**实验报告**

专业：

姓名：

学号：

日期：

地点：

课程名称： 通信原理实验 指导老师： 金向东、龚淑君 成绩：

实验名称： 熟悉Labview 实验类型： 综合型实验 同组学生姓名：

一、实验目的和要求

1. 熟悉LabVIEW通信设计套件
2. 会使用While/For循环处理数据
3. 熟悉多种数据类型
4. 会使用数组，访问和处理元素

二、实验设备

安装LabVIEW环境的电脑1台

三、实验概要

本实验课程的目的是在通用软件无线电外设（USRP）上应用、实践关于软件无线电的数字通信理论。USRP结合了LabVIEW软件，该软件是一个使用模块和连线的图形编程软件。因此有必要熟悉LabVIEW语言并能写一些代码。

四、实验内容和步骤

**4.1 建立项目和熟悉界面**

1)运行电脑中的软件：NI LabVIEW NXG 2.1；

2)新建一个项目:点击文件/新建/项目；

3)在项目中添加一个新的VI：点击文件/新建/VI；

4)将项目重命名为Lab1：在窗口左上角右击未命名项目并点击重命名；

5)每一个VI有两个窗口：前面板和程序框图。前面板是用户和VI进行交互的地方，它包括很多控制输入和显示输出参数端口。程序框图窗口包含从控制输入端接收输入数据，通过连接模块进行处理，并且通过显示输出进行显示的代码（流程图）。

**4.2 while循环**

While循环是一个控制流语句，用于重复执行一个函数，直到满足给定的条件终端。条件终端定义循环结束的时间。条件终端有两种设置：如果为True则继续和如果为True则停止。如果设置为True继续，则仅当布尔控件返回True时才进行循环。如果设置为True则停止，布尔值为True时，循环将中止。默认情况下，条件终端是“True则停止”。

例程1：使用while循环，产生一组0到10之间的随机整数，当产生的数与用户给定的数值匹配是，停止产生随机数。

1. 新建一个VI
2. 在程序框图窗口放置一个while循环（左侧程序流模块栏里选取）
3. 放置随机数模块产生随机数（数据类型/数值/随机数），产生0到1之间的随机浮点数。
4. 为了产生0到10之间的随机数，放置乘法模块和数值常量输入控件以及最近数取整模块（数据类型/数值）。
5. 创建一个数值显示输出端口，显示每一次循环产生的随机数：放置一个数值接线端（数据类型/数值），这个接线端默认的行为特性是“控制”（从前面板输入），需要改变其特性为“显示”（输出到前面板），重命名显示端口为Random Integer。
6. 创建一个数值输入控制端口（数值接线端），接收匹配的整数。重命名控制端为Number to match。
7. 使用“等于？”模块（数学/比较），比较两个数值的大小。
8. 在前面板窗口创建一个停止按钮（按钮/停止按钮），回到程序框图窗口，在未放置项将停止控件拖放在框图中。
9. 放置“或”模块（数据类型/布尔），将布尔控制两个输入分别与比较器的输出、停止按钮连接。当输出随机数与输入匹配或者按下停止按钮，循环将中止。
10. 回到前面板，将未放置项拖放进窗口界面。
11. 在前面板输入一个匹配数值，运行程序，观察结果。
12. 在程序框图窗口上方菜单条中，点击“高亮显示执行结果”，运行程序的时候，可以观察到程序框图中数据流的变化。

**4.3数据类型（簇）**

在LabVIEW中，每一个对象和连线都跟数据类型有关系。LabVIEW支持很多数据类型，它们由不同的颜色、形状区分。部分数据类型如图1所示：



图1 部分数据类型

例程2：接收一串字符串，将其反转；对给定的数组元素数值增加偏置量。

1. 新建一个VI
2. 在前面板，创建一个空白簇。
3. 在空白簇中放置一个字符串输入控件、一个数值输入控件和一个数值数组输入，将数组尺寸改为5（可通过鼠标拖拉数组外框或直接在属性里修改）。在簇外面再放置一个字符串输入控件和一个数值数组输入，将它们改为显示模式。
4. 在程序框图窗口将未放置项放置在框图界面里，在程序框图中放置簇属性模块（数据类型/簇），以便从簇中读取各个数值。将簇控件与簇属性模块相连，鼠标往下拖拉簇属性模块外框显示更多的属性端口。
5. 使用反转字符串函数（数据类型/字符串）反转字符串，使用“加”函数（数学/数值）将偏置与数组相加。
6. 在前面板输入控制值，运行程序，通过显示输出确认结果是否正确。
7. 将改变的值再结合放入簇：复制、黏贴前面板中的簇并将它的属性改为显示。
8. 将新的簇（未放置项）放置在程序框图中，为了将改变的字符串、数组和新的簇结合在一起，需要再次用到簇属性模块。因为要将数据写入簇，所以需要将簇属性设置为“全部为写入”。
9. 运行程序并比较结果。

**4.4 数组**

数值中的元素都是相同类型的数据，可以是数值、布尔、路径、字符串、波形和簇。当处理一批类似数据或作重复计算时，可以考虑使用数组。存储波形数据或循环中产生的数据时，使用数组是一个理想选择。

