要求程序的描述精确，无二义性。
3. 计算机编程是要求有很强的全局性和逻辑性，不存在起伏的问题。

计算机要求它的主人，考虑问题要全面，所有可能的情况及处理都要告诉它，要求学会沉稳，心态要稳定，要求交流的语句一定要明了含义单一。

怎样才能很快的学会 c 语言，更快的度过磨合期呢？C 语言的语法规则记忆理解当然是不可少的，除此之外还应注意以下几个方面：

1. 平衡心态，虽然不能做到"不以物喜，不以己悲"的水平，但至少不要浮躁，不要急于求成，欲速则不达。
2. 培养自身的全局意识，既能小无内，也能大无外，才行。
3. 严格按照程序设计过程设计程序，不要跳脱，天马行空，没有规矩是不成方圆的。
4. 努力提高自身的综合素质。程序是人思维的表达形式，是人处理问题思路和语言的结合体。你对客观看成到什么程度和你掌握的知识成正比。如果你对处理的问题不理解，不会处理，你怎么也写不出程序。
5. 学会交流，多交流，相互补益，同时团队合作也是很重要的。
6. C 语言的学习，一般的方式是，先学 C，然后是 C++，最好还要有汇编语言和微机原理基础，然后才是 Visual C++。这样的方式，对学习者来说，要花费很多时间和耐力。而在学校教学中，也没有时间深入学习 Windows 编程的实用技术了。

   其实，具有了 C 语言基础后，再有一些基本的 C++类的概念，就可以直接学习 Windows C 编程了。

一、走近 Windows C 语言

很多语言都把显示一个“Hello,World!”做为第一个入门程序， C 语言的第一个程序是这样的：

```
#include<stdio.h>
main()
{
   printf(“Hello,World!”);
}
```

假如把 main 函数写成带参数的 main 函数，应该是：

```
#include<stdio.h>
main(int arge,char *argv[])
```

```
{
   printf(“Hello,World!”);
}
```

Windows C 的第一个程序和这个程序在形式和原理上都是一致的，只是有两点不同：

1. 主函数接收的形参不只是命令行中的字符串的个数和字符串的首地址。

2. C 语言的很多函数在 Windows C 中都可以继续使用，但象 printf（）屏幕显示等函数就不能继续使用了。因为 Windows 是多任务操作系统，屏幕已不再为某一个应用程序所独 有，Windows C 应用程序要显示字符串，需要使用 Windows 提供的 API 函数，开自己的窗口

下面是一个最简单的，显示“Hello,World!”的 Windows C 程序：

```
#include<windows.h>
APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
{
   MessageBox(NULL,"Hello,World!","第一个 Windows C 程序
",MB_OKMB_ICONASTERISK);
}
```

主函数的形参有四个：

1) Hinstance：接收程序运行时当前实例的句柄；
2) HprivInstance：前一个实例的句柄；
3) LpCmdLine：程序命令行指针；
4) NcmdShow：一个用来指定窗口显示方式的整数。

这几个参数的使用我们会在深入的学习中介绍的。

显示 Hello,Word!字符串，我们使用了一个 MessageBox 函数，这个函数会在屏幕上显示一个对话框，它的原型是：

```
int MessageBox(HWND hWnd,LPCTSTR lpText,LPCTSTR lpCaption,UNIT uType)
```

四个参数分别是：

1) HWnd：父窗口的句柄；
2) LpText：要显示字符串的指针；
3) LpCaption：对话框标题字符串的指针；
4) UType：显示在对话框上的小图标的类型。

使用这个函数要包含 windows.h 头文件。

调试一下，怎么样？窗口上弹出了一个“第一个 Windows C 程序”对话框，上面有一行字：“Hello,World!”。

世界真的很美好啊！！

深入编程：

在 C 语言中，函数的声明，假如没有指明返回值类型，缺省值为 void，这个程序的主函数就没有返回值。不过，在 Windows 编程时，我们最好养成个好 习惯，指明函数的返回值类型，因为在 C++中，函数返回值类型是不可以缺省的。而我们在 Windows C 编程时，还是会用到 C++的一些概念，这样做，有利于以后深入地学习。

规范一点的程序应该是这样的：

```
#include<windows.h>
int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
{
MessageBox(NULL,"Hello,World!","第一个 Windows C 程序",MB_OKMB_ICONASTERISK);
return 0;
}
```

这里，我们声明的类型为 int 型，并且返回一个值 0，这样的函数就可以使用在复杂一点的函数调用中了。

在这一节中， 我们有几处都提到了句柄的概念，句柄和指针的概念不同，它是作为操作系统内部索引表中的一个值来使用的，这样可以防止应用程序直接访问名对象的内部结构， 体现了 Windows 资源治理的优越性。譬如说，一个窗口找开之后，好对应内存中的一个内存块，这个窗口所在的内存快地址往往会由操作系统做动态的调整， 但其却不会随之变化。不过，通过它可以访问这个窗口，所以在使用的时候，可以把它当做指针一样看待。

二、 获取本地计算机的主机名和 IP 地址

和 C 语言一样，函数是 Windows C 编程的最基本的单位。不过，Windows C 主要使用 API 函数，而网络编程则主要使用 Winsock 提供的 API 函数。

Winsock 是 90 年代初，为了方便网络编程，由 Microsoft 联合了其他几家公司共同制定的一套 WINDOWS 下的网络编程接口，它是通过 C 语 言的动态链接库方式提供给用户及软件开发者的，主要由 winsock.h 头文件和动态链接库 winsock.dll 组成，目前有两个版 本：

Winsock1.1 和 Winsock2.0。

在 Win32 平台上，访问众多的基层网络协议，Winsock 是首选接口。

用 Visual C++6.0 编译 Windows C 程序，使用 Winsock API 函数时，首先要把 wsock32.lib 添加到它的库模块中，否刚在链接的时候，会出现“error LNK2001”错误。添加 wsock32.lib 的具体步骤是：打开工程菜单，选择设置，在弹出的 Project settings 对话框中，点击 link 选项卡，然后在对象/库模块文本框中添加 wsock32.lib。

最简单的网络编程是获取本机的主机名和 IP 地址，这个程序使用了 WSAStart（）、WSAClenaup()、gethostname（）、gethostbyname（）四个 winsock API 函数,这四个函数的功能和使用方法介绍如下：

1． WSAStartup():

【函数原型】

int PASCAL FAR WSAStartup(WORD wVersionRequired, LPWSADATA lpWSAData);

【使用说明】

每一个使用 winsock 的应用程序，都必须进行 WSAStart 函数调用，并且只有在调用成功之后才能使用其它的 winsock 网络操作函数。

WVersionRequired：<输入>表示欲使用的Winsock 版本，这是一个 WORD 类型的整数，它的高位字节定义的是次版本号，低位字节定义的是主版本号。

LpWSAData：<输出>是一个指向 WSADATA 资料的指针。这个资料我们一般不使用。

返回值：调用成功返回 0；否则，返回出错信息。

2． WSAClenaup():

【函数原型】

int PASCAL FAR WSACleanup(void);

【使用说明】

winsock 使用后，要调用WSACleanup 函数关闭网络设备，以便释放其占用的资源。

3．gethostname()

【函数原型】

int PASCAL FAR gethostname (char FAR * name, int namelen);
【使用说明】

该函数可以获取本地主机的主机名，其中：

name：<输出>用于指向所获取的主机名的缓冲区的指针。

Namelen：<输入>缓冲区的大小，以字节为单位。

返回值：若无错误，返回 0；否则，返回错误代吗。

4．gethostbyname()

【函数原型】

strUCt hostent FAR * PASCAL FAR gethostbyname(const char FAR * name);
【使用说明】

该函数可以从主机名数据库中得到对应的“主机”。

该函数唯一的参数 name 就是前面调用函数 gethostname()得到的主机名。若无错误，刚返回一个指向 hostent 结构的批针，它可以标识一个“主机”列表。

Hostent 结构定义如下：

```
Struct hostent
{
char FAR * h_name;
char FAR FAR ** h_aliases;
short h_addrtype;
char FAR FAR ** h_addr_list;
}
```

其中：

h_name：<输入>主机名地址（PC）。
h_aliases：一个由主机备用名组成的空中止数组。

H_addrtype：返回地址的类型，对于 Winsock,这个域总是 PF_INET。
H_lenth：每个地址的长度（字节数），对应于 PF_INET 域应该为 4。
H_addr_list：应该以空指针结尾的主机地址的列表，返回的地址是以网络顺序排列的。

其中，h_addr_list[0]存放的就是本地主机的 4 个字节的 IP 地址，即：

h_addr_list[0][0].h_addr_list[0][1].h_addr_list[0][2].h_addr_list[0][3]

一个简单的用消息框显示主机名和 IP 地址的源程序如下：

```
#include<winsock.h>

int WSA_return;
WSADATA WSAData;

HOSTENT *host_entry;
char host_name[256];
char host_address[256];

int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
{
   WSA_return=WSAStartup(0x0101,&WSAData);

   if(WSA_return==0)
   {
gethostname(host_name,256);
host_entry=gethostbyname(host_name);
if(host_entry!=0)
{
   wsprintf(host_address,"%d.%d.%d.%d",
(host_entry->h_addr_list[0][0]&0x00ff),
(host_entry->h_addr_list[0][1]&0x00ff),
(host_entry->h_addr_list[0][2]&0x00ff),
(host_entry->h_addr_list[0][3]&0x00ff));

   MessageBox(NULL,host_address,host_name,MB_OK);

      }
   }
   WSACleanup();
```

```
    return 0;
}
```

深入编程：

前面显示 IP 地址的时候，我们使用的是消息框，规范一点的编程应该使用对话框，如何编辑一个对话框，很多书中都有介绍，编辑的对话框可参考图 5 的运行界面。

头文件 Get_IP.h 如下：

```
BOOL APIENTRY Hostname_ipDlgPro(HWND hDlg,UINT message,WPARAM
wParam,LPARAM lParam);
```

这个程序只使用了一个对话框过程，一般把这个过程的声明放在头文件中。

源程序 Get_IP.c：

```
#include<winsock2.h>
#include"Get_IP.h"
#include"resource.h" //这个头文件在创建资源的时候会自动生成,
//并会在插入资源时自动生成控件标识号.
int WSA_return;
WSADATA WSAData;

HOSTENT *host_entry;
char host_name[256];
char host_address[256];

int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
{
   WSA_return=WSAStartup(0x0101,&WSAData);
   if(WSA_return==0)
   {
gethostname(host_name,256);
host_entry=gethostbyname(host_name);
if(host_entry!=0)
{
   wsprintf(host_address,"%d.%d.%d.%d",
(host_entry->h_addr_list[0][0]&0x00ff),
(host_entry->h_addr_list[0][1]&0x00ff),
```

```
(host_entry->h_addr_list[0][2]&0x00ff),
(host_entry->h_addr_list[0][3]&0x00ff));
}
   }
   WSACleanup();
   DialogBox(hInstance,"DIALOG1",NULL,(DLGPROC)Hostname_ipDlgPro);
   return 0;
}

BOOL APIENTRY Hostname_ipDlgPro(HWND hDlg,UINT message,
WPARAM wParam,LPARAM lParam)
{
   switch(message)
   {
case WM_INITDIALOG:
   return(TRUE);
case WM_COMMAND:
   if(LOWORD(wParam)==IDOK)
   {
SetDlgItemText(hDlg,IDC_EDIT1,host_name);
SetDlgItemText(hDlg,IDC_EDIT2,host_address);
SetDlgItemText(hDlg,IDCANCEL,"确定");
   }
   if(LOWORD(wParam)==IDCANCEL)
EndDialog(hDlg,TRUE);
return(TRUE);
   break;
   }
   return(FALSE);
}
```

三、利用 VisualC++6.0 编译 Windows C 程序

利用 Visual C++6.0 编译 Windows C 程序一般要经过以下四个步骤：新建项目、添加代码、添加资源和编译链接。下面我们简单地介绍一下程序上面介绍的规范的获取本机的主机名和 IP 地址程序的编译过程：

(一) 新建项目

1．启动 MicrosoftVisualC++,然后在【文件】菜单中先择【新建】命令，弹出如图 1 所示的【新建】对话框：

7. 点击查看大图

8.

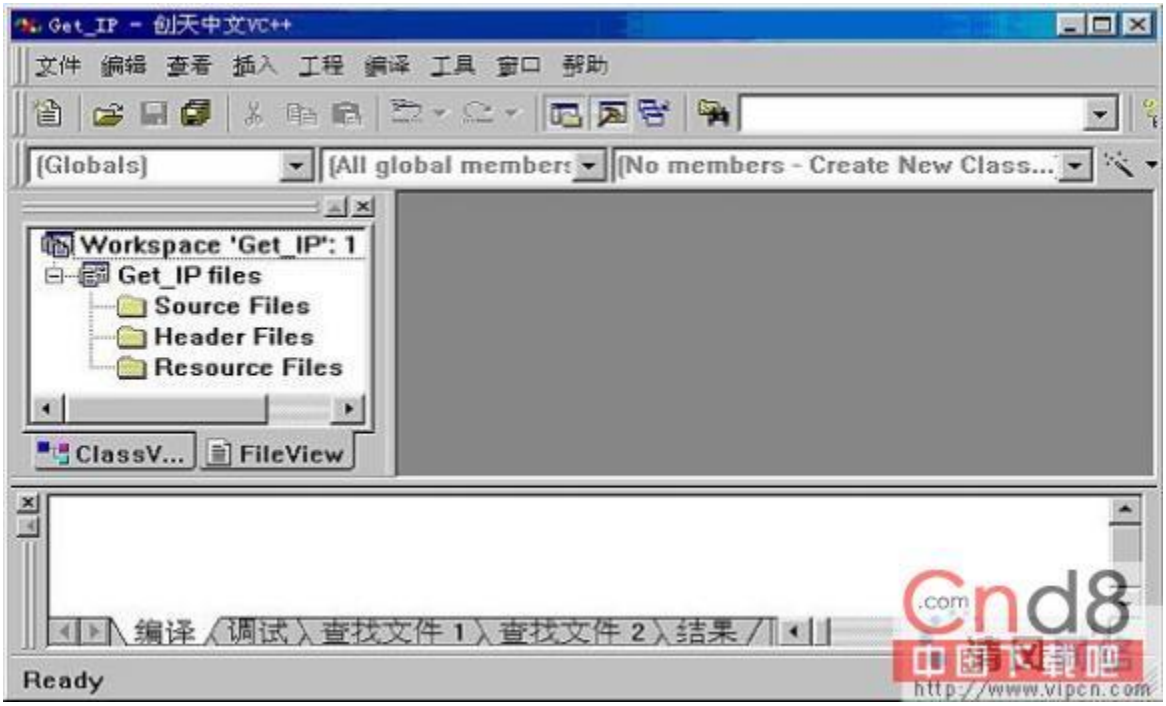
图 1

2．在【新建】对话框中，系统打开的是默认的【工程】选项卡，【工程】选项卡左侧的列表框中有多种建立工程的方式，我们选中“Win32 Application”选项。

3． 在【位置】文本框中输入新建工程的路径（例如：F:），在【工程】文本框中输入工程名称（例如：Get_IP）。

4． 选中【平台】列表框中的 Win32 复选框，然后单击【确定】按钮。

5． 在随后的对话框中，都选择默认设置，完成后，进入图 2 示界面：



9. 点击查看大图

10.

图 2

(二) 添加代码

在 VisualC++6.0 中，源代码一般存放在源代码文件和头文件中，往项目中添加源代码是非常方便的，为项目新建一个源代码文件一般要按下述方法操作：

1. 选择【工程】【添加工程】【新建】选项，弹出图 3 所示【新建】对话框：

11. 点击查看大图

12.

图 3

2． 在对话框的【文件】选项卡中，左侧的列表框选中“C++ Source File”选项，右侧选中【添加工程】复选框，并在【文件】文本框中输入源文件名（例如：Get_IP.c）。

3． 单击【确定】按钮，【新建】对话框将被闭，用户就可以在新建的 Get_IP.c 中输入程序的源代码了。

4． 添加头文件 Get_IP.h 的方法和上面所述过程一样，只是在【文件】选项卡中，左侧的列表框要先中“C/C++ Header File”选项。在【文件】文本框中输入头文件名（例如：Get_IP.h）。

(三) 添加资源

在添加资源前，必须在项目中先添加一个资源文件，然后可利用 Visual C++6.0 提供的资源编辑器为项目新建一个资源，具体步骤如下：

1． 选择【工程】【添加工程】【新建】选项，弹出图 3 所示【新建】对话框。

2． 在对话框的【文件】选项卡中，左侧的列表框选中“Rsource Script”选项，右侧选中【添加工程】复选框，并在【文件】文本框中输入资源文件名（例如：Get_IP.rc）。

3． 单击确定，回到主窗口后，选择【插入】【资源】选项，打开【插入资源】对话框，如图 4 所示， 在【资源类型】列表框中选中“Dialog”选项，单击【新建】按钮，返回主窗口后，即可利用对话框编辑器进行编辑了。编辑后的对话框如图

图 4

图 5

(四) 编译链接

在添加了源代码与资源文件后，就可以对程序编译连接了，可按 Ctrl+F7 键编译，按 F7 键连接，按 Ctrl+F5 键运行程序。在连接前是要注重，资源文件 Get_IP.rc 也要进行编译。

由于这个程序引用了 Winsock API 函数，在编译连接前，还要添加 wsock32.dll，具体方法前面已经介绍过，这里就不再赘述了。

一点看法：

利用 C 语言编写 Windows 应用程序有两种方式：一种是 Windows C 编程方式，另一种是 Visual C++编程方式。在一般情况下，Visual C++编程方式编写的程序源代码量小、开发时的工作量小、工作难度也较小，但编译后的代码量较大，运行速度略低；而 Windows C 编程方式编写的程序源代码量虽然较大，但可执行代码效率高。随着技术的进步，Visual C++编程方式已被广泛采用，但象网络编程等一些对速度要求高、对硬件操作较多的程序，大多数还是用 Windows C 编程方式开发的。另外，学习 Windows C 程序设计，还有助于更深入地了解 Windows 的内幕和 Windows API。

Objective-C 基础语法快速入门

Objective-C 是 Mac 软件开发领域最主要的开发语言，假如我们对面向对象的思维已经 C 语言都很熟悉的话，对于我们学习 Objective-C 将会非常有用。假如我们对 C 语言还不熟悉的话，那我们需要学习一下 C 语言。

方法调用(Calling Methods)

为了能够尽快上手，我们先来看一些简单的例子。Objective-C 语法里面基本的方法调用是这样的：

```
1. [object method];
2.
3. [object methodWithInput:input];
4.
```

对象的方法可以返回值:

```
1. output = [object methodWithOutput];
2.
3. output = [object methodWithInputAndOutput:input];
4.
```

我们也可以在类里面调用如何创建对象的方法。下面的这个例子里面，我们调用了 NSString 类的 string 方法：

```
1. id myObject = [NSString string];
2.
```

id 的类型意味着 myObject 这个变量可以指向任意类型的变量。当我们编译这个应用程序的时候，并不知道他实现的真实的类和方法。

在这个例子里面，很明显这个对象的类型应该是 NSString，所以我们可以改一下他的类型：

```
1. NSString* myString = [NSString string];
2.
```

现在 myString 就是一个 NSString 类型的变量。这个时候假如我们试图使用一个 NSString 没有实现的方法时，编译器就会警告我们。

一定要注意在对象类型的右边有一个星号。所有的 Objective-C 对象变量都是指针类型的。id 类型已经预先被定义成一个指针类型了。所以我们不需要再加星号。

嵌套消息调用(Nested Messages)

在许多编程语言里面嵌套消息，或者嵌套函数看起来就像这样：

```
1. function1 ( function2() );
```

function2 的返回值被传递给 function1 当输入参数。在 Objective-C 里面，嵌套消息调用就像这样：

```
1. [NSString stringWithFormat:[prefs format]];
```

我们应该尽量避免在一行代码里面嵌套调用超过两个。因为这样的话，代码的可读性就不太好。

多参输入的方法(Multi-Input Methods)

多个输入参数的方法。在 Objective-C 里面，一个方法名可以被分割成几段。在头文件里面，就应该这样子来定义一个多输入参数的方法：

```
1. -
   (BOOL)writeToFile:(NSString *)path atomically:(BOOL)useAuxiliaryFile;
2.
```

我们这样来调用它:

```
1. BOOL result = [myData writeToFile:@"/tmp/log.txt" atomically:NO];
2.
```

参数不一定要给它命名。在运行期系统里面这个方法真实的名字叫 writeToFile:atomically:。

```
1. Accessors(Getter & Setter)
2.
```

在 Objective-C 里面所有的实例对象默认都是私有的。所有在大多数情况下我们需要用 accessors 去读取或者设置变量的值。有两个语法都支持这样的操作，这个时传统的老的语法：

```
1. [photo setCaption:@"Day at the Beach"];
2. output = [photo caption];
3.
```

