第 5 章

进阶数据类型

本章将介绍 LabVIEW 中的进阶数据类型，包括数组和簇。

在数组的讲解中，介绍数组的概念和使用，以及与数组相关的函数节点的使用，同时结合 For 循环结构讲解通过结构对数组的数据进行运算。

在簇的讲解中，介绍簇的概念，以及捆绑和解除捆绑的方法。

LabVIEW 编程环境最大的特点就是图形化，数据流和数据可视化是 LabVIEW 编程环境提供的最具特点的方式。最后，本章深入介绍 LabVIEW 中的数据流的编程方式，分别介绍在前面板的数据可视化、前面板控件的可视化操作，以及程序框图中的数据流可视化。

5.1 数组

5.1.1 数组的概念

数组就是同种数据类型的一个集合。例如，将每分钟的温度值记录下来组成的数组就是一个数值型的数组；一串数字信号就是一个布尔类型的数组；在串口通信中每次得到的字符串组成的集合也可以是一个字符串类型的数组，如图 5.1 所示。

数组的基本组成元素是索引和元素。元素在数组中是数据类型相同的数据，这些元素按照一定的顺序排列，每个元素在数组中的位置通过索引来标记。

5.1.2 数组的元素数据类型

在 LabVIEW 中，很多种数据类型都可以作为数组的元素，包括数值、布尔、字符串，也包含一些更加复杂的数据类型，如枚举、图片、簇、引用等数据类型，如图 5.2 所示。基本上，除了数组不可以作为数组的元素外，其他类型的数据都可以作为数组的元素。

在前面板中的大部分控件都可以选择作为数组的元素。图表类的控件不可以作为数组的元素，因为图表本身包含了大量数据，其实质也是一个数组。

5.1.3 数组的类型

根据结构的不同，可以将数组分为一维数组、二维数组、三维数组等，在前面板中可以根

据数组索引表示的不同判别数组的维数，如图 5.3 所示。

程序中主要使用的是一维数组和二维数组，再高维度的数组较少使用，因为很少有与之对应的物理含义。

图 5.1　数值、布尔和字符串类型的数组

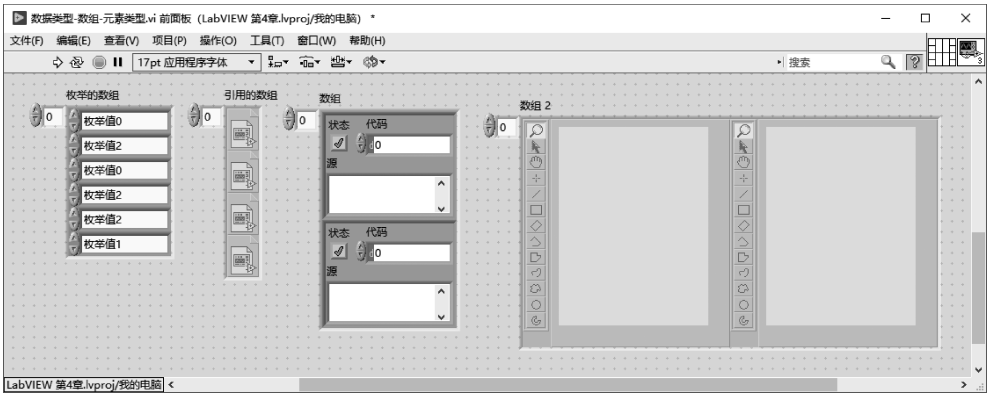

图 5.2　不同元素组成的数组

5.1.4　数组的索引

因为数组的元素类型完全一致，所以对数组的操作主要是使用索引进行的。索引就是元素在数组结构中位置的唯一标识。首先需要操作索引指定到需要处理的元素，然后根据这个元素的数据类型，使用这种数据类型的节点或函数进行处理。

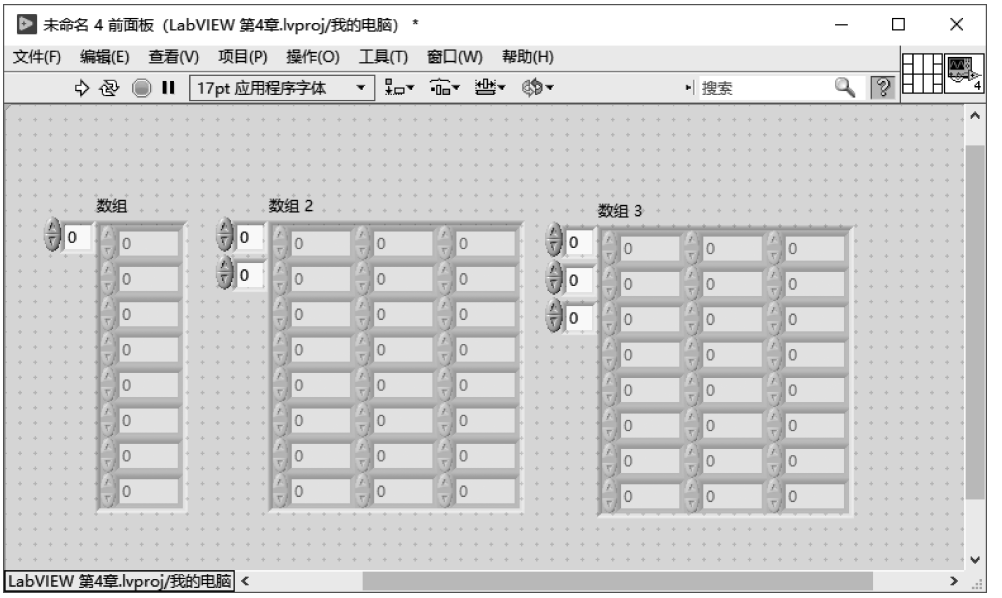

图 5.3　一维、二维和三维数组

数组的索引和数组的维度是一致的，一维数组具有一个索引，二维数组具有两个索引，三维数组具有 3 个索引，如图 5.4 所示。

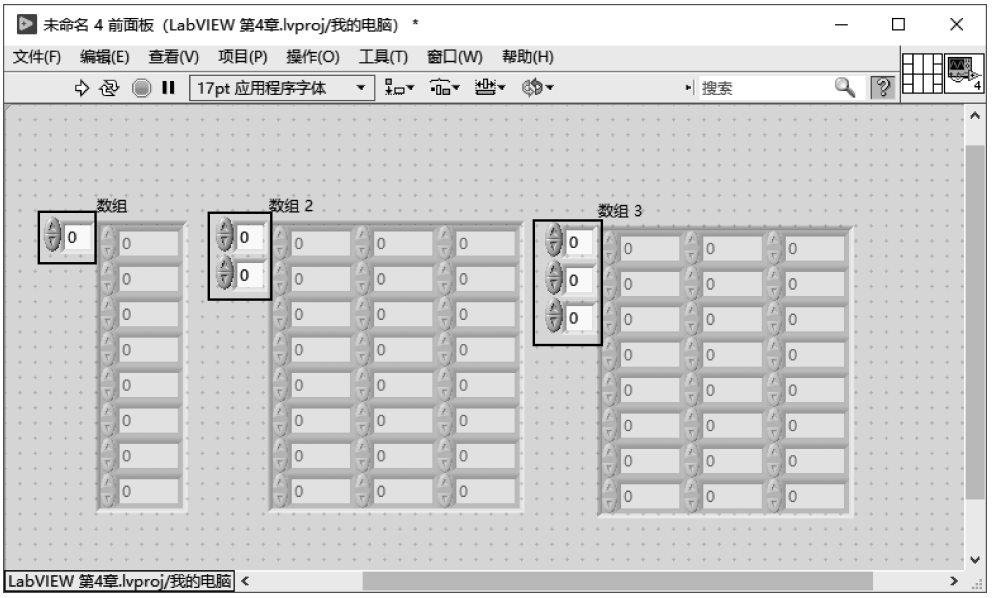

图 5.4　数组的索引

微课视频

5.1.5 数组的使用

数组常常和 For 循环配合使用，因为 For 循环可以在隧道启用索引，这样可以非常方便地进行数组元素的索引。

微课视频

1. 对数值类型数据的二值化实例

在一些算法中会用到数值的二值化处理。需要进行二值化的原始数据是一组数值型的数据，数组的元素是浮点型，二值化后会变为布尔型。一般会设定一个门限值，数值元素的数值大于或等于这个门限值时将其二值化为 1，小于门限值时将其二值化为 0。

具体操作步骤如下。

(1) 在 LabVIEW 菜单栏中执行“文件”→“新建 VI”命令，创建一个空白 VI。

(2) 在 LabVIEW 菜单栏中执行“文件”→“保存”命令，将文件命名为“数据类型-数组-数组操作”。

(3) 初始化一个二维数组，数组的维度是 10×5，数组的元素是 0～1 的随机数。

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“结构”选板，选择“For 循环”结构放置在程序框图中，依次创建两个 For 循环，并将一个 For 循环放置在另一个 For 循环的内部。

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“数值”选板，选择“常量”数值放置在程序框图中，依次建立两个数值常量，并且输入 10 和 5，将这两个数值常量连接至 For 循环的总数接线端。

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“数值”选板，选择“随机数(0-1)”函数节点放置在程序框图的 For 循环中。将“随机数(0-1)”函数节点通过两个 For 循环的隧道输出，如图 5.5 所示。

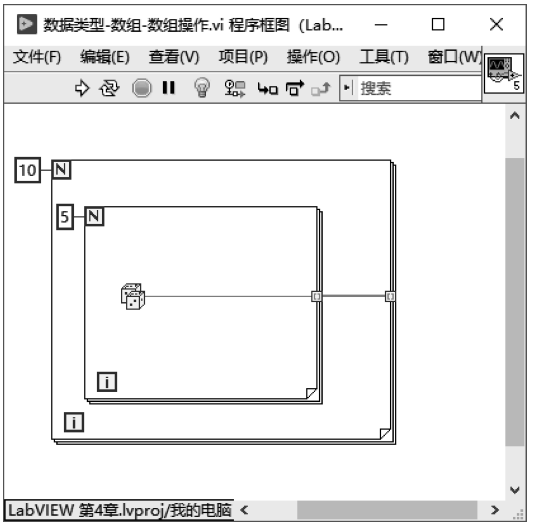

图 5.5 通过 For 循环创建二维数值数组

（4）在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“数组”选板，选择“数组大小”函数节点放置在程序框图中，如图 5.6 所示，将产生的数组数据连接到“数组大小”函数节点的“数组”接线端。

