也就是说，使用由多个字符构成的长名字也是可以的。甚至可以说，写成这样的情况更加普遍。

下面我们就举一个例子作为证据来证明程序是指令和数据的集合。请诸位看代码清单 1.1。这里列出了一段用名为 C 语言的编程语言编写的程序。C 语言中要在每条指令的末尾写一个分号“;”。第一行的“int a, b, c;”表示接下来要使用名为 a、b、c 的整数变量，其中 int 是 integer（整数）的缩写，用于告诉计算机“要用的是整数”。下一行的“a = 10”表示把整数 10 赋值给变量 a。同样地，“b = 20;”表示把整数 20 赋值给变量 b。等号“=”是赋值给变量的指令。再来看最后一行的“c = Average(a, b);”，这一行表示把变量 a 和 b 传给函数的参数，并将运算结果赋值给变量 c。其中使用了一个名为 Average 的神秘函数，它的作用是返回两个参数的平均值。通过上面这个例子，诸位就应该能明白程序确实只是由指令和数据构成的了吧。

代码清单 1.1　C 语言的程序示例片段

```
int a, b ,c;
a = 10;
b = 20;
c = Average(a, b);
```

虽然程序就是这样，但是那些稍微有些编程经验的人也许会说：代码清单 1.1 所示的程序逻辑简单，而真正的程序是使用了各种各样的语法、比这复杂得多得多的东西，绝不是用指令和数据的集合就能解释清楚的。其实并不是像他们想的那样，无论是多么复杂的程序，都只不过是指令和数据的集合。下面我们再拿出一个证据。

在一般的编程过程中，都要先编译再执行。所谓编译就是把用 C 语言等编程语言编写的文件（源文件）转换成用机器语言（原生代码）编写的文件。假设我们先把代码清单 1.1 中的代码保存到文件 MyProg.c 中，

然后经过编译就可以生成可执行的程序文件 MyProg.exe 了。接下来使用能查看文件内容的工具查看 MyProg.exe，其内容应该与代码清单 1.2 类似。可以看到里面仅仅是数值的罗列（这里用十六进制数表示）。

代码清单 1.2　机器语言的程序示例

```
C7 45 FC 01 00 00 00 C7 45 F8 02 00 00 00 8B 45
F8 50 8B 4D FC 51 E8 82 FF FF FF 83 C4 08 89 45
F4 8B 55 F4 52 68 1C 30 42 00 E8 B9 03 00 00 83
```

请选择一个代码清单 1.2 中的数值，随便哪个都可以。这个数值代表什么呢？是表示赋值或加法等指令的种类呢，还是表示将成为指令执行对象的数据呢？也有这样的可能（不过这终归是想象），第一个数值 C7 表示指令，第二个数值 45 表示数据。在诸位所使用的 Windows 个人计算机中，应该会有若干个以 .exe 为扩展名的可执行程序文件。无论是哪个程序，其内容都是数值的罗列，每个数值要么是指令，要么是数据。

1.4　对计算机来说什么都是数字

计算机有计算机的处理方法，这是三大原则中的最后一点。计算机本身只不过是为我们处理特定工作的机器。如果计算机能自己干活的话，那么笔者一定会买几百台，让它们先替自己完成一整年的工作。但是，并没有这种会挣钱的计算机，计算机终究只是受人支配的工具。

迄今为止，使用计算机的目的就是为了提高手工作业的效率。例如，文字处理软件可以提高编写文档的效率；电子邮件可以提高传统邮件寄送的效率。总之，作为可以提高工作效率的工具，有些靠手工作业完成的业务可以直接交给计算机处理。但是也有很多手工作业无法直接由计算机处理。也就是说，在用计算机替代手工作业的过程中，要想顺应计算机的处理方法，有时就要违背人们的思维习惯。请诸位特别留心这一点。

用数字表示所有信息，这就是一个很具有代表性的计算机式的处理方法，这一点也正是和人类的思维习惯最不一样的地方。例如，人们会用“蓝色”“红色”之类的词语描述有关颜色的信息。可是换作计算机的话，就不得不用数字表示颜色信息。例如，用“0,0,255”表示蓝色，用“255,0,0”表示红色，用“255,0,255”表示由蓝色和红色混合而成的紫色。不光是颜色，计算机对文字的处理也是如此。计算机内部会先把文字转换成相应的数字再做处理，这样的数字叫作“字符编码”。总之计算机会把什么都用数字来表示。

