

# 剑指 Offer

## 名企面试官精讲典型编程题

何海涛◎著



```
    double Power(double base, int exponent) {
        if(exponent < 0) {
            return 1 / Power(base, -exponent);
        }
        if(exponent == 0) {
            return 1;
        }
        if(exponent % 2 == 1) {
            return base * Power(base, exponent - 1);
        } else {
            return Power(base * base, exponent / 2);
        }
    }

    bool equal(double num1, double num2) {
        if((num1 - num2) > 0.000001)
            if(num1 - num2 < -0.000001)
                return true;
            else
                return false;
        else
            return true;
    }

    double PowerWithUnsignedExponent(double base, unsigned int exponent) {
        if(exponent == 0)
            return 1;
        if(exponent == 1)
            return base;
        double result = PowerWithUnsignedExponent(base, exponent / 2);
        result *= result;
        if(exponent % 2 == 1)
            result *= base;
        return result;
    }
```



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>

# 剑指Offer

---

## 名企面试官精讲典型编程题

( 纪念版 )

何海涛 著

电子工业出版社  
Publishing House of Electronics Industry  
北京•BEIJING

## 内 容 简 介

《剑指 Offer——名企面试官精讲典型编程题（纪念版）》是为纪念本书英文版全球发行而推出的特殊版本，在原版基础上新增大量本书英文版中的精选题目，系统整理基础知识、代码质量、解题思路、优化效率和综合能力这 5 个面试要点。全书分为 8 章，主要包括面试流程：讨论面试每一环节需要注意的问题；面试需要的基础知识：从编程语言、数据结构及算法三方面总结程序员面试知识点；高质量代码：讨论影响代码质量的 3 个要素（规范性、完整性和鲁棒性），强调高质量代码除完成基本功能外，还能考虑特殊情况并对非法输入进行合理处理；解题思路：总结编程面试中解决难题的有效思考模式，如在面试中遇到复杂难题，应聘者可利用画图、举例和分解这 3 种方法将其化繁为简，先形成清晰思路再动手编程；优化时间和空间效率：读者将学会优化时间效率及空间换时间的常用算法，从而在面试中找到最优解；面试必备能力：总结应聘者如何充分表现学习和沟通能力，并通过具体面试题讨论如何培养知识迁移、抽象建模和发散思维能力；综合面试案例：总结哪些面试举动是不良行为，而哪些表现又是面试官所期待的行为；英文版面试题增补，优选久经欧美知名企业面试考验的经典题目，帮助国内读者开阔视野、增补技能。

《剑指 Offer——名企面试官精讲典型编程题（纪念版）》适合即将走向工作岗位的大学生阅读，也适合作为正在应聘软件行业的相关就业人员和计算机爱好者的参考书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

### 图书在版编目（CIP）数据

剑指 Offer：名企面试官精讲典型编程题：纪念版 /何海涛著. — 北京：电子工业出版社，2014.6

ISBN 978-7-121-23245-9

I. ①剑… II. ①何… III. ①程序设计—工程技术人员—资格考试—习题集 IV. ①TP311.1-44

中国版本图书馆 CIP 数据核字(2014)第 102607 号

策划编辑：张春雨

责任编辑：徐津平

特约编辑：赵树刚

印 刷：北京丰源印刷厂

装 订：三河市鹏成印业有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本：787×980 1/16 印张：19.5 字数：499.2 千字

版 次：2012 年 1 月第 1 版

2014 年 6 月第 2 版

印 次：2014 年 6 月第 1 次印刷

定 价：55.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：  
(010) 88254888。

质量投诉请发邮件至 [zlts@phei.com.cn](mailto:zlts@phei.com.cn)，盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

服务热线：(010) 88258888。

# 纪念版序言

---

《剑指 Offer——名企面试官精讲典型编程面试题》一书从 2011 年年底出版以来，已经两年多过去了。在这段时间里，我自己的生活和工作都发生了很大的变化。写书的时候儿子小呼呼还没有出生，我还只能透过他妈妈的肚皮感受他的胎动。这次在为纪念版添加新内容的时候，他会时不时跑过来要求坐到我的膝盖上，然后在笔记本的触摸屏上指指点点。当时我还在思科工作，现在已经重新回到了微软。工作之余，我在《剑指 Offer——名企面试官精讲典型编程面试题》这本书的基础上增加不少内容用英文出版了 *Coding Interviews: Questions, Analysis & Solutions* 并在全球多个国家发行。

*Coding Interviews* 一书出版之后，国内就有不少读者在询问什么时候可以把新增加的内容添加到中文版的《剑指 Offer——名企面试官精讲典型编程面试题》里。这次趁着博文视点张春雨编辑的邀请出纪念版的机会，我从英文版选取新增加的 17 个典型面试，集中放在本书的第 8 章。这些新增加的题目，有些涵盖了新的知识点。比如原版本中没有回溯法相关的内容，这次新增了两个需要用回溯法解决的面试题（面试题 66 和面试题 67）。正则表达式是编程面试时经常出现的内容，本次新增了两个正则表达式匹配的问题（面试题 53 和面试题 54）。

这次新增的内容有些是原有内容的延伸。比如原书的面试题 35 要求找出字符串中第一个只出现一次的字符。这次新增的面试题 55 把要求改为从一个字符流里找出第一个只出现一次的字符。再比如在原书的面试

题 23 中讨论了如何把二叉树按层打印到一行里，这次新增了两个按层打印二叉树的面试题。面试题 60 要求把二叉树的每一层打印单独打印到一行，面试题 61 要求按之字形顺序打印二叉树。

通过读者的 E-mail 和微博私信，我很高兴地得知《剑指 Offer——名企面试官精讲典型编程面试题》一书陪伴着很多读者找到心仪的工作，拿到满意的 Offer。实际上这本书不仅仅是一本关于求职面试的工具书，同时还是一本关于编程的技术书。书中有大量的篇幅讨论数据结构和算法，讨论如何才能写出高质量的代码。这些技能在面试的时候有用，在平时的开发工作中同样有用。希望本书能陪伴更多的读者在职场里成长。

何海涛

2014 年 5 月 25 日深夜于上海中山公园

# 推荐序一

---

海涛 2008 年在我的团队做过软件开发工程师。他是一个很细心的员工，对面试这个话题很感兴趣，经常和我及其他员工讨论，积累了很多面试方面的技巧和经验。他曾跟我提过想要写本有关面试的书，三年过后他把书写出来了！他是一个有目标、有耐心和持久力的人。

我在微软做了很多年的面试官，后面七年多作为把关面试官也面试了很多应聘者。应聘者要想做好面试，确实应把面试当作一门技巧来学习，更重要的是要提高自身的能力。我遇到很多应试者可能自身能力也不差但因为不懂得怎样回答提问，不能很好发挥。也有很多校园来的应聘者也学过数据结构和算法分析，可是到处理具体问题时不能用学过的知识来有效地解决问题。这些朋友读读海涛的这本书，会很受益，在面试中的发挥也会有很大提高。这本书也可以作为很好的教学补充资料，让学生不只学到书本知识，也学到解决问题的能力。

在我汇报的员工中有面试发挥很好但工作平平的，也有面试一般但工作优秀的。对于追求职业发展的人来说，通过面试只是迈过一个门槛而不是目的，真正的较量是在入职后的成长。就像学钓鱼，你可能在有经验的垂钓者的指导下能钓到几条鱼，但如果没学到垂钓的真谛，离开了指导者你可能就很难钓到很多鱼。我希望读这本书的朋友不要只学一些技巧来对付面试，而是通过学习如何解决面试中的难题来提高自己的编程和解决问题的能力，进而提高自信心，在职场中能迅速成长。

徐鹏阳 (Pung Xu)

Principal Development Manager, Search Technology Center Asia

Microsoft

# 推荐序二

---

I had the privilege of working with Harry at Microsoft. His background and industry experience are a great asset in learning about the process and techniques of technical interviews. Harry shares practical information about what to expect in a technical interview that goes beyond the core engineering skills. An interview is more than a skills assessment. It is the chance for you and a prospective employer to gauge whether there is a mutual fit. Harry includes reminders about the key factors that can determine a successful interview as well as success in your new job.

