

《编程之美—微软技术面试心得》



《编程之美—微软技术面试心得》2008年3月28日电子工业出版社首次发行，首印10000册，截至2008年4月28日10000册全部售完。

China-pub、当当网连续6周计算机图书销量排行**第一**。

2008年5月12日第2次印刷，正式上市。

中国象棋将帅问题

下过中国象棋的朋友都知道，双方的“将”和“帅”相隔遥远，并且它们不能照面。在象棋残局中，许多高手能利用这一规则走出精妙的杀招。假设棋盘上只有“将”和“帅”二子（如图 1-3 所示）（为了下面叙述方便，我们约定用 A 表示“将”， B 表示“帅”）：

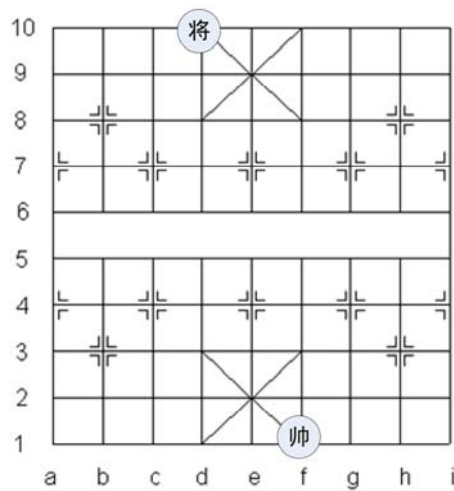


图 1-3

A 、 B 二子被限制在己方 3×3 的格子里运动。例如，在如上的表格里， A 被正方形 $\{d_{10}, f_{10}, d_8, f_8\}$ 包围，而 B 被正方形 $\{d_3, f_3, d_1, f_1\}$ 包围。每一步， A 、 B 分别可以横向或纵向移动一格，但不能沿对角线移动。另外， A 不能面对 B ，也就是说， A 和 B 不能处于同一纵向直线上（比如 A 在 d_{10} 的位置，那么 B 就不能在 d_1 、 d_2 以及 d_3 ）。

请写出一个程序，输出 A 、 B 所有合法位置。要求在代码中只能使用一个变量。

分析与解法

问题的本身并不复杂，只要把所有 A 、 B 互相排斥的条件列举出来就可以完成本题的要求。由于本题要求只能使用一个变量，所以必须首先想清楚在写代码的时候，有哪些信息需要存储，并且尽量高效率地存储信息。稍微思考一下，可以知道这个程序的大体框架是：

遍历 A 的位置

 遍历 B 的位置

 判断 A 、 B 的位置组合是否满足要求。

 如果满足，则输出。

因此，需要存储的是 A 、 B 的位置信息，并且每次循环都要更新。为了能够进行判断，首先需要创建一个逻辑的坐标系，以便检测 A 何时会面对 B 。这里我们想到的方法是用 1~9 的数字，按照行优先的顺序来表示每个格点的位置（如图 1-4 所示）。这样，只需要用模余运算就可以得到当前的列号，从而判断 A 、 B 是否互斥。

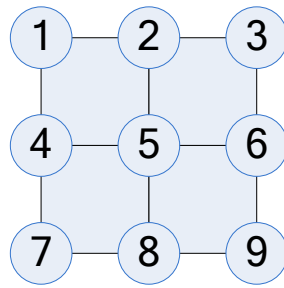


图 1-4 用 1~9 的数字表示 A 、 B 的坐标

第二，题目要求只用一个变量，但是我们却要存储 A 和 B 两个子的位置信息，该怎么办呢？

可以先把已知变量类型列举一下，然后做些分析。

对于 `bool` 类型，估计没有办法做任何扩展了，因为它只能表示 `true` 和 `false` 两个值；而 `byte` 或者 `int` 类型，它们能够表达的信息则更多。事实上，对本题来说，每个子都只需要 9 个数字就可以表达它的全部位置。

一个 8 位的 `byte` 类型能够表达 $2^8=256$ 个值，所以用它来表示 A 、 B 的位置信息绰绰有余，因此可以把这个字节的变量（设为 b ）分成两部分。用前面的 4 bit 表示 A 的位置，用后面的 4 bit 表示 B 的位置，那么 4 个 bit 可以表示 16 个数，这已经足够了。

