华南师范大学

South China Normal University

LabVIEW 虚拟仪器程序设计

设计性物理实验课程

qq群：342873855

课前完成任务

1.安装LabVIEW

2. 小组合作完成任务：

查阅资料，了解研究现状（知网，外文数据库，论坛）

（1）虚拟仪器（远程实验）在教学中的应用（大学、中学）

（2）虚拟仪器（远程实验）前沿科技

（3）你的想法，解决什么教学问题（针对某个点）

下次课以小组形式汇报，要ppt，5分钟

第一讲 LabVIEW编程基础

1.一个vi及相关概念

2.数据类型

3.程序结构

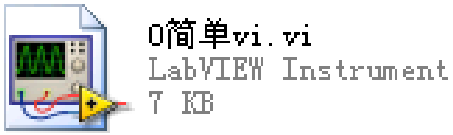
4.图形显示

一个vi及相关概念

什么是vi

vi：LabVIEW 程序被称为vi（Virtual Instrument），并以.vi作为扩展名

新建vi：“启动界面”或“文件”创建



一个vi及相关概念

LabVIEW编程环境

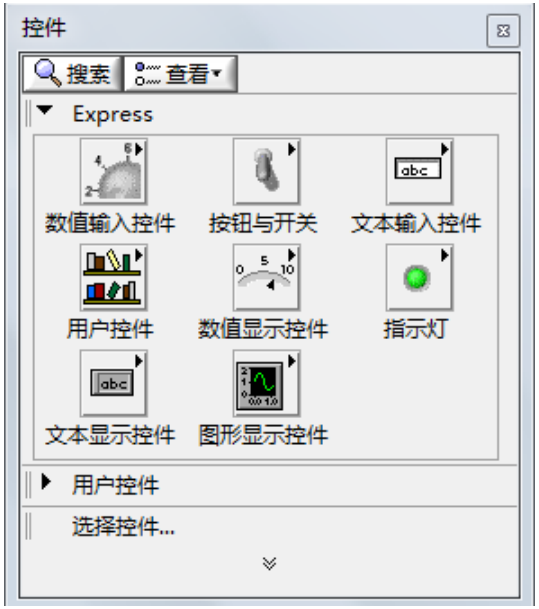
图a vi的前面板

前面板：图形用户界面，该界面上有交互式的输入和输出两类控件

一个vi及相关概念

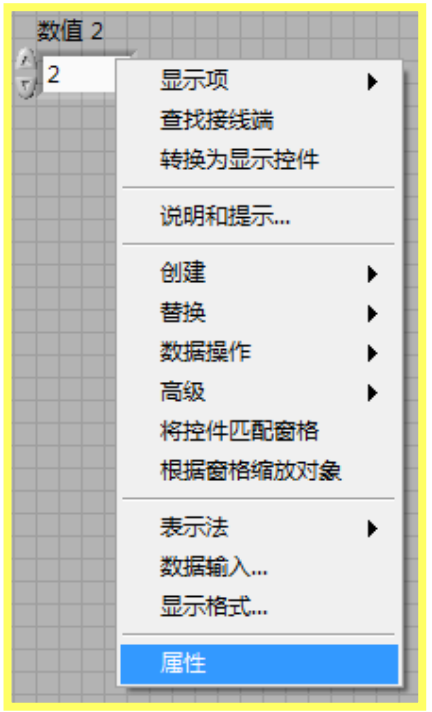
LabVIEW编程环境

编辑区:右键
打开控件选板



编辑区:Shift+右键
打开工具选板

控件:右键
打开控件选项



*控制介绍


数值输入控件
数值输入控件
水平填充滑...
水平指针滑...
垂直填充滑...
垂直指针滑...
旋钮
转盘
带边框颜色盒


数值显示控件
数值显示控件
水平进度条
水平刻度条
垂直进度条
垂直刻度条
仪表
量表
液罐
温度计


按钮与开关
翘板开关
垂直翘板开关
滑动开关
垂直滑动杆...
水平摇杆开关
垂直摇杆开关
开关按钮
OFF
OK
CANCEL
STOP
文本按钮
确定按钮
取消按钮
停止按钮

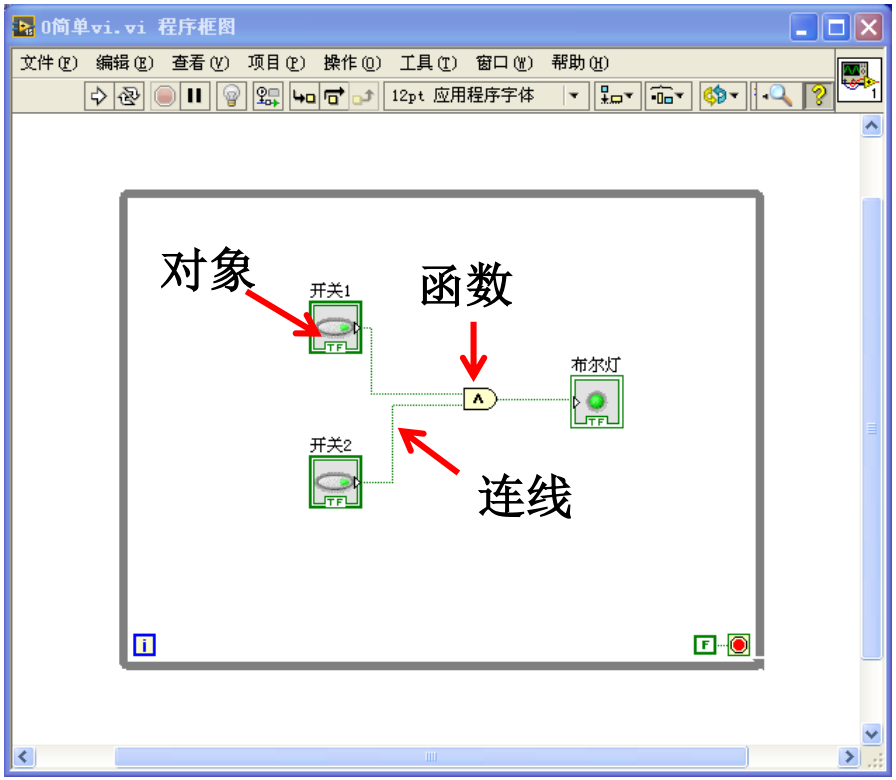

修饰
细线
垂直平滑盒
上凸盒
平面盒
下凹盒
粗线
水平平滑盒
上凸框
平面框
下凹框
细分隔线
水平按钮框
上凸圆形
平面圆形
下凹圆形
粗分隔线
上凸盒
上凸圆盒
平面圆盒
下凹圆盒
带箭头细线
下凹圆盒
上凸左向三...
平面左向三...
下凹左向三...
带箭头粗线
加粗下凹盒
上凸右向三...
平面右向三...
下凹右向三...