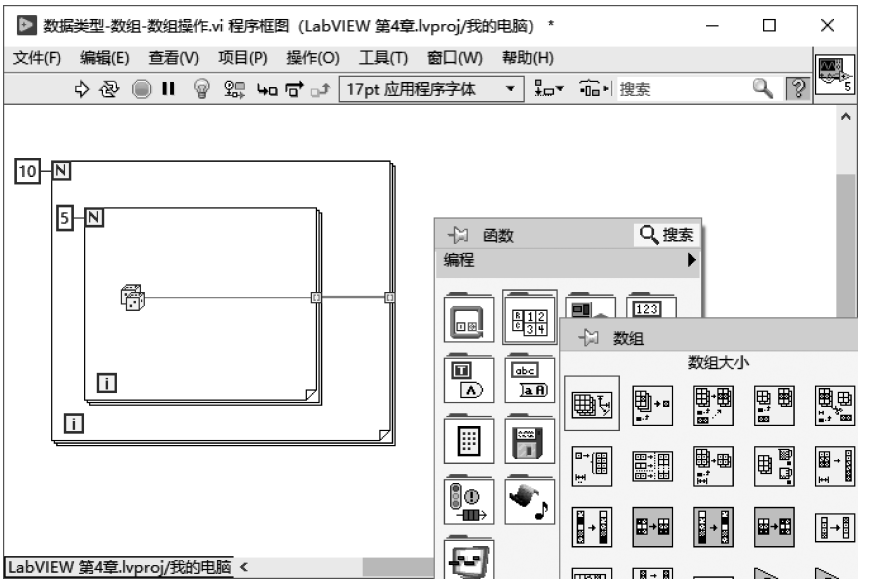

图 5.6　通过“数组大小”函数节点获取数组的维度信息

右击“数组大小”函数节点的“大小”输出端，在弹出的菜单中选择“创建输出控件”，这样可以获得创建数组的维度信息，如图 5.7 所示。

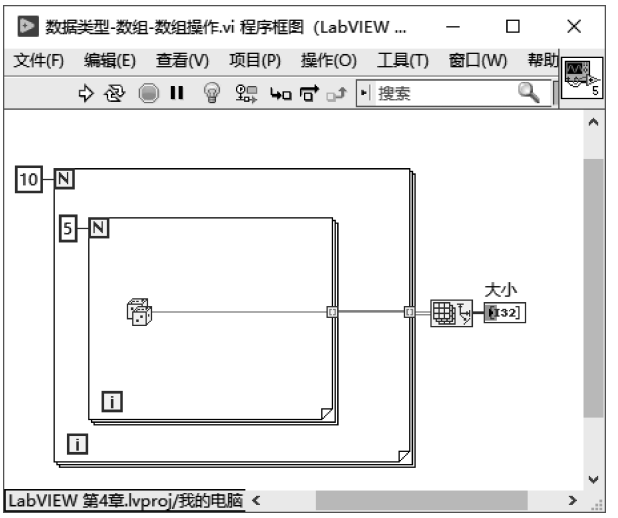

图 5.7　为“数组大小”函数节点创建输出控件

（5）索引数组中的元素。在程序框图中右击，打开“函数”选板，选择“编程”选板→“数组”选板，选择“索引数组”函数节点放置在程序框图中，如图 5.8 所示。

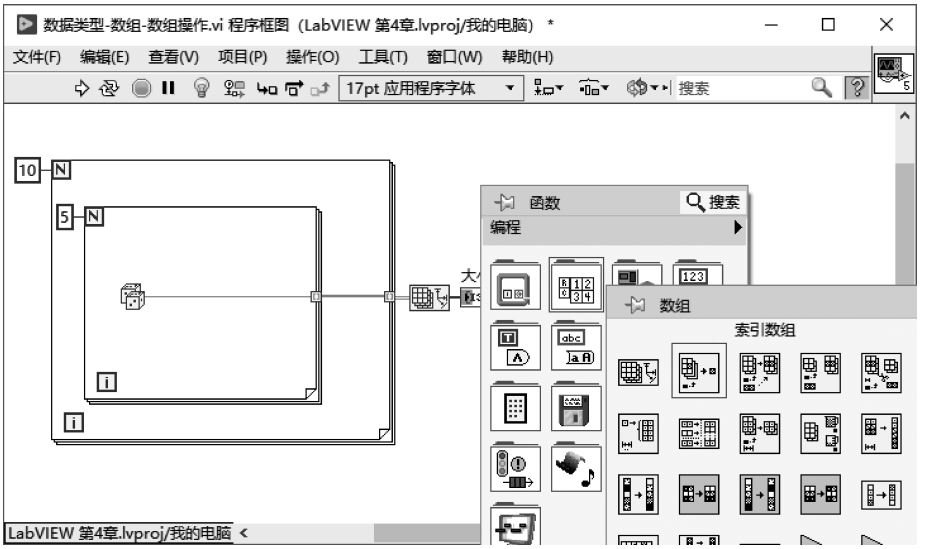

图 5.8　放置“索引数组”函数节点

将“数值大小”函数节点的“大小”输出端连接到“索引数组”函数节点的“数组”接线端。拖动“索引数组”函数节点的下端，其显示出两个“索引”输入端。创建两个整型常量 0 和 1，分别连接到“索引数组”函数节点的索引 0 和索引 1，如图 5.9 所示。这样“索引数组”函数节点得到的就是通过 For 循环生成的数组的列和行信息。

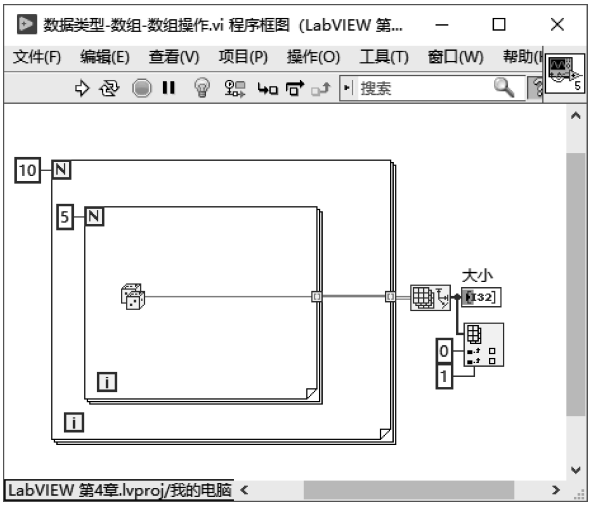

图 5.9　通过“索引数组”函数节点获取数组的列和行信息

（6）通过 For 循环索引数组元素。在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“结构”选板，选择“For 循环”结构放置在程序框图中。创建两个 For 循环，将一个“For 循环”结构放置在另一个“For 循环”结构之中。将“索引数组”函数节点的索引 0 输出的值连接到外面的“For 循环”结构的总数接线端上，将“索引数组”函数节点的索引 1 输出的值连接到内部的“For 循环”结构的总数接线端上，如图 5.10 所示。

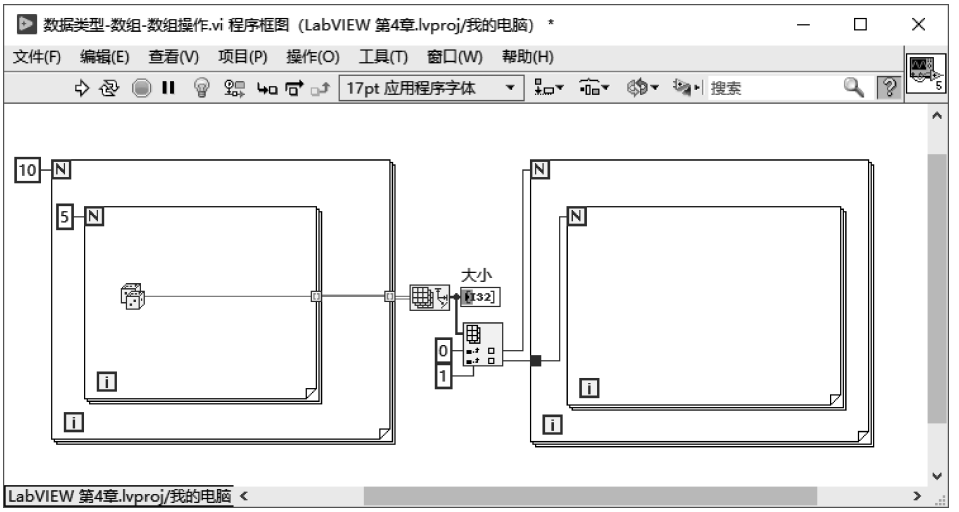

图 5.10　通过“索引数组”函数节点获取数组的列和行信息作为“For 循环”结构的循环次数

（7）通过“条件”结构创建二值化判断算法。将生成的 10×5 随机数组通过两个隧道连接到内部的“For 循环”结构。在前面板中创建双精度浮点型的“数值”输入控件，在程序框图中将“数值”控件的接线端通过两个隧道连接到内部的 For 循环中。

可以看到通过两个“For 循环”结构的隧道，已经将生成的二维数组中的元素索引出来，所以在内部的“For 循环”结构中每次处理的数据是生成二维数组中的元素随机数值。内部的“For 循环”结构将随机数值与数值常量通过“大于或等于?”函数节点进行比较，输出为布尔型结果，将布尔型结果通过“条件”结构判断，输出整型常量 1 或 0。

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“比较”选板，选择“大于或等于?”节点放置在程序框图中。

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“结构”选板，选择“条件”结构放置在程序框图中。

为“条件”结构的条件分支分别创建数值常量 0 和 1，并依次通过“条件”结构和两个“For 循环”结构的隧道输出，如图 5.11 和图 5.12 所示。

（8）在程序框图中创建数组的显示控件，如图 5.13 所示。

（9）在前面板中为产生的随机数组和最终二值化的结果创建显示控件。在前面板中通过鼠标拖动，将创建的“数组”“数组 2”显示控件的元素显示完全。将二值化门限值的“数值”输入控件设置为 0.5。

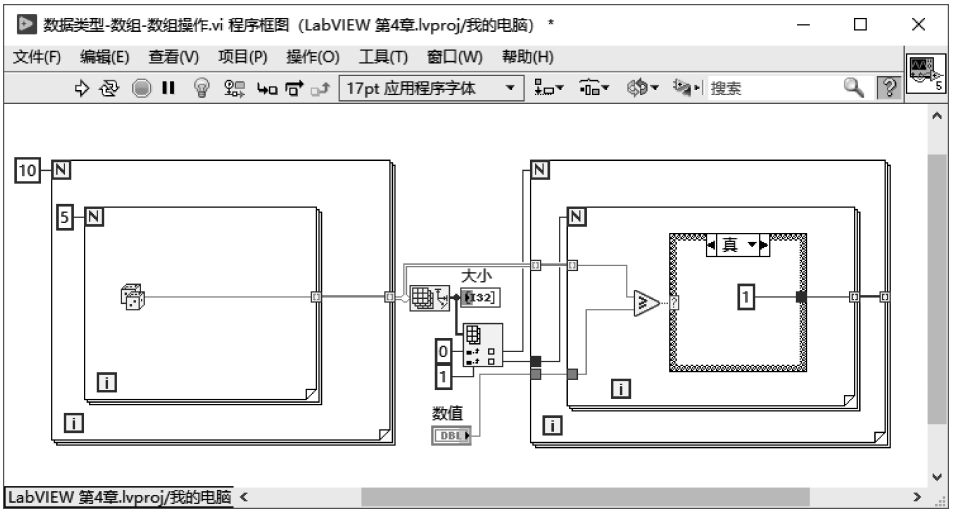

图 5.11　元素大于或等于门限值的条件分支

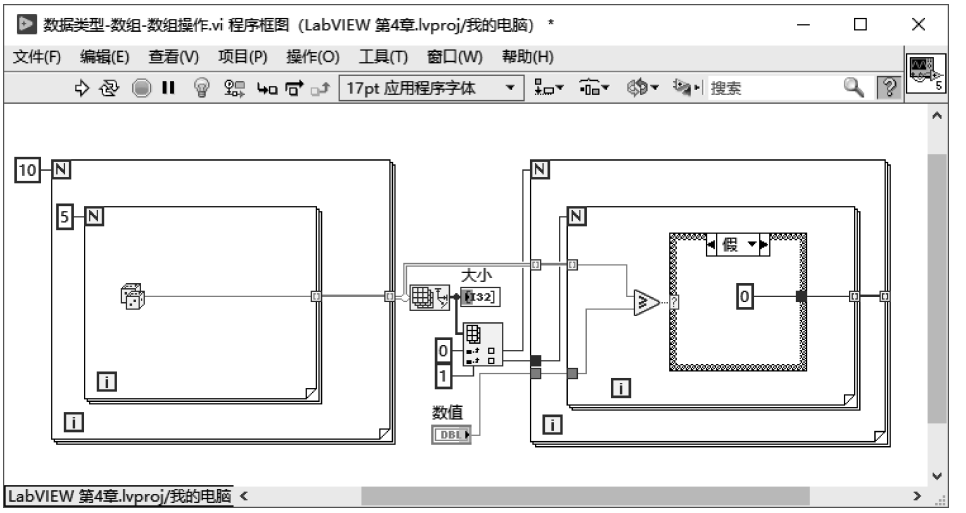

图 5.12　元素小于门限值的条件分支

单击“运行”按钮，执行程序可以得到二值化之后的数组值。如图 5.14 所示，“数组”显示控件显示的是二值化之前的数组，“数组 2”显示控件显示的是二值化之后的数组。

(10) 为了更加直观地观察二值化的效果，通过“强度图”显示控件观察结果。在前面板空白处右击，选择“控件”选板→“新式”选板→“图形”选板，选择“强度图”显示控件放置在前面板中，依次创建两个强度图。单击“强度图”显示控件的幅值选择框，双击默认的上限 100，输入 1，将强度图的显示范围修改为 0～1。

