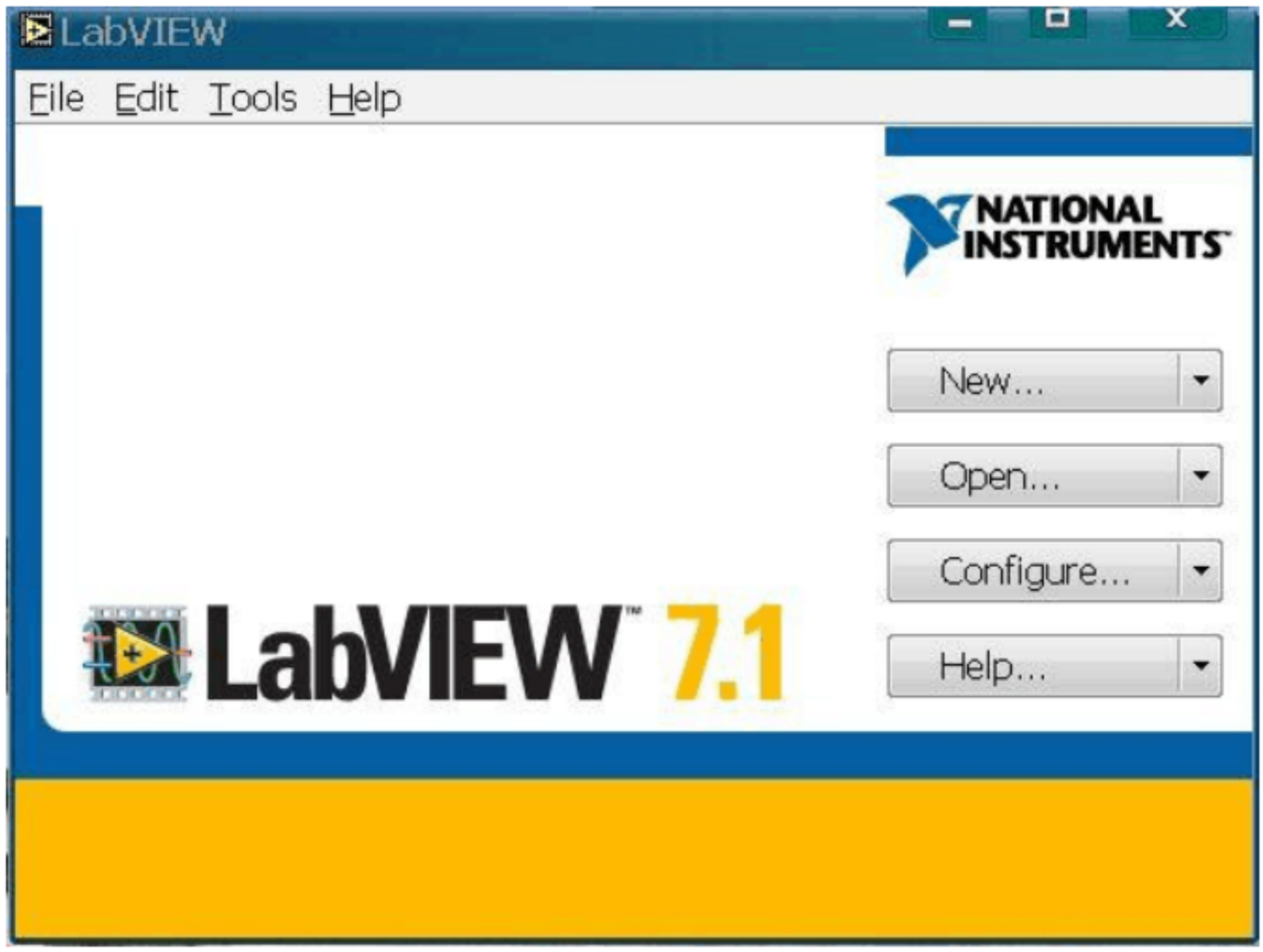
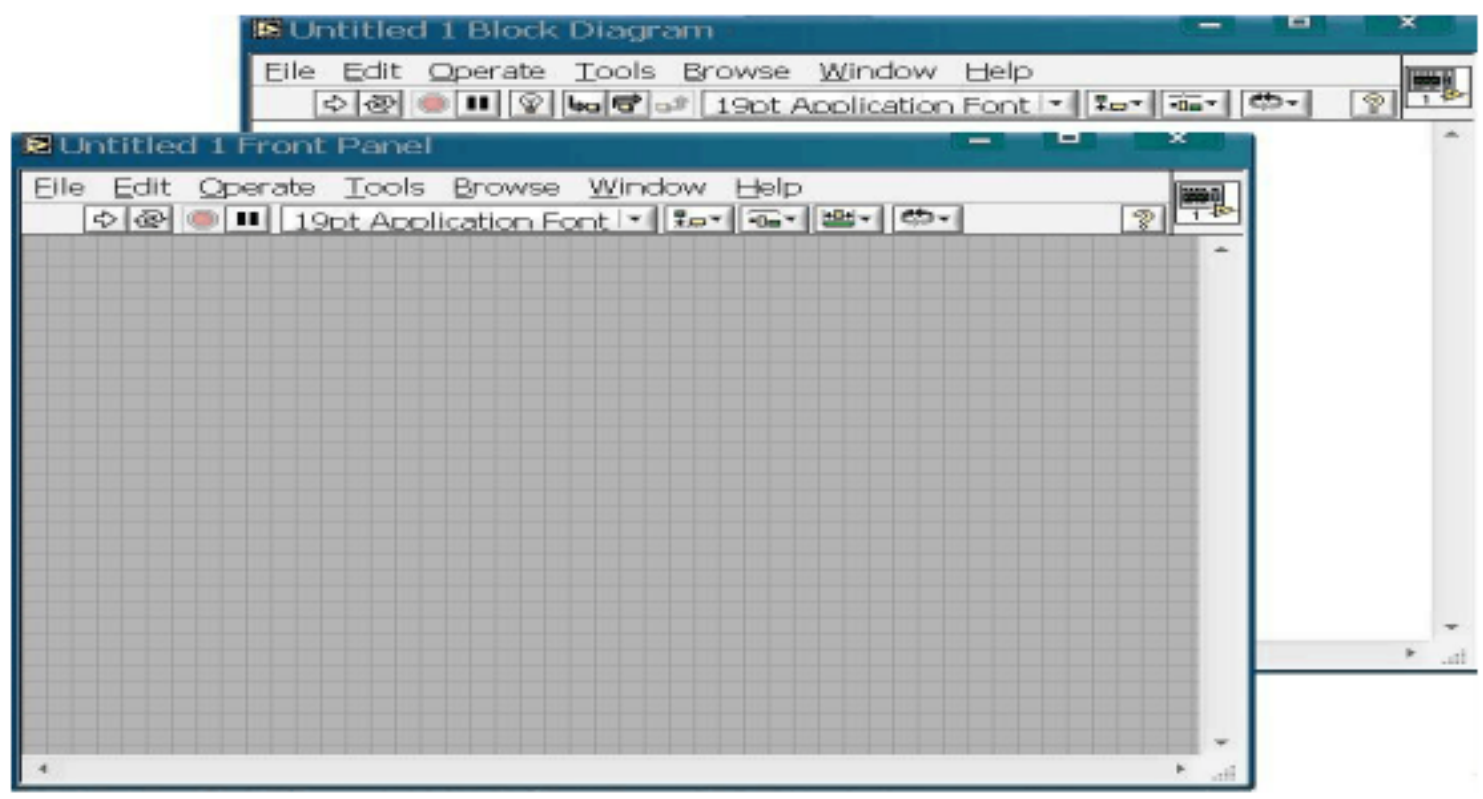
第一讲：认识　Labview

1.1 Labview　简介

在开始菜单里找见　NI Labview7.1　点击打开，会出现如下界面：



从 File>>New VI　或者从右半部分中的　New>>Blank VI　都可以打开如下界面：



上图中前图是虚拟仪器的前面板，　是用户使用的人机界面，　后面的是程序框图界面（即后面板）。

在 LabVIEW的用户界面上，应特别注意它提供的操作模板，包括工具（Tools）模板、控制（ Controls ）模板和函数（ Functions ）模板。这些模板集中反映了该软件的功能与特征。下面我们来大致浏览一下。

工具模板（ Tools Palette ）

该模板提供了各种用于创建、修改和调试 VI 程序的工具。如果该模板没有出现，则可以在 Windows菜单下选择 ShowTools Palette 命令以显示该模板。当从模板内选择了任一种工具后，鼠标箭头就会变成该工具相应的形状。当从 Windows菜单下选择了 Show Help Window 功能后，把工具模板内选定的任一种工具光标放在流程图程序的子程序（ SubVI）或图标上，就会显示相应的帮助信息。

工具图标有如下几种：

	图标	名称	功 能
1		Operate Value（操作值）	用于操作前面板的控制和显示。使用它向数字或字符串控制中键入值时，工具会变成标签工具
2		Position/Size /Select （选择）	用于选择、移动或改变对象的大小。当它用于改变对象的连框大小时，会变成相应形状。
3		Edit Text （编辑文本）	用于输入标签文本或者创建自由标签。当创建自由标签时它会变成相应形状。
4		Connect Wire（连线）	用于在流程图程序上连接对象。如果联机帮助的窗口被打开时，把该工具放在任一条连线上，就会显示相应的数据类型。
5		Object Shortcut Menu（对象菜单）	用鼠标左键可以弹出对象的弹出式菜单。
6		Scroll Windows（窗口漫游）	使用该工具就可以不需要使用滚动条而在窗口中漫游。
7		Set/Clear Breakpoint（断点设置／清除）	使用该工具在 VI 的流程图对象上设置断点。
8		Probe Data（数据探针）	可在框图程序内的数据流线上设置探针。通过控针窗口来观察该数据流线上的数据变化状况。
9		Get Color （颜色提取）	使用该工具来提取颜色用于编辑其他的对象。
10		Set Color （颜色设置）	用来给对象定义颜色。它也显示出对象的前景色和背景色。

下面的两个模板是多层的，其中每一个子模板下还包括多个对象。

控制模板（ Control Palette　）

注意：只有打开前面板时才能调用该模板

该模板用来给前面板设置各种所需的输出显示对象和输入控制对象。每个图标代表一类子模板。如果控制模板不显示，可以用　Windows 菜单的　Show Controls Palette　功能打开它，也可以在前面板的空白处，点击鼠标右键，以弹出控制模板。

控制模板如右图所示，它包括如下所示的一些子模板。子模板中包括的对象，我们在功能中用文字简要介绍。

	图标	子模板名称	功　　能
1		Numeric（数值量）	数值的控制和显示。包含数字式、指针式显示表盘及各种输入框。
2		Boolean（布尔量）	逻辑数值的控制和显示。包含各种布尔开关、按钮以及指示灯等。
3		String & Path（字 符串和 路径）	字符串和路径的控制和显示。
4		Array & Cluster （数组和簇）	数组和簇的控制和显示。
5		List & Table（列表和表格）	列表和表格的控制和显示
6		Graph（图形显示）	显示数据结果的趋势图和曲线图。
7		Ring &Enum(环与枚举）	环与枚举的控制和显示。
8		I/O（输入 / 输出功能）	输入/ 输出功能。于操作　OLE、ActiveX　等功能。
9		Refnum	参考数
10		Digilog Controls（数字控制）	数字控制
11		Clussic Controls（经典控制）	经典控制，指以前版本软件的面板图标。

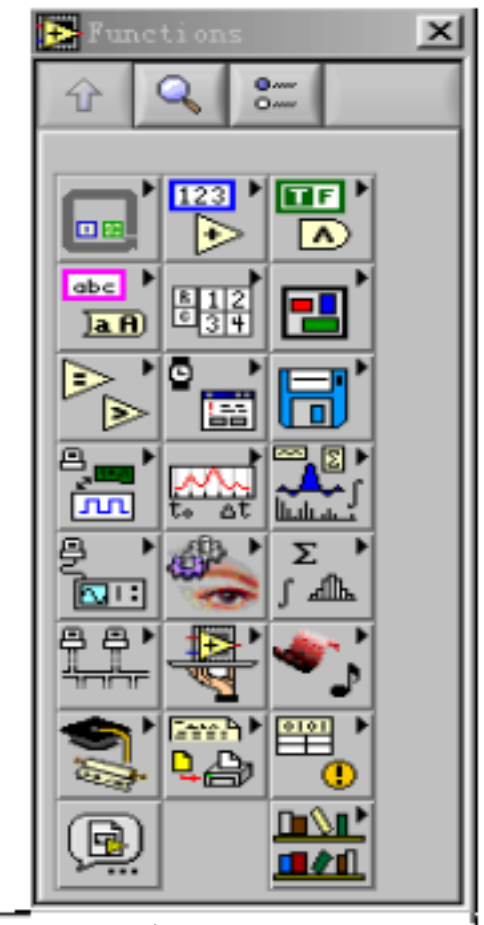
12		Activex	用于 ActiveX 等功能。
13		Decorations（装饰）	用于给前面板进行装饰的各种图形对象。
14		Select a Controls（控制选择）	调用存储在文件中的控制和显示的接口。
15		User Controls（用户控制）	用户自定义的控制和显示。

功能模板 (Functions Palette)

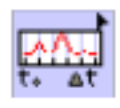
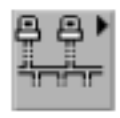
注：只有打开了流程图程序窗口（即后面板） ，才能出现功能模板。

功能模板是创建流程图程序的工具。该模板上的每一个顶层图标都表示一个子模板。若功能模板不出现，则可以用 Windows菜单下的 ShowFunctions Palette 功能打开它，也可以在流程图程序窗口的空白处点击鼠标右键以弹出功能模板。

功能模板如右图所示， 其子模块如下所示。（个别不常用的子模块未包含）



	图标	子模板名称	功 能
1		Structure （结构）	包括程序控制结构命令，例如循环控制等，以及全局变量和局部变量。
2		Numeric（数值运算）	包括各种常用的数值运算，还包括数制转换、三角函数、对数、复数等运算，以及各种数值常数。
3		Boolean（布尔运算）	包括各种逻辑运算符以及布尔常数。
4		String（字符串运算）	包含各种字符串操作函数、数值与字符串之间的转换函数，以及字符 (串) 常数等。
5		Array （数组）	包括数组运算函数、数组转换函数，以及常数数组等。
6		Cluster （簇）	包括簇的处理函数，以及群常数等。这里的群相当于 C语言中的结构。
7		Comparison(比较)	包括各种比较运算函数， 如大于、 小于、 等于。
8		Time & Dialog （时间 和对 话	包括对话框窗口、时间和出错处理函数等。

		框）	
9		File I/O（文件输入/ 输出）	包括处理文件输入 / 输出的程序和函数。
10		Data Acquisition（数据采集）	包括数据采集硬件的驱动，以及信号调理所需的各种功能模块。
11		Waveform（ 波形）	各种波形处理工具
12		Analyze（分析）	信号发生、时域及频域分析功能模块及数学工具 。
13		Instrument I/O（仪器输入 / 输出）	包括 GPIB(488、488.2) 、串行、 VXI 仪器控制的程序和函数，以及 VISA 的操作功能函数。
14		Motion & Vision（运动与景像）	
15		Mathematics（数学）	包括统计、曲线拟合、公式框节点等功能模块，以及数值微分、积分等数值计算工具模块。
16		Communication（通讯）	包括 TCP、DDE、ActiveX 和 OLE等功能的处理模块。
17		Application Control （应用控制）	包括动态调用 VI 、标准可执行程序的功能函数。
18		Graphics & Sound（图形与声音）	包括 3D、OpenGL、声音播放等功能模块。包括调用动态连接库和 CIN 节点等功能的处理模块。
19		Tutorial （示教课程）	包括 LabVIEW示教程序。
20		Report Generation（文档生成）	
21		Advanced(高级功能)	
22		Select a VI（选择子 VI）	
23		User Library（用户子 VI 库）	

下面我们通过练习掌握如何应用 Labview7.1

练习一： 建立一个测量温度和容积的 VI

步骤如下：

1. 选择 FileoNew VI，打开一个新的前面板窗口。
2. 从 Controls>>All Controls>>numeric 中选择 Tank 放到前面板中。（注：如果前面板中没有 Controls 模版，可在菜单栏选 window>>show controls palette, 即可打开或直接点击鼠标右键）
3. 在标签（Tank）文本框中输入“容积” ，然后在前面板中的其他任何位置单击一下。
4. 同样从 Controls>>All Controls>>numeric 中选择 Thermometer 放到前面板中
5. 在标签文本框中输入 “温度计” ，然后在前面板中的其他任何位置单击一下。
6. 把容器显示对象的显示范围设置为 0.0 到 1000.0。
   a. 双击容器坐标的 10.0 标度，使它高亮显示。在坐标中输入 1000，再在前面板中的其他任何地方单击一下。 这时 0.0 到 1000.0 之间的增量将被自动显示。
7. 在容器旁配数据显示。
   将鼠标移到容器上，点右键，在出现的快速菜单中选 Visible Iterms>>Digital Display 即可。
   前面板如下图所示：

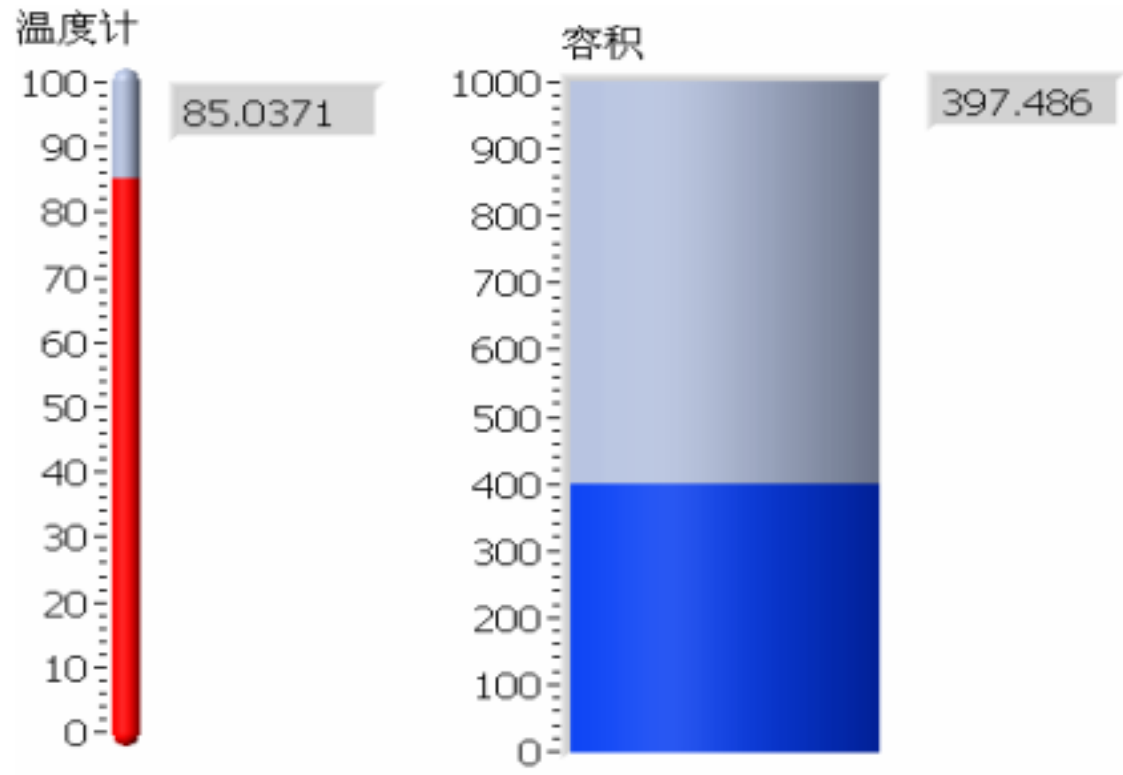


图 1－1 练习一的前面板图

8. Windows>>Show block Diagram 打开流程图窗口，在窗口中建立如下程序：

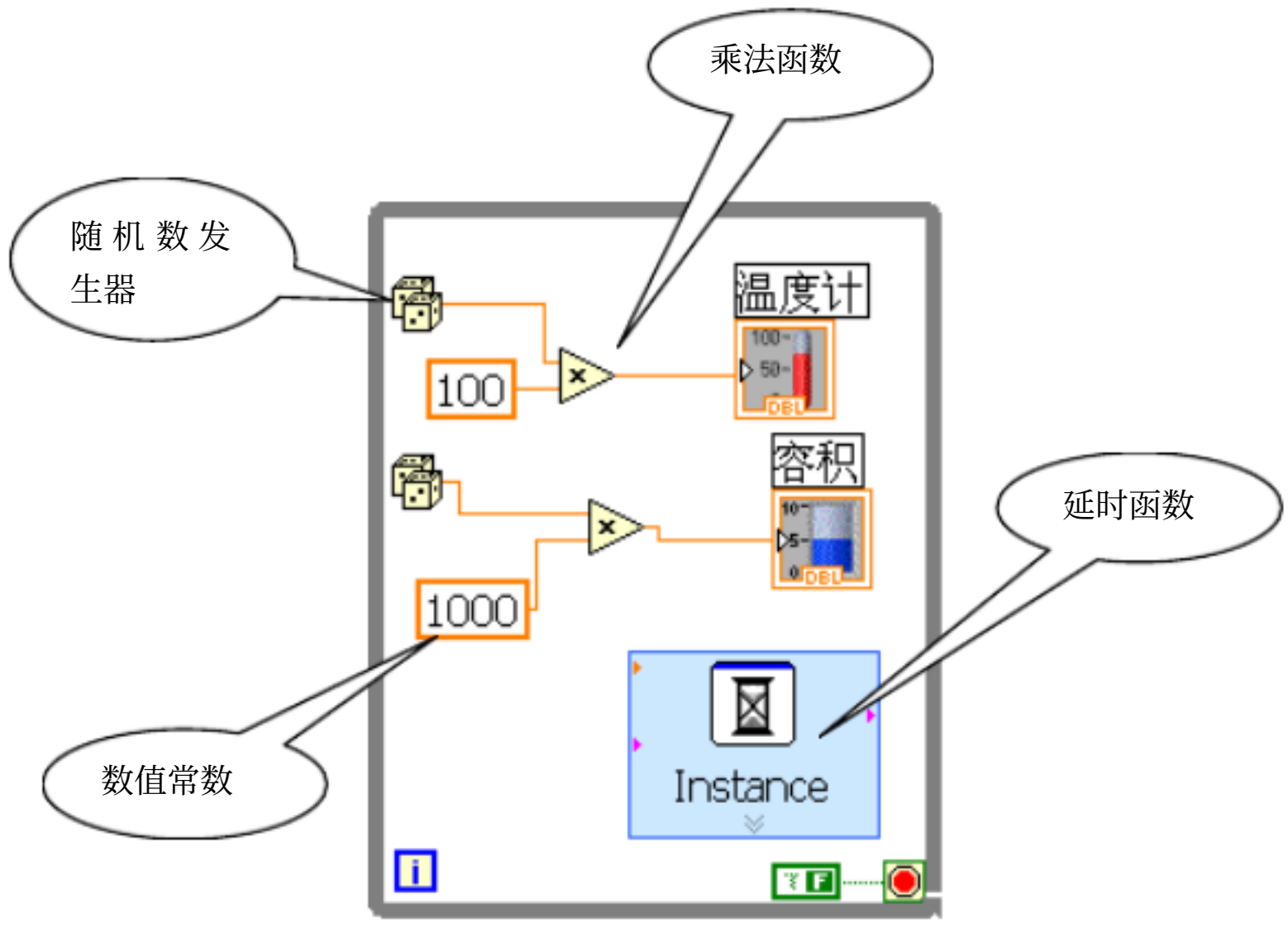


图 1－2　练习一的流程图

该流程图中新增的对象有两个乘法器、两个数值常数、两个随机数发生器、一个延时函数，一个 while 循环，一个布尔常量，温度和容积对象是由前棉板的设置自动带出来的。

a. 乘法器（ Multiply ）、随机数发生器（ Random number (0--1) ）和数值常数（ Numberic constant ）由 Functions>>All Functions>>Numeric 中拖出。(注：如果面板中没有 Functions 模版，可以从菜单栏中选 Window>>Show Functions palette 或点击鼠标右键)

b. 延时函数（ Time delay ）可以从 Functions>>All Functions>>Time&Dialog>>Time delay 中拖出，在自动弹出的对话框中输入要延时的时间（比如 0.5s ）。