下拉列表与枚举
文本下拉列表
菜单下拉列表
枚举
图片下拉列表
文本与图片...

一个vi及相关概念

LabVIEW编程环境

窗口→显示程序框图
或 ctrl + E



图b vi的程序框图

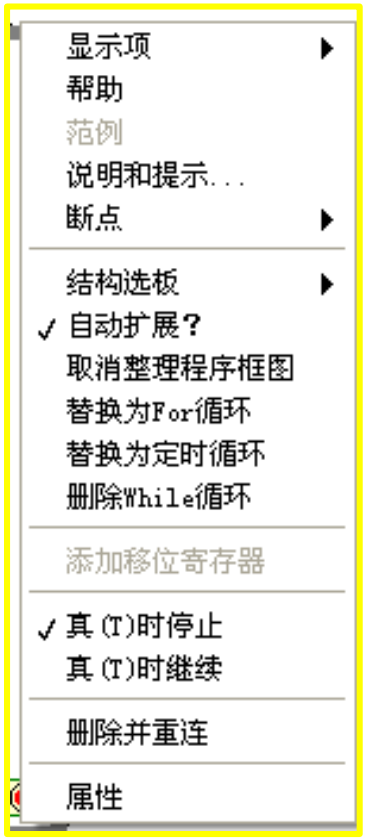
程序框图：图形化源代码的集合，图形化编程语言也称G语言。

一个vi及相关概念

LabVIEW编程环境

编辑区:右键
打开函数选板

函数:右键
打开函数选项



帮助→显示即时帮助
或 ctrl+H

*数据运算

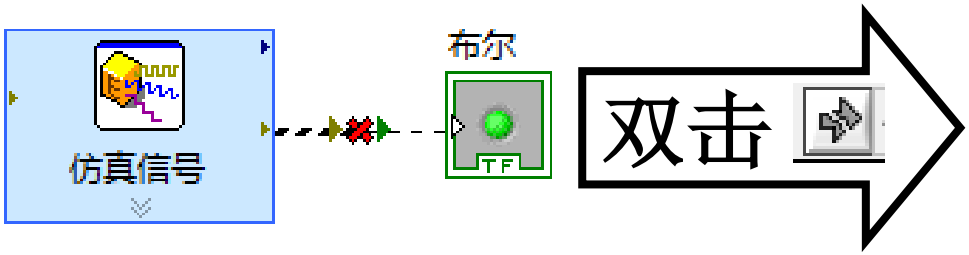
❖ 算术运算符

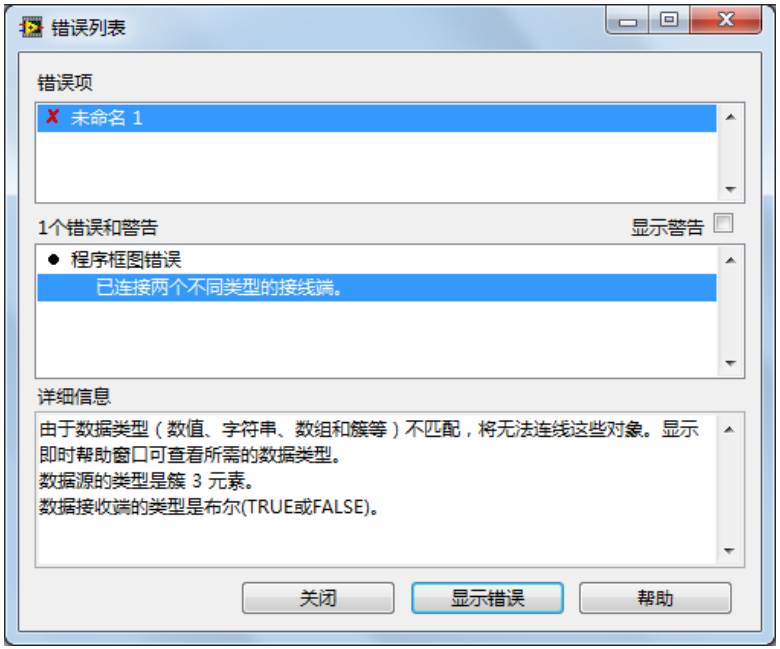
❖ 关系运算符

❖ 逻辑运算符

程序调试

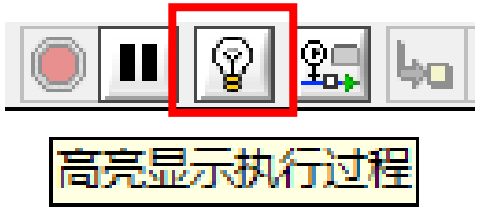
1.错误查找


布尔
TF
仿真信号
双击


错误列表
错误项
未命名 1
1个错误和警告
显示警告
程序框图错误
已连接两个不同类型的接线端。
详细信息
由于数据类型（数值、字符串、数组和簇等）不匹配，将无法连线这些对象。显示即时帮助窗口可查看所需的数据类型。
数据源的类型是簇 3 元素。
数据接收端的类型是布尔(TRUE或FALSE)。
关闭
显示错误
帮助

程序调试

2.高亮运行： 在程序框图观察数据流方向。



3.断点和探针设置：

断点表示程序执行到此处会暂停。探针查看当流程图程序流经某一根连接线时的数据值。

4.单步运行： 程序每执行一步都要停下来。

5.调试：

运行、连续运行、中止运行、暂停

数据类型

常见数据类型

图示	类型	简介
	数值数据类型	右键属性可以配置进制
	字符串数据类型	可以直接输入中文等字符
	布尔数据类型	布尔类型只有两个值，false 和 true
	数组	同种数据类型的多个数据的集合
	簇	不同种数据类型的多个数的集合
	枚举	一对数据(一个字符串和相应的数值)

连线类型	标量	一维数组	二维数组	颜色
数值				橙色（浮点数），蓝色（整数）
布尔型				绿色
字符串				粉红色

数据类型

数值型数据

- 数值型数据(右键→表示法修改类型)

*LabVIEW 数值型数据表

终端	数值数据类型	存储在磁盘的大小	数值范围
I8	单字节整型	8	–128 to 127
I16	双字节整型	16	–32,768 to 32,767
I32	长整型	32	–2,147,483,648 to 2,147,483,647
I64	64 位整型	64	$-1e^{19}$ to $1e^{19}$
U8	无符号单字节整型	8	0 to 255
U16	无符号双字节整型	16	0 to 65,535
U32	无符号长整型	32	0 to 4,294,967,295
U64	无符号 64 位整型	64	0 to $2e^{19}$
SGL	单精度	32	Minimum positive number: $1.40e^{-45}$ Maximum positive number: $3.40e^{+38}$ Minimum negative number: $-1.40e^{-45}$ Maximum negative number: $-3.40e^{+38}$
DBL	双精度	64	Minimum positive number: $4.94e^{-324}$ Maximum positive number: $1.79e^{+308}$ Minimum negative number: $-4.94e^{-324}$ Maximum negative number: $-1.79e^{+308}$

