第 3 章

基本数据类型

在阐述本章内容之前，首先对 LabVIEW 中有关数据的 3 个概念进行梳理。

第 1 个概念是数据的组织形式，即数据以什么形式在虚拟仪器程序即 VI 中加以呈现。在 LabVIEW 中，数据的组织形式有 3 种，分别是输入控件、显示控件和常量。其中，输入控件和显示控件都在前面板上的控件选板上；而常量，却是在程序框图面板的函数选板上。一般而言，输入控件是用来输入参数的；而显示控件，则用来显示 VI 的测量、分析、计算及处理结果。

第 2 个概念是数据的表现形式。以数值型数据为例，如图 3.1 所示，它可以表现为数值输入控件、仪表(表盘)、量表和滑动杆等多种形式，它们都是从实际需求中衍生而来的。实际生活和工作场景中，有各式各样的测量仪表，如温度计、速度计、电能表、水表，等等，虽然它们的外观各不相同，所反映的物理量也不同，但其大小属性是相同的，即都是数值。

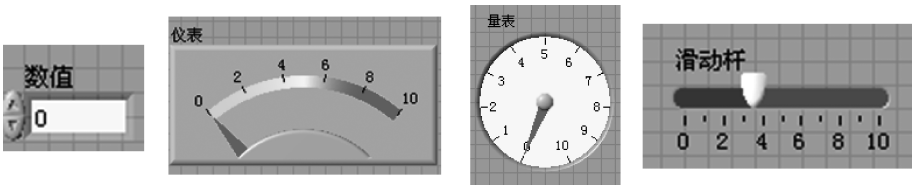


图 3.1　数值的表现形式

第 3 个概念是数据类型。LabVIEW 中，除了基本的数据类型，例如数值、字符串和布尔量等之外，还提供有复合数据类型，包括数组、簇、波形和 DDT 等[1-4]。本章主要学习 LabVIEW 中的基本数据类型，主要有数值、字符串、布尔量、枚举/下拉列表和路径。

3.1　数值

本节介绍最基本的数据类型——数值。在 LabVIEW 中，数值控件有很多种表现形式，并且还提供有很多个对数值的操作函数。

3.1.1　数值控件

数值控件又分为数值输入控件和数值显示控件，它们均位于“控件”选板→“新式”→“数值”子选板上。数值输入控件和数值显示控件各自都有很多种表现形式，如图 3.1 和图 3.2 所示。在控件选板上，数值输入控件和数值显示控件又分为新式、银色、系统和经典等，即还具有不同的风格，使用者可根据自己的喜好选择使用。

图 3.2　数值输入控件和数值显示控件

3.1.2　数值的数据类型

LabVIEW 以浮点数、定点数、整数、无符号整数以及复数等不同数据类型表示数值数据。那么，LabVIEW 中的数值数据类型是如何进行设置的呢？

下面，以一个数值输入控件为例进行介绍（数值显示控件以及常量是类似的）。首先，在前面板上创建一个数值输入控件，然后，经鼠标操作来到程序框图面板。这时，程序框图面板上已经出现了一个数值输入控件的图标，它与在前面板上生成的数值输入控件相对应，如

图 3.3 所示。此情况下，LabVIEW 默认生成的数值的数据类型为双精度 64 位实数。这个信息是如何得到的呢？一个办法是，通过查看该数值输入控件在程序框图面板上的显示图标来判断其当前的数据类型。因为在 LabVIEW 中，不同数据类型的数值控件的图标颜色和形式是不一样的，如图 3.3 所示的数值输入控件的图标是橙色的，而且下面有标识“DBL”，这表明，该数值输入控件中的数据当前的数据类型为双精度浮点数。LabVIEW 中的数值数据类型有多种，除了实数（橙色）和整数（蓝色）通过颜色可以快速地辨识出来外，想要知道某数值输入控件中当前的具体数据信息，仅靠其图标上的标识来判断，还不能保证准确无误。鉴于此，一个简便、可靠的办法，是调用 LabVIEW 的即时帮助功能。具体地，在程序框图上，选中所关注的数值输入控件的图标，然后按下 Ctrl＋H 组合键，就会在程序框图面板上弹出一个即时帮助窗口，会显示出该数值输入控件当前的数值数据类型，如图 3.4 所示。

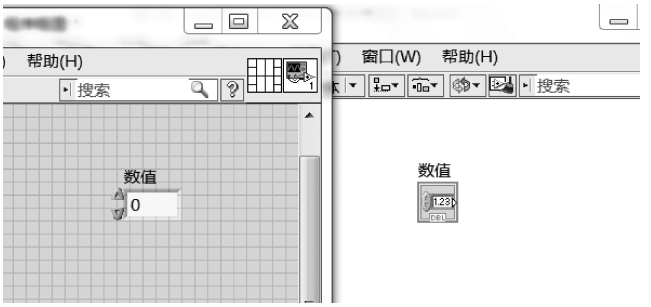


图 3.3　在前面板和程序框图面板中的数值输入控件

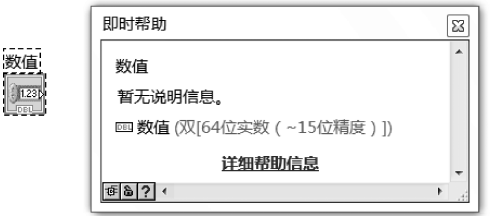


图 3.4　即时帮助中显示的数值输入控件的数据类型信息

另外，数值输入控件当前的数值数据类型也是可以改变的。如图 3.5 所示，改变数值输入控件当前的数据类型的方法如下：首先，在程序框图上选中所关注数值输入控件的图标，右击，选择“表示法”，可以看到共有 15 种数据类型，且当前选中的是“DBL”；改为选择下方的“I32”，随即，程序框图中该输入控件的图标就变成了蓝色，即时帮助窗口中给出的信息也改为 32 位的整数，如图 3.6 所示。如此，就将数值输入控件中的双精度浮点数改成了整型数。LabVIEW 中 15 种数值的数据类型各自的具体含义，请见表 3.1。