例程3：给定一个数值数组，找出其中的最大元素。

1. 新建一个空白VI
2. 在前面板添加一个数值数组输入，通过往下拖拉数组外框将元素数量增加到5。将数值数组输入重命名为“Array”
3. 添加一个数值输入控件，将它改为显示模式并重命名为“Maximum value”。
4. 在前面板添加一个For循环，将“Array”终端放在循环外面，“Maximum value”终端放在循环里面。
5. 将数值终端与For循环的左边框相连，在边框上会产生一个“隧道”，隧道使得循环与外面的模块能够进行数据通信。隧道有索引使能功能，当它设置为自动索引使能时，输入循环结构的数组元素在每一次循环的时候送入一个元素。如果索引没有使能，循环结构将一次调用数组中的全部元素。
6. 为了在For循环中对输入数组的元素逐个跟设定的最大值进行比较，需要在For循环中添加一个“大于？”函数（数学/比较）。再放入一个“选择”模块（程序流）。
7. 假如当前输入循环的数组元素比最大值大，需要更新最大值。将隧道输出与“大于？”模块的X端相连，同时连接到“选择”模块的真值输入端。将“大于？”模块的输出与“选择”模块的选择端连接。
8. 我们需要存储当前的最大值以备下次循环迭代的时候使用。移位寄存器是将两次相邻迭代联系起来的数据传输通道。将“选择”模块的输出连接到循环的右侧边框线上，会默认建立一个“隧道”，点击鼠标右键，将它改为移位寄存器模式，这时鼠标光标也变成移位寄存器的图标，点击循环的左边框，完成移位寄存器的放置。（移位寄存器的左侧方框可以看作是来自上一次迭代结果的输入，右侧方框是本次迭代的输出）
9. 移位寄存器的左侧方框是上一次迭代产生的最大值，用来与当前输入的数组元素进行比较。将移位寄存器的左侧方框与“大于？”模块的Y端以及“选择”模块的假值连接。
10. 将“选择”模块的输出与Maximum value显示控件相连。
11. 在前面板给array输入数值，运行程序确认是否正确。

五、问题回答

**问题 1**

1) 完成以上示例，在实验报告中提供程序框图和前面板运行结果。

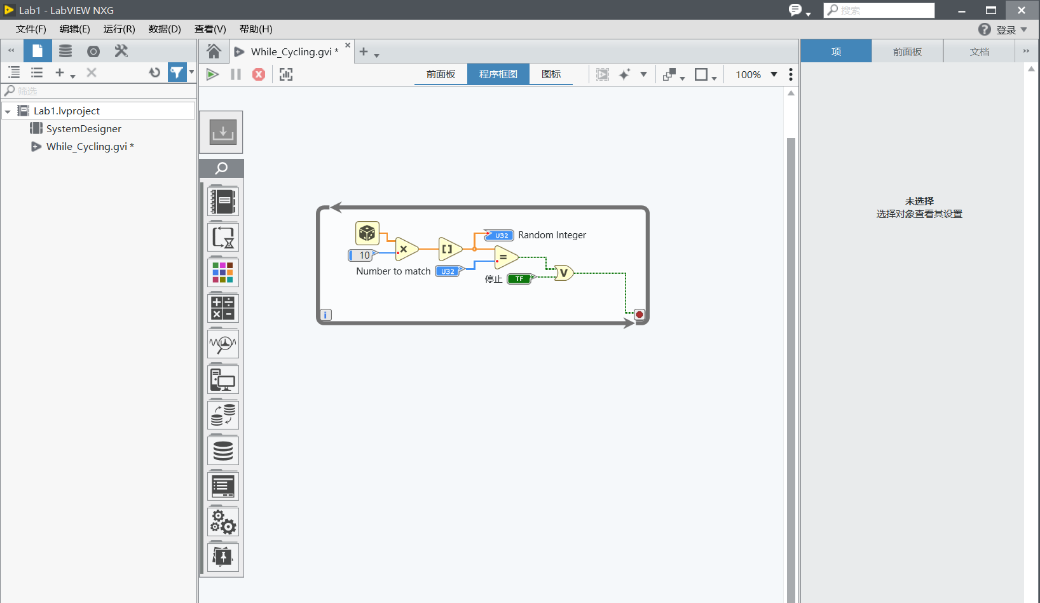
2) 给 while 循环增加一些延时（比如 1 秒钟），以便能观察到数据的变化。