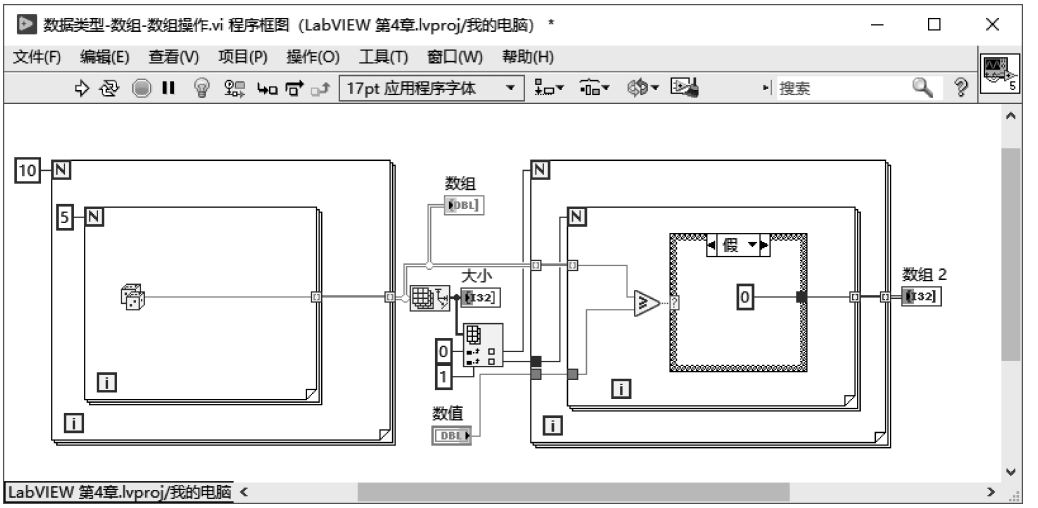

图 5.13　为数组创建显示控件

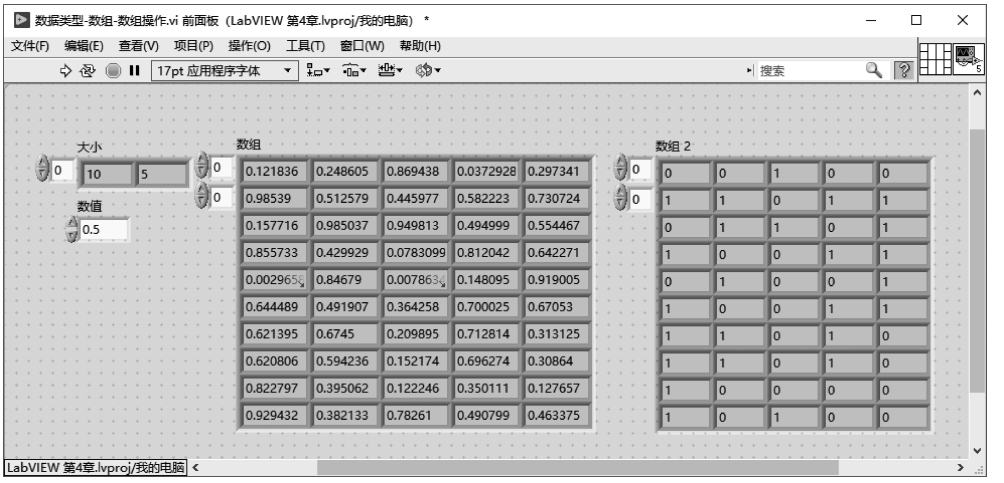

图 5.14　通过元素操作二值化之后的结果

在程序框图中，将两个“强度图”显示控件的接线端分别连接到产生的随机数组和二值化的数组，如图 5.15 所示。

单击“运行”按钮，观察二值化的结果，可以通过“强度图”显示控件看到二值化前后数组的变化，如图 5.16 所示。

这个实例实际上来自自动驾驶中的图像识别，使用二值化将图像中有标志性的图形提取出来，如道路上的道路线，如图 5.17 所示。

2. 自动索引和循环次数

For 循环是处理数组最方便的结构，因为 For 循环可以通过若干次的循环执行索引数

组中的元素。实际上 LabVIEW 中的 For 循环可以更加高效地处理数组，因为 For 循环可以自动对数组进行索引。

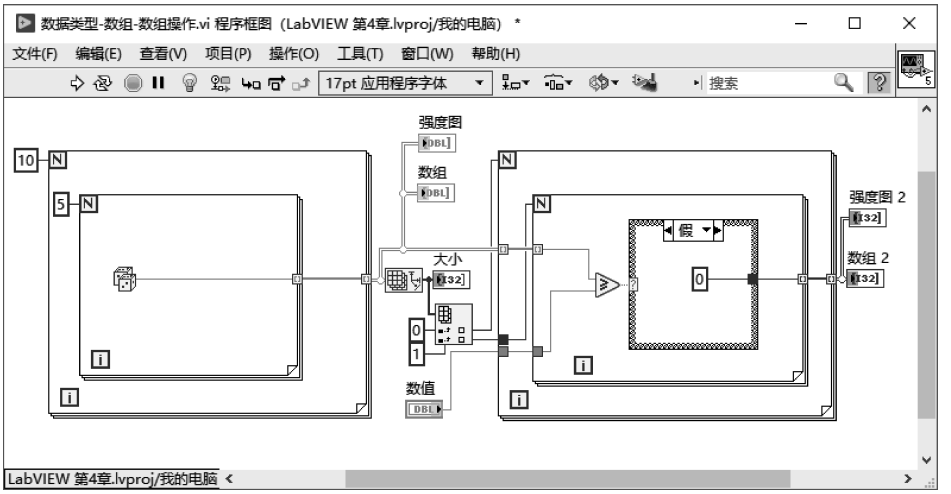

图 5.15　将随机数组和二值化数组连接入强度图

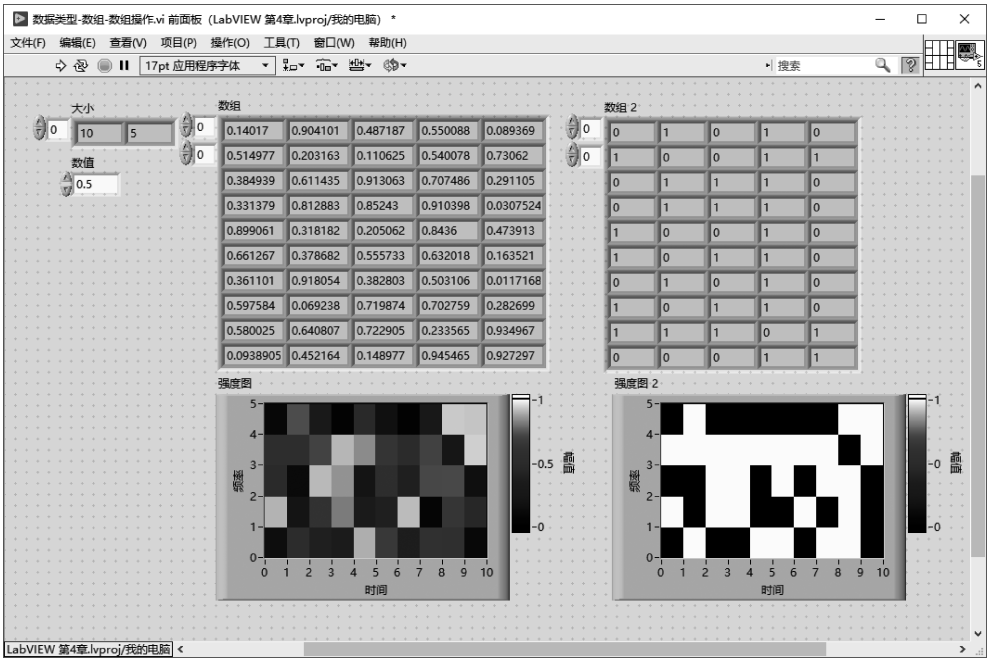

图 5.16　二值化数据在“强度图”显示控件中的显示

例如，通过一个 For 循环产生一个 10 个元素的数组，并且将其输入第二个 For 循环中，如图 5.18 所示。

图 5.17　在自动驾驶中使用二值化识别路标

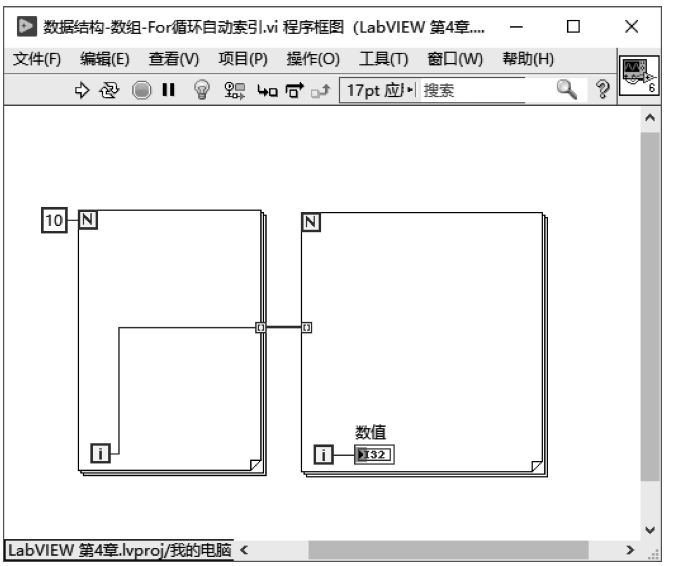

图 5.18　For 循环自动索引

注意，如果没有给第二个 For 循环的总数接线端赋值，程序并不会报错。For 循环会将输入数据的维度值作为 For 循环的循环次数。在本例中，输入的数组维度值为 10，所以第二个 For 循环执行 10 次。观察第二个 For 循环，循环计数器显示的是 9，因为计数是从 0 开始的，循环了 10 次的终值即为 9，如图 5.19 所示。

3. 隧道和自动索引

1）数组通过隧道进入 For 循环

当数组元素进入 For 循环的时候，会创建一个隧道。在默认的情况下，当数据通过隧道进入 For 循环，并不是所有的元素都一次性地进入，而是每次进入一个元素，进入元素的索引是当前 For 循环中循环计数器的值。

2）数组通过隧道流出 For 循环

数据流出 For 循环时，在隧道上默认会自动为每次流出的元素创建索引，所以当数据通

过隧道流出 For 循环时，会自动变成数组的数据类型。

这种数据通过隧道进入和流出 For 循环的方式十分便于处理数组元素。只需要将数组通过隧道进入 For 循环，不需要设置 For 循环的总数接线端，就可以处理数组中的单个元素。当单个元素流出 For 循环的时候，又会自动根据索引组成原来的数组。

图 5.19　For 循环实际循环次数

4. 隧道和禁用索引

右击数据流入的隧道，在弹出的菜单中选择“禁用索引”后，For 循环不会在隧道处索引数据，数组的全部元素会作为一个整体进入 For 循环。

当禁用输入隧道索引后，需要对 For 循环的总数接线端赋值。将数组维度信息输入总数接线端的结果与在隧道自动索引等效。这时如果需要索引数组中的元素，就需要使用循环计数器索引数组中的元素，如图 5.20 所示。

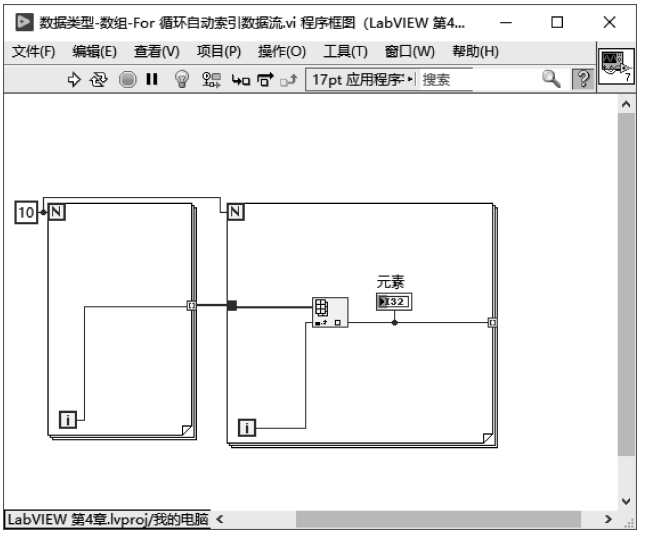

图 5.20　禁用索引使用循环次数的 For 循环

在输出隧道上禁用了索引后，输出的是最后一个流入隧道的值。

5.2　簇

5.2.1　簇的概念

簇是一组元素的集合，集合中元素的数据类型可以不同。例如，LabVIEW 中用于错误处理的错误簇，是由布尔、数组和字符串的数据类型组成的，如图 5.21 所示。

图 5.21　错误簇的结构

5.2.2　簇的操作


微课视频

簇中的每个元素都有标签，可以用来标识，所以可以通过标签对簇进行操作。在通过标签索引到元素之后，通过对应的数据类型的节点和函数进行处理。

1. 错误簇


微课视频

在设计程序的时候经常会遇到错误的处理。LabVIEW 提供的节点中，大部分节点都会包含一个错误簇的选项。在执行的过程中，错误簇会反馈当前代码的执行情况。

错误簇的构成如下。

1）数值

数值表示当前错误对应的错误代码，LabVIEW 提供的子函数和节点中大部分都预定义了一个错误代码。通过这个错误代码可以在“帮助”菜单中查询到详细的错误信息。

2）布尔

布尔表示当前错误的状态，默认值为假，也就是没有错误。当出现错误以后，这个布尔值为真。

3）字符串

字符串包含了解释当前错误的信息的文本。

在 LabVIEW 的执行过程中，当一个节点出现错误时会为这个节点的错误簇赋值，并且将这个错误状态依次传递到最后。

实际上，当一个节点或子 VI 接收传过来数据流的时候，如果错误簇是有错误状态，这个节点或子 VI 就不会执行，而只是单纯地将错误簇向后传递。

在程序设计中可以手动改变错误簇处理的情况。例如，认为某种错误情况并不严重的时候，可以手动将这个错误清理掉，并且让程序正常继续向后执行。

2. 创建一个错误处理 VI 实例

接下来通过一个对错误处理的实例讲解簇的操作，具体步骤如下。

（1）在 LabVIEW 菜单栏中执行“文件”→“新建 VI”命令，创建一个空白 VI。

（2）在 LabVIEW 菜单栏中执行“文件”→“保存”命令，将文件命名为“数据结构-簇-错误处理”。

（3）使用包含若干错误簇的数组模拟错误输入的情况，并且通过索引输入具体的错误情况。

在前面板中右击，在弹出的菜单中选择“控件”选板→“新式”选板→“数据容器”选板，选择“数组”控件放置在前面板中，如图 5.22 所示。

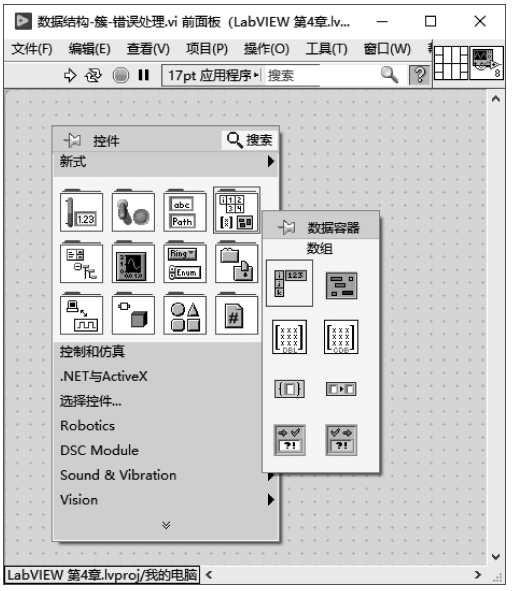

图 5.22　在前面板中添加数组

在前面板中右击，在弹出的菜单中选择“控件”选板→“新式”选板→“数据容器”选板，选择“错误输入 3D”控件放置在前面板中，如图 5.23 所示。将创建的“错误簇”输入控件拖动到刚才创建的数组中。