问题在于：如何使用 bit 级的运算将数据从这一 `byte` 变量的左边和右边分别存入和读出。

下面是做法：

- 将 byte b (10100101) 的右边 4 bit (0101) 设为 n (0011)：

首先清除 b 右边的 bits，同时保持左边的 bits:

```
11110000 ( LMASK )
& 10100101 ( b )
-----
10100000
```

然后将上一步得到的结果与 n 做或运算

```
10100000 ( LMASK & b )
^ 00000011 ( n )
-----
10100011
```

- 将 byte b (10100101) 左边的 4 bit (1010) 设为 n (0011)：

首先，清除 b 左边的 bits，同时保持右边的 bits:

```
00001111 ( RMASK )
& 10100101 ( b )
-----
00000101
```

现在，把 n 移动到 byte 数据的左边

```
 $n \ll 4 = 00110000$ 
```

然后对以上两步得到的结果做或运算，从而得到最终结果。

```
00000101 ( RMASK & b )
^ 00110000 (  $n \ll 4$  )
-----
00110101
```

- 得到 byte 数据的右边 4 bits 或左边 4 bits (e.g. 10100101 中的 1010 以及 0101) :

清除 b 左边的 bits , 同时保持右边的 bits

```
00001111 ( RMASK )
& 10100101 (  $b$  )
```

```
-----
00000101
```

清除 b 的右边的 bits , 同时保持左边的 bits

```
11110000 ( LMASK )
& 10100101 (  $b$  )
```

```
-----
10100000
```

将结果右移 4 bits

```
10100000 >> 4 = 00000101
```

最后的挑战是如何在不声明其他变量约束的前提下创建一个 for 循环。可以重复利用 1byte 的存储单元，把它作为循环计数器并用前面提到的存取和读入技术进行操作。还可以用宏来抽象化代码，例如：

```
for (LSET(b, 1); LGET(b) <= GRIDW * GRIDW; LSET(b, (LGET(b) + 1)))
```

【解法一】

代码清单 1-6

```
#define HALF_BITS_LENGTH 4
// 这个值是记忆存储单元长度的一半，在这道题里是4bit
#define FULLMASK 255
// 这个数字表示一个全部bit的mask，在二进制表示中，它是11111111。
#define LMASK (FULLMASK << HALF_BITS_LENGTH)
// 这个宏表示左bits的mask，在二进制表示中，它是11110000。
#define RMASK (FULLMASK >> HALF_BITS_LENGTH)
// 这个数字表示右bits的mask，在二进制表示中，它表示00001111。
#define RSET(b, n) (b = ((LMASK & b) ^ n))
// 这个宏，将b的右边设置成n
#define LSET(b, n) (b = ((RMASK & b) ^ (n << HALF_BITS_LENGTH)))
// 这个宏，将b的左边设置成n
#define RGET(b) (RMASK & b)
// 这个宏得到b的右边的值
#define LGET(b) ((LMASK & b) >> HALF_BITS_LENGTH)
// 这个宏得到b的左边的值
#define GRIDW 3
// 这个数字表示将帅移动范围的行宽度。
#include <stdio.h>
#define HALF_BITS_LENGTH 4
```

```

#define FULLMASK 255
#define LMASK (FULLMASK << HALF_BITS_LENGTH)
#define RMASK (FULLMASK >> HALF_BITS_LENGTH)
#define RSET(b, n) (b = ((LMASK & b) ^ n))
#define LSET(b, n) (b = ((RMASK & b) ^ (n << HALF_BITS_LENGTH)))
#define RGET(b) (RMASK & b)
#define LGET(b) ((LMASK & b) >> HALF_BITS_LENGTH)
#define GRIDW 3

int main()
{
    unsigned char b;
    for(LSET(b, 1); LGET(b) <= GRIDW * GRIDW; LSET(b, (LGET(b) + 1)))
        for(RSET(b, 1); RGET(b) <= GRIDW * GRIDW; RSET(b, (RGET(b) + 1)))
            if(LGET(b) % GRIDW != RGET(b) % GRIDW)
                printf("A = %d, B = %d\n", LGET(b), RGET(b));

return 0;
}