9. 连线：把鼠标放到函数端子上，当鼠标自动变为连线模式并出现一闪一闪时，点击一下鼠标左键然后找要连接的函数端子，当鼠标出现一闪一闪时，再次点击鼠标左键，这时就把两个函数连接起来了。

10. 最后选 Functions>>All Functions>>Structures>>While Loop ，在已编好的程序左上角点击鼠标左键，然后移动鼠标，直到出现的虚线把程序全包含起来再点鼠标左键，就创建了一个 While 循环。在右下角条件端子上点击鼠标右键，再弹出的菜单里选 Creat constant 就可，目的是让循环能够维持下去。

11. 在前面板中，单击 Run（运行） 按钮，运行该 VI

12. 在后面板的工具栏中点击类似灯泡状的按钮，就可以看见程序中各个数据流的走向。

13. 选择 FileoSave ，把该 VI 保存到任意目录下。

14. 选择 FileoClose ，关闭该 VI 。

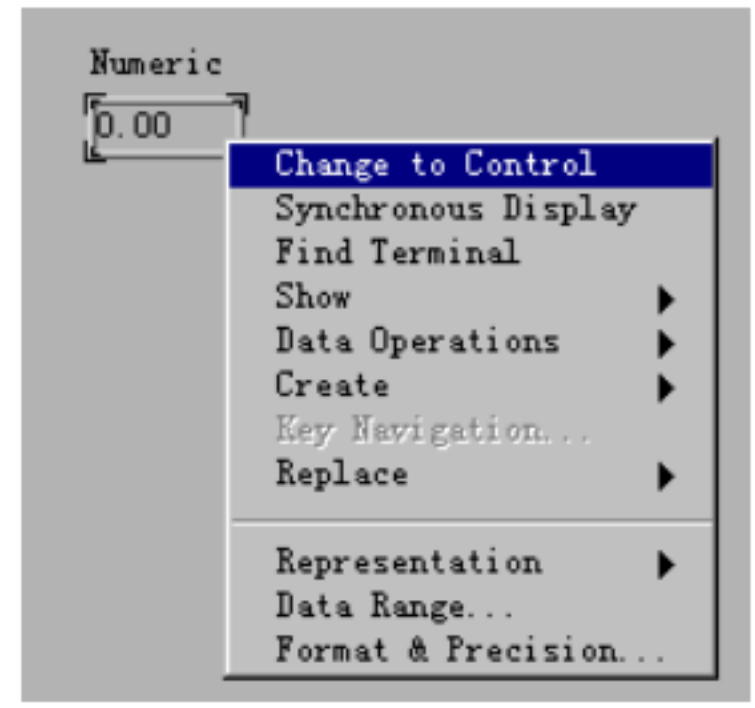
练习一 结束

附注与说明：

1. 显示对象（ Indicator ）、控制对象（ Control ）和数值常数对象

显示对象和控制对象都是前面板上的控件，前者有输入端子而无输出端子，后者正好相反，它们分别相当于普通编程语言中的输出参数和输入参数。 数值常

数对象可以看成是控制对象的一个特例。



在前面板中创建新的控制对象或显示对象时，LabVIEW都会在流程图中创建对应的端子。端子的符号反映该对象的数据类型。 例如，DBL 符号表示对象数据类型是双精度数； TF 符号表示布尔数；I16 符号表示 16 位整型数；ABC符号表示对象数据类型是字符串。

一个对象应当是显示对象还是控制对象必须弄清楚，否则无法正确连线。 有时他们的图标是相似或相同的，可以根据需要明确规定它是显示对象还是控制对象。 方法是将鼠标移到图标上，然后点右键， 可出现快速菜单 （例见右图）。如果菜单中的第一项是 Chang to Control ，说明这是一个显示对象，你可以根据需要，将其变为控制对象。如果菜单中的第一项是 Changto Indicator ，说明这是一个控制对象，你也可以根据需要，将其变为显示对象。

2. 关于连线

连线是程序设计中较为复杂的问题。 流程图上的每一个对象都带有自己的连线端子， 连线将构成对象之间的数据通道。 因为这不是几何意义上的连线， 因此并非任意两个端子间都可连线，连线类似于普通程序中的变量。数据单向流动，从源端口向一个或多个目的端口流动。 不同的线型代表不同的数据类型。 下面是一些常用数据类型所对应的线型和颜色：

类型	颜色	标量	一维数组	二维数组
整形数	兰色			
浮点数	橙色			
逻辑量	绿色			
字符串	粉色			
文件路径	青色			

当需要连接两个端点时，在第一个端点上点击连线工具（从工具模板栏调用），然后移动到另一个端点，再点击第二个端点。端点的先后次序不影响数据流动的方向。

当把连线工具放在端点上时， 该端点区域将会闪烁， 表示连线将会接通该端点。当把连线工具从一个端口接到另一个端口时， 不需要按住鼠标键。 当需要连线转弯时， 点击一次鼠标键， 即可以正交垂直方向地弯曲连线， 按空格键可以改变转角的方向。

接线头是为了帮助正确连接端口的连线。 当把连线工具放到端口上， 接线头就会弹出。接线头还有一个黄色小标识框，显示该端口的名字。

线型为波折号的连线表示坏线。 出现坏线的原因有很多， 例如：连接了两个控制对象；源端子和终点端子的数据类型不匹配 （例如一个是数字型， 而另一个是布尔型）。可以通过使用定位工具点击坏线再按下 <Delete> 来删除它。选择 EditoRemove BadWires 或者按下 <Ctrl-B> 可以一次删除流程图中的所有坏线。当 VI 无法运行，或者显示 Signal has Loose Ends （信号丢失终端）的错误

信息时，这是一个快捷的调试方法。

程序调试技术

（以下操作大家可以用练习一尝试）

1. 找出语法错误

如果一个 VI 程序存在语法错误，则在面板工具条上的运行按钮会变成一个折断的箭头，表示程序不能被执行。这时该按钮被称作错误列表。点击它，则 LabVIEW弹出错误清单窗口， 点击其中任何一个所列出的错误， 选用 Find 功能，则出错的对象或端口就会变成高亮。

2. 设置执行程序高亮

在 LabVIEW的工具条上有一个画着灯泡的按钮，这个按钮叫做“ 高亮执行 ”按钮上。 点击这个按钮使它变成高亮形式， 再点击运行按钮， VI 程序就以较慢的速度运行，没有被执行的代码灰色显示，执行后的代码高亮显示，并显示数据流线上的数据值。这样，你就可以根据数据的流动状态跟踪程序的执行。

3. 断点与单步执行

为了查找程序中的逻辑错误，有时希望流程图程序一个节点一个节点地执行。使用断点工具可以在程序的某一地点中止程序执行，用探针或者单步方式查看数据。使用断点工具时，点击你希望设置或者清除断点的地方。断点的显示对于节点或者图框表示为红框，对于连线表示为红点。当 VI 程序运行到断点被设置处，程序被暂停在将要执行的节点，以闪烁表示。按下单步执行按钮，闪烁的节点被执行，下一个将要执行的节点变为闪烁，指示它将被执行。你也可以点击暂停按钮，这样程序将连续执行直到下一个断点。

4. 探针

可用探针工具来查看当流程图程序流经某一根连接线时的数据值。从 Tools 工具模板选择探针工具，再用鼠标左建点击你希望放置探针的连接线。这时显示器上会出现一个探针显示窗口。 该窗口总是被显示在前面板窗口或流程图窗口的上面。在流程图中使用选择工具或连线工具，在连线上点击鼠标右键，在连线的弹出式菜单中选择“ 探针 ”命令，同样可以为该连线加上一个探针。

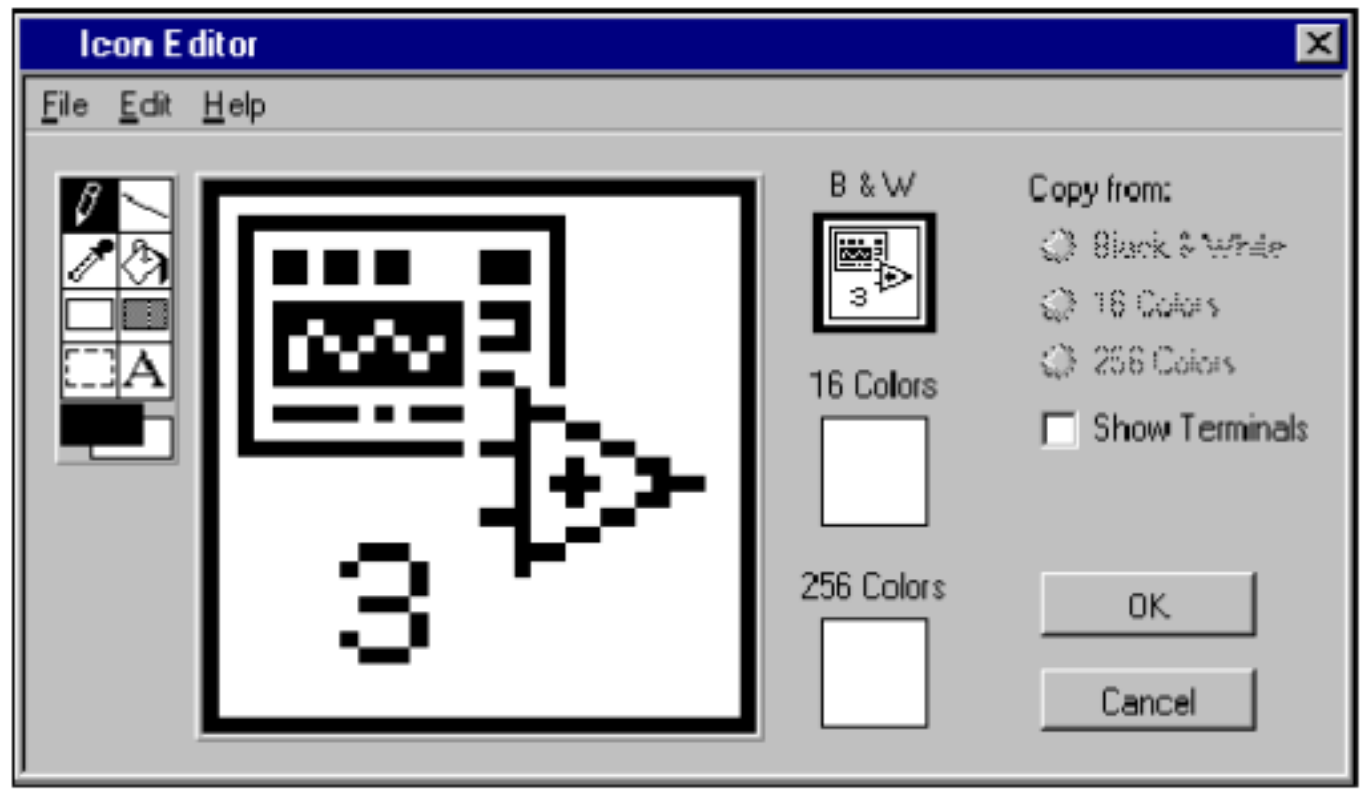
1.2 子 VI 的建立

子 VI （SubVI）相当于普通编程语言中的子程序，也就是被其他的 VI 调用的 VI。可以将任何一个定义了图标和联接器的 VI 作为另一个 VI 的子程序。在流程图中打开 FunctionsoSelect a VI,. ，就可以选择要调用的子 VI 。构造一个子 VI 主要的工作就是定义它的 图标和联接器 。

每个 VI 在前面板和流程图窗口的右上角都显示了一个默认的图标。启动图标编辑器的方法是， 用鼠标右键单击面板窗口的右上角的默认图标， 在弹出菜单中选择 Edit Icon 。

下图显示了图标编辑器的窗口。 可以用窗口左边的各种工具设计像素编辑区

中的图标形状。 编辑区右侧的一个方框中显示了一个实际大小的图标。 图标编辑器的具体使用细节参阅 练习二



图标编辑器窗口

联接器是 VI 数据的输入输出接口。 如果用面板控制对象或者显示对象从子 VI 中输出或者输入数据， 那么这些对象都需要在联接器面板中有一个连线端子。您可以通过选择 VI 的端子数并为每个端子指定对应的前面板对象以定义联接器。

定义联接器的方法是，用鼠标右键单击面板窗口中的图标窗口，在快捷菜单中选择 Show Connector。

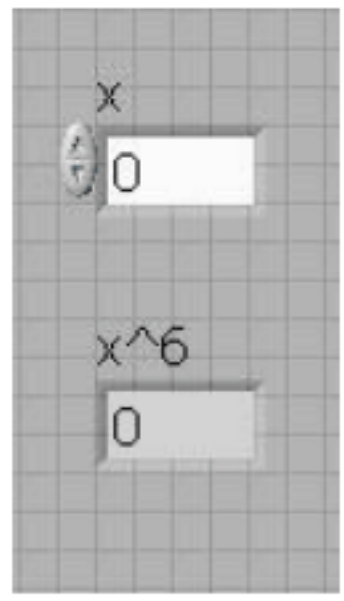
联接器图标会取代面板窗口右上角的图标。 LabVIEW 自动选择的端子连接模式是控制对象的端子位于联接器窗口的左边， 显示对象的端子位于联接器窗口右边。选择的端子数取决于前面板中控制对象和显示对象的个数。

联接器中的各个矩形表示各个端子所在的区域， 可以用它们从 VI 中输入或者输出数据。 如果必要， 也可以选择另外一种端子连接模式。 方法是在图标上单击鼠标右键单出快捷菜单，选择 Show Connector，再次弹出快捷菜单，选择 Patterns 。下面我们通过一个练习说明具体操作。

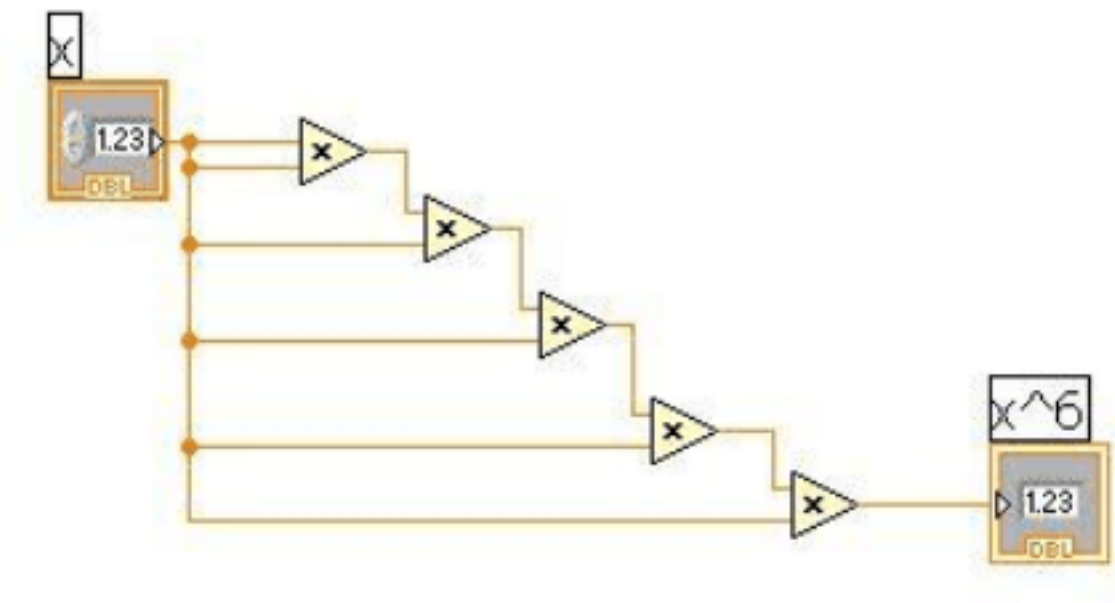
练习二：子 VI 的建立及调用

步骤如下：

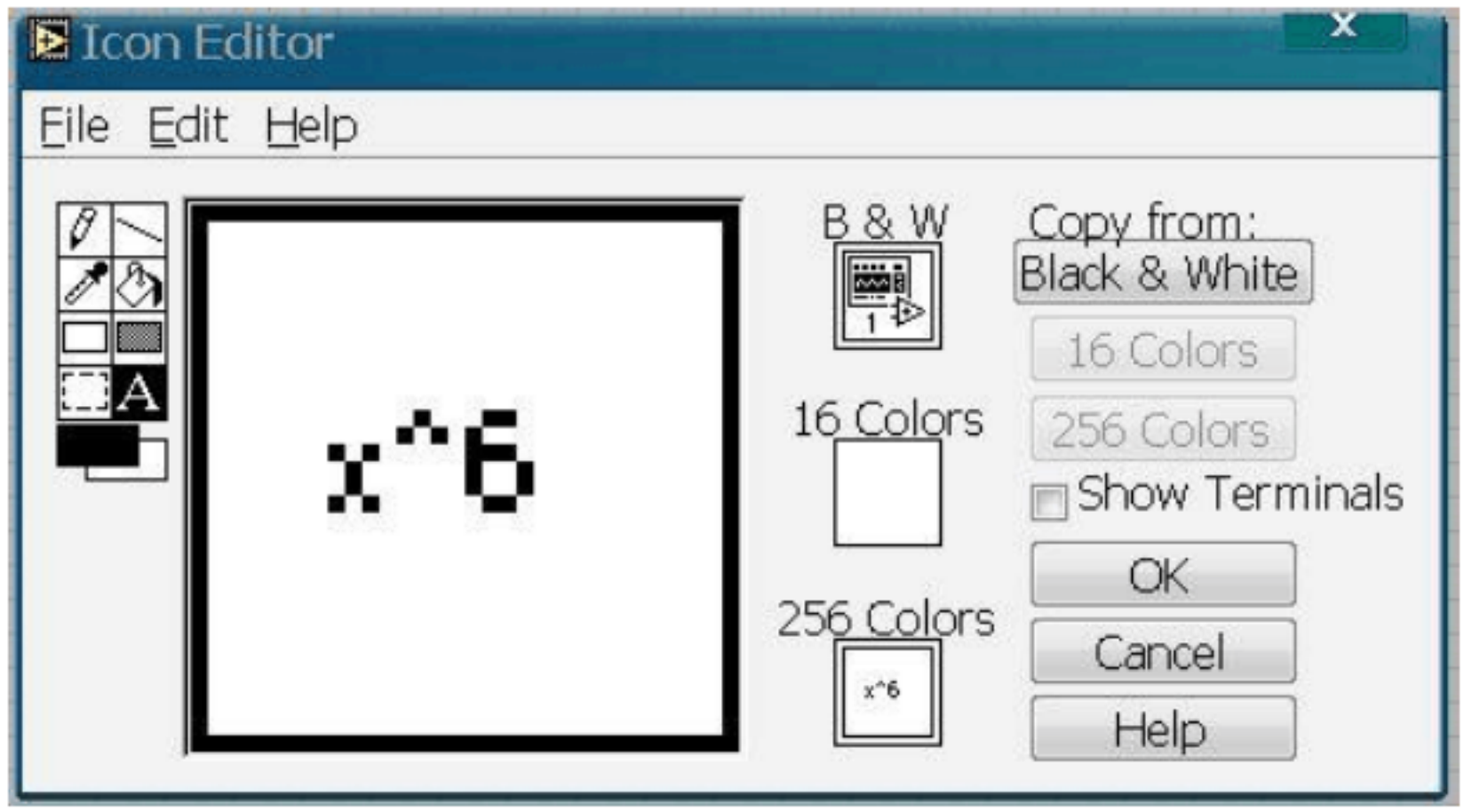
1. 打开一个 NEW VI
2. 在前面板中放置一个 Number control 控件 (数据输入控件) 和一个 Number Indicator 控件（数据显示）， 都在 All controls>>numberic 模板下并把标签分别改为 X 和 X^6, 如下图：



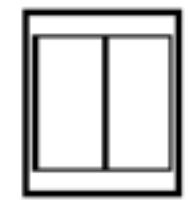
3. 后面板程序图如下图连接，这是一个计算 X 的 6 次方程序

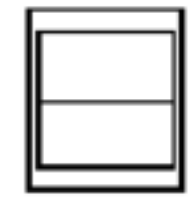


4. 在前面板中，用鼠标右键单击窗口右上角的图标，在快捷菜单中选择 **Edit Icon**…，也可以双击图标激活图标编辑器。 注意 只能在 前面板 中编辑图标和联接器。
5. 删除默认图标。使用 **Select Tool**（矩形框），单击并拖动想要删除的部分，按下**<Delete>**。也可以通过双击工具框中的阴影矩形删除图标。
6. 用 **Text Tool**（文本工具） A 创建文本（写好后可以用方向键移动文字在图标中的位置）。得到图标将如下图所示。

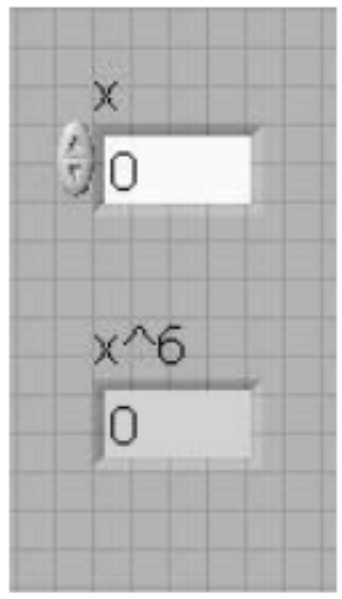


7. 单击 OK，关闭编辑器。新创建的图标就显示在屏幕右上角的图标窗口中。
8. 用鼠标右键单击前面板中的图标窗口，在快捷菜单中选择 Show Connector，设置联接器端子连接模式。在默认情况下，LabVIEW会根据前面板中的控制对象和显示对象的数目确定联接器的端子连接模式。因为前面板中有两个对象，所以联接器有两个端子，如右图所示。用鼠标右键单击联接器窗口，在快捷菜单中选择 Rotate 90 Degrees（旋转 90 度），注意联接器窗口的变化，如右图所示。

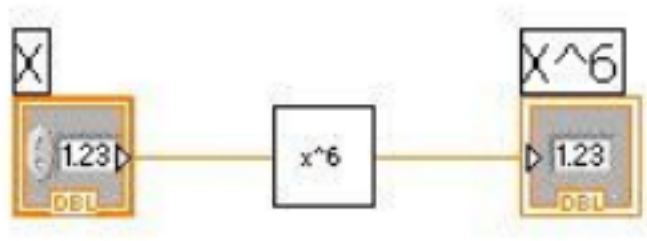




9. 端子连接到 X 和 X^6：
   a. 点击联接器左部端子。光标自动变成连线工具，同时端子变成黑色。
   b. 单击 X 控件对象。一个移动的虚线框把它包围起来，选中的端子的颜色变为与控制 / 显示对象的数据类型一致的颜色。
   如果单击前面板中的任何空白区域以后，虚线消失，选中的端子变暗，这表示您已经成功地把显示对象和上部端子连接起来。如果端子是白色，则表示没有连接成功。 c. 重复步骤 a 和 b，把右部的端子和 X^6 连接起来。
   c. 用鼠标右键单击联接器，在快捷菜单中选择 Show Icon .
10. 选择 FileoSave ，保存该 VI，保存名为 X^6。这样这个 VI 就完成了，并也可以作为子 VI 被其他的 VI 调用。子 VI 的图标在主 VI 的流程图中代表它。
11. 调用子 VI。新建一个 New VI
12. 在前面板中放置一个 Number control 控件和一个 Number Indicator 控件，并把标签分别改为 X 和 X^6, 如下图



