

НИУ ВШЭ
Факультет физики

Л.А. Моргун

Информатика для физиков

Введение в LabView

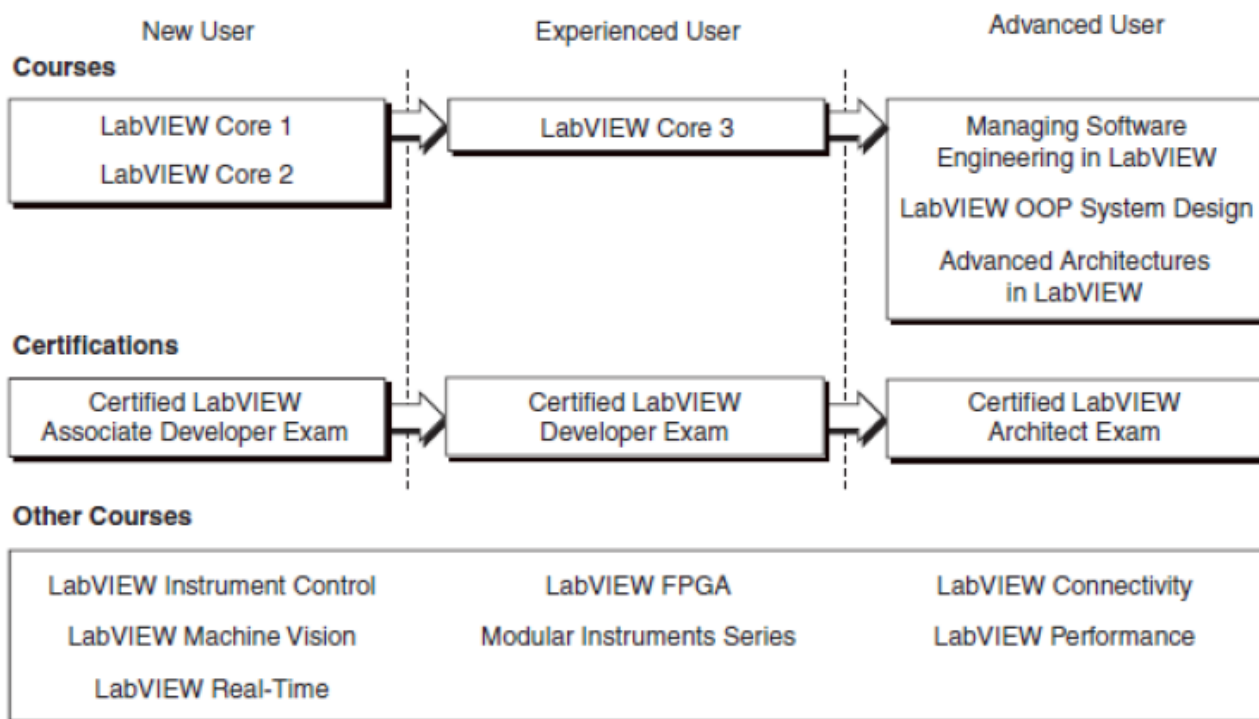
Москва, 2017

Оглавление

| | |
|---|----|
| Введение..... | 3 |
| Что же такое «LabVIEW»?..... | 3 |
| NI LabVIEW — история создания | 3 |
| Так что же такое LabVIEW? | 4 |
| LabVIEW – программа и возможности языка | 4 |
| Цель курса..... | 8 |
| План занятия..... | 9 |
| Виртуальные приборы (VI)..... | 9 |
| Состав VI | 9 |
| Окно лицевой панели | 9 |
| Окно блок-диаграммы..... | 9 |
| Начинаем проектировать VI | 10 |
| Project Explorer..... | 11 |
| Лицевая панель | 12 |
| Числовые органы управления и индикации..... | 13 |
| Булевские органы управления и индикации | 13 |
| Строковые органы управления и индикации | 13 |
| Панель инструментов окна лицевой панели | 13 |
| Блок-диаграмма | 14 |
| Автоматическое соединение объектов..... | 16 |
| Соединение объектов вручную | 16 |
| Панель инструментов блок-диаграммы | 16 |
| Выбор инструмента | 17 |
| Инструмент Operating | 17 |
| Инструмент Positioning | 17 |
| Инструмент Labeling | 18 |
| Инструмент Wiring..... | 19 |
| Потоковое программирование | 19 |
| Самопроверка (короткий тест) | 20 |
| Разработка простейшего VI | 20 |

Введение

Данные материалы основаны на руководстве по курсу LabVIEW Core 1 – сертификационных курсов по LabVIEW, которые проходят в специализированных центрах. По результатам прохождения этих курсов слушатели получают сертификаты.



Что же такое «LabVIEW»?

LabVIEW — это один из основных продуктов компании National Instruments. Прежде всего надо отметить, что LabVIEW — это аббревиатура, которая расшифровывается как **L**aboratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench. Уже в названии прослеживается ориентация на лабораторные исследования, измерения и сбор данных. Действительно, построить SCADA — систему в LabVIEW несколько проще чем при использовании «традиционных» средств разработки. В данной статье мне хотелось бы показать, что возможная область применения LabVIEW несколько шире. Это принципиально иной язык программирования, или если хотите целая «философия» программирования. Функциональный язык, заставляющий несколько иначе мыслить и порой предоставляющий совершенно фантастические возможности для разработчика. Является ли LabVIEW языком программирования вообще? Это спорный вопрос — здесь нет стандарта, как, например, ANSI C. В узких кругах разработчиков мы говорим, что пишем на языке «G». Формально такого языка не существует, но в этом и заключается прелесть этого средства разработки: от версии к версии в язык вводятся всё новые конструкции. Сложно представить, что в следующей реинкарнации Си появится, например, новая структура для for-цикла. А в LabVIEW такое вполне возможно.

Впрочем надо заметить, что LabVIEW входит в рейтинг языков программирования TIOBE, занимая на данный момент тридцатое место — где-то между Прологом и Фортраном.

NI LabVIEW — история создания

Компания National Instruments была создана в 1976 году тремя основателями — Джеффом Кодоски (Jeff Kodosky), Джеймсом Тручардом (James Truchard) и Биллом Новлиным (Bill Nowlin) в американском городе Остин (Austin), штат Техас. Основной специализацией компании являлись инструментальные средства для измерений и автоматизация производства.

Л.А. Моргун. Введение в LabVIEW

Первая версия LabVIEW увидела свет спустя десять лет после создания компании — в 1986 году (это была версия для Apple Mac). Инженеры NI решили бросить вызов «традиционным» языкам программирования и создали полностью графическую среду разработки. Основным идеологом графического подхода стал Джефф. Год за годом выпускались новые версии. Первой кроссплатформенной версией (включая Windows) была третья версия, выпущенная в 1993 году.

В Остине и по сегодняшний день располагается головной офис компании. Сегодня в компании работают почти четыре тысячи человек, а офисы находятся почти в сорока странах (есть также офис и в России).

Так что же такое LabVIEW?

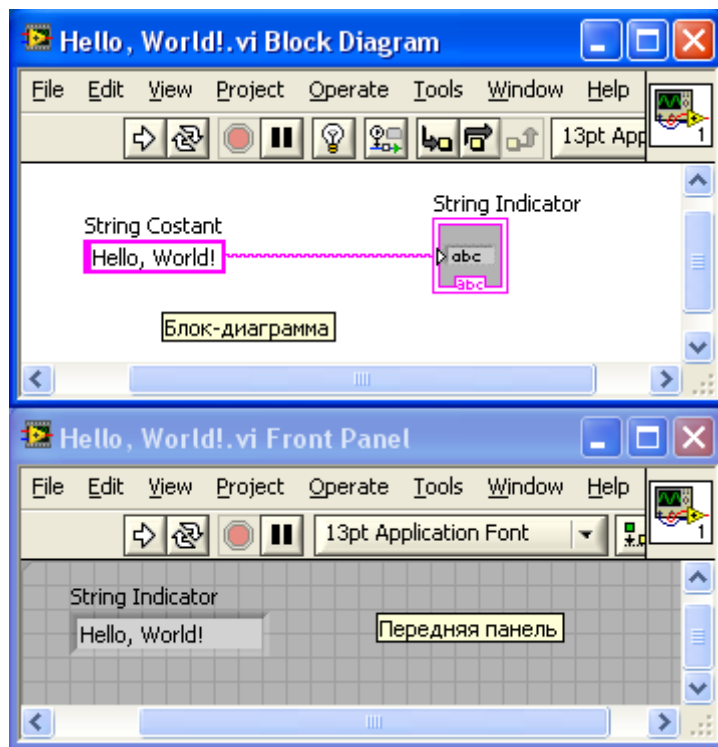
LabVIEW — это кроссплатформенная графическая среда разработки приложений. LabVIEW — в принципе универсальный язык программирования. И хотя этот продукт порой тесно связан с аппаратным обеспечением National Instruments, он тем не менее не связан с конкретной машиной. Существуют версии для Windows, Linux, MacOS. Исходные тексты переносимы, а программы будут выглядеть одинаково во всех системах. Код, сгенерированный LabVIEW также может быть также исполнен на Windows Mobile или PalmOS (справедливости ради надо отметить, что поддержка PalmOS прекращена, впрочем здесь сама Palm больше виновата). Этот язык может с успехом использоваться для создания больших систем, для обработки текстов, изображений и работы с базами данных.

