**从武侠小说到程序员面试**

2015年 2月 1日 | [评论](http://zh.lucida.me/blog/from-wuxia-to-programmer-interview/" \l "disqus_thread)

作者：[Lucida](http://lucida.me/about)

* 微博：[@peng\_gong](http://www.weibo.com/pegong/)
* 豆瓣：[@figure9](http://www.douban.com/people/figure9/)

原文链接：<http://lucida.me/blog/from-wuxia-to-programmer-interview/>

**概要**

受到[浣花洗剑录](http://www.amazon.cn/gp/product/B00DFTXX20/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00DFTXX20&linkCode=as2&tag=lucida-23)和[射雕英雄传](http://www.amazon.cn/gp/product/B00ES5S4BK/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00ES5S4BK&linkCode=as2&tag=lucida-23)的启发，本文使用武侠小说中的人物和事件来类比并分析当前程序员面试存在的弊端，并尝试给出一个更好的程序员面试方案。

估计你会认为我在胡扯，程序员面试和[浣花洗剑录](http://www.amazon.cn/gp/product/B00DFTXX20/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00DFTXX20&linkCode=as2&tag=lucida-23)怎么能扯上关系，但它们确实有关系，而且是很有意思的关系：

**两个故事**



[古龙](http://zh.wikipedia.org/wiki/%E5%8F%A4%E9%BE%99)的中期作品[浣花洗剑录](http://www.amazon.cn/gp/product/B00DFTXX20/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00DFTXX20&linkCode=as2&tag=lucida-23)讲了一个这样的故事：来自东瀛的白衣人为了追求武道，远渡重洋前往中原向各路高手挑战，见人杀人，见佛杀佛，直到中原第一高手紫衣侯出手才以一招险胜白衣人，但紫衣侯也深受重伤不久而逝，紫衣侯的后人（也就是主角）在紫衣侯师兄的教导和各种奇遇下学得绝世武功，在白衣人第二次到访中原时将白衣人击败。

相对于楚留香系列和陆小凤系列，浣花洗剑录的故事情节并不出彩，但我觉得很有意思的是白衣人初到中原向各路高手挑战下的战书——一截枯枝：

*白袍人目中却又露出不屑之色，突然后退几步，只见剑光一闪，立刻回鞘，拔剑、挥剑、插剑，三个动作一眨眼已完成。等到清平门八弟子定睛去瞧时，他手中已多了段枯枝。原来他方才一拔剑，便已削下这段枯枝。*