13. 后面板操作。在 Functions>>All Functions>>Select a VI ，选中并打开刚才保存的 VI，并连接好线如下图：



14. 运行该程序

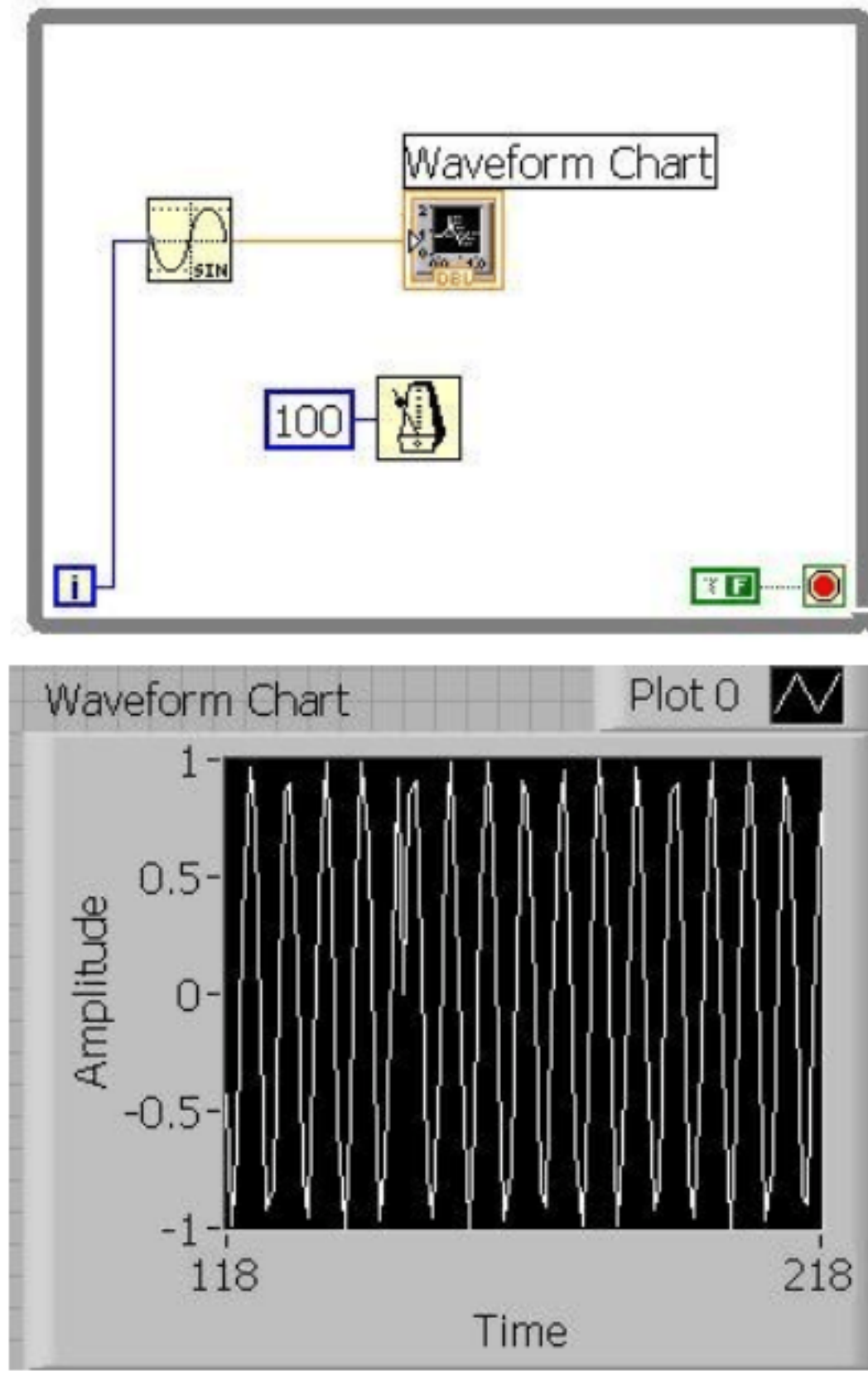
练习二 结束

1.3 使用三种图表模式

目的：查看　VI　分别在三种模式下执行时图表的显示。

练习三：

建立前面板及流程图如下



步骤：

1. 在 前 面 板 放 置　chart　（ Controls>>All　Controls>>Graph>>Waveform chart ）控件

2. 在后面板放置　Sine　函数（ Functions>>All Functions>>Numberic>> Trigonometric>>sine　）及　Wait　Until　Next　ms　Multiple　函 数（Functions>>All Functions>>Time&Dialog>>　Wait Until Next ms Multiple　）并在左端子建立常数如　100。最后放置　while　循环

该程序中利用一个　while　循环产生连续的　sin(i)　函数值，并及时地在　chart　图表上显示出来，　现在前面板上的　chart　是一个 strip　，这是一个坐标式显示器，与纸带式图表记录器相似。　每接受一个新数据，　新数据就将显示在右侧，　而原有数据移动到左侧

3. 用鼠标选中 chart，点击右键，可在快速菜单中选择　AdvancedoUpdate Mode

子菜单。可以选择更换其他两种更新模式。

示波器模式是一个返回式的显示器，与示波器类似。每接受一个新数据时，它就把新数据绘制在原有数据的右侧。 当数据曲线到达显示区的右边缘时， VI 会删除全部图形，从左边缘重新开始绘制曲线。 示波器模式显然要快于条状图模式，因为它不会因为滚动产生溢出。

扫描模式更接近于示波器模式， 但是当数据曲线到达显示区的右边时， 不会变成空白，而是会出现一个移动的垂线，标记新数据的开始，并当 VI 添加新数据时穿过整个显示区。

练习三 结束。

第二讲 程序结构

循环结构

While 循环

While 循环可以反复执行循环体的程序， 直至到达某个边界条件。 它类似于普通编程语言中的 Do 循环和 Repeat-Until 循环。While 循环的框图是一个大小可变的方框， 用于执行框中的程序， 直到条件端子接收到的布尔值为 FALSE。

该循环有如下特点：

计数从 0 开始（i=0 ）。

先执行循环体，而后 i+1 ，如果循环只执行一次，那么循环输出值 i=0 。

循环至少要运行一次。

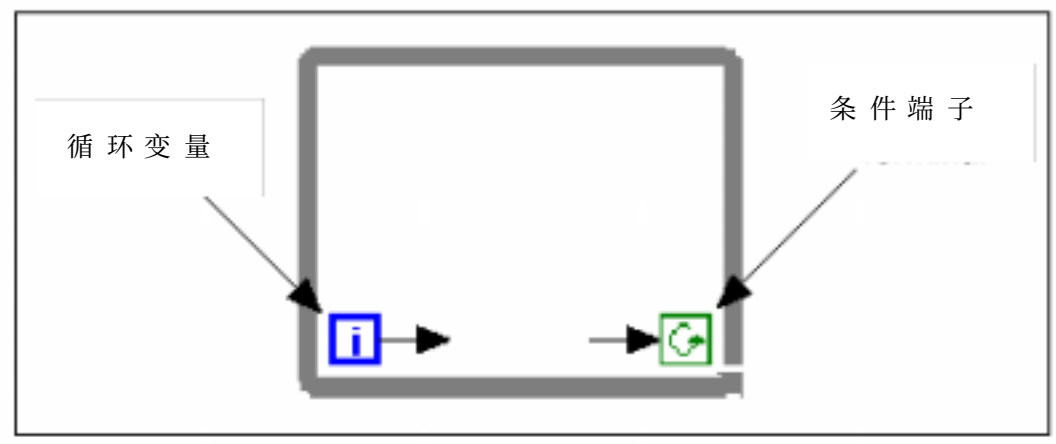
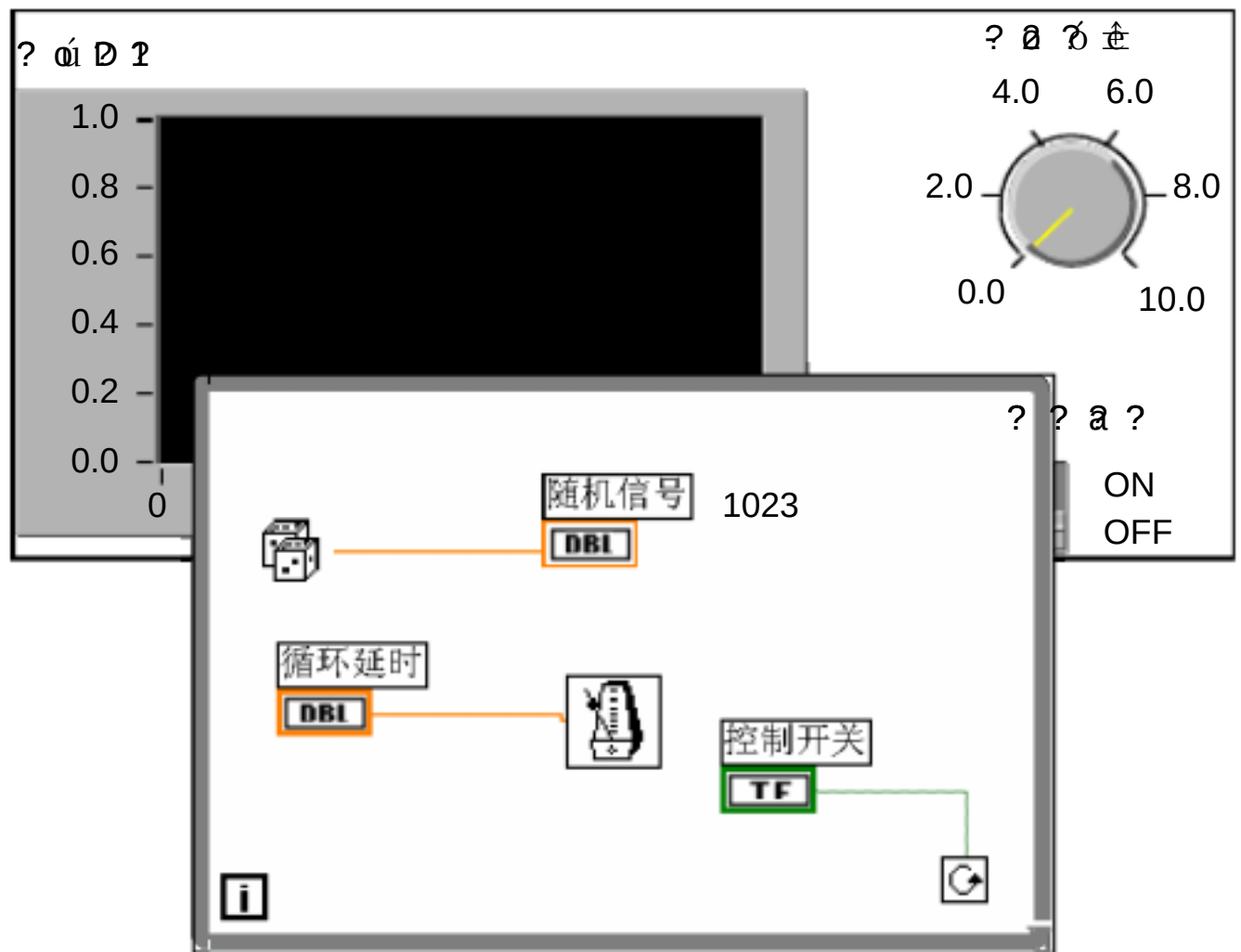


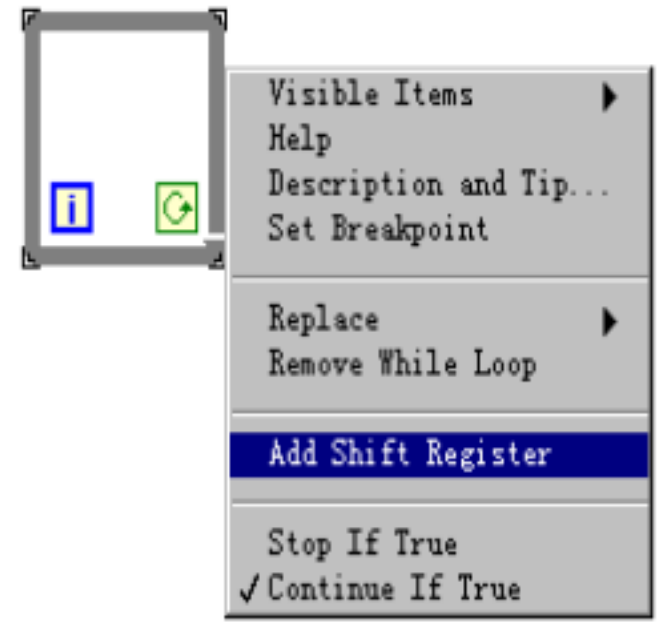
图 4－1 While 循环示意图

由于在练习一中已讲过 While 循环的应用， 大家可以自己做下面的练习， 其前面板和流程图如下图所示：



2.1.1.1 移位寄存器（ Shift Register ）

移位寄存器可以将数据从一个循环周期传递到另外一个周期。 在程序设计中， 经常要用到它．创建一个移位寄存器的方法是，用鼠标右键单击循环的左边或者右边，在快捷菜单中选择 **Add Shift Register**。如右图所示。



移位寄存器在流程图上用在循环边框上相应的一对端子来表示。右边的端子中存储了一个周期完成后的数据，这些数据在这个周期完成之后将被转移到左边的端子，赋给下一个周期。移位寄存器可以转移各种类型的数据——数值、布尔数、数组、字符串等等。它会自动适应与它连接的第一个对象的数据类型。下图表示了它的工作过程.

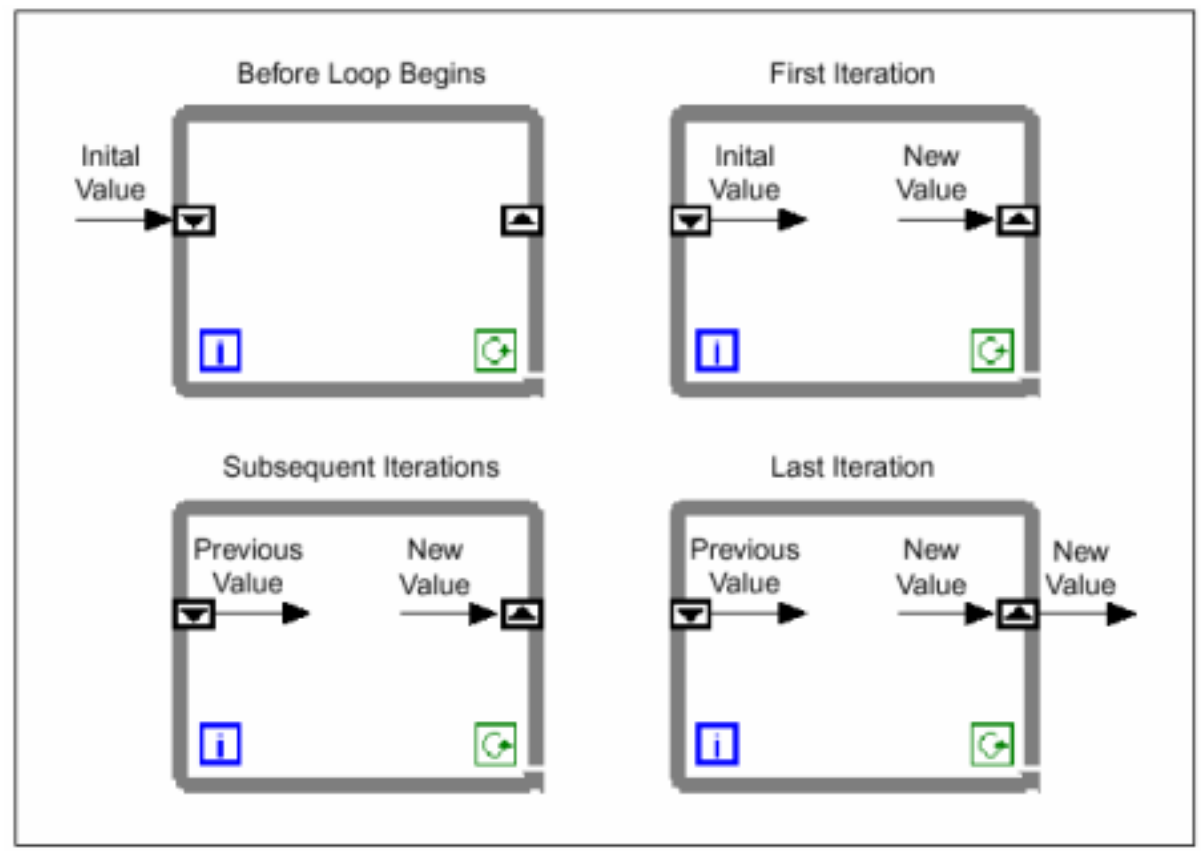


图 2－3 Shift Register 的工作过程

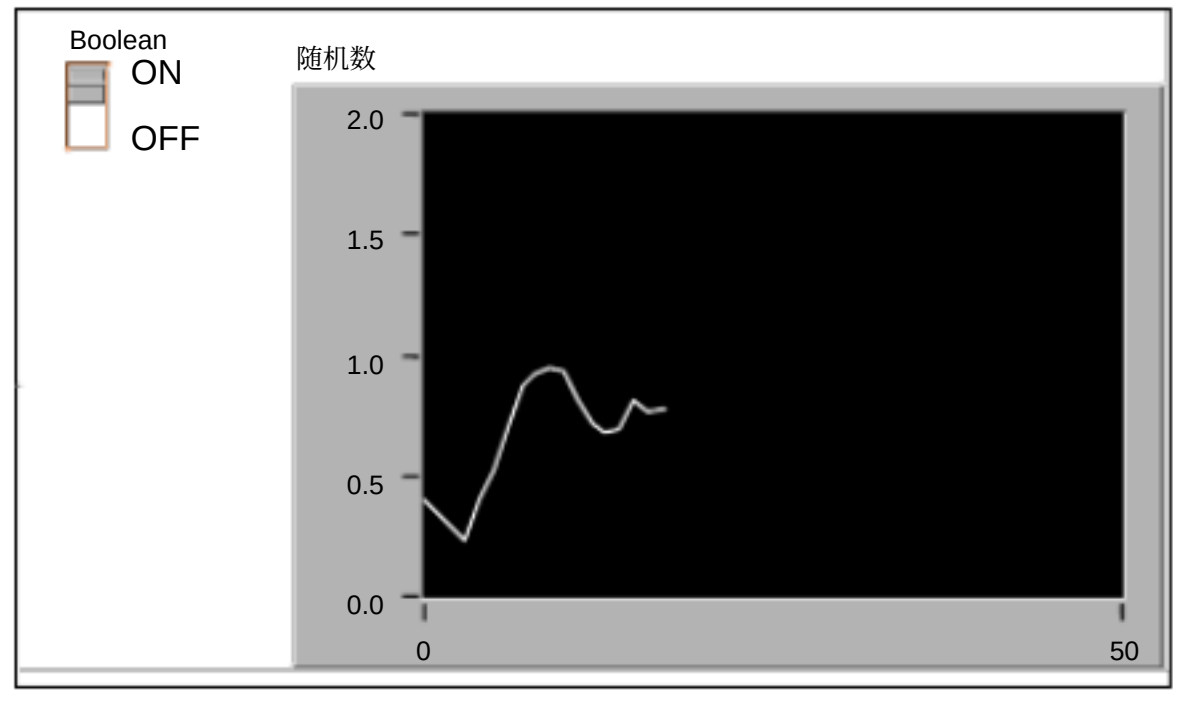
可以令移位寄存器记忆前面的多个周期的数值。 这个功能对于计算数据均值非常有用。还可以创建其他的端子访问先前的周期的数据， 方法是用鼠标右键单击左边或者右边的端子，在快捷菜单中选择 Add Element。例如，如果某个移位寄存器左边的端口含有三个元素，那么就可以访问前三个周期的数据。

练习四　使用移位寄存器

目的：创建一个可以在图表中显示运行平均数的　VI。

前面板

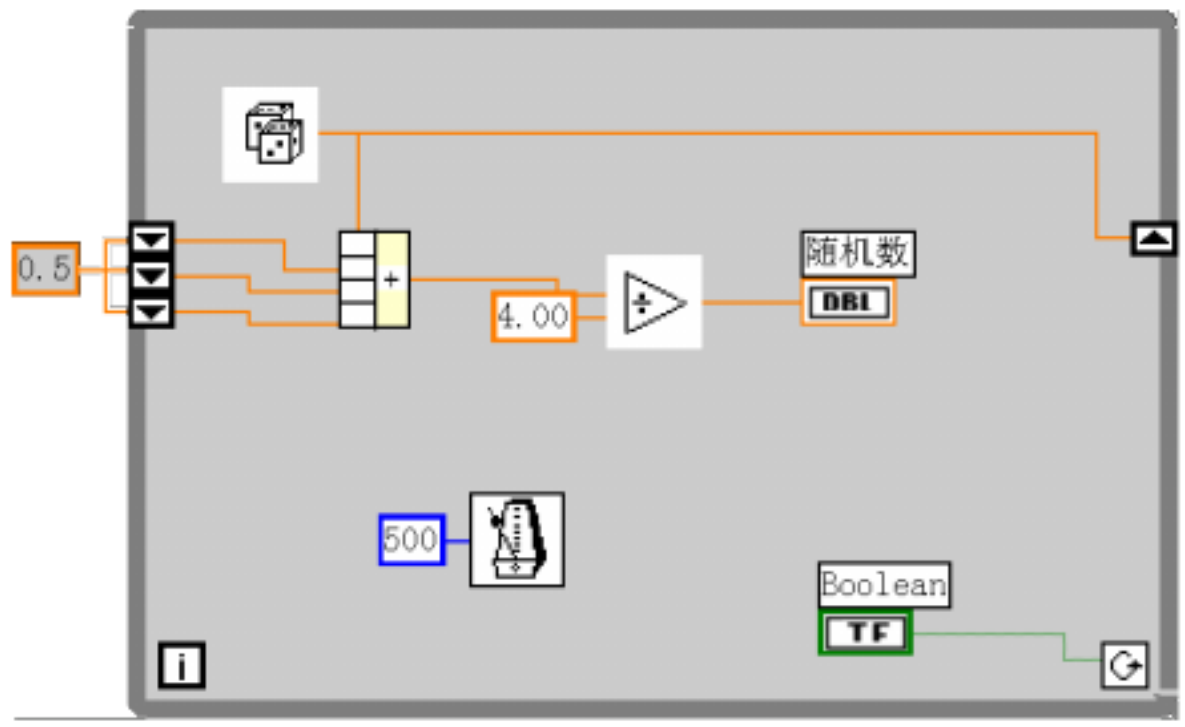
1. 打开一个新的前面板，按照下图所示创建对象。
2. 把波形图表的坐标范围改为　0.0 到 2.0 。
3. 在添加开关之后，用鼠标右键单击它，在快捷菜单中选择　Mechanical ActionoLatch When Pressed　，再选择 OperateoMake Current Values Default　，把 ON状态设置为默认状态。



练习四　的前面板

流程图

1. 按下图创建流程图。



练习四　的流程图

2. 在流程图中添加　While　循环 (FunctionsoStructures　) ，创建移位寄存器。
   a. 用鼠标右键单击　While　循环的左边或者右边，在快捷菜单中选择　Add Shift Register　。
   b. 用鼠标右键单击寄存器的左端子，在快捷菜单中选择　AddElement，添加一个寄存器。用同样的方法创建第三个元素。
3.Random Number (0 – 1) 函数（ FunctionsoNumeric ）——产生　0 到 1 之间的某个随机数。
4.Compound Arithmetic　函数（ FunctionsoNumeric ）——在本练习中，它将返