Harry takes you through a set of interview questions to share his insight into the key aspects of the question. By understanding these questions, you can learn how to approach any question more effectively. The basics of languages, algorithms and data structures are discussed as well as questions that explore how to write robust solutions after breaking down problems into manageable pieces. Harry also includes examples to focus on modeling and creative problem solving.

The skills that Harry teaches for problem solving can help you with your next interview and in your next job. Understanding better the key problem solving techniques that are analyzed in an interview can help you get the first job after university or make your next career move.

Matt Gibbs

Direct of Development, Asia Research & Development

Microsoft Corporation

# 目 录

## CONTENTS

<b>第 1 章 面试的流程 .....</b>	<b>1</b>
1.1 面试官谈面试.....	1
1.2 面试的三种形式.....	2
1.2.1 电话面试 .....	2
1.2.2 共享桌面远程面试 .....	3
1.2.3 现场面试 .....	4
1.3 面试的三个环节 .....	5
1.3.1 行为面试环节 .....	5
应聘者的项目经验 .....	6
应聘者掌握的技能 .....	7
回答“为什么跳槽” .....	8
1.3.2 技术面试环节 .....	10
扎实的基础知识 .....	10
高质量的代码 .....	11
清晰的思路 .....	14
优化效率的能力 .....	15
优秀的综合能力 .....	16
1.3.3 应聘者提问环节 .....	17
1.4 本章小结 .....	18
<b>第 2 章 面试需要的基础知识 .....</b>	<b>20</b>
2.1 面试官谈基础知识.....	20

2.2 编程语言 .....	22
2.2.1 C++ .....	22
面试题 1：赋值运算符函数 .....	24
经典的解法，适用于初级程序员 .....	25
考虑异常安全性的解法，高级程序员必备 .....	26
2.2.2 C# .....	27
面试题 2：实现 Singleton 模式 .....	31
不好的解法一：只适用于单线程 .....	31
不好的解法二：可用于多线程但效率不高 .....	32
可行的解法：同步锁前后两次判断 .....	33
推荐的解法一：利用静态构造函数 .....	34
推荐的解法二：按需创建实例 .....	34
解法比较 .....	35
2.3 数据结构 .....	36
2.3.1 数组 .....	36
面试题 3：二维数组中的查找 .....	38
2.3.2 字符串 .....	42
面试题 4：替换空格 .....	44
$O(n^2)$ 的解法，不足以拿到 Offer .....	45
$O(n)$ 的解法，搞定 Offer 就靠它 .....	46
2.3.3 链表 .....	49
面试题 5：从尾到头打印链表 .....	51
2.3.4 树 .....	53
面试题 6：重建二叉树 .....	55
2.3.5 栈和队列 .....	58
面试题 7：用两个栈实现队列 .....	59
2.4 算法和数据操作 .....	62
2.4.1 查找和排序 .....	63
面试题 8：旋转数组的最小数字 .....	66
2.4.2 递归和循环 .....	71
面试题 9：斐波那契数列 .....	73
效率很低的解法，面试官不会喜欢 .....	73
面试官期待的实用解法 .....	74

O(logn)但不够实用的解法 .....	74
解法比较 .....	75
2.4.3 位运算 .....	77
面试题 10：二进制中 1 的个数 .....	78
可能引起死循环的解法 .....	79
常规解法 .....	79
能给面试官带来惊喜的解法 .....	80
2.5 本章小结 .....	82
<b>第 3 章 高质量的代码 .....</b>	<b>84</b>
3.1 面试官谈代码质量 .....	84
3.2 代码的规范性 .....	86
3.3 代码的完整性 .....	87
从 3 方面确保代码的完整性 .....	87
3 种错误处理的方法 .....	88
面试题 11：数值的整数次方 .....	90
自以为题目简单的解法 .....	90
全面但不够高效的解法，离 Offer 已经很近了 .....	90
全面又高效的解法，确保能拿到 Offer .....	92
面试题 12：打印 1 到最大的 n 位数 .....	94
跳进面试官陷阱 .....	94
在字符串上模拟数字加法 .....	94
把问题转换成数字排列 .....	97
面试题 13：在 O(1)时间删除链表结点 .....	99
面试题 14：调整数组顺序使奇数位于偶数前面 .....	102
只完成基本功能的解法，仅适用于初级程序员 .....	102
考虑可扩展性的解法，能秒杀 Offer .....	104
3.4 代码的鲁棒性 .....	106
面试题 15：链表中倒数第 k 个结点 .....	107
面试题 16：反转链表 .....	112
面试题 17：合并两个排序的链表 .....	114
面试题 18：树的子结构 .....	117
3.5 本章小结 .....	121

<b>第 4 章 解决面试题的思路.....</b>	<b>123</b>
<b>4.1 面试官谈面试思路.....</b>	<b>123</b>
面试题 19：二叉树的镜像 .....	125
<b>4.2 画图让抽象问题形象化.....</b>	<b>125</b>
面试题 20：顺时针打印矩阵 .....	127
<b>4.3 举例让抽象问题具体化.....</b>	<b>131</b>
面试题 21：包含 min 函数的栈 .....	132
面试题 22：栈的压入、弹出序列 .....	134
面试题 23：从上往下打印二叉树 .....	137
面试题 24：二叉搜索树的后序遍历序列 .....	140
面试题 25：二叉树中和为某一值的路径 .....	143
<b>4.4 分解让复杂问题简单化.....</b>	<b>146</b>
面试题 26：复杂链表的复制 .....	147
面试题 27：二叉搜索树与双向链表 .....	151
面试题 28：字符串的排列 .....	154
<b>4.5 本章小结.....</b>	<b>158</b>
<b>第 5 章 优化时间和空间效率 .....</b>	<b>160</b>
<b>5.1 面试官谈效率.....</b>	<b>160</b>
<b>5.2 时间效率.....</b>	<b>162</b>
面试题 29：数组中出现次数超过一半的数字.....	163
基于 Partition 函数的 O(n)算法 .....	163
利用数组特点的 O(n)算法 .....	165
解法比较 .....	166
面试题 30：最小的 k 个数 .....	167
O(n)的算法，只当可以修改输入数组时可用 .....	167
O(nlogk)的算法，适合处理海量数据 .....	168
解法比较 .....	169
面试题 31：连续子数组的最大和 .....	171
举例分析数组的规律 .....	171
应用动态规划法 .....	173
面试题 32：从 1 到 n 整数中 1 出现的次数.....	174

不考虑效率的解法, 想拿 Offer 有点难.....	174
明显提高效率的解法, 让面试官耳目一新.....	175
面试题 33: 把数组排成最小的数 .....	177
5.3 时间效率与空间效率的平衡.....	181
面试题 34: 丑数 .....	182
逐个判断整数是不是丑数的解法 .....	182
创建数组保存已经找到的丑数的解法 .....	183
面试题 35: 第一个只出现一次的字符 .....	186
面试题 36: 数组中的逆序对 .....	189
面试题 37: 两个链表的第一个公共结点 .....	193
5.4 本章小结.....	196
<b>第 6 章 面试中的各项能力 .....</b>	<b>198</b>
6.1 面试官谈能力.....	198
6.2 沟通能力和学习能力.....	200
沟通能力 .....	200
学习能力 .....	200
善于学习、沟通的人也善于提问 .....	201
6.3 知识迁移能力.....	203
面试题 38: 数字在排序数组中出现的次数.....	204
面试题 39: 二叉树的深度 .....	207
重复遍历结点的解法, 不足以打动面试官.....	209
只遍历结点一次的解法, 正是面试官喜欢的.....	209
面试题 40: 数组中只出现一次的数字 .....	211
面试题 41: 和为 s 的两个数字 VS 和为 s 的连续正数序列 .....	214
面试题 42: 翻转单词顺序 VS 左旋转字符串 .....	218
6.4 抽象建模能力.....	222
面试题 43: n 个骰子的点数 .....	223
基于递归求骰子点数, 时间效率不够高.....	223
基于循环求骰子点数, 时间性能好 .....	224
面试题 44: 扑克牌的顺子 .....	226
面试题 45: 圆圈中最后剩下的数字 .....	228
经典的解法, 用循环链表模拟圆圈 .....	229