```

【输出】

格子位置用 N 来表示， $N = 1, 2, \dots, 8, 9$ ，依照行优先的顺序，如图 1-5 所示：

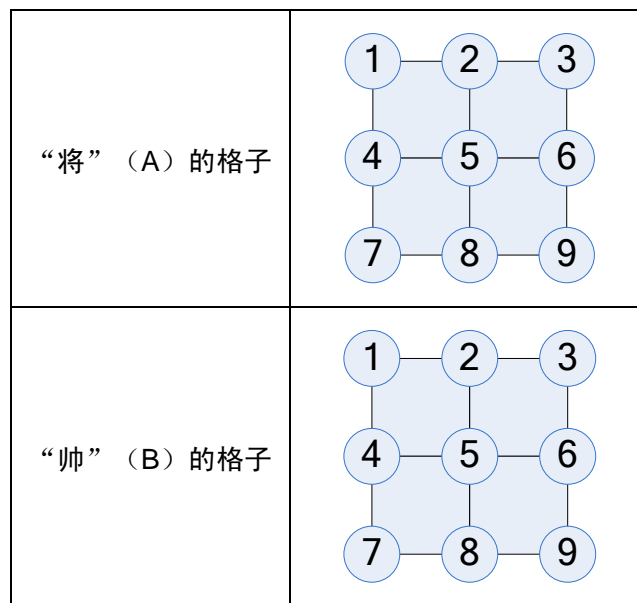


图 1-5

$A = 1, B = 2$	$A = 4, B = 2$	$A = 7, B = 2$
$A = 1, B = 3$	$A = 4, B = 3$	$A = 7, B = 3$
$A = 1, B = 5$	$A = 4, B = 5$	$A = 7, B = 5$
$A = 1, B = 6$	$A = 4, B = 6$	$A = 7, B = 6$
$A = 1, B = 8$	$A = 4, B = 8$	$A = 7, B = 8$
$A = 1, B = 9$	$A = 4, B = 9$	$A = 7, B = 9$
$A = 2, B = 1$	$A = 5, B = 1$	$A = 8, B = 1$
$A = 2, B = 3$	$A = 5, B = 3$	$A = 8, B = 3$
$A = 2, B = 4$	$A = 5, B = 4$	$A = 8, B = 4$
$A = 2, B = 6$	$A = 5, B = 6$	$A = 8, B = 6$
$A = 2, B = 7$	$A = 5, B = 7$	$A = 8, B = 7$
$A = 2, B = 9$	$A = 5, B = 9$	$A = 8, B = 9$
$A = 3, B = 1$	$A = 6, B = 1$	$A = 9, B = 1$
$A = 3, B = 2$	$A = 6, B = 2$	$A = 9, B = 2$
$A = 3, B = 4$	$A = 6, B = 4$	$A = 9, B = 4$
$A = 3, B = 5$	$A = 6, B = 5$	$A = 9, B = 5$
$A = 3, B = 7$	$A = 6, B = 7$	$A = 9, B = 7$
$A = 3, B = 8$	$A = 6, B = 8$	$A = 9, B = 8$

考虑了这么多因素，总算得到了本题的一个解法，但是 MSRA 里却有人说，下面的一小段代码也能达到同样的目的：

```
BYTE i = 81;
while(i--)
{
    if(i / 9 % 3 == i % 9 % 3)
        continue;
    printf("A = %d, B = %d\n", i / 9 + 1, i % 9 + 1);
}
```

但是很快又有另一个人说他的解法才是效率最高的：

代码清单 1-7

```
struct {
    unsigned char a:4;
    unsigned char b:4;
} i;

for(i.a = 1; i.a <= 9; i.a++)
for(i.b = 1; i.b <= 9; i.b++)
    if(i.a % 3 == i.b % 3)
        printf("A = %d, B = %d\n", i.a, i.b);
```

读者能自己证明一下么？¹

¹ 这一题目由微软亚洲研究院工程师 Matt Scott 提供，他在学习中国象棋的时候想出了这个题目，后来一位应聘者给出了比他的“正解”简明很多的答案，他们现在成了同事。

《移山之道——VSTS 软件开发指南》



《编程之美——微软技术面试心得》作者之一邹欣 力作。

第一本由微软 Visual Studio Team System 一线开发人员所写的原创精品！

第一本直接针对中国软件开发人员写的循序渐进、基于案例的教材！

第一本通过微软自身的实践直接介绍微软开发流程（MSF）的实用手册！

China-pub 地址：<http://www.china-pub.com/35373>

读者评论：

全书内容写的很不错，以一种别具匠心的风格展示软件生命周期各个阶段管理以及应用。

内容也很简洁没有那么多的理论，比较适合要步入 PM 或者其他技术人员学习参考，尤其是里面的一些看似很短的对话却体现了一个很深刻的道理。希望作者能多写些这方面的著作。支持原创作品。 读者： thirston_bill