LabVIEW — весьма высокоуровневый язык. Однако ничто не мешает включать «низкоуровневые» модули в LabVIEW-программы. Даже если вы хотите использовать ассемблерные вставки — это тоже возможно, надо лишь сгенерировать DLL и вставить вызовы в код. С другой стороны, высокоуровневый язык позволяет запросто производить весьма нетривиальные операции с данными, на которые в обычном языке могли уйти многие строки (если не десятки строк) кода. Впрочем, ради справедливости надо отметить, что некоторые операции низкоуровневых языков (например, работу с указателями), не так просто реализовать в LabVIEW ввиду его «высокоуровневости». Разумеется, язык LabVIEW включает основные конструкции управления, имеющие аналоги и в «традиционных» языках:

- переменные (локальные или глобальные)
- ветвление (case structure)
- For – циклы с проверкой завершения и без
- While – циклы
- Группировка операций.

LabVIEW – программа и возможности языка

В LabVIEW разрабатываемые программные модули называются «Virtual Instruments» (Виртуальные Инструменты) или, попросту, VI. Они сохраняются в файлах с расширением *.vi. VIs – это кирпичики, из которых состоит LabVIEW – программа. Любая LabVIEW программа содержит как минимум один VI. В терминах языка Си можно достаточно смело провести аналогию с функцией с той лишь разницей, что в LabVIEW одна функция содержится в одном файле (можно также создавать библиотеки инструментов). Само собой разумеется, один VI может быть вызван из другого VI. В принципе каждый VI состоит из двух частей — Блок-Диаграмма (Block Diagram) и Передняя Панель (Front Panel). Блок-диаграмма — это программный код (точнее визуальное графическое представление кода), а Передняя панель — это интерфейс. Вот как выглядит классический пример Hello, World!:

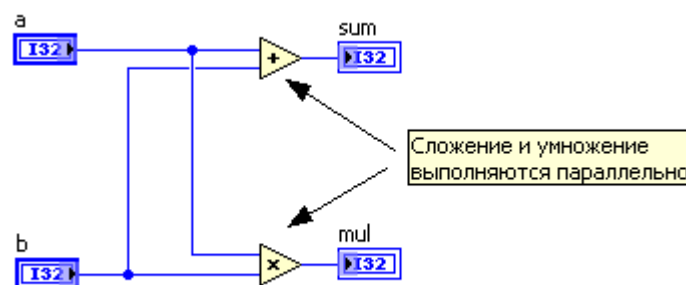


В основе LabVIEW лежит парадигма потоков данных. В вышеприведённом примере константа и терминал индикатора соединены между собой линией. Эта линия называется Wire. Можно назвать её «проводом». По проводам передаются данные от одних элементов другим. Вся эта концепция называется Data Flow. Суть Блок Диаграммы — это узлы (ноды), выходы одних узлов присоединены ко входам других узлов. Узел начнёт выполнение только тогда, когда придут все необходимые для работы данные. На диаграмме вверху две ноды. Одна из них — константа. Этот узел самодостаточен — он начинает выполнение немедленно. Второй узел — индикатор. Он отобразит данные, которые передаёт константа (но не сразу, а как только данные придут от константы).

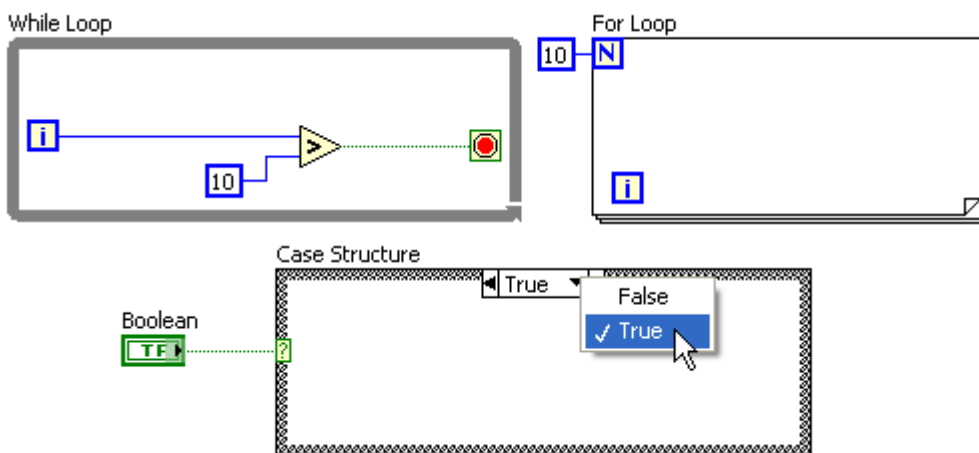
Вот чуть более сложный пример: сложение и умножение двух чисел. В традиционных языках мы напишем что-то вроде

```
int a, b, sum, mul;
//...
sum = a + b;
mul = a * b;
```

Вот как это выглядит в LabVIEW:

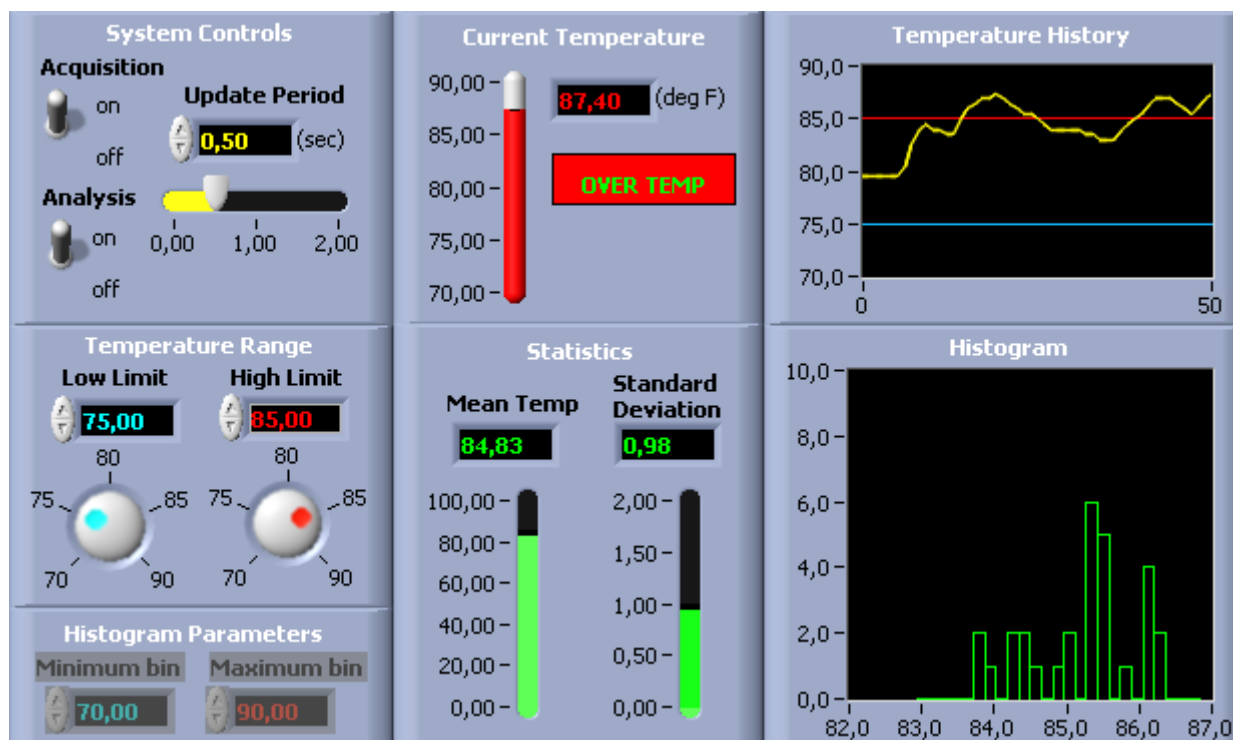


Обратите внимание на то, что сложение и умножение автоматически выполняются параллельно. На двухпроцессорной машине будут автоматически задействованы оба процессора. А вот как выглядят while / for циклы и if / then / else структура:



Как уже упоминалось, все элементы будут выполняться параллельно. Вам не нужно задумываться о том, как распараллелить задачу на несколько потоков, которые можно выполнять параллельно на нескольких процессорах. В последних версиях можно даже явно указать на каком из процессоров должен выполняться тот или иной while-цикл. Сейчас существуют надстройки и для текстовых языков, позволяющие запросто добиться поддержки многопроцессорных систем, однако так просто, как на LabVIEW, это пожалуй нигде не реализовано. (ну вот, я всё же скатился на сравнение с текстовыми языками). Если уж мы заговорили о многопоточности, то надо также отметить, что в распоряжении разработчика богатый выбор инструментов для синхронизации потоков — семафоры, очереди, рандеву, и т.д.