第二行的代码其实并非直接去读对象实例的变量。事实上它调用的是名叫 caption 的方法。在 Objective-C 里大多数情况下我们不需要给 getters 加 get 的前缀。

无论什么时候我们见到方括号，其实我们都是向一个对象或者一个类发送了一个消息。

Dot Syntax

在 Objective-C 2.0 里面，新增加了一个"."操作的语法。在 Mac OS X 10.5 里面就使用了 Objective-C 2.0 语法:

```
1. photo.caption = @"Day at the Beach";
2. output = photo.caption;
3.
```

我们两种方式都可以使用。但是在一个工程里面最好保持风格一致，只使用某一种。"." 操作只能够被使用在 setters 和 getters 里面，而不能用在一般意思的方法上。

创建对象

主要有两种方式来创建一个对象。第一种办法像这面这样：

```
1. NSString* myString = [NSString string];
2.
```

这是一种非常习惯性的风格。在这种方式情况下，我们创建的是系统自动释放(autoreleased)类型的对象。关于自动释放类型 autoreleased，我们以后会深入讨论一下。然而在许多情况下，我们需要手动的去创建对象：

```
1. NSString* myString = [[NSString alloc] init];
2.
```

这是一个嵌套的方法调用。第一个调用的 NSString 自己的 alloc 方法。这是一个相对比较底层的调用，因为他创建了内容，以及实例化了一个对象。

第二块代码调用了新创建对象的 init 方法。这个 init 方法实现了比较常用的基本设置，比如创建实例对象的参数。对于一般开发人员而言，实现这个客户的类的具体的细节并不清楚。

在一些情况下，我们可以用不通的初始化方式去赋初值：

```
1. NSNumber* value = [[NSNumber alloc] initWithFloat:1.0];
```

```
2.
```

基本的内存管理

假如我们正在为 Mac OS X 开发一个应用程序，我们可以选择是否启用垃圾回收机制。这就意味着我们不需要去考虑内存管理，除了一个特别复杂的情形我们需要处理一下。

然而，我们有的时候我们的开发环境没有垃圾回收机制，比如 iPhone 开发的时候就没有垃圾回收机制。在这种情况下，我们就需要了解一些基本的内存管理方面的概念。

假如我们手动的通过 alloc 创建了一个对象，我们需要用完这个对象后 release 它。我们不需要手动的去 release 一个 autoreleased 类型的对象，假如真的这样去做的话，我们的应用程序将会 crash。

这里有两个例子：

```
1. // string1 will be released automatically
2.
3. NSString* string1 = [NSString string];
4.
5. // must release this when done
6.
7. NSString* string2 = [[NSString alloc] init];
8.
9. [string2 release];
10.
```

就这个教程而言，我们可以人为 autoreleased 对象会在当前函数方法调用完成后被释放。

当然了，还有很多关于内存管理的只是我们需要学习，但是这需要我们了解更多的基本概念以后才能去涉及。

设计一个类的 Interface

就 Objective-C 语言而言，创建一个类非常简单。它非常典型的分成了两个部分。

类的接口通常保存在 ClassName.h 文件里，它定义了实例的参数，以及一些公开的方法。

类的实现在 ClassName.m 文件里。它包含了真正运行的代码和那些方法。它还经常定义一些私有的方法。这些私有的方法对于子类是不可见的。

这里有一个接口文件的大概。类名 Photo，所以文件名叫 Photo.h:

```
1. #import
2.
3. @interface Photo : NSObject {
```

```
4.
5. NSString* caption;
6.
7. NSString* photographer;
8.
9. }
10.
11.  @end
12.
```

首先，我们把 Cocoa.h import 进来。Cocoa 的应用程序的所有的基本的类大多都是这样做的。#import 宏指令会自动的避免把同一个文件包含多次。

@interface 符号表明这是 Photo 类的声明。冒号指定了父类。上面这个例子父类就是 NSObject。

在大括弧里面，有两个变量：caption 和 photographer。两个都是 NSString 类型的。当然了，他们也可以是任何别的类型包括 id 类型的。

最后@end 结束整个声明。

增加方法

让我们为成员变量加一些 getters:

```
1. #import
2.
3. @interface Photo : NSObject {
4.
5. NSString* caption;
6.
7. NSString* photographer;
8.
9. }
10.
11.  - caption;
12.
13.  - photographer;
14.
15.  @end
16.
```

别忘记，Objective-C 方法不需要加 get 前缀。一个单独小横杆表明它是一个实例的方法。假如是一个加号的话，那就说明它是一个类的方法。

编译器默认的方法的返回类型为 id。还有所有的方法的参数的默认类型也都是 id 类型的。所以上面的代码从技术上讲是对的。但是很少这么用。我们还是给它加上返回类型吧：

```
1. #import
```

```
2.
3. @interface Photo : NSObject {
4.
5. NSString* caption;
6.
7. NSString* photographer;
8.
9. }
10.
11.  - (NSString*) caption;
12.
13.  - (NSString*) photographer;
14.
15.  @end
16.
```

下面我们再加上 setters：

```
1. #import
2.
3. @interface Photo : NSObject {
4.
5. NSString* caption;
6.
7. NSString* photographer;
8.
9. }
10.
11.  - (NSString*) caption;
12.
13.  - (NSString*) photographer;
14.
15.  - (void) setCaption: (NSString*)input;
16.
17.  - (void) setPhotographer: (NSString*)input;
18.
19.  @end
20.
```

Setters 不需要返回任何值，所以我们把它的类型指定为 void.

类的实现

我们通过实现 getters 来创建一个类的实现：

```
1. #import "Photo.h"
2.
3. @implementation Photo
4.
5. - (NSString*) caption {
6.
7. return caption;
8.
```

```
9. }
10.
11.  - (NSString*) photographer {
12.
13.  return photographer;
14.
15.  }
16.
17.  @end
18.
```

这部分的代码由@implementation 再来加上类名开始，以@end 结束。就跟类的接口定义一样，所有的方法跟接口定义里的一样。所有的对象都必要既要定义也要实现。

假如我们以前也写过代码的话，Objective-C 里面的 getters 看上去跟别的差不多。所以我们下面就来介绍 setters，它需要一点说明。

```
1. - (void) setCaption: (NSString*)input
2.
3. {
4.
5. [caption autorelease];
6.
7. caption = [input retain];
8.
9. }
10.
11.  - (void) setPhotographer: (NSString*)input
12.
13.  {
14.
15.  [photographer autorelease];
16.
17.  photographer = [input retain];
18.
19.  }
20.
```

每个 setter 处理两个变量。第一个是当前存在对象的应用。第二个是新的输入对象。在支持垃圾回收的开发环境里，我们只要直接赋新值就可以了：

```
1. - (void) setCaption: (NSString*)input {
2.
3. caption = input;
4.
5. }
6.
```

但是假如我们不可以用垃圾回收机制的话，我们就需要先 retain 旧的对象，然后 retain 新的对象。

有两种方法可以释放一个引用对象：release 和 autorelease。标准的 release 会直接删除引用。autorelease 方法会在将来的某个时候去 release 它。在它声明周期结束前， 它会毫无疑问的存在。在本例中，上面 setPhotographer 中的 photographer 对象，将会在函数结束的时候被释放。

在 setter 里面用 autorelease 是安全的，因为新对象跟老的对象有可能是同一个对象有可能指向的是同一个对象。对于一个我们即将 retain 的对象，我们不应该立即 release 它。

这个也许现在看起来会困惑，但是随着我们的学习，会越来越能理解它。现在我们不需要立刻完全理解它。

初始化

我们可以创建一个初始化方法去给类的实例的成员变量赋初值：

```
1. - (id) init
2.
3. {
4.
5. if ( self = [super init] )
6.
7. {
8.
9. [self setCaption:@"Default Caption"];
10.
11.  [self setPhotographer:@"Default Photographer"];
12.
13.  }
14.
15.  return self;
16.
17.  }
18.
```

上面的代码感觉没啥好解释的，虽然第二行代码好像看上去没啥用。这个是一个单等于号，就是把[super init]的值赋给了 self。

它基本上是在调用父类去实现它的初始化。这个 if 代码段是设置默认值之前验证初始化是否成功。

释放资源 Dealloc

这个 dealloc 方法是在当一个对象希望被从内容里面删除的时候调用。这个我们释放在子类里面引用成员变量的最好的时机：

```
1. - (void) dealloc
2.
```

```
3. {
4.
5. [caption release];
6.
7. [photographer release];
8.
9. [super dealloc];
10.
11.  }
12.
```

开始两行我们发送了 release 通知给了两个成员变量。我们不要在这里用 autorelease。用标准的 release 更快一点。

最后一行的[super dealloc];非常重要。我们必须要发送消息去让父类清除它自己。假如不这么做的话，这个对象其实没有被清除干净，存在内存泄露。

dealloc 在垃圾回收机制下不会被调用到。取而代之的是，我们需要实现 finalize 方法。

更多关于内存管理

Objective-C 的内存管理系统基于引用记数。所有我们需要关心的就是跟踪我们引用，以及在运行期内是否真的释放了内存。

用最简单的术语来解释，当我们 alloc 一个对象的时候，应该在某个时候 retain 了它。每次我们调用了 alloc 或者 retain 之后，我们都必须要调用 release。

这就是引用记数理论。但是在实践的时候，只有两种情况我们需要创建一个对象：

1. 成为一个类的成员变量

2. 只临时的在一个函数里面被使用

在更多的时候，一个成员变量的 setter 应该仅仅 autorelease 旧的对象，然后 retain 新的对象。我们只需要在 dealloc 的时候调用 release 就可以了。

所以真正需要做的就是管理函数内部的 local 的引用。唯一的原则就是：假如我们 alloc 或者 copy 了一个对象，那么我们在函数结束的时候需要 release 或者 autorelease 它。假如我们是通过别的方式创建的，就不管。

这里是管理成员对象的例子：

```
1. - (void) setTotalAmount: (NSNumber*)input
2.
3. {
4.
```

```
5. [totalAmount autorelease];
6.
7. totalAmount = [input retain];
8.
9. }
10.
11.  - (void) dealloc
12.
13.  {
14.
15.  [totalAmount release];
16.
17.  [super dealloc];
18.
19.  }
20.
```

这里是本地引用的例子。我们只需要 release 我们用 alloc 创建的对象：

```
1. NSNumber* value1 = [[NSNumber alloc] initWithFloat:8.75];
2.
3. NSNumber* value2 = [NSNumber numberWithFloat:14.78];
4.
5. // only release value1, not value2
6.
7. [value1 release];
8.
```

这里是用本地引用对象去设一个成员变量的例子：

```
1. NSNumber* value1 = [[NSNumber alloc] initWithFloat:8.75];
2.
3. [self setTotal:value1];
4.
5. NSNumber* value2 = [NSNumber numberWithFloat:14.78];
6.
7. [self setTotal:value2];
8.
9. [value1 release];
10.
```

注意到如何管理本地引用其实都是一样的。不管你是否把它设给了一个成员变量。我们无须考虑 setters 的内部实现。

如果我们很好的理解了这些的话，我们基本上理解了 80%的 Objective-C 内存管理方面的内容了。

属性 Properties

前面我们写 caption 和 author 的 accessors 的时候，你可以已经注意到了代码非常简明，应该可以被抽象提取出来。

属性在 Objective-C 里是一个新的功能。他可以让我们自动的生成 accessors，另外还有一些别的优点。我们可以把上面 Photo 的类转成用属性来实现：

上面那个类原先的实现是这样：

```
1. #import
2.
3. @interface Photo : NSObject {
4.
5. NSString* caption;
6.
7. NSString* photographer;
8.
9. }
10.
11.  - (NSString*) caption;
12.
13.  - (NSString*) photographer;
14.
15.  - (void) setCaption: (NSString*)input;
16.
17.  - (void) setPhotographer: (NSString*)input;
18.
19.  @end
20.
```

假如用属性来实现就是这样：

```
1. #import
2.
3. @interface Photo : NSObject {
4.
5. NSString* caption;
6.
7. NSString* photographer;
8.
9. }
10.
11.  @property (retain) NSString* caption;
12.
13.  @property (retain) NSString* photographer;
14.
15.  @end
16.
```

@property 是 Objective-C 来声明属性的编译指令。括号里面的"retain"指明了 setter 需要 retain 输入的对象。这行其他的部分指定了属性的类型以及名字。

下面让我们来看看这个类的实现：

```
1. #import "Photo.h"
2.
```

```
3. @implementation Photo
4.
5. @synthesize caption;
6.
7. @synthesize photographer;
8.
9. - (void) dealloc
10.
11.  {
12.
13.  [caption release];
14.
15.  [photographer release];
16.
17.  [super dealloc];
18.
19.  }
20.
21.  @end
22.
```

@synthesize 指令自动的生成了我们的 setters 和 getters。所以我们只需要实现类的 dealloc 方法。

Accessors 只有当他们原先没有的时候，才会被生成。所以可以放心大胆的去用 @synthesize 来指定属性。而且可以随意实现你自己的 getter 和 setter。编译器会自己去找哪个方法没有。

属性声明还有别的选项，但是限于篇幅层次，我们下次再介绍。

Logging

在 Objective-C 里，往 console 写日记非常简单。事实上 NSLog()跟 C 语言的 printf()两个函数几乎完全相同，除了 NSLog 是用额外的“%@”去获得对象。

```
1. NSLog ( @"The current date and time is: %@", [NSDate date] );
2.
```

我们可以 log 一个对象到 console 里去。NSLog 函数调用要输出对象的 description 方法，然后打印返回的 NSString。我们可以在自己的类里重写 description 方法，这样我们就可以得到一个自定义的字符串。

调用 nil 对象的方法(Calling Methods on Nil)

在 Objective-C 里，nil 对象被设计来跟 NULL 空指针关联的。他们的区别就是 nil 是一个对象，而 NULL 只是一个值。而且我们对于 nil 调用方法，不会产生 crash 或者抛出异常。

这个技术被 framework 通过多种不同的方式使用。最主要的就是我们现在在调用方法之前根本无须去检查这个对象是否是 nil。假如我们调了 nil 对象的一个有返回值的方法，那么我们会得到一个 nil 返回值。

我们可以通过 nil 对象让我们的 dealloc 函数实现看上去更好一些：

```
1. - (void) dealloc
2. {
3. self.caption = nil;
4. self.photographer = nil;
5. [super dealloc];
6. }
7.
```

之所以可以这么做是因为我们给把 nil 对象设给了一个成员变量，setter 就会 retain nil 对象(当然了这个时候 nil 对象啥事情也不会做)然后 release 旧的对象。这个方式来释放对象其实更好，因为这样做的话，成员变量连指向随机数 据的机会都没有，而通过别的方式，出现指向随机数据的情形机会不可避免。

注意到我们调用的 self.VAR 这样的语法，这表示我们正在用 setter，而且不会引起任何内存问题。假如我们直接去设值的话，就会有内存溢出：

```
1. // incorrect. causes a memory leak.
2. // use self.caption to go through setter
3. caption = nil;
4.
```

Categories

Categories 是 Objective-C 里面最常用到的功能之一。 基本上 category 可以让我们给已经存在的类增加方法，而不需要增加一个子类。而且不需要知道它内部具体的实现。

如果我们想增加某个 framework 自带的类的方法，这非常有效。如果我们想在我们程序工程的 NSString 能够增加一个方法，我们就可以使用 category。甚至都不需要自己实现一个 NSString 的子类。

比如，我们想在 NSString 里面增加一个方法来判断它是否是一个 URL，那我们就可以这么做：

```
1. #import
2.
3. @interface NSString (Utilities)
4.
5. - (BOOL) isURL;
6.
7. @end
8.
```

这跟类的定义非常类似。区别就是 category 没有父类，而且在括号里面要有 category 的名字。名字可以随便取，但是习惯叫法会让人比较明白 category 里面有些什么功能的方法。

这里是具体的实现。但是要注意，这本身并不是一个判断 URL 很好的实现。我们主要是为了整体的了解 category 的概念。

```
1. #import "NSString-Utilities.h"
2.
3. @implementation NSString (Utilities)
4.
5. - (BOOL) isURL
6.
7. {
8.
9. if ( [self hasPrefix:@"http://"] )
10.
11.  return YES;
12.
13.  else
14.
15.  return NO;
16.
17.  }
18.
19.  @end
20.
```

现在我们可以在任何的 NSString 类对象里都可以调用这个方法了。下面的代码在 console 里面打印的"string1 is a URL"：

```
1. NSString* string1 = @"http://www.CocoaDev.cn/";
2.
3. NSString* string2 = @"Pixar";
4.
5. if ( [string1 isURL] )
6.
7. NSLog (@"string1 is a URL");
8.
9. if ( [string2 isURL] )
10.
11.  NSLog (@"string2 is a URL");
12.
```

跟子类不一样，category 不能增加成员变量。我们还可以用 category 来重写类原先的存在的方法，但是这需要非常非常小心。

记住，当我们通过 category 来修改一个类的时候，它对应用程序里的这个类所有对象都起作用。

后记

上面 Objective-C 的比较基础的大概的讲了一下。Objective-C 还是比较好上手的。没有特别的语法需要去学习。而且一些概念在 Objective-C 里面被反复运用。

C++面向对象快速入门

1.初窥输入输出

C++并没有直接定义进行输入输出的任何语句，这种功能由标准库提供（iostream 库）。定义了四个 IO 对象：cin、cout、cerr、clog

```
#include<iostream>

using namespace std;
int main()
{
    int var1, var2, sum;
    cout << "input var1 and var2" << endl;
    cin >> var1 >> var2;
    sum = var1+var2;
    cout << "the sum is:" << sum << endl;
return 0;
}
```

注意这里输出操作返回的是输出流本身

endl 是一个特殊值，叫做操纵符（manipulator），把它写入输出流就会具有输出换行的作用，并刷新与设备关联的 buffer

读入未知数目的输入：

```
#include<iostream>

using namespace std;
```

```
int main()
{
    int var, sum;
    while (cin >> var)
        sum += var;
    cout << sum << endl;
    return 0;
}
```

这里我们可以看出，每次输入一个空格，cin 就读取下一个值，输入完后 enter，cin 结束。但是如果你要求 cin 两个值，现在输入一个之后 enter，将会继续读取下一个值
如果你在输入的过程中输入了更多的值以至于当前的 cin 没有全部读到变量中，但是仍然读入到输入流，你在下次 cin 的时候就会获取上次没有赋给变量的那个值

2.类的简介

C++通过定义类来定义自己的数据结构
Sales_item item 说明 item 是类型 Sales_item 的一个对象（对象是指内存中具有类型的区域）

3.变量和基本类型

初始化的两种类型：复制初始化和直接初始化

```
int var(1024);  //direct-initialization
int var = 1024; //copy-initialization
```

对类类型的对象来说，有些初始化仅能用直接初始化完成。

变量初始化规则：

1.内置类型变量的初始化

在函数体外定义的变量初始化成 0

在函数体内定义的变量不初始化

2.类类型变量的初始化

每个类都通过构造函数定义了该类型的对象如何进行初始化，但是如果我们定义某个类的变量时没有提供初始化式，这个类也可以定义初始化时的操作。通过默认构造函数来实现。

声明和定义

变量的声名用于向程序表明变量的类型和名字
变量的定义用于为变量分配存储空间，也可以同时指定初始值

extern 声名不是定义，不分配存储空间

引用

```
1. int ival = 1024;
2. int &refval = ival;
```

对 refval 操作就是对 ival 操作，引用只是对象的另外一个名字。

const 引用是指向 const 对象的引用

```
1. const int ival = 1024;
2. const int &refval = ival;
3. int &refval2 = ival; //error
```

这样的话可以读取 refval 但是不能修改 refval

注解：非 const 引用只能绑定到与该引用同类型的对象，const 引用则可以绑定到不同但相关的类型的对象或绑定到右值

C++支持 struct 关键字，如果使用 class 定义类，定义在第一个访问标号前的任何成员都隐式指定为 private，struct 则都是 public。

C++的第三章主要介绍了一下标准库和最重要的标准库类型：string 和 vector

当我们使用某个命名空间中的名字，一般都要通过 std::cout 这样的写法来引用以避免重名，但是如果我们觉得麻烦的话也可以通过 using 声明：

```
1. using namespace std;
2. using std::cout;
```

但是有一种情况必须总是使用完全限定的标准库名字：在头文件中。理由是头文件中的内容会被预处理器复制到程序中。如果在头文件中加上 using 声明就相当于在整个程序中加上了这个声明。

标准库 String 类型

1.String 对象的定义和初始化

```
1. string s1;          //默认构造函数为空串
2. string s2(s1);      //初始化 s2 为 s1 的一个副本，修改 s2 不会影响 s1
3. string s3 = "value";
4. string s3("value");  //将 s3 初始化为一个字符串字面值副本
5. string s4(n, 'c');   //将 s4 初始化为字符 c 的 n 个副本
```