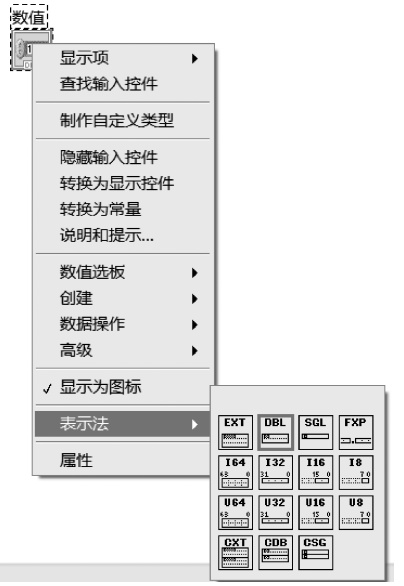


图 3.5　改变数值输入控件的数据类型

图 3.6　数值输入控件的数据类型为整型

表 3.1　LabVIEW 的 15 种数值数据类型

缩写	含　义
EXT	扩展精度浮点数，保存到存储介质时，LabVIEW 会将其保存为独立于平台的 128 位格式。内存中，数据的大小和精度会根据平台的不同而有所不同，只在的确有需要时，才会使用扩展精度的浮点型数值。扩展精度浮点数的算术运行速度，会因所使用平台的不同而有所不同
DBL	双精度浮点数，具有 64 位 IEEE 双精度格式，是双精度下数值对象的默认格式，即大多数情况下，应使用双精度浮点数
SGL	单精度浮点数，具有 32 位 IEEE 单精度格式。如所用计算机的内存空间有限，且实施的应用和计算等绝对不会出现数值范围溢出情况，应使用单精度浮点数
FXP	定点型
I64	64 位整型　　（－1e19～1e19）
I32	有符号长整型　　（－2 147 483 648～2 147 483 647）
I16	双字节整型　　（－32 768～32 767）

续表

缩写	含　义	
I8	单字节整型	(－128～127)
U64	无符号 64 位整型	(0～2e19)
U32	无符号长整型	(0～4 294 967 295)
U16	无符号双字节整型	(0～65 535)
U8	无符号单字节整型	(0～255)
CXT	扩展精度浮点复数	
CDB	双精度浮点复数	
CSG	单精度浮点复数	

数据类型是一个很基础的概念,不难懂,但是要学清楚,否则 VI 运行中出现问题时,可能很难找到出错的原因。在进行 VI 编程时,特别要注意对数据类型的正确使用。下面以例 3.1 进行说明。

【例 3.1】 查错示例:“求平均数。”

在第 2 章中,已经编写出了求平均数的 VI。对于求平均数这个命题,有的初学者编写的 VI 如图 3.7 和图 3.8 所示。可以看到,其中的 Result 显示控件是蓝色的,表明它当中的数据是整型的。而且,在除数即数值常量 2 与除法函数相连处出现了一个红点——表示这里发生了数据类型的强制转换,即整型数被转换成了浮点数。同样,在 Result 显示控件的输入端子上也出现了一个红点,这是因为,橙色的连线代表传输的是浮点数,而蓝色的 Result 显示控件代表接收到的应是整型数据,所以,在此处又发生了数据类型的强制转换。

这个 VI 通过了程序编译,并没有语法上的错误,但是当它运行完毕后,就会出现错误。如图 3.7 所示,当输入 1 和 2,结果本应该是 1.5,但此 VI 的计算结果却为 2。问题就出在 Result 控件的数据类型上。回到该 VI 的程序框图上,将 Result 显示控件的数据类型改为“DBL”即双精度浮点数,然后再运行此 VI,就会得到正确的结果了。

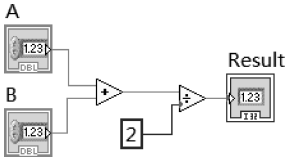


图 3.7　求平均数 VI 的程序框图

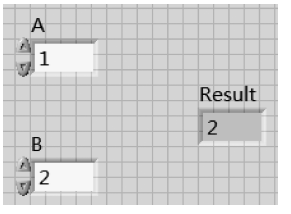


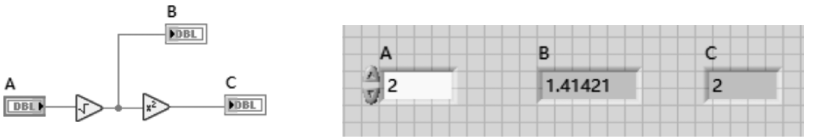
图 3.8　求平均数 VI 的前面板

在实际编程时,要注意数据类型的强制转换。在 LabVIEW 的程序框图中,如果连线上出现红点,则表示该处发生了数据类型的强制转换。在调试程序时,要格外注意这样的强制转换是否合适。

【例 3.2】 输入参数 A,求其平方根,结果为 B;然后再对 B 进行平方运算得到结果 C,

请问 A 和 C 相等吗?

为例 3.2 编写的 VI,如图 3.9 所示。在程序框图中,调用了“平方根”和“平方”两个函数,它们都位于“函数”选板→“编程”→“数值”子选板上。在前面板上,为输入控件 A 输入值“2”,然后运行该 VI,会显示出计算结果,如图 3.9(b)所示。可以看出,C 的计算结果也是 2,那么,是不是可以依此判断 A 与 C 是相等的呢?