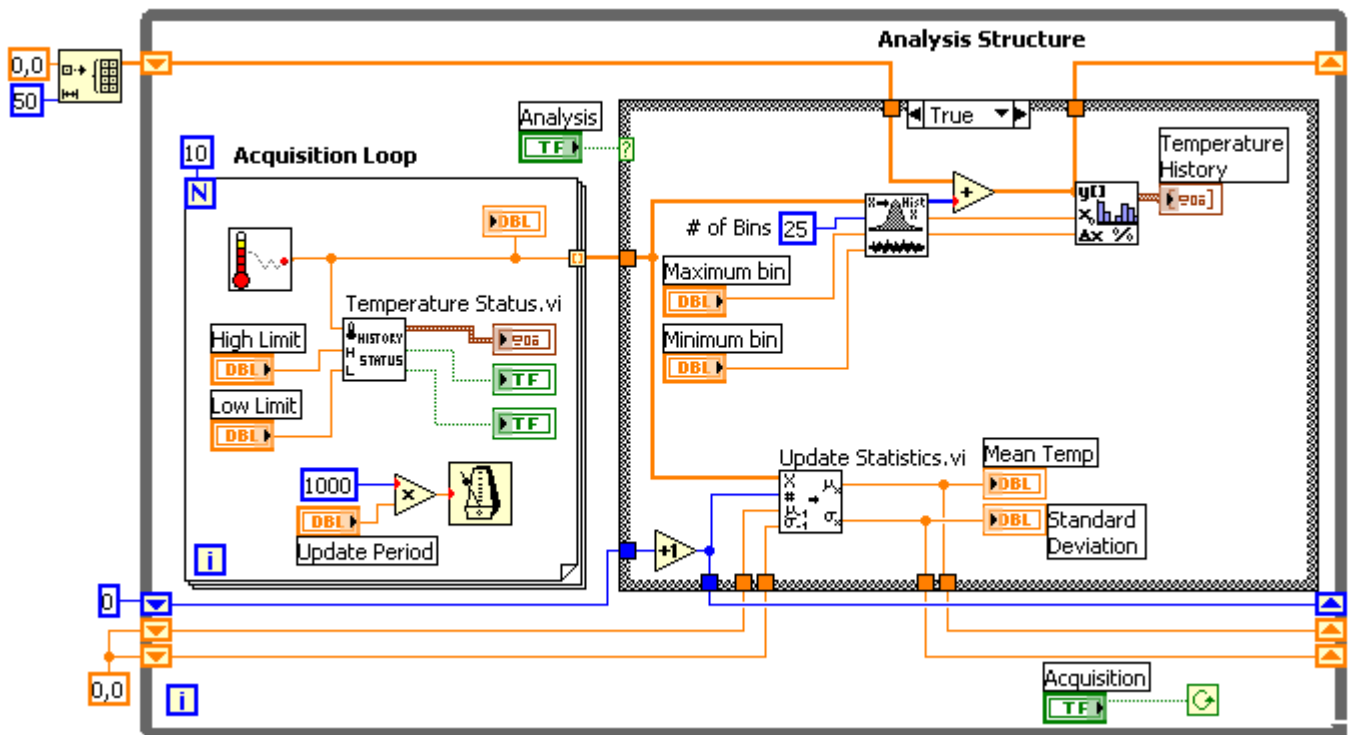
LabVIEW включает в себя богатые наборы элементов для построения пользовательских интерфейсов. Уж на что быстро «набрасывались» интерфейсы в Delphi, а в LabVIEW этот процесс происходит ещё стремительнее.



Стандартная поставка LabVIEW включает в себя также блоки для работы с ini файлами, реестром, функции для работы с двоичными и тестовыми файлами, математические функции, мощные инструменты для построения графиков (а куда же без этого в лаборатории-то), а в дополнение к уже упомянутой возможности вызовов DLL, LabVIEW позволяет работать с ActiveX компонентами и .NET. Начиная с восьмой версии в LabVIEW была добавлена поддержка классов — язык стал объ-

ектно-ориентированным. Реализованную поддержку нельзя назвать полной, однако основные черты объектно-ориентированных языков — наследование и полиморфизм присутствуют. Также функциональность языка можно расширить дополнительными модулями, например, NI Vision Toolkit — для обработки изображений и машинного зрения и другие. А при помощи модуля Application Builder можно сгенерировать исполняемый exe-файл. С помощью Internet Toolkit можно работать с ftp серверами, с помощью Database Connectivity Toolkit — с базами данных и т.д.

Часто можно услышать мнение, что графический код плохо читаем. Действительно, с непривычки обилие иконок и проводников несколько шокирует. Также начинающие разработчики создают программы-«простыни» и программы-«спагетти». Однако опытный LabVIEW-разработчик никогда не создаст диаграмм, превышающих размер экрана, даже если программа состоит из сотен модулей. Хорошо разработанная программа фактически «самодокументируется», поскольку



в основе уже лежит графическое представление.

Цель курса

Этот курс научит вас:

- Понимать назначение лицевых панелей, блок-диаграмм, иконок и панелей подключения
- Использовать программные структуры и типы данных LabVIEW
- Применять различные способы редактирования и отладки
- Создавать и сохранять VI так, чтобы их можно было использовать в качестве subVI
- Отображать и регистрировать данные

В этом курсе *не* изучаются:

- Все встроенные VI, функции или объекты; Обратитесь к *LabVIEW Help* для получения дополнительной информации о свойствах LabVIEW, не представленных в настоящем курсе
- Теория аналого-цифрового преобразования
- Принцип действия последовательного порта
- Принцип действия шины GPIB
- Разработка драйверов измерительных приборов
- Проектирование законченных приложений всеми студентами аудитории; обратитесь к поисковику примеров NI Example Finder, выбрав в меню **Help>>Find Examples**, чтобы найти примеры VI, которые можно использовать и включать в создаваемые вами VI

План занятия

На этом занятии объясняется, как ориентироваться в среде LabVIEW – как использовать меню, панели инструментов, палитры, инструменты, справки и стандартные диалоговые окна LabVIEW. Вы также научитесь запускать VI и получите общее представление о лицевой панели и блок-диаграмме. В конце урока вам предстоит создать простой VI, который выполняет сбор, обработку и представление данных.

Виртуальные приборы (VI)

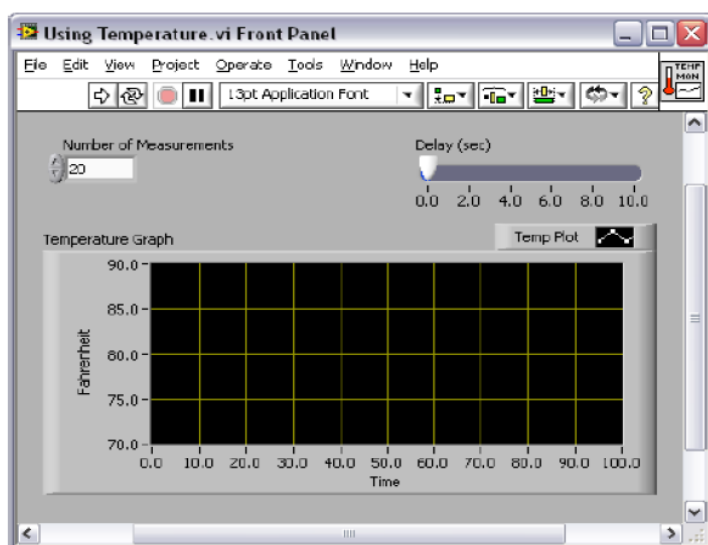
Программы, спроектированные в LabVIEW, называются виртуальными измерительными приборами, или VI, поскольку их внешний вид и функционирование имитируют реальные измерительные приборы, такие, как осциллографы и мультиметры. LabVIEW содержит полный набор инструментов для сбора, обработки, отображения и регистрации данных, а также средства, которые помогают отладить разрабатываемую программу.

Состав VI

VI, созданные в LabVIEW, состоят из трех основных компонентов: окна лицевой панели, блок-диаграммы и иконки/панели подключения.

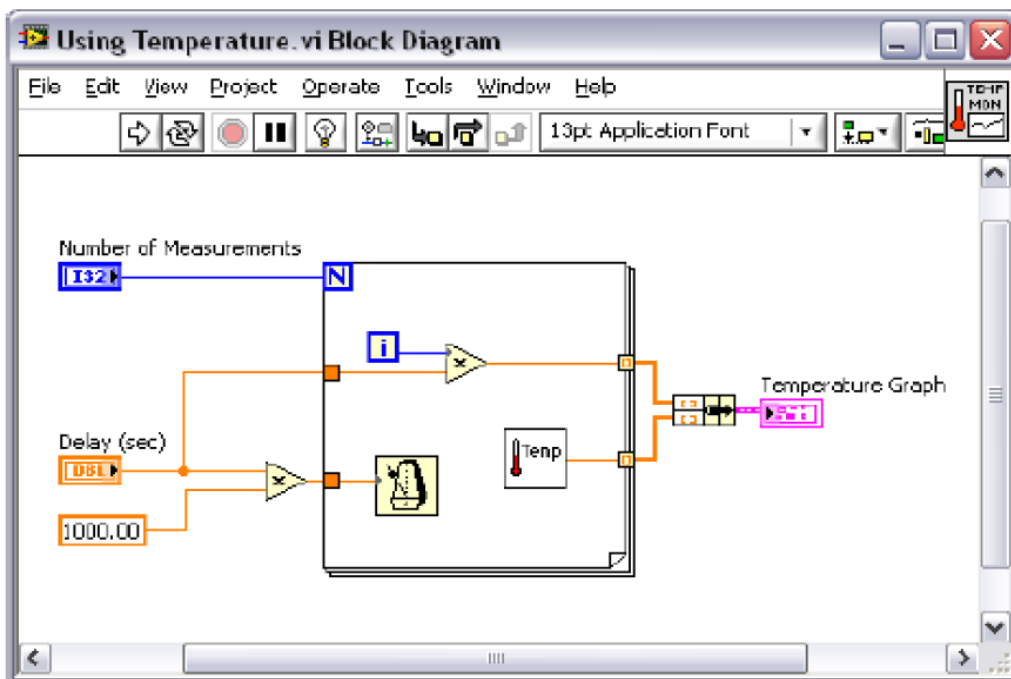
Окно лицевой панели

Окно лицевой панели является пользовательским интерфейсом VI. На рисунке приведен пример окна лицевой панели. Вы создаете окно лицевой панели вместе с органами управления и индикаторами, которые являются соответственно входными и выходными интерактивными терминалами VI.