复习

计算机采用的数制是二进制。
一位（1bit）就是0或者1。

0 1

两位（2bit）,其中一位是高位，
一位是低位）就有四种情况了。

00 01 10 11

……
以此类推，八位就有2^8=256种情况了。

00000000 00000001 00000010 …… 11111111

我们把八位二进制定义为一个字节（1byte）。 256就可以描述（0~255）这256个整数了，这时256个数是无符号数；如果我们想用八位二进制表示有符号数，那么可以定义首位为符号位，首位为0是正数，首位为1是负数。这样剩下的七位就表示数值的绝对值①。所以有符号单字节的范围为（-128~127）。

01111111 ≡ +127　　10000000 ≡ -128

图2-1：十进制+127的八位二进制表示，十进制-128的八位二进制表示

练习：

比较两个整型数据的大小（输入），当数据a大于数据b时，布尔灯亮，否则不亮。

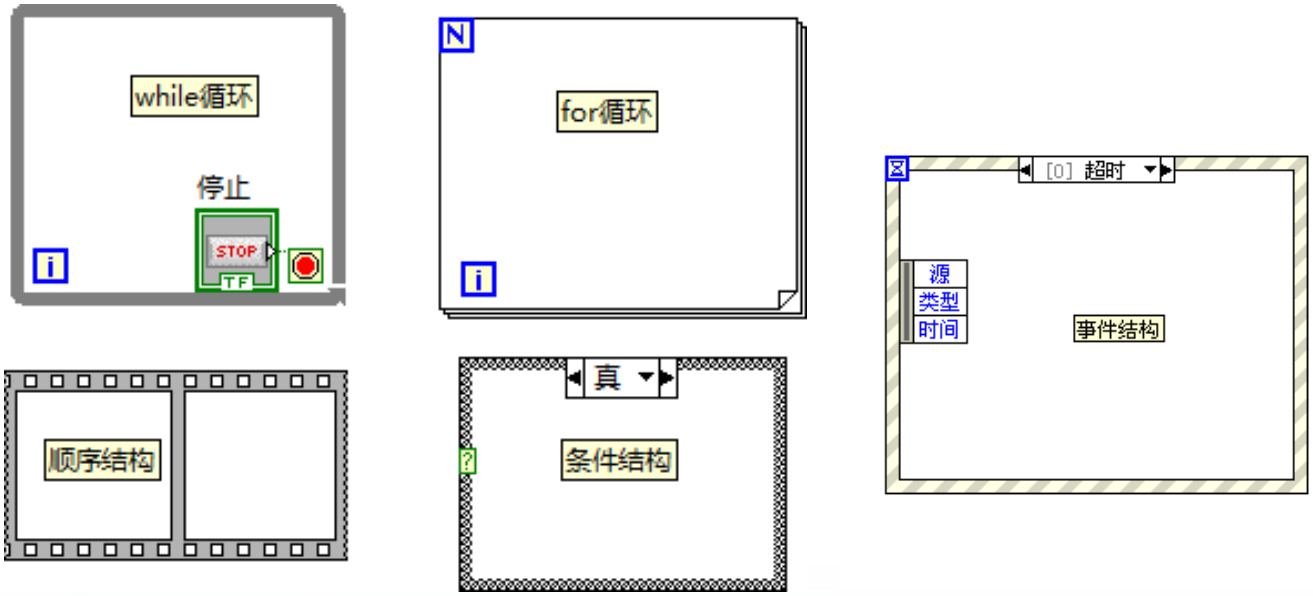
提示:在基本程序外加一个大的while循环把程序给框起来，保持程序的持续运行。

常用程序结构

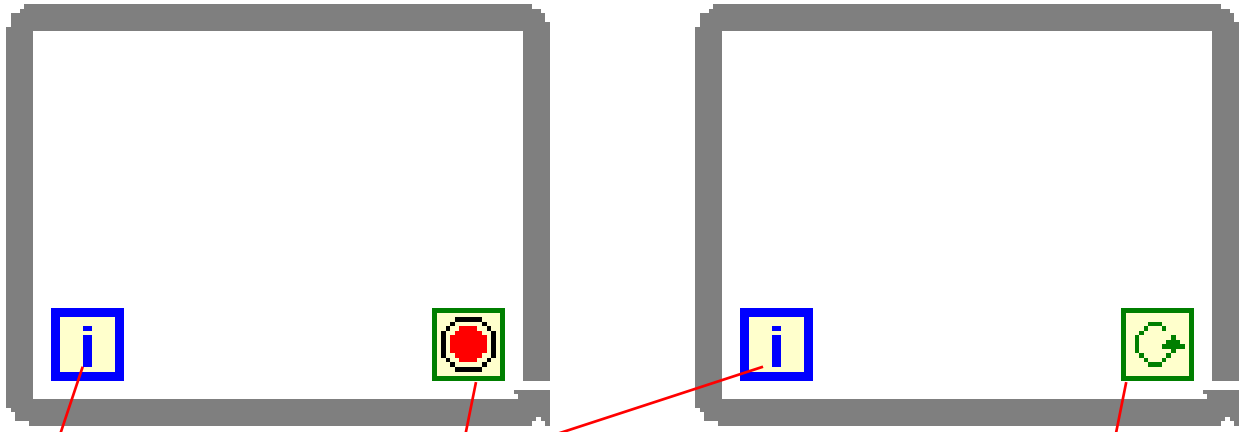
C/C++/Java中

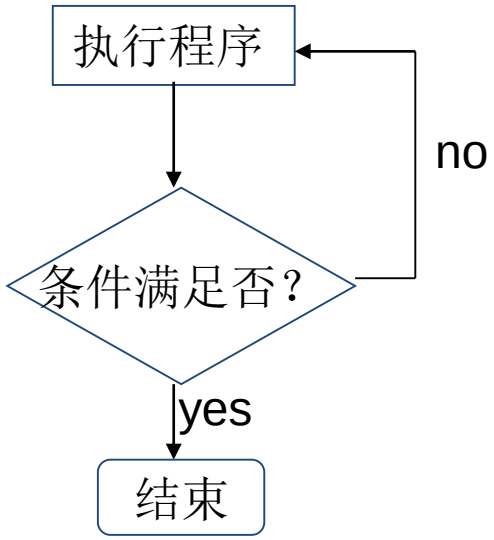
分支（选择）语句：if-else ,switch-case;

循环语句：while 循环,do-while循环,for循环;