3) 增加一个布尔指示器，当随机数大于 5 时打开，小于或等于 5 时关闭。

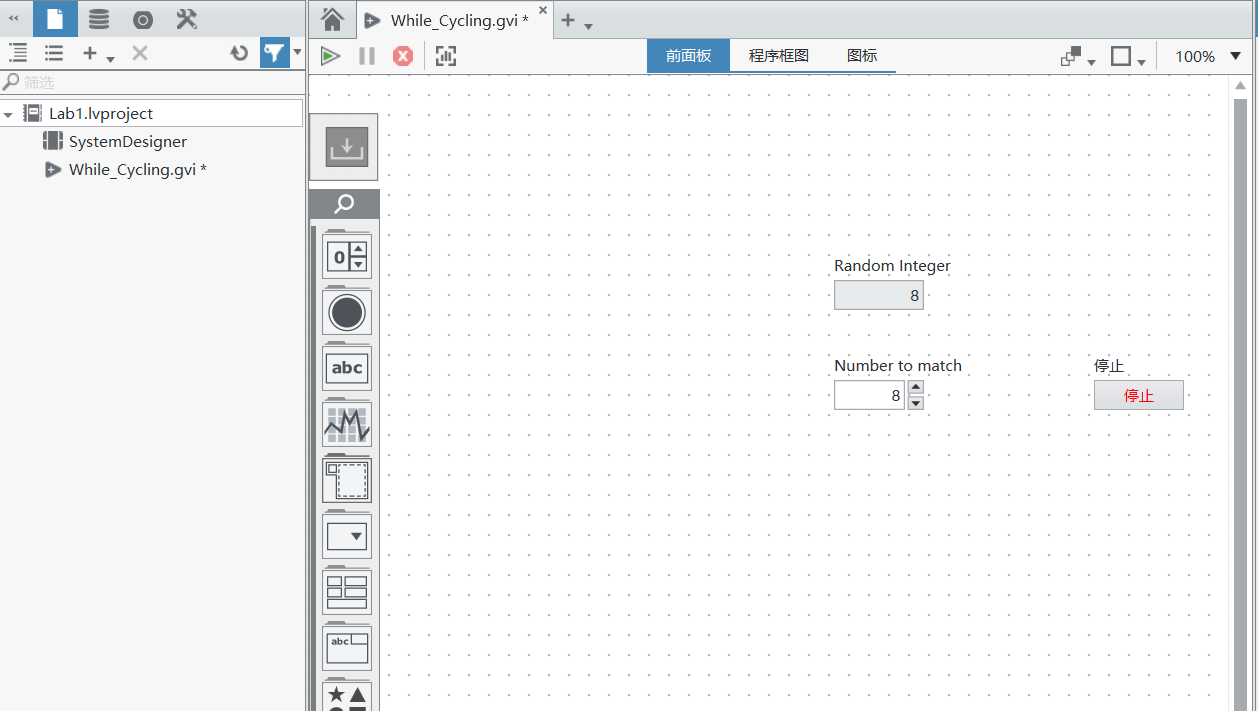
4) 增加一个输出端显示循环的次数

1) 完成以上示例，在实验报告中提供程序框图和前面板运行结果。

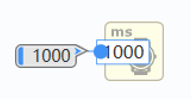
可以观察到运行时前面板数据刷新很快。



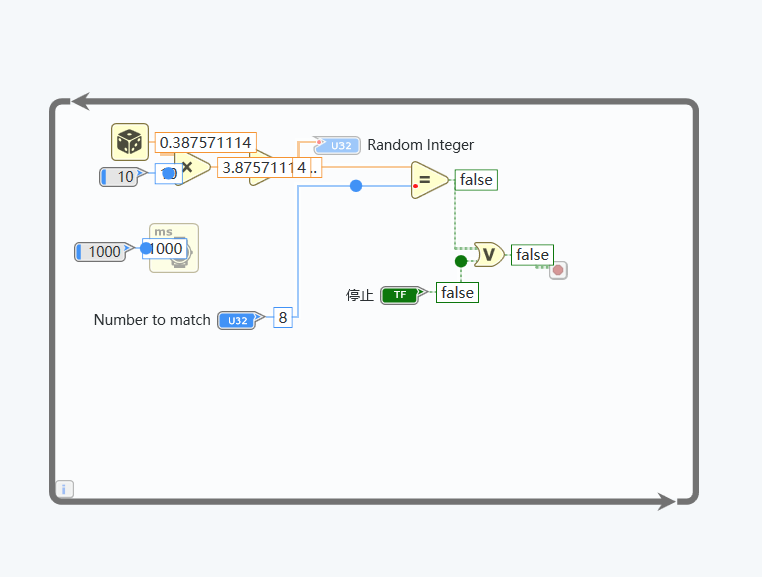
程序框图



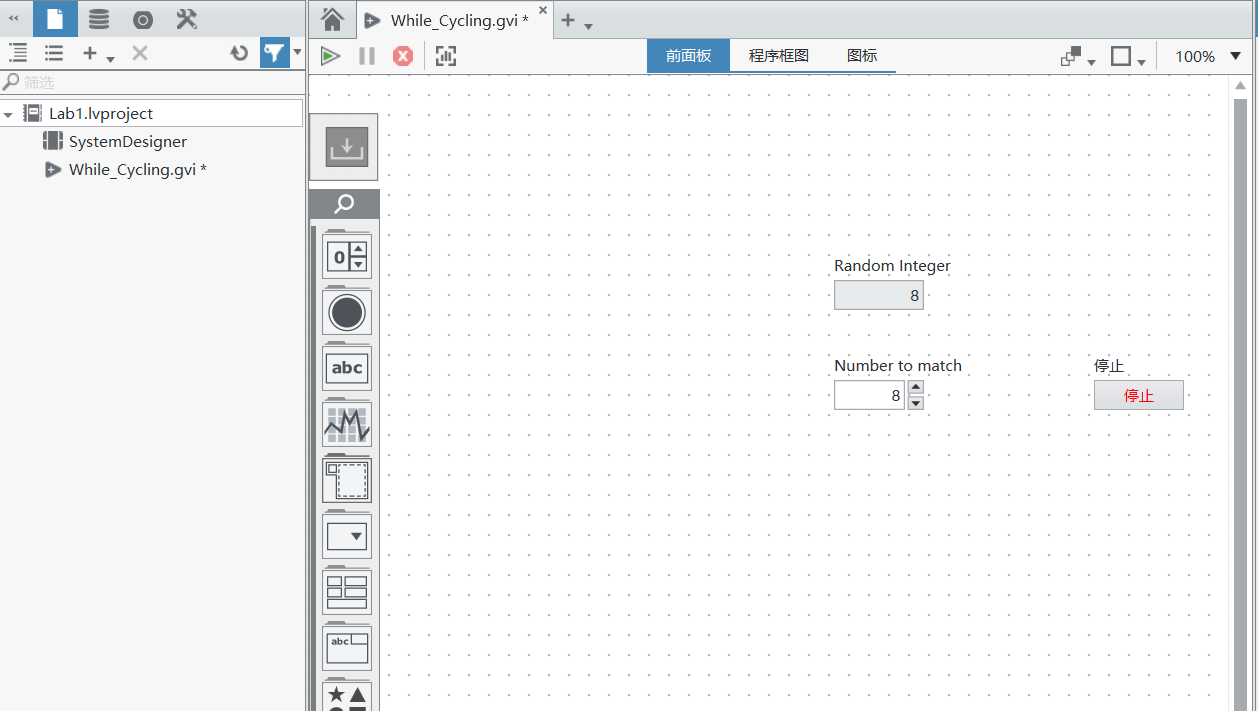
前面板图

2) 给 while 循环增加一些延时（比如 1 秒钟），以便能观察到数据的变化。

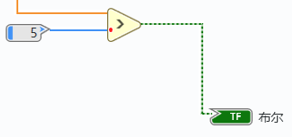
增加延时后，前面板数据刷新明显变慢，易于观察。

