妨试试从勾画反映程序整体流程的粗略流程图下手。只要在此之上慢慢地细化流程，就能得到详细的流程图。接下来再按照流程图所示的流程埋头编写程序就轻松了。

4.6 特殊的程序流程——中断处理

最后，稍微介绍一下两种特殊的程序流程——中断处理和事件驱动（Event Driven）。首先说明中断处理。

中断处理是指计算机使程序的流程突然跳转到程序中的特定地方，这样的地方被称为中断处理例程（Routine）或是中断处理程序（Handler），而这种跳转是通过 CPU 所具备的硬件功能实现的。人们通常把中断处理比作是接听电话。假设诸位都正坐在书桌前处理文件，这时突然来电话了，诸位就不得不停下手头的工作去接电话，接完电话再回到之前的工作。像这样由于外部的原因使正常的流程中断，中断后再返回到之前流程的过程就是中断处理流程。

在第 2 章微型计算机的电路图中已经展示过，在 Z80 CPU 中有 $\overline{\text{INT}}$ 和 $\overline{\text{NMI}}$ 两个引脚，它们可以接收从 I/O 设备发出的中断请求信号[①]。以硬件形式连接到 CPU 上的 I/O 模块会发出中断请求信号，CPU 根据该信号执行相应的中断处理程序。在诸位使用的个人计算机上，中断请求信号是由连接到周边设备上的 I/O 模块发出的。例如每当用户按下键盘上的按键，键盘上的 I/O 模块就会把中断请求信号发送给 CPU。CPU 通过这种方式就可以知道有按键被按下，于是就会从 I/O 设备读入数据（如图 4.14 所示）。CPU 并不会时刻监控键盘是否有按键被按下。

① $\overline{\text{INT}}$ 引脚用于处理一般的中断请求。$\overline{\text{NMI}}$ 引脚则用于即使 CPU 屏蔽了中断，也可在执行中的指令结束后立刻响应中断请求的情况。

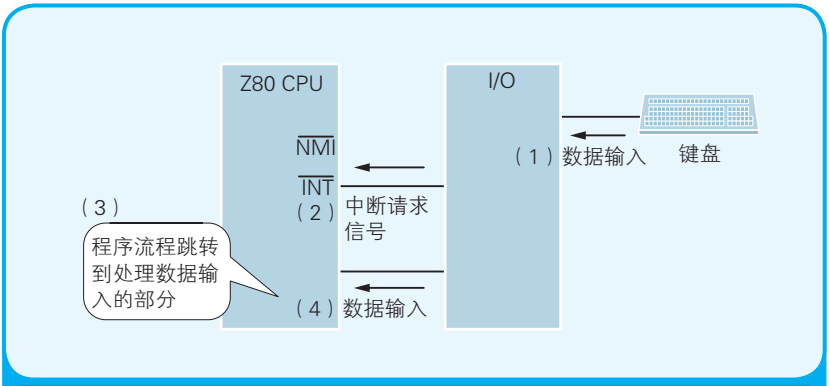

图 4.14　中断请求信号由连接到周边设备上的 I/O 发出

中断处理以从硬件发出的请求为条件，使程序的流程产生分支，因此可以说它是一种特殊的条件分支。可是，在诸位编写的程序中并不需要编写有关中断处理的代码。因为处理中断请求的程序，或是内置于被烧录在计算机 ROM 中的 BIOS 系统（Basic Input Output System，基本输入输出系统）中，或是内置于 Windows 等操作系统中。诸位只需要先记住以下两点即可：计算机具有硬件上处理中断的能力；中断一词的英文是 Interrupt。

4.7　特殊的程序流程——事件驱动

程序员们经常用事件驱动的方式编写那些工作在 GUI（Graphical User Inteface，图形用户界面）环境中的应用程序，例如 Windows 操作系统中的应用程序。这听起来好像挺复杂的，但其实如果把事件驱动想象成是两个程序在对话，理解起来就简单了。

下面看一个实际的例子吧。代码清单 4.5 中列出了一段用 C 语言编写的 Windows 应用程序，这里只列出了程序的骨架。在程序中有

WinMain() 和 WndProc() 两个函数（代码块）。WinMain() 是在程序启动时被调用的主例程（Main Routine）。而 WndProc() 并不会被诸位所编写的程序本身调用，Windows 操作系统才是 WndProc() 的调用者。这种机制就使得 Windows 和诸位所编写的应用程序这两个程序之间可以进行对话。

代码清单 4.5　用 C 语言编写的 Windows 应用程序的骨架

```
/* 主例程 */
int APIENTRY WinMain(HINSTANCE hInst, HINSTANCE hPrevInst,
                     LPSTR lpCmdLine, int nCmdShow){
    …
}

/* 窗口过程 */
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg,
                         WPARAM wParam, LPARAM lParam){
    …
}
```