循环结构——while循环





循环计数

循环条件

（为真停止）

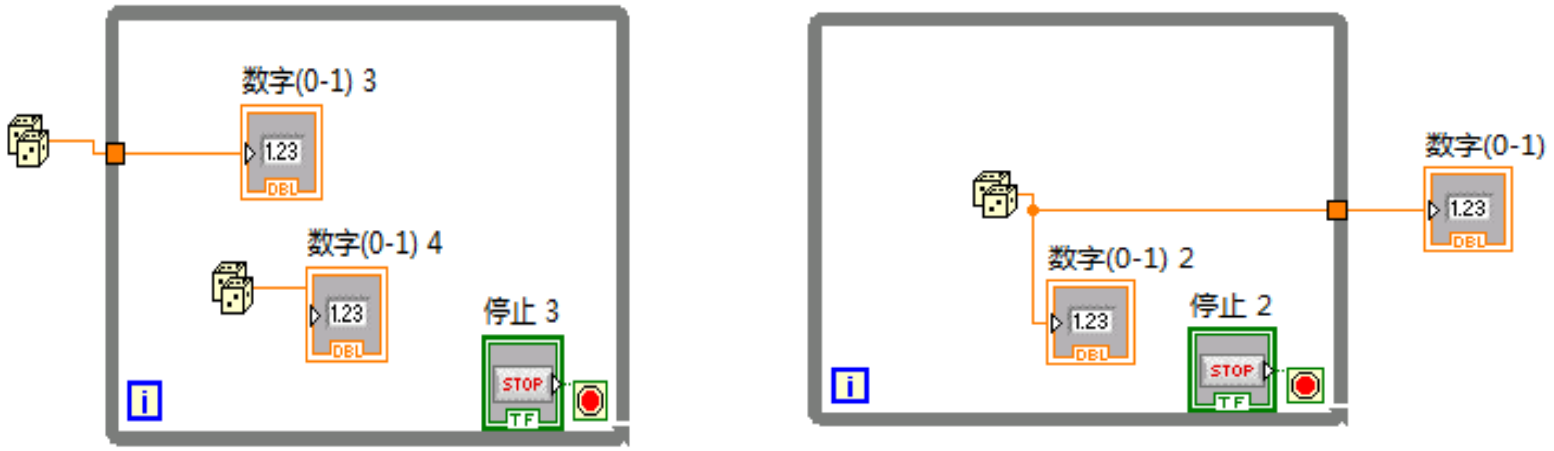
循环条件

（为真继续执行）

While循环迭代计数总是从零开始。 While循环将至少执行一次。

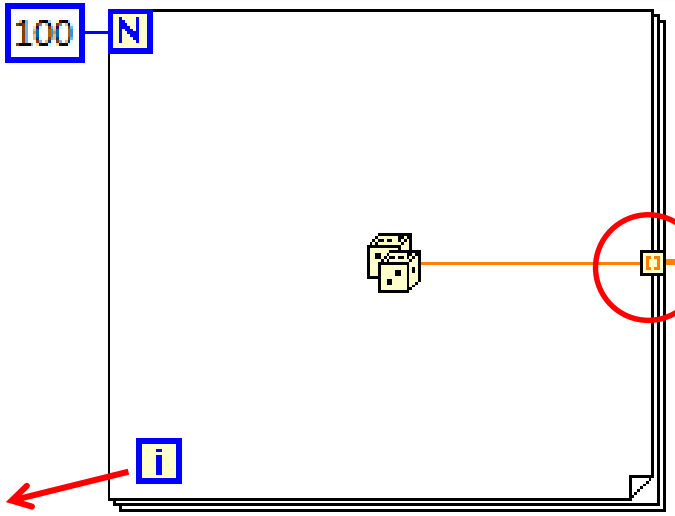
与C语言中的“do…while”循环类似

循环结构的数据传输——隧道



- 循环外的随机数产生后通过一个橙色实心方块（*叫做隧道*）进入循环内，接着才开始执行循环。
- **这里需要着重注意的是：LabVIEW的循环在开始循环前就送入外部数据，开始循环后就不再接受外部数据，所以随机数一直不变**
- **同样的道理，开始循环后是不送出数据到外部的，只有直到循环结束，送出的数据是最后一次循环的数据。**