Окно блок-диаграммы

После создания лицевой панели в блок-диаграмму добавляется программный код с использованием функций, представленных в графической форме, которые управляют объектами на фронтальной панели. На рисунке приведен пример окна блок-диаграммы. В окне блок-диаграммы содержится исходный код программы. Объекты, размещенные на лицевой панели, появляются на блок-диаграмме в виде терминалов.



Иконка и панель подключения

Иконка и панель подключения позволяют использовать и просматривать VI внутри другого VI. VI, используемый внутри другого VI, называется subVI, и является аналогом функции в текстовом языке программирования. Чтобы использовать VI в качестве subVI, нужно его снабдить иконкой и панелью подключения.



Иконка отображается в верхнем правом углу окна лицевой панели и окна блок-диаграммы каждого VI и является графическим представлением VI. Пример иконки "по умолчанию" показан слева. В иконке может содержаться как текст, так и изображения. Если вы используете VI в качестве subVI, то он обозначается на блок-диаграмме VI иконкой. Иконка "по умолчанию" содержит число, которое показывает, сколько новых VI вы открыли после того, как запустили LabVIEW.



Чтобы использовать VI в качестве subVI, необходимо создать панель подключения, показанную слева. Панель подключения является набором терминалов иконок, которые соответствуют органам управления и индикаторам данного VI, это похоже на список параметров функции, вызываемой в текстовых языках программирования. Доступ к панели подключения осуществляется щелчком правой кнопки мыши по иконке в верхнем правом углу окна лицевой панели. В окне блок-диаграммы панель подключения открыть нельзя.

Начинаем проектировать VI

После запуска LabVIEW на экране открывается окно Getting Started. Это окно следует использовать для создания новых VI и проектов, выбора каких-либо из последних открытых файлов LabVIEW, поиска примеров и справочной информации в *LabVIEW Help*. Кроме того, из стартового окна обеспечивается доступ к информации и ресурсам, которые помогут вам изучить LabVIEW, например, к специальным руководствам, разделам справочной системы, а также Интернет-ресурсам по адресу ni.com/manuals

Чтобы открыть новый проект из окна **Getting Started**, выберите пункт **Empty Project** в списке **New**. В этом случае открывается новый проект без имени, в который можно добавить файлы и затем сохранить его. Чтобы открыть новый пустой VI, который не связан с проектом, выберите пункт **Blank VI** в списке **New** в окне **Getting Started**.

Project Explorer

Проекты предназначены для группирования файлов LabVIEW, а также файлов других типов, создания спецификации строителя приложений, развертывания или загрузки файлов в целевые устройства. Когда вы сохраняете проект, LabVIEW создает файл проекта (.lvproj), который включает в себя ссылки на файлы проекта, информацию о конфигурации, сборке приложения, развертывании и т.д. Проекты необходимы для построения приложений и совместно используемых библиотек. Из проектов работают с различными целевыми устройствами – реального времени (RT), программируемыми логическими интегральными схемами (FPGA) или карманными компьютерами (PDA). Дополнительная информация по применению проектов совместно с модулями LabVIEW Real-Time, FPGA и PDA приведена в соответствующей документации на модули.

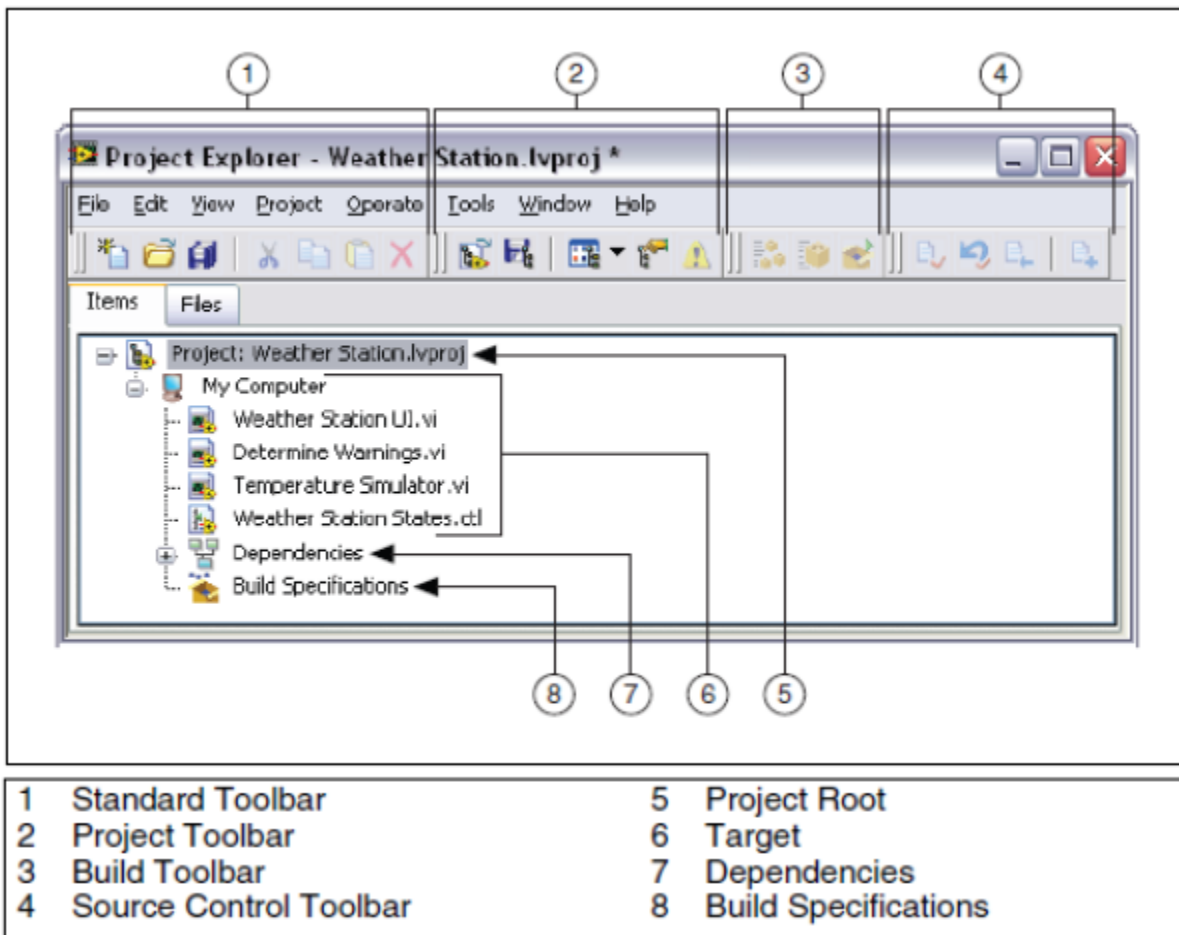
Чтобы создать проект, выполните следующие действия:

1. Выберите команду меню **File»New Project** для вывода на экран окна Project Explorer. Для этой же цели можно также выбрать пункт **Project»Empty Project** в диалоговом окне New.
2. Добавьте элементы для целевого устройства, которые вы хотите включить в проект.
3. Для сохранения проекта выберите команду меню **File»Save Project**.

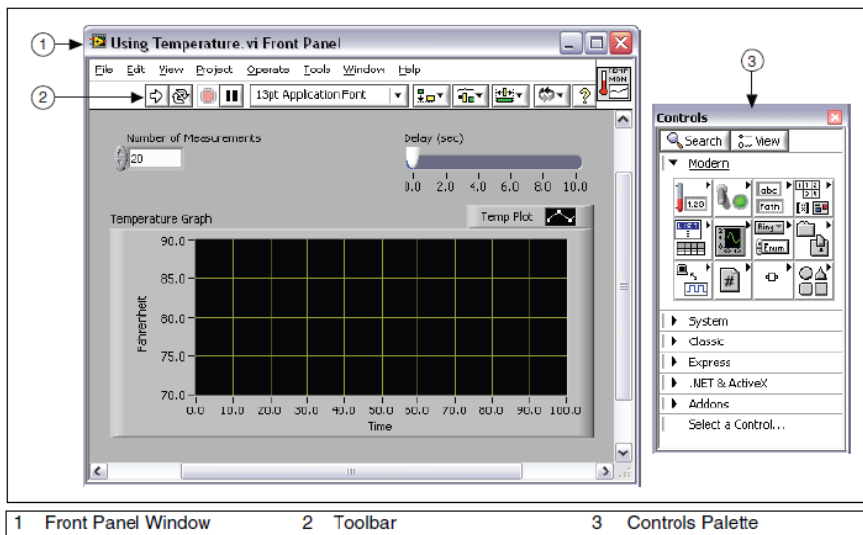
В проект можно добавлять существующие файлы. Чтобы добавлять в проект LabVIEW файлы VI или текстовые файлы, используйте пункт **My Computer** (или пункт, представляющий другое целевое устройство) в окне **Project Explorer**.

