

```

4      for (i = 0; i < n; i++) {
5          if (a[i] > b[i]) {
6              long t = a[i];
7              a[i] = b[i];
8              b[i] = t;
9          }
10     }
11 }

```

在随机数据上测试这个函数，得到的 CPE 大约为 13.50，而对于可预测的数据，CPE 为 2.5~3.5，其预测错误惩罚约为 20 个周期。

用功能式的风格实现这个函数是计算每个位置 i 的最大值和最小值，然后将这些值分别赋给 $a[i]$ 和 $b[i]$ ：


```

1  /* Rearrange two vectors so that for each i, b[i] >= a[i] */
2  void minmax2(long a[], long b[], long n) {
3      long i;
4      for (i = 0; i < n; i++) {
5          long min = a[i] < b[i] ? a[i] : b[i];
6          long max = a[i] < b[i] ? b[i] : a[i];
7          a[i] = min;
8          b[i] = max;
9      }
10 }

```

对这个函数的测试表明无论数据是任意的，还是可预测的，CPE 都大约为 4.0。（我们还检查了产生的汇编代码，确认它确实使用了条件传送。）

在 3.6.6 节中讨论过，不是所有的条件行为都能用条件数据传送来实现，所以无可避免地在某些情况中，程序员不能避免写出会导致条件分支的代码，而对于这些条件分支，处理器用分支预测可能会处理得很糟糕。但是，正如我们讲过的，程序员方面用一点点聪明，有时就能使代码更容易被翻译成条件数据传送。这需要一些试验，写出函数的不同版本，然后检查产生的汇编代码，并测试性能。

 **练习题 5.9** 对于归并排序的合并步骤的传统的实现需要三个循环[98]：

```

1  void merge(long src1[], long src2[], long dest[], long n) {
2      long i1 = 0;
3      long i2 = 0;
4      long id = 0;
5      while (i1 < n && i2 < n) {
6          if (src1[i1] < src2[i2])
7              dest[id++] = src1[i1++];
8          else
9              dest[id++] = src2[i2++];
10     }
11     while (i1 < n)
12         dest[id++] = src1[i1++];
13     while (i2 < n)
14         dest[id++] = src2[i2++];
15 }

```

对于把变量 $i1$ 和 $i2$ 与 n 做比较导致的分支，有很好的预测性能——唯一的预测错误