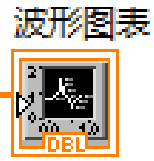
循环结构——for循环

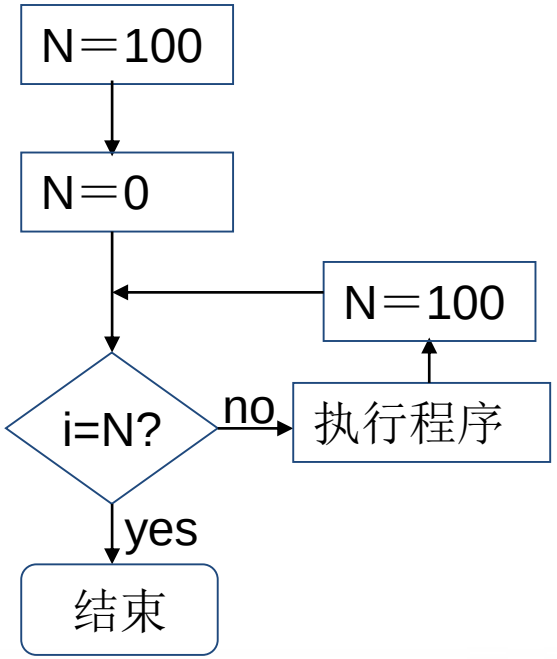


循环总数

循环计数

自动索引隧道





与C语言中的for循环类似

```
for(i=0;i<N;i++){ ;}
```

循环结构的自动索引隧道

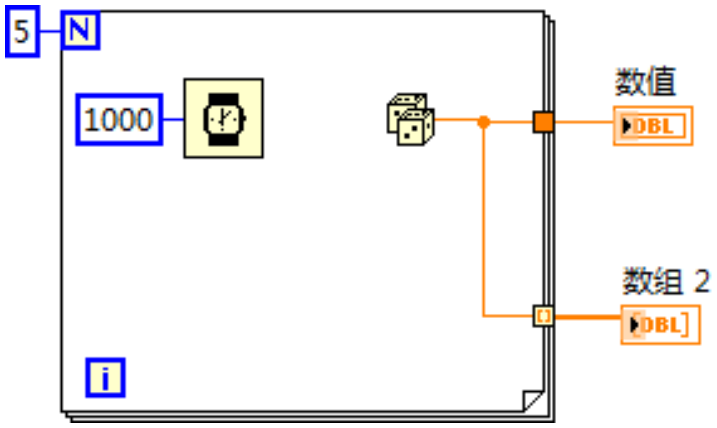


图1　程序框图

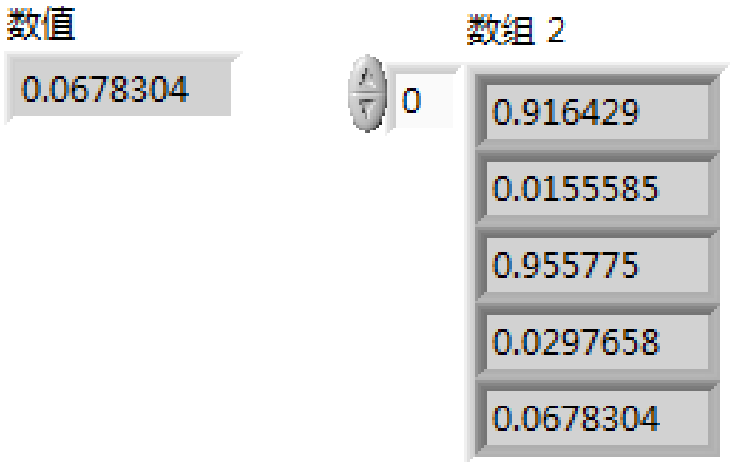


图2　前面板

自动索引隧道是把每次循环的输出数据都记下来，最后循环结束时一次性送出去，所以接收端是一个一维数组。

思考

1、如何用循环实现

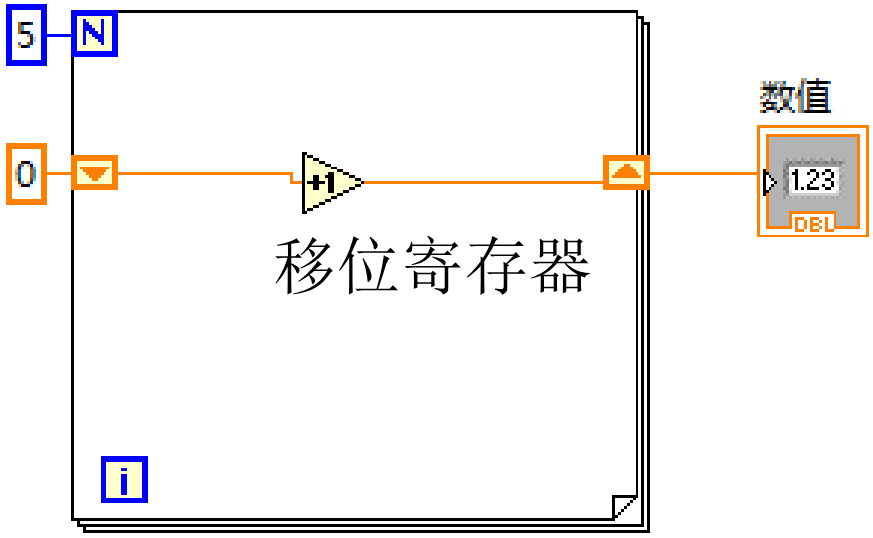
1+1+1+．．．+1=？

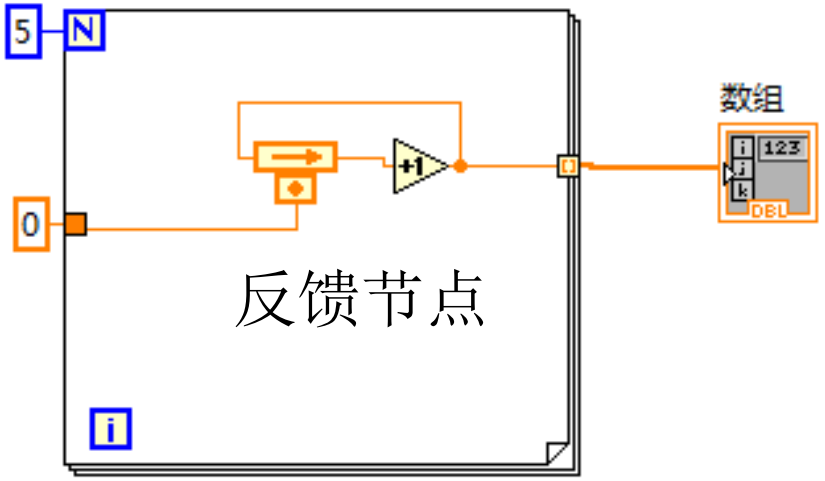
100个1相加

2、 1+2+3+．．．+100=？

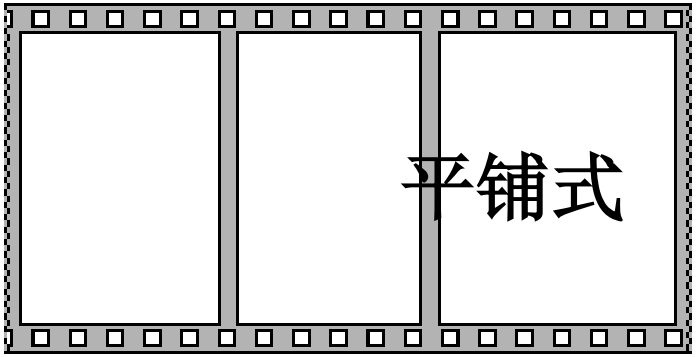
循环结构——移位寄存器和反馈节点

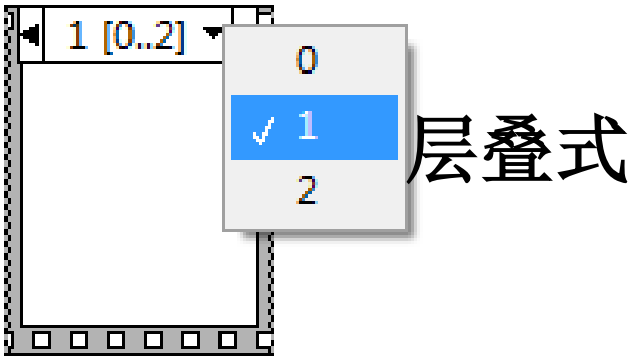
移位寄存器可用于将上一次循环的值传递至下一次循环。右侧接线端含有一个向上的箭头，用于存储每次循环结束时的数据。