回两个周期产生的随机数的和。 如果要加入其他的输入， 只需用右键单击某个输入，从快捷菜单中选择 Add Input 。

5. 除法函数（ FunctionsoNumeric ）——在本练习中，它用于返回最近四个随机数的平均值。

6. 数值常数（FunctionsoNumeric ）——在 While 循环的每个周期， RandomNumber (0 – 1) 函数将产生一个随机数。 VI 就将把这个数加入到存储在寄存器中的最近三个数值中。 Random Number (0– 1) 再将结果除以 4，就能得到这些数的平均值（当前数加上以前的三个数） 。然后再将这个平均值显示在波形图中。

7.Wait Until Next ms Multiple 函数 （FunctionsoTime & Dialog ）——它将确保循环的每个周期不会比毫秒输入快。 在本练习中，毫秒输入的值是 500 毫秒。如果用鼠标右键单击图标， 从快捷菜单中选择 VisibleoLabel ，就可以看到 Wait Until Next ms Multiple 的标签。

8. 用鼠标右键单击 Wait Until Next ms Multiple 功能函数的输入端子，在快捷菜单中选择 Create Constant 。出现一个数值常数，并自动与功能函数连接。

9. 将 Constant 设置为 500。这样连接到函数的数值常数设置了 500 毫秒的等待时间。因此循环每半秒执行一次。注意， VI 用一个随机数作为移位寄存器的初始值。如果没有设置移位寄存器端子的初始值， 它就含有一个默认的数值， 或者上次运行结束时的数值，因此开始得到的平均数没有任何意义。

10. 执行该 VI ，观察过程。

11. 把该 VI 保存。

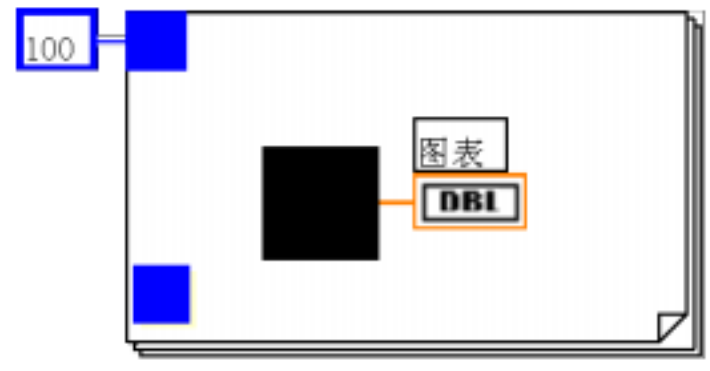
练习 四 结束。

附注：移位寄存器的初值：

上面的练习中对移位寄存器设置了初值 0.5 。如果不设这个初值，默认的初值是０。 在这个例子中， 一开始的计算结果是不对的， 只有到循环完 3 次后移位寄存器中的过去值才填满，即第 4 次循环执行后可以得到正确的结果。

2.1.2 For 循环

For 循环用于将某段程序执行指定次数。 和 While 循环一样， 它不会立刻出现在流程图中， 而是出现一个小的图标， 而后您可以修改它的大小和位置。 具体的方法是， 先单击所有端子的左上方， 然后按下鼠标， 拖曳出一个包含所有端子的矩形。释放鼠标时就创建了一个指定大小和位置的 For 循环。



For 循环将把它的框图中的程序执行指定的次数， For 循环具有下面这两个端子：

N: 计数端子（输入端子）——用于指定循环执行的次数。

I: 周期端子（输出端子）——含有循环已经执行的次数。

上图显示了一可以产生　100 个随机数并将数据显示在一个图表上的　　For 循环。在该例中，　i　的初值是 0，终值是 99。

练习五　　使用 For 循环

目的：用 For 循环和移位寄存器计算一组随机数的最大值。

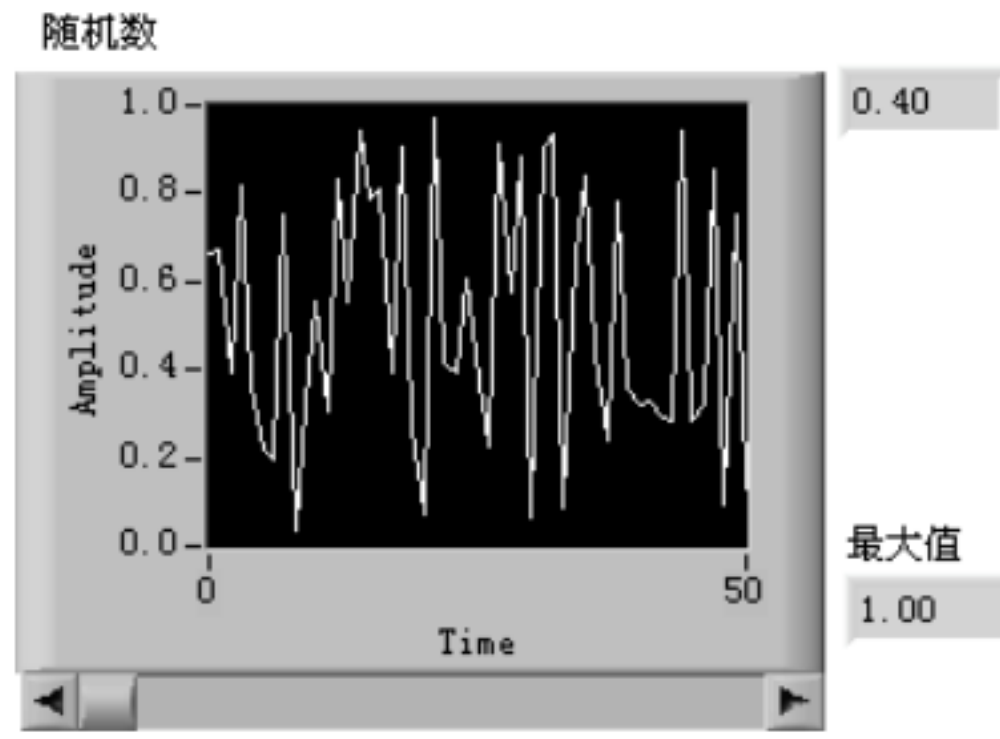
１．打开一个新的前面板，按照下图创建对象。

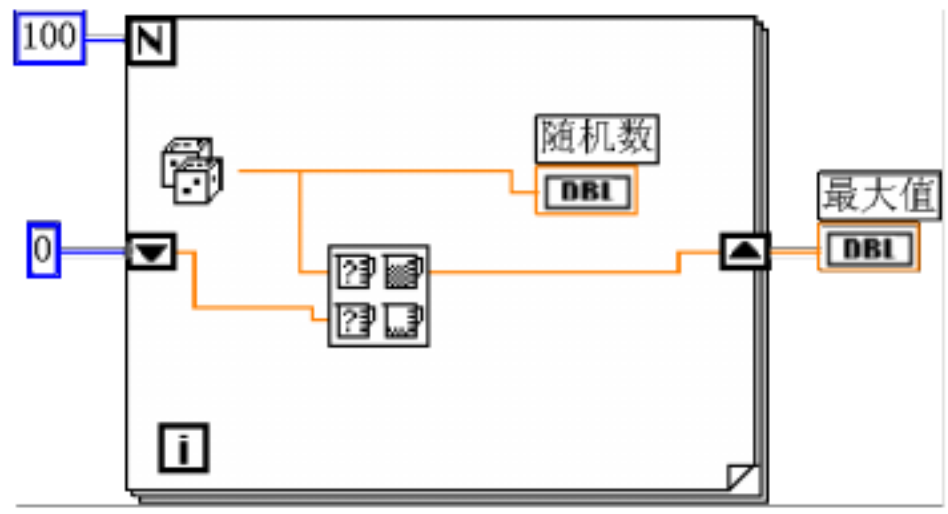
a. 将一个数字显示对象放在前面板，设置它的标签为“ 最大值 ”。

b. 将一个波形图表放在前面板，设置它的标签为“ 随机数 ”。将图表的纵坐标范围改为 0.0 到 1.0 。

c. 在图表的快捷菜单中选择 Visible ItemsoScrollbar 和 Digital Display ，并隐藏 Plot Legend 。

d. 用移位工具修改滚动栏的大小。





练习五　的前面板和流程图

２．按照上图画流程图。

３．在流程图中放置一个　For 循环（ FunctionsoStructures　）。

４．在 For 循环的边框处单击鼠标右键，　在快捷菜单中选择　AddShift　Register 。

５．将下列对象添加到流程图。

a.Random Number (0 – 1) 函数（ FunctionsoNumeric ）——产生 0 到 1 之间的某个随机数。

b. 数值常数（ FunctionsoNumeric ）——在这个练习中需要将移位寄存器的初始值设成 0。

c.Max&Min函数（FunctionsoComparison ）——输入两个数值，再将它们的最大值输出到右上角， 最小值输出到右下角。 这里只需要最大值， 只用连接最大值输出。

d. 数值常数（ FunctionsoNumeric ）——For 循环需要知道需要执行的次数。本练习中是 100 次。

6．按照上图连接各个端子。

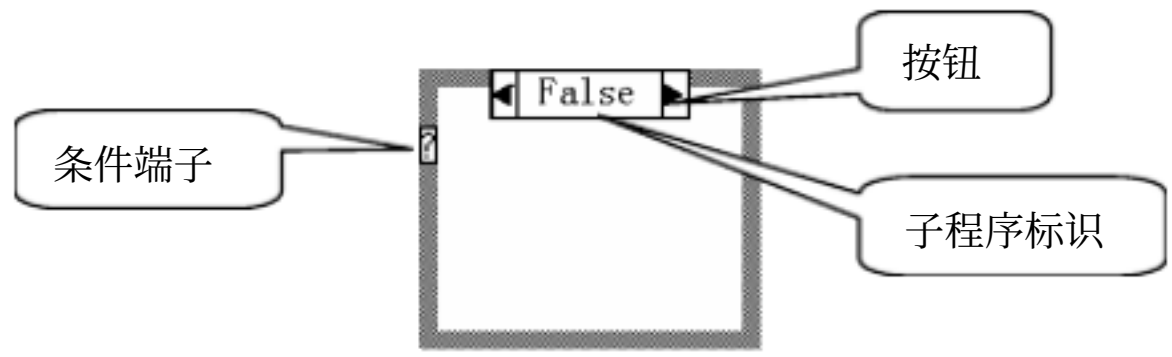
7．运行该 VI。

8．将该 VI 保存。

练习五 结束。

2.2 分支结构： Case

Case结构含有两个或者更多的子程序（ Case），执行那一个取决于与选择端子或者选择对象的外部接口相连接的某个整数、布尔数、字符串或者标识的值。必须选择一个默认的 Case以处理超出范围的数值，或者直接列出所有可能的输入数值。 Case 结构见下图，各个子程序占有各自的流程框，在其上沿中央有相应的子程序标识： Ture、False 或 1、 2、 3 ，。按钮用来改变当前显示的子程序（各子程序是重叠放在屏幕同一位置上的）。



练习六 使用 Case 结构

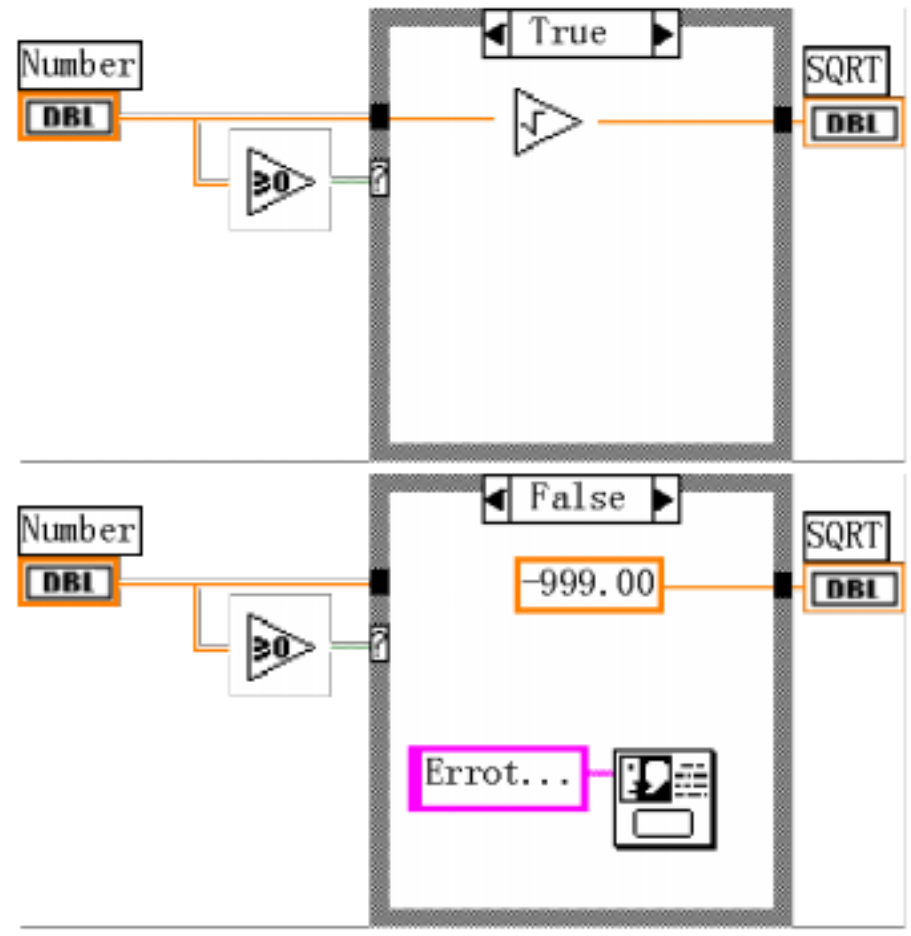
目的：创建一个 VI 以检查一个数值是否为正数。 如果它是正的， VI 就计算它的平方根，反之则显示出错。

前面板

1. 打开一个新的前面板，并按照下图所示创建对象。控制对象用于输入数值，显示对象用于显示该数值的平方根。

流程图

2. 照下图创建流程图。



练习六的面板和流程图

3. 从 FunctionsoStructures 中选择一个 Case 结构，并放置在在流程图中。Case 结构是一个可以改变大小的方框。 先来做 Ture 的情况，照流程图上半部分构造。

a. Greater Or Equal To 0? 函数（FunctionsoComparison ）——如果输入数值大于或者等于 0 就会返回一个 TRUE值。

b. Square Root 函数（FunctionsoNumeric ）——返回输入数值的平方根。连好线

c. 点击 Case框的选择按钮，转入 False 情况编程
数值常数（FunctionsoNumeric ）——这里用于显示错误的代数值 -999.00 。

d. One Button Dialog 函数（FunctionsoTime & Dialog ）——在这里它用于显示一个对话框，内容是 Error... 。

e. 字符串常数 (FunctionsoStrin g)——用 Edit Text Tools 在对话框中输入字符串。

该 VI 在 TRUE或者 FALSE情况下都会执行。如果输入的数值大于等于 0，VI 会执行 TRUECase，返回该数的平方根，否则将会输出－ 999.00，并显示一个对话框，内容为 Error... 。

返回前面板，运行该 VI。修改标签为 Number的数字式控制对象的数值，分别尝试一个正数和负数。注意，当把数字式控制对象的值改为负数时， LabVIEW 会显示 Case结构的 FALSE Case中设置的出错信息。

4. 保存该 VI 。

VI 的算法

本练习中的流程图功能相当于代码式编程语言中的下列伪代码：

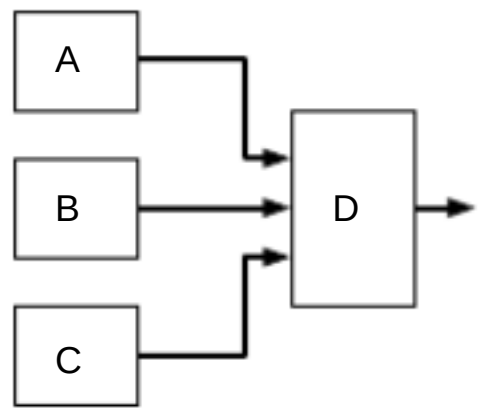
```
if (Number >= 0) then
    Square Root Value = SQRT(Number)
else
    Square Root Value = -999.00
    Display Message "Error.. "
end if
```

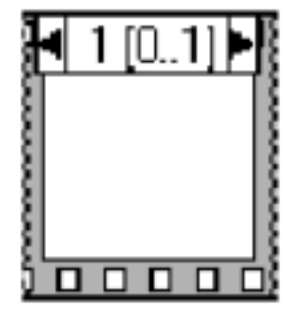
练习 六 结束。

2.3 顺序结构和公式节点

2.3.1 顺序结构（ Sequence Structure ）

在代码式的传统编程语言中，默认的情况是，程序语句按照排列顺序执行，但 LabVIEW中不同，它是一种图形化的数据流式编程语言。 在下图左图中， 假设有 A、B、C、D4 个节点，其数据流向如右图所示。按照数据流式语言的约定，





顺序结构的说明

任何一个节点只有在所有的输入数据有效时才会执行，所以图中，当且仅当 A、B、C3 个节点执行完，使得 D 节点的 3 个输入数据都到达 D节点后， D 节点才执行。但是你要注意，这里并没有规定 A、B、C3 个节点的执行顺序。在 LabVIEW中这种情况下， A、B、C的执行顺序是不确定的，如果你需要对它们规定一个确定的顺序，那就需要使用本节介绍的“ 顺序结构 ” 。

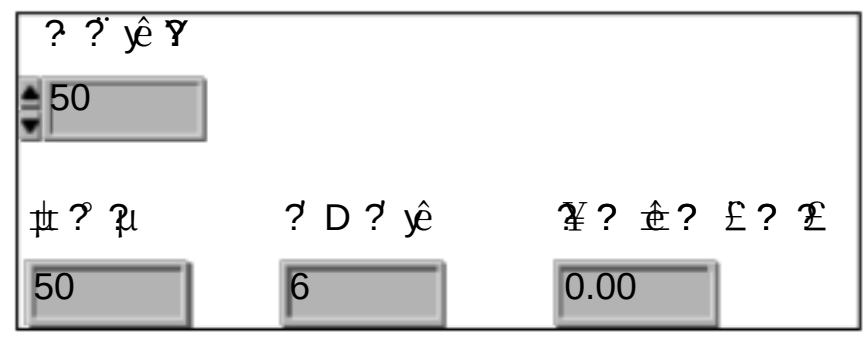
上图中的右边是顺序结构的图标，它看上去像是电影胶片。它可以按一定顺序执行多个子程序。 首先执行 0 帧中的程序， 然后执行 1 帧中的程序， 逐个执行下去。与 Case结构类似，这多帧程序在流程图中占有同一个位置。

练习 七 使用顺序结构

目的：创建一个 VI ，计算生成等于某个给定值的随机数所需要的时间。

前面板

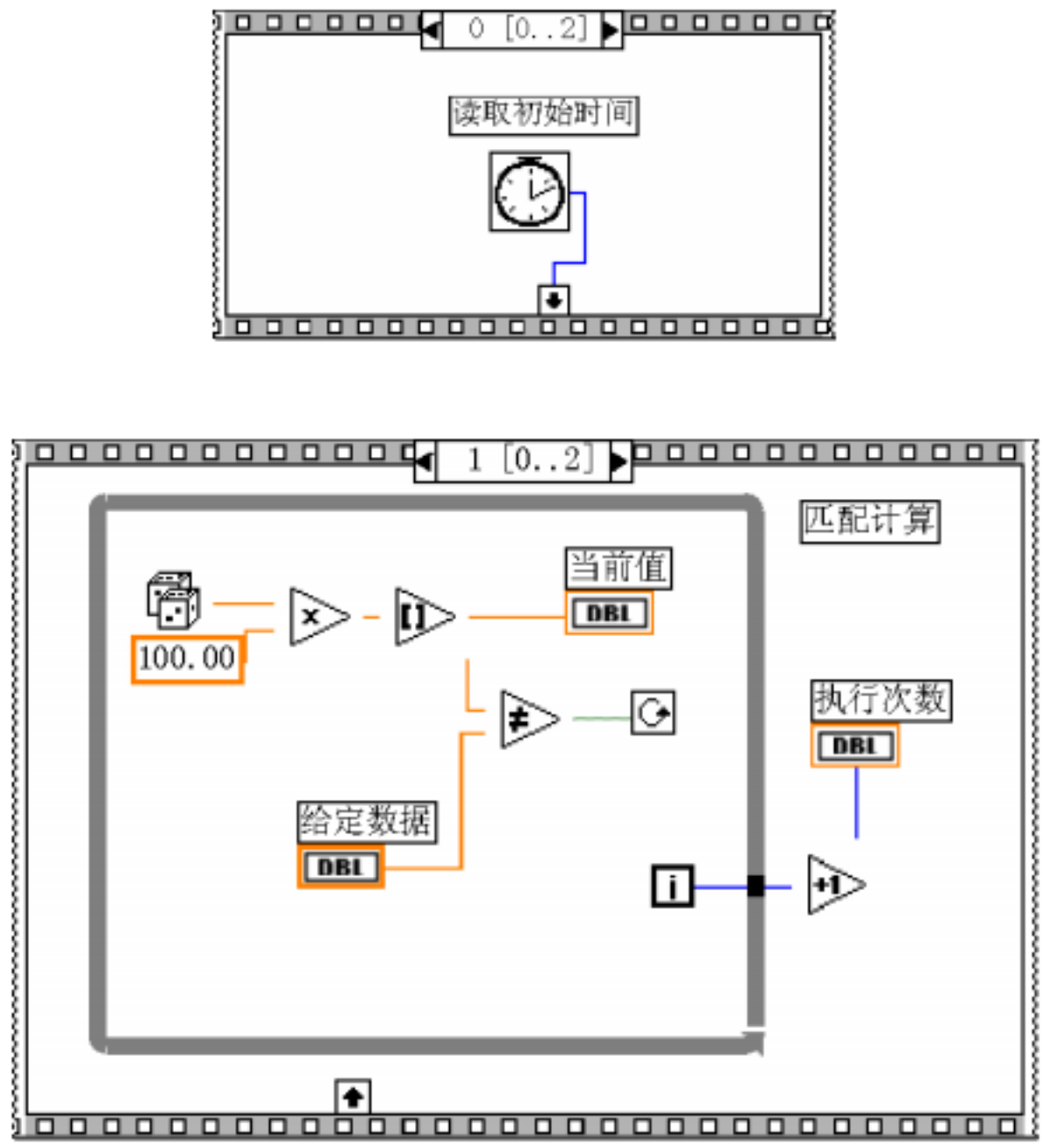
打开一个新的前面板，并按照下图所示创建对象。



练习七的前面板

我们约定数据是 0 到 100 范围的整数。当前值用于显示当前产生的随机数。“执行次数”用于显示达到指定值循环执行的次数。匹配时间用来显示达到指定值所用的时间。

流程图



练习 七 的流程图（共 3 帧）

1. 在流程图中放置顺序结构 (FunctionsoStructures)。
2. 用鼠标右键单击帧的边框，在快捷菜单中选择 Add Frame After ，创建一个

新帧。重复这个步骤，再创建一个帧。共 3 桢。

3. 选中第 0 桢，设置读取初始时间（子）程序

4. 第 0 帧的下边框上含有一个小方框，其中有一个箭头。这个方框叫做顺序局部变量，可以在同一个顺序结构中的各个帧之间传递数据。用鼠标右键单击第 0 帧的底部边框，选择 Add Sequence Local ，创建顺序局部变量。顺序局部变量显示为一个空的方块。 当您将某个功能函数与顺序局部变量相连时， 方块中的箭头就会自动显示。

5. Tick Count (ms) 函数（FunctionsoTime &Dialog ）——返回启动到现在的时间（以毫秒为单位）。在这里例子里需要使用两个这个函数。另一个在第 2 帧中。 

6. 按图连好线。转入第 1 帧。该帧是匹配计算，内含一个循环结构。该图中使用的新函数有：

 Round to Nearest 函数（ FunctionsoNumeric ）——在该例中，它用于取 0 到 100 之间的随机数到距离最近的整数。

 Not Equal?函数（ Functionso Comparison）——在该例中，它将随机数和前面板中设置的数相比较，如果两者不相等会返回 TRUE值，否则返回 FALSE。