*只听他缓缓道：“拿去给你师父瞧瞧！”转身远远走开，坐到树下一方青石上，不言不动，似已入定。*

在场的清平门弟子并没有意识到这段枯枝的特别，但他们的师傅却被这段枯枝震慑住了：

*白三空双眉紧皱，接过枯枝，起先随意瞧了几眼，然后目光突然瞬也不瞬地凝注在那枯枝切口上，竟看得呆住了。*

之后清平门弟子胡不愁设法把这段枯枝送到中原第一高手紫衣侯手里，以激他出手挽救中原武林，紫衣侯一生自视甚高，但也被这段枯枝所折服：

*众人也不知那枯枝究竟有何好看处，紫衣侯为何竟瞧得如此入神，直过了三四盏茶功夫，紫衣侯方自缓缓长叹一声，道：“好高明的剑法！好快速的剑法！好精深的剑法……”*

旁人并不理解一款枯枝有什么好看，紫衣侯解释道是你们的功力不够：

*铃儿却忍不住问道：“难道侯爷只是瞧了瞧这段枯枝便可看出那人剑法的高低不成？”*

*紫衣侯道：“正是！”*

*铃儿道：“从哪里看出来的？”*

*紫衣侯长叹一声，道：“你剑法到了我这样的造诣，便可自这枯枝切口上看出来了。否则我纵然向你解释三天三夜，你也不会懂的。”*

*铃儿怔了怔，苦笑道：“看起来我一辈子也不会懂了。”*

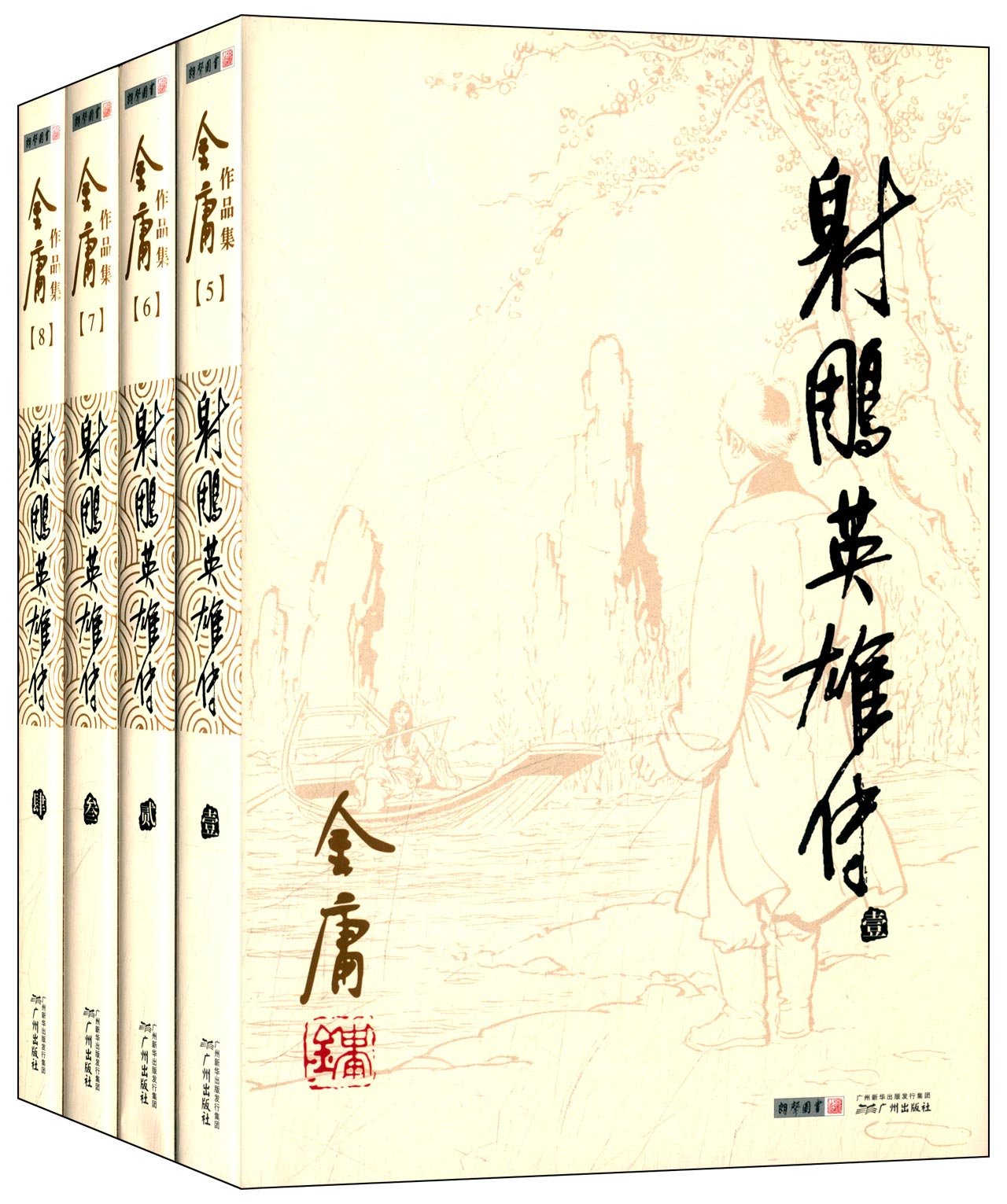
随后紫衣侯以一剑刺中一人七处大穴，并将此人作为战书送至白衣人：

*白衣人道：“这算什么战书？”虽然他能无论见着什么惊奇之事面上都不动声色，但此刻语声中也不免露出诧异之情。*

*王半侠双手一分，撕开了岑陬之衣襟，只见他双肩前胸七道剑痕，伤口早已结疤，骤眼望去，也和寻常伤痕没什么两样，只是这剑痕都在肩井、乳泉等大穴之上，纵横上下，去路分明，剑痕与剑痕之间还有条淡淡的红线，仔细一瞧，亦是剑锋划出来的。白衣人不等王半侠说话，目光立即被这剑痕吸引，脚步也开始移动，一步步走向岑陬面前。*

同样，旁人看不明白伤痕有什么特别之处，但白衣人却异常激动：

*白衣人再也不瞧他一眼，挥起长剑，剑尖向天，微微颤抖。白衣人语声也微微颤抖，仰天道：“天地无极，终于还是有一人能作我的对手……”突然垂首跪下，满头长发四散披落，似是感激苍天终能赐给他一个对手，又似在赞佩苍天之能，竟能造出个能与他作对手的英雄！*