创新的解法，拿到 Offer 不在话下 .....	230
<b>6.5 发散思维能力 .....</b>	<b>232</b>
面试题 46：求 $1+2+\dots+n$ .....	233
利用构造函数求解 .....	234
利用虚函数求解 .....	234
利用函数指针求解 .....	235
利用模板类型求解 .....	236
面试题 47：不用加减乘除做加法 .....	237
面试题 48：不能被继承的类 .....	239
常规的解法：把构造函数设为私有函数 .....	239
新奇的解法：利用虚拟继承 .....	240
<b>6.6 本章小结 .....</b>	<b>241</b>
<b>第 7 章 两个面试案例 .....</b>	<b>243</b>
7.1 案例一：（面试题 49）把字符串转换成整数 .....	244
7.2 案例二：（面试题 50）树中两个结点的最低公共祖先 .....	252
<b>第 8 章 英文版新增面试题 .....</b>	<b>261</b>
8.1 数组 .....	261
面试题 51：数组中重复的数字 .....	261
面试题 52：构建乘积数组 .....	263
8.2 字符串 .....	265
面试题 53：正则表达式匹配 .....	265
面试题 54：表示数值的字符串 .....	267
面试题 55：字符流中第一个不重复的字符 .....	269
8.3 链表 .....	270
面试题 56：链表中环的入口结点 .....	270
面试题 57：删除链表中重复的结点 .....	273
8.4 树 .....	275
面试题 58：二叉树的下一个结点 .....	275
面试题 59：对称的二叉树 .....	277

面试题 60：把二叉树打印成多行 .....	278
面试题 61：按之字形顺序打印二叉树 .....	280
面试题 62：序列化二叉树 .....	283
面试题 63：二叉搜索树的第 k 个结点 .....	285
面试题 64：数据流中的中位数 .....	286
8.5 栈和队列 .....	290
面试题 65：滑动窗口的最大值 .....	290
8.6 回溯法 .....	294
面试题 66：矩阵中的路径 .....	294
面试题 67：机器人的运动范围 .....	296

## 第1章

# 面试的流程

### 面试官谈面试

“对于初级程序员，我一般会偏向考查算法和数据结构，看应聘者的基本功；对于高级程序员，我会多关注专业技能和项目经验。”

——何幸杰（SAP，高级工程师）

“应聘者要事先做好准备，对公司近况、项目情况有所了解，对所应聘的工作真的很有热情。另外，应聘者还要准备好合适的问题问面试官。”

——韩伟东（盛大，高级研究员）

“应聘者在面试过程首先需要放松，不要过于紧张，这有助于后面解决问题时开拓思路。其次不要急于编写代码，应该先了解清楚所要解决的问题。这时候最好先和面试官多做沟通，然后开始做一些整体的设计和规划，这有助于编写高质量和高可读性的代码。写完代码后不要马上提交，最好自己 review 并借助一些测试用例来走几遍代码，找出可能出现的错误。”

——尧敏（淘宝，资深经理）

“‘神马’都是浮云，应聘技术岗位就是要踏实写程序。”

——田超（微软，SDE II）

# 1.2

## 面试的三种形式

如果应聘者能够通过公司的简历筛选环节，那恭喜他取得了阶段性的成功。但要想拿到心仪的 Offer，应聘者还有更长的路要走。大部分公司的面试都是从电话面试开始的。通过电话面试之后，有些公司还会有一两轮远程面试。面试官让应聘者共享自己的桌面，远程观察应聘者编写及调试代码的过程。如果前面的面试都很顺利，应聘者就会收到现场面试的邀请信，请他去公司接受面对面的面试。整个面试的流程我们可以用图 1.1 表示。

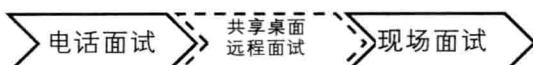


图 1.1 面试的形式和流程

注：只有少数公司有共享桌面远程面试环节。

### 1.2.1 电话面试

顾名思义，电话面试是面试官以打电话的形式考查应聘者。有些面试官会先和应聘者预约好电话面试的时间，而还有些面试官却喜欢搞突然袭击，一个电话打过去就开始面试。为了应付这种突然袭击，建议应聘者在投出简历之后的一两个星期之内，要保证手机电池能至少连续通话一个小时。另外，应聘者不要长时间呆在很嘈杂的地方。如果应聘者身在闹市的时候突然接到面试电话，那么双方就有可能因为听不清对方而倍感尴尬。

电话面试和现场面试最大的区别就是应聘者和面试官是见不到对方的，因此双方的沟通只能依靠声音。没有了肢体语言、面部表情，应聘者清楚地表达自己想法的难度就比现场面试时要大很多，特别是在解释复杂算法的时候。应聘者在电话面试的时候应尽可能用形象化的语言把细节说清楚。例如，在现场面试的时候，应聘者如果想说一个二叉树的结构，可以用笔在白纸上画出来，就一目了然。但在电话面试的时候，应聘者就需要把二叉树中有哪些结点，每个结点的左子结点是什么、右子结点是什么都要说得很清楚，只有这样面试官才能准确地理解应聘者的思路。

很多外企在电话面试时都会加上英语面试的环节，甚至有些公司全部面试都会用英语进行。电话面试时应聘者只能听到面试官的声音而看不到他的口型，这对应聘者的听力提出了更高的要求。如果应聘者在面试的时候没有听清楚或者听懂面试官的问题，千万不要不懂装懂、答非所问，这是面试的大忌。当不确定面试官的问题的时候，应聘者一定要大胆地向面试官多提问，直到弄清楚面试官的意图为止。



#### 面试小提示：

应聘者在电话面试的时候应尽可能用形象的语言把细节说清楚。

如果在英语面试时没有听清或没有听懂面试官的问题，应聘者要敢于说 Pardon。

### 1.2.2 共享桌面远程面试

共享桌面远程面试（Phone-Screen Interview）是指利用一些共享桌面的软件（比如微软的 Live Meeting、思科的 WebEx 等），应聘者把自己电脑的桌面共享给远程的面试官。这样两个人虽然没有坐在一起，但面试官却能通过共享桌面观看应聘者编程和调试的过程。目前只有为数不多的几家公司会在邀请应聘者到公司参加现场面试之前，先进行一两轮共享桌面的远程面试。

这种形式的面试，面试官最关心的是应聘者的编程习惯及调试能力。通常面试官会认可应聘者下列几种编程习惯：

- **思考清楚再开始编码。**应聘者不要一听到题目就匆忙打开编程软件如 Visual Studio 开始敲代码，因为在没有形成清晰的思路之前写出的代码通常会漏洞百出。这些漏洞被面试官发现之后，应聘者容易慌张，这个时候再修改代码也会越改越乱，最终导致面试的结果不理想。更好的策略是应聘者应先想清楚解决问题的思路，算法的时间、空间复杂度各是什么，有哪些特殊情况需要处理等，然后再动手编写代码。
- **良好的代码命名和缩进对齐习惯。**一目了然的变量和函数名，加以合理的缩进和括号对齐，会让面试官觉得应聘者有参与大型项目的

开发经验。

- **能够单元测试。**通常面试官出的题目都是要求写函数解决某一问题，如果应聘者能够在定义函数之后，立即对该函数进行全面的单元测试，那就相当于向面试官证明了自己有着专业的软件开发经验。如果应聘者是先写单元测试用例，再写解决问题的函数，我相信面试官定会对你刮目相看，因为能做到测试在前、开发在后的程序员实在是太稀缺了，他会毫不犹豫地抛出绿色的橄榄枝。

通常我们在写代码的时候都会遇到问题。当应聘者运行代码发现结果不对之后的表现，也是面试官关注的重点，因为应聘者此时的反应、采取的措施都能体现出他的调试功底。如果应聘者能够熟练地设置断点、单步跟踪、查看内存、分析调用栈，能很快发现问题的根源并最终解决问题，那么面试官将会觉得他的开发经验很丰富。调试能力是在书本上学不到的，只有通过大量的软件开发实践才能积累出调试技巧。当面试官发现一个应聘者的调试功底很扎实的时候，他在写面试报告的时候是不会吝啬赞美之词的。



#### 面试小提示：

在共享桌面远程面试过程中，面试官最关心的是应聘者的编程习惯及调试能力。

### 1.2.3 现场面试

在通过电话面试和共享桌面远程面试之后，应聘者不久就会收到 E-mail，邀请他去公司参加现场面试（Onsite Interview）。

去公司参加现场面试之前，应聘者应做好以下几点准备：

- **规划好路线并估算出行时间。**应聘者要事先估算在路上需要花费多长时间，并预留半小时左右的缓冲时间以应对堵车等意外情况。如果面试迟到，那至少印象分会大打折扣。
- **准备好得体的衣服。**IT 公司通常衣着比较随意，应聘者通常没有必要穿着正装，一般舒服干净的衣服都可以。