 Increment 函数（FunctionsoNumeric ）——在该例中，它将 While 循环的计数器加 1。

7. 按图连好线。转入第 2 帧

在第 0 帧中， Tick Count （ms）功能函数将以毫秒为单位表示当前时间。这个数值被连到顺序局部变量， 这样它就可以被后续的帧使用。 在第 1 帧中，只要函数返回的值与指定值不等， VI 就会持续执行 While 循环。在第 2 帧中， Tick Count （ms）功能函数以毫秒为单位返回新的时间。 VI 从中减去原来的时间（由第 0 帧通过顺序局部变量提供）就可以计算出花费的时间。

8. 返回前面板，在 Number to Match 控制对象中输入一个数值，执行该 VI。

9. 把该 VI 保存。

练习 七 结束。

附注与说明： 设置数据范围

在设定一个数据对象时， 可以设置对输入数据的限制， 利用快捷键选择 Data Range， 选项，将会出现如下对话框 ：

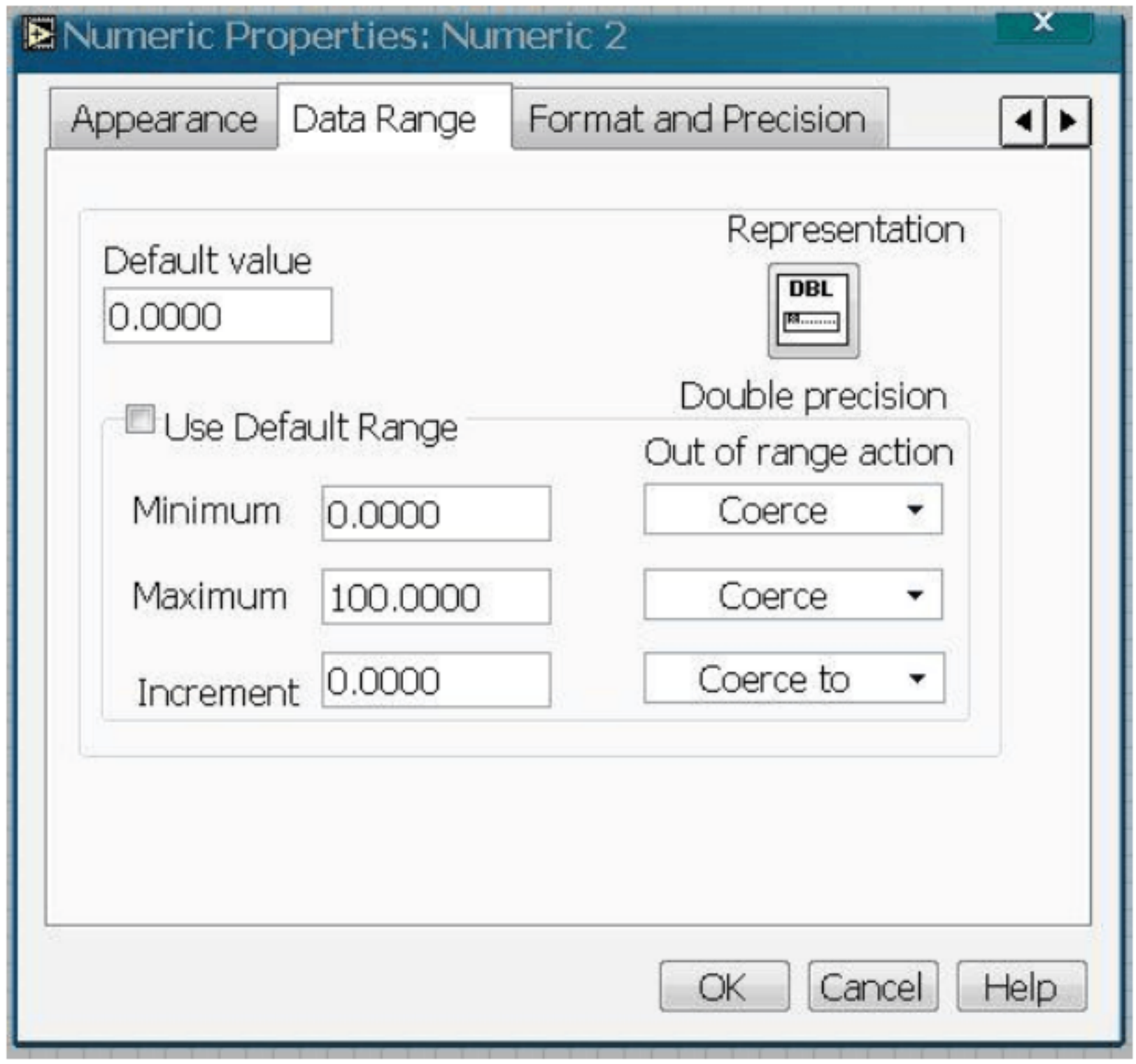


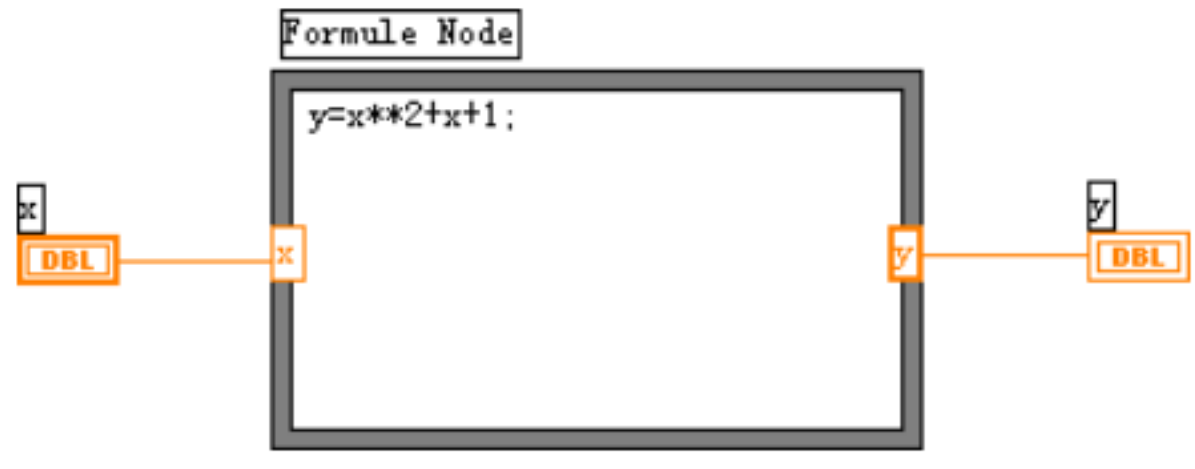
图　设置数据范围

它可以防止用户创建的控制对象或显示对象的值超出某个预设的范围。　　您可以选择忽略这个值，　将它强制修改到范围以内，　或暂停程序的执行。　在程序执行时，如果发生溢出错误，　溢出错误符号将显示在工具栏中的执行按钮的位置。　　而且，一个立体的黑框将把发生溢出的控制对象包围起来。

2.3.2　公式节点（ Formula Node）

公式节点是一个大小可变的方框，　可以利用它直接在流程图中输入公式。　从 FunctionsoStructures　　中选择公式节点就可以把它放到流程图中。当某个等式有很多变量或者非常复杂时，这个功能就非常有用。例如等式：　　$y = x^2 + x + 1$

使用公式节点可以表示为：



公式节点示意图

利用公式节点可以直接输入一个或者多个复杂的公式，　而不用创建流程图的很多子程序。使用文本编辑工具来输入公式。　创建公式节点的输入和输出端子的方法是，用鼠标右键单击第　0 帧的底部边框，选择　Add Input (Add Output)　。再在节点框中输入变量名称。　变量名对大小写敏感。然后就可以在框中输入公式。每个公式语句都必须以分号（；）结尾。

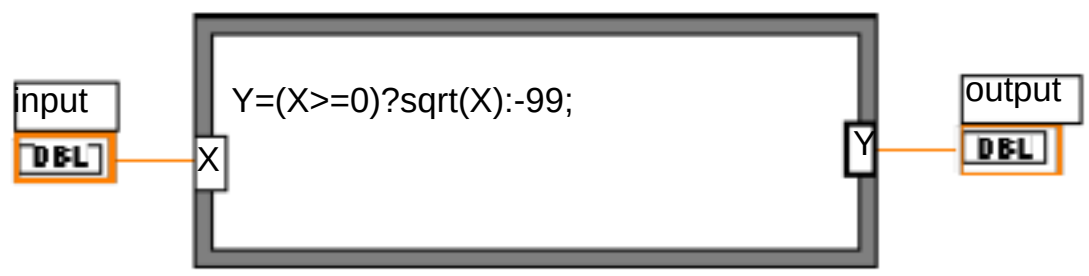
公式节点的帮助窗口中列出了可供公式节点使用的操作符、函数和语法规定。一般说来，　它与 C 语言非常相似，　大体上一个用　C 写的独立的程序块都可能用到公式节点中。但是仍然建议不要在一个公式节点中写过于复杂的代码程序。

下面这个例子显示了如何在一个公式节点中执行不同条件时的数据发送。

请阅读下面这段程序代码，　如果 X 为正数，它将算出　X 的平方根并把该值赋给 Y，如果 X 为负数，程序就给　Y 赋值-99。

```
if (x >= 0) then
    y = sqrt(x)
else
    y = -99
end if
```

可以用公式节点取代上面这段代码，如下图所示：



注意：公式节点中变量字母　X,Y 大、小写是有区别的，开方的函数　sqrt(X)　中函数名称是小写。

练习　八　　使用公式节点

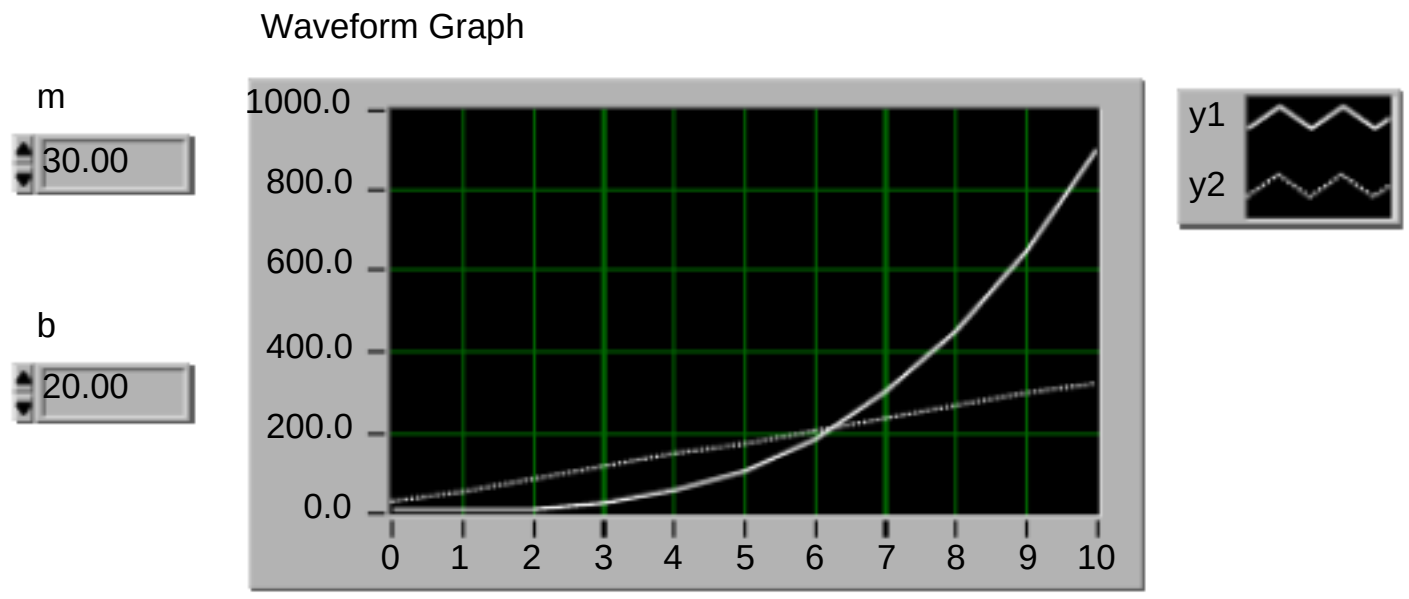
目的：创建一个 VI，它用公式节点计算下列等式：

$y1 = x^3 - x^2 + 5$

$y2 = m* x + b$

x 的范围是从　0 到 10。可以对这两个公式使用同一个公式节点，　并在同一个图表中显示结果。

前面板

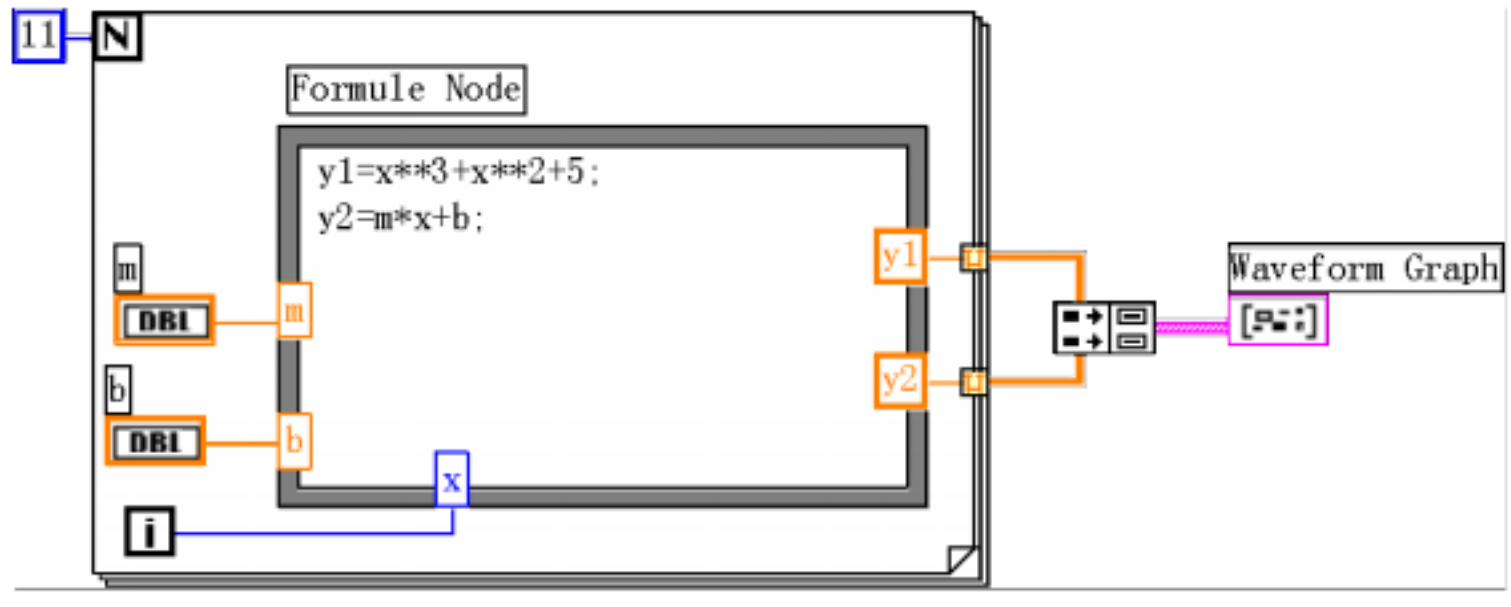


练习 八 的面板

打开一个新的前面板， 按照上图（该图中包含运行结果） 创建前面板中的对象。波形图显示对象用于显示等式的图形。该 VI 使用两个数字式控制对象来输入 m和 b 的值。

流程图

按照下图创建流程图。



练习 八 的流程图

在创建某个输入或者输出端子时， 必须给它指定一个变量名。 这个变量名必须与公式节点中使用的变量名完全相符。

公式节点中，在边框上单击鼠标右键，在快捷菜单中选择 Add Input ，可以创建三个输入端子。在快捷菜单中选择 Add Output ，创建输出端子。

x 的范围是从 0 到 10（包括 10），就必须连接 11 到计数端子。

Build Array （FunctionsoArra y）——在这个例子中，它用于将两个数据构成数组形式提供给一个多曲线的图形中。 通过用变形工具拖拉边角就可以创建两个输入端子。 

返回前面板，尝试给 m和 b 赋以不同的值再执行该 VI 。

把该 VI 保存。

练习 八 结束。

第三讲 数据类型：数组、簇和波形

3.1 概述

数组是同类型元素的集合。 一个数组可以是一维或者多维， 如果必要， 每维最多可有 $2^{31}-1$ 个元素。可以通过数组索引访问其中的每个元素。 索引的范围是 0 到 n － 1，其中 n 是数组中元素的个数。图 3 － 1 所显示的是由数值构成的一维数组。注意第一个元素的索引号为 0，第二个是 1，依此类推。数组的元素可以是数据、字符串等，但所有元素的数据类型必须一致。

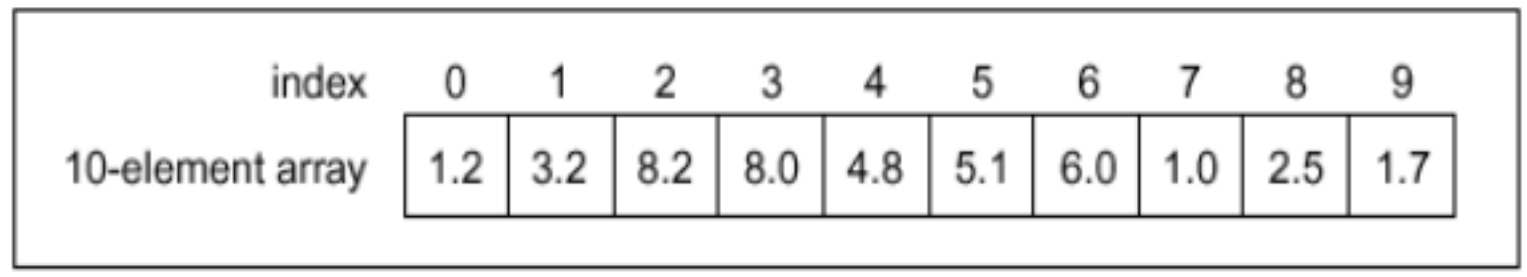


图 3－1　数组示意图

簇（Cluster ）是另一种数据类型，它的元素可以是不同类型的数据。它类似于 C 语言中的 stuct 。使用簇可以把分布在流程图中各个位置的数据元素组合起来，这样可以减少连线的拥挤程度。减少子　　VI 的连接端子的数量。

波形（Waveform）可以理解为一种簇的变形，它不能算是一种有普遍意义的数据类型，但非常实用。

3.2　数组的创建及自动索引

3.2.1　创建数组

一般说来，创建一个数组有两件事要做，　首先要建一个数组的“ 壳 ”（shell ），然后在这个壳中置入数组元素（数或字符串等）　。

如果需要用一个数组作为程序的数据源，　可以选择 FunctionsoArrayoArray Constant ，将它放置在流程图中。 然后再在数组框中放置数值常量、　布尔数还是字符串常量。 下图显示了在数组框放入字符串常量数组的例子。　左边是一个数组壳，中间的图上已经置入了字符串元素，右边的图反映了数组的第 0 个元素为：”ABC”，后两个元素均为空。

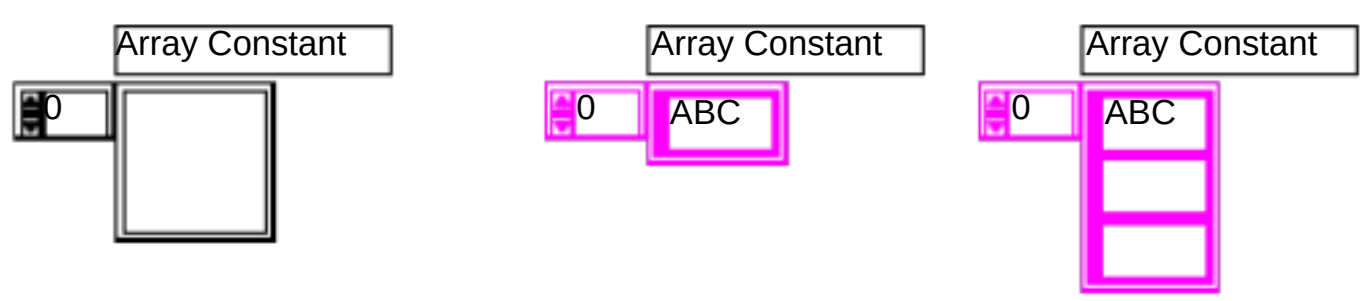


图 3－1　数组的创建

在前面板中创建数组的方法是，从　Controls　模板中选择 Array & Cluster　，把数组放置在前面板中，然后选择一个对象（例如数值常量）插入到数组框中。这样就创建了一个数值数组。也可以直接在前面板中创建数组和相应的控制对象，然后将它们复制或者拖曳到流程图中，　创建对应的常数。 还有很多在流程图中创建和初始化数组的方法，有些功能函数也可以生成数组。

3.2.2　数组控制对象、常数对象和显示对象

通过把数组与数值、布尔数、字符串或者簇组合在一起，可以在前面板和流程图中创建任何一种控制对象、　常数对象和显示对象。 数组元素不能是数组、 图表或者图形。

3.2.3　自动索引

For 循环和　While 循环可以自动地在数组的上下限范围内编索引和进行累计。这些功能称为自动索引。 在启动自动索引功能以后，　当把某个外部节点的任何一维元素连接到循环边框的某个输入通道时，　该数组的各个元素就将按顺序一个一个地输入到循环中。 循环会对一维数组中的标量元素，　或者二维数组中的一

维数组等编制索引。 在输出通道也要执行同样的工作——数组元素按顺序进入一维数组，一维数组进入二维数组，依此类推。

在默认情况下， 对于每个连接到 For 循环的数组都会执行自动索引功能。 可以禁止这个功能的执行，方法是用鼠标右键单击通道（输入数组进入循环的位置），在快捷菜单中选择 Disable Indexing 。

练习 九 创建一个自动索引的数组

目的：使用 For 循环的自动索引功能创建数组，并用一个图形（ Graph）显示该数组。

前面板

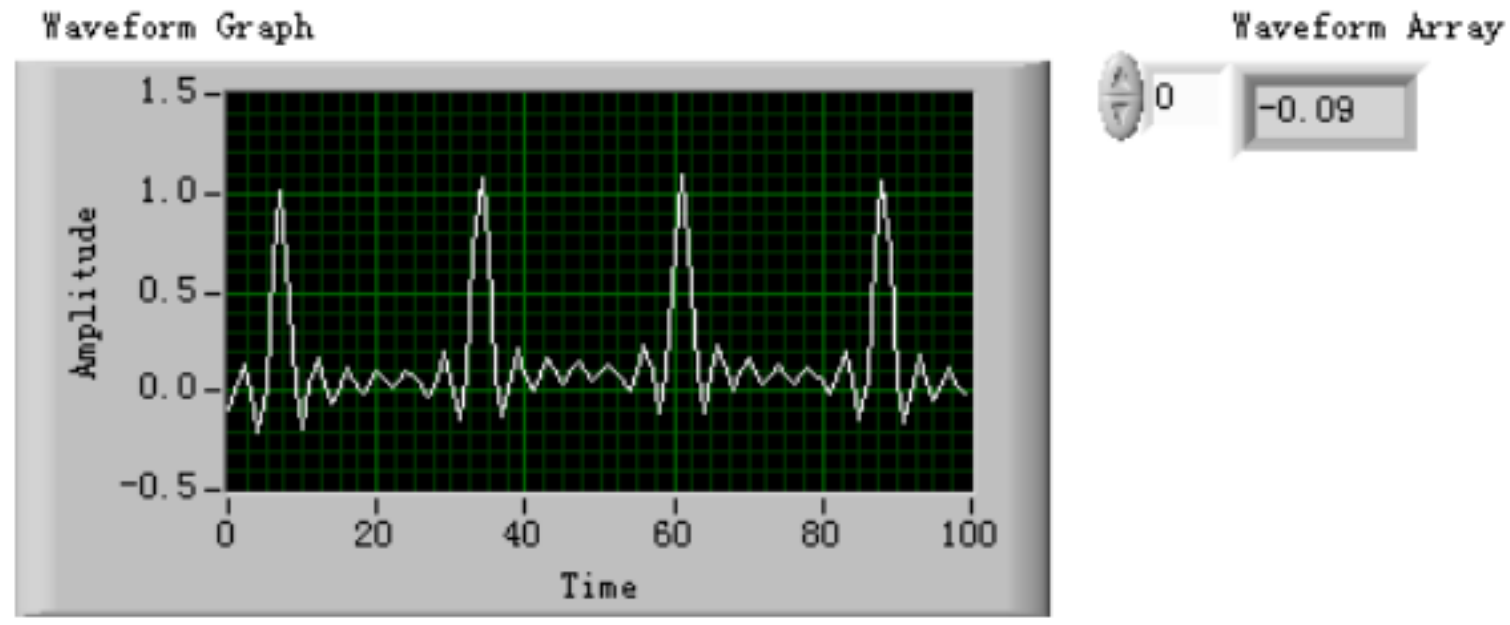
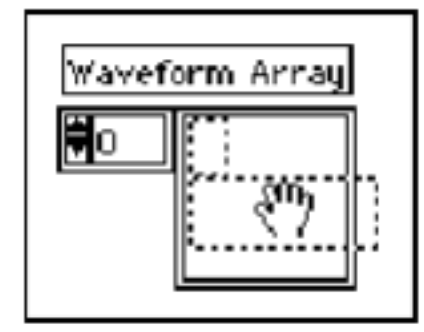