当然，我们也可以通过 cin 来获取 string 的值，对于 string 类型的输入操作符有以下规定：

a.读取并忽略开头所有的空白字符
b.读取字符直至再次遇到空白字符，读取终止

那么如何读取多个包含空格的字符串呢？getline
这 个函数接受两个参数：一个输入流对象和一个 string 对象，getline 从输入流的下一行读取，并保存读取的内容到 string 中，不包括换行符。和 输入操作符不一样的地方在于 getline 不会忽略开头的换行符，只要遇到换行符哪怕是第一个读取也会结束，string 参数就是空。

2.string 对象的操作

```
s.empty()  //如果 s 为空串则返回 true，否则返回 false
s.size()   //获取 s 的长度
s[n]       //s 的第几个字符，从 0 开始计数
s1 + s2    //把 s1 和 s2 连接起来返回新的字符
s1 = s2    //把 s1 内容替换成 s2 的副本
v1 == v2   //比较是否相等
!=, <, <=  //就是惯有含义
```

string::size_type 类型：
string 类类型和许多其他库类型都定义了一些配套类型。通过这些配套类型，库类型的使用就能与机器无关，size_type 类型定义为与 unsigned 型具有相 同含义，而且可以满足足够大能够存储任意 string 对象的长度。为了避免溢出，保存一个 string 对象 size 最安全的方法就是使用标准库类型 string::size_type

```
for(string::size_type i=0; i<s.size(); i++)
```

关系操作符的比较规则：
a.如果两个对象长度不同，且短的对象与长的对象前面部分匹配，则短的对象小于长的对象
b.如果两个对象自负不同，则比较第一个不匹配的字符

我们在进行字符串字面值连接的时候可以任意匹配相加：

```
string s2 = s1 + " " + "world";
```

标准库 vector 类型
我们把 vector 称作容器，是因为它可以包含其他对象，一个容器中的所有对象都必须是同一种类型的
vector 是一个类末模板(class template)，使用模板可以编写一个类定义或函数定义，而用于多个不同的数据类型

```
vector<int> ivec;
```

```
2. vector(sales_item) sales_vec;
```

1.vector 对象的定义和初始化

套路基本一样

```
1. vector<T> v1;  //默认构造函数 v1 为空
2. vector<T> v2(v1);
3. vector<T> v3(n,i);
4. vector<T> v4(n);
```

值初始化的问题：如果保存内置类型(int)的元素，用 0 值初始化，如果是有构造函数的类类型(string)，就用类的构造函数来进行初始化

2.vector 对象的操作

```
1. v.empty()          //判断是否为空
2. v.size()           //返回元素个数
3. v.push_back(t)     //在末尾添加一个值为 t 的元素
4. v[n]               //返回位置为 n 的元素
5. v1 = v2            //将 v1 的元素替换成 v2 中元素的副本
6. v1 == v2           //判断是否相等
7. !=, <, <=          //保持惯有含义
```

vector 对象的 size：

使用 size_type 的时候，必须指出该类型是哪里定义的。vector 类型包括元素类型

```
1. vector<int>::size_type    //ok
2. vector::size_type         //error
```

3.迭代器简介

迭代器是一种检查容器内元素并遍历元素的数据类型(每种标准容器都定义了一种迭代器类型 vector<int>::iterator iter)

```
for(vector<string>::iterator iter = string_vec.begin();iter != string_vec.end(); iter++)
```

当然还有一种 const_iterator 的类型，这种类型只能读取容器内的元素。

不要把 const_iterator 对象和 const 的 iterator 对象混淆了，前者是可以任意读取可以 iter++，但是 const vector<int>::iterator iter 的话*iter 可变，iter 不可变。

标准库 bitset 类型

定义 bitset 的时候，要明确 bitset 含有多少位（bitset<32> bitvec;）

1.bitset 对象的定义和初始化

```
bitset<n> b;          //b 有 n 位，都是 0
bitset<n> b(u);       //b 是 unsigned long 型 u 的一个副本
bitset<n> b(s);       //b 是 string 对象 s 中含有的位串的副本
bitset<n> b(s, pos, n);  //b 是 s 中从位置 pos 开始的 n 个位的副本
```

同样，bitset 对象上一样有很多的操作，这里就不一一例举了

C# 快速入门

C# 是一种具有 C++ 特性，Java 样式及 BASIC 快速建模特性的编程语言。如果你已经知晓 C++ 语言，本文将在不到一小时的时间内带 你快速浏览 C# 的语法。如果熟悉 Java 语言，Java 的编程结构、打包和垃圾回收的概念肯定对你快速学习 C# 大有帮助。所以我在讨 论 C# 语言构造的时候会假设你知道 C++。

本文通过一系列例程以简短但全面的方式讨论了 C# 语言构造和特性，所以你仅需略览代码片刻，即可了解其概念。

注意：本文不是为 C# 宗师而写。有很多初学者的 C# 文章，这只是其中之一。

接下来关于 C# 的讨论主题：

编程结构
命名空间
数据类型
变量
运算符与表达式
枚举
语句
类与结构
修饰符
属性
接口
函数参数
数组
索引器
装箱与拆箱
委托
继承与多态

以下主题不会进行讨论：

C++ 与 C# 的共同点
诸如垃圾回收、线程、文件处理等概念
数据类型转换
异常处理
.NET 库

编程结构

和 C++ 一样，C# 是大小写敏感的。半角分号（;）是语句分隔符。和 C++ 有所区别的是，C# 中没有单独的声明（头）和实现（CPP）文件。所有代码（类声明和实现）都放在扩展名为 cs 的单一文件中。

看看 C# 中的 Hello World 程序。

C# code
?

```
1  using System;
2
3  namespace MyNameSpace
4
5  {
6
7  class HelloWorld
8
9  {
10     static void Main(string[] args)
11     {
12         Console.WriteLine ("Hello World");
13      }
14 }
15
16 }
```

C# 中所有内容都打包在类中，而所有的类又打包在命名空间中（正如文件存与文件夹中）。
和 C++ 一样，有一个主函数作为你程序的入口点。C++ 的主函数名为 main，而 C# 中是大写 M 打头的 Main。

类块或结构定义之后没有必要再加一个半角分号。C++ 中是这样，但 C# 不要求。

命名空间

每个类都打包于一个命名空间。命名空间的概念和 C++ 完全一样，但我们在 C# 中比在 C++ 中更加频繁的使用命名空间。你可以用点（.）定界符访问命名空间中的类。上面的 Hello World 程序中，MyNameSpace 是其命名空间。

现在思考当你要从其他命名空间的类中访问 HelloWorld 类。

C# code

?

```
using System;
namespace AnotherNameSpace
{
    class AnotherClass
    {
        public void Func()
        {
            Console.WriteLine ("Hello World");
        }
    }
}
```

现在在你的 HelloWorld 类中你可以这样访问：

C# code

?

```
using AnotherNameSpace; // 你可以增加这条语句
namespace MyNameSpace
{
class HelloWorld
{
    static void Main(string[] args)
    {
        AnotherClass obj = new AnotherClass();
```

```
9           obj.Func();
10      }
11  }
12  }
```

在 .NET 库中，System 是包含其他命名空间的顶层命名空间。默认情况下存在一个全局命名空间，所以在命名空间外定义的类直接进到此全局命名空间中，因而你可以不用定界符访问此类。
你同样可以定义嵌套命名空间。

Using

#include 指示符被后跟命名空间名的 using 关键字代替了。正如上面的 using System。System 是最基层的命名空间，所有其他命名空间和类都包含于其中。System 命名空间中所有对象的基类是 Object。

变量

除了以下差异，C# 中的变量几乎和 C++ 中一样：

C# 中（不同于 C++）的变量，总是需要你在访问它们前先进行初始化，否则你将遇到编译时错误。故而，不可能访问未初始化的变量。
你不能在 C# 中访问一个“挂起”指针。
超出数组边界的表达式索引值同样不可访问。
C# 中没有全局变量或全局函数，取而代之的是通过静态函数和静态变量完成的。

数据类型
所有 C# 的类型都是从 object 类继承的。有两种数据类型：

基本/内建类型
用户定义类型

以下是 C# 内建类型的列表：

类型	字节	描述
byte	1	unsigned byte
sbyte	1	signed byte
short	2	signed short
ushort	2	unsigned short
int	4	signed integer
uint	4	unsigned integer
long	8	signed long
ulong	8	unsigned long
float	4	floating point number
double	8	double precision number
decimal	8	fixed precision number
string	-	Unicode string
char	-	Unicode char
bool	true, false	boolean

注意：C# 的类型范围和 C++ 不同。例如：long 在 C++ 中是 4 字节而在 C# 中是 8 字节。
bool 和 string 类型均和 C++ 不同。bool 仅接受真、假而非任意整数。

用户定义类型文件包含：

类 （class）
结构（struct）
接口（interface）

以下类型继承时均分配内存：

值类型
参考类型

值类型

值类型是在堆栈中分配的数据类型。它们包括了：

除字符串，所有基本和内建类型
结构

枚举类型

引用类型

引用类型在堆（heap）中分配内存且当其不再使用时，将自动进行垃圾清理。和 C++ 要求用户显示创建 delete 运算符不一样，它们使用新运算符创建，且没有 delete 运算符。在 C# 中它们自动由垃圾回收系统回收。

引用类型包括：

类
接口
集合类型如数组
字符串

枚举

C# 中的枚举和 C++ 完全一样。通过关键字 enum 定义。

例子：

C# code

?

```
1enum Weekdays
2{
3    Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday
4}
```

类与结构

除了内存分配的不同外，类和结构就和 C++ 中的情况一样。类的对象在堆中分配，并使用 new 关键字创建。而结构是在栈（stack）中进行分配。C# 中的结构属于轻量级快速数据类型。当需要大型数据类型时，你应该创建类。

例子：

C# code

?

```
struct Date
{
    int day;
    int month;
    int year;
}

class Date
{
    int day;
    int month;
    int year;
    string weekday;
    string monthName;
    public int GetDay()
    {
        return day;
    }
    public int GetMonth()
    {
        return month;
    }
```

```
22      public int GetYear()
23      {
24          return year;
25      }
26      public void SetDay(int Day)
27      {
28          day = Day ;
29      }
30      public void SetMonth(int Month)
31      {
32          month = Month;
33      }
34      public void SetYear(int Year)
35      {
36          year = Year;
37      }
38      public bool IsLeapYear()
39      {
40          return (year/4 == 0);
41      }
42      public void SetDate (int day, int month, int year)
43      {
44      }
        //...
    }
45
46
```

属性

如果你熟悉 C++ 面向对象的方法，你一定对属性有自己的认识。对 C++ 来说，前面例子中 Date 类的属性就是 day、month 和 year，而你添加了 Get 和 Set 方法。C# 提供了一种更加便捷、简单而又直接的属性访问方式。

所以上面的类应该写成这样：

C# code
?

```
1  using System;
2  class Date
3  {
4      public int Day{
5          get {
6              return day;
7          }
8          set {
9              day = value;
10         }
11     }
12     int day;
13
14     public int Month{
15         get {
16             return month;
         }
```

```
17          set {
18              month = value;
19          }
20      }
21      int month;
22
23      public int Year{
24          get {
25              return year;
26          }
27          set {
28              year = value;
29          }
30      }
31      int year;
32
33      public bool IsLeapYear(int year)
34      {
35          return year%4== 0 ? true: false;
36      }
37      public void SetDate (int day, int month, int year)
38      {
39          this.day = day;
40          this.month = month;
41          this.year = year;
      }
```

```
42}
43
44
```

这里是你 get 和 set 属性的方法：

C# code

```
1
  class User
2
  {
3
     public static void Main()
4
     {
5
          Date date = new Date();
6
          date.Day = 27;
7
          date.Month = 6;
8
          date.Year = 2003;
9
          Console.WriteLine
10
           ("Date: {0}/{1}/{2}", date.Day, date.Month, date.Year);
11
      }
  }
12
```

修饰符

你必须知道 C++ 中常用的 public、private 和 protected 修饰符。我将在这里讨论一些 C# 引入的新的修饰符。

readonly

readonly 修饰符仅用于修饰类的数据成员。正如其名字说的，一旦它们已经进行了写操作、直接

初始化或在构造函数中对其进行了赋 值，readonly 数据成员就只能对其进行读取。
readonly 和 const 数据成员不同之处在于 const 要求你在声明时进行直接初始 化。看下面的例程：

C# code
?

```
class MyClass
{
    const int constInt = 100; //直接进行
    readonly int myInt = 5; //直接进行
    readonly int myInt2;

    public MyClass()
    {
        myInt2 = 8;         //间接进行
    }
    public Func()
    {
        myInt = 7; //非法
        Console.WriteLine(myInt2.ToString());
    }
}
```

sealed

带有 sealed 修饰符的类不允许你从它继承任何类。所以如果你不想一个类被继承，你可以对该类使用 sealed 关键字。

C# code
?

```
1sealed class CanNotbeTheParent
2{
3    int a = 5;
4}
```

unsafe

你可以使用 unsafe 修饰符在 C# 中定义一个不安全上下文。在不安全上下文中，你可以插入不安全代码，如 C++ 的指针等。参见以下代码：

C# code
?

```
1public unsafe MyFunction( int * pInt, double* pDouble)
2{
3    int* pAnotherInt = new int;
4    *pAnotherInt  = 10;
5    pInt = pAnotherInt;
6    ...
7    *pDouble = 8.9;
8}
```

接口

如果你有 COM 的思想，你马上就知道我在说什么了。接口是只包含函数签名而在子类中实现的抽象基类。在 C# 中，你可以用 interface 关 键字声明这样的接口类。.NET 就是基于这样的接口的。C# 中你不能对类进行多重继承——这在 C++ 中是允许的。通过接口，多重继承的精髓得以实 现。即你的子类可以实现多重接口。（译注：由此可以实现多重继承）

C# code
?

```
1  using System;
2  interface myDrawing
3  {
4      int originx
5      {
6          get;
7          set;
8      }
9      int originy
10     {
11         get;
12         set;
13     }
14     void Draw(object shape);
15 }
16
17 class Shape: myDrawing
18 {
19     int OriX;
20     int OriY;
21
22     public int originx
23     {
24         get{
25             return OriX;
26         }
```

```
26          set{
27              OriX = value;
28          }
29      }
30      public int originy
31      {
32          get{
33              return OriY;
34          }
35          set{
36              OriY = value;
37          }
38      }
39      public void Draw(object shape)
40      {
41          //... // 做要做的事
42      }
43
44      // 类自身的方法
45      public void MoveShape(int newX, int newY)
46      {
47      //.....
48      }
49  }
50
```

数组

数组在 C# 中比 C++ 中要高级很多。数组分配于堆中，所以是引用类型的。你不能访问数组边界外的元素。所以 C# 防止你引发那种 bug。同时也提供了迭代数组元素的帮助函数。foreach 是这样的迭代语句之一。C++ 和 C# 数组的语法差异在于：

方括号在类型后面而不是在变量名后面
创建元素使用 new 运算符
C# 支持一维、多维和交错数组（数组的数组）

例子：

C# code

?

```
1 int[] array = new int[10]; // int 型一维数组
2 for (int i = 0; i < array.Length; i++)
3     array[i] = i;
4
5 int[,] array2 = new int[5,10]; // int 型二维数组
6 array2[1,2] = 5;
7
8 int[,,] array3 = new int[5,10,5]; // int 型三维数组
9 array3[0,2,4] = 9;
10
11 int[][] arrayOfarray = new int[2]; // int 型交错数组 - 数组的数组
12 arrayOfarray[0] = new int[4];
13 arrayOfarray[0] = new int[] {1,2,15};
```

索引器

索引器用于书写一个可以通过使用 [] 像数组一样直接访问集合元素的方法。你所需要的只是指定待访问实例或元素的索引。索引器的语法和类属性语法相同，除了接受作为元素索引的输入参数外。

例子：

注意：CollectionBase 是用于建立集合的库类。List 是 CollectionBase 中用于存放集合列表的受保护成员。

C# code
?

```
1 class Shapes: CollectionBase
2 {
3     public void add(Shape shp)
4     {
5         List.Add(shp);
6     }
7
      //indexer
8     public Shape this[int index]
9     {
10        get {
11            return (Shape) List[index];
12        }
13        set {
14            List[index] = value ;
15         }
```

```
16      }
17}
18
```

装箱/拆箱

装箱的思想在 C# 中是创新的。正如前面提到的，所有的数据类型，无论是内建的还是用户定义的，都是从 System 命名空间的基类 object 继承的。所以基础的或是原始的类型打包为一个对象称为装箱，相反的处理称为拆箱。

例子：

C# code
?

```
1class Test
2{
3   static void Main()
4   {
5      int myInt = 12;
6      object obj = myInt ;       // 装箱
7      int myInt2 = (int) obj;    // 拆箱
8   }
9}
```

例程展示了装箱和拆箱两个过程。一个 int 值可以被转换为对象，并且能够再次转换回 int。当某种值类型的变量需要被转换为一个引用类型时，便会产生一个对象箱保存该值。拆箱则完全相反。当某个对象箱被转换回其原值类型时，该值从箱中拷贝至适当的存储空间。

函数参数

C# 中的参数有三种类型：

按值传递/输入参数
按引用传递/输入-输出参数
输出参数

如果你有 COM 接口的思想，而且还是参数类型的，你会很容易理解 C# 的参数类型。

按值传递/输入参数
值参数的概念和 C++ 中一样。传递的值复制到了新的地方并传递给函数。

例子：

C# code
?

```
1SetDay([/color]5);
2//...
3void SetDay(int day)
4{
5    //....
6}
```

Java 快速入门

C++和 Java 语言之间的不同可以追溯到它们各自的传统，它们有着不同的设计目标。

- **C++** 被設計成主要用在系統性應用程式設計上，對 C 語言進行了擴展。對於 C 語言這個為執行效率設計的程序式程式設計語言, C++ 特別加上了以下這些特性的支持：靜態類型的面向对象程序设计的支持、異常處理、RAII 以及泛型。另外它還加上了一個包含泛型容器和算法的 C++函式庫。
- **Java** 最開始是被設計用來支持網絡計算。它依賴一個虛擬機在來保證安全和可移植性。Java 包含一個可擴展的庫用以提供一個完整的的下層平台的抽象。Java 是一种静态面向对象语言，它使用的語法類似 C++，但与之不兼容。为了使更多的人使用更易用的語言，它进行了全新的设计。

不同的开发目标导致 C++ 和 Java 这两种语言的不同的规则以及设计上的平衡点不同。 如下列出不同点：

C++	Java
除了一些比較少見的情況之外和 C 語言相容	沒有對任何之前的語言向前相容。但在語法上受 C/C++ 的影響很大
一次编写多处编译	一次编写多处运行
允許程序式程式設計和面向对象程序设计	鼓勵(或者說必須)面向对象的程式設計方式
允许直接调用原生的系统库	要通过 JNI 调用, 或者 JNA
能直接使用底层系统接口	在一个保护模式下的虚拟机中运行
只提供物件的類型和類型名	是反射的, 允許元程式設計和運行時的動態生成代碼
有多種二進制相容標準 (例如:微軟和 Itanium/GNU)	一種二進制相容標準，允許運行時庫的正確性檢查
可选的自动边界检查. (例如: `vector` 和 `string` 这两个容器的 `at()` 方法)	一般都有做边界检查。HotSpot (java)(Sun 的虚拟机实现) 可以去掉边界检查
支持原生的无符号数学运算	不支持原生的无符号数学运算
对所有的数字类型有标准的范围限制，但字节长度是跟实现相关的。标准化的类型可以用 typdef 定义 (uint8_t, ..., uintptr_t)	在所有平台上对所有的基本类型都有标准的范围限制和字节长度
支持指针，引用，传值调用	基本类型总是使用传值调用。物件以可以为空的參考的方式传递（相当于在 C++里使用指向 `class` 或者 `struct` 参数的指標）。[1]
顯式的記憶體管理，但有第三方的框架可以提供垃圾搜集的支持。支持析搆函式。	自動垃圾搜集(可以手動觸發)。沒有析搆函式的概念，對 `finalize()` 的使用是不推荐的
支持类 class，结构 struct，联合 union，可以在堆疊或者栈里為它們動態分配記憶體	只支持類別，只在堆疊中為物件分配記憶體。Java SE 6 在栈為一些物件分配記憶體的使用了逃逸分析的優化方法
允许显式的覆盖(也叫重写)类型	严格的类型安全，除了变宽的类型转换。Java 1.5 开始支持自动类型包装和解包装