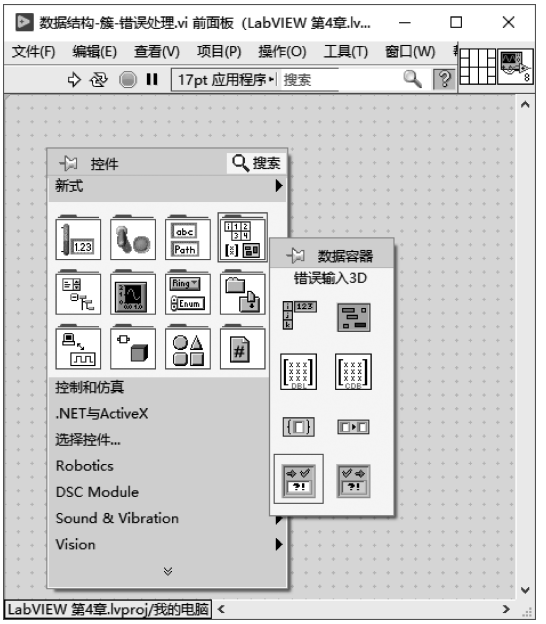

图 5.23　向数组添加错误簇元素

将鼠标指针放在“错误簇”数组下边沿处，向下拖动使数组可以显示 3 个错误簇元素。

同时，为了模拟典型的错误情况，对错误簇数组进行初始化，初始化之后的错误簇数组如图 5.24 所示。

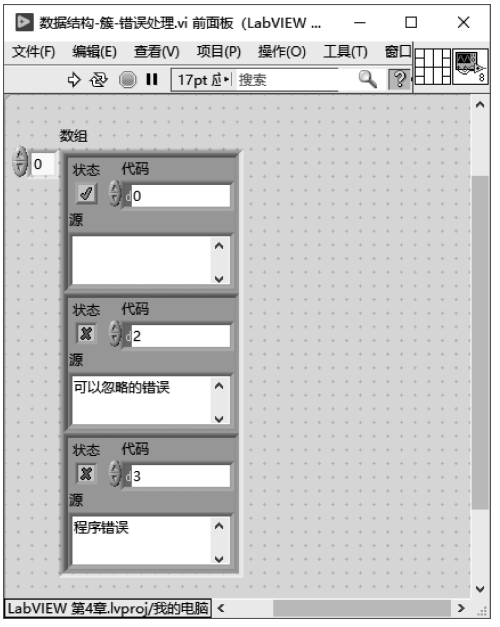

图 5.24　初始化之后的错误簇数组

(4) 在程序框图空白处右击,打开“函数”选板,选择“编程”选板→“数组”选板,选择“索引数组”函数节点放置在程序框图中,将“数组”控件的接线端连至“索引数组”函数节点的数组接线端,右击“索引数组”函数节点的“索引”输入端,在弹出的菜单中选择“创建输入控件”,如图 5.25 所示。

图 5.25 通过索引取出错误簇元素

这样可以通过索引将数组中的错误簇取出进行处理。

(5) 解除捆绑错误簇元素,提取出错误代码信息。

在程序框图空白处右击,打开“函数”选板,选择“编程”选板→“簇、类和变体”选板,选择“按名称解除捆绑”函数节点放置在程序框图中,如图 5.26 所示。

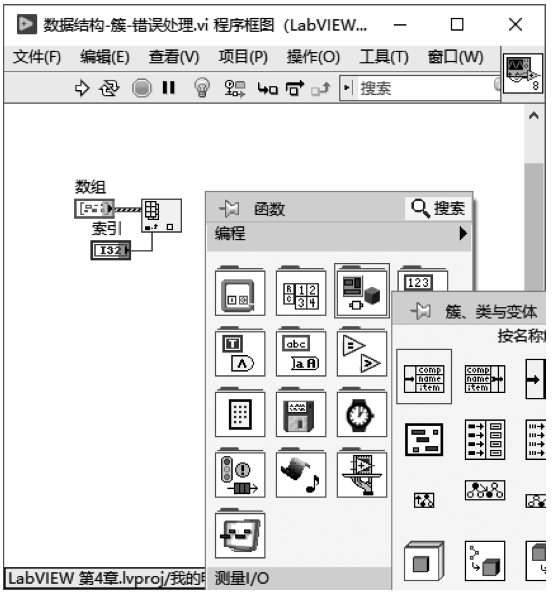

图 5.26 通过解除捆绑获取错误簇中的元素

将错误簇通过“索引数组”函数节点得到的数据连接到“按名称解除捆绑”函数节点的输入端。单击“按名称解除捆绑”函数节点，从下拉菜单中可以看到错误簇包含的 3 个元素：status，code，source。选择 code 一项得到错误簇中的错误代码，如图 5.27 所示。

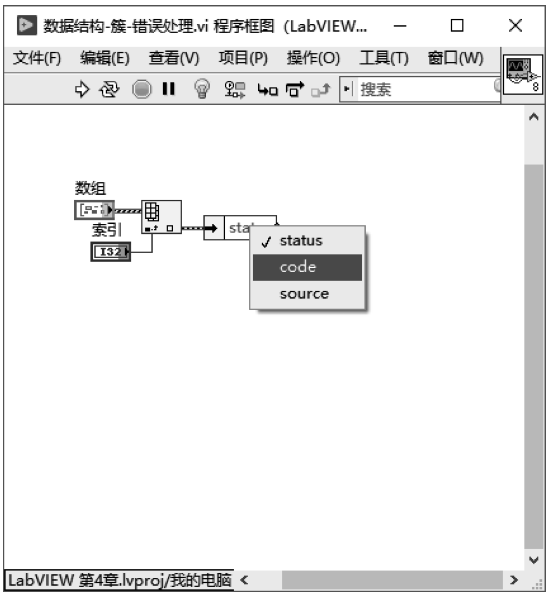

图 5.27　提取出错误簇的 code 元素

(6) 通过“条件”结构判断当前的错误代码。如果错误代码为 2，那么清除这个错误。如果是其他的错误，那么向后传递。

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“结构”选板，选择“条件”结构放置在程序框图中，将“按名称解除捆绑”函数节点输出的错误代码连接到条件结构的选择器接线端。

在“条件选择器标签”中输入 2，然后右击条件分支，在弹出的菜单中选择“编程”选板→“对话框与用户界面”选板，选择“清除错误”函数节点放置在程序框图中，如图 5.28 所示。

将条件选择器接线端的输入端接入“清除错误”函数节点的“清除指定错误”输入端。将索引处的错误输入“清除错误”函数节点的“错误输入”端，将“清除错误”函数节点的“错误输出”通过隧道输出“条件”结构。

在隧道处右击，在弹出的菜单中选择“创建显示控件”，如图 5.29 所示。

可以看到，当错误代码为 2 时，错误簇中的所有数据将会被清除，输出一个没有错误的状态。这样就将错误代码为 2 的错误状态清除。

接下来为“条件选择器标签”的 0 值定义条件分支，当错误代码为 0 时，代表没有错误，在这个条件分支中将索引的错误值直接输出到错误输出端，如图 5.30 所示。

(7) 增加基于错误判断的代码执行结构。一般在程序中执行的每个节点和子 VI 都会有错误簇的输入，当有错误传入时就会直接跳过当前的代码。在编辑一个算法模块时，一般

使用“条件”结构判断输入的错误情况，并且直接将错误簇输入“条件”结构的选择器接线端，“条件”结构会自动索引其中的布尔值。这时“条件”结构有两个条件分支，分别对应没有错误和有错误的情况。

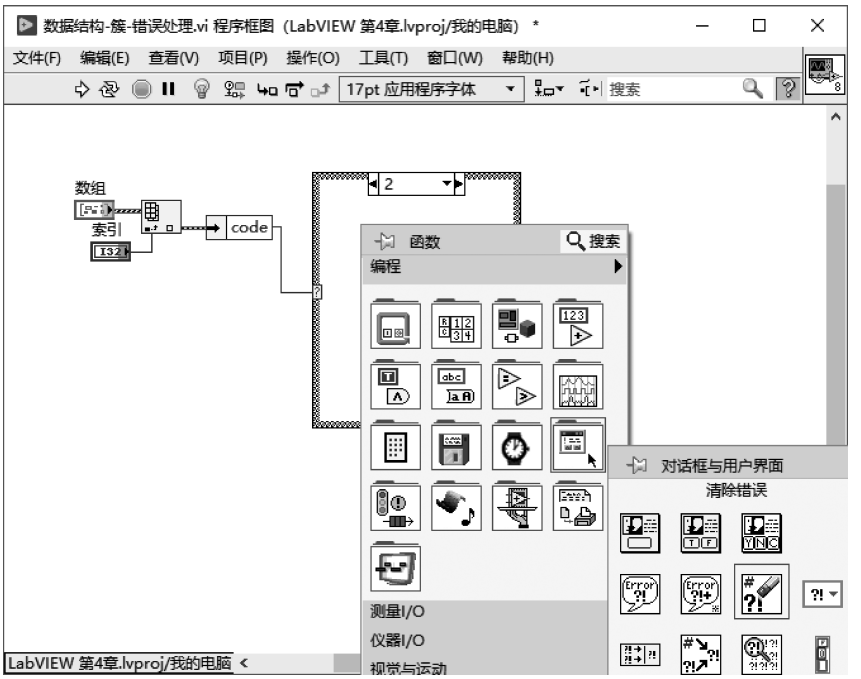

图 5.28　添加“清除错误”函数节点

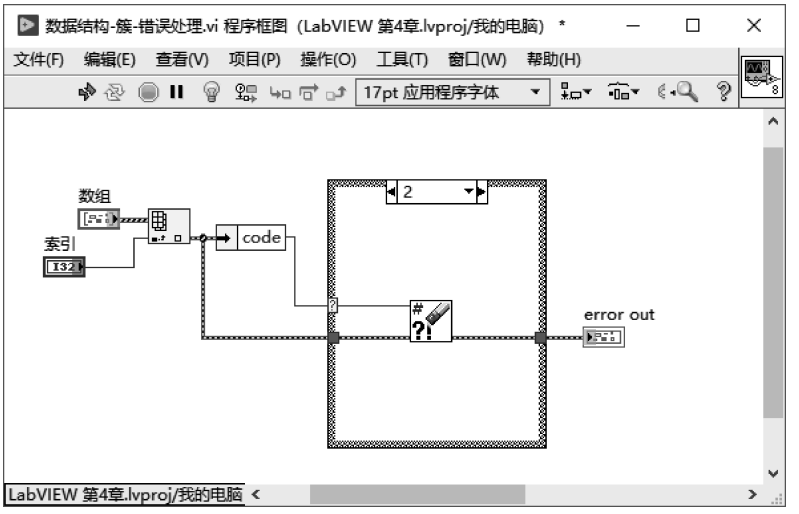

图 5.29　通过“条件”结构的分支清除错误

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“结构”选板，选择“条件”结构放置在程序框图中，将 error out 错误簇接线端直接连接到“条件”结构的选择器接线端。