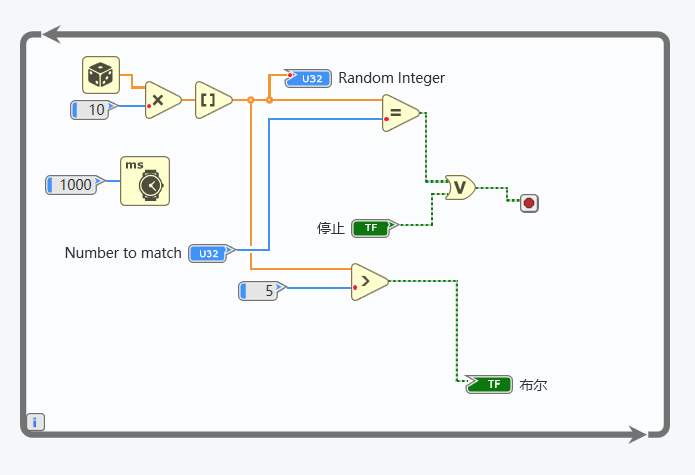


程序框图

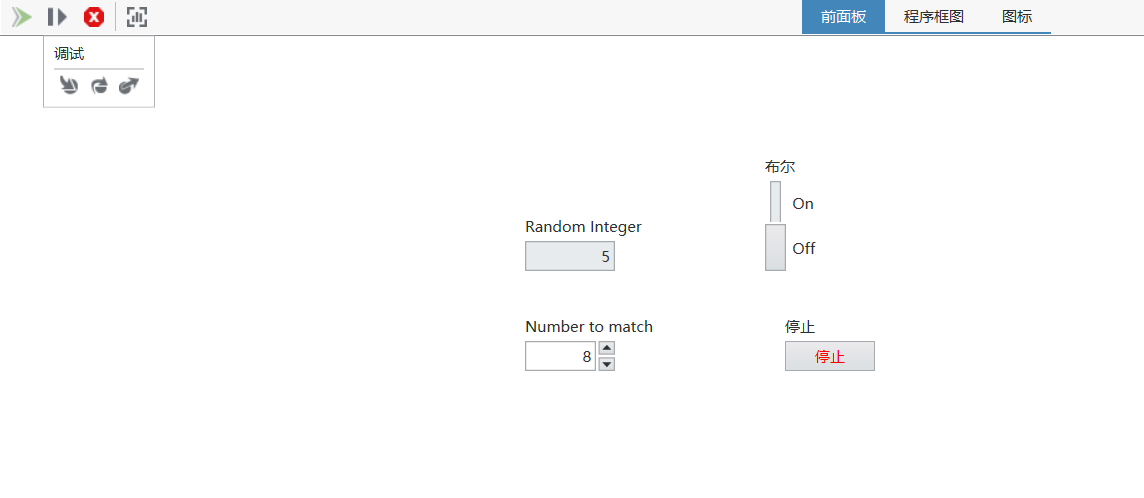


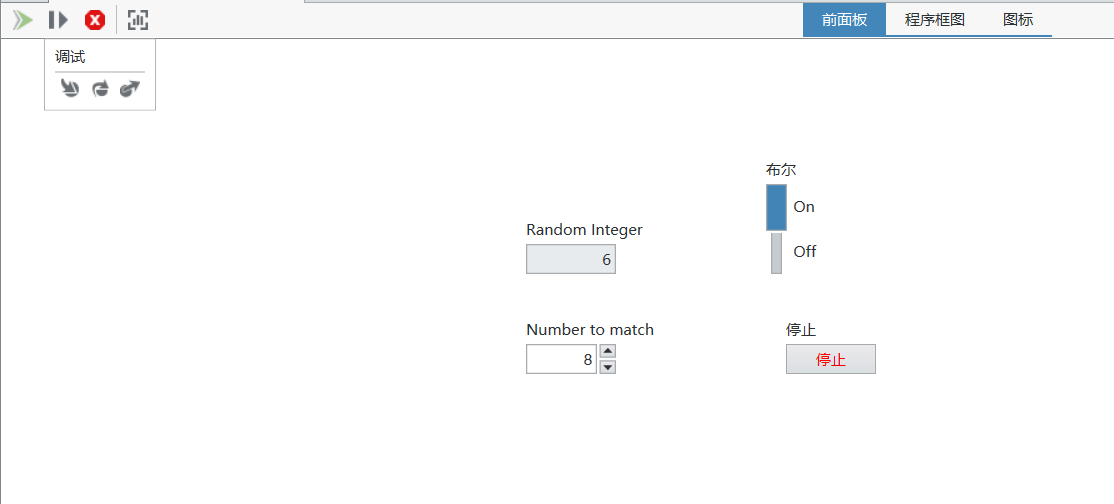
前面板图

3) 增加一个布尔指示器，当随机数大于 5 时打开，小于或等于 5 时关闭。



程序框图

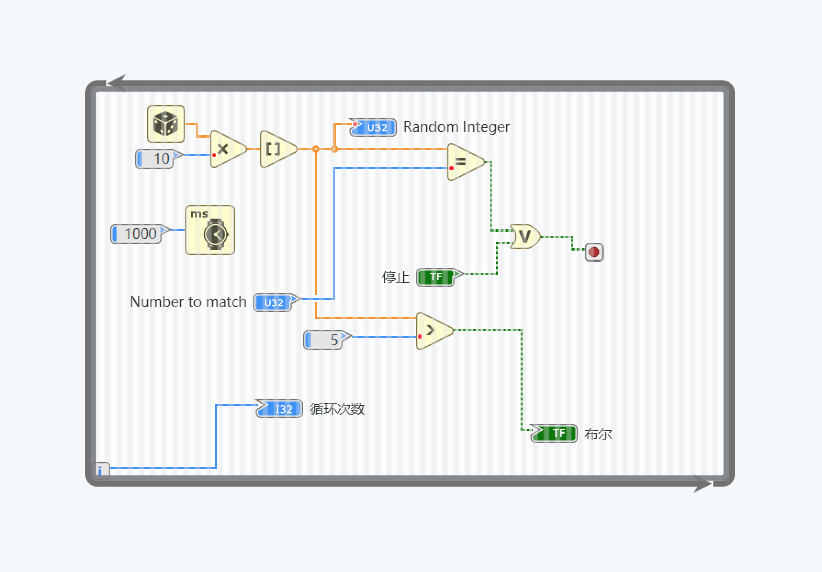




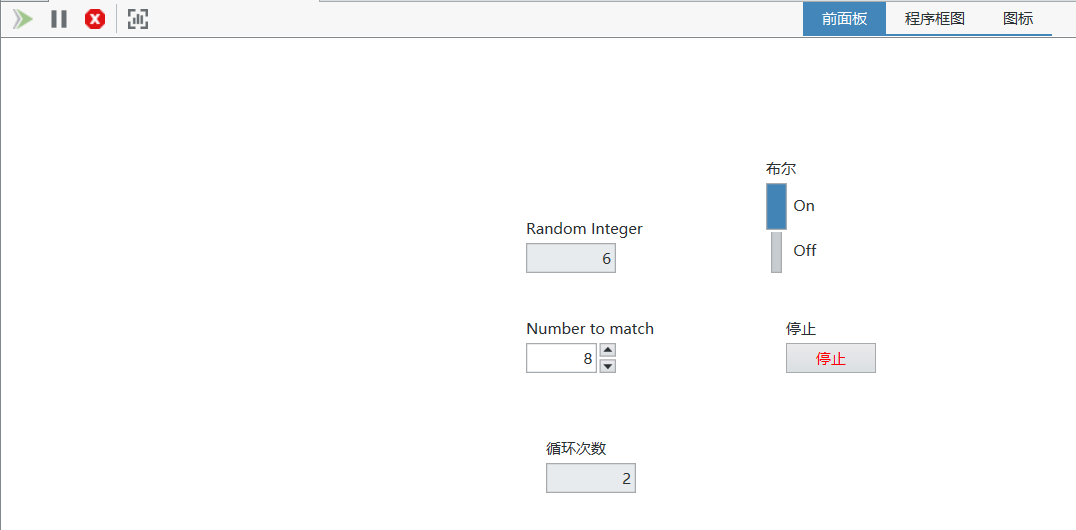
前面板图

4) 增加一个输出端显示循环的次数

While 模块本身就具备记录循环次数的功能，在左下角“i”。只要增加一个显示模块，将两者连接即可。



程序框图