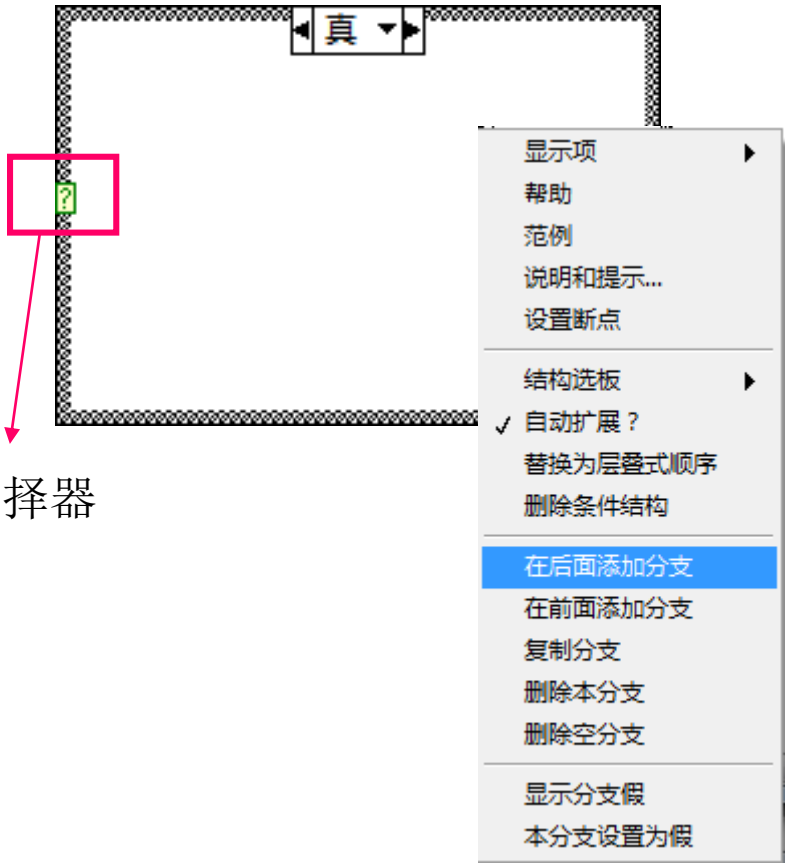
顺序结构





顺序结构是一系列顺序执行的有序帧集合。顺序结构顺序执行帧 0，然后是帧 1、帧 2，直到最后一个帧 。只有最后一个帧 执行完毕，数据才会离开结构。

条件结构

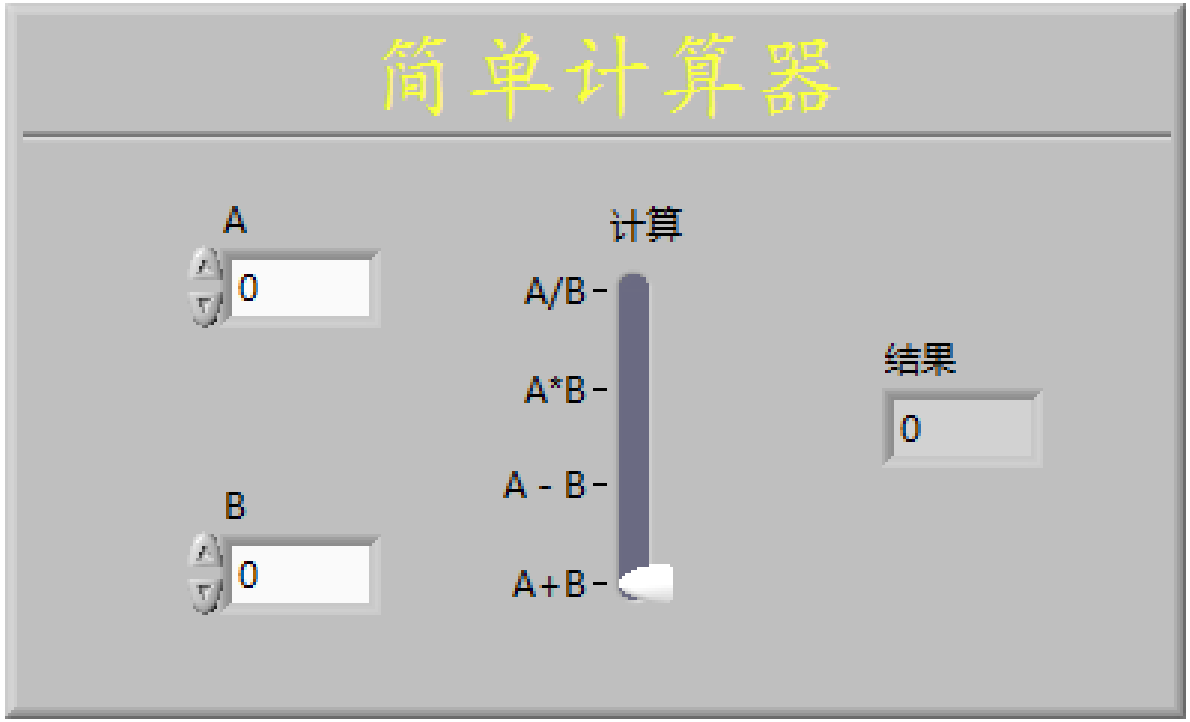


条件结构是一种由输入的条件进行选择执行分支的结构。

根据输入的类型可实现类似于C语言中的if-else语句或switch-case语句

练习：设计一个计算器，可以实现加减乘除运算

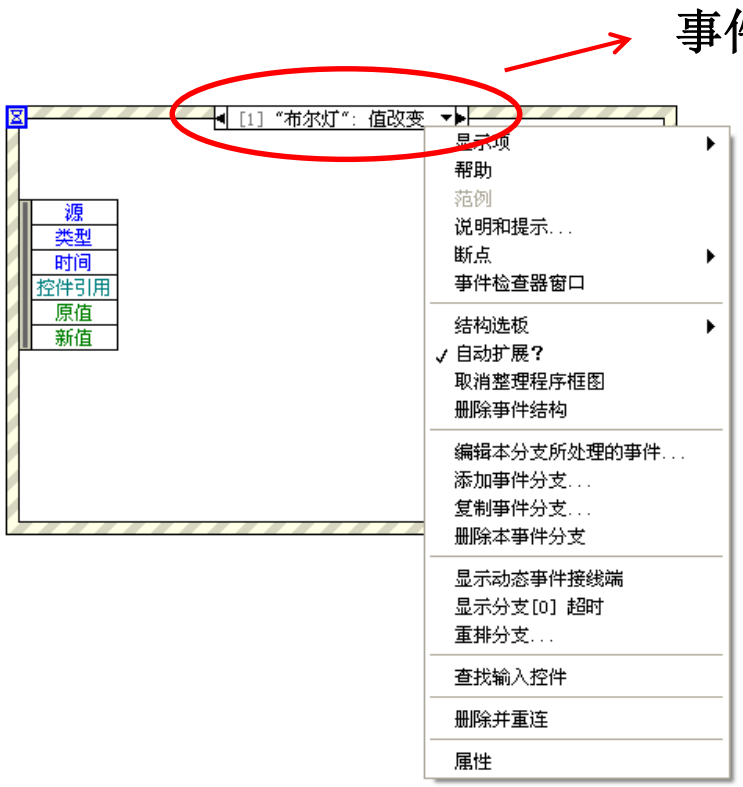
提示：条件选择输入可以用滑动杆



练习：设计一个程序，实现对红绿灯的控制

提示：条件选择输入可以用枚举

事件结构

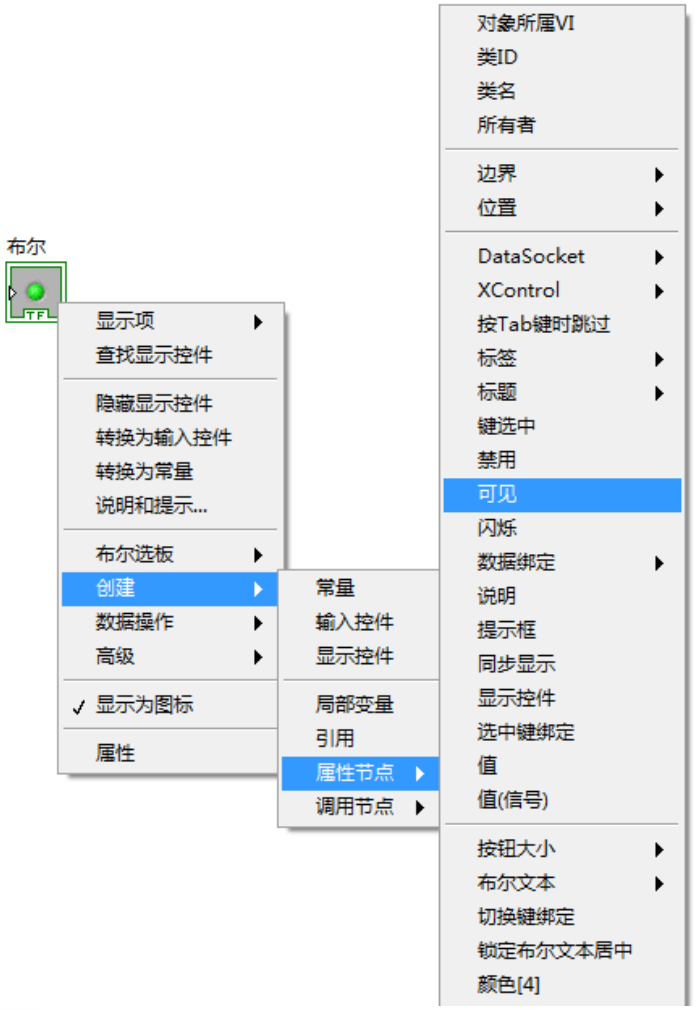


事件结构，当“事件”发生时，程序作出相应的相应。类似于C语言中的事件和事件处理器

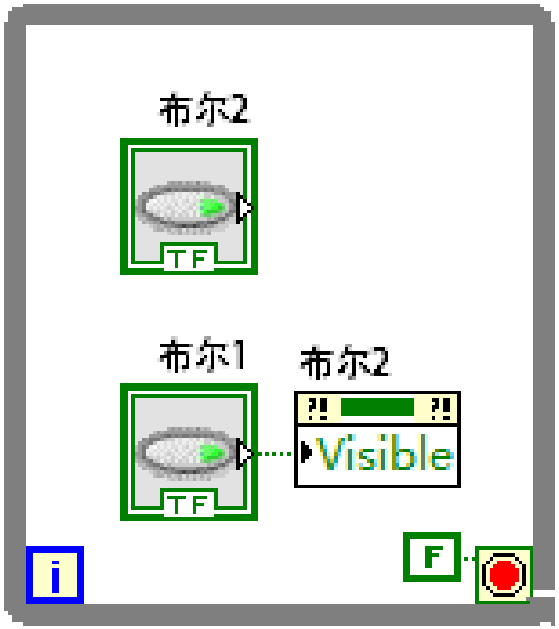
属性节点 & 局部变量

属性节点 ——

允许用户编程控制前板对象的属性：如颜色、显示、位置、数字显示格式等。



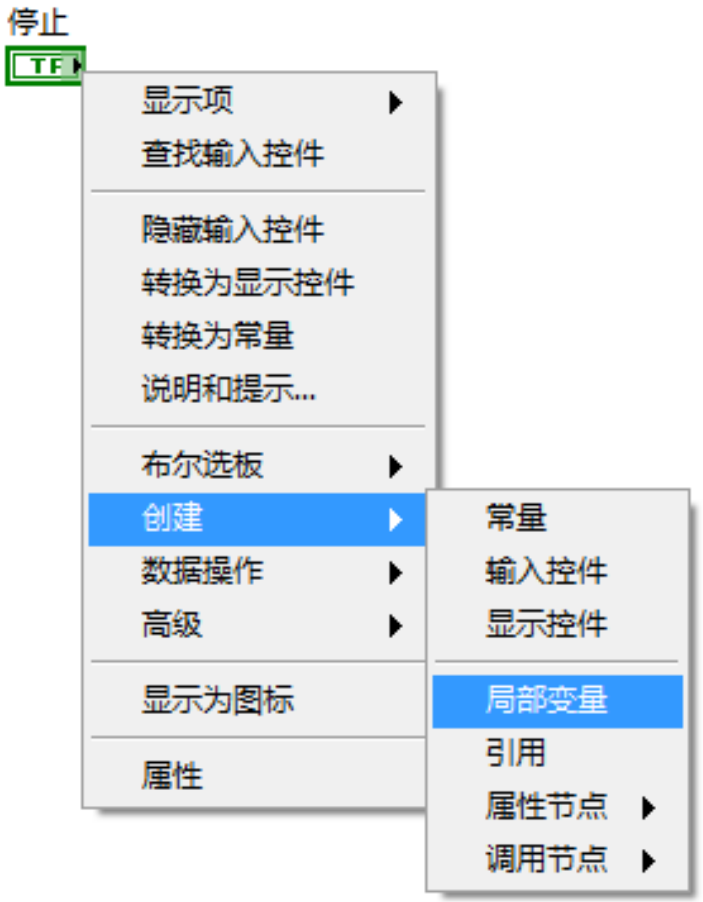
例：布尔灯1被按下时，布尔灯2才显示



属性节点 & 局部变量

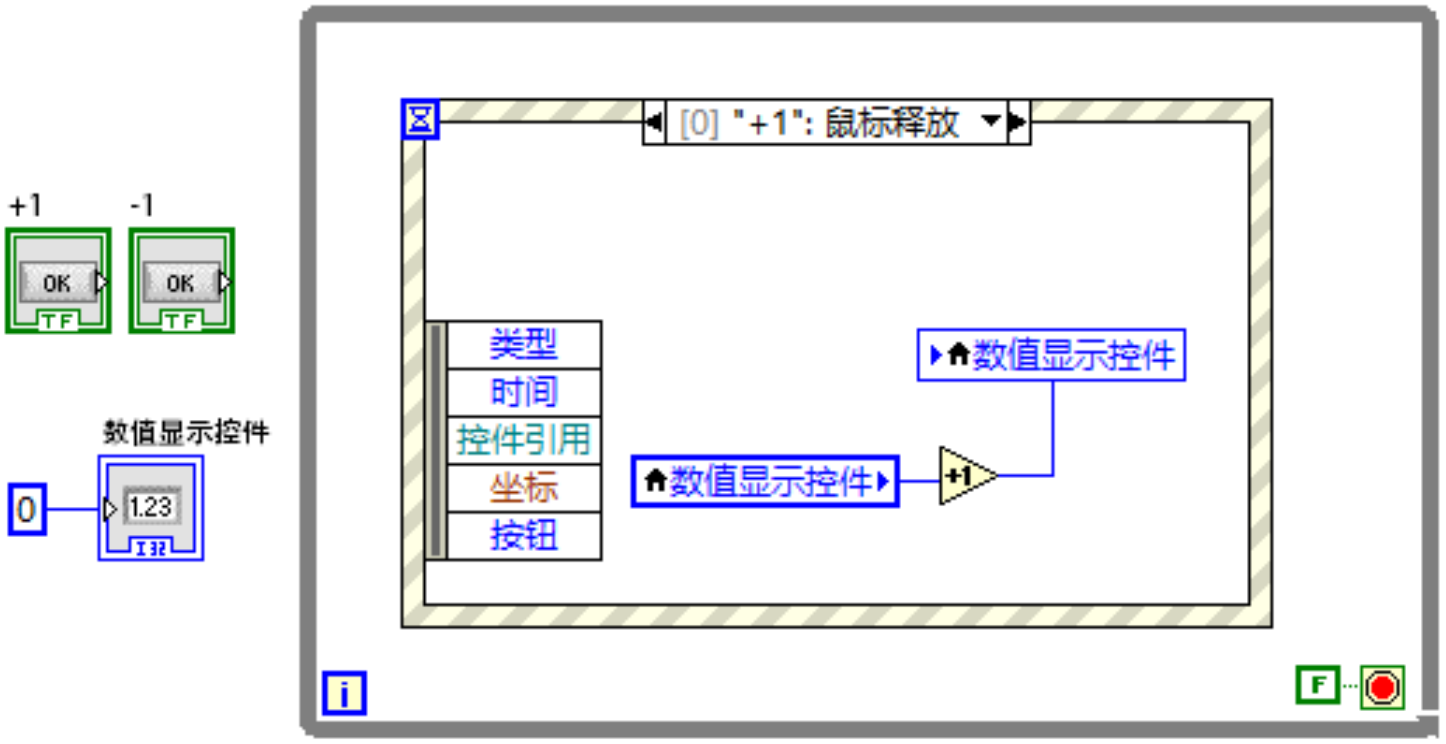