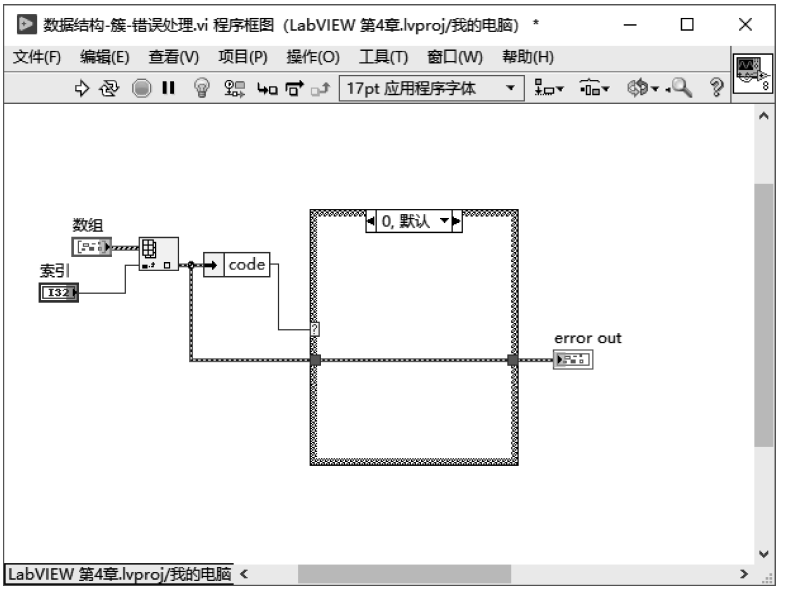

图 5.30　通过“条件”结构定义错误代码为 0 的条件分支

选择“条件选择器标签”的错误条件分支，在空白处右击，打开“函数”选板，选择“编程”选板→“对话框与用户界面”选板，选择“单按钮对话框”函数节点放置在程序框图中，如图 5.31 所示。

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“字符串”选板，选择“字符串常量”函数节点放置在程序框图中，在字符串常量文本框中输入“程序错误”，连接到“单按钮对话框”输入端，如图 5.32 所示。

这样，当程序出现错误并传递到第二个“条件”结构的时候，就会弹出“程序错误”对话框。

(8) 测试在不同错误情况下的结果。在前面板中，依次在“索引”文本框中输入 0，1，2，3，模拟不同的错误输入情况。

在“索引”文本框中输入 0，并单击“运行”按钮。此时从簇的数组中索引第一个错误代码为 0 的错误簇。这个错误簇的状态是没有错误，所以在第一个“条件”结构执行的是 0 条件分支，没有错误的错误簇流到第二个“条件”分支。第二个条件结构执行“无错误”分支，如图 5.33 所示。

在“索引”文本框中输入 1，单击“运行”按钮。此时从簇的数组中索引第一个错误代码为 2 的错误簇。在第一个条件分支中通过“按名称解除捆绑”函数节点得到的错误代码是 2，在第一个“条件”结构处执行 2 条件分支，错误被清除，没有错误的错误簇向后传递，在第二个“条件”结构中执行没有错误的分支。

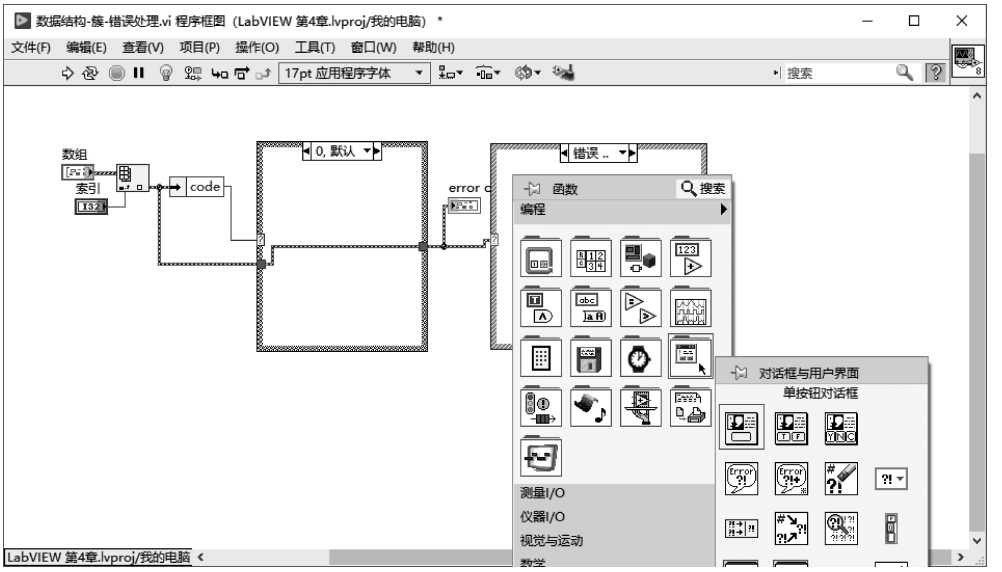

图 5.31　使用错误簇作为“条件”结构的判断

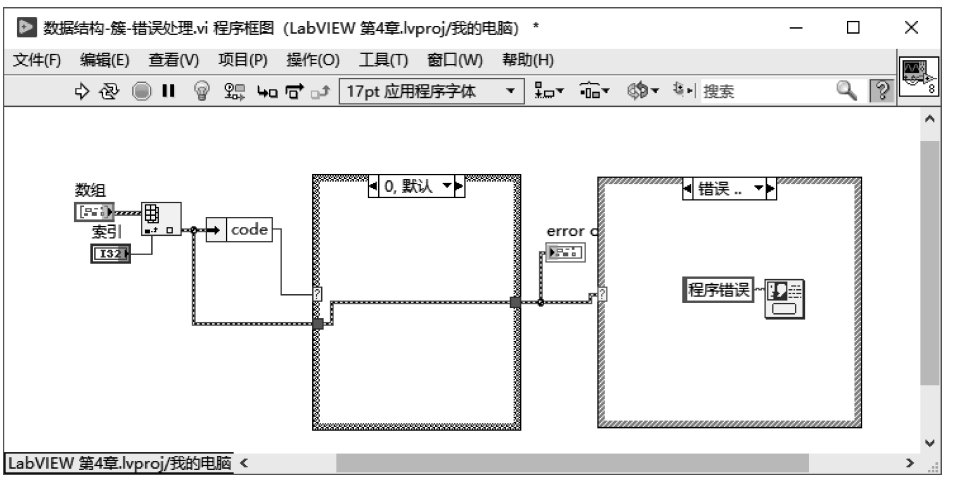

图 5.32　定义“条件”结构错误分支弹出对话框

在“索引”文本框中输入 2，单击“运行”按钮。此时错误簇输入的情况是错误代码为 3 的情况，通过“按名称解除捆绑”函数节点得到的错误代码是 3，在第一个“条件”结构处执行 0 条件分支，错误直接传递到后面，在第二个“条件”结构中执行有错误分支，弹出“程序错误”对话框，如图 5.34 所示。

在“索引”文本框中输入 3，单击“运行”按钮。输入为 3 时，在第一个“条件”结构处没有对应的条件分支，这时执行默认的条件分支，错误簇直接向后传递。在第二个“条件”结构处执行无错误分支。

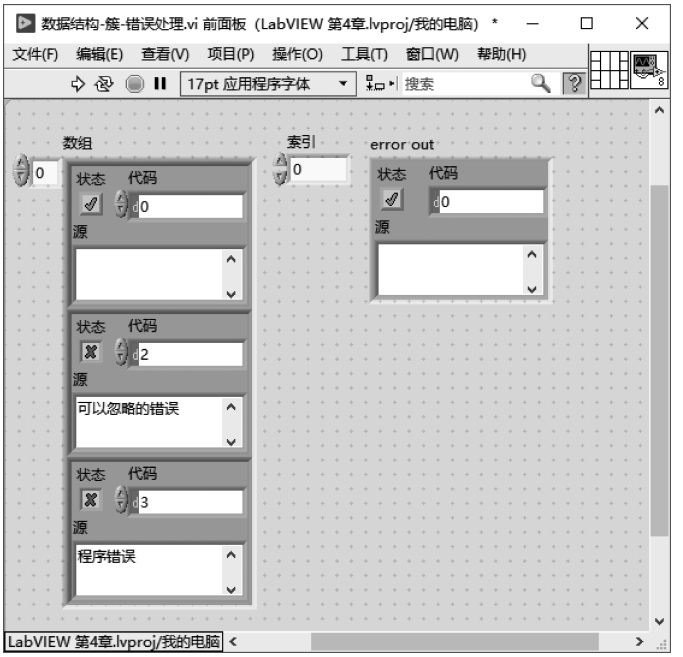

图 5.33　错误代码为 0 时的情况

图 5.34　错误代码为 3 的情况

5.2.3 簇的数据捆绑功能

在进行 LabVIEW 程序设计时，簇是使用得非常多的一种数据结构。簇可以进行数据的捆绑，有效的数据组合可以优化和管理代码。通过将功能相近或相似的数据进行捆绑，可以更好地将程序模块化。

1. 程序框图中的数据捆绑

簇可以理解为数据的一种打包，在 LabVIEW 这种数据流的编程环境中，簇可以大大优化代码的编写。例如，通过数据的捆绑，多条数据流的连线可以合并成一条簇的数据连线。

如图 5.35 所示，在程序框图中通过数据的打包，将布尔型的数据、数值型的数据和字符串型的数据打包在一条错误簇中进行传递。

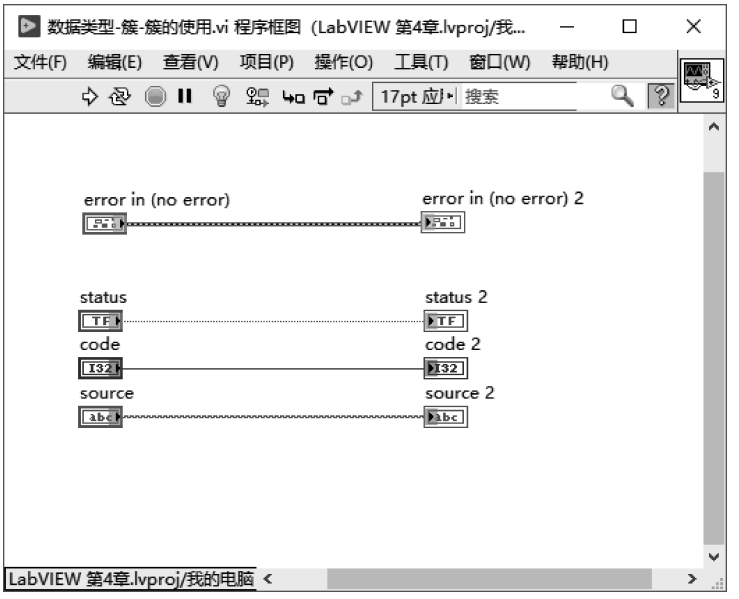

图 5.35 程序框图中的数据捆绑

2. 连线板中的数据捆绑

在 VI 的连线板上，如果为每个输入输出控件的数据元素创建一个接线端，就会导致 VI 的连线端过多，不便于操作。将数据通过簇打包，可以节省连线端，同时含义也更加明确。

如图 5.36 所示，在左图中使用错误簇打包错误信息，接线板只需要使用一个接线端，在引用时，只需要通过一个接线端获取有关错误的全部信息；右图中，表示错误的布尔型数据、数值型数据和字符串型数据没有被打包在一起，在接线板上需要 3 个接线端传递数据，如果需要引用，需要同时连接 3 个接线端才能获得所有的错误信息。

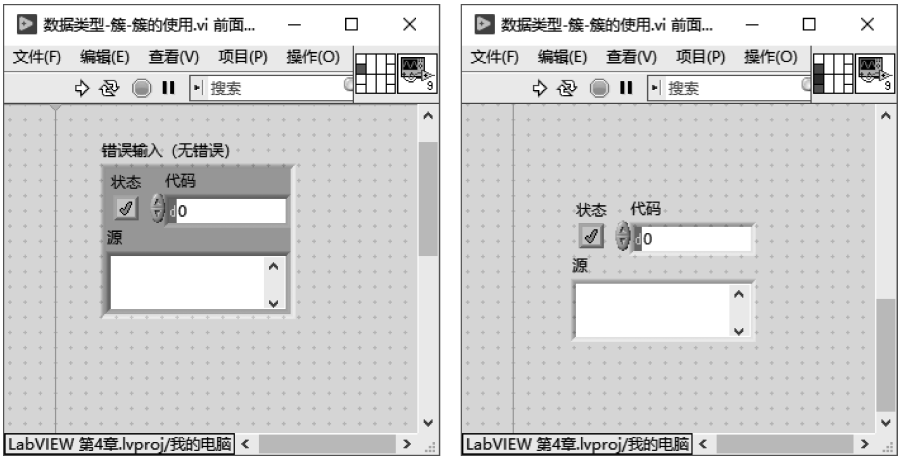

图 5.36　连线板中的数据捆绑

5.3　LabVIEW 中数据流的可视化

LabVIEW 是图形化的环境，对数据的操作和显示有非常好的可视化支持。这样可以更加方便地观察数据，实现数据和算法的逻辑，提升调试和原型系统设计的效率，以及加速对程序的开发和维护。

这种可视化的支持来自以下两方面。

(1) 前面板数据的可视化。

(2) 程序框图数据的可视化。

接下来会分别讲解在前面板和程序框图中数据的可视化。

5.3.1　前面板的数据可视化

LabVIEW 的核心是基于数据流的方式进行程序设计，所以在前面板中以显示和操作数据为核心提供了众多的数据操作和显示方式。前面板是进行人机交互的界面，所以 LabVIEW 也十分适合进行人机交互界面设计。