В проект можно добавлять элементы следующими способами:

- Щелкните правой кнопкой мыши по ветви проекта **My Computer** и выберите из контекстного меню команду **Add»File**, чтобы добавить файл. Для этой же цели можно также выбрать команду **Project»Add to Project»File** из меню Project Explorer.
- Чтобы добавить автоматически обновляющуюся папку, щелкните правой кнопкой мыши по ветви проекта, представляющей целевое устройство, и из контекстного меню выберите команду **Add»Folder (Auto-populating)**. Для этой же цели можно выбрать команду меню **Project»Add To Project»Add Folder (Auto-populating)**. LabVIEW непрерывно контролирует и обновляет папку в соответствии с изменениями, вносимыми в проект и на диск.
- Чтобы добавить виртуальную папку, щелкните правой кнопкой мыши по ветви, представляющей целевое устройство, и выберите из контекстного меню команду **Add»Folder (Snapshot)**. Для этой же цели можно выбрать команду меню **Project»Add To Project»Add Folder (Snapshot)**. Когда вы выберете папку на диске, LabVIEW создает в проекте новую виртуальную папку с тем же именем, что и папка на диске. LabVIEW также создает элементы проекта, которые отображают содержимое всей папки, включая файлы и содержимое вложенных папок. При выборе папки на диске добавляется все содержимое папки, включая файлы и содержимое вложенных папок.
- Чтобы добавить новый пустой VI, щелкните правой кнопкой мыши по элементу, соответствующему целевому устройству, и выберите из контекстного меню команду **New»VI**. Для этой же цели можно также выбрать команду меню **File»New VI** или **Project»Add To Project»New VI**.
- Выберите в правом верхнем углу лицевой панели или блок-диаграммы иконку VI и перетащите ее на элемент, соответствующий целевому устройству.
- Выберите элемент или папку из файловой системы на вашем компьютере и перетащите его/ее на элемент, соответствующий целевому устройству.



Лицевая панель

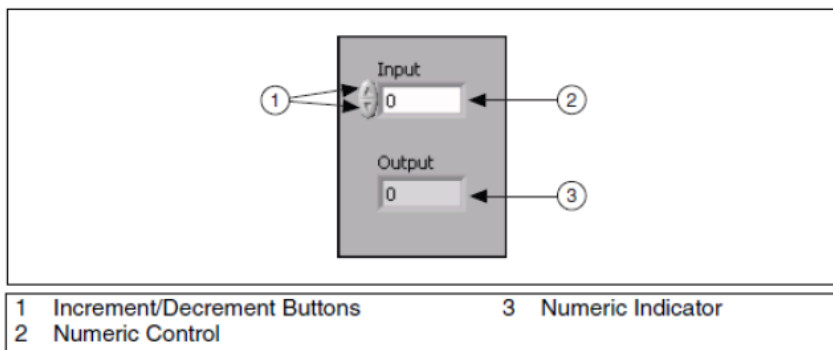


Когда вы открываете новый или существующий VI, на экране появляется окно лицевой панели, которая является пользовательским интерфейсом VI.

Вы создаете лицевую панель с органами управления и индикаторами, которые являются соответственно интерактивными входными и выходными терминалами VI. К органам управления относятся регуляторы, кнопки, лимбы и другие устройства ввода. К индикаторам относятся графики (графические экраны), светодиодные индикаторы и т.д. Органы управления симулируют входные устройства измерительных приборов и поставляют данные в

блок-диаграмму VI. Индикаторы симулируют выходные устройства измерительных приборов, и на них отображаются данные, которые собраны или сгенерированы на блок-диаграмме.

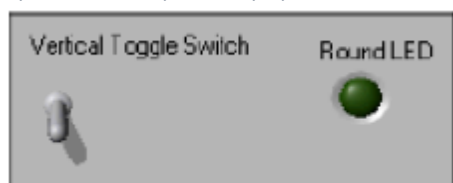
Числовые органы управления и индикации



Числовой тип данных может служить для представления чисел различных типов, например, целочисленного или вещественного. Такие объекты, как шкальные индикаторы и лимбы также служат для представления числовых данных. Чтобы ввести или изменить значения числового элемента управления, щелкните инструментом Operating по кнопкам инкремента и декремента или щелкните дважды инструментом Labeling либо Operating по числу, а затем введите новое

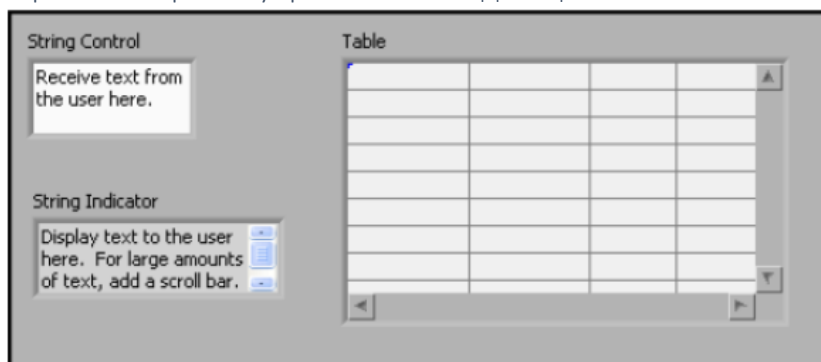
число и нажмите на клавишу <Enter>.

Булевские органы управления и индикации



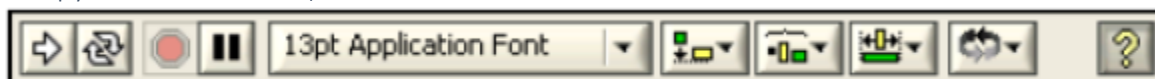
Булевский тип данных служит для представления данных, которые могут принимать только два значения: TRUE (ON) и FALSE (OFF). Булевские органы управления и индикации применяют для ввода и отображения булевских величин. Булевские объекты симулируют переключатели, кнопки и светодиодные индикаторы.

Строковые органы управления и индикации



Строковый тип данных представляет собой последовательность ASCII-символов. Строковые органы управления служат для получения такой текстовой информации от пользователя, как пароль или имя пользователя. Используйте строковые индикаторы для того, чтобы вывести пользователю текстовую информацию.

Панель инструментов окна лицевой панели



Чтобы запустить VI, щелкните мышью по кнопке Run. Если необходимо, LabVIEW компилирует VI. Вы можете запускать VI, если стрелка на кнопке Run сплошь закрашена в белый цвет, как показано слева. Такой вид стрелки показывает также, что VI можно использовать в качестве subVI, если создадите панель подключения VI.



Чтобы запустить VI на исполнение до момента, когда вы прервете или приостановите исполнение, щелкните мышью по кнопке Run Continuously. Вы можете еще раз щелкнуть по этой кнопке, чтобы запретить непрерывное исполнение.



В процессе выполнения VI появляется кнопка Abort Execution. Щелкните мышью по этой кнопке, чтобы немедленно остановить VI, если нет никаких других способов его остановить. Если VI используется несколькими VI более высокого уровня, кнопка становится недоступной.