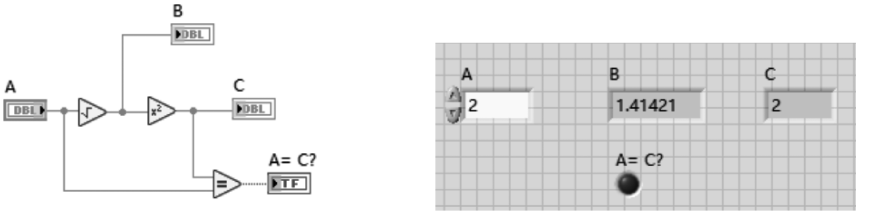


(a) 程序框图　　(b) 前面板

图 3.9　求平方根 VI

为了判断 A 与 C 是不是完全相等,可以在图 3.9 所示的 VI 基础上再进行编程实现。编写的 VI 如图 3.10 所示。其中,调用了“等于”函数,它位于“函数”选板→“编程”→“比较”子选板上。“等于”函数的结果为一个布尔量。当“等于”函数的两个输入参数相等时,输出结果为真;不相等时,输出结果为假。

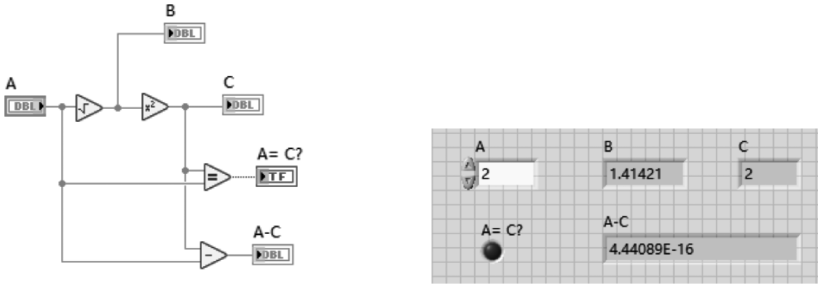
同样,在前面板上将输入控件 A 的值设为 2,运行 VI,会发现布尔量为假,如图 3.10(b)所示。VI 运行结果表明:A 和 C 是不相等的。



(a) 程序框图　　(b) 前面板

图 3.10　判断是否相等

将 A 与 C 进行相减运算,程序框图如图 3.11(a)所示,运行结果如图 3.11(b)所示。可以看出,A 减 C 的结果并不为 0,而是等于一个很小的数。



(a) 程序框图　　(b) 前面板

图 3.11　判断是否相等

常见问题 9：如果两个数是浮点数类型，如何判断它们是否相等？

在实际编程时，如果要比较两个浮点数的大小，要格外注意：要慎用“等于”运算，否则，程序可能会出现意想不到的错误。如例 3.2 所示，A 和 C 看似相等，但其实并不严格相等。那么，如果在实际编程时，需要比较两个浮点数的大小，该如何编程呢？常采用的方法是：将 A 和 C 做减法，取其绝对值，判断其绝对值是否小于一个很小的数。如果为真，则认为 A 和 C 近似相等。

3.1.3 数值函数

LabVIEW 提供有很多个用于操作数值的函数（也有教材称为“数值操作函数”），它们均位于“函数”选板→“编程”→“数值”子选板上，如图 3.12 所示。这些操作数值的函数图标都很形象，使用起来也比较简单，可以根据实际需求选择相应的数值函数。在“数值”子选板之下的“转换”子选板上，如图 3.13 所示，提供有很多个可实现数值数据类型转换的函数，如此，就可以通过编程的方式改变数值的数据类型了。

图 3.12 操作数值的函数

下面，通过例 3.3，介绍“随机数”函数和“表达式节点”的使用要点。

【例 3.3】 “随机数”函数和“表达式节点”函数的使用。

为例 3.3 编写好的 VI 如图 3.14 所示，其中调用了“表达式节点”函数。“表达式节点”

图 3.13　转换子选板

函数用于计算含有单个变量的表达式。使用“表达式节点”函数时，要注意采用正确的语法、运算符和函数，具体内容请参考 LabVIEW 的帮助文件。

图 3.14　例 3.3 的 VI 的程序框图和前面板

“随机数”函数的图标，外观看起来像两个错落放置在一起的骰子，调用它，可以生成数值范围在 0～1 的一个随机数，在需要生成随机信号的编程场合，经常会用到它。

3.2　字符串

LabVIEW 中，字符串是指 ASCII 字符的集合，用于文本传送、文本显示及数据存储等。在对数字化仪器和设备进行控制操作时，控制命令和数据等大多是按字符串格式加以传输的。

3.2.1 字符串控件

LabVIEW 中的字符串控件，位于“控件”选板→“新式”→“字符串与路径”子选板和“列表与表格”子选板上。字符串控件也分为输入控件和显示控件两种。

图 3.15 展示的是字符串组合框控件的使用示例。该控件可以写入多个字符串，每个称为一个“项”，并对应一个“值”。选中组合框控件，右击，弹出快捷菜单，选择“属性”→“编辑项”，可对“项”和“值”进行编辑，如图 3.16 所示。

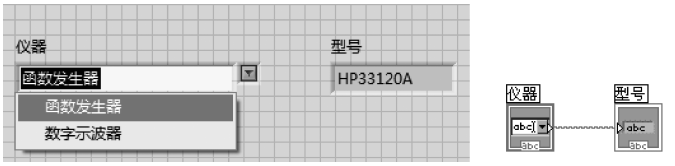


图 3.15 字符串组合框控件

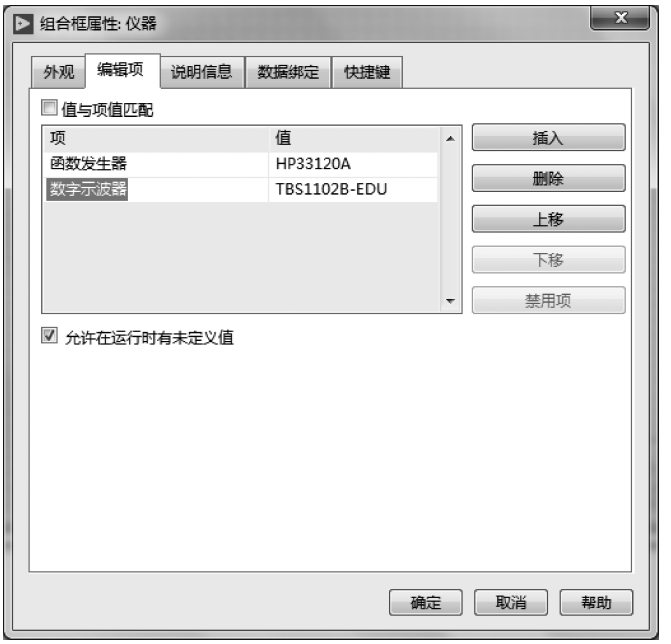


图 3.16 字符串组合框控件的属性设置

3.2.2 字符串的显示方式

字符串的显示方式有四种：第一种是 Normal Display，即正常显示，它是字符串控件的默认设置；第二种是\Codes Display，即\代码显示，用以查看在正常方式下不可显示的字符代码，其在程序调试、向仪器设备传输字符时较为常用；第三种是 Password Display，即口