1. 波形图与波形图表

LabVIEW 前面板为数值型的数据提供了非常多的显示控件，可以根据需要选择不同的控件进行数据的显示。波形图和波形图表是使用最广泛的用于显示波形数据的显示控件。

1) 单条波形显示实例

在前面板空白处右击，依次选择“控件”选板→“新式”选板→“图形”选板，可以选择“波形图表”“波形图”显示控件放置在前面板中，如图 5.37 所示。波形图和波形图表一般用来将数值通过波形的方式显示，这样会使数据更加直观。

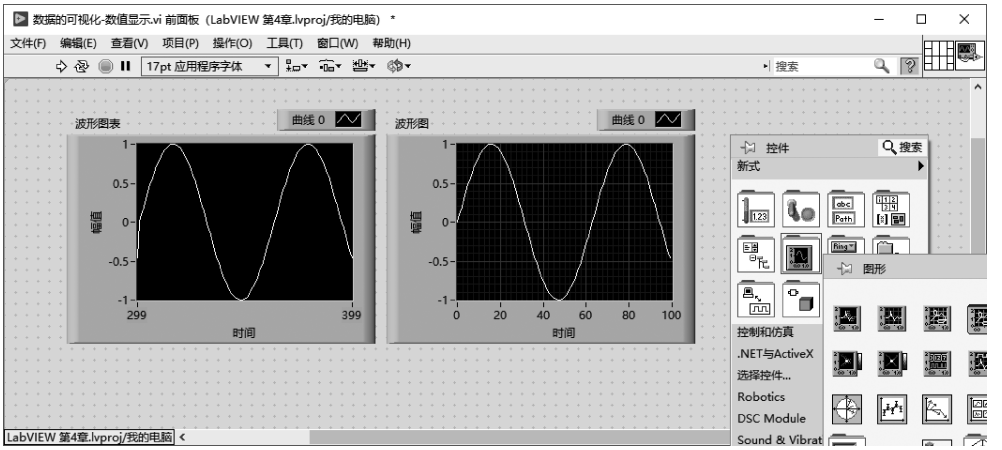

图 5.37　使用波形图表和波形图显示波形数据

这个实例的结构简单，编辑的过程不再赘述。实例的程序框图如图 5.38 所示，通过“For 循环”结构和“正弦”函数节点生成正弦波形数据，并通过“波形图”和“波形图表”显示控件进行显示。

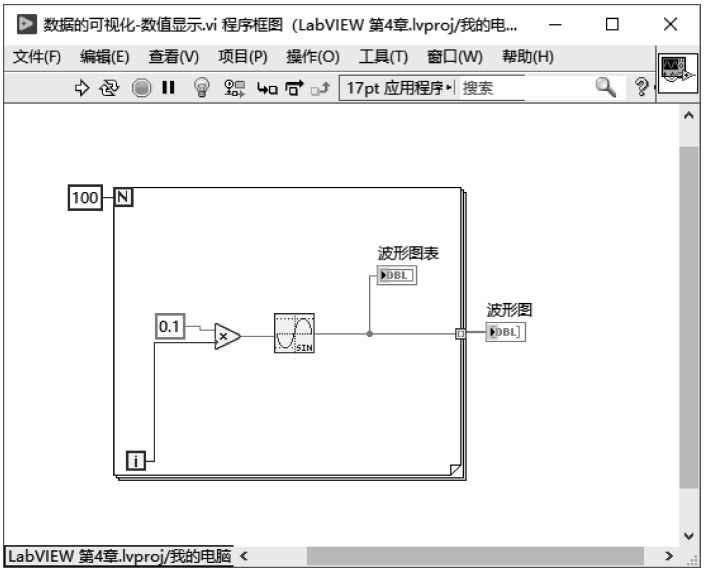

图 5.38　波形图表和波形图的使用方法

波形图和波形图表的使用方法有所不同，在程序框图中可以看到波形图表和波形图在 For 循环中的不同位置。

“波形图”显示控件接收的数据格式是数组，所以“波形图”接线端在 For 循环的外面，正弦波形数组输入“波形图”显示控件。

“波形图表”显示控件可以接收连续的单个数据并保存一定长度的数据。在本实例中，既可以将“波形图表”显示控件放置在 For 循环内部，因为可以接收单个的元素并且保留一定的历史数据；也可以放在 For 循环外面，接收数组格式的数据。

因为波形图表可以保留历史数据，所以也称为趋势图。

2）多条波形显示实例

基于“波形图”和“波形图表”显示控件的显示机制不同，可以通过构建数据结构显示多条波形，接下来通过实例进行讲解，具体操作步骤如下。

（1）使用“波形图”显示控件显示多条曲线。

因为“波形图”显示控件是基于数组的，需要使用“波形图”显示控件显示多条曲线的时候，需要使用通过“创建数组”函数节点合并多个包含数组的数据，如图 5.39 所示。

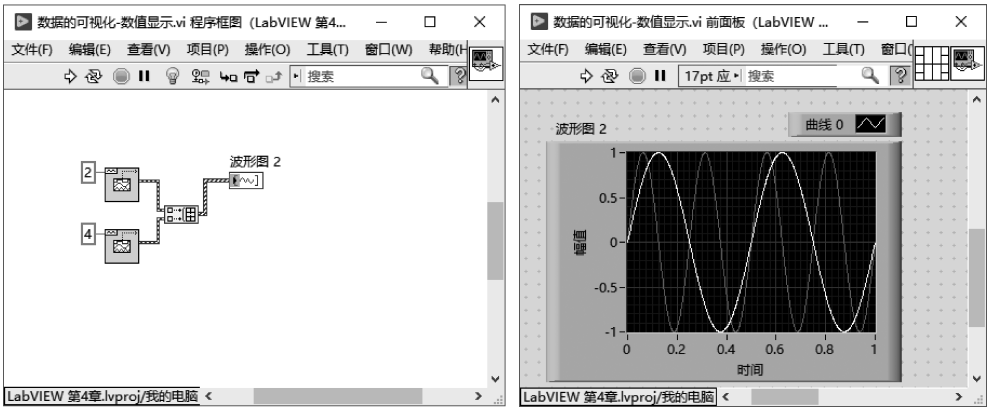

图 5.39　波形图显示多条曲线

在程序框图中，通过“编程”选板→“信号处理”选板→“信号生成”选板中“基本函数发生器”函数节点产生频率为 2Hz 和 4Hz 的两条波形；通过“编程”选板→“数组”选板中的“创建数组”函数节点将两条波形合并，这样就可以通过“波形图”显示控件同时显示这两条波形曲线。

（2）使用“波形图表”显示控件显示多条曲线。

“波形图表”显示控件基于单个数据，如果需要显示多条波形，要将多个波形中的元素先合并成一个簇，然后再组成簇的数组进行显示。可以使用“编程”选板→“数组、类和变体”选板中的“捆绑”函数节点合并多条波形中的元素，然后再输入“波形图表”显示控件进行显示，如图 5.40 所示。

2. XY 图实例

XY 图用于显示一组或多组二维数据，其中二维数据每个点都包含 X、Y 信息。XY 图输入的数据类型是基于簇的，簇中的数组包含了 X 数组和 Y 数组。

接下来通过实例讲解 XY 图的使用，具体步骤如下。

1）使用 XY 图显示一条曲线

如图 5.41 所示，使用 XY 图显示一条曲线，X 轴是一组正弦值，Y 轴是一组余弦值。

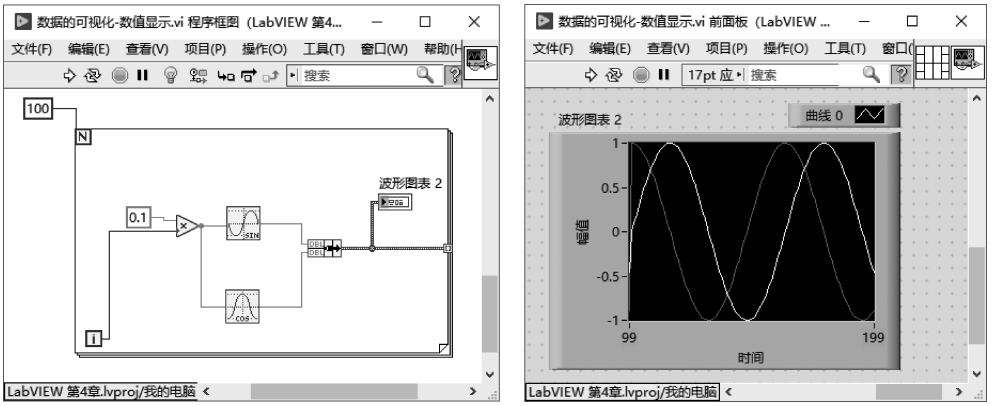

图 5.40 波形图表显示多条曲线

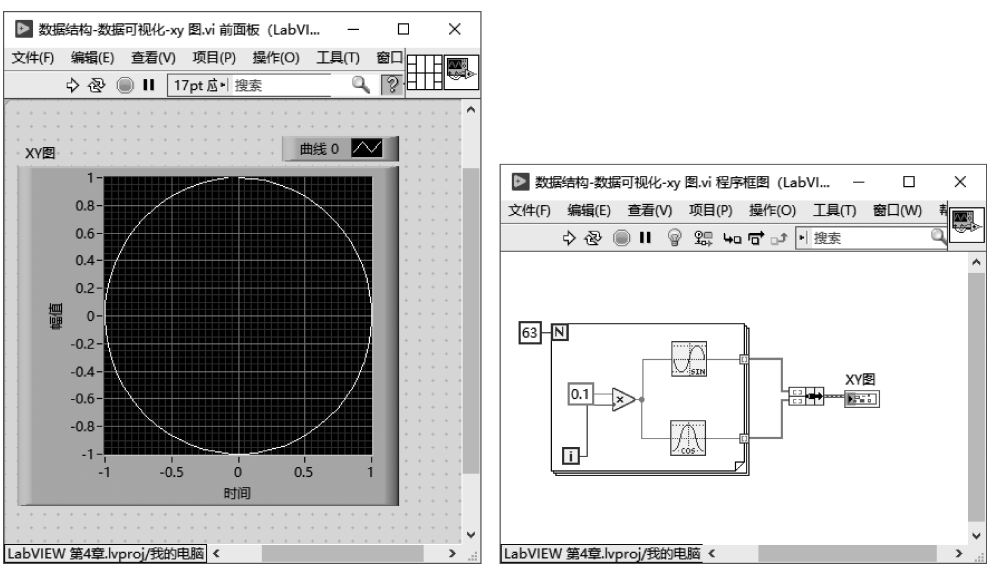

图 5.41 使用 XY 图显示单条曲线

在程序框图中右击，打开“函数”选板，选择“数学”选板→“初等与特殊函数”选板→“三角函数”选板，选择“正弦”函数节点和“余弦”函数节点，放置在程序框图中，通过 For 循环产生正弦波形数组和余弦波形数组，通过使用“编程”选板→“数组、类和变体”选板中的“捆绑”函数节点合并两个数组，将结果连入“XY 图”显示控件的接线端。单击“运行”按钮，可以看到 XY 图显示的图形是一个圆。

2）使用 XY 图显示多条曲线

如果需要显示多条曲线，可以通过“创建数组”函数节点将多个包含 X、Y 数组的簇合并成数组输入 XY 图。

如图 5.42 所示，基于上面的实例生成两条曲线，其中第二条曲线的 X、Y 值是第一条曲

线的 2 倍。通过数值的乘法将第二条曲线的幅度变成第一条曲线的 2 倍，同时通过创建数组，将数据的簇合并。

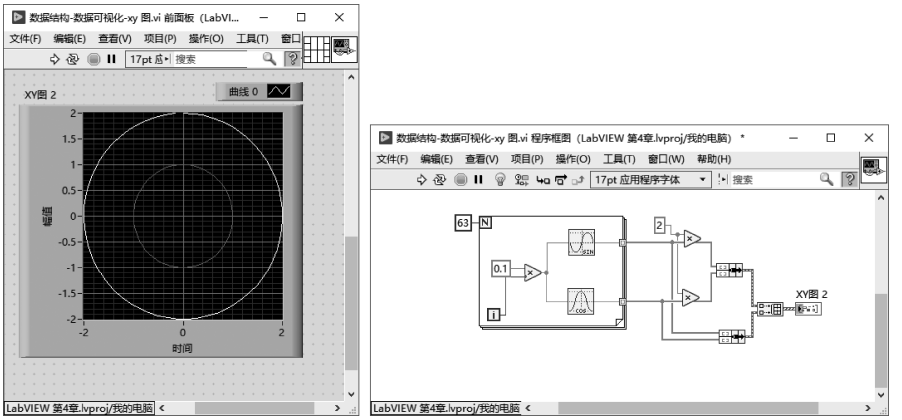

图 5.42　使用 XY 图显示多条曲线

5.3.2　前面板控件的可视化操作

在 LabVIEW 的前面板中，每种数据类型的控件都提供了用于数据显示操作方法，可以不通过编程，直接在前面板对数据进行操作。

1. 数值型控件的按钮

在数值型的控件中，可以通过键盘输入值，还可以通过控件自带的按钮等方法改变控件的值，如“数值”输入控件的增量/减量按钮、“滑动杆”输入控件的滑动钮、“旋钮”输入控件的旋转钮等，如图 5.43 所示。