其实里面讲 VSTS 部分并不比我们公司内部讲师讲的详细，但作者说明了为什么要这么用，另外作者让我弄明白了什么是 MSF，也让我有马上将自己项目移至 VSTS 服务器上去的冲动。

目前项目还是处于很松弛的管理管理状态，代码管理也不正规，而该书还提出了一个在微软敏捷 4 之上的新的方法——移山方法。我就打算用这个管理我自己的项目。 读者： kiciro

因为在书店中看了一看这书，感觉很不错，所以买了它。

当然我买这本书的目的并不是在学习 VSTS 的用法，因为我觉得在这本书中有很多的技巧与道理，出于学习这些技巧与道理的角度，我觉得不管你用不用 VSTS 都可以来看看这本书。 读者： ooyuan

经典的一本项目管理之道：故事化的情节，实战型的经历，让你不知不觉中了解整个项目开发的过程，就当看小说一样，只要你能翻到书的最后一页，你就受益非浅。 豆瓣读者

博文视点好书集锦



《Web设计创新思维》

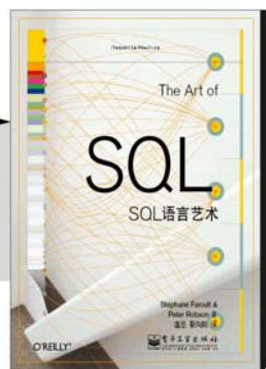
Cameron Adams, Mark Boulton, Andy Clarke, Simon Collison, Jeff Croft, Derek Featherstone, Ian Lloyd, Ethan Marcotte, Dan Rubin, Rob Weychert 著
夏伟频 译

- 10大世界级Web设计高手打造
- 全彩印刷，让你的网站更动态、吸引人

《SQL语言艺术》

Stéphane Faroult, Peter Robson 著
温昱 靳向阳 译

- 2008最奇缺数据库书：SQL性能与调校
- 借鉴《孙子兵法》智慧，传授 25年SQL性能与调校经验



《JavaScript语言精髓与编程实践》

周爱民 著

- 畅销书《大道至简》作者周爱民又一力作
- Web2.0狂潮中最耀眼的动态语言

《LoadRunner性能测试实战》

陈绍英 刘建华 金成姬 著

- 2007“金牛奖”原创图书大奖
- 国内原创第一本LoadRunner性能测试实战之书

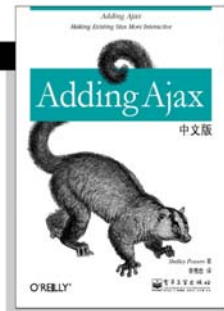




《构建可扩展的Web站点》

Cal Henderson 著
徐宁 译

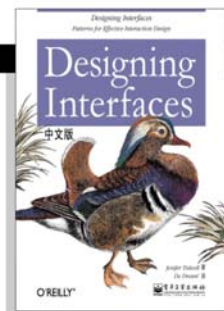
- 著名Web 2.0代表网站www.flickr.com架构师/工程经理 Cal Henderson倾力之作
- 助您构建最优秀的Web 2.0应用



《Adding Ajax中文版》

Shelley Powers 著
李秀忠 译

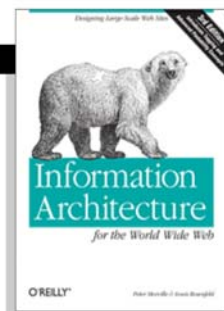
- 全新Web2.0视角，完美RIA体验
- 全面阐述在既有站点中实现Ajax的原则、方法、技巧与实践



《Designing Interfaces 中文版》

Jenifer Tidwell 著
De Dream' 译

- 业界大牛，经典力作
- 全面体现2008 技术热点：交互设计、用户体验



《Information Architecture for the World Wide Web (第3版)》

Peter Morville, Louis Rosenfeld 著

- 经典巨著最新第3版
- 教你设计大规模Web站点