另一个故事来自[金庸](http://zh.wikipedia.org/wiki/%E9%87%91%E5%BA%B8)的[射雕英雄传](http://www.amazon.cn/gp/product/B00ES5S4BK/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00ES5S4BK&linkCode=as2&tag=lucida-23)里郭靖和黄蓉到陆家庄的那一段，裘千丈（但众人以为他是裘千仞）通过嘴冒青烟和肉掌碾砖等“绝技”让众人以为他身怀绝世武功：

**嘴冒青烟**

*陆庄主生怕要是不去，这位发起娇嗔来，非惊动裘千仞不可，当下命庄丁放轻脚步，将自己扶过去，俯眼窗纸，在黄蓉弄破的小孔中向里一张，不禁大奇，只见裘千仞盘膝而坐，双目微闭，嘴里正喷出一缕缕的烟雾，连续不断。*

**肉掌碾砖**

*裘千仞站起身来，走到天井之中，归座时手中已各握了一块砖头。只见他双手也不怎么用劲，却听得格格之声不绝，两块砖头已碎成小块，再捏一阵，碎块都成了粉末，簌簌簌的都掉在桌上。席上四人一齐大惊失色。*

**但“绝技”并非真正的功力，即便是当时初出茅庐的郭靖，一掌就把裘千丈打飞：**

*裘千仞见他左臂扫来，口中却说“吃我一掌”，心道：“你臂中套拳，谁不知道？”双手搂怀，来撞他左臂。哪知郭靖这招“龙战于野”是降龙十八掌中十分奥妙的功夫，左臂右掌，均是可实可虚，非拘一格，眼见敌人挡他左臂，右掌忽起，也是蓬的一声，正击在他右臂连胸之处，裘千仞的身子如纸鹞断线般直向门外飞去。*

之后众人才发现这位“裘千仞”是个假冒：嘴冒青烟是把茅点燃藏在袖里吸一口喷一口；肉掌碾砖是用面粉做的砖；而轻功水上飘则是提前在水底打了暗桩。

**程序员面试**

也许你会疑惑这两个故事和程序员面试有什么关系，先拿白衣人来说：

1. 东瀛修炼绝世武功（在校刻苦学习技术）；
2. 远赴中原挑战群雄（即将毕业开始求职）；
3. 拔剑削枯枝作战书（撰写简历进行面试）；
4. 惊动中原第一高手（简历/面试得到赏识）；
5. 海上决战名扬天下（得到Offer搞定工作）。

写到这里，不用说你也知道为什么会提到裘千丈这个金庸小说里略搞笑的人物。对应到程序员，这大概会是一个简历很华丽，面试时非常能侃，可以非常流利的回答一些常见面试问题（因为刷过题库），但实际工作起来却错误百出的人物。