局部变量——

出现不便于连线的情况：我们创建的控件的克隆品，两者中的数据是等价的。



注：LabVIEW中不建议滥用局部变量，因为不符合数据流的思想，滥用可能给程序带来混乱。

例：


+1
-1
OK
TF
OK
TF
数值显示控件
0
1.23
I32
[0] "+1": 鼠标释放
类型
时间
控件引用
坐标
按钮
数值显示控件
数值显示控件
+1
i
F

设计一个简单的数据处理程序

- 程序内容：

A、一维数值型数组作为数据输入；

B、功能（两个按钮）：（1）取平均值;（2）取标准差；

C、数值显示控件显示结果；

提示：平均值与标准差函数在【函数→数学→概率与统计】中可找到

常用图形显示控件

➢**波形图表**可以将采集数据逐个地、实时地显示出来，连接成实时变化的曲线或图形。常用来观察采集数据的变化趋势。（可接收标量、数组或簇类型数据）

➢**波形图**用来一次显示出已经生成的一组数据。常用来显示对采集到的数据处理后的结果。（可接收数组类型数据）

➢**XY图**专门用于表示曲线中纵坐标（Y）值随横坐标（X值）变化的规律。不能实时显示。（可接收数组或簇类型数据）

练习：李萨如图形

给大家提供的是半成品的程序——李萨如图形（练习）.vi
请大家依照给出的参数方程将它完善。
下面给出一些提示。

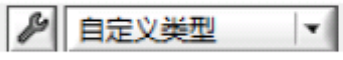
1. Ctrl+B　可删除所有错误连线。
2. 相同的控件可以直接Ctrl+C复制，然后鼠标左击放置位置，Ctrl+V粘贴。