图 3－2 练习 九 的面板

1. 打开一个新的前面板。
2. 选择 ControlsoArray & Cluster ，在前面板中放置一个数组。设置它的标签为 Waveform Array 。

3. 选择 ControlsoNumeric ，在数组框中插入一个数字式显示对象。如右图所示。它用于显示数组的内容。



4. 选择 Controlso Graph，在前面板中放置一个波形图。设置它的标签为 Waveform Graph。
5. 隐藏图例和模板。
6. 用鼠标右键单击图形，并在快捷菜单中取消选中 Y ScaleoAutoscale Y ，禁止自动坐标功能。
7. 使用文本工具，把 Y 轴的范围改为 -0.5 到 1.5 。
8. 按照下图创建流程图。

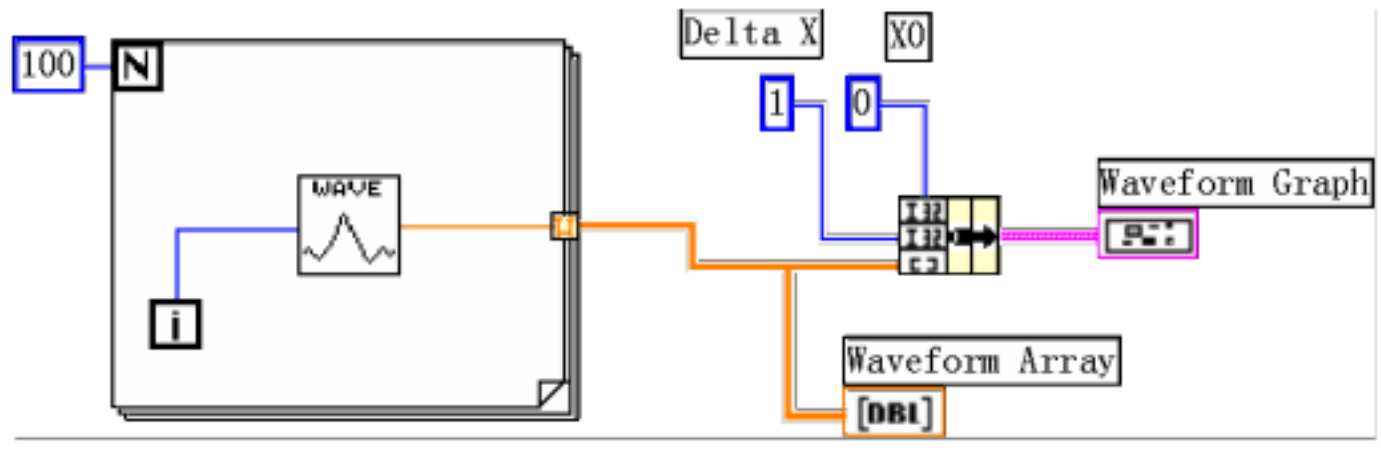


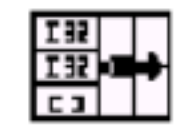
图 3－2　练习 九 的流程图

由 FunctionsoSelect a VI,　寻找 LabVIEW\activity　目录下的 Generate Waveform VI　，它的作用是返回波形中的某一点。这个 VI 需要输入一个索引，我们将循环周期连接到这个输入。

注意 Generate Waveform VI 连出来的连线在循环边界变成一个数组时会变粗，正是在这个边界处形成了一维数组。

For 循环会自动累计边界内的数组。 这种功能叫做自动索引。 在这个例子中，连接到循环计数输入的数值常数令　For 循环创建了一个由　100 个元素组成的数组。

Bundle 函数（ Functionso Cluster　）——将图块中的各个组件组合成一个簇，在正确连接以前需要改变该函数的图标的大小。将移位工具放在图标的左下角。变形工具会变成如左图所示，拖曳鼠标直到出现第三个输入端子。



数值常数（ FunctionsoNumeric ）——三个数值常数用于设置　For 循环执行的周期数 N=100，初始 X=0和 delta X=1 。

9. 从前面板执行该　VI。该 VI 将把自动索引后的波形图数组显示在波形图中。

10. 把 X 的 delta　值改为 0.5，X的初始值改为　20。再次执行该　VI。注意，波形图现在同样显示　100 个点，而每个点的初始值为　20，X 的 delta　值为 0.5（见 X 轴）。

11. 只需在显示器中输入元素的索引号就可以查看波形数组中的任何元素。如果输入的数比数组的元素个数大，　那么显示器将变暗，　表示您没有为该元素设置索引。

如果需要一次查看多个元素，　可以通过改变数组显示对象的大小来实现。　把定位工具放置在数组框的右下角。工具将变成右图所示的变形工具。当工具变形时，　用鼠标拖曳数组的右边或者下边。　数组现在就可以按照元素索引的上升顺序显示多个素，以某个与指定索引对应的元素开始，如下图所示。

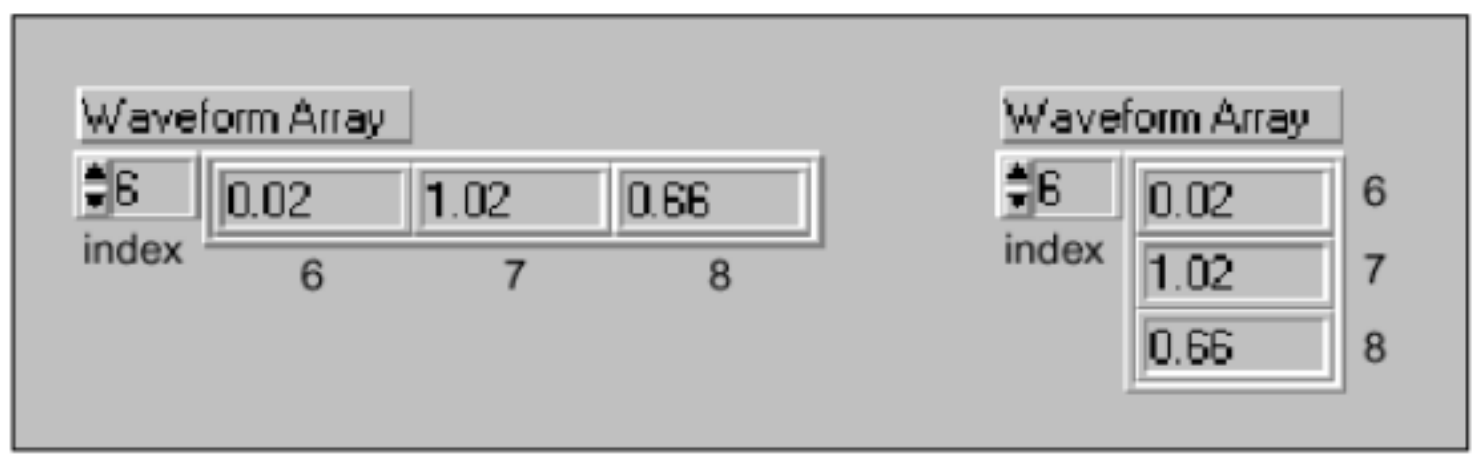


图 3－3　练习 九 中多个数组元素的同时观察

在前面的流程图中，您为波形图指定了初始的　X 值和 delta　X 值。默认的 X 初始值是 0，delta　X 值是 1。这样，也可以把波形数组直接连接到波形图端子，而无需指定初始的　X 值和 delta X　值，如图 3－4 所示。

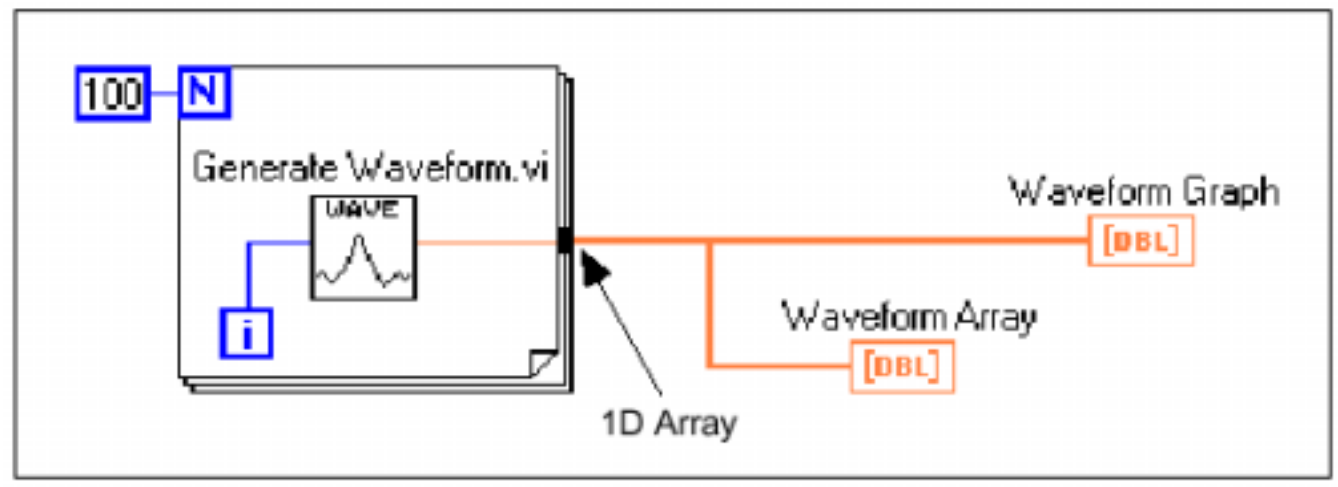


图 3－4　练习 九使用默认 X 及 Delta X 时简化后的流程图

12. 按上图删除 Bundle 功能函数和它所连接的常数对象。方法是用移位工具选择该功能函数和连接的常数对象，按下 <Delete>。再选择 EdtoRemove Bad Wires。按照上图完成流程图的连线。
13. 执行该 VI。注意初始的 X 值是 0， delta X 值是 1。

多图区图形

可以创建含有多条曲线的图形， 方法是创建一个数组， 用它来汇集传给单图区图形的类型的数据元素。

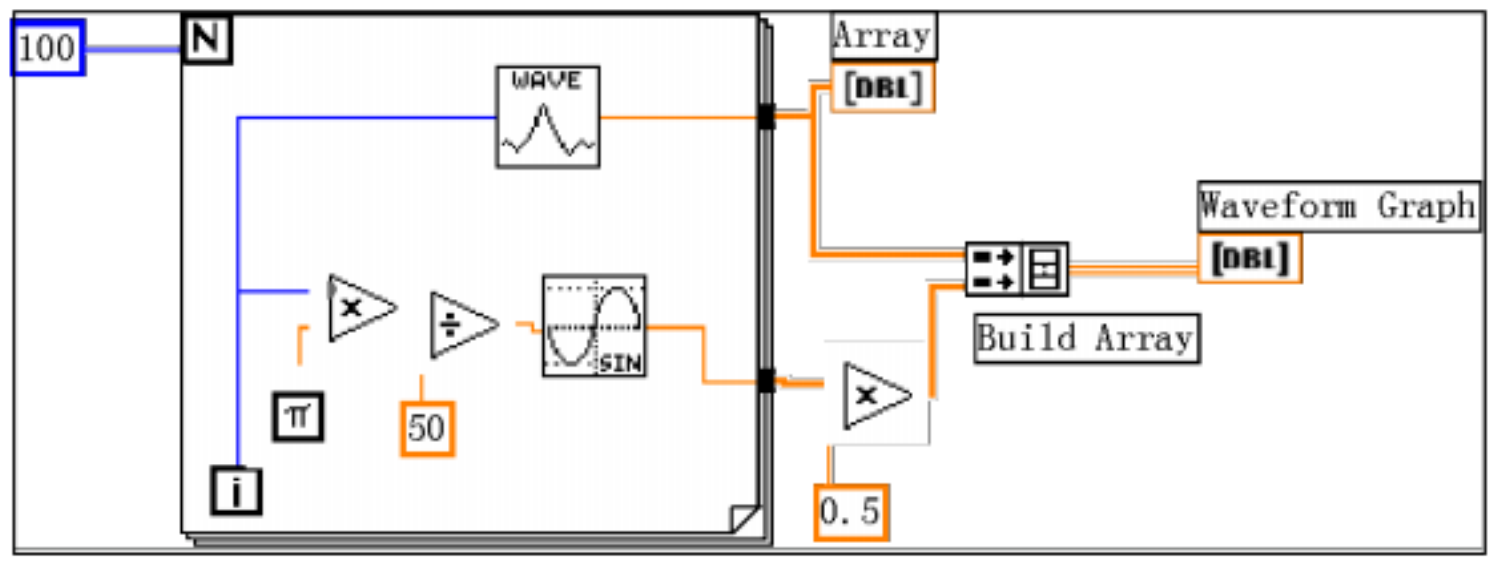


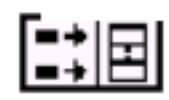
图 3－5　练习 九 多图区图形的流程图

14. 按照上图创建流程图。

正弦函数（ FunctionsoNumericoTrigonometric ）——在这里，它用于在 For 循环中创建一个由数据点组成的数组，表示一个正弦波周期。

Pi 常数（FunctionsoNumerico Additional Numeric Constants ）

Build Array （FunctionsoArra y）——在这里，它用于创建合适的数据结构（一个二维数组），在波形图中绘制两条曲线。。用移位工具拖曳边角可以增大该函数的面积，创建两个输入端子。



15. 返回前面板，执行该 VI 。注意同一个波形中的两个图区。默认情况下，它们的 X 初始值都是 0，delta X 初始值都是 1。下图是该程序的运行结果（前面板未改动）。
16. 把该 VI 保存为 LabVIEW\Activity 目录中的 Graph Waveform Arrays.vi 。
17. 可以修改图形中的某个图区的外观。 方法是，用鼠标右键单击这个图形， 再从弹出菜单选择对应的图例。

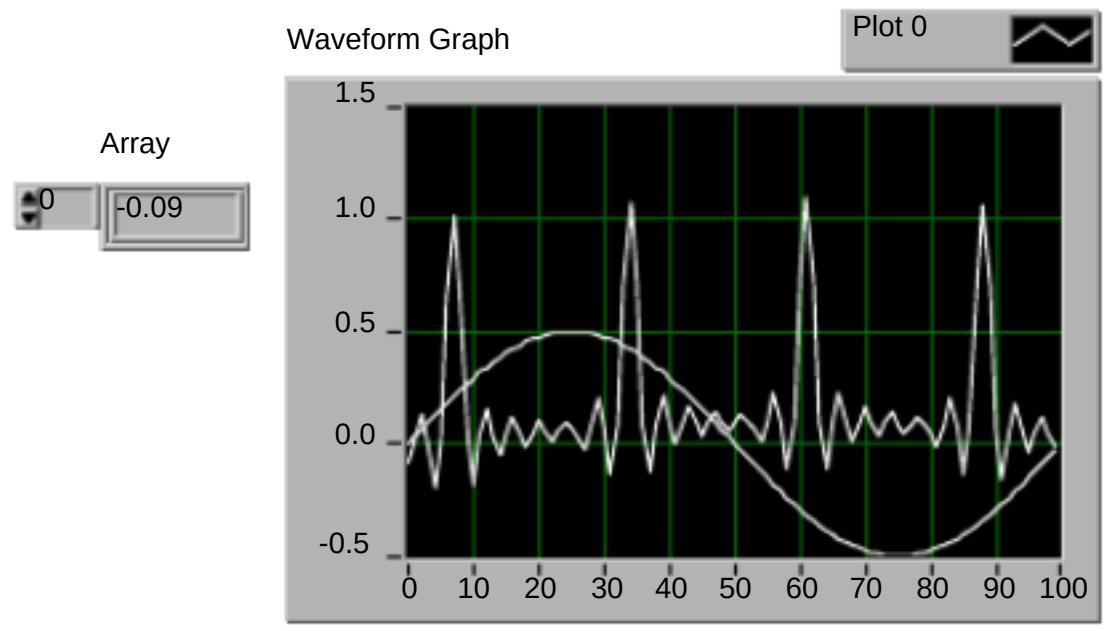


图 3－6　练习 九 多图区图形的面板显示

练习　九　结束。

在上面这个例子中，由于计算端子连接了一个值为　100 的常数对象，所以 For 循环将执行 100 次。下面这个例子显示了另外一种控制循环执行次数的方法。

练习 十　对输入数组使用自动索引功能

目的：打开并执行一个　VI，它将在一个　For 循环中使用自动索引功能处理一个数组。

1. 选择 FileoOpen , , 打开 Examples\General\arrays.llb　中的 Separate Array Values VI　。

2. 打开流程图。下面的示意图显示的是在　TRUE 和 FALSE 时的情况。

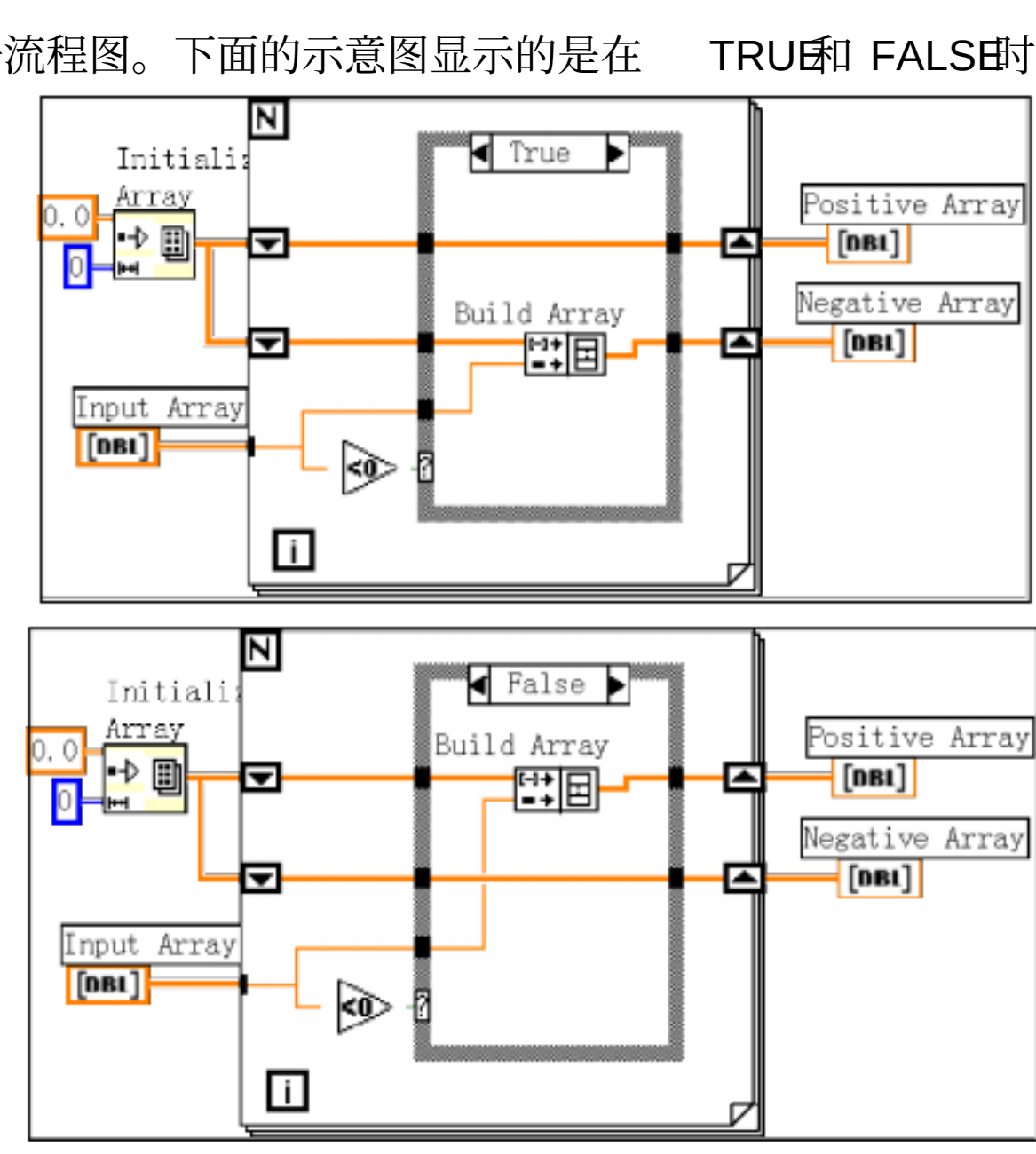


图 3－7　练习 十 的流程图

注意，Input　Array 引出的连线与 For 循环外的粗线不同，表示这是一个数

组，而循环内部的细线则表示这是一个数组元素。　　数组元素在每个循环期间将自动编号。

用自动索引功能设置　For　循环的计数器

注意，计数器端子还没有连线。　当您对某个进入　For 循环的数组使用自动索引功能时，　循环就将根据数组的大小执行相应的次数，　　这样就无需连接某个值到计数器的端口。如果对一个以上的数组使用自动索引功能，或者在使用自动索引功能之外还需要设置计数器时，　　实际的循环次数将是其中最小的数。

3. 执行该 VI。在输入的八个数中，可以看到　4 个属于正数数组，另外　4 个属于负数数组。

4. 从流程图中将一个值为　5 的常数对象连接到　For　循环的计数器端子。　执行该 VI。可以看到尽管输入数组仍然有八个　元素，但是　3 个位于正数数组，另外　2 个位于负数数组。　这说明，　如果设置了　N 并开启了自动索引功能，　　那么实际循环的次数将取较小的数。

5. 关闭该 VI，不要保存任何修改。

注：练习　十　的算法说明

下面是一段伪代码，解释上面的算法，假定输入数组为　　A（已赋值），B（正数）、C（负数）。Sbr　、Scr 分别是与 B 数组、C 数组对应的右寄存器数组，　Sbl　、Scl 分别是与 B 数组、C 数组对应的左寄存器数组，　size　运算为测数组实际大小，ins　运算为将一个数插入数组中最左边的空位。

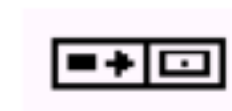
B=0	初始化
C=0	
K=size(A(.))	测 A 数组大小
For i=0 to k-1	
p=A(i)	取第 I 个元素值
if　p>=0 then	
Ins p,Sbr	将 p 值插入右寄存器
Else	
Ins p,Scr	
end if	
Sbl=Sbr	右寄存器值送给左寄存器
Scl=Scr	
Next i	
B=Sbr	右寄存器值送到正数组
C=Scr	
Print B	显示
Print C	
End	