- 注意面试邀请函里的面试流程。如果面试有好几轮，时间也很长，那么你在面试过程中可能会觉得疲劳并思维变得迟钝。比如微软对技术职位通常有五轮面试，连续几个小时处在高压的面试之中，人难免会变得精疲力尽。因此应聘者可以带一些提神的饮料或者食品，在两轮面试之间提神醒脑。
- 准备几个问题。每一轮面试的最后，面试官都会让应聘者问几个问题，应聘者可以提前准备好问题。

现场面试是整个面试流程中的重头戏。由于是坐在面试官的对面，应聘者的一举一动都看在面试官的眼里。面试官通过应聘者的语言和行动，考查他的沟通能力、学习能力、编程能力等综合实力。本书接下来的章节将详细讨论各种能力。

## 1.3

### 面试的三个环节

通常面试官会把每一轮面试分为三个环节（如图 1.2 所示）：首先是行为面试，面试官参照简历了解应聘者的过往经验；然后是技术面试，这一环节很有可能会要求应聘者现场写代码；最后一个环节是应聘者问几个自己最感兴趣的问题。下面将详细讨论面试的这三个环节。

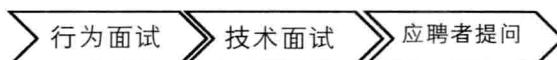


图 1.2 面试的三个环节

#### 1.3.1 行为面试环节

面试开始的 5~10 分钟通常是行为面试的时间。在行为面试这个环节里，面试官会注意应聘者的性格特点，深入地了解简历中列举的项目经历。由于这一环节一般不会问技术难题，因此也是一个暖场的过程，应聘者可以利用这几分钟时间调整自己的情绪，进入面试的状态。

不少面试官会让应聘者做一个简短的自我介绍。由于面试官手中拿着应聘者的简历，而那里有应聘者的详细信息，因此此时的自我介绍不用花

很多时候，用 30 秒到 1 分钟的时间介绍自己的主要学习、工作经历就即可。如果面试官对你的某一段经历或者参与的某一个项目很感兴趣，他会有针对性地提几个问题详细了解。

### 1. 应聘者的项目经验

应聘者自我介绍之后，面试官接着会对照应聘者的简历去详细了解他感兴趣的项目。应聘者在准备简历的时候，建议用如图 1.3 所示的 STAR 模型描述自己经历过的每一个项目。



图 1.3 简历中描述项目的 STAR 模型

- **Situation:** 简短的项目背景，比如项目的规模，开发的软件的功能、目标用户等。
- **Task:** 自己完成的任务。这个要写详细，要让面试官对自己的工作一目了然。在用词上要注意区分“参与”和“负责”：如果只是加入某一个开发团队写了几行代码就用“负责”，那就很危险。面试官看到简历上应聘者“负责”了某个项目，他可能就会问项目的总体框架设计、核心算法、团队合作等问题。这些问题对于只是简单“参与”的人来说，是很难回答的，会让面试官认为你不诚实，印象分会减去很多。
- **Action:** 为了完成任务自己做了哪些工作，是怎么做的。这里可以详细介绍。做系统设计的，可以介绍系统架构的特点；做软件开发的，可以写基于什么工具在哪个平台下应用了哪些技术；做软件测试的，可以写是手工测试还是自动化测试，是白盒测试还是黑盒测试等。
- **Result:** 自己的贡献。这方面的信息可以写得具体些，最好能用数

字加以说明。如果是参与功能开发，可以说按时完成了多少功能；如果做优化，可以说性能提高的百分比是多少；如果是维护，可以说修改了多少个 Bug。

举个例子，笔者用下面一段话介绍自己在微软 Winforms 项目组的经历：

Winforms 是微软.NET 中的一个成熟的 UI 平台 (**Situation**)。本人的工作是在添加少量新功能之外主要负责维护已有的功能 (**Task**)。新的功能主要是让 Winforms 的控件的风格和 Vista、Windows 7 的风格保持一致。在维护方面，对于较难的问题我用 WinDbg 等工具进行调试 (**Action**)。在过去两年中我总共修改了超过 200 个 Bug (**Result**)。

如果在应聘者的简历中上述 4 类信息还不够清晰，面试官可能会追问相关的问题。除此之外，面试官针对项目经验最常问的问题还包括如下几个类型：

- 你在该项目中碰到的最大的问题是什么，你是怎么解决的？
- 从这个项目中你学到了什么？
- 什么时候会和其他团队成员（包括开发人员、测试人员、设计人员、项目经理等）有什么样的冲突，你们是怎么解决冲突的？

应聘者在准备简历的时候，针对每一个项目经历都应提前做好相应的准备。只有准备充分，应聘者在行为面试这个环节才可以表现得游刃有余了。



#### 面试小提示：

在介绍项目经验（包括在简历上介绍和面试时口头介绍）时，应聘者不必详述项目的背景，而要突出介绍自己完成的工作及取得的成绩。

## 2. 应聘者掌握的技能

除了应聘者参与过的项目之外，面试官对应聘者掌握的技能也很感兴趣，他有可能针对简历上提到的技能提出问题。和描述项目时要注意“参与”和“负责”一样，描述技能掌握程度时也要注意“了解”、“熟悉”和“精通”的区别。

“了解”指对某一个技术只是上过课或者看过书，但没有做过实际的项目。通常不建议在简历中列出只是肤浅地了解一点的技能，除非这项技术应聘的职位的确需要。比如某学生读本科的时候学过《计算机图形学》这门课程，但一直没有开发过与图形绘制相关的项目，那就只能算是了解。如果他去应聘 Autodesk 公司，那他可以在简历上提一下他了解图形学。Autodesk 是一个开发三维设计软件的公司，有很多职位或多或少都会与图形学有关系，那么了解图形学的总比完全不了解的要适合一些。但如果他是去应聘 Oracle，那就没有必要提这一点了，因为开发数据库系统的 Oracle 公司大部分职位与图形学没有什么关系。

简历中我们描述技能的掌握程度大部分应该是“熟悉”。如果我们在实际项目中使用某一项技术已经有较长的时间，通过查阅相关的文档可以独立解决大部分问题，我们就熟悉它了。对应届毕业生而言，他毕业设计所用到的技能，可以用“熟悉”；对已经工作过的，在项目开发过程中所用到的技能，也可以用“熟悉”。

如果我们对一项技术使用得得心应手，在项目开发过程中当同学或同事向我们请教这个领域的问题我们都有信心也有能力解决，这个时候我们就可以说自己精通了这项技术。应聘者不要试图在简历中把自己修饰成“高人”而轻易使用“精通”，除非自己能够很轻松地回答这个领域里的绝大多数问题，否则就会适得其反。通常如果应聘者在简历中说自己精通某一项技术，面试官就会对他有很高的期望值，因此会挑一些比较难的问题来问。这也是越装高手就越容易露馅的原因。曾经碰到一个在简历中说自己精通 C++ 的应聘者，连成员变量的初始化顺序这样的问题都被问得一头雾水，那最终的结果也就可想而知了。

### 3. 回答“为什么跳槽”

在面试已经有工作经验的应聘者的时候，面试官总喜欢问为什么打算跳槽。每个人都有自己的跳槽动机和原因，因此面试官也不会期待一个标准答案。面试官只是想通过这个问题来了解应聘者的性格，因此应聘者可以大胆地根据自己的真实想法来回答这个问题。但是，应聘者也不要说什么就说什么，以免给面试官留下负面的印象。