毫无疑问，没有一家公司想招聘裘千丈这样的“高手”，而白衣人这样的真正高手则是任何一家公司都梦寐以求。所以问题来了——如何鉴别一个人是白衣人这样的真正高手，而不是裘千丈这样的“高手”呢？

**你一辈子也不会懂**

[浣花洗剑录](http://www.amazon.cn/gp/product/B00DFTXX20/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00DFTXX20&linkCode=as2&tag=lucida-23)中有这样一个细节：

*铃儿却忍不住问道：“难道侯爷只是瞧了瞧这段枯枝便可看出那人剑法的高低不成？”*

*紫衣侯道：“正是！”*

*铃儿道：“从哪里看出来的？”*

*紫衣侯长叹一声，道：“你剑法到了我这样的造诣，便可自这枯枝切口上看出来了。否则我纵然向你解释三天三夜，你也不会懂的。”*

*铃儿怔了怔，苦笑道：“看起来我一辈子也不会懂了。”*

据我了解，一些公司把程序员招聘的决定权交给HR，这无疑是最蠢的决定——HR和猎头可以确定程序员的背景，并通过求职者的以往经历来推测程序员的能力，但就像铃儿看不出枯枝的奥妙，HR和猎头无法鉴别程序员的能力（除非他们以前也是优秀的程序员）。鉴别程序员能力这项工作，还是留给程序员最为适合。而且优秀的程序员往往需要至少同样优秀的程序员去发掘。

但即便是程序员自己去面试程序员也依然存在问题，就像裘千丈在射雕英雄传里面糊弄群雄一样。

**轻功水上漂**

裘千丈在射雕英雄传里先后“表演”了水上漂、嘴冒青烟、指划酒杯和肉掌碾砖这些“绝技”，在场的众人（其中不乏陆冠英这样的高手）却没有一个人识破，如果不是郭靖这个二货傻乎乎的冲上去比划，恐怕众人还会被继续糊弄下去。

回到程序员面试，大多数笔试/面试题目都可以在网上找到，而一些公司在招聘时为了省事甚至直接到网上搜题，这就导致看似很高的程序员面试门槛实际变的很低——得到一份还不错的工作并不需要花一两年系统的学习计算机技术，而只需一两个月到leetcode、CareerCup以及未名求职版刷题目。原本很有区分度的算法题目也变的毫无价值——谁知道你是自己想出来的还是背出来的。就像轻功水上漂，谁知道你是真的功力深厚，还是提前在水底打了暗桩。

所以算法题目是一个很尴尬的存在——为了考察程序员的水平，不可能不考算法题目，但一旦考算法题目，求职者就可以通过背题的方式答题从而使得考察变的毫无意义。面试者接下来会找更难的题目，但相对于面试题的数量无法与求职者的数量相比，所以最后还是会陷入这种出题——背题的恶性循环，这个恶性循环的直接后果就是公司招进来一票“裘千丈”，而一些水平不错但没有背题目的程序员却被拒之门外。

那是不是就没有办法了呢？我不这么认为，让我们回到浣花洗剑录的那段枯枝：

**枯枝**

在浣花洗剑录里，白衣人远赴中原挑战群雄，他并没有表演水上漂或是嘴冒青烟这种外表华丽的“绝技”，而只是削下一段枯枝作为战书。而这段在众人眼中平淡无奇的枯枝却震慑了中原第一高手紫衣侯：

*众人也不知那枯枝究竟有何好看处，紫衣侯为何竟瞧得如此入神，直过了三四盏茶功夫，紫衣侯方自缓缓长叹一声，道：“好高明的剑法！好快速的剑法！好精深的剑法……”*

重剑无锋，大巧不工。程序设计也是如此。程序设计能力并不一定需要通过复杂算法才能体现。程序员面试需要考察深度，这里的深度是程序员对程序设计以及编程语言的理解，也是其在多年编程经验中得到的感悟。