令显示，在这种显示方式下，用户输入的字符均改由字符 * 代替；第四种是 Hex Display，即十六进制显示，字符以对应的十六进制 ASCII 码的形式显示，在程序调试和 VI 通信上比较常用。图 3.17 所示的 VI，给出了同一段字符串的四种显示方式。LabVIEW 中的一些特殊字符及其含义，提供在表 3.2 中。

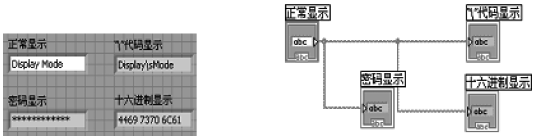


图 3.17　字符串的四种显示方式

表 3.2　LabVIEW 中的特殊字符

代码	LabVIEW 中的含义	代码	LabVIEW 中的含义
\b	退格符	\t	制表符
\f	进格符	\s	空格符
\n	换行符	\\	反斜线：\
\r	回车符	%%	百分比符号

3.2.3　字符串函数

LabVIEW 中提供有可对字符串进行操作的若干个函数，简称字符串函数，它们位于“函数”选板→“编程”→“字符串”子选板上，常用的字符串函数见表 3.3。下面将通过三个示例，对常用的字符串函数进行介绍。

表 3.3　字符串函数

序号	名　称	图标和连接端口	功 能 说 明
1	转换为大写字母	字符串 ~~~ 所有大写字母字符串	将输入字符串转换为大写形式
2	转换为小写字母	字符串 ~~~ 所有小写字母字符串	将输入字符串转换为小写形式
3	格式化写入字符串	格式字符串 初始字符串 ~~~ 结果字符串 错误输入(无错误) ~~~ 错误输出 输入1(0) … 输入n(0)	将字符串、数值、路径或布尔量转换为字符串格式
4	电子表格字符串至数组转换	分隔符(Tab) 格式字符串 电子表格字符串 ~~~ 数组 数组类型(2D Dbl)	将电子表格格式的字符串转换成数组

续表

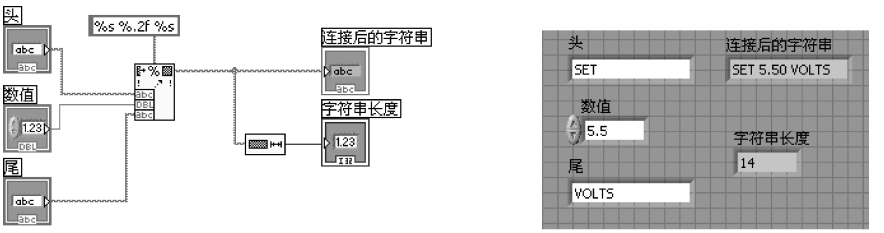
序号	名　称	图标和连接端口	功 能 说 明
5	格式化日期/时间字符串	时间格式化字符串(%c) 时间标识 UTC格式 日期/时间字符串	以指定的格式显示日期/时间字符串
6	字符串长度	字符串 长度	输出(提供)字符串长度
7	连接字符串	字符串0 字符串1 … 字符串n-1 连接的字符串	将几个字符串连接起来,组成一个新字符串
8	截取字符串	字符串 偏移量(0) 长度(剩余) 子字符串	从输入字符串的偏移量位置开始,取出所要求长度的子字符串
9	替换子字符串	字符串 子字符串("") 偏移量(0) 长度(子字符串长度) 结果字符串 替换子字符串	在字符串中的指定位置插入、删除或替换子字符串
10	扫描字符串	格式字符串 输入字符串 初始扫描位置 错误输入(无错误) 默认1(0dbl) … 剩余字符串 扫描后偏移量 错误输出 输出1 …	根据格式化字符串的要求,提取并转化字符串
11	搜索替换字符串	多行?(F) 忽略大小写?(F) 替换全部?(F) 输入字符串 搜索字符串 替换字符串("") 偏移量(0) 错误输入(无错误) 结果字符串 替换数量 替换后偏移量 错误输出	查找并替换指定的字符串
12	匹配正则表达式	字符串 正则表达式 偏移量(0) 子字符串之前 匹配子字符串 子字符串之后 匹配后偏移量	从偏移量开始,查找字符串的正则表达式,找到后,按它的位置将输入字符串分为三段

【例 3.4】 “格式化写入字符串”函数的使用。

为例 3.4 编写好的 VI 的程序框图如图 3.18(a)所示,其中调用了“格式化写入字符串”函数,将字符串“头”、数值和字符串“尾”连接在一起,生成新的字符串;并调用了“字符串长度”函数。该 VI 的前面板如图 3.18(b)所示,可见,在前面板上,是将字符串“头”设置为“SET”,将数值设为“5.5”,将字符串“尾”设为“VOLTS”。运行此 VI 可以看到,连接后的字符串为“SET 5.50 VOLTS”,且计算出了此字符串的长度为 14。

注意:“格式化写入字符串”函数图标边框上沿的中间处,是进行字符串连接的格式输

入端口，双击该函数图标，可以弹出对话框，如图 3.19 所示，在该对话框内，可对连接字符串的格式进行设置。



(a)　　(b)