在回答这个问题时不要抱怨，也不要流露出负面的情绪。负面的情绪通常是能够传染的，当应聘者总是在抱怨的时候，面试官就会担心如果把

他招进来的话他将成为团队负面情绪的传染源，从而影响整个团队的士气。应聘者应尽量避免以下 4 个原因：

- **老板太苛刻。**如果面试官就是当前招聘的职位的老板，他听到应聘者抱怨现在的老板苛刻时，他肯定会想要是把这个人招进来，接下来他就会抱怨我也苛刻了。
- **同事太难相处。**如果应聘者说他周围有很多很难相处的同事，面试官很有可能会觉得这个人他本身就很难相处。
- **加班太频繁。**对于大部分 IT 企业来说，加班是家常便饭。如果正在面试的公司也需要经常加班，那等于应聘者说他不想进这家公司。
- **工资太低。**现在的工资太低的确是大部分人跳槽的真实原因，但不建议在面试的时候对面试官抱怨。面试的目的是拿到 offer，我们要尽量给面试官留下好印象。现在假设你是面试官，有两个人来面试：一个人一开口就说现在工资太低了，希望新工作能加多少多少工资；另一个说我只管努力干活，工资公司看着给，相信公司不会亏待勤奋的员工。你更喜欢哪个？这里不是说工资不重要，但我们要清楚面试不是谈工资的时候。等完成技术面试之后谈 offer 的时候，再和 HR 谈工资也不迟。通过面试之后我们就掌握主动了，想怎么谈就怎么谈，如果工资真的开高了 HR 会和你很客气地商量。

笔者在面试的时候，通常给出的答案是：现在的工作做了一段时间，已经没有太多的激情了，因此希望寻找一份更有挑战的工作。然后具体论述为什么有些厌倦现在的职位，以及面试的职位我为什么会有兴趣。笔者自己跳过两次槽，第一次从 Autodesk 跳槽到微软，第二次从微软跳槽到现在的思科。从面试的结果来看，这样的回答都让面试官很满意，最终也都拿到了 offer。

当时在微软面试被问到为什么要跳槽时，笔者的回答是：我在 Autodesk 开发的软件 Civil 3D 是一款面向土木行业设计软件。如果我想在现在的职位上得到提升，就必须加强土木行业的学习，可我对诸如计算土方量、道路设计等没有太多兴趣，因此出来寻找机会。

在微软工作两年半之后去思科面试的时候，笔者的回答是：我在微软的主要工作是开发和维护.NET 的 UI 平台 Winforms。由于 Winforms 已经非常成熟，不需要添加多少新功能，因此我的大部分工作都是维护和修改 BUG。两年下来，调试的能力得到了很大的提高，但长期如此自己的软件

开发和设计能力将不能得到提高，因此想出来寻找可以设计和开发系统的职位。同时，我在过去几年里的工作都是开发桌面软件，对网络了解甚少，因此希望下一个工作能与网络相关。众所周知，思科是个网络公司，这里的软件和系统或多或少都离不开网络，因此我对思科的职位很感兴趣。

### 1.3.2 技术面试环节

面试官在通过简历及行为面试大致了解应聘者的背景之后，接下来就要开始技术面试了。一轮 1 小时的面试，通常技术面试会占据 40~50 分钟。这是面试的重头戏，对面试的结果起决定性作用。虽然不同公司里不同面试官的背景、性格各不相同，但总体来说他们都会关注应聘者 5 种素质：扎实的基础知识、能写高质量的代码、分析问题时思路清晰、能优化时间效率和空间效率，以及学习沟通等各方面的能力（如图 1.4 所示）。



图 1.4 应聘者需要具备的素质

应聘者在面试之前需要做足准备，对编程语言、数据结构和算法等基础知识有全面的了解。面试的时候如果遇到简单的问题，应聘者一定要注重细节，写出完整、鲁棒的代码。如果遇到复杂的问题，应聘者可以通过画图、举具体例子分析和分解复杂问题等方法先理清思路再动手编程。除此之外，应聘者还应该不断优化时间效率和空间效率，力求找到最优的解法。在面试过程中，应聘者还应该主动提问，以弄清楚题目的要求，表现自己的沟通能力。当面试官前后问的两个问题有相关性的时候，尽量把解决前面问题的思路迁移到后面的问题中去，展示自己良好的学习能力。如果能做到这么几点，那么通过面试获得心仪的职位将是水到渠成的事情。

#### 1. 扎实的基础知识

扎实的基本功是成为优秀程序员的前提条件，因此面试官首要关注的

应聘者素质就是是否具备扎实的基础知识。通常基本功在编程面试环节体现在3个方面：编程语言、数据结构和算法。

首先，每个程序员至少要掌握一两门编程语言。面试官从应聘者在面试过程中写的代码及跟进的提问中，能看出其编程语言掌握的熟练程度。以大部分公司面试要求的C++举例。如果写的函数需要传入一个指针，面试官可能会问是否需要为该指针加上const，把const加在指针不同的位置是否有区别；如果写的函数需要传入的参数是一个复杂类型的实例，面试官可能会问传入值参数和传入引用参数有什么区别，什么时候需要为传入的引用参数加上const。

其次，数据结构通常是编程面试过程中考查的重点。在参加面试之前，应聘者需要熟练掌握链表、树、栈、队列和哈希表等数据结构，以及它们的操作。如果我们留意各大公司的面试题，就会发现链表和二叉树相关的问题是很多面试官喜欢问的问题。这方面的问题看似比较简单，但要真正掌握也不容易，特别适合在这么短的面试时间内检验应聘者的基本功。如果应聘者事先对链表的插入和删除结点了如指掌，对二叉树的各种遍历方法的循环和递归写法都烂熟于胸，那么真正到了面试的时候也就游刃有余了。

最后，大部分公司都会注重考查查找、排序等算法。应聘者可以在了解各种查找和排序算法的基础上，重点掌握二分查找、归并排序和快速排序，因为很多面试题都只是这些算法的变体而已。比如面试题8“旋转数组的最小数字”和面试题38“数字在排序数组中出现的次数”的本质是考查二分查找，而面试题36“数组中的逆序对”实际上是考查归并排序。少数对算法很重视的公司比如谷歌或者百度，还会要求应聘者熟练掌握动态规划和贪婪算法。如果应聘者对动态规划算法很熟悉，那么他就能很轻松地解决面试题31“连续子数组的最大和”。

在本书的第2章“面试需要的基础知识”中，我们将详细介绍应聘者需要熟练掌握的基础知识。

## 2. 高质量的代码

只有注重质量的程序员，才能写出鲁棒稳定的大型软件。在面试过程中，面试官总会格外关注边界条件、特殊输入等看似细枝末节但实质至关重要的地方，以考查应聘者是否注重代码质量。很多时候，面试官发现应聘者写出来的代码只能完成最基本的功能，一旦输入特殊的边界条件参数就会错误百出甚至程序崩溃。

总有些应聘者很困惑：面试的时候觉得题目很简单，感觉自己都做出来了，可最后为什么被拒了呢？面试被拒有很多种可能，比如面试官认为你性格不适合、态度不够诚恳等。但在技术面试过程中，这些都不是最重要的。技术面试的面试官一般都是程序员，程序员通常没有那么多想法，他们只认一个理：题目做对、做完整了，就让你通过面试；否则失败。所以遇到简单题目却被拒的情况，应聘者应认真反思在思路或者代码中存在哪些漏洞。

以微软面试开发工程师时最常用的一个问题为例：把一个字符串转换成整数。这个题目很简单，很多人都能在三分钟之内写出如下不到 10 行的代码：

```
int StrToInt(char* string)
{
    int number = 0;
    while(*string != 0)
    {
        number = number * 10 + *string - '0';
        ++string;
    }

    return number;
}
```

---

看了上面的代码，你是不是觉得微软面试很容易？如果你真的这么想，那你可能又要被拒了。

通常越是简单的问题，面试官的期望值就会越高。如果题目很简单，面试官就会期待应聘者能够很完整地解决问题，除了完成基本功能之外，还要考虑到边界条件、错误处理等各个方面。比如这道题，面试官不仅仅是期待你能完成把字符串转换成整数这个最起码的要求，而且希望你能考虑到各种特殊的输入。面试官至少会期待应聘者能够在不需要提示的情况下，考虑到输入的字符串中有非数字字符和正负号，要考虑到最大的正整数和最小的负整数以及溢出。同时面试官还期待应聘者能够考虑到当输入的字符串不能转换成整数时，应该如何做错误处理。当把这个问题的方方面面都考虑到的时候，我们就不会再认为这道题简单了。

除了问题考虑不全面之外，还有一个面试官不能容忍的错误就是程序不够鲁棒。以前面的那段代码为例，只要输入一个空指针，程序立即崩溃。这样的代码如果加入到软件当中，将是灾难。因此当面试官看到代码中对空指针没有判断并加以特殊处理的时候，通常他连往下看的兴趣都没有。