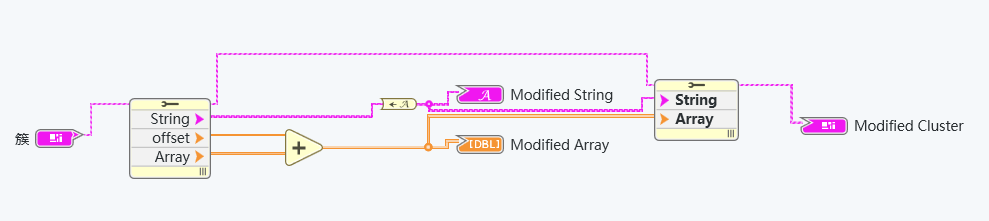
前面板图

**问题 2**

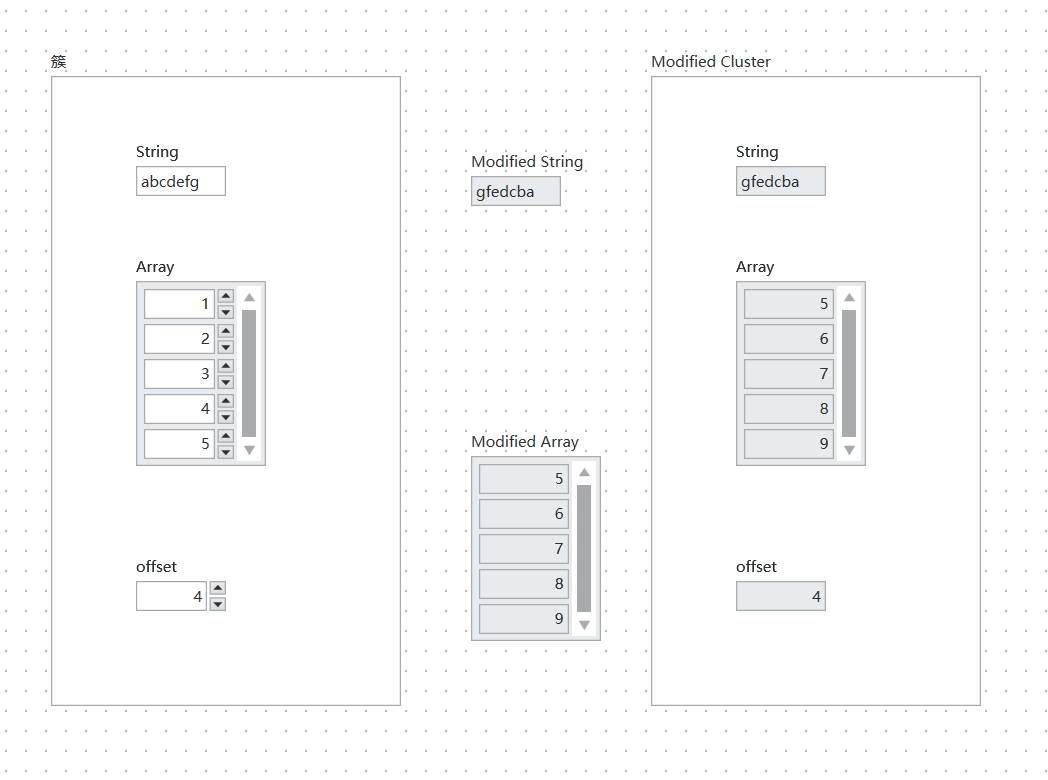
1) 完成上面的示例，在实验报告中提供程序框图和前面板运行结果。

2) 是否能将一个整型控制输入与双精度类型显示输出相连？整型输入能否与字符串输出相连？如果不可以，需要使用什么模块进行类型转换？

1) 完成上面的示例，在实验报告中提供程序框图和前面板运行结果。



程序框图

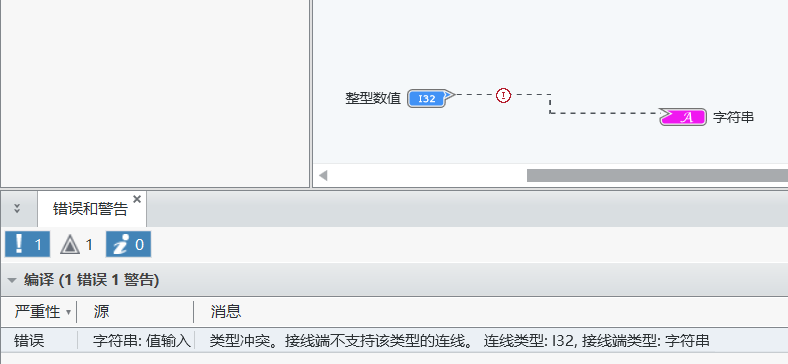


前面板图

2) 是否能将一个整型控制输入与双精度类型显示输出相连？整型输入能否与字符串输出相连？如果不可以，需要使用什么模块进行类型转换？

整型控制输入与双精度类型显示输出可以直接相连。

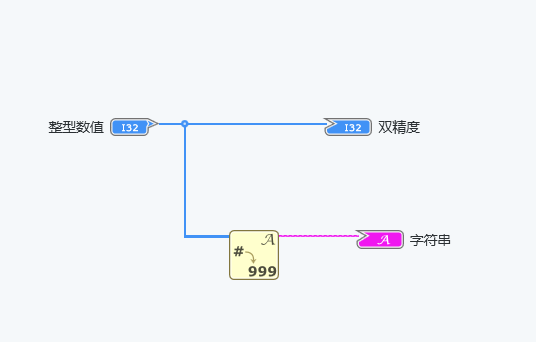