图 3.18　格式化写入字符串函数应用举例 VI 的程序框图和前面板

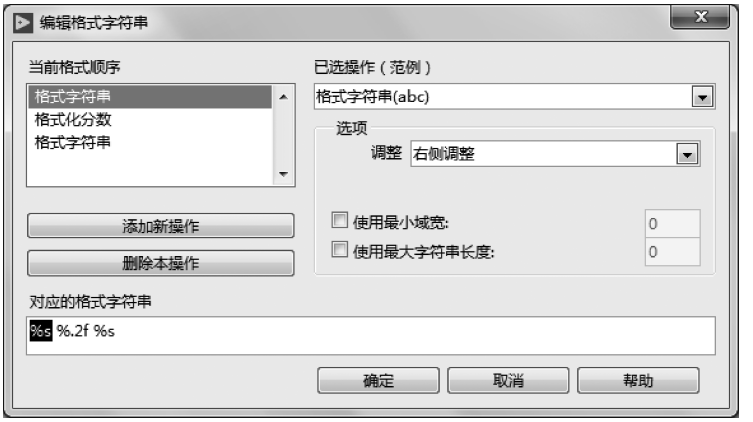


图 3.19　编辑字符串格式的界面

【例 3.5】　字符串的分解。

为例 3.5 编写的 VI 中，调用了“截取字符串”和“扫描字符串”函数，具体是要将输入字符串“VOLTS DC+1.345E+02”中的“DC”和数值“1.345 E+02”分解出来。该例题 VI 的程序框图和前面板，如图 3.20 所示。

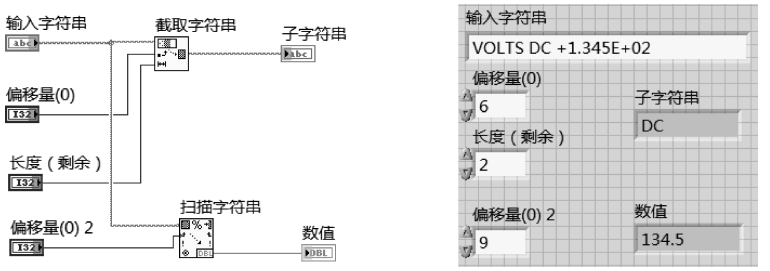


图 3.20　字符串分解示例 VI 的程序框图和前面板

在实际应用中，例如计算机从下位机（单片机）接收到的数据都是字符串类型的，那经常要做的一项工作，就是要从一段字符串中提取出实际感兴趣的信息。例 3.5 就实现了类似的功能，如提取出的“DC”，就表明是直流电压；提取出的“1.345E+02”，意味着获得了当前直流电压数值的大小。例 3.5 的实现方法，是已知要提取的元素在整个字符串中的位置，以此为根据，将所感兴趣的元素提取出来。那么，如果不知道所感兴趣元素的具体位置，又该如何实现上述目标呢？对此，例 3.6 给出了另外一种实现思路。

【例 3.6】 利用“匹配正则表达式”函数进行字符串的分解。

为例 3.6 编写的 VI 中，调用了“匹配正则表达式”函数，用以实现字符串的分解。该 VI 的前面板和程序框图，如图 3.21 所示，其中，[Dd]表示字符串第一个字符是大写或小写的 D，[Cc]表示字符串第二个字符是大写或小写的 C，如此，就将源字符串中的子字符串“DC”找到了，并将源字符串从“DC”处分解成了三段，匹配之前为 VOLTS，匹配之后为字符串“+1.345E+02”，再将其转换成数值类型，即输出数字“134.5”。

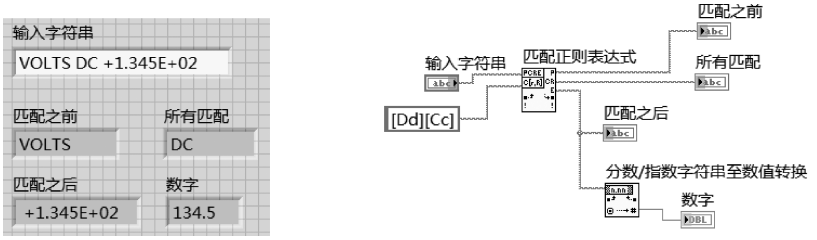


图 3.21 “匹配正则表达式”函数使用示例 VI 的前面板和程序框图

正则表达式的功能非常强大，例 3.6 只给出了一个简单应用示例。有关正则表达式的语法，请参看 LabVIEW 的帮助文件。从例 3.5 和例 3.6 的 VI 实现方式的比较可以看出，为实现相同的功能，利用 LabVIEW 可以有多种方法，故在实际进行编程时，要根据已知条件设计自己的 VI。

3.3 布尔量

布尔量只有两个状态，即，要么真，要么假。布尔控件位于“控件”选板→“新式”→“布尔”子选板上，如图 3.22 所示。与布尔量相对应，每个布尔控件都具有两个值，即真和假。布尔控件的表现形式有很多种，例如有指示灯、开关或按钮，等等。对布尔量实施操作的函数，简称布尔函数，它们位于“函数”选板→“编程”→“布尔”子选板上，如图 3.23 所示。

在使用按钮控件时，要注意其“机械动作”属性的设置。选中按钮控件，右击，在弹出的快捷菜单中选择“机械动作”，如图 3.24 所示。可以看到，LabVIEW 中定义有按钮的 6 种机械动作。按钮各种机械动作所代表的含义见表 3.4。

图 3.22　布尔控件子选板

图 3.23　布尔函数子选板

图 3.24　停止按钮的机械动作

表 3.4 LabVIEW 中按钮的机械动作

图标	含义	图标	含义	图标	含义
	单击时转换		释放时转换		保持转换直至释放
	单击时触发		释放时触发		保持触发直至释放

3.4 枚举与下拉列表

LabVIEW 中，枚举控件位于“控件”→“新式”→“下拉列表与枚举”子选板上，如图 3.25 所示。下拉列表和枚举，多用于 LabVIEW 编程中具有多个分支的情况，经常与条件结构配合使用。有关条件结构的具体使用方法将在第 4 章介绍。下面通过一个例子，介绍下拉列表和枚举控件的使用方法。