当然，不是所有与鲁棒性相关的问题都和前面的代码那样明显。再举一个很多人都曾经被面试过的问题：求链表中的倒数第 k 个结点。有不少人在面试之前在网上看过这个题目，因此知道思路是用两个指针，第一个指针先走  $k-1$  步，然后两个指针一起走。当第一个指针走到尾结点的时候，第二个指针指向的就是倒数第 k 个结点。于是他大笔一挥，写下了下面的代码：

```
ListNode* FindKthToTail(ListNode* pListHead, unsigned int k)
{
    if(pListHead == NULL)
        return NULL;

    ListNode *pAhead = pListHead;
    ListNode *pBehind = NULL;

    for(unsigned int i = 0; i < k - 1; ++ i)
    {
        pAhead = pAhead->m_pNext;
    }

    pBehind = pListHead;

    while(pAhead->m_pNext != NULL)
    {
        pAhead = pAhead->m_pNext;
        pBehind = pBehind->m_pNext;
    }

    return pBehind;
}
```

---

写完之后，应聘者看到自己已经判断了输入的指针是不是空指针并做了特殊处理，于是以为这次面试必定能顺利通过，可是他没有想到的是这段代码中仍然有很严重的问题：当链表中的结点总数小于 k 的时候，程序还是会崩溃。另外，当输入的 k 为 0 时，同样也会引起程序崩溃。因此，几天之后他收到的仍然不是 Offer 而是拒信。

要想很好地解决前面的问题，最好的办法是在动手写代码之前想好测试用例。只有把各种可能的输入事先都想好了，才能在写代码的时候把各种情况都做相应的处理。写完代码之后，也不要立刻给面试官检查，而是先在心里默默地运行。当输入之前想好的所有测试用例都能得到合理的输出时，再把代码交给面试官。做到了这一步，通过面试拿到 Offer 就是顺理成章的事情了。

在本书的第 3 章“高质量的代码”中，我们将详细讨论提高代码质量的方法。



### 面试小提示：

面试官除了希望应聘者的代码能够完成基本的功能之外，还会关注应聘者是否考虑了边界条件、特殊输入（比如 NULL 指针，空字符串等）及错误处理。

### 3. 清晰的思路

只有思路清晰，应聘者才有可能在面试过程中解决复杂的问题。有些时候面试官会有意出一些比较复杂的问题，以考查应聘者能否在短时间内形成清晰的思路并解决问题。对于确实很复杂的问题，面试官甚至不期待应聘者能在面试不到一个小时的时间里给出完整的答案，他更看重的可能还是应聘者是否有清晰的思路。面试官通常不喜欢应聘者在没有形成清晰思路之前就草率地开始写代码，这样写出来的代码容易逻辑混乱、错误百出。

应聘者可以用几个简单的方法帮助自己形成清晰的思路。首先是举几个简单的具体例子让自己理解问题。当我们一眼看不出问题中隐藏的规律的时候，可以试着用一两个具体的例子模拟操作的过程，这样说不一定就能通过具体的例子找到抽象的规律。其次可以试着用图形表示抽象的数据结构。像分析与链表、二叉树相关的题目，我们都可以画出它们的结构来简化题目。最后可以试着把复杂的问题分解成若干个简单的子问题，再一一解决。很多基于递归的思路，包括分治法和动态规划，都是把复杂的问题分解成一个或者多个简单的子问题。

比如把二叉搜索树转换成排序的双向链表这个问题就很复杂。遇到这个问题，我们不妨先画出一两个具体的二叉搜索树，直观地感受二叉搜索树和排序的双向链表有哪些联系。如果一下子找不出转换的规律，我们可以把整个二叉树看成 3 个部分：根结点、左子树和右子树。当我们递归地把转换左右子树这两个子问题解决之后，再把转换左右子树得到的链表和根结点链接起来，整个问题也就解决了（详见面试题 27 “二叉搜索树与双向链表”）。

在本书的第 4 章“解决面试题的思路”中，我们将详细讨论遇到复杂问题时如何采用画图、举例和分解问题等方法帮助我们解决问题。



### 面试小提示：

如果在面试的时候遇到难题，我们有3种办法分析、解决复杂的问题：画图能使抽象问题形象化，举例使抽象问题具体化，分解使复杂问题简单化。

#### 4. 优化效率的能力

优秀的程序员对时间和内存的消耗锱铢必较，他们很有激情地不断优化自己的代码。当面试官出的题目有多种解法的时候，通常他会期待应聘者最终能够找到最优解。当面试官提示还有更好的解法的时候，应聘者不能放弃思考，而应该努力寻找在时间消耗或者空间消耗上可以优化的地方。

要想优化时间或者空间效率，首先要知道如何分析效率。即使是同一个算法，用不同方法实现的效率可能也会大不相同，我们要能够分析出算法及其代码实现的效率。例如求斐波那契数列，很多人喜欢用递归公式 $f(n)=f(n-1)+f(n-2)$ 求解。如果分析它的递归调用树，我们就会发现有大量的计算是重复的，时间复杂度以n的指数增加。但如果我们将先求 $f(1)$ 、 $f(2)$ ，再根据 $f(1)$ 和 $f(2)$ 求出 $f(3)$ ，接下来根据 $f(2)$ 、 $f(3)$ 求出 $f(4)$ ，并以此类推用一个循环求出 $f(n)$ ，这种计算方法的时间效率就只有 $O(n)$ ，比前面递归的方法要好得多。

要想优化代码的效率，我们还要熟知各种数据结构的优缺点，并能选择合适的数据结构解决问题。我们在数组中根据下标可以用 $O(1)$ 时间完成查找。数组的这个特征可以用来实现简单的哈希表解决很多问题，比如面试题35“第一个只出现一次的字符”。为了解决面试题30“最小的k个数”，我们需要一个数据容器来存储k个数字。在这个数据容器中，我们希望能够快速地找到最大值并且能快速地替换其中的数字。经过权衡，我们发现二叉树比如最大堆或者红黑树都是实现这个数据容器的不错选择。

要想优化代码的效率，我们也要熟练掌握常用的算法。面试中最常用的算法是查找和排序。如果从头到尾顺序扫描一个数组，我们需要 $O(n)$ 时间才能完成查找操作。但如果数组是排序的，应用二分查找算法就能把时间复杂度降低到 $O(\log n)$ （如面试题8“旋转数组的最小值”和面试题38“数字在排序数组中出现的次数”）。排序算法除了能够给数组排序之外，还能用来解决其他问题。比如快速排序算法中的Partition函数能够用来在n个数里查找第k大的数字，从而解决面试题29“数组中出现次数超过一半的

数字”和面试题 30 “最小的 k 个数”。归并排序算法能够实现在  $O(n \log n)$  时间统计  $n$  个数字中的逆序对数目（面试题 36 “数组中的逆序对”）。

在本书的第 5 章“优化时间空间效率”中，我们将详细讨论如何从时间效率和空间效率两方面去做优化。

## 5. 优秀的综合能力

在面试过程中，应聘者除了展示自己的编程能力和技术功底之外，还需要展示自己的软技能（Soft Skills），诸如自己的沟通能力和学习能力。随着软件系统的规模越来越大，软件开发已经告别了单打独斗的年代，程序员与他人的沟通变得越来越重要。在面试过程中，面试官会观察应聘者在介绍项目经验或者算法思路时是否观点明确、逻辑清晰，并以此判断其沟通能力的强弱。另外，面试官也会从应聘者说话的神态和语气来判断他是否有团队合作的意识。通常面试官不会喜欢高傲或者轻视合作者的人。

IT 行业知识更新很快，因此程序员只有具备很好的学习能力才能跟上知识更替的步伐。通常面试官有两种办法考查应聘者的学习能力。面试官的第一种方法是询问应聘者最近在看什么书、从中学到了哪些新技术。面试官可以用这个问题了解应聘者的学习愿望和学习能力。面试官的第二种方法是抛出一个新概念，接下来他会观察应聘者能不能在较短时间内理解这个新概念并解决相关的问题。比如面试官要求应聘者计算第 1500 个丑数。很多人都没有听说过丑数这个概念。这个时候面试官就会观察应聘者面对丑数这个新概念时，能不能经过提问、思考、再提问的过程，最终找出丑数的规律从而找到解决方案（详见面试题 34 “丑数”）。