	(Autoboxing/Unboxing)
C++函式庫包括：語言支持，診斷工具，常用工具，字符串，本地化，容器，算法，迭代器，數值，輸入/輸出，C 函式庫。Boost 庫提供了更多的功能，包括執行緒和網絡 I/O。使用者必須在一大堆（大部分互相不相容）第三方 GUI 或者其他功能庫中進行選擇	函式庫在每次 Java 發布新版本的時候都會更新并增強功能。1.6 版本支持：本地化，日誌系統，容器和迭代器，算法，GUI 程式設計（但沒有用到系統的 GUI），圖形，多執行緒，網絡，平台安全，自省機制，動態類別加載，阻塞和非阻塞的 I/O，對於 XML、XSLT、MIDI 也提供了相關接口或者支持類別，數據庫，命名服務(例如 LDAP)，密碼學，安全服務(例如 Kerberos)，打印服務，WEB 服務。SWT 提供了一個系統相關的 GUI 的抽象
大部分运算符可以运算符重载	運算子的意義一般來說是不可變的，例外是 + 和 += 運算子被字符串多載了
完全的多重继承，包括虚拟继承	類別只允許單繼承，需要多繼承的情況要使用接口
支持编译期模板	泛型被用来达到和 C++模板类似的效果，但由于类型消除它们不能在编译期间从代码被编译成字节码
支持函式指標，函式物件，lambda（C++11）和接口	沒有函式指標機制。替代的概念是接口，Adapter 和 Listener 也是被廣泛使用的
沒有標準的代碼內嵌文檔機制。不過有第三方的軟體(例如 Doxygen)	Javadoc 标准文档生成系统
`const` 關鍵字用來定義不可改變的常量和成員函式	`final` 提供了一個限制版本的 `const`，等價於 type* `const` 的物件指標或者 `const` 的基本類型數據。沒有 `const` 成員函式，也沒有 `const` type* 指標的等價物
支持 `goto` 语句	支持循環標簽(label)和語句塊
源代碼可以寫成平台無關的（可以被 Windows、BSD、Linux、Mac OS X、Solaris 等編譯，不用修改），也可以寫成利用平台特有的特性。通常被編譯成原生的機器碼	被編譯成 Java 虛擬機的字節碼。和 Java 平台相關，但是源代碼一般來說是不依賴操作系統特有的特性的

C++ 是一門強大的語言，設計用在系統程式設計方面。Java 語言是設計成簡單易用易學習，并有一個強大的跨平台的庫。Java 函式庫對一個函式庫來說相當的大。但 Java 并不會提供所在平台的所有特性和接口。C++函式庫簡單健壯，提供容器和关联数组的支持。[2]

语言特性

语法

- Java 语法是上下文无关文法,可以用一个简单的 LALR 语法分析器来分析.而分析 C++就复杂多了;例如 `Foo<1>(3);` ,如果 Foo 是一个变量,那么它是一个比较的表达式,但如果 Foo 是一个类模板的名字,那么它会创建一个对象.
- C++允许名字空间级别的常量,变量和函数. 而所有这样的 Java 声明必须在一个类或者接口当中.
- 在 C++ 的声明中,一个类名可以用来声明一个此类对象的值. Java 里没办法做到这点. 在 Java 里对象不是值. 在 Java 的声明中,一个类名声明的是对此类的一个对象的 *引用*. 而在 C++ 里与之等价的做法是用 "*" 来声明一个指针.
- 在 C++ 里,"."操作符将一个对象作为一个左操作参数来访问这个对象的成员. 因为对象在 Java 里不是值,所有的对象都通过引用来访问,刚才的做法在 Java 里是无法实现的. 在 Java 里,"." 操作符是将一个对象的引用作为左操作参数来访问这个对象的成员.在 C++中和这种做法等价的是 "->".

C++

```
class Foo {           // 声明 Foo 类
public:
    int x;            // 成員變量

    Foo(): x(0) {     // Foo 的构造函数
Constructor for Foo,
    }                 //  初始化 x

    int bar(int i) {  // 成员函数 bar()
        return 3*i + x;
    }
};

Foo a;
// 声明 a 为一个 Foo 类的对象值,
// 使用其缺省的构造函数
// 如果你想要用其他的构造函数,
// 你可以用 "Foo a(args);"

Foo b = a;
// 拷贝 a 的内容到一个新的 Foo 类的变量 b
当中;
```

Java

```
class Foo {                 // 定义类 Foo
    public int x = 0;       // 成员变量,
                            //  以及其值的
初始化
    public Foo() {          // Foo 的 构造
函数
    }

    public int bar(int i) {// 成员方法
bar()
        return 3*i + x;
    }
}
Foo a;
// 声明 a 为一个 Foo 类的对象的引用
a = new Foo();
// 使用缺省的构造函数初始化
// 如果你想要用其他的构造函数,
// 你可以用 "Foo a = new Foo(args);"
Foo b = a.clone();
// 拷贝所有 a 这个实例的成员到 b,当且仅当,
// Foo 实现了一个 public 的 clone() 方法,
```

```
// 另一种可以选择的语法是 "Foo b(a)"
a.x = 5; // 修改 a 对象
cout << b.x << endl;
// 输出 0,因为 b 和 a 是两个对象

Foo *c;
// 声明 c 为指向一个 Foo 类对象的指针(初始值是
// 未定义的;可能指向任何地方)

c = new Foo();
// 将 c 绑定为一个新的 Foo 对象的引用
Foo *d = c;
// 将 d 绑定为和 c 同一个对象的引用
c->x = 5;
// 修改 c 指向的对象
a.bar(5);  // 对 a 调用 Foo::bar()
c->bar(5); // 对 *c 调用 Foo::bar()
cout << d->x << endl;
// 输出 5,因为 d 引用的对象和 c 一样
```

```
// 并且 clone() 返回一个新的这个对象的拷贝
a.x = 5; // 修改 a 对象
System.out.println(b.x);
// 输出 0,因为 b 和 a 是两个对象

Foo c;
// 声明 c 为一个指向 Foo 对象的指针
// (如果 c 是一个类的成员,那么初始值为空;
// 如果 c 是一个局部变量那么你在使用之前必须
// 对它进行初始化)
c = new Foo();
// 将 c 绑定为一个新的 Foo 对象的引用
Foo d = c;
// 将 d 绑定为和 c 同一个对象的引用
c.x = 5;
// 修改 c 指向的对象
a.bar(5); // 对 a 调用 Foo.bar()
c.bar(5); // 对 c 调用 Foo.bar()
System.out.println(d.x);
// 输出 5,因为 d 引用的对象和 c 一样
```

- 在 C++ 里,声明一个指向常量的指针是可能的, 也就是说, 你不能修改这个指针指向的对象的内容. 函数和方法也都保证不会修改用 "const" 关键字的指针指向的对象的内容,是强制常量正确性的. 在 Java 里这是不可能做到的. 你可以声明一个引用为 "final"(就像在 C++ 里声明一个 "const" 指针), 但这只是阻止你重新绑定这个引用; 你还是可以修改这个 "final" 引用指向的对象的.

C++

```
const Foo *a; // 你不能通过 a 修改 a 指向的对象
a = new Foo();
a->x = 5;
// 非法
Foo *const b = new Foo();
// 你可以声明一个 "const" 指针
b = new Foo();
// 非法, 你不能对它再次绑定
b->x = 5;
// 合法,你还是可以修改这个对象
```

Java

```
final Foo a; // 你可以通过 a 修改 a 指向的对象
a = new Foo(); // 只能在构造函数里
a.x = 5;
// 合法, 你仍然可以修改这个对象
final Foo b = new Foo();
// 你可以声明一个 "final" 引用
b = new Foo();
// 非法, 你不能对它再次绑定
b.x = 5;
// 合法,你还是可以修改这个对象
```

- C++ 支持 goto 语句; Java 强制结构化流程控制(structured control flow), 依赖 break 标签 和 continue 标签 语句来提供类似于 goto 的部分功能. 一些评论者指出这些标签化的流程控制打破了结构化编程的单退出点的特点.[3]
- C++ 提供了一些 Java 缺乏的低级特性. 在 C++ 里, 指针可以用来操作特定的内存位置, 这是在写低级操作系统模块的时候必须用到的. 类似的, 许多 C++ 编译期支持内联汇编， 在 Java

里, 这样的代码只能放在外来的库中,而且在调用的时候只能通过 JNI 来访问这些外来库提供的接口.

语义

- C++ 允许给函数/方法的参数设置缺省值, Java 不提供这个特性. 但是方法重载可以达到同样的效果.
- C++ 里最小的编译单位是一个函数; Java 里最小的编译单位是一个类. 在 C++ 里, 函数可以被单独编译. 在 Java 里, 要编译和维护单独的方法需要把它们移到超类或子类或者使用其他的代码重构的技巧.
- C++ 允许基本类型之间的一些隐式的转换, 也允许程序员对于用户自定义类型相关的隐式转换规则. 在 Java 里, 只有基本类型之间变宽类型的转换可以是隐式的; 其余的转换需要显式的类型转换语法.
    - 这造成的一个后果是,虽然在 Java 和 C++ 里循环的条件(`if`, `while` 和 `for` 里的退出条件)预期的都是一个布尔表达式, 但 `if(a = 5)` 这样的代码在 Java 里会导致编译错误,因为没有从整型到布尔的隐式变窄转换. 如果代码是 `if(a == 5)` 的输错的情况那么是很方便发现这个错误的. 而目前的 C++ 编译器一般来说只会针对这种情况产生一个警告.
- 对于传参数给函数的情况, C++ 支持引用传递和值传递. 在 Java 里, 参数总是值传递的.[4] 但在 Java 里,所有的非基本类型的值都只是对于对象的引用 (用 C++ 的术语来说, 它们是智能指针). 对象在 Java 里不是作为值直接被使用的,只有对象的引用可以被直接操作; 习惯于将对象当做值直接使用的 C++ 开发者经常会把这个跟引用传递搞混.
- Java 内建的类型在字节宽度和取值范围上是被虚拟机定义好的; 在 C++ 里, 内建的类型有定义一个最小取值范围, 但是其他的部分(字节宽度)可以被映射成具体平台上支持的原生类型.
    - 举个例子, Java 字符是 16 位的 Unicode 字符, 字符串是由这样的字符组成的序列. C++ 提供窄和宽两种字符,但实际的字符宽度是和平台相关的, 视所用的字符集而定. 字符串可以由这两种字符中的一种组成.
- 浮点数及其操作的精度和舍入方式在 C++ 里是平台相关的. Java 提供了一个可选的严格的浮点数模型,保证跨平台的一致性,不过可能会导致运行时效率比较差.
- 在 C++ 里, 指针可以作为内存地址直接操作. Java 没有指针 — 它只有对象引用和数组引用, 这两者都不允许直接用来访问内存地址. 在 C++ 里可以构造一个指向指针的指针,而 Java 的引用只能指向对象.
- 在 C++ 里, 指针可以指向函数或者方法（函数指针）. 在 Java 里的等价物是对象或者接口的引用.
- 虽然有使用栈内存分配的对象, C++ 还是支持区域资源管理, 一个用来自动管理内存和其他系统资源的技术,此技术支持确定性对象销毁(deterministic object destruction). 不过,区域资源管理在 C++ 里是不被保证的;它只是一个设计模式,所以需要依赖程序员遵守相关的规则. Java 通过使用垃圾搜集来支持自动内存管理,但对于其他的系统资源(窗口,通讯端口,线程), 如果垃圾搜集器无法决定它们是否不再被用到,那通常还是需要显式的释放的.
- C++ 的用户可自定义操作符重载的特性在 Java 里是不支持的. 唯一在 Java 里可以重载的操作符是 "+" 和 "+=" 操作符, 在字符串里重载为连接字符串.
- Java 的标准应用程序接口支持反射和动态加载任意代码.
- C++ 支持静态和动态的库连接.
- Java 支持泛型, 其主要目的是提供类型安全的容器. C++ 支持模板, 在泛型编程方面提供了更强的支持.

- Java 和 C++ 都对基本类型(也叫"内建"类型)和用户自定义类型(也叫"复合"类型). 在 Java 里, 基本类型只有值的语义,复合类型只有引用的语义. 在 C++ 里所有的值都有值语义,可以创建对于任何类型的引用,这样就允许通过引用语义来操作对象.
- C++ 支持任意类型的多重继承. 在 Java 里一个类只能从单个的类继承而来,但一个类可以实现多个的接口（换句话说,它支持类型的多重继承,但对于实现只能单继承(it supports multiple inheritance of types, but only single inheritance of implementation)）。
- Java 对于类和接口是显式区分的. 在 C++ 里多重继承和纯虚函数使得定义出类似于 Java 的接口的类是可能的,不过会有少许区别.
- Java 在语言和标准库都对多线程有良好的支持. `synchronized` 这个 Java 的关键字为了支持多线程应用提供了简单而安全的互斥锁 ,但同步(synchronized)区只能用 LIFO 的顺序离开. Java 也为更高阶的多线程同步提供了健壮而复杂的库. 在 C++ 里没有专门为多线程定义的内存模型; 但第三方库提供了和 Java 差不多的功能; 不过这些 C++ 库之间差异较大,一致性不好.
- C++ 方法可以声明为虚函数, 虚函数是在运行期根据对象的类型才确定的. C++ 方法缺省情况下不是虚的. 在 Java 里, 方法缺省情况下是虚的, 但可以使用 `final` 关键字使之声明为非虚的.
- C++ 枚举属于基本类型,支持和其他整数类型之间的转换和比较. Java 枚举实际上是类的实例(它们从 `java.lang.Enum<E>` 扩展而来),象其他的类一样可以定义构造函数,数据成员及方法.

资源管理

- Java 提供了自动化的垃圾搜集. 在 C++ 里内存管理通常通过构造函数,析构函数以及智能指针。C++ 标准允许垃圾搜集,但并不强制要求; 实际使用上垃圾搜集极少被用到. 强制使用自动垃圾搜集导致了在 Java 里编写实时软件是困难的.[3]
- C++ 可以申请任意的内存块.Java 只能通过对象实例化来申请内存. (注意:在 Java 里, 程序员可以通过创建一个字节数组模拟申请任意的内存块. 不过 Java 数组仍然是对象.)
- Java 和 C++ 在资源管理上使用不同的习语. Java 主要依赖只能回收内存的垃圾搜集机制,因为该机制如果用于回收使用中的非内存的系统资源可能是非常危险的。而 C++ 主要依赖 RAII (资源的获取就是初始化). 这反映了这两种语言的几方面的不同:
    - 在 C++ 里在栈里申请复合类型的对象是很平常的,一旦退出栈的范围就会被销毁. 在 Java 里复合类型的对象总是在堆里申请的内存,而后被垃圾搜集器搜集 (除非在虚拟机里使用了逃逸分析技术来将堆的内存申请转成栈的.
    - C++ 有析构函数, 而 Java 有 finalizer(en:finalizer). 两者都会在对象释放之前被调用, 但是它们有显著的不同. 一个 C++ 对象的析构函数必须被隐式(栈变量对象的情况)或者显式地调用来释放对象. 析构函数在对象释放之前同步地执行. 同步,协调的反初始化以及释放在 C++ 里满足 RAII 的要求. 在 Java 里, 对象的释放是被垃圾搜集器隐式处理的. 一个 Java 对象的 finalizer 在它被最后一次访问之后和在实际释放之前的某个时间点被异步(en:asynchrony) 地调用, 这个调用有可能一直不产生. 非常少的对象需要 finalizer; 只有那些在释放前必须保证一些清理工作一定要做的对象来说才是需要的 — 典型的情况是：释放对 JVM 来说是外部的资源. 在 Java 里，企图安全同步的释放某些系统资源，只能用显式的 try/finally 结构来进行.
    - 在 C++ 里是有可能有一个迷途指针的 – 过时的对一个已释放的对象的引用(en:reference (computer science)); 试图使用一个迷途指针的结果是导致程序错误. 在 Java 里, 垃圾搜集器不会销毁一个正在被引用的对象.

- 在 C++ 里未初始化过的基本类型对象是有可能存在的, Java 强制要做缺省初始化.
- 在 C++ 里有可能申请了一个对象,但对它没有任何引用. 这样的不可达对象(en:unreachable object)是无法被销毁的,导致了内存泄漏. 作为对比, 在 Java 里一个对象不会被回收直到它变得不可达(对于用户程序来说). (注意: *弱引用*(en:weak reference) 是被支持的, 这个特性让 Java 的垃圾搜集器能够识别不同 程度的可达性.) 在 Java 里垃圾搜集阻止了很多内存泄漏的情况, 但某些情况下泄漏仍然是可能的.[5]
- Java 更容易泄漏非内存资源, 而 C++ 的惯用做法更不会导致这种泄漏.

库

- C++ 对于许多平台相关的特性提供了跨平台的访问方式. 从 Java 到原生的操作系统和硬件相关的函数的直接访问需要用到 JNI(en:Java Native Interface).

运行时

- C++ 通常来说会直接被编译成机器码,被操作系统直接执行. Java 通常会被编译成字节码,被 Java 虚拟机和解释器或者即时编译器编译成机器码然后执行.
- 因为表达方式不受限制,低级的 C++ 语言特性(例如:不被检查的数组访问,原始指针,类型双关语(en:type punning))不能在编译期间或者运行期间可靠地被检查. 相关的编程错误会导致低级的缓存溢出和段错误(en:segmentation fault). 标准模板库 提供了高级的抽象(例如 vector,list 和 map)来帮助避免这样的错误. 在 Java 里, 低级错误不会发生或者会被 JVM 检测到并以异常的形式报告给应用.
- Java 语言在越界访问数组的时候一般来说会对数组进行边界检查(en:bounds checking). 这消除了导致程序不稳定的一个可能因素,但这是以执行速度更慢一些作为代价的. 在一些情况下,编译器分析(en:compiler analysis) 可以检测到不必要的边界检查并去掉. C++ 对于原生数组的越界访问没有要求特定的处理, 所以需要对于原生数组确认不越界. 但 C++ 标准库里的一部分库象 std::vector 也提供了可选的边界检查. 总的来说, Java 数组是"总是安全;严格限制;开销较多" ,而 C++ 原生数组是"可选的开销; 完全不限制; 有潜在的不安全."

模板 vs. 泛型

C++ 和 Java 都提供泛型编程的能力,分别是模板 和 泛型(en:Generics in Java). 虽然它们被创造用来解决类似的问题,有类似的语法,但实际上很不一样.

C++ 模板	Java 泛型
类和函数都可以使用模板.	类和方法都可以使用泛型.
参数可以是任意类型或者整型.	参数只能是能被引用的类型(非基本类型).
在编译的时候对于每种类型生成类或者函数的拷贝.	对于所有类型的参数,只有一个版本的类或者函数生成.

同一个类用不同类型生成的对象在运行期也是不同类型的	编译完成以后类型参数的类型是被消除的; 同一个类用不同类型参数生成的对象在运行期是相同类型的.
想要用到模板类或者函数的实现代码的话必须 include 它(只是声明是不够的).	有一个编译好的类文件里的类或者函数的签名就足以使用泛型了
模板可以被具体化 -- 可以为某个特定的模板参数提供单独的实现.	泛型不能被具体化.
模板参数可以有缺省参数(en:default argument)(只针对对于模板类,模板函数是没有此特性的).	泛型类参数无法拥有缺省参数.
不支持通配符. 返回的类型经常是嵌套的 typedef 形式的.	如果只用一次,那么支持通配符作为类型参数.
不直接支持设置类型参数的边界 (即, 不允许说明类型参数必须为某个类型的子类/父类), 但超编程提供了这个特性[6]	支持类型参数边界, 分别以 "extends" 和 "super" 来定义上界和下界; 同时允许定义类型参数之间的继承关系
允许生成有参模板的类的实例 (如 foo = new Foo<T>, T 为参数)	不允许生成有参模板类的实例 (除非使用反射)
模板类的类型参数可以用在 static 方法和变量上.	泛型类的类型参数无法用在 static 方法和变量上.
static 变量不在在不同的类型参数生成的类之间共享.	static 变量在不同类型参数生成的类的对象之间是共享的.
泛型类和函数在声明时不强制类参数的类限制. 使用错误的类参数会导致模板代码"不工作". 值得注意的是, 如果使用了错误的参数, 则错误信息将出现在定义模板的代码处 (而非调用模板的代码处), 说明 "不支持以该类型作为参数来实例化模板". 这种错误信息往往难以帮助人们找出真正的问题所在 (编程时究竟使用了何种 "错误的" 参数). 因此, 模板类或者函数的正确使用更依赖于正确的文档. 超编程以额外的代价提供了这些特性.	泛型类和函数在声明的时候强制了类参数的类限制(Generic classes and functions can enforce type relationships for type parameters in their declaration). 使用一个错误的参数会在使用它的时候导致一个类错误. 在泛型代码里操作和参数化类型只能按声明的时候保证安全的方式来使用. 这用失去弹性的代价来换取好得多的类型方面的安全性.

模板是图灵完全的 (参见 模板超编程). 泛型不是图灵完全的.