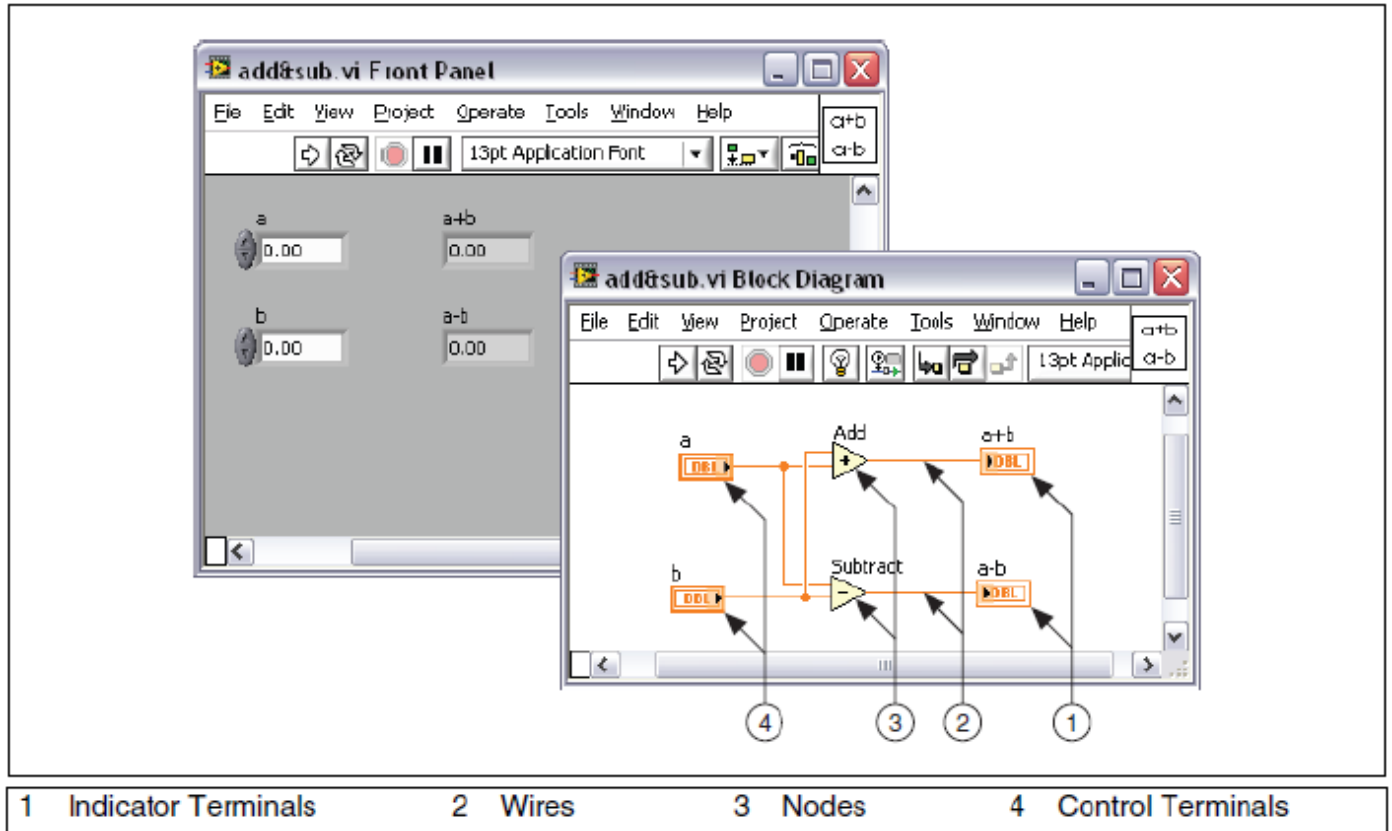


Чтобы приостановить выполнение VI, щелкните мышью по кнопке Pause. Когда вы щелкаете по кнопке Pause, LabVIEW подсвечивает место на блок-диаграмме, на котором приостановлено исполнение VI, а кнопка Pause становится красного цвета. Чтобы продолжить выполнение VI, щелкните еще раз по кнопке Pause.

13pt Application Font Чтобы изменить настройки шрифта для отдельных фрагментов VI, в том числе размер, стиль и цвет шрифта, выберите выпадающее меню Text Settings.

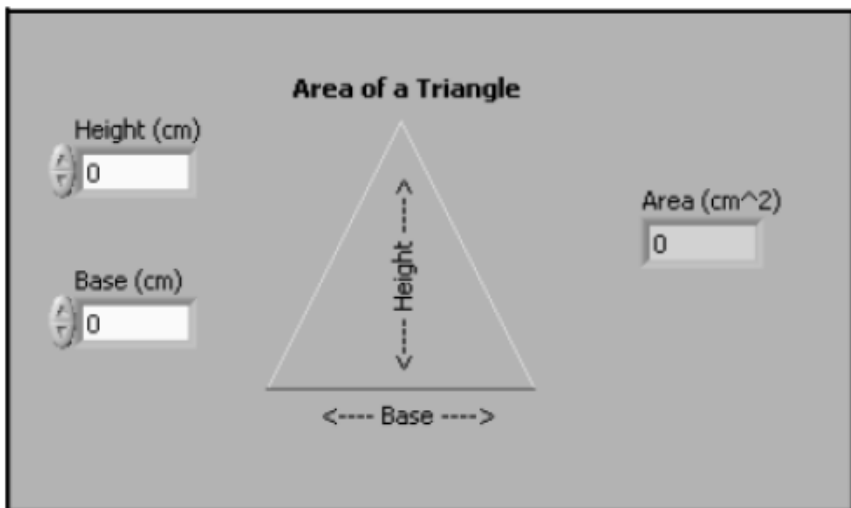
Остальные кнопки относятся к выравниванию элементов, выстраиванию вдоль линий и так далее. Намного проще разобратся с ними самостоятельно, чем читать документацию.

Блок-диаграмма



К объектам блок-диаграммы относятся терминалы, subVI, функции, константы, структуры, а также проводники, по которым данные передаются между объектами.

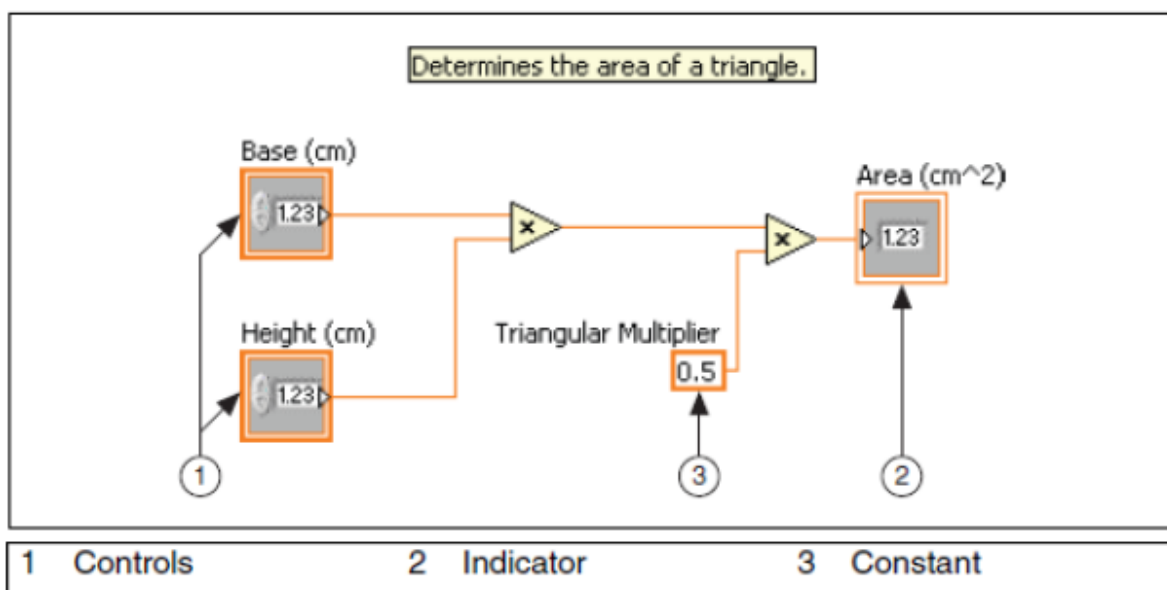
Объекты лицевой панели проявляются на блок-диаграмме в виде терминалов. Терминалы — это входные и выходные порты, через которые осуществляется обмен информацией между лицевой панелью и блок-диаграммой. Терминалы являются аналогами параметров и констант в текстовых языках программирования. Терминалы бывают у элементов управления, элементов индикации, а также у узлов. Терминалы элементов управления и индикации принадлежат соответствующим органам управления и индикации на лицевой панели. Данные, которые вы вводите в элементы управления на лицевой панели, поступают через соответствующие терминалы на блок-диаграмму. Здесь данные подаются на входы функций Add и Subtract. Когда эти функции завершают вычисления, они выдают новые значения данных, которые проходят на терминалы индикаторов, обновляя содержимое соответствующих индикаторов на лицевой панели.



Элементы управления, индикаторы и константы являются входами и выходами алгоритма, реализуемого блок-диаграммой. Рассмотрим реализацию алгоритма вычисления площади треугольника:

$$\text{Area} = 0.5 * \text{Base} * \text{Height}$$

Пользователь не может получить доступ к константе 0.5 и не может изменить ее, поскольку она, в соответствии с алгоритмом, не отображается на лицевой панели












Обратите внимание на то, что внешний вид терминалов Base (cm) и Height (cm) на блок-диаграмме отличается от внешнего вида терминала Area (cm²). Есть два отличительных признака между элементом управления и индикатором на блок-диаграмме. Первый из них стрелка на терминале, которая показывает направление потока данных. Стрелки на терминалах элементов управления показывают направление, по которому данные покидают терминалы, в то время как стрелка на терминалах индикаторов показывает направление, по которому данные поступают в терминалы. Второй отличительный признак рамка вокруг терминала. Рамка терминалов элементов управления более толстая, чем рамка терминалов индикаторов.

Узлы это объекты блок-диаграммы, у которых есть входы и/или выходы и которые выполняют действия в исполняемом VI. Они являются аналогами выражений, операторов, функций и подпрограмм в текстовых языках программирования. Узлами могут быть функции, subVI и структуры. Структуры являются элементами управления процессом, например, Case, For Loop, While Loop. Функции Add и Subtract на рисунке 2-15 являются узлами функций.

Данные передаются между объектами блок-диаграммы по проводникам. Каждый проводник имеет один источник данных, однако, его можно присоединять ко многим VI и функциям, которые считывают эти данные. Проводники, в зависимости от типов передаваемых по ним данных, бывают различных цветов, стилей и толщины.