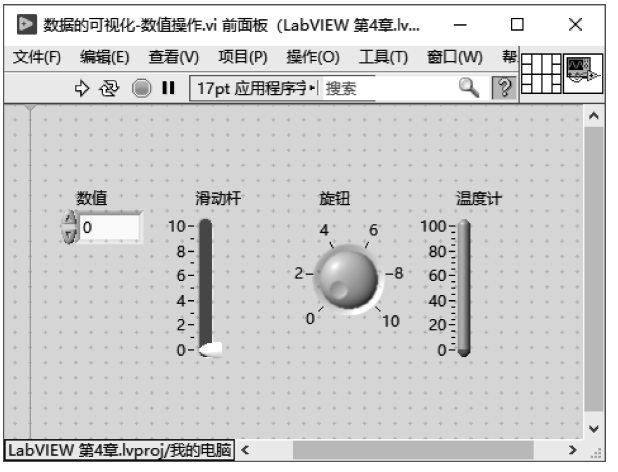

图 5.43　数值型控件的操作按钮

2. 波形图表和波形图的操作实例

在本实例中，将产生的波形输入波形图，并且在前面板的波形图中通过工具方法对波形进行操作。

(1) 在 LabVIEW 菜单栏中执行“文件”→“新建 VI”命令，创建一个空白 VI。

(2) 在 LabVIEW 菜单栏中执行“文件”→“保存”命令，将文件命名为“数据的可视化-数值操作”。

(3) 在程序框图空白处右击，打开“函数”选板，选择“信号处理”选板→“波形生成”选板，选择“基本波形发生器”函数节点放置在程序框图中，如图 5.44 所示，使用“基本函数发生器”节点创建两条波形曲线：

- 曲线 1：频率为 1Hz，采样信息为(Fs：100，采样数：100)；
- 曲线 2：频率为 2Hz，采样信息为(Fs：100，采样数：100)。

(4) 在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“数组”选板，选择“创建数组”函数节点放置在程序框图中。通过“创建数组”函数节点合并两条波形，输入“波形图”显示控件，如图 5.44 所示。

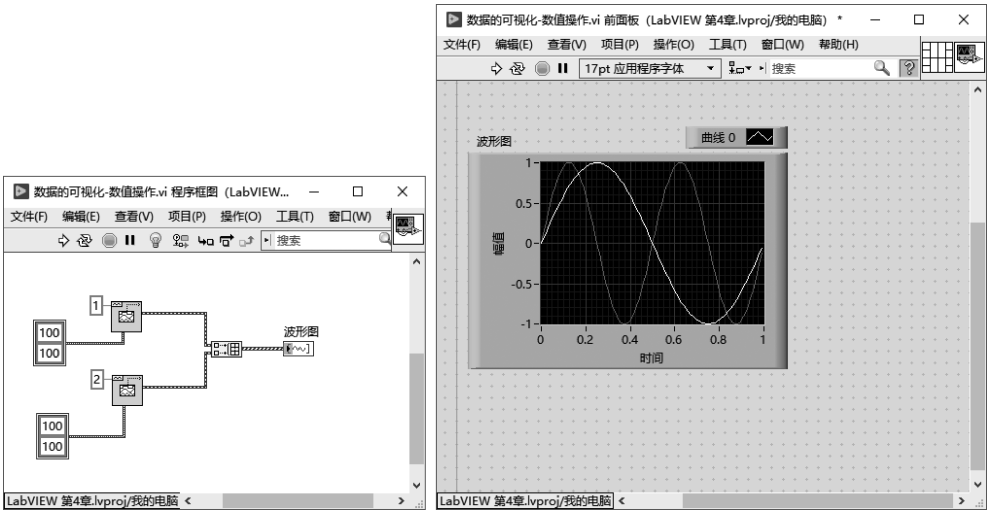

图 5.44　使用波形图显示两条曲线

(5) 通过改变“常用曲线”功能观察波形。波形图可以改变显示波形的方式，如波形线条的样式、粗细、颜色等属性。一般通过图例增强波形的显示。

例如，观察曲线的实际数据点。在波形图中看到的曲线是对原有的数据点进行了平滑和内插的，所以看到的是将各个点连接起来的一条平滑曲线。如果要观察原始的数据点，可以在波形图上右击，在弹出的菜单中选择“常用曲线”→“点”显示方式，如图 5.45 所示。