杂项

- Java 和 C++ 在使代码在不同的文件分开方面使用了不同的技术. Java 使用了一个包系统,这个系统对所有的程序都要指定了文件名和路径. 在 Java 里, 编译器负责导入可执行的类文件. C++ 使用了头文件源代码的包含系统来在不同的文件共享声明.
- 编译好的 Java 代码一般来说比 C++ 文件小,因为 Java 字节码(en:Java bytecode)一般来说比机器码要更紧凑[来源請求],Java 程序都不是静态链接的.
- C++ 编译多了一个文本预处理过程, Java 是没有的. 因此一些使用者在他们的编译过程之前增加了一个预处理的过程,这样能更好的支持需要条件编译的情况.
- 两个语言里数组都是定长的. 在 Java 里, 数组是一等对象(en:first-class object), 而在 C++ 里它们只是它们的基本类型元素的连续的序列, 经常用一个指向第一个元素的指针和一个可选的长度来引用. 在 Java 里, 数组是被边界检查的,而且知道它们的长度, 而在 C++ 里你可以将任意的序列当成一个数组. C++ 和 Java 都提供了相关的容器类(分别为 **std::vector** 和 **java.util.ArrayList**),可以改变大小.
- Java 的除法和模除操作符是定义成零截断的. C++ 没有定义这两个操作符是零截断的还是"负无穷截断"的. 在 Java 里-3/2 总是得到 -1, 但一个 C++ 编译器可能会返回 -1 或 -2, 视平台而定. C99 定义了和 Java 一样的除法方式. 两种语言都保证对于所有的 a 和 b(b!=0)(当 a 和 b 都是整型的时候)`(a/b)*b + (a%b) == a`. C++ 版本有时候会更快,因为它允许直接使用处理器的截断方式.
- 整型的长度在 Java 里是已定义好的(int 为 32-bit, long 为 64-bit), 而在 C++ 里整型和指针的长度是和编译器以及应用二进制接口相 关的. 因此仔细编写的 C++ 代码可以利用 64 位处理器的能力而又可以在 32 位处理器上工作. 但是需要很仔细的用可移植的方式编写. 作为对比, Java 的固定整型大小使得程序员无法做到这样,没办法利用处理器的字长会导致 Java 在 64 位处理器上表现较差.

性能

想运行一个编译好的 Java 程序，计算机上要运行 JVM；而编译好的 C++ 程序不需要额外的应用。比较早期的 Java 版本在性能上比静态编译的语言如 C++ 差得很多，这是因为用 C++ 是直接编译成一些机器指令，而当 Java 编译成字节码以后用 JVM 解释执行的时候又牵涉了不少额外的机器指令。 例如：

Java/C++ 语句	**C++ 生成的代码 (x86)**	**Java 生成的字节码**

vector[i]++;	mov edx,[ebp+4h]
	mov eax,[ebp+1Ch]
	inc dword ptr [edx+eax*4]
		aload_1
		iload_2
		dup2
		iaload
		iconst_1
		iadd

C++ 在大部分的情况下都比 Java 要快，[7] 有几个数值方面的基准测试的研究争辩说 Java 在某些情况下可能会比 C++ 的性能好得多。[8][9][10] 但有人说数值方面的基准测试对于语言的评估是不合适的，因为编译器都可以做相关的优化，甚至可能将被测试的代码彻底删除。[11][12][13] 如果涉及到一个真正现实应用的程序，Java 会因为很多原因导致性能变差：[14][15][16]

- 所有的对象都在堆里被申请。对于使用小对象的函数来说会导致很大的性能损失，因为在栈里申请内存几乎没有性能损失。
- 方法缺省是虚的。这对于小对象来说会因为虚表增加好几倍的内存使用。它也会引起性能损失，因为 JIT 编译器不得不对查虚表的过程做额外的优化。
- 即使使用标准的容器依然会有很多的类型转换，这会引起性能损失，因为需要遍历整个继承树。
- 虚拟机更进一步增加了内存的使用，因此降低了内存的局部性，增加了缓存命中失败率，从而导致整个程序变慢。
- 缺乏低级细节的操作方式使得开发者无法将程序进一步优化，因为编译器不支持。[17]

有人争论说，和 Java 相比 C++也有很多劣势：

- 指针使得优化变得困难，因为它们可能指向任意的数据。当然现在这一点也并非完全正确，因为一些现代的编译器引入了 "严格别名" 的规则 [18] 并且支持 C99 的关键字 *restrict*，从而严格限制了指针的使用，使其只能用于指向已知的变量 [19]
- Java 的垃圾搜集和使用 *malloc*/*new* 来申请内存相比能拥有更好的缓存连贯性，因为它的申请一般来说是顺序的。然而，始终有争论认为二者同样会导致内存的“零碎化”（即多次分配和回收之后内存空间会变得不连续），且并没有哪一个比对方有更明显的缓存优势。
- 运行时编译可能可以更好的优化代码，因为可以利用运行时的额外的信息，例如知道代码是在什么样的处理器上运行。然而当今的情况也并非完全如此，因为目前最先进的 C++ 编译器也会针对不同系统生成不同的目标代码，以期充分利用该系统的计算能力 [20]

此外，有争议的是，花在更复杂的 C++ 代码上的 debug 时间太多，用 Java 开发完全可以把这些时间用来优化 Java 代码。当然对于一个给定的程序来说两种语言能优化到什么程度也是一方面。最后，对于处理器负担很重的情况，例如视频渲染，C++ 能直接访问硬件，在同样一个硬件规格下 C++ 总是会比 Java 的表现好很多。

所有权控制

C++ 不是任何一个公司或者组织的商标，不被任何个人拥有。[21] Java 是 Sun 的商标，现在被甲骨文公司拥有。[22]

C++ 语言由 *ISO/IEC 14882* 定义，是一个 ISO 标准，由 *ISO/IEC JTC1/SC22/WG21* 委员会发布。 Java 语言由 *Java Language Specification*[1]定义，这是一本 Sun 公司（现在是甲骨文）出版的书。

参考文献

1. ^ Java is Pass-By-Value
2. ^ Java and C++ Library
3. ^ [3.0] [3.1] Robert C. Martin. Java vs. C++: A Critical Comparison (PDF). 1997.January.
4. ^ James Gosling, Bill Joy, Guy Steele, and Gilad Bracha, *The Java language specification*, third edition. Addison-Wesley, 2005. ISBN 0-321-24678-0 (see also online edition of the specification).
5. ^ "Java memory leaks -- Catch me if you can" by Satish Chandra Gupta, Rajeev Palanki, IBM DeveloperWorks, 16 Aug 2005
6. ^ Boost type traits library
7. ^ Cherrystone Software Labs. Algorithmic Performance Comparison Between C, C++, Java and C# Programming Languages. 2010-03-29 [2010-08-24].
8. ^ Bruckschlegel, Thomas. Microbenchmarking C++, C# and Java. Dr. Dobbs. 2005-06-17 [2010-10-24].
9. ^ "Performance of Java versus C++" by J.P. Lewis and Ulrich Neuman, USC, Jan. 2003 (updated 2004)
10. ^ "Java will be faster than C++" by Kirk Reinholtz, JPL, Apr 2001
11. ^ http://www.ibm.com/developerworks/java/library/j-jtp02225.html#2.0
12. ^ http://www.irrlicht3d.org/pivot/entry.php?id=446
13. ^ "Java (not really faster) than C++ benchmark illustrates
14. ^ http://www.jelovic.com/articles/why_java_is_slow.htm
15. ^ http://warp.povusers.org/grrr/java.html
16. ^ . 18 http://page.mi.fu-berlin.de/prechelt/Biblio//jccpprtTR.pdf. 缺少或|title=为空 (帮助)
17. ^ Clark, Nathan; Amir Hormati, Sami Yehia, Scott Mahlke. Liquid SIMD: Abstracting SIMD hardware using lightweight dynamic mapping. HPCA'07. 2007: 216–227.
18. ^ http://cellperformance.beyond3d.com/articles/2006/06/understanding-strict-aliasing.html
19. ^ Demystifying the Restrict Keyword
20. ^ Targeting IA-32 Architecture Processors for Run-time Performance Checking
21. ^ Bjarne Stroustrup's FAQ: Do you own C++?
22. ^ ZDNet: Oracle buys Sun; Now owns Java.

他们的对象访问格式也不一样

html5 快速入门

HTML 5 是近十年来 Web 开发标准最巨大的飞跃。和以前的版本不同，HTML 5 并非仅仅用来表示 Web 内容，它的新使命是将 Web 带入一个成熟的应用平台，在 HTML 5 平台上，视频，音频，图象，动画，以及同电脑的交互都被标准化。

什么是 HTML5？

HTML5 将成为 HTML、XHTML 以及 HTML DOM 的新标准。

HTML 的上一个版本诞生于 1999 年。自从那以后，Web 世界已经经历了巨变。

HTML5 仍处于完善之中。然而，大部分现代浏览器已经具备了某些 HTML5 支持。

HTML5 是如何起步的？

HTML5 是 W3C 与 WHATWG 合作的结果。

编者注：W3C 指 World Wide Web Consortium，万维网联盟。

编者注：WHATWG 指 Web Hypertext Application Technology Working Group。

WHATWG 致力于 web 表单和应用程序，而 W3C 专注于 XHTML 2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。

为 HTML5 建立的一些规则：

- 新特性应该基于 HTML、CSS、DOM 以及 JavaScript。
- 减少对外部插件的需求（比如 Flash）
- 更优秀的错误处理
- 更多取代脚本的标记
- HTML5 应该独立于设备
- 开发进程应对公众透明

新特性

HTML5 中的一些有趣的新特性：

- 用于绘画的 canvas 元素
- 用于媒介回放的 video 和 audio 元素
- 对本地离线存储的更好的支持
- 新的特殊内容元素，比如 article、footer、header、nav、section
- 新的表单控件，比如 calendar、date、time、email、url、search

浏览器支持

最新版本的 Safari、Chrome、Firefox 以及 Opera 支持某些 HTML5 特性。Internet Explorer 9 将支持某些 HTML5 特性。

1.HTML5 文件的 MIME 类型：

这个跟以往的版本没有改变，必须以 text/html 的 MIME 的类型传送，后缀名为.html/.htm

2.HTML5 文件的 DOCTYPE 宣告：

<!DOCTYPE HTML>(大小写均可以)

非常简单的宣告，但是必须一定要有。如果省掉的话，使得浏览器没有使用最新的演算绘制模式，可能会照成 html5 功能无法实现。

3.文字的编码的指定方式：

<meta charset="utf-8">或者

<meta http-equiv="Content-Type" content="text/html;charset=utf-8">

这两种方法不可以同时使用。

4.不可以使用结束标签的元素：

area,base,br,col,cmmand,embed,hr,img,input,keygen,link,meta,param,source

eg, 简单 html5 网页

[xhtml] view plaincopy

```
1. <!DOCTYPE HTML>
2. <html>
3. <head>
4. <meta charset="utf-8">
5. </head>
6. <body>
7.  <div>
8.   <p>简单的 html5 网页<p>
9.  </div>
10. </body>
11. </html>
```

javascript 快速入门

html 显示数据，css 负责样式的显示，javascript 负责完成页面的交互

js 使用

1.可以直接在 html 页面中，在 script 标签中写相应的 js 的代码

```
<script type="text/javascript">
     alert("hello world");
</script>
```

2.可以引入外部文件，通过 src 来指定外部文件的位置，特别注意不能省略 script 的结束标记

```
<script type="text/javascript" src="hello.js"></script>
```

hello.js 文件的内容：

```
alert("hello world");
```

变量

变量简介

对于 js 而言，是没有数据类型的(弱类型，没有数据类型)，全部都是通过 var 来完成变量的创建

```
<script type="text/javascript">
     var a = 19;
     alert(a);//19
     a = "hello";//自动完成类型转换
     alert(a);//hello
</script>
```

变量的作用域

```
<script type="text/javascript">
     var b = 12;//全局变量
     function test1(){
           var a = 10;//局部变量
           alert(a);
           c=22;//当在函数内部没有使用 var 来声明变量的时候，这个变量就会作为全局变量
声明
     }
     function test2(){
           alert(b);
           //alert(a);//局部变量无法访问
           alert(c);
```

```
    }
</script>
```

变量的类型

常用的类型有:Number,String,Array,Date

```
var a = 10.6;
// alert(typeof a);//显示变量类型

强制类型转换
a = "11";
//java 进行强制类型转换是(Number)a,而 js 是通过 Number(a)

//如果强制转换一个非数字的值为 Number 会得到一个 NaN 的值【NaN-->Not a Number】
var b = "abc";
//alert(Number(b));

//使用 parseInt 可以将字符串开头的几个数字转换为 int，但是如果开头不是数字，那就得到 NaN
b = "12px";
//alert(parseInt(b));

对于数组等对象而言，显示的结果就是 object 不会显示 Array
var as = ["a","b",1,2,3];
alert(typeof as);

判断 as 是否是 Array 的实例，如果是返回 true
alert(as instanceof Array);

布尔类型:true 和 false,在 js 中，非 0 就是 true，特别注意:NaN 是 false
当一个变量没有定义值的时候，是 undefined 类型，undefined 类型是 false
特别注意：在 js 中除了 NaN,undefined,0 这三个数是 false 外其余皆是 true

在 javascript 中，定义一个 function 就相当于 java 中的一个类，可以定义函数的属性和方法（行为）并且进行相关的操作和调用
```

函数的定义与调用

定义函数直接用 function 关键字，调用的时候，用函数名()调用即可：

```
1 var x = function() {
2         alert("x");
3 }
```

```
4 //此时 x 就是一个 function 函数
5 x();//调用(执行)x 这个函数
6
7 function fn() {
8         alert("fn");
9         //对于函数而言，直接写 return 就等于有返回值
10        return "100";
11}
12//此时是将 y 这个变量指向函数 fn,可以通过 y()来调用函数
13var y = fn;
14fn();//调用函数 fn
15y();//可以这样调用
16
17//将函数 fn 所执行的返回值传给 z 变量，所以 z 为 100
18var z = fn();
19alert(z);
20alert(y);
```

简单对象创建

对应 js 而言没有类的概念，只有对象的概念
对象创建基本认识：使用函数就可以创建对象

直接看代码：

```
1 //可以使用 function 来模拟 java 的类
2 function Person(name,age) {
3         //定义了一个 Person 的属性为 name
4         this.name = name;
5         //定义了 Person 的属性为 age
6         this.age = age;
7         this.address = "默认地址";
8
9         //如果没有用 this 声明，这个变量就仅仅只是一个局部变量，不是类的属性
10        var x = 10;
11
12        //创建了一个方法,匿名方法
13        this.say = function() {
14                alert(this.name+","+this.age);
15        }
16}
17//创建了一个对象 p1 是 Person 的对象
18var p1 = new Person("张三",12);
```

```
19alert(p1.name+","+p1.address+","+p1.x);//张三，默认地址，undefined
20//调用方法
21p1.say();
22
23var p2 = new Person("德华",22);
24p2.address = "香港";
25//可以通过对象["属性字符串"]完成对属性的调用
26alert(p2["name"]+","+p2["address"]);
27
28alert(typeof p1);
29alert(p1 instanceof Person);
30
31//在 js 中对于对象而言，可以通过 for in 来变量对象的属性
32for(var a in p1) {
33        //可以获取对象中的所有显示声明的属性
34        //循环打印出 p1 中的属性和值
35        alert(a+":"+p1[a]);
36}
```

Date 对象

```
1<script type="text/javascript">
2    var d = new Date();
3    //对于 js 而言，月的下标是从 0 开始的
4    document.write(d.getFullYear()+"年"+
5                        (d.getMonth()+1)+"月"+d.getDate()+"日"+
6                        "星期"+d.getDay());
7</script>
```

String 对象

处理字符串信息，常用方法如下所示

```
1 <script type="text/javascript">
2     var str1 = new String("abc");
3     var str2 = "abc";
4     alert(str1==str2);//true
5     var s = str2.concat("hello","world");
6     alert(s);//abchelloworld
7     //包含 start 不包含 end
8     s = s.slice(2,4);
9     alert(s);//ch
      var str = "hello world";
      //从 2 开始到 5 结束
```

```
    alert(str.substring(2,5));//llo
    //从 2 开始取 5 个字符
    alert(str.substr(2,5));//llo w

    str = "abc.txt";
    alert(str.substr(str.lastIndexOf(".")+1));//txt
</script>
```

array 对象

数组对象，没有大小限制，和 java 的 List 类似

```
<script type="text/javascript">
    //js 的 array 就是 java 中的 list 和 stack 的集合
    var as = new Array();
    as.push(11);
    as.push(22);
    alert(as);//11,22

    as = new Array(11,22,33,44,55,66,77,"111","222",23);
    alert(as);//11,22,33,44,55,66,77,111,222,23
    //一般使用以下方式定义数组
    as = [11,12,1,2,3];


    //转换为字符串通过---来完成连接
    alert(as.join("---"));//11---12---1---2---3
    //sort 只会通过字符串来排序
    alert(as.sort());//1,11,12,2,3
    //颠倒顺序
    alert(as.reverse());//3,2,12,11,1

    as = [1,2,3,4];
    //表示在索引为 2 的前面删除 0 个元素，并且增加两个元素 31 和 32-->1,2,31,32,3,4
    //as.splice(2,0,31,32);
    //表示在索引为 2 的前面删除 2 个元素，并且增加两个元素 31 和 32-->1,2,31,32
    as.splice(2,2,31,32);
    alert(as);//1,2,31,32
</script>
```

SQL 数据库快速入门必须掌握的四大基本语句

做一个系统的后台，基本上都少不了增删改查，作为一个新手入门，我们必须要掌握 SQL 四条最基本的数据操作语句：Insert，Select，Update 和 Delete！ 下面对这四个语句进行详细的剖析：

熟练掌握 SQL 是数据库用户的宝贵财富。在本文中，我们将引导你掌握四条最基本的数据操作语句—SQL 的核心功能—来依次介绍比较操作符、选择断言以及三值逻辑。当你完成这些学习后，显然你已经开始算是真正 SQL 入门了。

在我们开始之前，先使用 CREATE TABLE 语句来创建一个表。DDL 语句对数据库对象如表、列和视进行定义。它们并不对表中的行进行处理，这是因为 DDL 语句并不处理数据库中实际的数据。这些工作由另一类 SQL 语句—数据操作语言（DML）语句进行处理。

SQL 中有四种基本的 DML 操作：INSERT，SELECT，UPDATE 和 DELETE。由于这是大多数 SQL 用户经常用到的，我们有必要在此对它 们进行一一说明。在图 1 中我们给出了一个名为 EMPLOYEES 的表。其中的每一行对应一个特定的雇员记录。请熟悉这张表，我们在后面的例子中将要用到 它。

INSERT 语句

用户可以用 INSERT 语句将一行记录插入到指定的一个表中。例如，要将雇员 John Smith 的记录插入到本例的表中，可以使用如下语句：

```
INSERT INTO EMPLOYEES VALUES
 ('Smith','John','1980-06-10',
 'Los Angles',16,45000);
```

通过这样的 INSERT 语句，系统将试着将这些值填入到相应的列中。这些列按照我们创建表时定义的顺序排列。在本例中，第一个值“Smith”将填到第一个列 LAST_NAME 中；第二个值“John”将填到第二列 FIRST_NAME 中……以此类推。

我们说过系统会“试着”将值填入，除了执行规则之外它还要进行类型检查。如果类型不符（如将一个字符串填入到类型为数字的列中），系统将拒绝这一次操作并返回一个错误信息。

如果 SQL 拒绝了你所填入的一列值，语句中其他各列的值也不会填入。这是因为 SQL 提供对事务的支持。一次事务将数据库从一种一致性转移到另一种一致性。如果事务的某一部分失败，则整个事务都会失败，系统将会被恢复（或称之为回退）到此事务之前的状态。

回到原来的 INSERT 的例子，请注意所有的整形十进制数都不需要用单引号引起来，而字符串和日期类型的值都要用单引号来区别。为了增加可读性而在数字间插入逗号将会引起错误。记住，在 SQL 中逗号是元素的分隔符。

同样要注意输入文字值时要使用单引号。双引号用来封装限界标识符。

对于日期类型，我们必须使用 SQL 标准日期格式（yyyy-mm-dd），但是在系统中可以进行定义，以接受其他的格式。当然，2000 年临近，请你最好还是使用四位来表示年份。

既然你已经理解了 INSERT 语句是怎样工作的了，让我们转到 EMPLOYEES 表中的其他部分：

```
INSERT INTO EMPLOYEES VALUES
 ('Bunyan','Paul','1970-07-04',
 'Boston',12,70000);
INSERT INTO EMPLOYEES VALUES
 ('John','Adams','1992-01-21',
 'Boston',20,100000);
INSERT INTO EMPLOYEES VALUES
 ('Smith','Pocahontas','1976-04-06',
```

```
 'Los Angles',12,100000);
INSERT INTO EMPLOYEES VALUES
 ('Smith','Bessie','1940-05-02',
 'Boston',5,200000);
INSERT INTO EMPLOYEES VALUES
 ('Jones','Davy','1970-10-10',
 'Boston',8,45000);
INSERT INTO EMPLOYEES VALUES
 ('Jones','Indiana','1992-02-01',
 'Chicago',NULL,NULL);
```