--- ✖ --- Разорванный проводник имеет вид прерывистой черной линии с красным крестиком посередине, как показано слева. Обрывы проводников возникают из-за множества причин, в том числе, при попытке соединить два объекта с несовместимыми типами данных.

| Тип проводника | Скаляр | 1D массив | 2D массив | Цвет |
|----------------|---|---|--|---|
| Числовой |  |  |  | Оранжевый (с плавающей точкой), Синий (целочисленный) |
| Логический |  |  |  | Зеленый |
| Строковый |  |  |  | Розовый |

В LabVIEW, чтобы передавать данные в VI, используются проводники, соединяющие между собой множество терминалов. Следует подсоединять проводники к тем входам и выходам, которые совместимы с данными, передаваемыми по этим проводникам. Нельзя, например, соединять выход массива с числовым входом. Кроме того, необходимо, чтобы проводники соединяли только один выход, по крайней мере, с одним входом. Нельзя, например, соединять между собой два индикатора. Компонентами, которыми определяется совместимость соединения, являются тип данных элемента управления и/или индикатора и тип данных терминала.

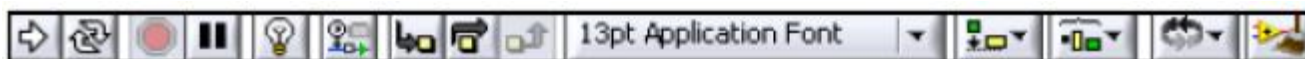
Автоматическое соединение объектов

Если приблизить выделенный объект к другим объектам на блок-диаграмме, LabVIEW нарисует временные проводники, которые показывают корректные соединения. Когда вы отпустите кнопку мыши, чтобы поместить объект на блок-диаграмму, LabVIEW автоматически подключит проводники. Вы можете также автоматически соединять объекты уже на блок-диаграмме. LabVIEW выполняет соединения наиболее совместимых терминалов и не соединяет терминалы, которые несовместимы. Автоматическое выполнение соединения включается клавишей пробела, когда вы перемещаете объект с помощью инструмента Positioning. Автоматическое выполнение соединений разрешено по умолчанию, если объект выбирается из палитры Functions или при копировании объекта на блок-диаграмме нажатием клавиши <Ctrl> и перетаскиванием объекта. Автоматическое выполнение соединений по умолчанию запрещено, если вы с помощью инструмента Positioning перемещаете объект, уже находящийся на блок-диаграмме. Установить настройки автоматического выполнения соединений можно, выбрав команду меню Tools»Options и пункт Block Diagram из списка Category.

Соединение объектов вручную

Когда инструмент Wiring перемещается над терминалом, появляется подсказка с именем терминала. Кроме того, терминал мерцает в окне Context Help и на иконке, помогая вам убедиться в том, что выполняемое соединение с данным терминалом корректно. Чтобы соединить объекты между собой, щелкните мышью, наведя инструмент Wiring на первый терминал, затем поместите курсор на второй терминал и снова щелкните мышью. После выполнения соединения можно щелкнуть правой кнопкой мыши по проводнику и выбрать из контекстного меню команду Clean Up Wire, чтобы LabVIEW автоматически выбрала путь прохождения этого проводника. Если у вас есть разорванные проводники, которые необходимо удалить, нажмите комбинацию клавиш <Ctrl-B>, чтобы удалить все разорванные проводники на блок-диаграмме.

Панель инструментов блок-диаграммы



Щелкните мышью по кнопке Highlight Execution, чтобы анимировать исполнение блок-диаграммы при запуске VI. Обратите внимание на движение потока данных по блок-диаграмме. Щелкните еще раз мышью по этой кнопке, чтобы запретить подсветку выполнения



Щелкните мышью по кнопке Retain Wire Values для сохранения значений данных в проводниках в каждой точке выполняемого потока, чтобы при установке пробника на проводник можно было получать последнее значение данных, прошедшее по проводнику. Эти последние значения будут доступны только, если VI успешно выполнится как минимум один раз.



Щелкните мышью по кнопке Clean Up Diagram, чтобы автоматически переразвести все имеющиеся проводники и упорядочить объекты на блок-диаграмме, скомпоновав их более аккуратно. Конфигурирование параметров перекомпоновки объектов на блок-диаграмме осуществляется командой меню Tools»Options, и выбором пункта Block Diagram: Cleanup из списка Category в открывающемся диалоговом окне Options.

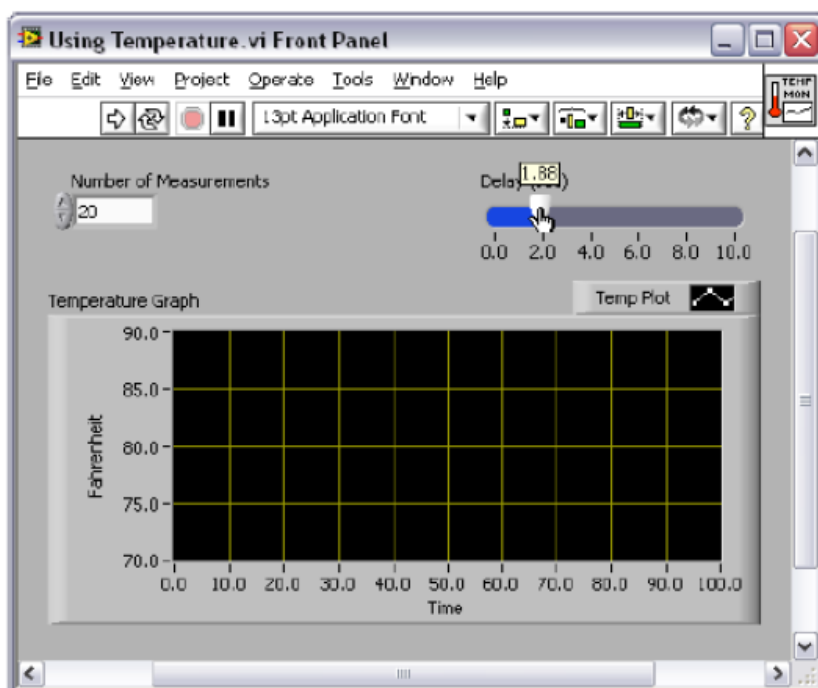
Выбор инструмента

С помощью инструментов LabVIEW можно создавать, модифицировать и отлаживать VI. Инструмент — это специальный режим работы курсора мыши. Режим работы курсора соответствует иконке выбранного инструмента. В зависимости от текущего положения мыши LabVIEW определяет, какой выбрать инструмент.

Инструмент Operating



Инструмент Operating становится рабочим, когда курсор мыши принимает вид, показанный слева. Этот инструмент используется для изменения значений элемента управления. Например, на рисунке ниже инструмент Operating перемещает указатель на элементе управления Horizontal Pointer Slide. Когда курсор мыши наводится на указатель, он автоматически выбирает инструмент Operating.



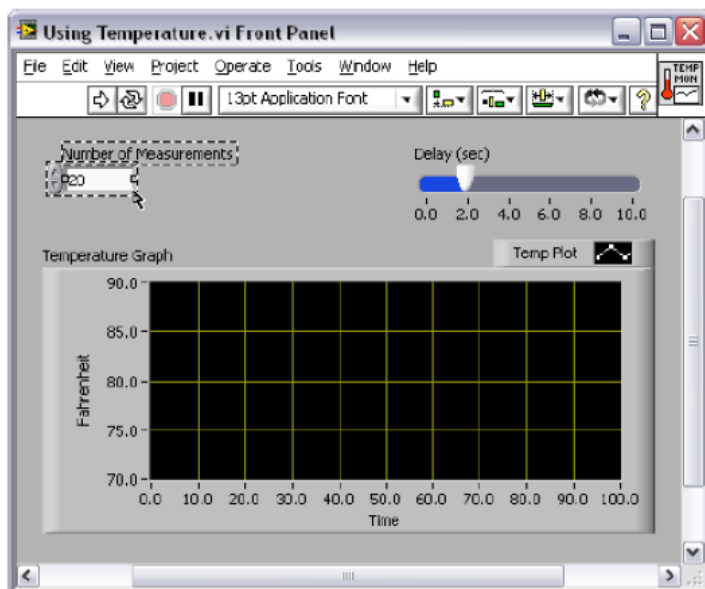
Инструмент Operating чаще всего используется в окне лицевой панели, однако, его можно также использовать и в окне блок-диаграммы, чтобы изменить значение константы типа Boolean.

Инструмент Positioning



Инструмент Positioning становится рабочим, когда курсор мыши принимает вид, показанный слева. Используйте этот инструмент для выделения и изменения размеров объектов. Например, на рисунке ниже инструмент Positioning выделяет числовой элемент управления Number of Measurements. После того, как объект выделен, его можно переместить, скопировать или удалить. Когда курсор мыши наводится на границу объекта, он автоматически превращается в инструмент Positioning.

Если курсор навести на реперную точку изменения размера объекта, режим курсора изменяется, показывая, что этот размер можно изменять.