**整型输入与字符串输出不能直接相连**，否则有如下报错：



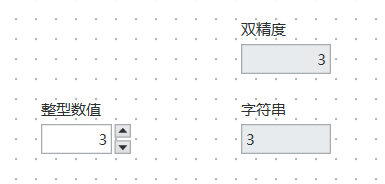
将整型输入转换为字符串输入的方法：添加“数值十进制数字符串”控件



测试结果如下：



程序框图



前面板图

**问题 3**

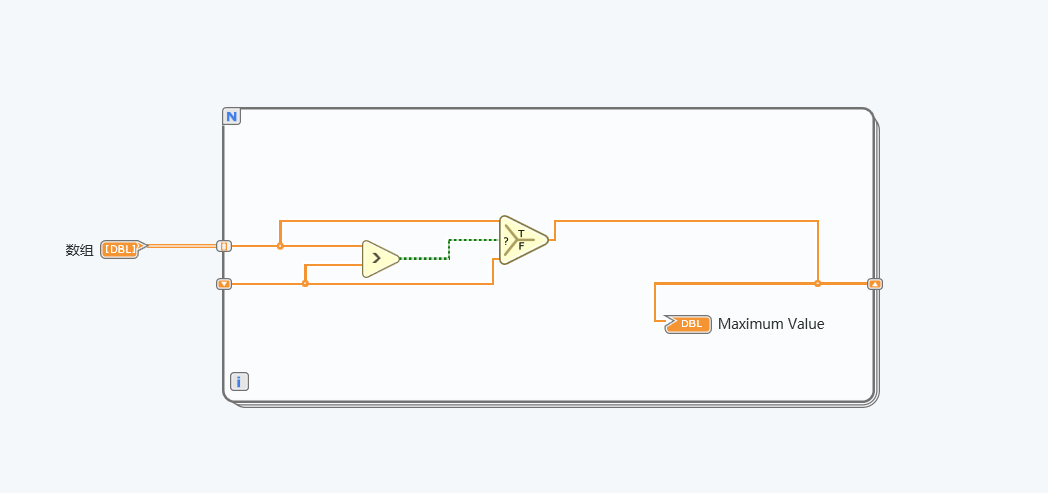
1) 完成上面的示例，在实验报告中提供程序框图和前面板运行结果。

2) 将数组元素最大值改为 0，再次运行程序，还能找到最大值吗？更改例程，使得每运行 一次程序都能找到最大值。

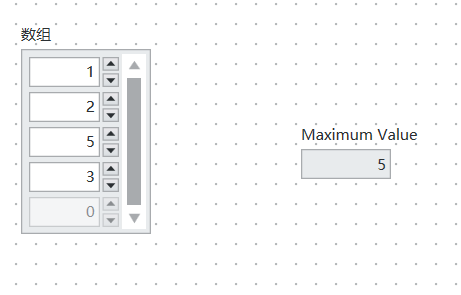
3) 在问题 2 的基础上，更改程序框图，能够得到最大值的索引值（在数组中的位置）。