在最后一项中，我们不知道 Jones 先生的工薪级别和年薪，所以我们输入 NULL（不要引号）。NULL 是 SQL 中的一种特殊情况，我们以后将进行详细的讨论。现在我们只需认为 NULL 表示一种未知的值。

有时，像我们刚才所讨论的情况，我们可能希望对某一些而不是全部的列进行赋值。除了对要省略的列输入 NULL 外，还可以采用另外一种 INSERT 语句，如下：

```
INSERT INTO EMPLOYEES(
 FIRST_NAME, LAST_NAME,
 HIRE_DATE, BRANCH_OFFICE)
VALUE(
 'Indiana','Jones',
 '1992-02-01','Indianapolis');
```

这样，我们先在表名之后列出一系列列名。未列出的列中将自动填入缺省值，如果没有设置缺省值则填入 NULL。请注意我们改变了列的顺序，而值的顺序要对 应新的列的顺序。如果该语句中省略了 FIRST_NAME 和 LAST_NAME 项（这两项规定不能为空），SQL 操作将失败。

让我们来看一看上述 INSERT 语句的语法图：

```
INSERT INTO table
 [(column { ,column})]
VALUES
 (columnvalue [{,columnvalue}]);
```

和前一篇文章中一样，我们用方括号来表示可选项，大括号表示可以重复任意次数的项（不能在实际的 SQL 语句中使用这些特殊字符）。VALUE 子句和可选的列名列表中必须使用圆括号。

SELECT 语句

SELECT 语句可以从一个或多个表中选取特定的行和列。因为查询和检索数据是数据库管理中最重要的功能，所以 SELECT 语句在 SQL 中是工作量最大的部分。实际上，仅仅是访问数据库来分析数据并生成报表的人可以对其他 SQL 语句一窍不通。

SELECT 语句的结果通常是生成另外一个表。在执行过程中系统根据用户的标准从数据库中选出匹配的行和列，并将结果放到临时的表中。在直接 SQL（direct SQL）中，它将结果显示在终端的显示屏上，或者将结果送到打印机或文件中。也可以结合其他 SQL 语句来将结果放到一个已知名称的表中。

SELECT 语句功能强大。虽然表面上看来它只用来完成本文第一部分中提到的关系代数运算“选择”（或称“限制”），但实际上它也可以完成其他两种关系运算—“投影”和“连接”，SELECT 语句还可以完成聚合计算并对数据进行排序。

SELECT 语句最简单的语法如下：

```
SELECT columns FROM tables;
```

当我们以这种形式执行一条 SELECT 语句时，系统返回由所选择的列以及用户选择的表中所有指定的行组成的一个结果表。这就是实现关系投影运算的一个形式。

让我们看一下使用图 1 中 EMPLOYEES 表的一些例子（这个表是我们以后所有 SELECT 语句实例都要使用的。而我们在图 2 和图 3 中给出了查询的实际结果。我们将在其他的例子中使用这些结果）。

假设你想查看雇员工作部门的列表。那下面就是你所需要编写的 SQL 查询：

```
SELECT BRANCH_OFFICE FROM EMPLOYEES;
```

以上 SELECT 语句的执行将产生如图 2 中表 2 所示的结果。

由于我们在 SELECT 语句中只指定了一个列，所以我们的结果表中也只有一个列。注意结果表中具有重复的行，这是因为有多个雇员在同一部门工作（记住 SQL 从所选的所有行中将值返回）。要消除结果中的重复行，只要在 SELECT 语句中加上 DISTINCT 子句：

```
SELECT DISTINCT BRANCH_OFFICE
FROM EMPLOYEES;
```

这次查询的结果如表 3 所示。

现在已经消除了重复的行，但结果并不是按照顺序排列的。如果你希望以字母表顺序将结果列出又该怎么做呢？只要使用 ORDER BY 子句就可以按照升序或降序来排列结果：

```
SELECT DISTINCT BRANCH_OFFICE
FROM EMPLOYEES
ORDER BY BRANCH_OFFICE ASC;
```

这一查询的结果如表 4 所示。请注意在 ORDER BY 之后是如何放置列名 BRANCH _OFFICE 的，这就是我们想要对其进行排序的列。为什么即使是结果表中只有一个列时我们也必须指出列名呢？这是因为我们还能够按照表中其他列进行排 序，即使它们并不显示出来。列名 BRANCH_ OFFICE 之后的关键字 ASC 表示按照升序排列。如果你希望以降序排列，那么可以用关键字 DESC。

同样我们应该指出 ORDER BY 子句只将临时表中的结果进行排序；并不影响原来的表。

假设我们希望得到按部门排序并从工资最高的雇员到工资最低的雇员排列的列表。除了工资括号中的内容，我们还希望看到按照聘用时间从最近聘用的雇员开始列出的列表。以下是你将要用到的语句：

```
SELECT BRANCH_OFFICE,FIRST_NAME,
 LAST_NAME,SALARY,HIRE_DATE
FROM EMPLOYEES
ORDER BY SALARY DESC,
 HIRE_DATE DESC;
```

这里我们进行了多列的选择和排序。排序的优先级由语句中的列名顺序所决定。SQL 将先对列出的第一个列进行排序。如果在第一个列中出现了重复的行时，这 些行将被按照第二列进行排序，如果在第二列中又出现了重复的行时，这些行又将被按照第三列进行排序……如此类推。这次查询的结果如表 5 所示。

将一个很长的表中的所有列名写出来是一件相当麻烦的事，所以 SQL 允许在选择表中所有的列时使用*号：

```
SELECT * FROM EMPLOYEES;
```

这次查询返回整个 EMPLOYEES 表，如表 1 所示。

下面我们对开始时给出的 SELECT 语句的语法进行一下更新（竖直线表示一个可选项，允许在其中选择一项。）：

```
SELECT [DISTINCT]
 (column [{, columns}])| *
FROM table [ {, table}]
```

[ORDER BY column [ASC] | DESC
[{, column [ASC] | DESC }]];

定义选择标准

在我们目前所介绍的 SELECT 语句中，我们对结果表中的列作出了选择但返回的是表中所有的行。让我们看一下如何对 SELECT 语句进行限制使得它只返回希望得到的行：

SELECT columns FROM tables [WHERE predicates];

WHERE 子句对条件进行了设置，只有满足条件的行才被包括到结果表中。这些条件由断言（predicate）进行指定（断言指出了关于某件事情的一种 可能的事实）。如果该断言对于某个给定的行成立，该行将被包括到结果表中，否则该行被忽略。在 SQL 语句中断言通常通过比较来表示。例如，假如你需要查询 所有姓为 Jones 的职员，则可以使用以下 SELECT 语句：

SELECT * FROM EMPLOYEES
WHERE LAST_NAME = 'Jones';

LAST_NAME = 'Jones'部分就是断言。在执行该语句时，SQL 将每一行的 LAST_NAME 列与“Jones”进行比较。如果某一职员的姓为“Jones”，即断言成立，该职员的信息将被包括到结果表中（见表 6）。

使用最多的六种比较

我们上例中的断言包括一种基于“等值”的比较（LAST_NAME = 'Jones'），但是 SQL 断言还可以包含其他几种类型的比较。其中最常用的为：

等于 =
不等于 <>
小于 <
大于 >
小于或等于 <=
大于或等于 >=

下面给出了不是基于等值比较的一个例子：

SELECT * FROM EMPLOYEES
WHERE SALARY > 50000;

这一查询将返回年薪高于$50,000.00 的职员（参见表 7）。

逻辑连接符

有时我们需要定义一条不止一种断言的 SELECT 语句。举例来说，如果你仅仅想查看 Davy Jones 的信息的话，表 6 中的结果将是不正确的。为了进一步定义一个 WHERE 子句，用户可以使用逻辑连接符 AND，OR 和 NOT。为了只得到职员 Davy Jones 的记录，用户可以输入如下语句：

SELECT * FROM EMPLOYEES
WHERE LAST_NAME = 'Jones' AND FIRST_NAME = 'Davy';

在本例中，我们通过逻辑连接符 AND 将两个断言连接起来。只有两个断言都满足时整个表达式才会满足。如果用户需要定义一个 SELECT 语句来使得当其中任何一项成立就满足条件时，可以使用 OR 连接符：

SELECT * FROM EMPLOYEES
WHERE LAST_NAME = 'Jones' OR LAST_NAME = 'Smith';

有时定义一个断言的最好方法是通过相反的描述来说明。如果你想要查看除了 Boston 办事处的职员以外的其他所有职员的信息时，你可以进行如下的查询：

SELECT * FROM EMPLOYEES
WHERE NOT(BRANCH_OFFICE = 'Boston');

关键字 NOT 后面跟着用圆括号括起来的比较表达式。其结果是对结果取否定。如果某一职员所在部门的办事处在 Boston，括号内的表达式返回 true，但是 NOT 操作符将该值取反，所以该行将不被选中。

断言可以与其他的断言嵌套使用。为了保证它们以正确的顺序进行求值，可以用括号将它们括起来：

```
SELECT * FROM EMPLOYEES
WHERE (LAST_NAME = 'Jones'
AND FIRST_NAME = 'Indiana')
OR (LAST_NAME = 'Smith'
AND FIRST_NAME = 'Bessie');
```

SQL 沿用数学上标准的表达式求值的约定—圆括号内的表达式将最先进行求值，其他表达式将从左到右进行求值。

以上对逻辑连接符进行了说明，在对下面的内容进行说明之前，我们再一次对 SELECT 语句的语法进行更新：

```
SELECT [DISTINCT]
 (column [{, column } ] )| *
FROM table [ { , table} ]
[ORDER BY column [ASC] | [DESC
[{ , column [ASC] | [DESC } ] ]
WHERE predicate [ { logical-connector predicate } ];
```

NULL 和三值逻辑

在 SQL 中 NULL 是一个复杂的话题，关于 NULL 的详细描述更适合于在 SQL 的高级教程而不是现在的入门教程中进行介绍。但由于 NULL 需要进行特殊处理，并且你也很可能会遇到它，所以我们还是简略地进行一下说明。

首先，在断言中进行 NULL 判断时需要特殊的语法。例如，如果用户需要显示所有年薪未知的职员的全部信息，用户可以使用如下 SELECT 语句：

```
SELECT * FROM EMPLOYEES
WHERE SALARY IS NULL;
```

相反，如果用户需要所有已知年薪数据的职员的信息，你可以使用以下语句：

```
SELECT * FROM EMPLOYEES
WHERE SALARY IS NOT NULL;
```

请注意我们在列名之后使用了关键字 IS NULL 或 IS NOT NULL，而不是标准的比较形式：COLUMN = NULL、COLUMN <> NULL 或是逻辑操作符 NOT（NULL）。

这种形式相当简单。但当你不明确地测试 NULL（而它们确实存在）时，事情会变得很混乱。

例如，回过头来看我们图 1 中的 EM-PLOYEES 表，可以看到 Indiana Jones 的工薪等级或年薪值都是未知的。这两个列都包含 NULL。可以想象运行如下的查询：

```
SELECT * FROM EMPLOYEES
WHERE GRADE <= SALARY;
```

此时，Indiana Jones 应该出现在结果表中。因为 NULL 都是相等的，所以可以想象它们是能够通过 GRADE 小于等于 SALARY 的检查的。这其实是一个毫无疑义的查 询，但是并没有关系。SQL 允许进行这样的比较，只要两个列都是数字类型的。然而，Indiana Jones 并没有出现在查询的结果中，为什么？

正如我们早先提到过的，NULL 表示未知的值（而不是象某些人所想象的那样表示一个为 NULL 的值）。对于 SQL 来说意味着这个值是未知的，而只要这个 值为未知，就不能将其与其他

值比较（即使其他值也是 NULL）。所以 SQL 允许除了在 true 和 false 之外还有第三种类型的真值，称之为“非确定”（unknown）值。

如果比较的两边都是 NULL，整个断言就被认 为是非确定的。将一个非确定断言取反或使用 AND 或 OR 与其他断言进行合并之后，其结果仍是非确定的。由于结果表中只包括断言值为“真”的行，所以 NULL 不可能满足该检查。从而需要使用特殊的操作符 IS NULL 和 IS NOT NULL。

UPDATE 语句

UPDATE 语句允许用户在已知的表中对现有的行进行修改。

例如，我们刚刚发现 Indiana Jones 的等级为 16，工资为$40,000.00，我们可以通过下面的 SQL 语句对数据库进行更新（并清除那些烦人的 NULL）。

```
UPDATE EMPLOYEES
SET GRADE = 16, SALARY = 40000
WHERE FIRST_NAME = 'Indiana'
 AND LAST_NAME = 'Jones';
```

上面的例子说明了一个单行更新，但是 UPDATE 语句可以对多行进行操作。满足 WHERE 条件的所有行都将被更新。如果，你想让 Boston 办事处中的所有职员搬到 New York，你可以使用如下语句：

```
UPDATE EMPLOYEES
SET BRANCH_OFFICE = 'New York'
WHERE BRANCH_OFFICE = 'Boston';
```

如果忽略 WHERE 子句,表中所有行中的部门值都将被更新为'New York'。

UPDATE 语句的语法流图如下面所示：

```
UPDATE table
SET column = value [{, column = value}]
[ WHERE predicate [ { logical-connector predicate}]];
```

DELETE 语句

DELETE 语句用来删除已知表中的行。如同 UPDATE 语句中一样，所有满足 WHERE 子句中条件的行都将被删除。由于 SQL 中没有 UNDO 语句或是 “你确认删除吗？”之类的警告，在执行这条语句时千万要小心。如果决定取消 Los Angeles 办事处并解雇办事处的所有职员，这一卑鄙的工作可以由以下这条语句来实现：

```
DELETE FROM EMPLOYEES
WHERE BRANCH_OFFICE = 'Los Angeles';
```

如同 UPDATE 语句中一样，省略 WHERE 子句将使得操作施加到表中所有的行。

DELETE 语句的语法流图如下面所示：

```
DELETE FROM table
[WHERE predicate [ { logical-connector predicate} ] ];
```

现在我们完成了数据操作语言（DML）的主要语句的介绍。我们并没有对 SQL 能完成的所有功能进行说明。SQL 还提供了许多的功能，如求平均值、求和以 及其他对表中数据的计算，此外 SQL 还能完成从多个表中进行查询（多表查询，或称之为连接）的工作。这种语言还允许你使用 GRANT 和 REVOKE 命令控 制使用者的数据访问权限

Required Software

1. Java™ 1.5.x
2. ssh 与 sshd

如果没有安装请自行安装。我以 CentOS 4.6 为例。

下载 hadoop，http://apache.mirror.phpchina.com/hadoop/core/ 我下载的是 0.17.1 版本。

解压 hadoop-0.17.1.tar.gz，然后 conf/hadoop-env.sh 设置 JAVA_HOME ，我是可 JAVA_HOME 去注释，值自己的路径。如：

```
export JAVA_HOME=/usr/java/jdk1.6.0_06
```

如果不设置启动后用不了。

先从简单开始。

1、Local (Standalone) Mode ，叫单机模式。

```
[chenlb@master hadoop-0.17.1]$ bin/hadoop jar hadoop-0.17.1-examples.jar grep conf output 'dfs[a-z.]+'
```

```
[chenlb@master hadoop-0.17.1]$ cat output/*
```

如果，正常可以看到内容。像这样。

```
3    dfs.
3    dfs.class
2    dfs.period
2    dfs.replication
...  ...
```

2、Pseudo-Distributed Mode，虚拟分布模式。
vi conf/hadoop-site.xml

```
<configuration>
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000/</value>
```

```
</property>
<property>
    <name>mapred.job.tracker</name>
    <value>hdfs://master:9001/</value>
</property>
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/home/chenlb/hadoop-0.17.1/tmp/</value>
</property>
</configuration>
```

在/etc/hosts 里添加本机 ip 对应 master，例如我的：172.16.249.210　master
保证可以无密码登录。请看那一篇文章：
http://www.blogjava.net/chenlb/archive/2008/07/03/212293.html

用 ssh localhost 试一下是否免密码登录。

格式化分布式文件系统：

```
[chenlb@master hadoop-0.17.1]$ bin/hadoop namenode -format
```

启动 Hadoop：

```
[chenlb@master hadoop-0.17.1]$ bin/start-all.sh
```

默认可以在${HADOOP_HOME}/logs 里看到日志。

可以用 web 看浏览 NameNode 和 JobTracker

- NameNode - http://localhost:50070/
- JobTracker - http://localhost:50030/

把文件放到分布式文件系统里：

```
[chenlb@master hadoop-0.17.1]$ bin/hadoop dfs -put conf input
```

此时已经在分布文件系统里建立了 input 文件夹。而 conf 是本地的文件夹。

执行示例：

```
[chenlb@master hadoop-0.17.1]$ bin/hadoop jar hadoop-*-examples.jar grep input output 'dfs[a-z.]+'
```

这里 input 和 output 都是分布式文件系统的的文件夹，而且 output 在分布式文件系统里不存在，否则报错（也可以先删除它 bin/badoop dfs -rmr output）。

耐心等待。结束后可以查看。

```
[chenlb@master hadoop-0.17.1]$ bin/hadoop dfs -get output output
[chenlb@master hadoop-0.17.1]$ cat output/*
```

也可以直接在分布式文件系统里查：

```
[chenlb@master hadoop-0.17.1]$ bin/hadoop dfs -cat output/*
```

成功运行后可以关闭它了：

```
[chenlb@master hadoop-0.17.1]$ bin/stop-all.sh
```

Script 快速入门与查表

在测试程序当中，“让一部分事情自动去做”往往是绝对优先的选择，通常最新的菜鸟也要选择脚本语言，学着“让它自动去做”来作为第一项基本功，本篇不属于脚本高手的参考，而是旨在指导菜鸟如何用三天时间学会快速编程。

本篇文章包括脚本语言用到的较全的概念，而且这也是菜鸟写出脚本的最基本条件，累赘少许我们开始吧。

一. 正则表达

处理杂乱无章的字符文件数据，如何获得有效信息，就要使用特定的过滤办法，正则表达的概念正用于此，它集成一系列特殊符号附加特定的 pattern 办法来实现强大的过滤功能，这里很值得玩味。

1. 特殊符号表：

^ 锚定行的开始 如：'^grep'匹配所有以 grep 开头的行。

$ 锚定行的结束 如：'grep$'匹配所有以 grep 结尾的行。

. 匹配一个非换行符的字符 如：'gr.p'匹配 gr 后接一个任意字符，然后是 p。

* 匹配零个或多个先前字符 如：'*grep'匹配所有一个或多个空格后紧跟 grep 的行。 .*一起用代表任意字符。

[] 匹配一个指定范围内的字符，如'[Gg]rep'匹配 Grep 和 grep。

[^] 匹配一个不在指定范围内的字符，如：'[^A-FH-Z]rep'匹配不包含 A-R 和 T-Z 的一个字母开头，紧跟 rep 的行。

* 重复零次或更多次

+ 重复一次或更多次

? 重复零次或一次

{n} 重复 n 次

{n,} 重复 n 次或更多次

{n,m} 重复 n 到 m 次

\w 匹配文字和数字字符，也就是[A-Za-z0-9]，如：'G\w*p'匹配以 G 后跟零个或多个文字或数字字符，然后是 p。

\d 任意一个数字，0~9 中的任意一个

\s 包括空格、制表符、换页符等空白字符的其中任意一个

\b 匹配一个单词边界，也就是单词和空格之间的位置，不匹配任何字符

\W 匹配任意不是字母和数字的字符

\S 匹配任意不是空白符的字符

\D 匹配任意非数字的字符

\B 匹配不是单词开头或结束的位置

| 左右两边表达式之间 "或" 关系，匹配左边或者右边

( ) 在被修饰匹配次数的时候，括号中的表达式可以作为整体被修饰；取匹配结果的时候，括号中的表达式匹配到的内容可以被单独得到

反向引用 \1, \2, ... 通常结合()来使用的，整体代表参数,属于贪婪法则。

2. 注意事项：

ab^bc 和 ab$bc 当中的^和$都失去了作开头和结尾公用的目的，相当于直接使用^和$来使用，等同于加转义字符 ab\^bc 和 ab\$bc

[^]单独这样使用没有意义。

[.*]里面的点和美元符号并不表示任何字符匹配任何次数，而是直接指.和*,如果其中加入[\.*]来对他做转义反而会报错，这是因为[]表示选择当中的一个字符作为可能过滤。

3. 实例和工具：

网络上的实例到处都是，直接看搜一篇就搞定了，关键是要记住。推荐一个工具 Regex Match Tracer.