知识迁移能力是一种特殊的学习能力。如果我们能够把已经掌握的知识迁移到其他领域，那么学习新技术或者解决新问题就会变得容易。面试官经常会先问一个简单的问题，再问一个很复杂但和前面的简单问题相关的问题。这个时候面试官期待应聘者能够从简单问题中得到启示，从而找到解决复杂问题的窍门。比如面试官先要求应聘者写一个函数求斐波那契数列，再问一个青蛙跳台阶的问题：一只青蛙一次可以跳上 1 级台阶，也可以跳上 2 级台阶。请问这只青蛙跳上  $n$  级台阶总共有多少种跳法。应聘者如果具有较强的知识迁移能力，就能分析出青蛙跳台阶问题实质上只是斐波那契数列的一个应用（详见面试题 9 “斐波那契数列”）。

还有不少面试官喜欢考查应聘者的抽象建模能力和发散思维能力。面试官从日常生活中提炼出问题，比如面试题 44 “扑克牌中的顺子”，考查应聘者能不能把问题抽象出来用合理的数据结构表示，并找到其中的规律解决这个问题。面试官也可以限制应聘者不得使用常规方法，这要求应聘者具备创新精神，能够打开思路从多角度去分析、解决问题。比如在面试题 47 “不用加减乘除做加法”中，面试官期待应聘者能够打开思路，用位运算实现整数的加法。

我们将在本书的第 6 章“面试中的各项能力”中用具体的面试题详细讨论上述能力在面试中的重要作用。

### 1.3.3 应聘者提问环节

在结束面试前的 5~10 分钟，面试官会给应聘者机会问几个问题，应聘者的问题的质量对面试的结果也有一定的影响。有些人的沟通能力很强，马上就能想到有意思的问题。但对于大多数人而言，在经受了面试官将近一小时的拷问之后可能已经精疲力竭，再迅速想出几个问题难度很大。因此建议应聘者不妨在面试之前做些功课，为每一轮面试准备 2~3 个问题，这样到提问环节的时候就游刃有余了。

面试官让应聘者问几个问题，主要是想了解他最关心的问题有哪些，因此应聘者至少要问一两个问题，否则面试官就会觉得你对我们公司、职位等都不感兴趣，那你来面试做什么？但是也不是什么问题都可以在这个时候问。如果问题问得比较合适，对应聘者来说是个加分的好机会；但如果问的问题不太合适，面试官对他的印象就会大打折扣。

有些问题是不适合在技术面试这个环节里问的。首先是不要问和自己的职位没有关系的问题，比如问“公司未来五年的发展战略是什么”。如果应聘的职位是 CTO，而面试官是 CEO，这倒是个合适的问题。如果应聘的只是一线开发的职位，那这个问题离我们就太远了，与我们的切身利益没有多少关系。另外，坐在对面的面试官很有可能只是一个在一线开发的程序员，他该怎么回答这个关系公司发展战略的问题呢？

其次是不要问薪水。技术面试不是谈薪水的时候，要谈工资要等通过面试之后和 HR 谈。而且让面试官觉得你最关心的问题就是薪水，给面试官留下的印象也不好。

再次是不要立即打听面试结果，比如问“您觉得我能拿到 Offer 吗”之类的问题。现在大部分公司的面试都有好几轮，最终决定应聘者能不能通过面试，是要把所有面试官的评价综合起来的。问这个问题相当于白问，因为问了面试官也不可能告诉应聘者结果，还会让面试官觉得他没有自我评估的能力。

最后推荐问的问题是与招聘的职位或者项目相关的问题。如果这种类型的问题问得很到位，那么面试官就会觉得你对应聘的职位很有兴趣。不过要问好这种类型的问题也不容易，因为首先对应聘的职位或者项目的背景要有一定的了解。我们可以从两方面去了解相关的信息：一是面试前做足功课，到网上去收集一些相关的信息，做到对公司成立时间、主要业务、职位要求等都了然于胸；二是面试过程中留心面试官说过的话。有不少面试官在面试之前会简单介绍与招聘职位相关的项目，其中会包含其他渠道无法得到的信息，比如项目进展情况等。应聘者可以从中找出一两个点，然后向面试官提问。

下面的例子是笔者去思科面试时问的几个问题。一个面试官介绍项目时说这次招聘是项目组第一次在中国招人，目前这个项目所有人员都在美国总部。这轮面试笔者最后问的问题是：这个项目所有的老员工都在美国，那怎么对中国这一批新员工进行培训？中国的新员工有没有机会去美国总部学习？最后一轮面试是老板面试，她介绍说正在招聘的项目组负责开发一个测试系统，思科用它来测试供应商生产的网络设备。这一轮笔者问的几个问题是：这个组是做测试系统的，那这个组的人员是不是也要参与网络设备的测试？是不是需要学习硬件测试相关的知识？因为我们测试的对象是网络设备，那么这个职位对网络硬件的掌握程度有没有要求？

## 1.4

### 本章小结

本章重点介绍了面试的流程。通常面试是从电话面试开始的。接下来可能有一两轮共享桌面远程面试，面试官通过桌面共享软件远程考查应聘者的编程和调试能力。如果应聘者的表演足够优秀，那么公司将邀请他到公司去接受现场面试。

一般每一轮面试都有三个环节。首先是行为面试环节，面试官在这一环节中对照简历询问应聘者的项目经验和掌握的技能。接下来就是技术面试环节，这是面试的重头戏。在这一环节里，面试官除了关注应聘者的编程能力和技术功底之外，还会注意考查他的沟通能力和学习能力。在面试的最后通常面试官会留几分钟时间让应聘者问几个他感兴趣的问题。

本章的 1.3.2 节是全书的大纲。本节介绍了面试官关注应聘者 5 个方面的素质：基础知识是否扎实、能否写出高质量的代码、思路是否清晰、是否有优化效率的能力，以及包括学习能力、沟通能力在内的综合素质是否优秀。在接下来的第 2 章到第 6 章中我们将一一深入探讨这 5 个方面的素质。

## 第 2 章

# 面试需要的基础知识

## 2.1

### 面试官谈基础知识

“C++的基础知识，如面向对象的特性、构造函数、析构函数、动态绑定等，能够反映出应聘者是否善于把握问题本质，有没有耐心深入一个问题。另外还有常用的设计模式、UML 图等，这些都能体现应聘者是否有软件工程方面的经验。”

——王海波（Autodesk，软件工程师）

“对基础知识的考查我特别重视 C++中对内存的使用管理。我觉得内存管理是 C++程序员特别要注意的，因为内存的使用和管理会影响程序的效率和稳定性。”

——蓝诚（Autodesk，软件工程师）

“基础知识反映了一个人的基本能力和基础素质，是以后工作中最核心的能力要求。我一般考查：（1）数据结构和算法；（2）编程能力；（3）部分数学知识，如概率；（4）问题的分析和推理能力。”

——张晓禹（百度，技术经理）

“我比较重视四块基础知识：（1）编程基本功（特别喜欢字符串处理这一类的问题）；（2）并发控制；（3）算法、复杂度；（4）语言的基本概念。”

——张珺（百度，高级软件工程师）

“我会考查编程基础、计算机系统基础知识、算法以及设计能力。这些是一个软件工程师的最基本的东西，这些方面表现出色的人，我们一般认为是有发展潜力的。”

——韩伟东（盛大，高级研究员）

“（1）对OS的理解程度。这些知识对于工作中常遇到的内存管理、文件操作、程序性能、多线程、程序安全等有重要帮助。对于OS理解比较深入的人对于偏底层的工作上手一般比较快。（2）对于一门编程语言的掌握程度。一个热爱编程的人应该会对某种语言有比较深入的了解。通常这样的人对于新的编程语言上手也比较快，而且理解比较深入。（3）常用的算法和数据结构。不了解这些的程序员基本只能写写‘Hello World’。”