这么说还是很玄，所以我在这里举一个实例：

恐怕这道题会是你见过的最简单的面试题——**使用C语言把字母转换成大写，不能使用库函数**。

以至于很多面试者听到这道题时的第一反应都是：



但我并没有打算开玩笑，你可以试着用C写一个大写转换，然后继续阅读本文。

比较有意思的是，一部分面试者给出了类似这样的答案：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <stdio.h>  int main() {  char c = 'a';  printf("a的大写是%c\n", c - 32);  return 0;  } |

其实要是写成这样也就没有往下问的必要了 –\_–#

当然不少面试者还是比较靠谱：

|  |  |
| --- | --- |
| 1  2  3 | char daxie(char c) {  return c - 32;  } |

这时我会建议面试者不要使用拼音命名，并会提示如果输入的字母不是小写程序会怎么样，一般来说面试者都会在这时引入范围检查，但有些人会写成这样：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | char to\_upper(char c) {  if (c >= 'a' && c <= 'z') {  return c - 32;  } else {  printf('Input error!');  return 0;  }  } |

如果要写成这样也没有往下问的必要了（个人怀疑是看谭浩强学的C） –\_–#

相对靠谱的那部分面试者会给出这样的答案：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | char to\_upper(char c) {  if (c >= 'a' && c <= 'z') {  return c - 32;  }  return c;  } |

这已经很接近我的及格要求，接下来我会问面试者能不能改善它的可读性（Readability），一些面试者会在命名上下文章（比如把参数c重命名为input）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | char to\_upper(char input) {  int offset = 32;  if (input >= 'a' && input <= 'z') {  return input - offset;  }  return input;  } |

这时我会提示能不能去掉这个诡异的32，一般来说能到这一步的面试者都可以反应过来：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | char to\_upper(char input) {  if (input >= 'a' && input <= 'z') {  return input - 'a' + 'A';  }  return input;  } |

这就是我的及格要求。一般我会提示面试者能不能继续改进可读性，但遗憾的是，到现在也没有一个面试者能在这一步给出我满意的答案：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | char to\_upper(char input) {  if ('a' <= input && input <= 'z') {  return input - 'a' + 'A';  }  return input;  } |

其实就是用'a' <= input && input <= 'z'替代input >= 'a' && input <= 'z'——这个技巧源自于[代码大全](http://www.amazon.cn/gp/product/B0061XKRXA/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B0061XKRXA&linkCode=as2&tag=lucida-23)，[代码大全](http://www.amazon.cn/gp/product/B0061XKRXA/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B0061XKRXA&linkCode=as2&tag=lucida-23)里面专门有一节讲解如何编写可读的布尔表达式。从这里我可以看出这些面试者都没有读过[代码大全](http://www.amazon.cn/gp/product/B0061XKRXA/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B0061XKRXA&linkCode=as2&tag=lucida-23)，考虑到[代码大全](http://www.amazon.cn/gp/product/B0061XKRXA/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B0061XKRXA&linkCode=as2&tag=lucida-23)几乎是程序设计的必读书籍，我可以推断出这些面试者很可能没有阅读习惯，而不阅读的程序员一般都不会太出色。

刚刚提到，到了这一步其实也只是过了及格线而已（如果你能写出可读的布尔表达式，我会在内心提前给你打个优秀），接下来我会询问能不能进一步提升性能，少数面试者在提示下会想到使用数组：

|  |  |
| --- | --- |
| 1  2  3  4 | char to\_upper(char input) {  static char convert\_table[] = { ... };  return convert\_table[input];  } |

如果面试者能提到他是从[C语言标准库](http://www.amazon.cn/gp/product/B00IZW4DK8/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00IZW4DK8&linkCode=as2&tag=lucida-23)里面学到这个技巧，加10分 :–)

有的面试者会想到使用宏：

|  |  |
| --- | --- |
| 1  2  3 | static char convert\_table[] = {...};  #define TO\_UPPER(input) convert\_table[input] |

这时我会询问宏的优点和缺点，以及在这里使用宏会不会有错误。总之就是确定面试者确实理解宏，而不是从哪里（比如[编程之美](http://www.amazon.cn/gp/product/B0016K8XGQ/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B0016K8XGQ&linkCode=as2&tag=lucida-23)之类的面试书籍）背了一个答案出来。