图 3.25 下拉列表和枚举控件

【例 3.7】 设计一个简易的计算器，当在其前面板上选择不同的功能时，它应给出相应的计算结果。

对此例，如图 3.26 所示，选中一个枚举控件，将其拖曳到前面板上；选中此控件，右击，在弹出的快捷菜单（如图 3.27 所示）中选择编辑项，如此，会弹出如图 3.28 所示的界面，随后，在项的表格中，可以输入项的名称，例如，在此例中输入“相加”，单击右侧的插入按钮，便可以添加新的项。以上述相同的操作，再创建另外两项“相乘”和“相减”功能，如图 3.29 所示。

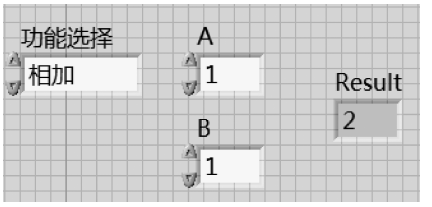


图 3.26 前面板

在为此例编写的 VI 的程序框图中，调用了一个条件结构，它位于“函数”选板→“编程”→“结构”子选板上。将“枚举”控件连至条件结构的选择器端子上，如此，条件结构会自动辨识出其中的两个分支，如图 3.30 所示。剩余的分支，需要再经手动添加上去。如图 3.31 所示，具体地，选中条件分支，右击，在弹出的快捷菜单中选择“在后面添加分支”，如此，就可将

后一分支设置好。而条件结构是按照这些分支在枚举控件中的值的属性依次添加的。例如，默认的分支是值 0 和 1，对应于本例而言，是“相乘”和“相减”。这样，继续添加的分支为值 2，与之对应的是“相加”。最终的三个分支如图 3.32 所示。然后，再在条件结构的各个分支中加入相应的代码，如图 3.33 所示。

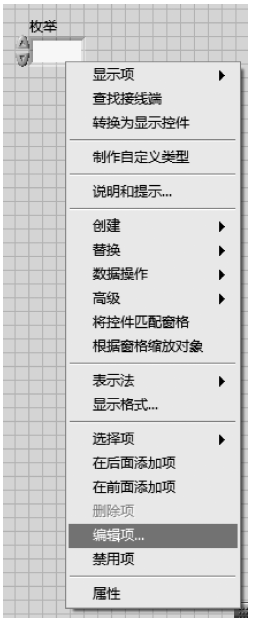


图 3.27　枚举控件的快捷菜单

图 3.28　编辑项界面

图 3.29　编辑项界面

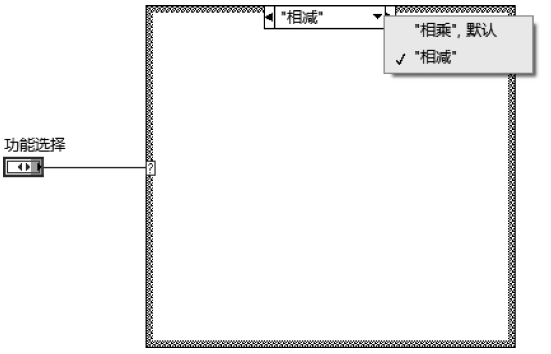


图 3.30　默认的两个分支

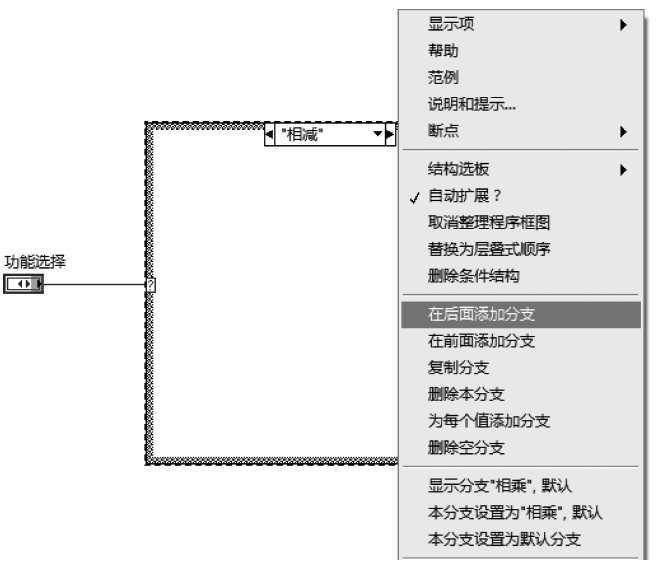


图 3.31　添加新的分支

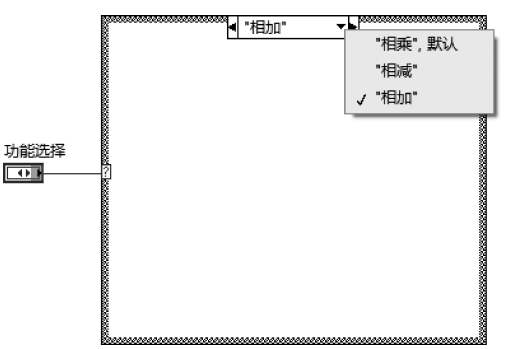
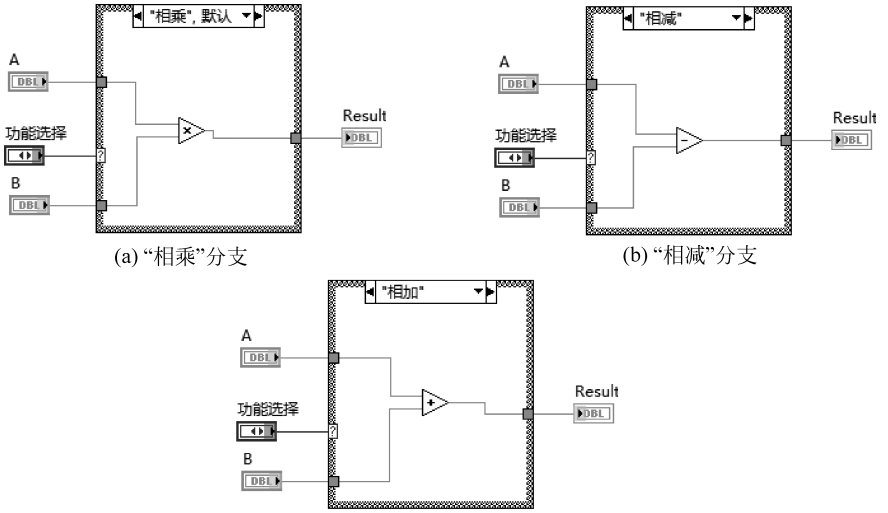