通常把用户在应用程序中点击鼠标或者敲击键盘这样的操作称作“事件”（Event）。负责检测事件的是 Windows。Windows 通过调用应用程序的 WndProc() 函数通知应用程序事件的发生。而应用程序则根据事件的类型做出相应的处理。这种机制就是事件驱动。可以说事件驱动也是一种特殊的条件分支，它以从 Windows 送来的通知为条件，根据通知的内容决定程序下一步的流程。

要实现事件驱动，就必须把应用程序中的 WndProc() 函数（称为窗口过程，Window Procedure）的起始内存地址告诉 Windows。这一步将在应用程序 WinMain() 中作为初始化处理被执行。

事件驱动是一种适用于 GUI 环境的编程风格，在这种环境中用户可以通过鼠标和键盘来操作应用程序。虽然事件驱动的流程也可以用流程图表示，但是由于要排列很多的菱形符号（表示条件分支），画起

来会很复杂。所以下面介绍便于表示事件驱动的“状态转化图”。状态转化图中有多个状态，反映了由于某种原因从某个状态转化到另一个状态的流程。工作在 GUI 环境中的程序，其显示在画面上的窗口就有若干个状态。例如，如图 4.15 所示的计算器应用程序就可以看作包含三个状态：“显示计算结果”“显示第一个输入的数”以及“显示第二个输入的数”。随着用户按下不同种类的按键，状态也会发生转变。在状态转化图中，在矩形中写上状态的名称，用箭头表示状态转化的方向，并且在箭头上标注引起状态转化的原因（事件）（如图 4.16 所示）。

图 4.15　Windows 附带的计算器应用程序

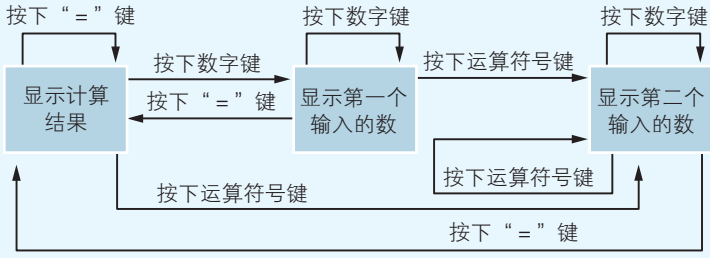

图 4.16　计算器应用程序的状态转化图

对于那些觉得画图很麻烦的人，笔者推荐使用“状态转化表”（如表 4.2 所示）。因为制表的话，用 Microsoft Excel 等表格软件就可以完成，修改起来也要比图方便。在状态转化表中，行标题是带有编号的状态，列标题是状态转化的原因，而单元格中是目标状态的编号。

表 4.2 计算机程序的状态状态转化表的例子

状态 / 状态转化的原因	按下数字键	按下“=”键	按下运算符号键
(1) 显示计算结果	→(2)	→(1)	→(3)
(2) 显示第一个输入的数	→(2)	→(1)	→(3)
(3) 显示第二个输入的数	→(3)	→(1)	→(3)

☆ ☆ ☆

也许读完中断处理和事件驱动的这两节，诸位会觉得稍微有些混乱，但是程序的流程还是只有顺序执行、条件分支和循环这三种，这一点是没有改变的。其中的顺序执行是最基本的程序流程，这是因为 CPU 中的 PC 寄存器的值会自动更新。条件分支和循环，在高级语言中用程序块表示，在机器语言和汇编语言中用跳转指令表示，在硬件上是通过把 PC 寄存器的值设为要跳转到的目的地的内存地址来实现。只要能充分理解这些概念就 OK 了。

在接下来的第 5 章，笔者将更加详细地介绍在本章略有涉及的算法。敬请期待！

COLUMN

来自企业培训现场

电阻颜色代码的谐音助记口诀

无论是哪个行业，都有那么一些数字、结论是从业者必知必会的知识，不得不加以记忆。例如，对于硬件工程师来说，电阻的颜色代码（用于表示电阻值的颜色的搭配）就必须要烂熟于心。在电阻的表面上，可以用 10 种不同颜色的色环来分别表示 0～9 的数字。为了记忆颜色代码，有人还编出了谐音助记口诀，在从业者之间广为流传。