练习　十　结束。

3．3　数组功能函数

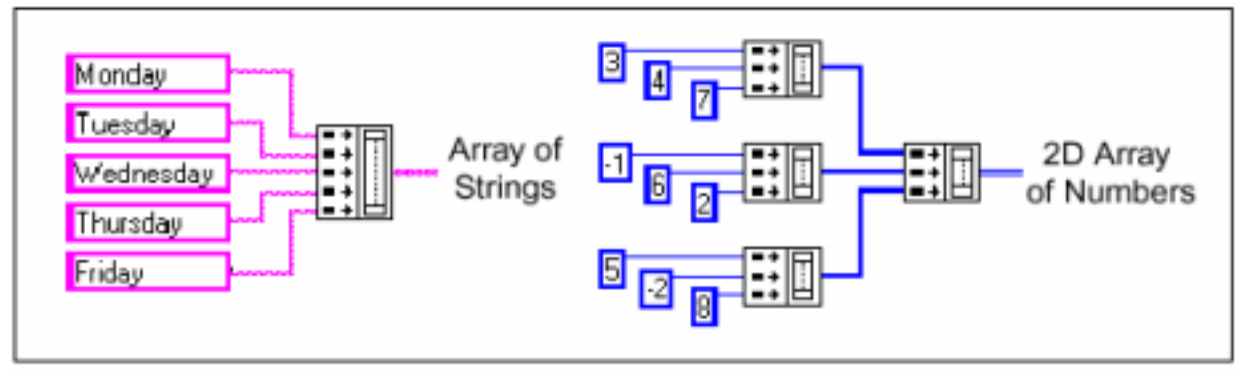
LabVIEW提供了很多用于操作数组的功能函数，位于 Functionso Array 中。其中包括 Replace Array Element,、Search 1DArray、Sort 1DArray、 Reverse 1D Array 和 Multiply Array Elements 等等。

创建数组——Build Array 函数（FunctionsoArray ）用于根据标量值或者其他的数组创建一个数组。



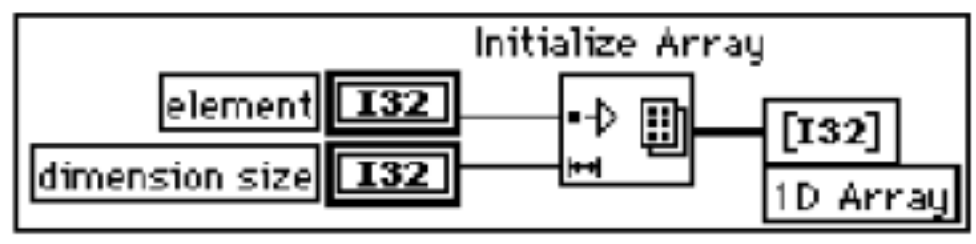
开始时， Build Array 函数具有一个标量输入端子。您可以根据需要向该功能函数中加入任意数量的输入， 输入可以是标量或者数组。 如果要添加其他的输入，用鼠标单击函数的左侧，在弹出菜单中选择 Add Element Input 或者 Add Array Input 。还可以用变形工具来增大节点的面积（把移位工具放置在某个对象的边角就会变成变形光标）。也可以使用变形光标或者选择 RemovdnputRemovdnput 来删除输入。

下图显示了利用流程图中的常数对象的值创建和初始化数组的两种方法。 左侧的方法是， 将 5 个字符串常数放入一个一维字符串数组中。 右侧的方法是， 将三组数值常数放入三个一维数值数组， 再将这三个数组组成一个二维数组。 这样最后产生的是一个 3x3 的数组，三列分别是 3, 4, 7; －1, 6, 2; 5, －2, 8. 。



还可以通过结合其他的含有标量元素的数组来创建数组。例如，假设有两个数组，三个标量元素，可把它们组成一个新的数组，顺序是：数组 1，标量 1，标量 2，数组 2，标量 3。

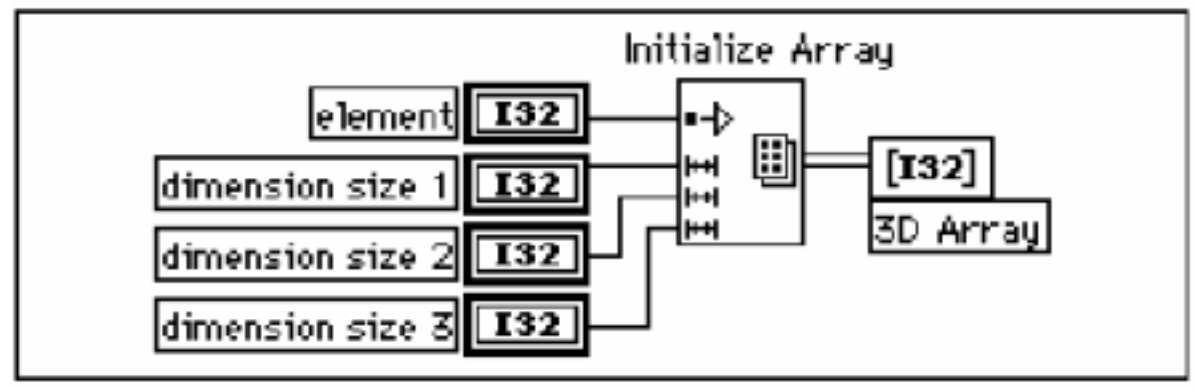
初始化数组（ Initialize Array ）——用于创建所有元素值都相等的数组。下图中，该功能函数创建了一个一维数组。



元素输入端子决定每个元素的数据类型和数值， 维长度输入端子决定数组的长度，例如，假设元素类型是长整型，值为 5，维长度为 100，那么创建的数组是一个一维的、 由 100 个值为 5 的长整型元素组成的数组。 也可以从前面板控制端子、流程图常数或者程序其他部分的计算结果得到输入。

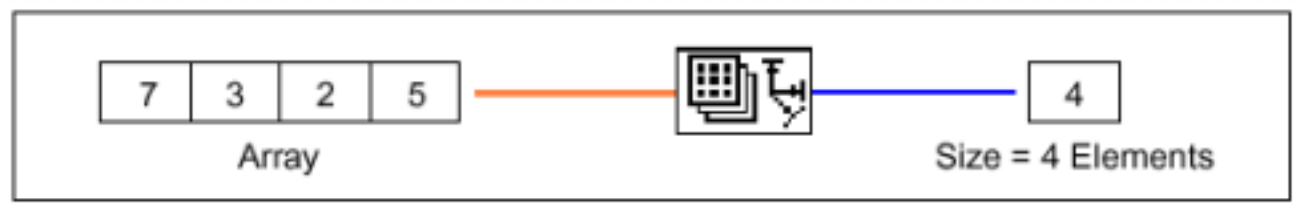
创建和初始化一个多维数组的方法是，用鼠标右键单击函数的右下侧，在弹出菜单中选择 Add Dimension。还可以使用变形光标来增大初始化数组节点的面积，为每个增加的维添加一个维长度输入端子。 也可以通过缩小节点的方法来删除维，即从函数的弹出菜单中选择 Remove Dimension，或者使用变形光标。下

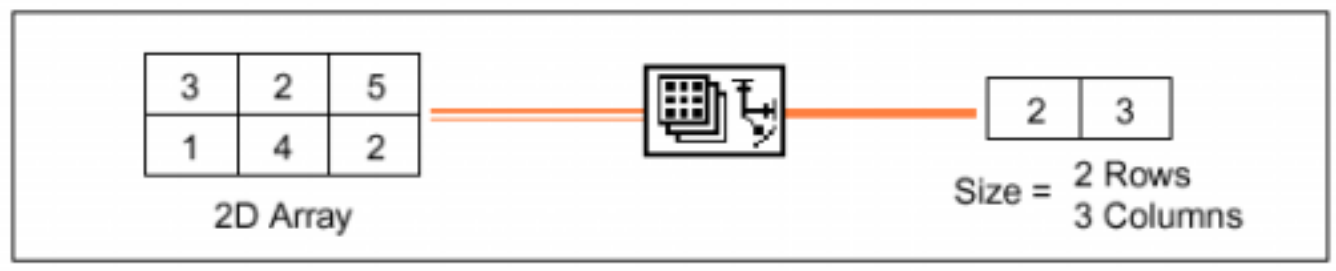
面的示意图显示了怎样初始化一个三维数组。



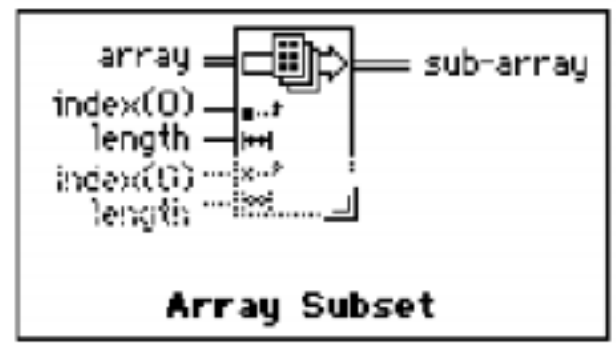
如果所有的维长度输入都是　0，该函数会创建一个具有指定数据类型和维数的空数组。

数组大小——Array Size　函数，返回输入数组中的元素个数。

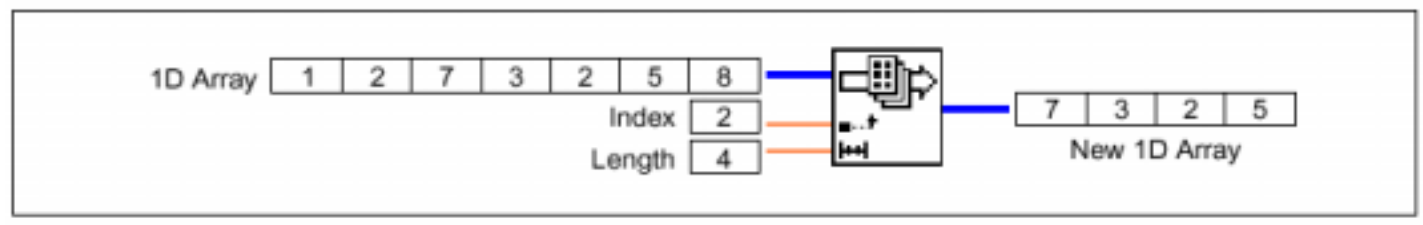


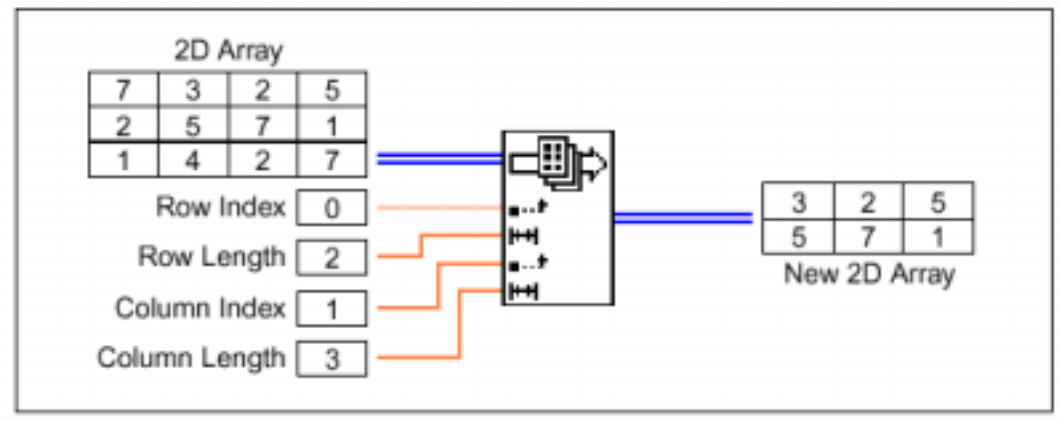


数组子集（ Array Subset ）——选取数组或者矩阵的某个部分。

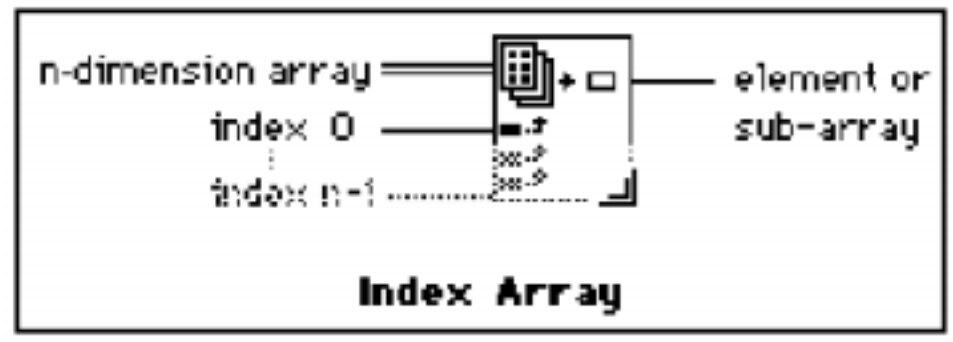


该函数可以返回从某个指针开始的部分数组，并包括了长度元素。下图显示了一些数组子集的例子，注意，数组索引从　0 开始。

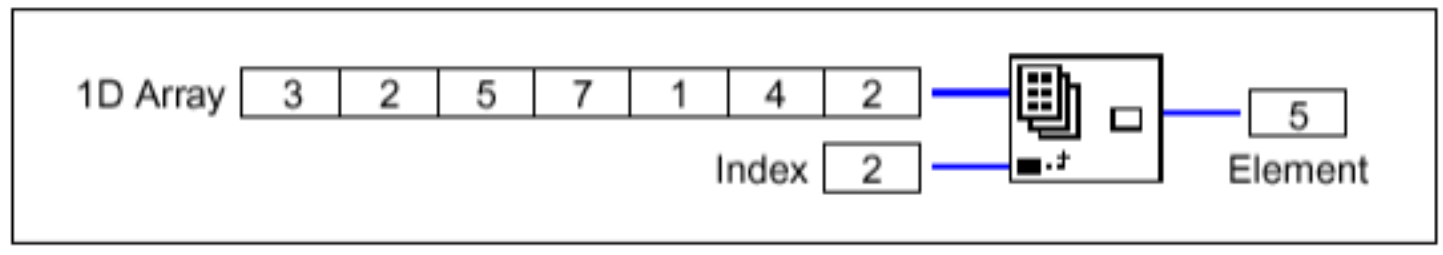




索引数组（ Index Array ）——用于访问数组中的某个元素。

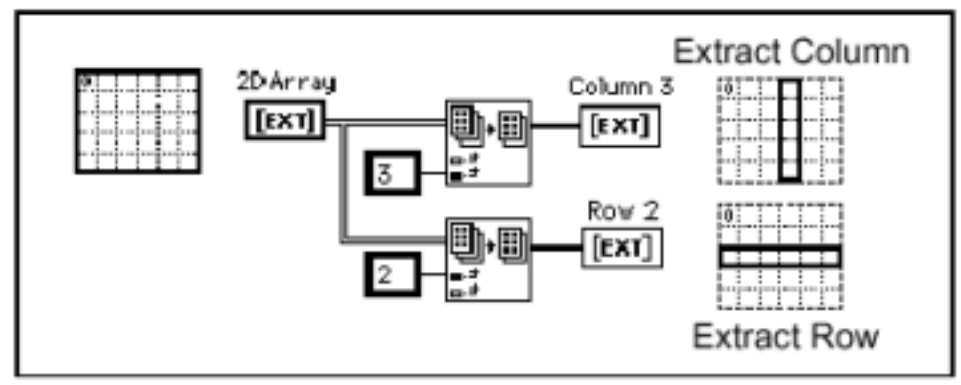


下图显示了一个索引函数的例子，它用于访问数组中的第三个元素。注意，因为第一个元素的索引为 0，所以第三个元素的索引是 2。

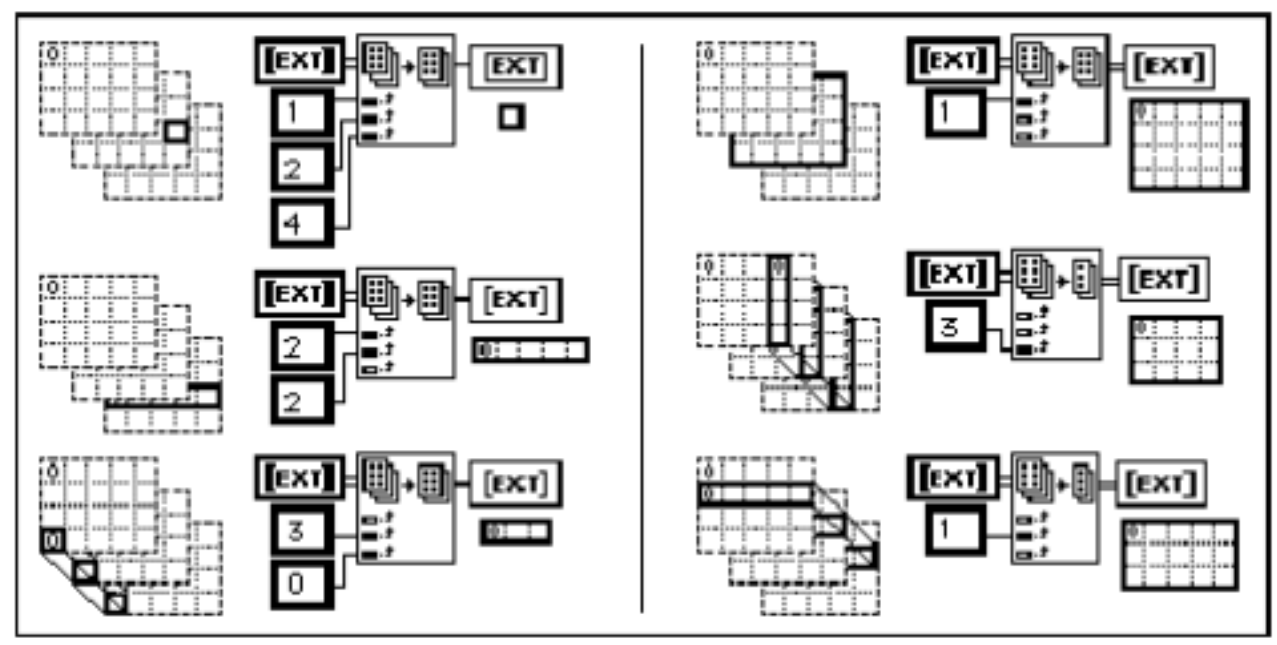


将一个二维数组与 Index Array 函数相连， Index Array 就会含 2 个索引端子。将一个三维数组与 Index Array 函数相连， Index Array 就会含 3 个索引端子。余类推。可以使用的索引端的符号是一个黑方快，被禁止使用的索引端（Disable Indexing ）是一个空心的小方框。当给一个被禁止使用的索引端连接上一个 Constant 或 Control 是它会自动变为黑方快， 即变为可以索引， 相反原来一个可以使用的索引端上连接的 Constant 或 Control 被删去时，索引端符号会自动变为空心的小方框，即变为禁止使用。

也可以按照任何维的组合提取子数组， 下面的示意图显示了怎样从一个二维数组中提取一个一维的行或者列数组。



还可以从一个三维数组中提取一个二维数组， 方法是禁止两个索引端子， 或者通过禁止一个索引端子提取一个一维数组。下图显示了从三维数组提取数组的各种方法。



下面的规则对使用剪切数组进行了规定：

输出对象的维数必须等于被禁止的索引端口的数目。例如

0 个索引端口被禁止＝标量元素

1 个索引端口被禁止＝二维元素

2 个索引端口被禁止＝三维元素

启动的端子所连接的数值必须指定输出元素。

这样，您就可以理解，上图中左下方的例子的作用是，利用 0 列和 3 行的所有元素产生一个一维数组， 而右上方的例子的作用是利用第一帧中的所有元素产生一个二维数组。新的第 0 个元素是与原有元素最近的元素。

练习 十一 使用创建数组功能函数

目的：使用创建数组函数，把一些元素和输出组织成一个更大的数组。

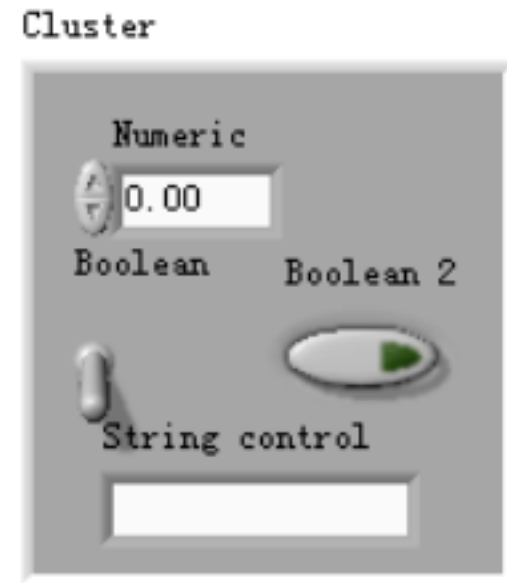
图 3－8 练习 十一 的面板和框图

1. 按照图 3－8 创建一个前面板。
2. 从 ControlsoNumeric 模板中选择一个数字控制对象放置在前面板中， 设置它的标签为 scalar 1 。
3. 复制并粘贴该数字显示对象，创建两个新的对象，并分别设置它们的标签为 scalar 2 和 scalar 3 。
4. 创建一个数字控制对象的数组，设置它的标签为 array 1 。复制并粘贴它，创建一个新的数组，设置它的标签为 array 2 。
5. 在 array 1 、 scalar 1 、 scalar 2 、scalar 3 、array 2 中输入数值 1 到 9。
6. 创建流程图。选择 FunctionsoArray ，在流程图中放置一个 Build Array 功能函数。用定位工具增大函数额面积，以容纳 5 个输入。
7. 把数组和标量与 Build Array 连接起来。创建输出的一维数组， 它由 array 1、scalar 1 、 scalar 2 、array 2 、scalar 3 中的元素所组成，如图所示。
8. 执行该 VI。可以看到 array 1 、scalar 1 、 scalar 2 、scalar 3 、array 2 中的数值出现在同一个一维数组中。
9. 保存该 VI。

练习 十一 结束。

3．5 簇

3．5．1　创建簇控制和显示



在前面板上放置一个簇壳（ Cluster shell ）就创建了一个簇。然后你可以将前面板上的任何对象放在簇中。例如数组，你也可以直接从 Control 工具板上直接拖取对象堆放到簇中。一个簇中的对象必须全部是 Control ，或全是 Indicator ，不能在同一个簇中组合 Control 与 Indicator ，因为簇本身的属性必须是其中之一。一个簇将是 Control 或 Indicator ，取决于其内的第一个对象的状态。如果需要可以使用工具重置簇的大小。右图所示是一个含４个 Control 的簇。也可以在流程图上用类似的方法创建簇常数。

如果你要求簇严格地符合簇内对象的大小，可在簇的边界上弹出快速菜单选择自动定义大小（ Autosizing ）

簇的序（ Order）

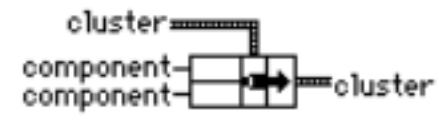
簇的元素有一个序，它与簇内元素的位置无关。簇内第一个元素的序为 0，第二个是 1，等等。如果你删除了一个元素，序号将自动调整。如果你想将一个簇与另一个簇连接，这两个簇的序和类型必须同一。

如果想改变簇内元素的序，可在快速菜单中选择 ReOrder Controls In Claster ，这时会出现一个窗口，在该窗口内可以修改序。

3．5．2　使用簇与子 VI 传递数据

一个 VI 的连接窗口最大有 28 个端子，如果你不希望使用全部 28 个端子传递数据，这既烦琐又易出错。通过把控制或显示对象捆绑成一个簇的方法，仅使用一个端子就可以实现该功能。