图 3.32　最终的三个分支



(a) “相乘”分支　(b) “相减”分支

(c) “相加”分支

图 3.33　例 3.7 简易计算器 VI 的程序框图

对例 3.7 所要求实现的功能编写 VI 时，也可改为利用“下拉列表”来实现。具体地，改写的 VI 的前面板和程序框图，如图 3.34 和图 3.35 所示。其中，利用“下拉列表”的道理与之前利用“枚举”控件是一样的，也是利用了条件结构。所以，这里只给出条件结构的一个分支的代码，而不再赘述。对“下拉列表”添加项和编辑项的操作方法，与对“枚举”控件的几乎一样，两者的区别，是当把“下拉列表”控件连至条件结构的选择器端子时，条件结构识别的不是标签，而是值，如图 3.35 所示。所以，使用“下拉列表”时，需要注意将前面板“下拉列表”的标签与条件结构中各个分支的值对应正确。

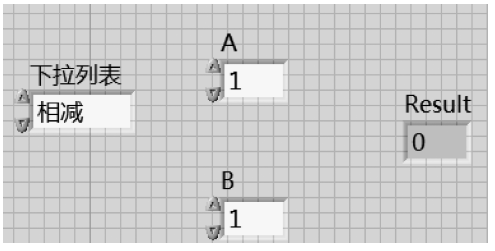


图 3.34　利用“下拉列表”实现的简易计算器 VI 的前面板

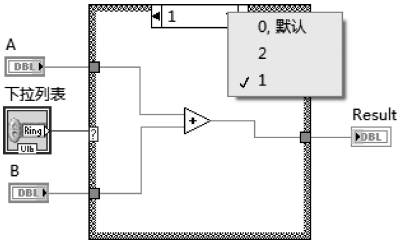


图 3.35　利用“下拉列表”实现的简易计算器 VI 的程序框图

在 LabVIEW 中，利用别的控件也可以实现上述功能，例如“滑动杆”控件、“组合框”控件等。使用“滑动杆”控件实现简易计算器的 VI 的前面板如图 3.36 所示。“滑动杆”控件位于“控件”选板→“新式”→“数值”子选板上。使用“滑动杆”控件时，需要进行以下设置，选中“滑动杆”控件，右击，在弹出的快捷菜单（如图 3.37 所示）中设置相关参数，具体包括：①选中“文本标签”；②在表示法中，将数据类型改为整型，如图 3.38 所示的 I8；③单击“属性”，在弹出的界面上输入文本标签值，如图 3.39 所示，这里的操作，与前述的“枚举”控件和“下拉列表”控件的操作相类似。

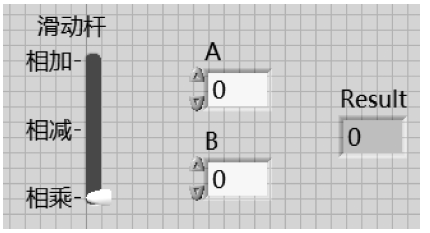


图 3.36　利用“滑动杆”实现的简易计算器 VI 的前面板

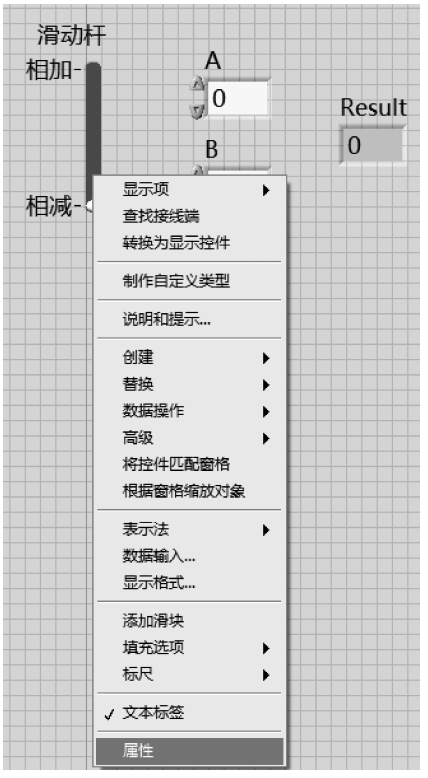


图 3.37　“滑动杆”的参数设置菜单

图 3.38　表示法设置

图 3.39　属性对话框

如图 3.40 所示，在利用“滑动杆”实现的简易计算器 VI 的程序框图中，当将“滑动杆”连接至条件结构的选择器标签上时，条件结构识别的也是“值”，即 0、1 和 2，所以，使用“滑动杆”控件时，也要注意条件结构中的分支要与“滑动杆”控件中的标签对应正确。