有的面试者会在一开始直接给出使用数组+宏的最优方案（我几乎可以直接确定他背过题目），这时我会要求他给出一个函数+非数组的实现。如果他写不好这个函数，那么依然无法通过。

可能你们以为到这里就完结了，其实还不是，考虑下C语言的EOF（即-1），以及to\_upper的应用场景，下面这段代码会出现什么问题？

|  |  |
| --- | --- |
| 1 | char c = to\_upper(getchar()); |

如果getchar()返回EOF，由于to\_upper接收的类型是char，如果该系统的char是无符号的话，就会出现转换问题，这也是为什么C标准库（ctype.h）中的toupper函数签名是int toupper(int c)而非char toupper(char c)。

接下来，让我们回顾这道简单的题目都考察了哪些点：

1. 函数的概念（而不是写在main里）;
2. 缩进和命名（而不是拼音）；
3. 使用可读的字面量（'a' - 'A'而非32）；
4. API设计（当to\_upper接收到非小写字母字符应该返回什么？0？报错？还是返回原值？考虑到to\_upper的应用场景是把一个字符串中的小写字母转化为大写，返回原值显然更合理）；
5. 是否有阅读习惯（至少可以看出你有没有认真的读过[代码大全](http://www.amazon.cn/gp/product/B0061XKRXA/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B0061XKRXA&linkCode=as2&tag=lucida-23)）；
6. 是否读过C标准库源码（指出toupper数组实现的出处）；
7. 数组的运用（使用转换表）；
8. 了解宏，以及宏的危害（使用宏）；
9. 是否背过这道题（在第一时间给出使用数组+宏的最优方案）；
10. EOF以及C标准库风格。

接下来我还会要求面试者测试这个函数并给出测试代码，这里恕不赘述。

这道题目就很像[浣花洗剑录](http://www.amazon.cn/gp/product/B00DFTXX20/ref=as_li_ss_tl?ie=UTF8&camp=536&creative=3132&creativeASIN=B00DFTXX20&linkCode=as2&tag=lucida-23)里的那段枯枝——它看起来非常简单，但实际并不简单——每个人都能削一段树枝，但削成什么样子就是另一回事；每个程序员都能写出大小写转换，但写到什么程度就是另一回事。

我认为这样的题目才是程序员面试的首选：

1. 它看似十分简单，但做好又非常困难；
2. 它能反映出很多问题——比如转化大小写这道题就至少反映出了10个。
3. 和复杂的算法题目不同，它不会让面试者卡壳（或无从下手），从而避免一些水平经验还不错的程序员被误拒；
4. 它没有标准答案——所以即便面试者把题目放在网络上也不会有丝毫影响，因为面试官的评价标准对面试者不透明；
5. 背题目是没有效果的——从而保证不会招进“裘千丈”这样的应试程序员；

可能有人会问，既然有诸多好处，为什么这些公司依然使用复杂的算法题目作为面试题？

我的答案是，排除对算法的盲目崇拜，因为这样的题目非常难出，而且对面试官的要求又很高，所以绝大多数面试官都选择去网上搜题目而不是自己出题这条捷径。殊不知这条捷径正是人才招聘失败的源泉——优秀的程序员因为没有背题而被拒绝，而水平平平的“裘千丈”们却因为背过题目而被录用，这些录用的“裘千丈”们又会用同样的方式招聘下一批更加糟糕的“裘千丈”，讽刺至级。

**结论**

1. 程序员招聘的决定权应在程序员手里，而不是HR；
2. 优秀的程序员往往需要至少同样优秀的程序员去发现；
3. 复杂的算法题目是一种很糟糕的考察程序员的方式；
4. 面试官应当去自己出题，而不是去网上搜现成的题目；
5. 面试官（以及公司）应该投入大量时间在程序员面试的题目，从而拒绝鱼目混珠，保证招聘质量。

以上。