可以看到实际的数据是一系列离散的数据点，如图 5.46 所示。

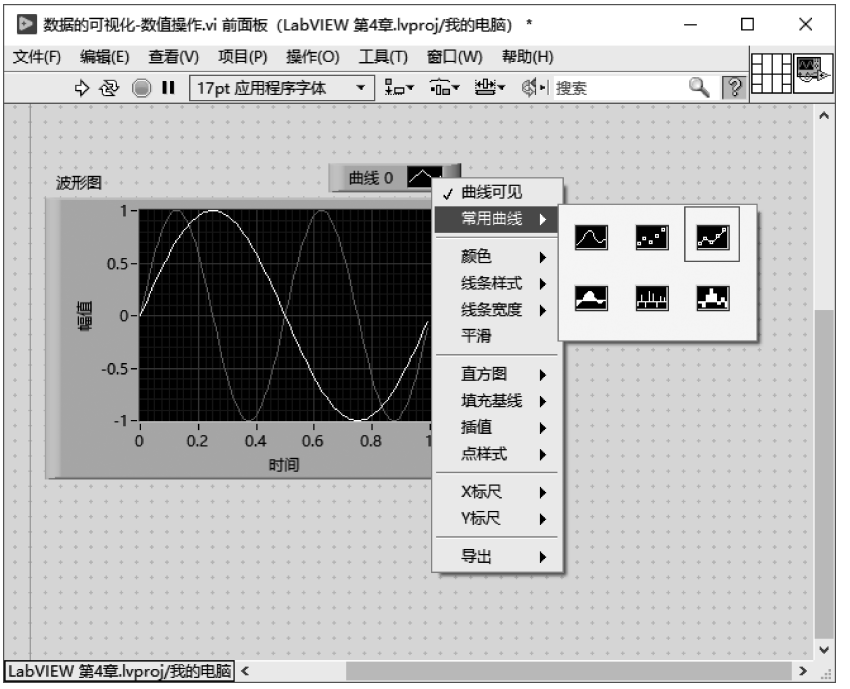

图 5.45　曲线显示样式菜单

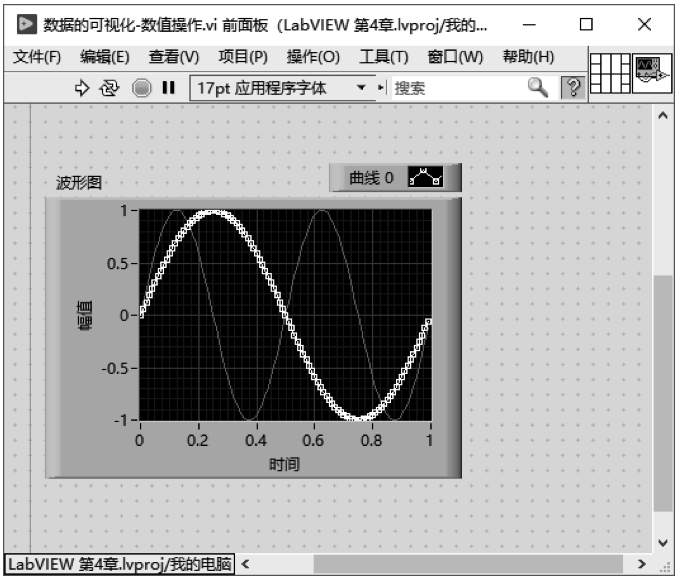

图 5.46　改变曲线显示样式的波形图

（6）通过改变图例中的颜色和线条宽度，可以增强曲线的显示。在波形图上右击，在弹出的菜单中选择“颜色”改变曲线的颜色，如图 5.47 所示。

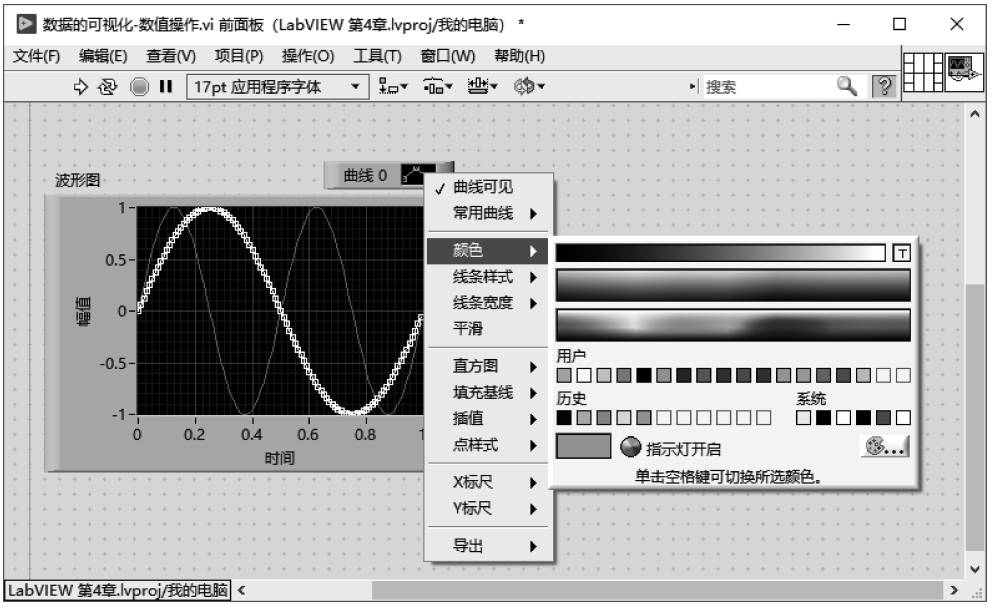

图 5.47　改变曲线颜色

在波形图上右击，在弹出的菜单中选择“线条宽度”改变曲线的宽度，如图 5.48 所示。

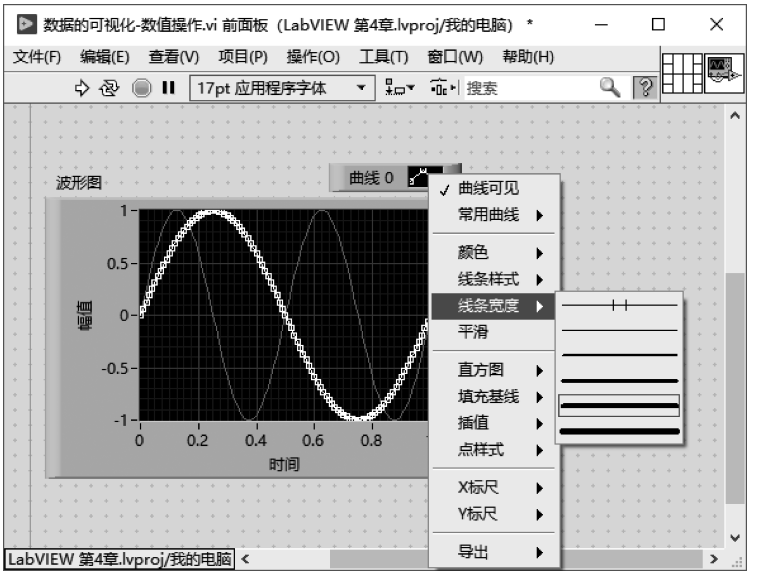

图 5.48　改变线条宽度

修改过颜色和线条宽度的波形如图 5.49 所示。

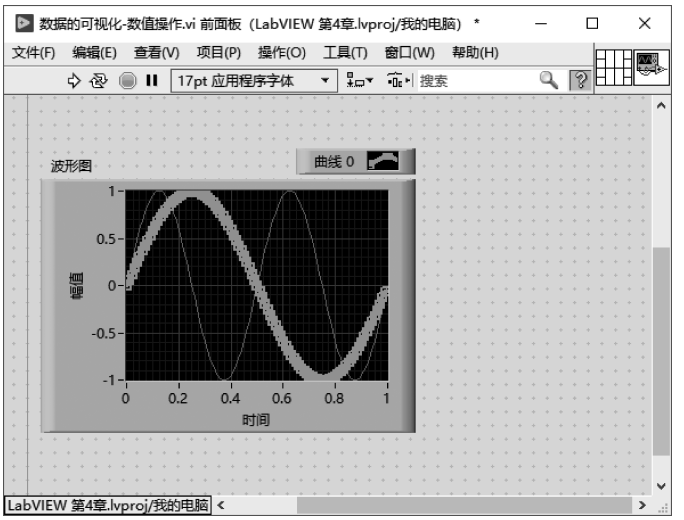

图 5.49　改变曲线显示颜色和宽度的波形图

(7) 通过“图形选板工具”观察波形。当需要对波形的细节进一步观察时,可以通过“图形选板工具”对波形进行放大、平移等操作。

在波形图上右击,在弹出的菜单中选择“显示”→“图形选板工具”,图形工具选板会出现在波形图的下方,可以对波形进行局部的观察,如图 5.50 和图 5.51 所示。

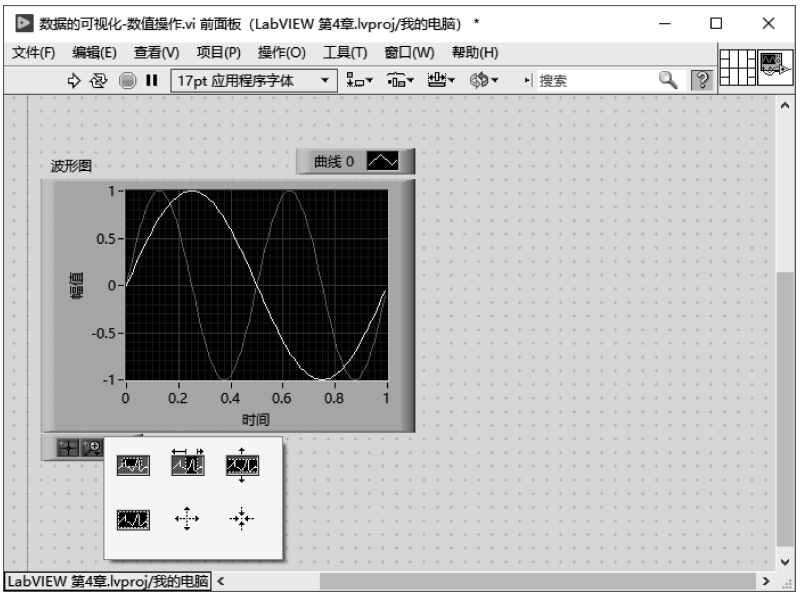

图 5.50　波形图的图形工具选板

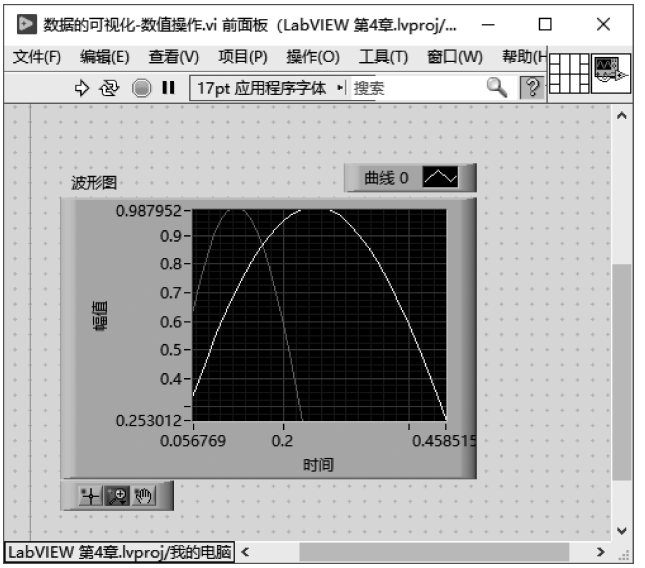

图 5.51 放大显示的波形图

(8) 通过“游标图例”进行波形的测量。在波形图上右击,在弹出的菜单中选择“显示项”→“游标图例”,“游标”窗口会显示在波形图的右侧,可以在“游标”窗口中对波形进行测量。

在“游标”窗口中右击,在弹出的菜单中选择“创建游标”→“单曲线”,如图 5.52 所示。

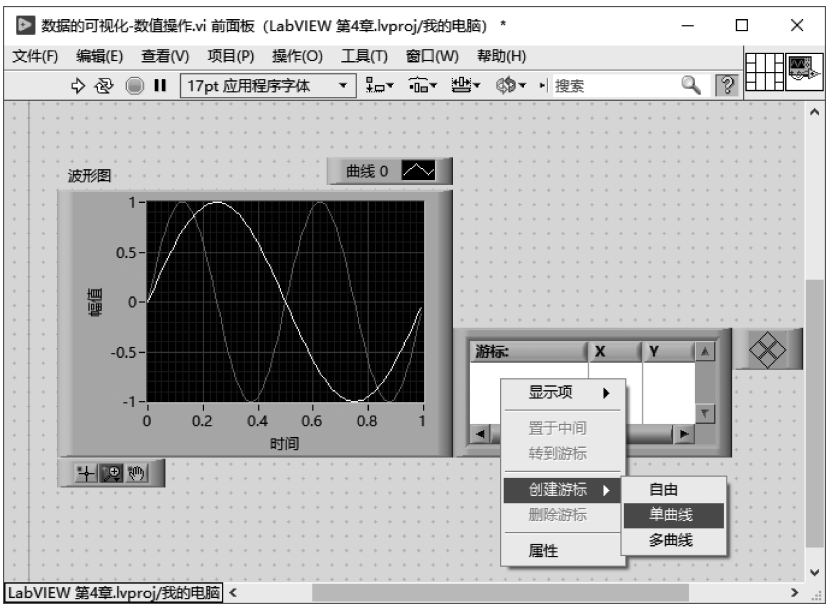

图 5.52 创建单曲线游标

通过两次创建游标，将游标分别放置在频率为 1Hz 和 2Hz 的波形的 1/4 周期处，通过游标的数值可以测量两个波峰的时间差。从游标的显示框中可以看到，选择到的频率为 2Hz 的波形，第一个波峰的坐标为(0.12,0.9980)；频率为 1Hz 的波形的第一个波峰的坐标为(0.25,1)；两个波峰的时间差是 0.25－0.12＝0.13s，这和理论值(1－0.5)/4＝0.125s 基本一致，如图 5.53 所示。

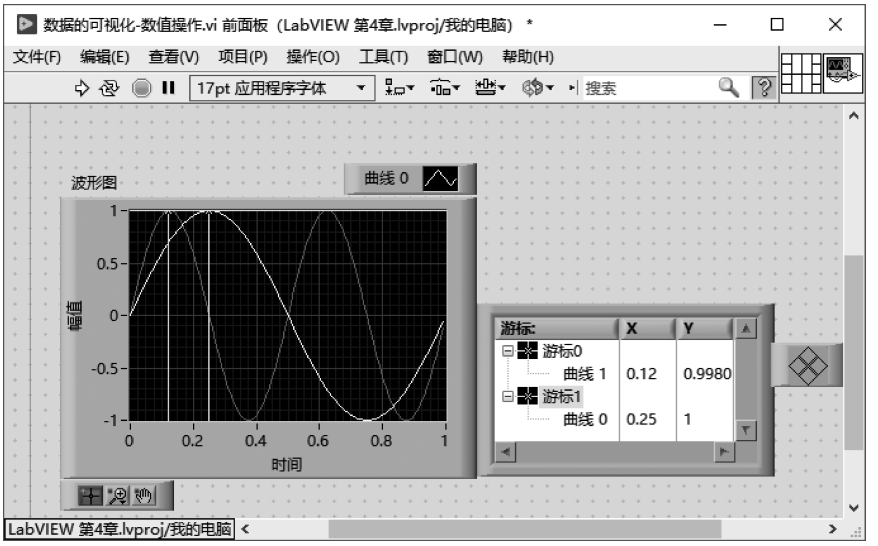

图 5.53　通过游标进行测量

可以注意到，通过游标测量得出的值与理论值有一定的误差。本实例中的误差来源于生成频率为 2Hz 的波形时，采样率和采样点数会导致波形生成的点不够密集，所以使用游标进行测量时，没有办法选择到与理论值一致的点。

如果使用“图形选板工具”放大频率为 2Hz 的波形，会发现这个现象。如图 5.54 所示，在测量中应该选取的点应该为(0.125,1)，但是生成的只有(0.12,0.9980)和(0.13,0.9980)这两个点。

5.3.3　程序框图中的数据流可视化

1. 数据类型的可视化

在程序框图中可以通过数据流的方式直观地看到不同节点和子函数之间数据的传递情况。在程序框图中，不同的数据类型使用不同的颜色和显示形式，通过观察连线，可以更加直观地理解程序的数据流。

在程序框图中，一些典型的数据类型与颜色的关系如下所示。

- 整型：蓝色；
- 浮点型：橘黄色；

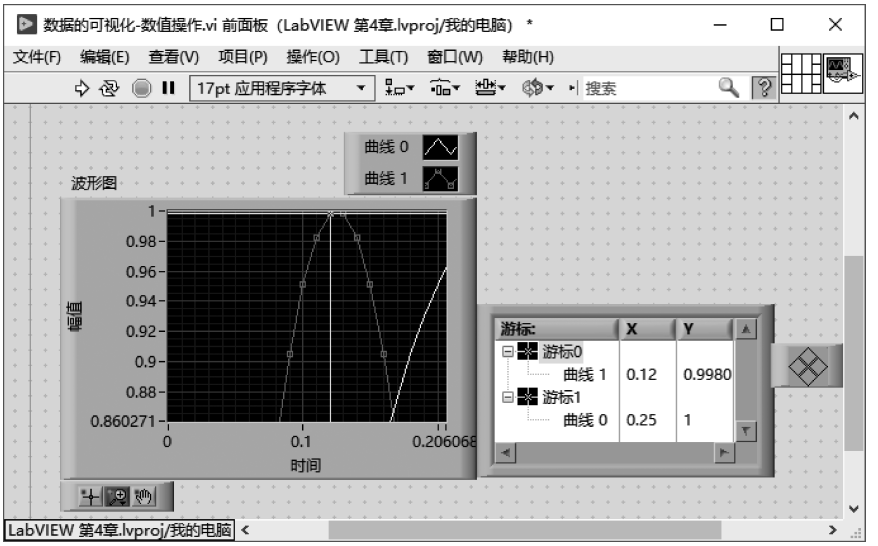

图 5.54 通过“图形选板工具”放大波形

- 布尔型：绿色；
- 字符串型：粉色。

在程序框图中，对于数据流的显示形式，在颜色的基础上通过线条的样式代表复杂的数据的类型格式，如图 5.55 所示。

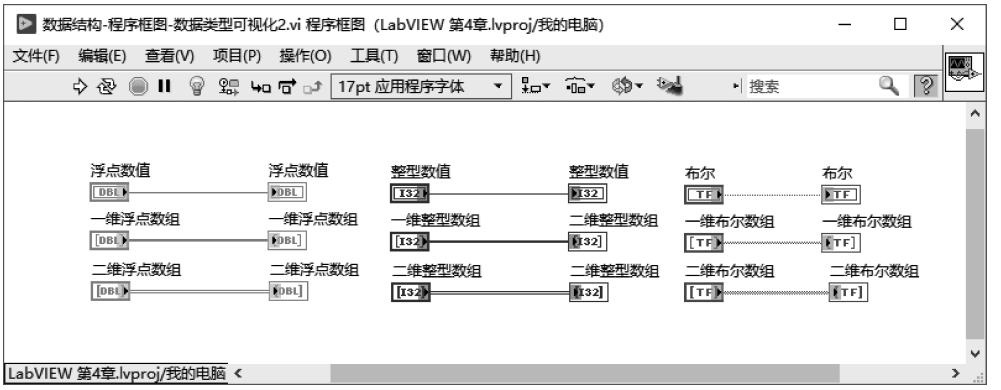

图 5.55 不同维度数组的数据流连线

2. 程序编译状态的可视化

在程序框图中，如果程序编译失败，工具栏中的“运行”按钮会显示为断线状态，同时在出现错误的地方也会有数据流断线的显示。如图 5.56 所示，如果将数值类型和布尔类型通过数据流连接在一起，它们之间的数据流会出现断线，在断线处出现无法连接的标识。

将鼠标指针放置在错误标识上，会出现错误的类型和详细提示信息。

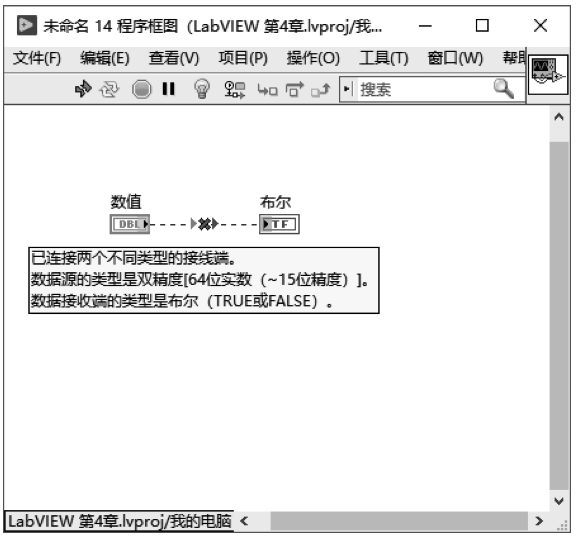

图 5.56　程序框图中编译错误导致的数据流断线