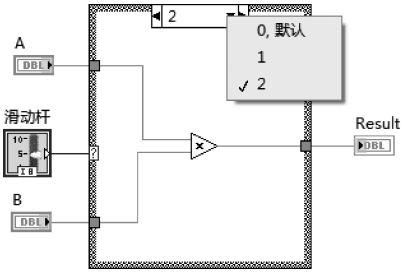


图 3.40　利用“滑动杆”控件实现的简易计算器 VI 的程序框图

在 3.2.1 节，曾学习过“组合框”控件，其数据类型属于字符串。按照图 3.16 所示的方法，编辑好“组合框”控件的“项”。对例 3.7 的命题，改用“组合框”控件实现的简易计算器 VI 的前面板和程序框图，分别如图 3.41 和图 3.42 所示。在该 VI 的程序框图中，将“组合框”控件连至条件结构的选择器端子上，随后，条件结构会自动识别两个分支(“真”和“假”)。注意，这里的“真”和“假”是带双引号的，所以是字符串类型。接下来，只需将“真”和“假”改成相应的标签，例如，“相加”和“相减”；因为存在三个分支，所以同前所述，还需要再添加新

的分支。

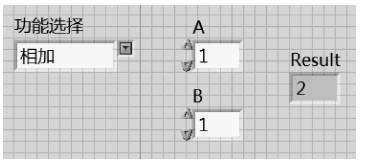


图 3.41　利用“组合框”控件实现的简易计算器 VI 的前面板

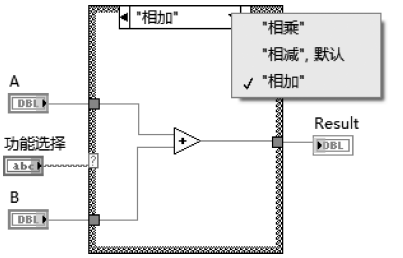


图 3.42　利用“组合框”控件实现的简易计算器 VI 的程序框图

可以看出，利用上面介绍的几种控件（“枚举”“下拉列表”“滑动杆”和“组合框”），都可以实现对多个不同状态的选择。

3.5　路径

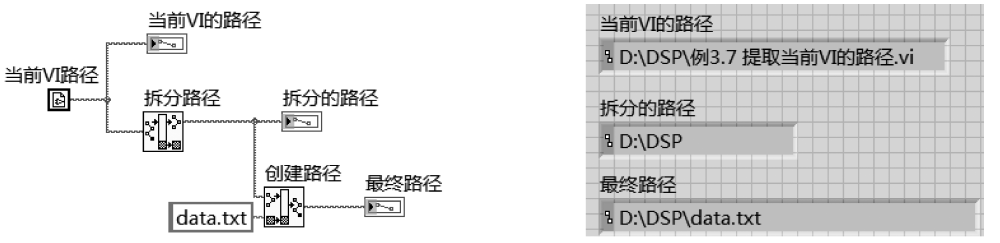
路径这种控件，位于“控件”选板→“新式”→“字符串与路径”子选板上，如图 3.43 所示。路径常量及函数，位于“函数”选板→“编程”→“文件 I/O” →“文件常量”子选板上，如图 3.44 所示。在 LabVIEW 中，路径用绿色表示。下面通过例 3.8 介绍 LabVIEW 中的路径操作。

图 3.43　路径控件

图 3.44　路径常量及函数

【例 3.8】 提取当前 VI 的路径。

这是利用 LabVIEW 编程时经常会用到的一个小功能，即如何获得当前 VI 的路径，一个编写好的实现该功能的 VI 的程序框图如图 3.45(a)所示。其中，调用了“当前 VI 的路径”函数，该函数位于“函数”选板→“编程”→“文件 I/O”→“文件常量”子选板上。从其前面板的运行结果(见图 3.45(b))，即控件“当前 VI 路径”的值可以看出，调用该函数得到的路径包含了当前 VI 的名称。而实际中，更希望得到此 VI 的位置，即要去掉 VI 名称之后，剩下前面的“D:\DSP”。这个功能，可以通过调用“拆分路径”函数实现，此函数位于“函数”选板→“编程”→“文件 I/O”子选板上。如此，如果想向此目录下写入一个新的文件，文件名称取名为“data.txt”，再调用“创建路径”函数，就可以得到新文件“data.txt”在 LabVIEW 中的路径了。



(a) 程序框图　　(b) 前面板

图 3.45　实现例 3.8 功能的 VI 的程序框图和前面板

3.6　本章小结

本章学习了 LabVIEW 中的基本数据类型，包括数值、字符串、布尔量、枚举和路径等，以及操作它们的函数。对于数值，要注意其数据类型的设置；在进行仪器控制和串口通信时，经常会用到字符串；在使用开关按钮控件时，要注意其机械动作的设置；对于多个状态的选择，可以使用枚举、下拉列表、滑动杆和组合框等控件来实现。

这些都是最基础的内容，并不难，需要理解清楚并能熟练使用。在接下来的章节中，还会频繁地使用到本章所介绍的这些基础知识。

本章习题

3.1　计算三角函数的值。在前面板上放置一个数值输入控件，分别求出其正弦和余弦值，并将结果输出显示在前面板上。

3.2　设计一个简易的计算器，在例 3.7 的基础上增加除法功能。

3.3　输入字符串“Current AC 1.2E-3 A”，提取出其中的子字符串“Current”“AC”和数值“0.0012”。

3.4　将字符串“SET”、数值“51.2”和字符串“Hz”连接在一起，生成新的字符串并计算出其长度，并且将所有结果在前面板显示出来。

3.5　实现指数函数的运算。在前面板输入数值 x，通过公式 $y=e^x$ 求出指数函数的值，并将结果输出到前面板上。

3.6　判断正负数。在前面板上输入数值 x，如果 $x>0$，指示灯变亮；反之，则指示灯为暗色。

参考文献

[1] Johnson G W, Jennings R. LabVIEW 图形编程[M]. 武嘉澍，陆劲昆，译. 北京：北京大学出版社，2002.

[2] 侯国屏，王珅，叶齐鑫. LabVIEW 7.1 编程与虚拟仪器设计[M]. 北京：清华大学出版社，2006.

[3] 黄松岭，吴静. 虚拟仪器设计基础教程[M]. 北京：清华大学出版社，2008.

[4] Bishop R H. LabVIEW 8 实用教程[M]. 乔瑞萍，林欣，译. 北京：电子工业出版社，2008.