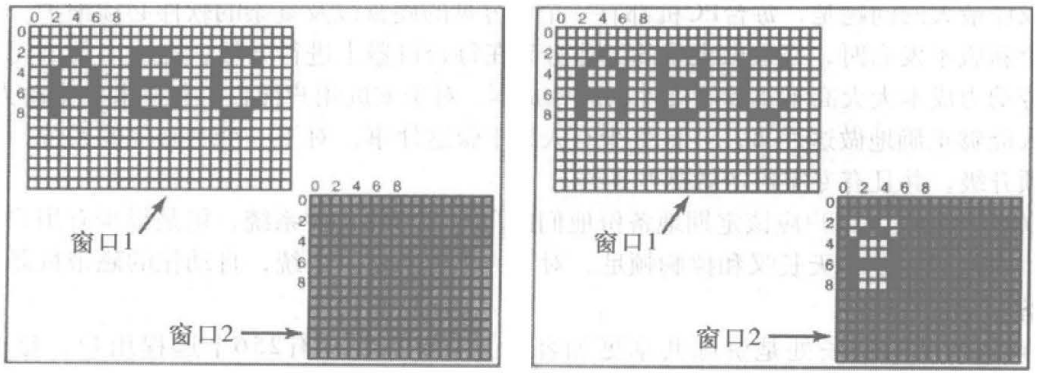

a) b)

图5-42 使用BitBlt复制位图：a) 复制前；b) 复制后

由于这样的缘故，Windows还支持一个称为DIB（Device Independent Bitmap，设备无关的位图）的数据结构。采用这种格式的文件使用扩展名.bmp。这些文件在像素之前具有文件与信息头以及一个颜色表，这样的信息使得在不同的设备之间移动位图十分容易。

5. 字体

在Windows 3.1版之前的版本中，字符表示为位图，并且使用BitBlt复制到屏幕上或者打印机上。这样做的问题是，正如我们刚刚看到的，在屏幕上有意义的位图对于打印机来说太小了。此外，对于每一尺寸的每个字符，需要不同的位图。换句话说，给定字符A的10点阵字型的位图，没有办法计算它的12点阵字型。因为每种字体的每一个字符可能都需要从4点到120点范围内的各种尺寸，所以需要的位图的数目是巨大的。整个系统对于文本来说简直是太笨重了。

图5-43 不同点阵尺寸的字符轮廓的一些例子

该问题的解决办法是TrueType字体的引入，TrueType字体不是位图而是字符的轮廓。每个TrueType字符是通过围绕其周界的一系列点来定义的，所有的点都是相对于（0，0）原点。使用这一系统，放大或者缩小字符是十分容易的，必须要做的全部事情只是将每个坐标乘以相同的比例因子。采用这种方法，TrueType字符可以放大或者缩小到任意的点阵尺寸，甚至是分数点阵尺寸。一旦给定了适当的尺寸，各个点可以使用幼儿园教的著名的逐点连算法连接起来（注意现代幼儿园为了更加光滑的结果而使用曲线尺）。轮廓完成之后，就可以填充字符了。图5-43给出了某些字符缩放到三种不同点阵尺寸的一个例子。

一旦填充的字符在数学形式上是可用的，就可以对它进行栅格化，也就是说，以任何期望的分辨率将其转换成位图。通过首先缩放然后栅格化，我们可以肯定显示在屏幕上的字符与出现在打印机上的字符将是尽可能接近的，差别只在于量化误差。为了进一步改进质量，可以在每个字符中嵌入表明如何进行栅格化的线索。例如，字母T顶端的两个衬线应该是完全相同的，否则由于舍入误差可能就不是这样的情况了。

5.7 瘦客户机

多年来，主流计算范式一直在中心化计算和分散化计算之间振荡。最早的计算机（例如ENIAC）虽然是庞然大物，但实际上是个人计算机，因为一次只有一个人能够使用它。然后出现的是分时系统，在分时系统中许多远程用户在简单的终端上共享一个大型的中心计算机。接下来是PC时代，在这一阶段用户再次拥有他们自己的个人计算机。

虽然分散化的PC模型具有长处，但是它也有着某些严重的不利之处，人们刚刚开始认真思考这些