捆绑（ Bundle）数  据



Bundle 功能将分散的元件集合为一个新的簇，或允许你重置一个已有的簇中的元素。可以用位置工具拖曳其图标的右下角以增加输入端子的个数。最终簇的序是取决于被捆绑的输入的顺序。右图中 Bundle 图标中部的 Claster 端子用于用新元素重置原簇中的元素。

分解（ Unbundle）簇

Unbundle 功能是 Bundle 的逆过程，它将一个簇分解为若干分离的元件。如果你要对一个簇分解，就必须知道它的元素的个数。 LabVIEW还提供一种可以根据元素的名字来捆绑或分解簇的方法，稍后介绍。

练习 十二　簇

目的：学习创建簇、分解簇，再捆绑簇并且在另一个簇中显示其内容。

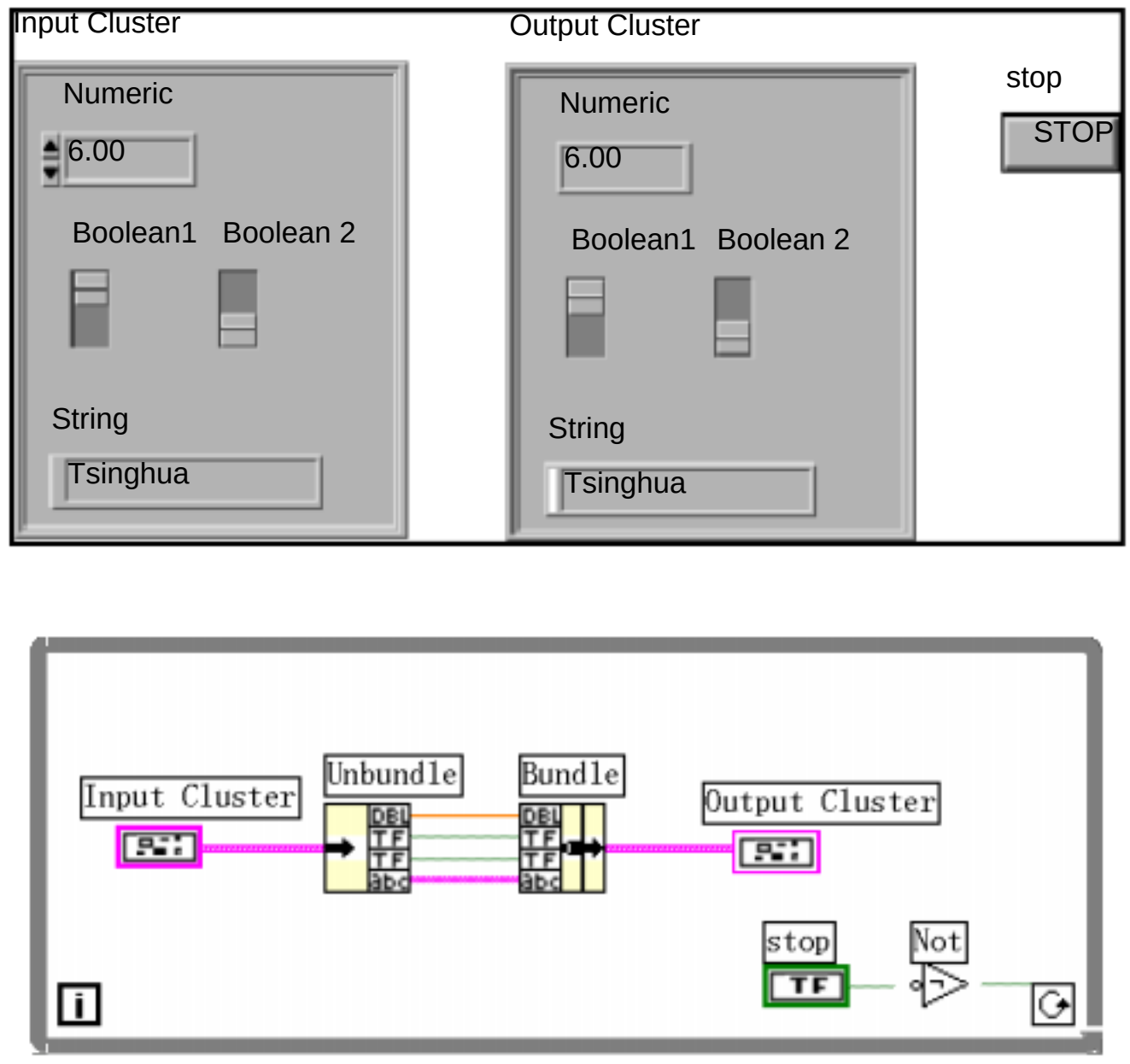


图３－１０　　练习３－４的面板和框图

１．打开前面板，创建一个簇壳（　Array & Cluster palette　），标签改为 Input Cluster ，拖曳至适当大小。

２．在这个簇壳中放置一个数字　Control ，两个布尔开关，和一个串　Control 。

３．仿照以上步骤，创建　Output Cluster　如上。注意将各　Control　改为相应的 indicator　。

４．用快速菜单查看两个簇的序是否一致，若有差别，改之。

５．在前面板上设置一个　[STOP]按钮。注意其缺省值为　FALSE，不要改变它的状态。

６．建立如上面所示的流程图。　注意在 [STOP]按钮与循环条件端子之间接入了一个 NOT函数，因为按钮缺省值为　FALSE，经 NOT函数后变为　TRUE，这就意味着当按钮状态不变时，循环继续执行，相反一旦按钮动作，则循环终止。

８．关闭并保存程序。

练习　十二　　结束

３．５．３　　用名称捆绑与分解簇

有时你并不需要汇集或分解整个簇，而仅仅需要对其一、两个元素操作。这时你可以用名称来捆绑与分解簇。在　　Cluster　工具模板中除了　　Bundle　及 Unbundle 功能外，还提供有　Bundle By Name 和 Unbundle By Name 功能。它们允许根据元素的名称 （而不是其位置） 来查询元素。与 Bundle 不同，使用 Bundle By Name可以访问你需要的元素，但不能创建新簇；它只能重置一个已经存在的簇的元素，同时你必须给　Bundle By Name 图标中间的输入端子一个输入以申明要替换其元素的簇。　Unbundle 可返回指定名称的簇元素，不必考虑簇的序和大

小。例如，如果你想重置上例中 Boolean 2 的值，就可以使用 Bundle By Name 功能而不必担心簇的序和大小。与此类似如果你要访问串的值，可以使用 Unbundle By Name 功能。

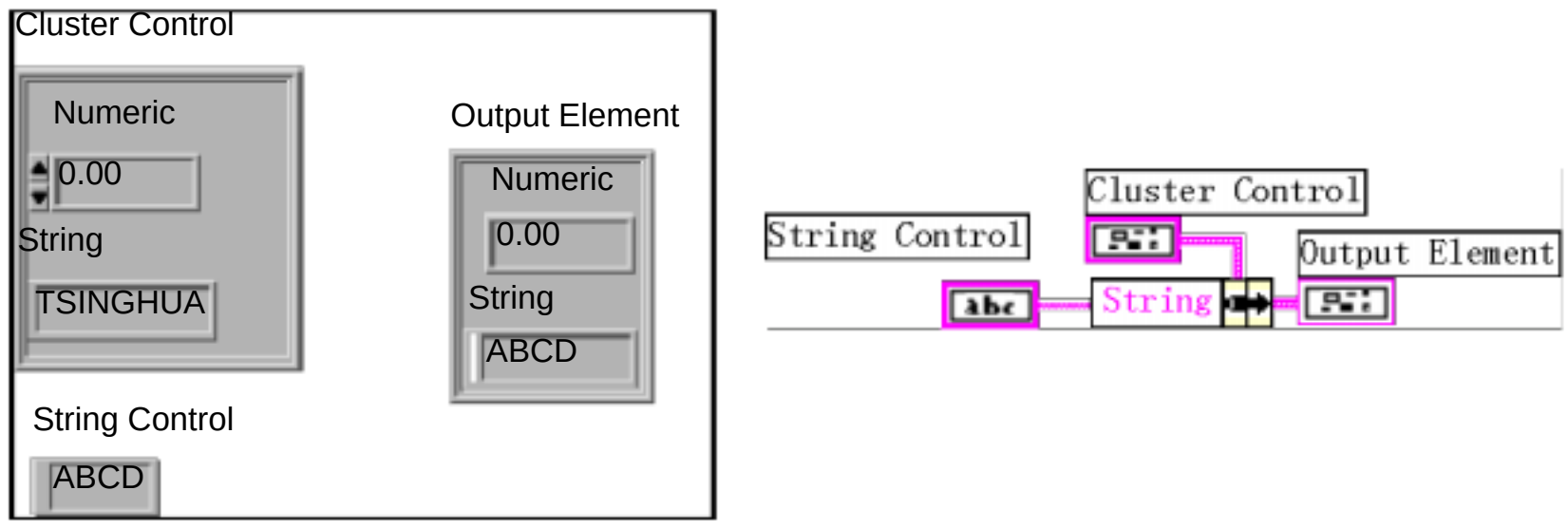


图 3－11 用名称操作簇

在上面的例子中， Cluster Control 中有两个元素，一个是数据类型（名称是 Numeric），另一个是字符串型（名称是 String ），另一个控制是字符串“ ABCD”，框图如右所示，运行该程序，即可将簇内的字符串值重置。（本例中为了使 Bundle By Name的输入端由 Numeric 变为 String ，需使用快速菜单中的 Select Item 项操作。）

3．5．4 数组和簇的互换

有时你会发现，将数组变为簇（或者相反）很方便。尤其是因为 LabVIEW包括的关于数组的操作功能多于簇。 例如，前面板上有一个多按钮的簇， 你希望颠倒这些按钮值的序。好了， Reverse 1D Array 功能正好可用，但是它仅可用于数组。这没关系，你可以使用功能 Cluster to Array 将簇转换为数组，使用 Reverse 1D Array 切换开关的值，最后再利用 Array to Claster 变换回簇。

3．6 Waveform数据类型

在数据采集和信号分析中经常要遇到波形数据，在 LabVIEW 6i 中增加了Waveform数据类型，使得波形的描述更加简洁。 Waveform数据类型包含了波形的数据 (Y) 、起始时刻 (t0) 和步长 ΔX, 使用 Waveform 模板的 Build Waveform 函数可以建立一个波形。许多用于数据采集和波形分析的 VI 和函数的缺省状态都接受或返回 Waveform数据类型。当你将一个 Waveform数据类型连接到 Waveform Graph 或 Chart 时，会自动画出相应的曲线。

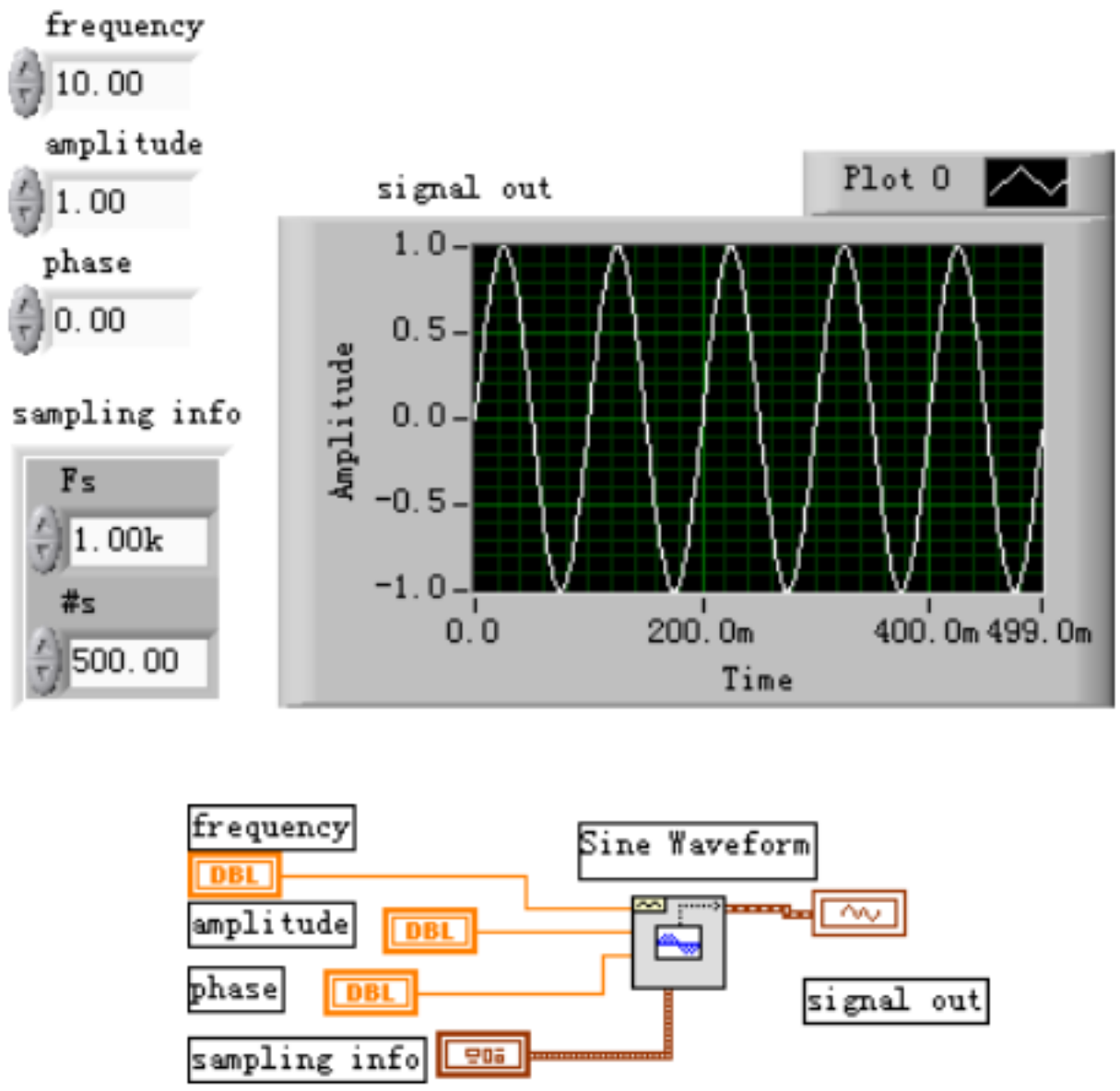


图 3－1 2　　使用 Waveform的波形发生例子

图 3－1 2是一个使用　Waveform 函数产生正弦波的例子。其中仅仅调用了 Sine Waveform一个函数，只要将有关参数指定，就可产生正弦波。Sine Waveform 实际上是一个子　VI，点击其图标，就可看到下层的程序，还是比较复杂的。在 LabVIEW 6I 以前的版本中用户就需要那样去编程。

Waveform数据类型是根据原有的数据类型进一步　“打包”组合而成，这种打包也不可避免地带来一些负作用，　有时还需要对　Waveform数据类型“解包”。有关这一数据类型的函数或　VI 在 Functionso　Waveform之中。它们有：

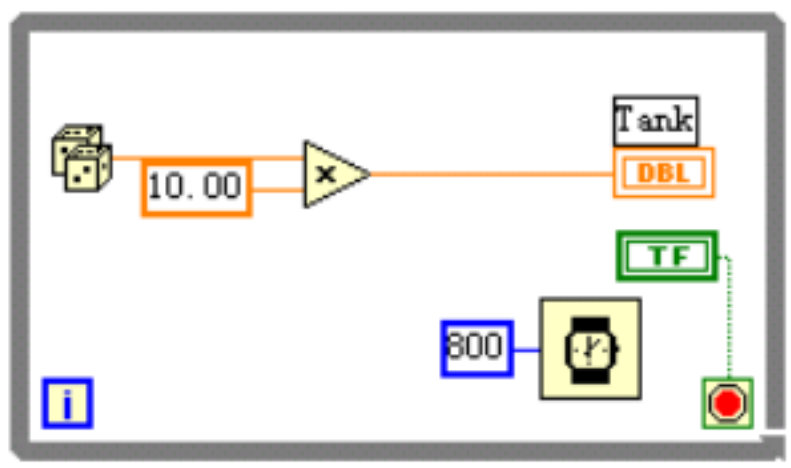
Build Waveform　：构造波形数据类型
Get Waveform Components：提取波形元素（ Y, Δ t,t0 ）
Set Waveform Attribute　　：设置波形属性
Get Waveform Attribute　：提取 Waveform属性

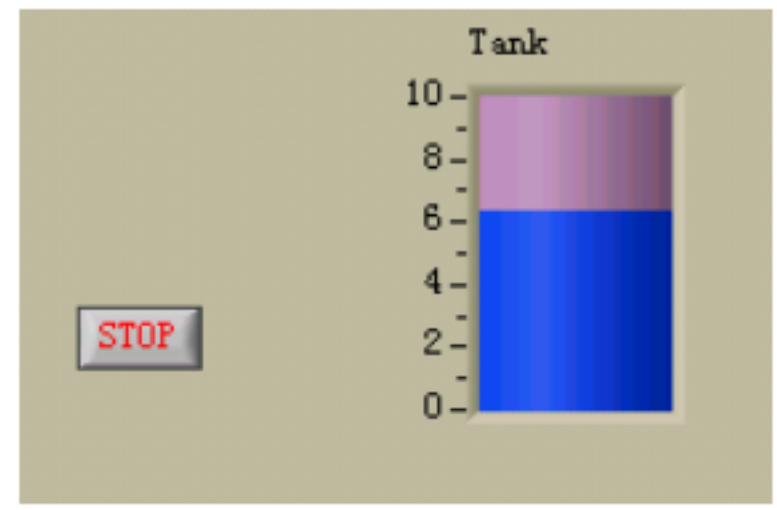
Waveform Operations ：波形运算
Waveform Generation　：波形发生
Waveform Measurements ：与波形相关的测量
Waveform File I/O　　：Waveform格式的文件 I/O

其中后 4 个是类目，分别还有下一层函数。

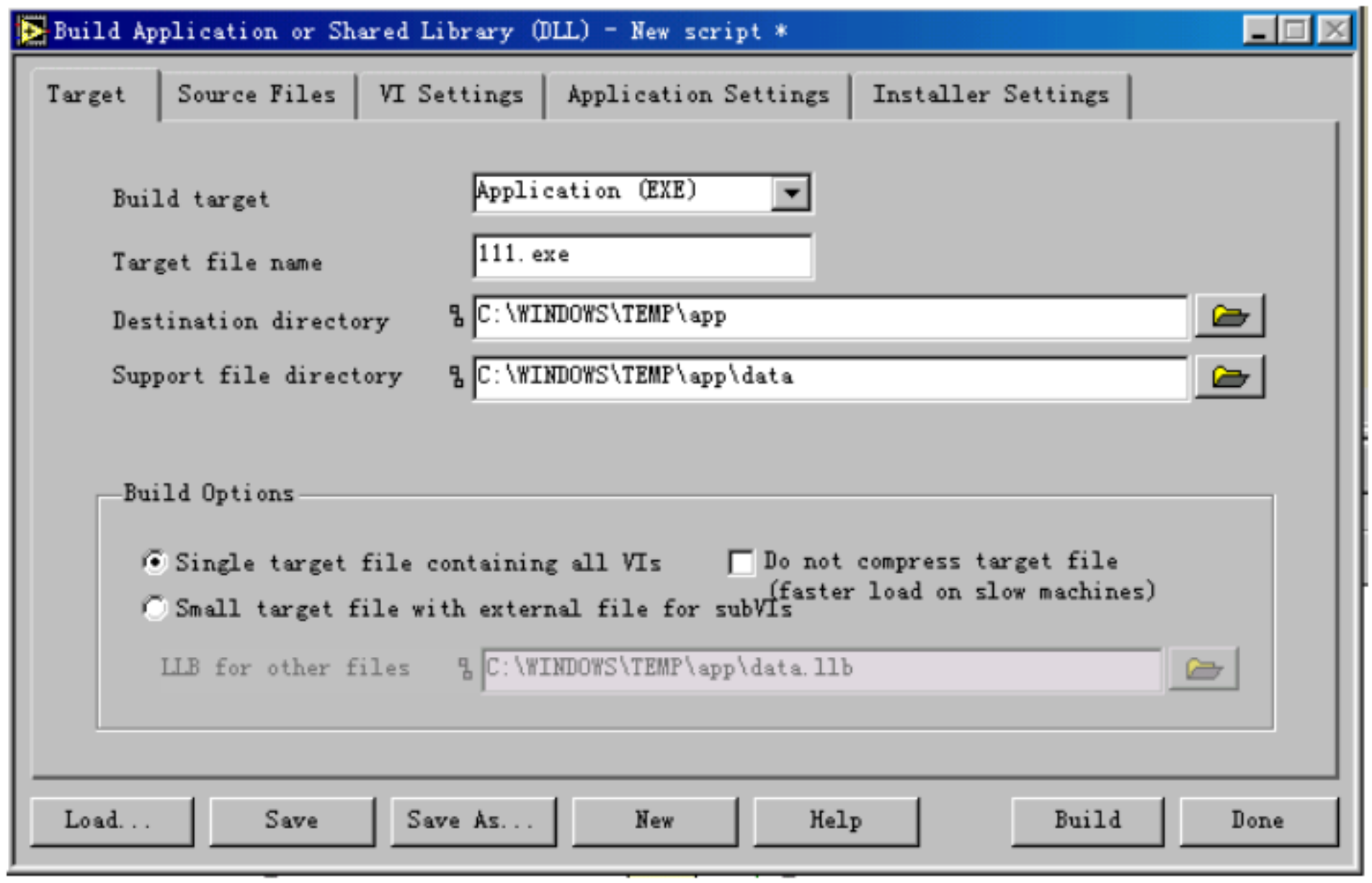
补充部分：安装文件的制作

首先，编写一个完整的程序并存盘，关闭文件。例如：





第二步：点击 tools 中的下拉菜单 Build Application or shared library，出现如下画面



在“ Target” 菜单中将“ Target File Name”改为文件名 111.exe。

在“ Source Files”菜单中添加准备生成安装文件的 VI 及其所需的一些支持文件。点击“ Add Top-Level VI …”，选中要添加的文件，另外如果程序中包含有子程序，在“ Add Support File …”中添加。

在“ Application Settings” 菜单中将“ Show LABVIEW Real-Time target selection dialog when launched”前的对勾去掉，该项功能为程序运行时选择在本机还是网络上运行，去掉对勾即默认在本机运行。

在“ Installer Settings” 菜单中选择“ Create installer”然后点击“ Build ”，生成安装文件。点击“ Done ”,选择“ No ”。安装文件默认的保存路径为 C:\WINDOWS\TEMP\app 。

第三步：安装文件。

在 C:\WINDOWS\TEMP\app\Installer\disks 中有 Setup 文件，点击安装。安装完后提示安装“ Welcome to the LABVIEW 6.0 Run-Time Engine Insallation Wizard ”。按照提示，完成安装。

第四步：运行程序。这时我们可以看到程序运行已经脱离 LABVIEW 环境。

对于程序界面的修改，可以在 LABVIEW 环境下的 File\VI Prorerties 中进行设置，在 Category中选择 Windows Appearance，在该菜单下的 Windows Title 中选择 Customize…，在出现的对话框中进行选择。

课 程 设 计 步 骤

1、 用 Function/waveform/waveform Genaration/sin waveform.vi 产生正弦波。

2、 设置波形数据，包括振幅、相位、频率、采样率的设置。

3、 用 Function/waveform/Get Waveform Components，从得到的正弦波输出（簇数据）中提取正弦波数据（数据类型不匹配，这里相当于从簇中提取数组数据）。

4、 用加法器把几个波形数据累加，得到输出。

5、 加开关，可以控制是否输入，用 case结构。

6、 加李萨如图形显示。（把 300、500、700Hz 信号通过累加器累加，并与 100Hz 信号通过 Bundle 函数合成，送到 XY Gaph 显示。）

7、 界面装饰。