——陈黎明（微软，SDE II）

## 2.2

### 编程语言

程序员写代码总是基于某一种编程语言，因此技术面试的时候直接或者间接都会涉及至少一种编程语言。在面试的过程中，面试官要么直接问语言的语法，要么让应聘者用一种编程语言写代码解决一个问题，通过写出的代码来判断应聘者对他使用的语言的掌握程度。现在流行的编程语言很多，不同公司开发用的语言也不尽相同。做底层开发比如经常写驱动的人更习惯用 C，Linux 下有很多程序员用 C++ 开发应用程序，基于 Windows 的 C# 项目已经越来越多，跨平台开发的程序员则可能更喜欢 Java，随着苹果 iPad、iPhone 的热销已经有很多程序员投向了 Objective C 的阵营，同时还有很多人喜欢用脚本语言如 Perl、Python 开发短小精致的小应用软件。因此，不同公司面试的时候对编程语言的要求也有所不同。每一种编程语言都可以写出一本大部头的书籍，本书限于篇幅不可能面面俱到。本书中所有代码都用 C/C++/C# 实现，下面简要介绍一些 C++/C# 常见的面试题。

#### 2.2.1 C++

国内绝大部分高校都开设 C++ 的课程，因此绝大部分程序员都学过 C++，于是 C++ 成了各公司面试的首选编程语言。包括 Autodesk 在内的很多公司在面试的时候会有大量的 C++ 的语法题，其他公司虽然不直接面试 C++ 的语法，但面试题要求用 C++ 实现算法。因此总的说来，应聘者不管去什么公司求职，都应该在一定程度上掌握 C++。

通常语言面试有 3 种类型。第一种类型是面试官直接询问应聘者对 C++ 概念的理解。这种类型的问题，面试官特别喜欢了解应聘者对 C++ 关键字的理解程度。例如：在 C++ 中，有哪 4 个与类型转换相关的关键字？这些关键字各有什么特点，应该在什么场合下使用？

在这种类型的题目中，`sizeof` 是经常被问到的一个概念。比如下面的面试片段，就反复出现在各公司的技术面试中。

**面试官：** 定义一个空的类型，里面没有任何成员变量和成员函数。对该类型求 `sizeof`，得到的结果是多少？

**应聘者：** 答案是 1。

面试官：为什么不是 0？

应聘者：空类型的实例中不包含任何信息，本来求 sizeof 应该是 0，但是当我们声明该类型的实例的时候，它必须在内存中占有一定的空间，否则无法使用这些实例。至于占用多少内存，由编译器决定。Visual Studio 中每个空类型的实例占用 1 字节的空间。

面试官：如果在该类型中添加一个构造函数和析构函数，再对该类型求 sizeof，得到的结果又是多少？

应聘者：和前面一样，还是 1。调用构造函数和析构函数只需要知道函数的地址即可，而这些函数的地址只与类型相关，而与类型的实例无关，编译器也不会因为这两个函数而在实例内添加任何额外的信息。

面试官：那如果把析构函数标记为虚函数呢？

应聘者：C++ 的编译器一旦发现一个类型中有虚拟函数，就会为该类型生成虚函数表，并在该类型的每一个实例中添加一个指向虚函数表的指针。在 32 位的机器上，一个指针占 4 字节的空间，因此求 sizeof 得到 4；如果是 64 位的机器，一个指针占 8 字节的空间，因此求 sizeof 则得到 8。

面试 C/C++ 的第二种题型就是面试官拿出事先准备好的代码，让应聘者分析代码的运行结果。这种题型选择的代码通常包含比较复杂微妙的语言特性，这要求应聘者对 C++ 考点有着透彻的理解。即使应聘者对考点有一点点模糊，那么最终他得到的结果和实际运行的结果可能就会差距甚远。

比如面试官递给应聘者一张有如下代码的 A4 打印纸要求他分析编译运行的结果，并提供 3 个选项：A. 编译错误；B. 编译成功，运行时程序崩溃；C. 编译运行正常，输出 10。

```
class A
{
private:
    int value;

public:
    A(int n) { value = n; }
    A(A other) { value = other.value; }

    void Print() { std::cout << value << std::endl; }
};

int _tmain(int argc, _TCHAR* argv[])
{
    A a = 10;
    A b = a;
    b.Print();
```

```
    return 0;
}
```

---

在上述代码中，复制构造函数 A(A other)传入的参数是 A 的一个实例。由于是传值参数，我们把形参复制到实参会调用复制构造函数。因此如果允许复制构造函数传值，就会在复制构造函数内调用复制构造函数，就会形成永无休止的递归调用从而导致栈溢出。因此 C++ 的标准不允许复制构造函数传值参数，在 Visual Studio 和 GCC 中，都将编译出错。要解决这个问题，我们可以把构造函数修改为 A(const A& other)，也就是把传值参数改成常量引用。

第三种题型就是要求应聘者写代码定义一个类型或者实现类型中的成员函数。让应聘者写代码的难度自然比让应聘者分析代码要高不少，因为能想明白的未必就能写得清楚。很多考查 C++ 语法的代码题围绕在构造函数、析构函数及运算符重载。比如面试题 1 “赋值运算符函数”就是一个例子。

为了让大家能顺利地通过 C++ 面试，更重要的是能更好地学习掌握 C++ 这门编程语言，这里推荐几本 C++ 的书，大家可以根据自己的具体情况选择阅读的顺序：

- 《Effective C++》。这本书很适合在面试之前突击 C++。这本书列举了使用 C++ 经常出现的问题及解决这些问题的技巧。该书中提到的问题也是面试官很喜欢问的问题。
- 《C++ Primer》。读完这本书，就会对 C++ 的语法有全面的了解。
- 《Inside C++ Object Model》。这本书有助于我们深入了解 C++ 对象的内部。读懂这本书后很多 C++ 难题，比如前面的 sizeof 的问题、虚函数的调用机制等，都会变得很容易。
- 《The C++ Programming Language》。如果是想全面深入掌握 C++，没有哪本书比这本书更适合的了。

## 面试题 1：赋值运算符函数

题目：如下为类型 CMyString 的声明，请为该类型添加赋值运算符函数。

```
class CMyString
{
public:
```

```

CMyString(char* pData = NULL);
CMyString(const CMyString& str);
~CMyString(void);

private:
    char* m_pData;
};

```

---

当面试官要求应聘者定义一个赋值运算符函数时，他会在检查应聘者写出的代码时关注如下几点：

- 是否把返回值的类型声明为该类型的引用，并在函数结束前返回实例自身的引用（即`*this`）。只有返回一个引用，才可以允许连续赋值。否则如果函数的返回值是`void`，应用该赋值运算符将不能做连续赋值。假设有3个`CMyString`的对象：`str1`、`str2`和`str3`，在程序中语句`str1=str2=str3`将不能通过编译。
- 是否把传入的参数的类型声明为常量引用。如果传入的参数不是引用而是实例，那么从形参到实参会调用一次复制构造函数。把参数声明为引用可以避免这样的无谓消耗，能提高代码的效率。同时，我们在赋值运算符函数内不会改变传入的实例的状态，因此应该为传入的引用参数加上`const`关键字。
- 是否释放实例自身已有的内存。如果我们忘记在分配新内存之前释放自身已有的空间，程序将出现内存泄露。
- 是否判断传入的参数和当前的实例（`*this`）是不是同一个实例。如果是同一个，则不进行赋值操作，直接返回。如果事先不判断就进行赋值，那么在释放实例自身的内存的时候就会导致严重的问题：当`*this`和传入的参数是同一个实例时，那么一旦释放了自身的内存，传入的参数的内存也同时被释放了，因此再也找不到需要赋值的内容了。

### ❖ 经典的解法，适用于初级程序员

当我们完整地考虑了上述4个方面之后，我们可以写出如下的代码：

```

CMyString& CMyString::operator =(const CMyString &str)
{
    if(this == &str)
        return *this;

    delete []m_pData;
    m_pData = NULL;
}

```

## 有关此电子图书的说明

本人由于一些便利条件，可以帮您提供各种中文电子图书资料，且质量均为清晰的 PDF 图片格式，质量要高于网上大量传播的一些超星 PDG 的图书。方便阅读和携带。只要图书不是太新，文学、法律、计算机、人文、经济、医学、工业、学术等方面 的图书，我都可以帮您找到电子版本。所以，当你想要看什么图书时，可以联系我。我的 QQ 是：**89039855**，大家可以在 QQ 上联系我。

此 PDF 文件为本人亲自制作，请各位爱书之人尊重个人劳动，敬请您不要修改此 PDF 文件。因为这些图书都是有版权的，请各位怜惜电子图书资源，不要随意传播，否则，这些资源更难以得到。