4) 完成以下 VI：产生两个数值范围在-0.5 到 0.5 之间的随机数，比较这两个随机数值的大 小，如果第一个数比第二个数大，Team A 得分；如果第二个数比第一个数大，Team B 得分。当其中一个队得分达到 50 时，程序停止运行。显示最终的两个随机数、循环的 次数、Team A 和 Team B 的得分以及哪一队赢。

1) 完成上面的示例，在实验报告中提供程序框图和前面板运行结果。



程序框图



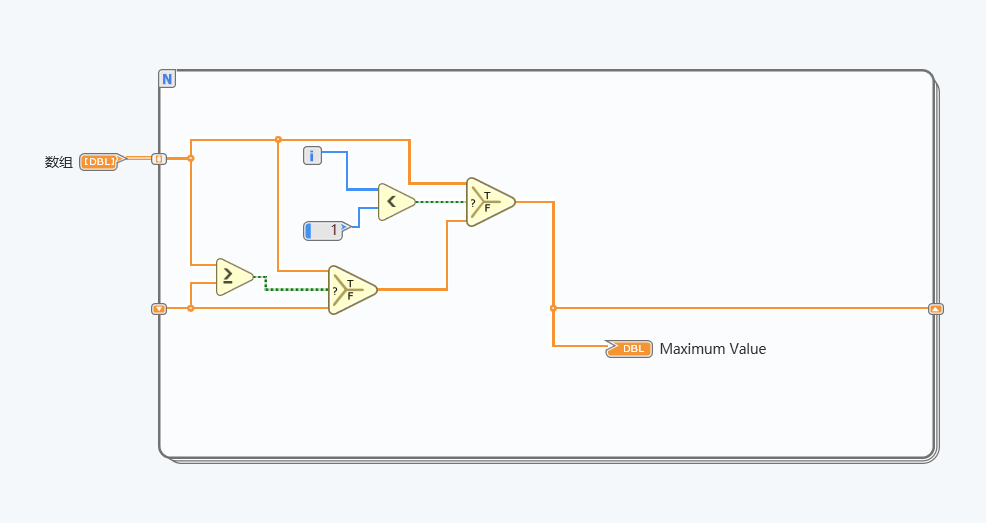
前面板图

2) 将数组元素最大值改为 0，再次运行程序，还能找到最大值吗？更改例程，使得每运行 一次程序都能找到最大值。

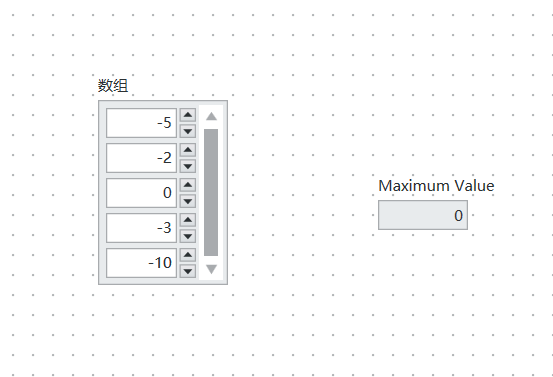
将数组元素最大值改为 0，再次运行程序，**不能找到正确的最大值**。这是移位寄存器没有赋初值导致的。