3. 相同类型的控件可以在某控件上右键弹出选板，如图1-4
4. 在控件接线端右键可以快速创建控件。
5. 输入、显示、常量可鼠标右键相互转化。
6. 菜单栏/显示即时帮助，鼠标移动到控件上可以查看控件用法。

拓展

自定义控件

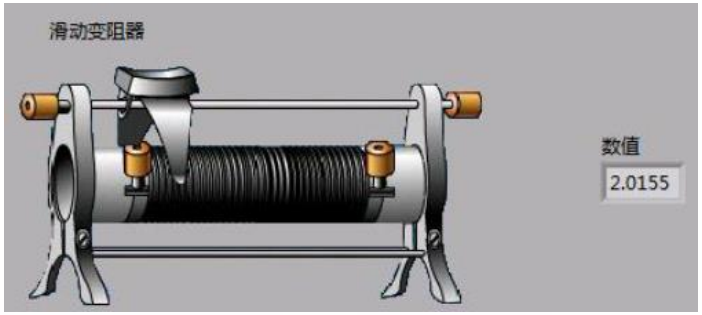
LabVIEW 中我们常用的两大类图标是控件和子 vi。控件诸如布尔灯、字符串等输入显示控件。它的文件后缀是“.ctl”。为了提高交互性，LabVIEW 允许自定义控件。

建立一个布尔灯，右键—“高级”—“自定义”。

进入界面后设置 

。“编辑”—“导入图片到剪切板”。对布尔灯右键，“从剪切板导入图片”—“假”。重复“编辑”—“导入图片到剪切板”导入另一张图片。对布尔灯右键，“从剪切板导入图片”—“真”。效果如下

你可以建立自己喜欢的控件，控件对于程序框图来说是一样的，但是在前面板上给人的感觉却截然不同，比如如下的变阻器。



程序编写、调试技巧

- 对较大的程序分模块编写成独函数，并且一步步调试，这样逻辑浅析，对运行结果实时探测。
- 尽量少用全局部量和局部变量，这样编写的程序执行时占用内存少。
- 对一些多次要用的功能编写各独立的VI，方便以后调用，不需要重复编程。
- 建立建全的错误处理机制，当有错误时能正确、迅速地处理。

课后任务

- 复习该节课内容，练习LabVIEW编程；
- 完成李萨如图形练习；
- 阅读设计性实验题目选择，定下题目，准备第三次课的开题；