网址 http://www.regexlab.com/zh/regref.htm

二. grep 锦囊

grep 是 linux 脚本语言里头最常用的一个命令，先掌握它的基本用法吧，我们可以 man grep，但只要理解和活用当中的几项就可以大显身手了。

1. man grep
-c 只输出匹配行的个数。
-i 不区分大小写（只适用于单字符）。
-h 查询多文件时不显示文件名。
-l 查询多文件时只输出包含匹配字符的文件名。
-n 显示匹配行及行号。
-s 不显示不存在或无匹配文本的错误信息。
-v 显示不包含匹配文本的所有行。
-V 显示软件版本信息
2. grep 正则
grep 结合正则表达才叫真，注意要用引号给它括起来，格式如下：
grep -option "pattern" source-file(s)
grep 的正则表达有点不同于上面的阐述，其实也是由于 BRE 和 ERE 的正则标准。grep 一般使用 BRE(Baisc Regular Express)标准.ERE 是 Extend.
几点不同列表如下，即多加了几道反斜杠来解释 meta 字符：
\(..\) 标记匹配字符，如'\(love\)'，love 被标记为 1。
\> 锚定单词的结束，如'grep\>'匹配包含以 grep 结尾的单词的行。
x\{m\} 重复字符 x，m 次，如：'0\{5\}'匹配包含 5 个 o 的行。
x\{m,\} 重复字符 x,至少 m 次，如：'o\{5,\}'匹配至少有 5 个 o 的行。
x\{m,n\}重复字符 x，至少 m 次，不多于 n 次，如：'o\{5,10\}'匹配 5--10 个 o 的行。

三. sed 匕首

sed 的功能比 grep 更加强大，它主要体现在对文件的搜索，替换，搬移，删除功能，面向的目标是文件的行，执行的时候是一行一行的读取文件再进行过滤处理，据此我从这四个部分进行讲解。
这四种状态包含 sed 的各种书写格式，这是容易混淆的地方，但是确实指示清晰理的地方。下文中 pattern 表示要使用的正则表达，address[X] 表示文本的位置，replacement 表示要去取代的源串，action 表示要进行的动作，file(s)表示即将处理里的目标文件。
1. sed 的命令参数
-f 表示用文件的形式编写 sed 命令，sed -f sedscript 即可。
-e 表示编辑文本，后面根据 pattern 或者 replacement 或者 action 处理文件。
-n 表示 silent 模式，如果要在终端打印出信息，必须明确制定 p 参数
2. sed 的搜索功能
1) 格式：sed -e "/pattern/" files
2) 这种搜索功能同 grep 大同小异，无非它在前面添加了一个斜杠尾部添加了一个斜杠表示分界。
3. sed 的替换功能
1) 格式： sed -e "s/pattern/replacement/[action]" files

replacement 是要替换经 pattern 正则表达后的文本。

2) 当 action 为 g 时,代表替换所有符合(match)的字串。

当 action 为十进位数字 m 时,代表替换行内第 m 个符合的字串。

当 action 为 p 时,代表替换第一个符合 pattern 的字串後,将资料输出标准输出档。

当 action 为 w wfile 时,代表替换第一个符合 pattern 的字串後,输出到 wfile 档内(如果 wfile 不存在,则会重新开启名为 wfile 的档案)。

当没有 action 时,则将资料行内第一个符合 pattern 的字串以 replacement 字串来替换。

4. sed 的删除功能

格式： sed -e "address1,addres2[action]" files

sed -e "/pattern/d" files

sed -e "/pattern1/,/pattern2/[action]" files

一三项正是都好起作用，表示从哪里行到哪里行，起匹配的哪里到哪里。

5. sed 的搬迁功能

这里要理解两个概念：pattern space 和 hold space.

pattern space 表示 sed 读取一行到内存缓存中，经 sed 的正则处理之后，再把结果送出到终端输出。

hold space 表示在 patternspace 里面的内容需要再进行一次处理，这时候就要把结果(类似中间状态)放到一个空间再做下一步处理，这个中间状态的空间就是 holdspace,例如两个处理结果分别得到，但两者结构又需要用对方的数据，这时候就要把一项放到 holdspace 来进行保存呢

1) 格式： sed -e "address1, address2 [action]"

sed -e "/pattern/[action] file" files

action 为下列列表

2) 列表：

!不执行函数参数。

=印出资料行数(line number)。

a 添加使用者输入的资料。

b label 将执行的指令跳至由:建立的参考位置。

c 以使用者输入的资料取代资料。

d 删除资料。

D 删除 pattern space 内第一个 newline 字母前的资料。

g 拷贝资料从 hold space。

G 添加资料从 hold space 至 pattern space。

h 拷贝资料从 pattern space 至 hold space。

H 添加资料从 pattern space 至 hold space。

l 印出 l 资料中的 nonprinting character 用 ASCII 码。

i 插入添加使用者输入的资料行。

n 读入下一笔资料。

N 添加下一笔资料到 pattern space。

p 印出资料。

P 印出 pattern space 内第一个 newline 字母前的资料。

q 跳出 sed 编辑。
r 读入它档内容。
s 替换字串。
t label 先执行一替换的编辑指令,如果替换成牛 p>则将编辑指令跳至: label 处执行。
w 写资料到它档内。
x 交换 hold space 与 pattern space 内容。
y 转换(transform)字元。

6. sed 的注意事项。
1) sed -e "s/test/& me/g" file 表示用 test me 代替 test, &表示的是 pattern 字符串（不是特殊符号）
2) 回想反向引用\1 \2. sed 的参数规则是如下的：sed -e "/(test) (me) (again)/[2 3 1]/g" ,它用的是方括号添上数字表示。这句话表示用 me again test 代替 test me again.
3) 待工作中补充。

四. awk 大刀
对字符过滤的又一大补充，正是 awk 这个砍刀，它不像 grep 仅仅面向查找行和 sed 面向编辑行，它是续 grep 和 sed 处理之后对字段进行处理的过滤工具。
awk 当中融入正则表达，awk 编程语法(类似 C 加少数 shell，变量定义不需要初始值)，内置函数处理与内置变量。
调用 awk:
第一种,命令行方式,如:awk [-F field-separator] ‘commands’ input-file(s)这里 commands 是真正的 awk 命令,[-F 域分隔符]是可选的,awk 默认使用空格分隔,因此如果要浏览域间有空格的文 本,不必指定这个选项,但如 果浏览如 passwd 文件,此文件各域使用冒号作为分隔符,则必须使用-F 选项: awk -F : ‘commands’ input-file；第二种,将所有 awk 命令插入一个文件,并使 awk 程序可执行,然后用 awk 命令解释器作为脚本的首行,以便通过键入脚本名称来调 用它第三种,将所有 awk 命令插入一个单独文件,然后调用: awk -f awk-script-file input-file -f 选项指明在文件 awk-script-file 的 awk 脚本,input-file 是使用 awk 进行浏览的文件名
awk 脚本:
awk 脚本由各种操作和模式组成,根据分隔符(-F 选项),默认为空格,读取的内容依次放置到对应的域中,一行一行记录读取,直到文件尾模式和动作:任 何 awk 语句都是由模式和动作组成,在一个 awk 脚本中可能有许多语句。模式部分决定动作语句何时触发及触发事件。动作即对数据进行的操作,如果省去模式 部分,动作将时刻保持执行状态模式可以是任何条件语句或复合语句或正则表达式,模式包含两个特殊字段 BEGIN 和 END,使用 BEGIN 语句设置计数和打 印头,BEGIN 语句使用在任何文本浏览动作之前,之后文本浏览动作依据输入文件开始执行;END 语句用来在 awk 完成文本浏览动作后打印输出文本总数和 结尾状态标志,有动作必须使用{}括起来，实际动作在大括号{}内指明,常用来做打印动作,但是还有更长的代码如 if 和循环 looping 语句及循环退出 等,如果不指明采取什么动作,awk 默认打印出所有浏览出的记录
域和记录:

awk 执行时,其浏览标记为$1,$2…$n,这种方 法称为域标记.使用$1,$3 表示参照第 1 和第 3 域,注意这里使用逗号分隔域,使用$0 表示使用所有域例:awk ‘{print $0}’ temp.txt > sav.txt 表示打印所有域并把结果重定向到 sav.txt 中 awk ‘{print $0}’ temp.txt|tee sav.txt 和上例相似,不同的是将在屏幕上显示出来
awk ‘{print $1,$4}’ temp.txt 只打印出第 1 和第 4 域 awk ‘BEGIN {print “NAME GRADE\n————-”} {print $1″\t”$4}’ temp.txt 表示打信息头,即输入的内容的第一行前加上”NAME GRADE\n————-”,同时内容以 tab 分开
awk ‘BEGIN {print “being”} {print $1} END {print “end”}’ temp 同时打印信息头和信息尾
条件操作符:
<、<=、==、!=、>=、~匹配正则表达式、!~不匹配正则表达式
匹配:awk ‘{if ($4~/ASIMA/) print $0}’ temp 表示如果第四个域包含 ASIMA,就打印整条
awk ‘$0 ~ /ASIMA/’ temp 表示只要整条包含 ASIMA 就打印出来
精确匹配:awk ‘$3==”48″ {print $0}’ temp 只打印第 3 域等于”48″的记录
不匹配: awk ‘$0 !~ /ASIMA/’ temp 打印整条不包含 ASIMA 的记录
不等于: awk ‘$1 != “asima”‘ temp
小于: awk ‘{if ($1<$2) print $1 “is smaller”}’ temp
设置大小写: awk ‘/[Gg]reen/’ temp 打印整条包含 Green,或者 green 的记录
任意字符: awk ‘$1 ~/^…a/’ temp 打印第 1 域中第四个字符是 a 的记录,^行首,.任意字符或关系匹配: awk ‘$0~/(abc)|(efg)/’ temp 使用|时,语句需要括起来 AND 与关系: awk ‘{if ($1==”a” && $2==”b”) print $0}’ temp
OR 或关系: awk ‘{if ($1==”a” || $1==”b”) print $0}’ temp
awk 内置变量:
ARGC 命令行参数个数 AGRV 命令行参数排列 ENVIRON 支持队列中系统环境变量的使用 FILENAME awk 浏览的文件名 FNR 浏览文件的记录数 FS 设置输入域分隔符,同- F 选项 NF 浏览记录的域个数 NR 已读的记录数 OFS 输出域分隔符
ORS 输出记录分隔符 RS 控制记录分隔符例: awk ‘END {print NR}’ temp 在最后打印已读记录条数 awk ‘{print NF,NR,$0} END {print FILENAME}’ temp

awk ‘{if (NR>0 && $4~/Brown/) print $0}’ temp 至少存在一条记录且包含 Brown

NF 的另一用法: echo $PWD | awk -F/ ‘{print $NF}’ 显示当前目录名
awk 操作符:
在 awk 中使用操作符,基本表达式可以划分成数字型、字符串型、变量型、域及数组元素
设置输入域到变量名:
awk ‘{name=$1;six=$3; if (six==”man”) print name ” is ” six}’ temp
域值比较操作:
awk ‘BEGIN {BASE=”27″} {if ($4<BASE) print $0}’ temp
修改数值域取值 :(原输入文件不会被改变)
awk ‘{if ($1==”asima”) $6=$6-1;print $1,$6,$7}’ temp
修改文本域:

awk ‘{if ($1==”asima) ($1==”desc”);print $1}’ temp
只显示修改记录 :(只显示所需要的,区别上一条命令,注意{})
awk ‘{if ($1==”asima) {$1==”desc”;print$1}}’ temp
创建新的输出域:
awk ‘{$4=$3-$2; print $4}’ temp
统计列值:
awk ‘(tot+=$3);END {print tot}’ temp 会显示每列的内容
awk ‘{(tot+=$3)};END {print tot}’ temp 只显示最后的结果
文件长度相加:
ls -l|awk ‘/^[^d]/ {print $9″\t”$5} {tot+=$5} END{print “totKB:” tot}’
只列出文件名:
ls -l|awk ‘{print $9}’ 常规情况文件名是第 9 域
awk 内置字符串函数:
gsub(r,s) 在整个$0 中用 s 替代 r
awk ‘gsub(/name/,”xingming”) {print $0}’ temp
gsub(r,s,t) 在整个 t 中用 s 替代 r
index(s,t) 返回 s 中字符串 t 的第一位置
awk ‘BEGIN {print index(“Sunny”,”ny”)}’ temp 返回 4
length(s) 返回 s 的长度
match(s,r) 测试 s 是否包含匹配 r 的字符串
awk ‘$1==”J.Lulu” {print match($1,”u”)}’ temp 返回 4
split(s,a,fs) 在 fs 上将 s 分成序列 a
awk ‘BEGIN {print split(“12#345#6789″,myarray,”#”)”‘
返回 3,同时 myarray[1]=”12″, myarray[2]=”345″, myarray[3]=”6789″
sprint(fmt,exp) 返回经 fmt 格式化后的 exp
sub(r,s) 从$0 中最左边最长的子串中用 s 代替 r(只更换第一遇到的匹配字符串)
substr(s,p) 返回字符串 s 中从 p 开始的后缀部分
substr(s,p,n) 返回字符串 s 中从 p 开始长度为 n 的后缀部分
printf 函数的使用:
字符转换: echo “65″ |awk ‘{printf “%c\n”,$0}’ 输出 A
awk ‘BEGIN {printf “%f\n”,999}’ 输出 999.000000
格式化输出:awk ‘{printf “%-15s %s\n”,$1,$3}’ temp 将第一个域全部左对齐显示
其他 awk 用法:
向一行 awk 命令传值:
awk ‘{if ($5<AGE) print $0}’ AGE=10 temp
who | awk ‘{if ($1==user) print $1 ” are in ” $2 ‘ user=$LOGNAME 使用环境变量
awk 数组:
awk 的循环基本结构 For (element in array) print array[element]
awk ‘BEGIN {record=”123#456#789″;split(record,myarray,”#”)}
END { for (i in myarray) {print myarray[i]} }

五. Bash 语法

所有脚本语言的语法都是简单容易的，稍稍用点功夫就可以完全掌握的但也是非常容易忘记的，我建议从以下几点记住并活用便可以很好的用在开发当中，现在逐一列出并讲述。

1. 符号对变量的定义

1) 基本定义：

${var}用{}来定义一个变量，若变量不长，可直接$var，若$var 的 var 后面跟了字符，则一定要用{}括起来表示是对之前已定义的变量取值。

若 var 是数字，则表示是脚本输入的参数，如果数字大于 10，必须要用{}括起来。

dollor$后面如果跟{}表示取变量，如果是跟()表示取()里面语句(注意是语句)执行结果的值为变量。

2) 定义预先处理：

${var:-word}表示 var 若没有被定义 NULL，则使用 word 初始化左值，注意左值不是 var 而是另一个变量

例： sh=${var:-licheng} 表示若 var 空$sh=licheng

${var:+word}表示 va 若没有定义或者为 NULL,则返回 NULL，否则返回 word,这个处理主要是做 var 是否曾被定义

${var:=word}类同于${var:-word}

3) 定义变量截断：

优先考虑的 linux 命令：dirname and basename

其次${var##.*} 或者${var#.*},这里的.和*不是正则表达式里面的概念，而是 DOS 系统匹配的概念，.没有什么意义表示点号，*表示匹配任何个字符，当然可以想 象?表示匹配任何一个字符。这里的"##"，"#"表示截断离前面(左边)最远的匹配子字符串和最近的匹配子字符串，保留后面(右边)的字符串做变量引 用。由此可知，截断的概念也类似于从字符串中取字符串做新变量定义声明。

最后有类似的${var%%.*}或者${var%.*}表示截断离后面(右边)最远的匹配子字符串和最近的匹配子字符串，保前面(左边)的字符串做变量引用

4) 参数的符号表示：

$# 表示所有参数的个数，常用 shift 命令来进行遍历。

$? 表示脚本语句运行时候的状态，测试运行的结果正确还是错误。

$* 表示脚本所有参数，主要要关注与"$@"的区别

$@ 表示脚本所有参数，主要要关注与"$*"的区别

"$*" 表示所有的参数当做一个字符串来进行处理，如两个参数$1=var1 $2=var2,则"$*"为"var1 var2"

"$@" 任然表示所有单独的个体参数，常给 for 遍历用。如两个参数$1=var1 $2=var2，则"$@"为"var1" "var2"

2. 基本语法

1) test 和 [] 测试条件命令。常用-x,-f,-z 等，可以查阅网上书籍。

2) if []; then
elif []; then
fi

3) case var in

p1)
… ;;
p2)
… ;;
*)
… ;;
esac 注意 case 没有像 C 语言当中的 default 项。

4) while [cond1] && { || } [cond2] …; do
…
done

5) for var in …; do
…
done
for ((cond1; cond2; cond3)) do
…
done

6) until [cond1] && { || } [cond2] …; do
…
done

3. 数组应用(待补充)

4. 函数应用

1) ./path 引入，类似于 C 的 include 包含定义和函数声明。

2) 函数参数由$1,$2,$3 使用，最好不要过多，若多分成更小模块编写。

5. shell 编程常用到的 linux 处理命令和注意事项(不断更新中)：

1) eval:eval 应用来源：[root@localhost] a="id | cut -f 4 -d ' ' " $a 出错，执行 eval $a 正确，也就是说，直接对命令赋值成变量来执行会导致被赋值的命令在参数解析上出错.

2) local:loacl 表示定义局部变量，由于函数经常定义在最前头，在用的时候有些变量会重用，在函数中使用 local 会防止修改函数前后同名的变量值，从而有效地在函数内部进行计算而不会影响前后变量。

3) declare and let: declare 可以声明一个即将可以做何种类型计算的变量声明，常用在数学计算，也用在数组定义，它有几个参数是应该熟悉的：

declare –a *name* ：表示数组 array。

declare –f *name* ：表示是 function 的名字。

declare –F *name* ：同上，但只显示 function 的名字。这个和上面的具体差异不太明白，但是这两者都 很少使用，先不理会它们。

declare –i *name* ：表示整数

declare –r *name* ：表示只读。不能使用 unset。对于只读变量，也可以使用 **readonly** *name* 的方式，相当于 declare –r *name* 。readonly 可以带三个选项：-f 表示这是个 function 的名字，-p 表示打印所有的 readonly 的名字，-a 表示这是个只读的数组。

declare –x *name* ：同 export，即不仅在当前的环境中起作用，也在外部的 shell 环境中起作用。

let 表示让一个变量进行数学运算。

实例

```
[root@localhost ~]#  a=2;echo $a; let a=a-1; echo $a
 [root@localhost ~]# declare -i a=2;echo $a; a=a-1; echo $a
```

6.shell 字符串的截取的问题：

6.1、Linux shell 截取字符变量的前 8 位，有方法如下：

```
1.expr substr “$a” 1 8
2.echo $a|awk ‘{print substr(,1,8)}’
3.echo $a|cut -c1-8
4.echo $
5.expr $a : ‘\(.\\).*’
6.echo $a|dd bs=1 count=8 2>/dev/null
```

6.2、按指定的字符串截取

1、第一种方法:

${varible##*string} 从左向右截取最后一个 string 后的字符串

${varible#*string}从左向右截取第一个 string 后的字符串

${varible%%string*}从右向左截取最后一个 string 后的字符串

${varible%string*}从右向左截取第一个 string 后的字符串

“*”只是一个通配符可以不要

例子：

```
$ MYVAR=foodforthought.jpg
$ echo ${MYVAR##*fo}
rthought.jpg
$ echo ${MYVAR#*fo}
odforthought.jpg
```

2、第二种方法：${varible:n1:n2}:截取变量 varible 从 n1 到 n2 之间的字符串。

可以根据特定字符偏移和长度，使用另一种形式的变量扩展，来选择特定子字符串。试着在 bash 中输入以下行：
$ EXCLAIM=cowabunga
$ echo ${EXCLAIM:0:3}
cow
$ echo ${EXCLAIM:3:7}
abunga

这种形式的字符串截断非常简便，只需用冒号分开来指定起始字符和子字符串长度。

6.3、按照指定要求分割：
比如获取后缀名
ls -al | cut -d “.” -f2

7. 各种炸碎补充：

1) $((_SLR&0x0))实现的操作时样_SLR 变量能够进行与操作，修改_SLR 的原始值。

Design Pattern 和系统设计快速入门

1、factory（工厂）追 MM 少不了请吃饭了，麦当劳的鸡翅和肯德基的鸡翅都是 MM 爱吃的东西，虽然口味有所不同，但不管你带 MM 去麦当劳或肯德 基，只管向服务员说“来四个鸡翅”就行了。麦当劳和肯德基就是生产鸡翅的 Factory 工厂模式：客户类和工厂类分开。消费者任何时候需要某种产品，只需向工厂请求即可。消费者无须修改就可以接纳新产品。缺点是当产品修改时，工厂类也要做相 应的修改。如：如何创建及如何向客户端提供。