讲师：请先什么都别想，跟着我说：黑灵芝、粽子叶、红孩儿、三乘轿、黄丝带、五缕须、蓝琉璃、钟子期、灰八哥、摆酒宴。

听众：黑灵芝、粽子叶……

讲师：好的，再来一遍。

听众：黑灵芝、粽子叶……老师，请问这到底是什么呢？

讲师：这是为了记忆电阻的颜色代码而编出的谐音助记口诀。可能听一回就记住了。下面请诸位看看手头的电阻，上面带有 4 种颜

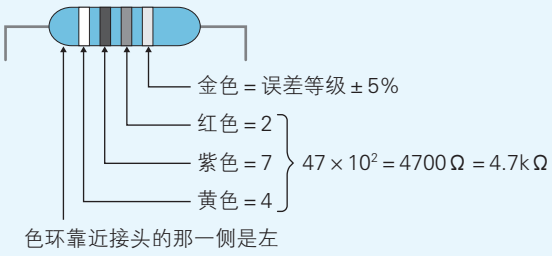

图 A　电阻的外观以及电阻值的计算方法

色的色环。请把金色或银色的色环放到右手边，然后从左边开始依次读出剩余 3 个色环的颜色。

听众：黄色、紫色、红色。

讲师：那就是黄丝带、钟子期、红孩儿。也就是说，所对应的数字是 4、7、2。由这 3 个数字表示的电阻值就是 $47 \times 10^2 \Omega = 4700 \Omega$，即 $4.7 k\Omega$。请注意，从左侧开始数，第 3 位的数字是几就表示是 10 的几次方（如表 A、图 A 所示）。

听众：原来如此！颜色代码已经一口气背下来了。

讲师：再补充一点，银色代表的误差等级是 ±10%，金色代表的误差等级是 ±5%。记忆时可以参考奥运会的奖牌，金牌比银牌的级别高，所以金色的误差更小。

表 A　电阻颜色代码的谐音助记口诀

数字	颜色	谐音助记口诀
0	黑	黑灵（零）芝
1	棕	粽（棕）子叶
2	红	红孩儿（二）
3	橙	三乘（橙）轿
4	黄	黄丝（四）带
5	绿	五缕（绿）须
6	蓝	蓝琉（六）璃
7	紫	钟子（紫）期（七）
8	灰	灰八哥
9	白	摆（白）酒（九）宴

第5章 与算法成为好朋友的七个要点

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

Algorithm 翻译成中文是什么？

中级问题

辗转相除法是用于计算什么的算法？

高级问题

程序中的“哨兵”指的是什么？

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。

初级问题：Algorithm 翻译成中文是“算法”。

中级问题：是用于计算最大公约数的算法。

高级问题：“哨兵”指的是一种含有特殊值的数据，可用于标识数据的结尾等。

解释

初级问题：算法（Algorithm）一词的含义，不仅能在计算机术语辞典上查到，就是用普通的英汉辞典也能查到。

中级问题：最大公约数指的是两个整数的公共约数中最大的数。使用辗转相除法，就可以通过机械的步骤求出最大公约数。

高级问题：字符串的末尾用 0 表示，链表的末尾用 –1 表示，像这种特殊的数据就是哨兵。在本章中，我们将展示如何在“线性搜索”算法中灵活地应用哨兵。

本章重点

程序是用来在计算机上实现现实世界中的业务和娱乐活动的。为了达到这个目的，程序员们需要结合计算机的特性，用程序来表示现实世界中对问题的处理步骤，即处理流程。在绝大多数情况下，为了达到某个目的需要进行若干步处理。例如为了达到“计算出两个数相加的结果”这个目的，就需要依次完成以下三个步骤，即“输入数值”“执行加法运算”“展示结果”。像这样的处理步骤，就被称为算法。

在算法中，有表示程序整体大流程的算法，也有表示程序局部小流程的算法。在第 4 章已经讲解过了表示大流程的算法。那么本章的重点就是表示小流程的算法。

5.1 算法是程序设计的“熟语”

学习编程语言与学习外语很像。为了将自己的想法完整地传达给对方，仅仅死记硬背单词和语法是不够的，只有学会了对话中常用的熟语，才能流利地对话。学习 C 语言、Java 和 BASIC 等编程语言也是如此。仅仅囫囵吞枣地把关键词和语法记下来，是无法流利地和计算机对话的，可是一旦了解了算法就能将自己的想法完整地传达给计算机了。因为算法就相当于是程序设计中的熟语。

“令人生畏且难以掌握”“和自己无缘”，诸位是不是会对算法留下这样的印象呢？诚然，有那种无法轻松理解、难以掌握的算法，但是并不是说只有把那种由智慧超群的学者才能想出的算法全部牢记心中才能编写程序，简单的算法也是有的。而且诸位自己也不妨去思考一些原创的算法。只要理清在现实世界解决问题的步骤，再结合计算机