所以，我们通过检测循环轮数来修复这个bug。当i=0时，即第一轮，我们不进行比较，而是直接将第一个数写入移位寄存器。之后的轮次和之前保持一致。

更改后具体程序如下：



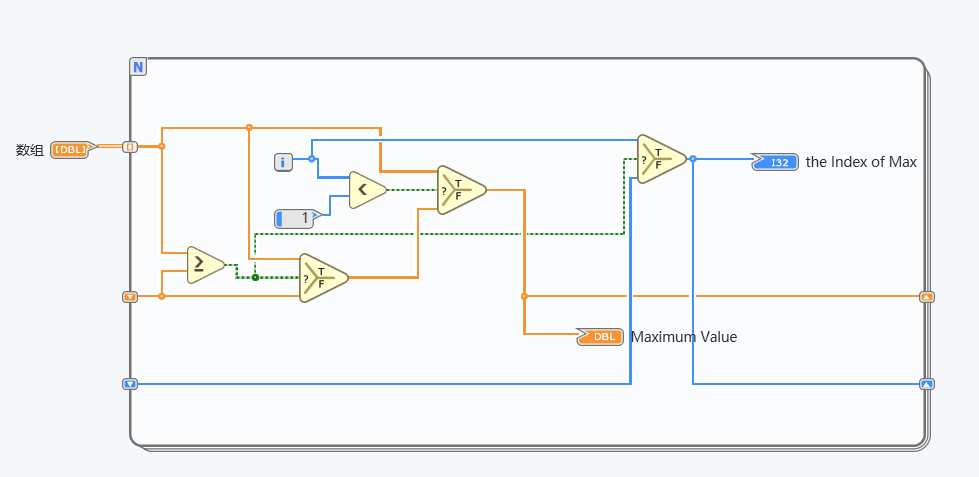
程序框图



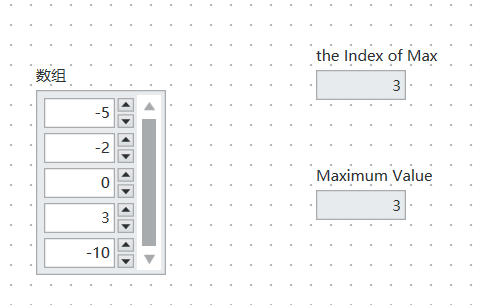
前面板图

3) 在问题 2 的基础上，更改程序框图，能够得到最大值的索引值（在数组中的位置）。

和问题 2 一致，第一轮时，我们直接为the Index of max和Maximum Value赋值。在后续的轮次中，如果出现更大的值，则对the Index of max和Maximum Value进行更新。



程序框图

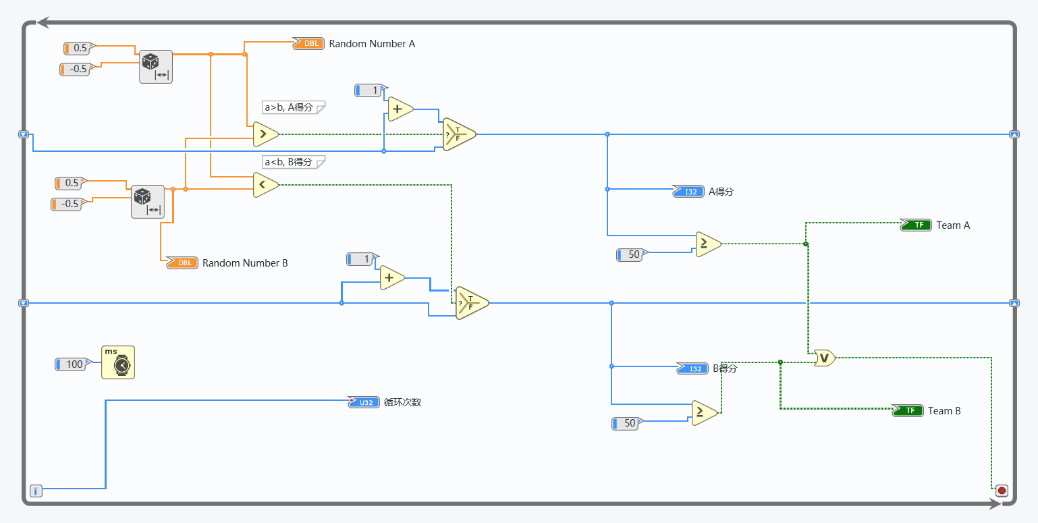


前面板图

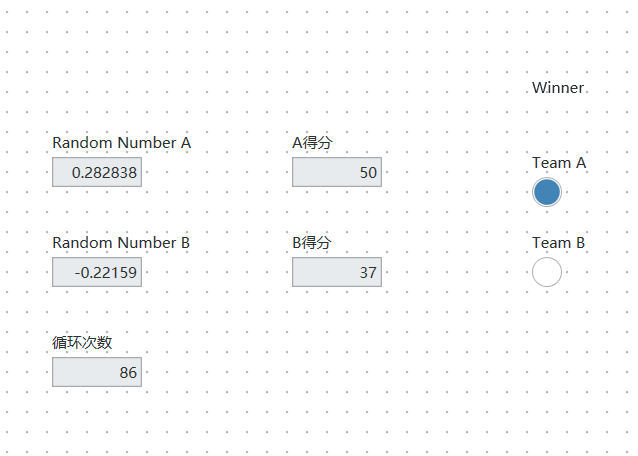
4) 完成以下 VI：产生两个数值范围在 -0.5 到 0.5 之间的随机数，比较这两个随机数值的大 小，如果第一个数比第二个数大，Team A 得分；如果第二个数比第一个数大，Team B 得分。当其中一个队得分达到 50 时，程序停止运行。显示最终的两个随机数、循环的次数、Team A 和 Team B 的得分以及哪一队赢。

程序如下：

首先生成两个 -0.5 到 0.5 之间的随机数 Random Number A和Random Number B。当A>B时，A队加分；A<B时，B队加分；A=B时，两队都不加分。当其中一个队得分达到 50 时，程序停止运行。



程序框图



前面板图