熟悉计算机的人经常会说出一些令人费解的话，例如“在这里打开文件，获得文件句柄”“把用公钥加密后的文件用私钥解密”。那么，他们所说的“文件句柄”是什么呢？——是数字。“公钥”是什么呢？——是数字。“私钥”呢？——当然还是数字。无论计算机所处理的信息是什么形式，只要把它们都当成是数字就可以了。虽然这有些违背人们的思维习惯，但是处理数字对计算机来说却是非常简单的。

下面笔者就讲一件自己年轻时的糗事吧。事情发生在一次与老程序员探讨问题时，我问他：“用某某程序处理的某某数据，在计算机内部也是用数字表示的吧？”老程序员听后，吃惊得张开了嘴，回了一句：“这不是明摆着吗！”

1.5 只要理解了三大原则，即使遇到难懂的最新技术，也能轻松应对

有关计算机三大原则的说明到此结束。只要理解了这三大原则，即使遇到难懂的最新技术，也能轻松应对。下面就给诸位看一个具体的例子。这里摘录了一段有关 .NET 技术的介绍，.NET 是微软公司率先提出的一种新技术。如果要正式地介绍 .NET 技术，就会像下面这样进行说明。

【有关 .NET 的说明之一】

微软公司率先提出了作为新一代互联网平台的 .NET 技术。作为 .NET 核心的 XML Web 服务使用通用技术 SOAP、XML，促使企业间的计算机协同工作。

真是不好理解的一段话啊。可是如果把 .NET 的核心技术对照着计算机三大原则再介绍一遍的话，就会像下面这样进行说明。

【有关 .NET 的说明之二】

计算机是执行程序的机器。程序是指令和数据的集合。为了使互联网上相互连接的计算机能通过程序协同工作，微软公司采用了 SOAP 以及 XML 规范。SOAP 是关于调用指令的规范，XML 则是定义数据格式的规范。

只要定义出了指令和数据的规范，装有符合规范的程序的计算机自然就可以相互协作了。所谓计算机的协同工作指的是，输入到一台计算机中的数据，可以通过互联网传送到与这台计算机相连的其他计算机上执行运算，运算所输出的结果再返回给这台计算机。像这样部署在其他计算机上能执行某种运算的程序就叫作 XML Web 服务。

这回怎么样？应该变得容易理解了吧？如果又想到了其他的问题，比如“为什么不得不遵循 SOAP 和 XML 的规范呢？”或者“实际看了看 SOAP 和 XML 的规范，才发现也很复杂。”那么就可以把答案归结为“因为那些都是适合计算机的处理方式”。

1.6　为了贴近人类，计算机在不断地进化

围绕着计算机的技术正在以狂奔般的速度不断进化，与其说是日新月异，倒不如说是“秒新分异”。虽然也许有人会觉得眼前的已经够

用了，希望能停留在现有的技术水平上。但是计算机的进化是不会停止的，因为计算机还远远没有到达完善的地步。

计算机进化的目的只有一个——与人类更加相近。要想贴近人类，就必须从计算机的处理方式中摒弃不符合人们思维习惯的部分。请对照着计算机三大原则之一的“计算机有自己的处理方式”来记忆这个结论。

举例来说，键盘这种不好用的输入设备进化成了好用的鼠标。平面的 2D（二维）游戏进化成了立体的 3D（三维）游戏。无论是哪一种进化，都是为了使计算机的处理方式更加贴近人类。

这样发展下去的话，也许计算机进化的最终形态就是机器人了，有着与人类一样的外表，可以使用人类的语言。例如在 1985 年茨城县筑波市举办的筑波世博会上，就展示出了一台用 CCD 照相机识别乐谱，弹奏钢琴的机器人。也许有人会觉得：“数码音乐什么的用个人计算机不是也能完成吗?”但是这个发明的意义在于机器人能和人类做相同的事了。就在不久前，本田公司开发出的两足步行机器人也成为了热议的话题。也许又有人会觉得：“为什么非要特地用两只脚行走呢，装上轮子能动起来不也一样吗?”但是这个发明的意义还是在于机器人能和人类做相同的事了。有乐谱和钢琴就能演奏，人能走的道路或台阶它也能走，这样的机器人无疑才能更加方便地应用于人类社会。