2、builder（建造）?MM 最爱听的就是“我爱你”这句话了，见到不同地方的 MM,要能够用她们的方言跟她说这句话哦，我有一个多种语言翻译 机， 上面每种语言都有一个按键，见到 MM 我只要按对应的键，它就能够用相应的语言说出“我爱你”这句话了，国外的 MM 也可以轻松搞掂，这就是我的“我爱你” builder。（这一定比美军在伊拉克用的翻译机好卖） 建造模式：将产品的内部表象和产品的生成过程分割开来，从而使一个建造过程生成具有不同的内部表象的产品对象。建造模式使得产品内部表象可以独立的变化， 客户不必知道产品内部组成的细节。建造模式可以强制实行一种分步骤进行的建造过程。

3、factory method（工厂方法）请 MM 去麦当劳吃汉堡，不同的 MM 有不同的口味，要每个都记住是一件烦人的事情，我一般采用 Factory Method 模式，带着 MM 到服务员

那儿，说“要一个汉堡”，具体要什么样的汉堡呢，让 MM 直接跟服务员说就行了。工厂方法模式：核心工厂类不再负责所有 产品的创建，而是将具体创建的工作交给子类去做，成为一个抽象工厂角色，仅负责给出具体工厂类必须实现的接口， 而不接触哪一个产品类应当被实例化这种细节。

4、prototype（原始）跟 MM 用 QQ 聊天，一定要说些深情的话语了，我搜集了好多肉麻的情话，需要时只要 copy 出来放到 QQ 里面就行了， 这 就是我的情话 prototype 了。（100 块钱一份，你要不要） 原始模型模式：通过给出一个原型对象来指明所要创建的对象的类型，然后用复制这个原型对象的方法创建出更多同类型的对象。原始模型模式允许动态的增加或减 少产品类，产品类不需要非得有任何事先确定的等级结构，原始模型模式适用于任何的等级结构。缺点是每一个类都必须配备一个克隆方法。

5、singleton（单例）?俺有 6 个漂亮的老婆，她们的老公都是我，我就是我们家里的老公 Sigleton，她们只要说道“老公”，都是指的 同一个人，那就是我(刚才做了个梦啦，哪有这么好的事) 单例模式：单例模式确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例单例模式。单例模式只应在有真正的“单一实例”的需求时才可使 用。[b:9ceca65206]结构型模式[/b:9ceca65206]

6、adapter（适配器）?在朋友聚会上碰到了一个美女 Sarah，从香港来的，可我不会说粤语，她不会说普通话，只好求助于我的朋友 kent 了，他作为我和 Sarah 之间的 Adapter，让我和 Sarah 可以相互交谈了(也不知道他会不会耍我) 适配器（变压器）模式：把一个类的接口变换成客户端所期待的另一种接口，从而使原本因接口原因不匹配而无法一起工作的两个类能够一起工作。适配类可以根据 参数返还一个合适的实例给客户端。

7、bridge（桥梁）?早上碰到 MM，要说早上好，晚上碰到 MM，要说晚上好；碰到 MM 穿了件新衣服，要说你的衣服好漂亮哦，碰到 MM 新做的 发 型，要说你的头发好漂亮哦。不要问我“早上碰到 MM 新做了个发型怎么说”这种问题，自己用 BRIDGE 组合一下不就行了桥梁模式：将抽象化与实现化脱耦， 使得二者可以独立的变化，也就是说将他们之间的强关联变成弱关联，也就是指在一个软件系统的抽象化和实现化之间使用 组合/聚合关系而不是继承关系，从而使两者可以独立的变化。

8、composite（合成）?Mary 今天过生日。“我过生日，你要送我一件礼物。”“嗯，好吧，去商店，你自己挑。”“这件 T 恤挺漂亮， 买，这 条裙子好看，买，这个包也不错， 买。”“喂，买了三件了呀，我只答应送一件礼物的哦。”“什么呀，T 恤加裙子加包包，正

好配成一套呀，小姐，麻烦你包起 来。”“……”，MM 都会用 Composite 模式了，你会了没有？ 合成模式：合成模式将对象组织到树结构中，可以用来描述整体与部分的关系。合成模式就是一个处理对象的树结构的模式。合成模式把部分与整体的关系用树结构 表示出来。合成模式使得客户端把一个个单独的成分对象和由他们复合而成的合成对象同等看待。

9、decorator（装饰）?Mary 过完轮到 Sarly 过生日，还是不要叫她自己挑了，不然这个月伙食费肯定玩完，拿出我去年在华山顶上照 的照 片，在背面写上“最好的的礼物，就是爱你的 Fita”，再到街上礼品店买了个像框（卖礼品的 MM 也很漂亮哦），再找隔壁搞美术设计的 Mike 设计了一个漂 亮的盒子装起来……，我们都是 Decorator，最终都在修饰我这个人呀，怎么样，看懂了吗？装饰模式：装饰模式以对客户端透明的方式扩展对象的功能， 是继承关系的一个替代方案，提供比继承更多的灵活性。动态给一个对象增加功能，这些功能可以 再动态的撤消。增加由一些基本功能的排列组合而产生的非常大量的功能。

10、facade（门面）?我有一个专业的 Nikon 相机，我就喜欢自己手动调光圈、快门，这样照出来的照片才专业，但 MM 可不懂这些，教了半 天也不会。幸好相机有 Facade 设计模式，把相机调整到自动档，只要对准目标按快门就行了，一切由相机自动调整，这样 MM 也可以用这个相机给我拍张照片 了。 门面模式：外部与一个子系统的通信必须通过一个统一的门面对象进行。门面模式提供一个高层次的接口，使得子系统更易于使用。每一个子系统只有一个门面类， 而且此门面类只有一个实例，也就是说它是一个单例模式。但整个系统可以有多个门面类。

11、fly weight（享元）?每天跟 MM 发短信，手指都累死了，最近买了个新手机，可以把一些常用的句子存在手机里，要用的时候，直接拿出来，在前面加上 MM 的 名字就可以发送了，再不用一个字一个字敲了。共享的句子就是 Flyweight，MM 的名字就是提取出来的外部特征，根据上下文情况使用。 享元模式：FLYWEIGHT 在拳击比赛中指最轻量级。享元模式以共享的方式高效的支持大量的细粒度对象。享元模式能做到共享的关键是区分内蕴状态和外蕴 状态。内蕴状态存储在享元内部，不会随环境的改变而有所不同。外蕴状态是随环境的改变而改变的。外蕴状态不能影响内蕴状态，它们是相互独立的。将可以 共享的状态和不可以共享的状态从常规类中区分开来，将不可以共享的状态从类里剔除出去。客户端不可以直接创建被共享的对象，而应当使用一个工厂对象负责创 建被共享的对象。享元模式大幅度的降低内存中对象的数量。

12、proxy（代理）?跟 MM 在网上聊天，一开头总是“hi,你好”,“你从哪儿来呀？”“你多大了？”“身高多少呀？”这些话，真烦人，写 个程 序做为我的 Proxy 吧，凡是接收到这些话都设置好了自动的回答，接收到其他的话时再通知我回答，怎么样，酷吧。 代理模式：代理模式给某一个对象提供一个代理对象，并由代理对象控制对源对象的引用。代理就是一个人或一个机构代表另一个人或者一个机构采取行动。某些情 况下，客户不想或者不能够直接引用一个对象，代理对象可以在客户和目标对象直接起到中介的作用。客户端分辨不出代理主题对象与真实主题对象。代理模式 可以并不知道真正的被代理对象，而仅仅持有一个被代理对象的接口，这时候代理对象不能够创建被代理对象，被代理对象必须有系统的其他角色代为创建并传入。 [b:9ceca65206]行为模式[/b:9ceca65206]

13、Chain Of Responsibleity(责任链)?晚上去上英语课，为了好开溜坐到了最后一排，哇，前面坐了好几个漂亮的 MM 哎，找张纸条，写上“Hi,可以做我 的女朋友 吗？如果不愿意请向前传”，纸条就一个接一个的传上去了，糟糕，传到第一排的 MM 把纸条传给老师了，听说是个老处女呀，快跑! 责任链模式：在责任链模式中，很多对象由每一个对象对其下家的引用而接 起来形成一条链。请求在这个链上传递，直到链上的某一个对象决定处理此请求。客户并不知道链上的哪一个对象最终处理这个请求，系统可以在不影响客户端的情 况下动态的重新组织链和分配责任。处理者有两个选择：承担责任或者把责任推给下家。一个请求可以最终不被任何接收端对象所接受。

14、command(命令)?俺有一个 MM 家里管得特别严，没法见面，只好借助于她弟弟在我们俩之间传送信息，她对我有什么指示，就写一张纸条 让她 弟弟带给我。这不，她弟弟又传送过来一个 COMMAND，为了感谢他，我请他吃了碗杂酱面，哪知道他说：“我同时给我姐姐三个男朋友送 COMMAND，就 数你最小气，才请我吃面。”， 命令模式：命令模式把一个请求或者操作封装到一个对象中。命令模式把发出命令的责任和执行命令的责任分割开，委派给不同的对象。命令模式允许请求的一方和 发送的一方独立开来，使得请求的一方不必知道接收请求的一方的接口，更不必知道请求是怎么被接收，以及操作是否执行，何时被执行以及是怎么被执行的。 系统支持命令的撤消。

15、interpreter(解释器)?俺有一个《泡 MM 真经》，上面有各种泡 MM 的攻略，比如说去吃西餐的步骤、去看电影的方法等等，跟 MM 约 会时， 只要做一个 Interpreter，照着上面的脚本执行就可以了。 解释器模式：给定一个语言后，解释器模式可以定义出其文法的一种表示，并同时提供一个解释器。客户端可以使用这个解释器来解释这个语言中的句子。解释器模 式将描述怎样在有了一个简单的文法后，使用模式设计解释这些语句。在解释器模式里面提到的语言是指任何解释器对象能够解释的任何组合。在解释器模式中需要定义一个代表文法的命令类的等级结构，也就是一系列的组合规则。每一个命令对象

都有一个解释方法，代表对命令对象的解释。命令对象的等级结构中的对象 的任何排列组合都是一个语言。

16、iterator(迭代子)?我爱上了 Mary，不顾一切的向她求婚。 Mary：“想要我跟你结婚，得答应我的条件” 我：“什么条件我都答应，你说吧” Mary：“我看上了那个一克拉的钻石” 我：“我买，我买，还有吗？” Mary：“我看上了湖边的那栋别墅” 我：“我买，我买，还有吗？” Mary：“我看上那辆法拉利跑车” 我脑袋嗡的一声，坐在椅子上，一咬牙：“我买，我买，还有吗？” …… 迭代子模式：迭代子模式可以顺序访问一个聚集中的元素而不必暴露聚集的内部表象。多个对象聚在一起形成的总体称之为聚集，聚集对象是能够包容一组对象的容 器对象。迭代子模式将迭代逻辑封装到一个独立的子对象中，从而与聚集本身隔开。迭代子模式简化了聚集的界面。每一个聚集对象都可以有一个或一个以上的 迭代子对象，每一个迭代子的迭代状态可以是彼此独立的。迭代算法可以独立于聚集角色变化。

17、mediator(调停者)?四个 MM 打麻将，相互之间谁应该给谁多少钱算不清楚了，幸亏当时我在旁边，按照各自的筹码数算钱，赚了钱的从 我这里 拿，赔了钱的也付给我，一切就 OK 啦，俺得到了四个 MM 的电话。 调停者模式：调停者模式包装了一系列对象相互作用的方式，使得这些对象不必相互明显作用。从而使他们可以松散偶合。当某些对象之间的作用发生改变时，不会 立即影响其他的一些对象之间的作用。保证这些作用可以彼此独立的变化。调停者模式将多对多的相互作用转化为一对多的相互作用。调停者模式将对象的行为 和协作抽象化，把对象在小尺度的行为上与其他对象的相互作用分开处理。

18、memento(备忘录)?同时跟几个 MM 聊天时，一定要记清楚刚才跟 MM 说了些什么话，不然 MM 发现了会不高兴的哦，幸亏我有个备忘录， 刚才与 哪个 MM 说了什么话我都拷贝一份放到备忘录里面保存，这样可以随时察看以前的记录啦。备忘录模式：备忘录对象是一个用来存储另外一个对象内部状态的快照的 对象。备忘录模式的用意是在不破坏封装的条件下，将一个对象的状态捉住，并外部 化，存储起来，从而可以在将来合适的时候把这个对象还原到存储起来的状态。

19、observer(观察者)?想知道咱们公司最新 MM 情报吗？加入公司的 MM 情报邮件组就行了，tom 负责搜集情报，他发现的新情报不用一 个一个 通知我们，直接发布给邮件组，我们作为订阅者（观察者）就可以及时收到情报啦 观察者模式：观察者模式定义了一种一队多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态上发生变化时，会通知所有观察者对象， 使他们能够自动更新自己。

20、state(状态)?跟 MM 交往时，一定要注意她的状态哦，在不同的状态时她的行为会有不同，比如你约她今天晚上去看电影，对你没兴趣的 MM 就 会说“有事情啦”，对你不讨厌但还没喜欢上的 MM 就会说“好啊，不过可以带上我同事么？”，已经喜欢上你的 MM 就会说“几点钟？看完电影再去泡吧怎么 样？”，当然你看电影过程中表现良好的话，也可以把 MM 的状态从不讨厌不喜欢变成喜欢哦。状态模式：状态模式允许一个对象在其内部状态改变的时候改变行 为。这个对象看上去象是改变了它的类一样。状态模式把所研究的对象的行为包装在不同的状 态对象里，每一个状态对象都属于一个抽象状态类的一个子类。状态模式的意图是让一个对象在其内部状态改变的时候，其行为也随之改变。状态模式需要对每一个 系统可能取得的状态创立一个状态类的子类。当系统的状态变化时，系统便改变所选的子类。

21、strategy(策略)?跟不同类型的 MM 约会，要用不同的策略，有的请电影比较好，有的则去吃小吃效果不错，有的去海边浪漫最合适，单 目的 都是为了得到 MM 的芳心，我的追 MM 锦囊中有好多 Strategy 哦。策略模式：策略模式针对一组算法，将每一个算法封装到具有共同接口的独立的类中，从 而使得它们可以相互替换。策略模式使得算法可以在不影响到客户端的 情况下发生变化。策略模式把行为和环境分开。环境类负责维持和查询行为类，各种算法在具体的策略类中提供。由于算法和环境独立开来，算法的增减，修改都不 会影响到环境和客户端。

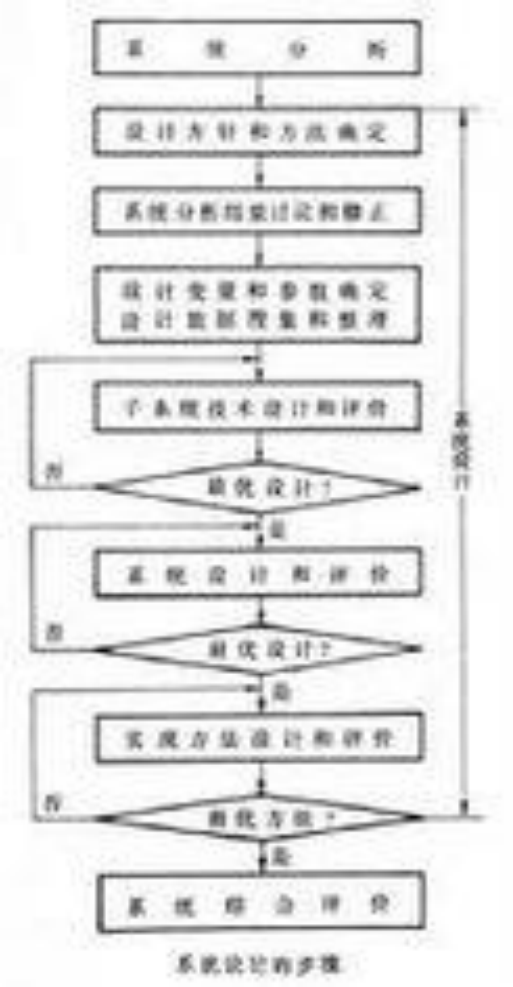
22、Ttemplate method(模板方法)??看过《如何说服女生上床》这部经典文章吗？女生从认识到上床的不变的步骤分为巧遇、打破僵局、展开追求、接吻、前戏、动手、 爱抚、进去八大步骤(Template method)，但每个步骤针对不同的情况，都有不一样的做法，这就要看你随机应变啦(具体实现)； 模板方法模式：模板方法模式准备一个抽象类，将部分逻辑以具体方法以及具体构造子的形式实现，然后声明一些抽象方法来迫使子类实现剩余的逻辑。不同的子类 可以以不同的方式实现这些抽象方法，从而对剩余的逻辑有不同的实现。先制定一个顶级逻辑框架，而将逻辑的细节留给具体的子类去实现。

23、visitor(访问者)?情人节到了，要给每个 MM 送一束鲜花和一张卡片，可是每个 MM 送的花都要针对她个人的特点，每张卡片也要根据 个人的特点来挑，我一个人哪搞得清楚，还是找花店老板和礼品店老板做一下 Visitor，让花店老板根据 MM 的特点选一束花，让礼品店老板也根据每个人特 点选一 张卡，这样就轻松多了； 访问者模式：访问者模式的目的是封装一些施加于某种数据结构元素之上的操作。一旦这些操作需要修改的话，接受这个操作的数据结构可以保持不变。访问者模式 适用于数据结构相对未定的系统，它把数据结构和作用于结构上的操作之间的耦合解脱开，使得操作集合可以相对自由的演化。访问者模式使得增加新的操作变 的很容易，就是增加一个新的访问者类。访问者模式将有关的行为集中到一个访问者对象中，而不是分散到一个个的节点类中。当使用访问者模式时，要将尽可能多 的对象浏览逻辑放在访问者类中，而不是放到它的子类中。访问者模式可以跨过几个类的等级结构访问属于不同的等级结构的成员类。

简介

系统设计是新系统的物理设计阶段。根据系统分析阶段所确定的新系统的逻辑模型、功能要求，在用户提供的环境条件下，设计出一个能在计算机网络环境上实施的方案，即建立新系统的物理模型。

这个阶段的任务是设计软件系统的模块层次结构，设计数据库的结构以及设计模块的控制流程，其目的是明确软件系统"如何做"。这个阶段又分两个步骤：概要设计和详细设计。概要设计解决软件系



统的模块划分和模块的层次机构以及数据库设计；详细设计解决每个模块的控制流程，内部算法和数据结构的设计。这个阶段结束，要交付概要设计说明书和设计说明，也可以合并在一起，称为设计说明书。

在系统分析的基础上，设计出能满足预定目标的系统的过程。系统设计内容主要包括：确定设计方针和方法,将系统分解为若干子系统,确定各子系统的目标、功能及其相互关系，决定对子系统的管理体制和控制方式，对各子系统进行技术设计和评价，对全系统进行技术设计和评价等。图中表示系统设计的各个步骤。

系统设计通常应用两种方法：一种是归纳法，另一种是演绎法。应用归纳法进行系统设计的程序是：首先尽可能地收集现有的和过去的同类系统的系统设计资料；在对这些系统的设计、制造和运行状况进行分析研究的基础上,根据所设计的系统的功能要求进行多次选择,然后对少数几个同类系统作出相应修正，最后得出一个理想的系统。演绎法是一种公理化方法，即先从普遍的规则和原理出发，根据设计人员的知识和经验，从具有一定功能的元素集合中选择能符合系统功能要求的多种元素，然后将这些元素按照一定形式进行组合（见系统结构），从而创造出具有所需功能的新系统。在系统设计的实践中，这两种方法往往是并用的。

系统设计原则

阶段开发原则

系统框架和数据结构全面设计，具体功能实现分阶段进行。网站的建设过程可以采取以下三期：第一期工程搭建网站的基本构架，实现电子商务网的大部分功能，初步实现网上交易；第二期工程实现网上竞价系统的全部功能；第三期工程实现网站在线的 B to B 交易。

易用性原则

方便上网客户浏览和操作，最大限度地减轻后台管理人员的负担，做到部分业务的自动化处理。

业务完整性原则

对于业务进行中的特殊情况能够做出及时、正确的响应，保证业务数据的完整性。

业务规范化原则

在系统设计的同时，也为将来的业务流程制定了较为完善的规范，具有较强的实际操作性。

可扩展性原则

系统设计要考虑到业务未来发展的需要，要尽可能设计得简明，各个功能模块间的耦合度小，便于系统的扩展。如果存在旧有的数据库系统，则需要充分考虑兼容性。

系统设计的方法

系统设计的方法主要包括结构化生命周期法（又称瀑布法）、原型化方法（迭 代法）、面向对象方法。按时间过程来分，开发方法分为生命周期法和原型法，实际上还有许多处于中间状态的方法。原型法又按照对原型结果的处理方式分为试验 原型法和演进原型法。试验原型法只把原型当成试验工具，试了以后就抛掉，根据试验的结论做出新的系统。演进原型法则把试好的结果保留，成为最终系统的一部 分。按照系统的分析要素，可以把开发方法分为三类： 1、面向处理方法（Processing Oriented ， 简称 PO） 2、向数据方法（Data Oriented ， 简称 DO） 3、向对象的方法（Object Oriented ， 简称 OO）