若与十几年前相比，诸位身边的个人计算机也在逐渐贴近人类。20 世纪 80 年代中期盛行的个人计算机操作系统是 MS-DOS，其操作方法是靠在全黑的画面上敲入字符，把命令传给计算机。进入 90 年代后，MS-DOS 进化成了 Windows，用户可以在图形界面上通过鼠标的操作直观地下达命令（如图 1.3 所示）。开发出 Windows 的美国微软公司，正将目标锁定在用户体验（User Experience）上，旨在开发出超过现有 Windows、更加贴近人类的用户界面（计算机的操作方法）。

Windows XP 和 Office XP 末尾的 XP，代表的就是 Experience（体验）。Windows 若能这样不断进化下去，早晚会有一天，面向个人计算机的语音输入和手写输入等技术将变得极为普及。

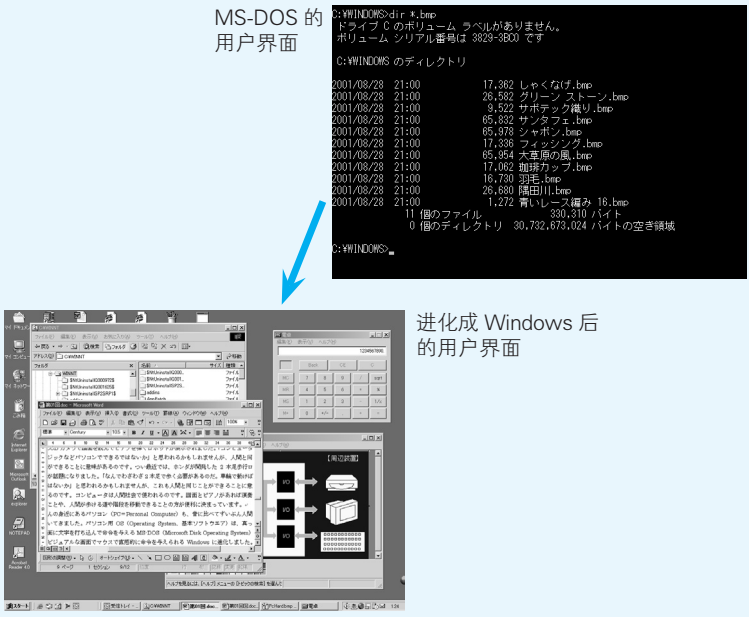

图 1.3　为了贴近人类，个人计算机操作系统也在进化

诸位读者当中应该也有对编程感兴趣的人吧。编程方法也在进化，进化的成果是诞生了两种编程方法，面向组件编程（Component Based Programming）和面向对象编程（Object Oriented Programming）。这两者的进化目标一致，都是使程序员可以在编程中继续沿用人类创造事物时的方法。面向组件编程的方法是通过将组件（程序的零件）组装到一起完成程序；面向对象编程的方法是先如实地对现实世界的业务建模，之后再把模型搬到程序中。使用符合人类思维习惯的编程方法，可以实现高效率的开发。

但是，偏偏有这类程序员，他们对面向组件编程敬而远之，明明有各种各样现成的组件可供使用，却什么功能都要自己亲手做，仿佛不这样编程就不舒心。还有的程序员误认为面向对象编程难以理解。像这样的程序员人数还不少，特别是在昔日的计算机发烧友当中。总之就是因为他们太习惯于配合计算机的处理方式了，反倒认为计算机贴近人类这一发展趋势是在添乱。

笔者则认为，无论是刚入行的技术人员，还是有资历的老工程师，都应该由衷地欢迎技术的进化，坦率地接受新技术。如果是用祖传技艺制作出来的传统手工艺品的话，也许还有价值，但是没有人会稀罕靠一成不变的方法编写出的程序。

1.7 稍微预习一下第 2 章

作为第 2 章的预习，在本章的最后先来简单地介绍一下计算机（特别是个人计算机）硬件的组成要素。这里讲得不会很难，请先看一下图 1.4，体会一下图中的要点。如图所示，计算机内部主要由被称作 IC 的元件组成。虽然在 IC 家族当中有功能各异的各种 IC，但是在这里希望诸位记住的只有三种：CPU（处理器）、内存以及 I/O。

CPU 是计算机的大脑，在其内部可对数据执行运算并控制内存和 I/O。内存用于存储指令和数据。I/O 负责把键盘、鼠标、显示器等周边设备和主机连接在一起，实现数据的输入与输出。

在诸位所使用的 Windows 个人计算机中，多数都只装有一枚名为 Pentium（奔腾）的 CPU 吧。内存的数量则会根据所需存储的大小（少则 32MB，多则 256MB）装有多条。I/O 也会根据周边设备的多少装配有多个。可以认为个人计算机背板上有多少个插孔就有多少个 I/O。

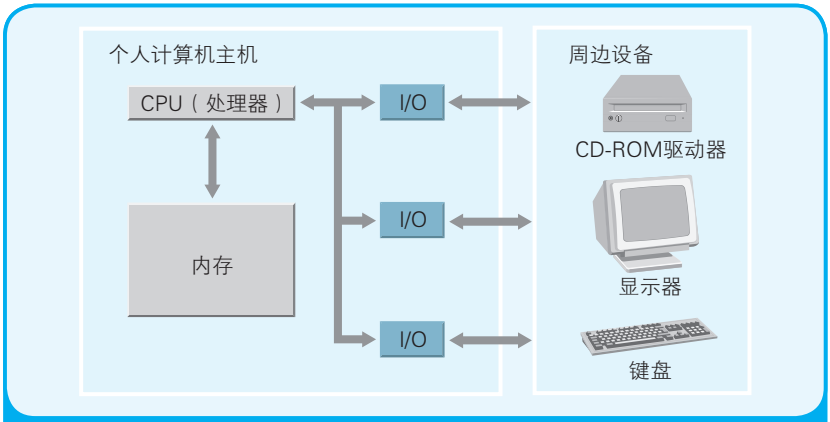

图 1.4　计算机硬件的组成要素

只要用电路把 CPU、内存以及 I/O 上的引脚相互连接起来，为每块 IC 提供电源，再为 CPU 提供时钟信号，硬件上的计算机就组装起来了，还是非常简单的吧。所谓时钟信号，就是由内含晶振[①]的、被称作时钟发生器的元件发出的滴答滴答的电信号。如果是 Pentium CPU 的话，所使用的时钟信号会从几百 MHz 到 2GHz 不等。

☆　　☆　　☆

诸位辛苦了，至此第 1 章就结束了。想必诸位都已经理解了计算机的三大原则以及计算机为什么要进化了吧。因为这些知识真的非常重要，所以如果第一遍没有读懂，就请再反复多读几遍。也可以叫上公司的同事、学校的同学一起讨论本章的内容。如果能让有资历的老工程师也加入讨论，那么效果会更加显著。

在接下来的第 2 章中，我们将尝试着动手“制造”一台计算机。说是制造，也只不过是在纸上进行的“模拟体验”，而且笔者会带着诸位做，所以请不要担心。敬请期待！

① 一种利用石英晶体（又称水晶）的压电效应产生高精度振荡频率的电子元件。——译者注

第2章 试着制造一台计算机吧

热身问答

在阅读本章内容前，让我们先回答下面的几个问题来热热身吧。

初级问题

CPU 是什么的缩写?

中级问题

Hz 是表示什么的单位?

高级问题

Z80 CPU 是多少比特的 CPU ?

怎么样？被这么一问，是不是发现有一些问题无法简单地解释清楚呢？下面，笔者就公布答案并解释。

答案

初级问题：CPU 是 Central Processing Unit（中央处理器）的缩写。

中级问题：Hz（赫兹）是频率的单位。

高级问题：Z80 CPU 是 8 比特的 CPU。

解释

初级问题：CPU 是计算机的大脑，负责解释、执行程序的内容。有时也将 CPU 称作“处理器”。

中级问题：通常用 Hz 来表示驱动 CPU 运转的时钟信号的频率。1 秒发出 1 次时钟信号就是 1Hz，所以 100MHz（兆赫兹）的话就是 100 × 100 万 = 1 亿次 / 秒。M（兆）代表 100 万。

高级问题：CPU 上数据总线的条数，或者 CPU 内部参与运算的寄存器的容量，都可以作为衡量 CPU 性能的比特数。在 Z80 CPU 中，无论是数据总线的条数还是寄存器的容量都是 8 比特，所以 Z80 CPU 是一款 8 比特的 CPU。而在 Windows 个人计算机中广泛使用的 Pentium（奔腾）CPU 则是 32 比特的 CPU。