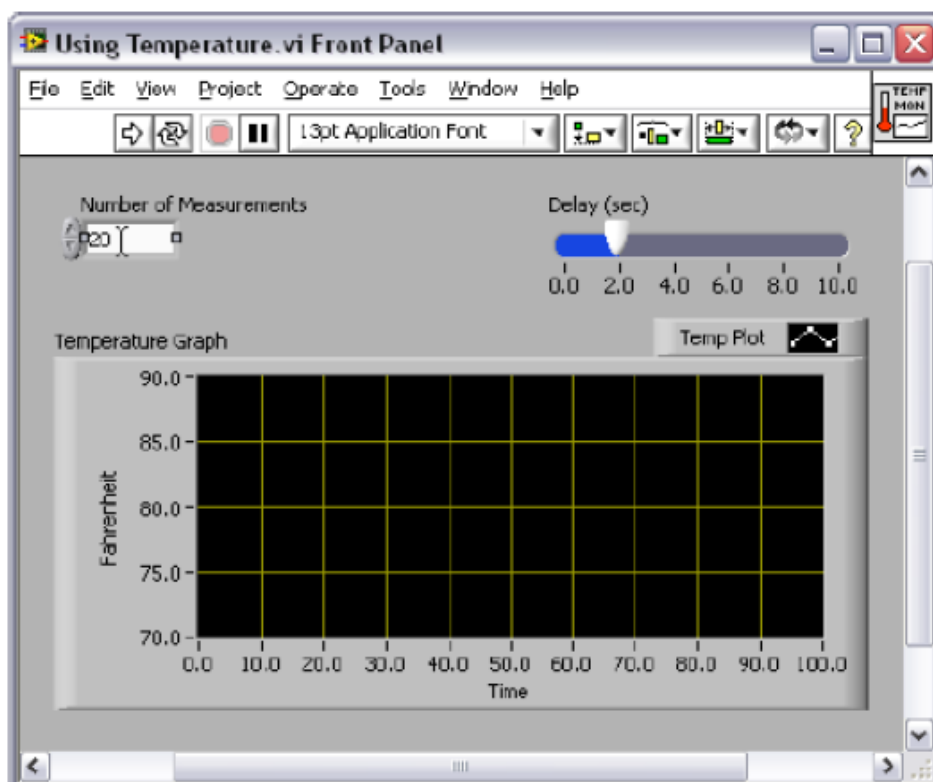


Инструмент Labeling



Инструмент Labeling становится рабочим, когда курсор мыши принимает вид, показанный слева. Этот инструмент используйте для ввода текста в элемент управления, редактирования текста и создания свободных меток.

Например, на рисунке ниже с помощью инструмента Labeling вводится текст в числовой элемент управления Number of Measurements. Когда курсор мыши наведен на внутреннюю часть этого элемента управления, он автоматически превращается в инструмент Labeling. Щелкните один раз мышью, чтобы поместить курсор внутрь элемента управления. Затем сделайте двойной щелчок, чтобы выделить находящийся в элементе текст.



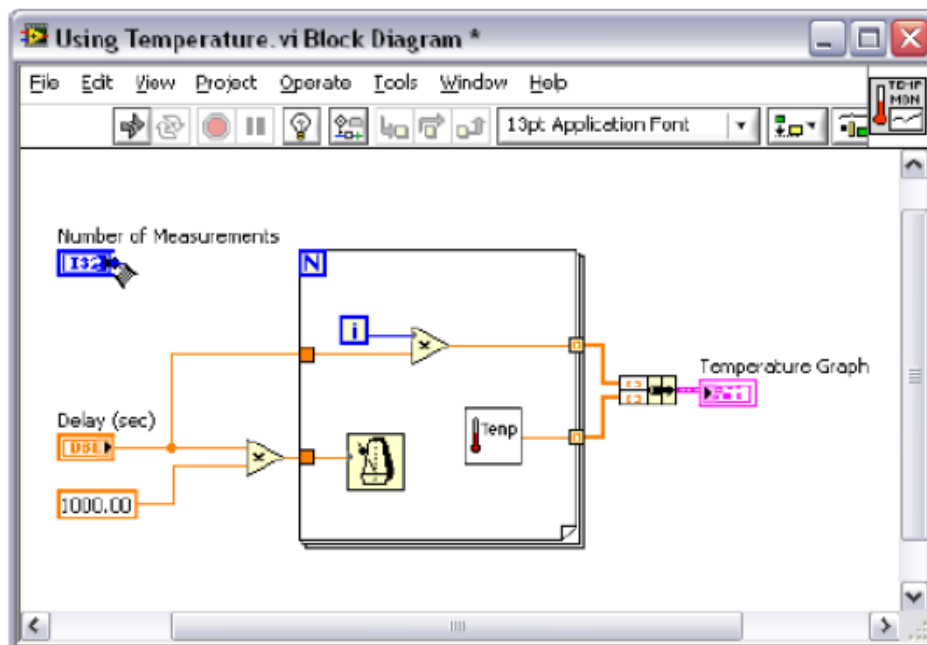
Когда курсор находится вне определенной области окна лицевой панели или окна блок-диаграммы, в которой возможен выбор конкретного режима использования мыши, курсор принимает вид перекрестия. Если разрешен автоматический выбор инструмента, можно, щелкнув дважды по любому свободному месту, выбрав тем самым инструмент Labeling, и создать свободную метку.

Инструмент Wiring



Инструмент Wiring становится рабочим, когда курсор мыши принимает вид, показанный слева. Этот инструмент используется для соединения объектов на блок-диаграмме между собой. Например, на рисунке ниже инструмент Wiring соединяет терминал Number of Measurements с терминалом количества итераций структуры For Loop.

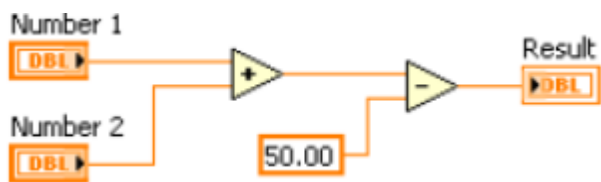
Когда курсор мыши наведен на выходную или входную точку терминала или на проводник, курсор автоматически превращается в инструмент Wiring.



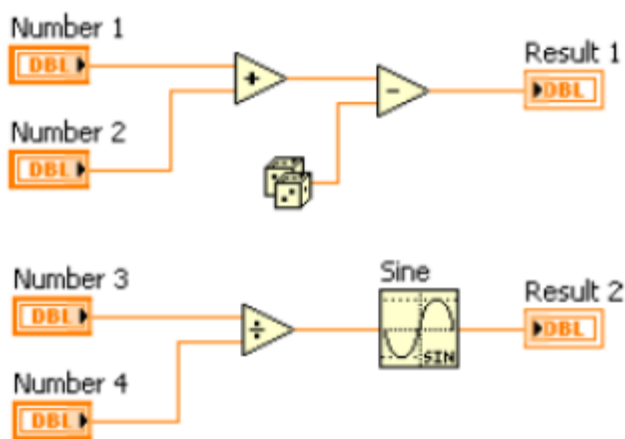
С инструментом Wiring в основном работают в окне блок-диаграммы и при создании панели подключения в окне лицевой панели.

Потоковое программирование

В LabVIEW VI выполняются под управлением потока данных, и узел блок-диаграммы выполняется только тогда, когда получит все требуемые входные данные. По завершении выполнения узел выдает выходные данные и передает их следующему узлу на пути распространения потока данных. Продвижение данных через узлы определяет порядок выполнения VI и функций на блок-диаграмме. Программы на Visual Basic, C++, JAVA и в большинстве других текстовых языках программирования выполняются в соответствии с моделью управления потоком команд, согласно которой последовательность программных элементов определяет порядок выполнения программы.

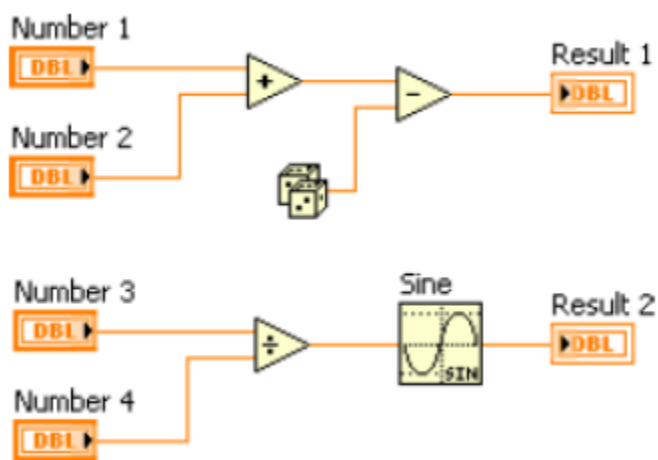


В качестве примера потокового программирования рассмотрим блок-диаграмму, приведенную на рисунке, которая складывает два числа, и затем из суммы этих чисел вычитает число 50.00. В этом случае блок-диаграмма выполняется слева направо не из-за того, что объекты размещены в таком порядке, а вследствие того, что функция Subtract не может выполняться до тех пор, пока не выполнится функция Add, которая передаст данные функции Subtract. Следует помнить о том, что узел выполняется только тогда, когда данные доступны на всех его входных терминалах, и выдает данные на выходные терминалы только тогда, когда узел закончит выполнение.



Проанализируем, какой сегмент программного кода на рисунке будет выполняться в первую очередь функция Add, Random Number или Divide. Вы не сможете это узнать, поскольку данные на входах функций Add и Divide доступны одновременно, а у функции Random Number нет входов. В ситуации, когда один сегмент программного кода должен выполняться раньше другого, причем функции взаимно независимы относительно данных, следует пользоваться другими методами программирования, например, которые используют кластеры ошибок, чтобы принудительно задать порядок выполнения программы.

Самопроверка (короткий тест)



Какая функция выполняется первой: Add или Subtract?

- **Add**
- Subtract
- Неизвестно какая

Какая из функций выполняется первой: Sine или Divide?

- Sine
- **Divide**
- Неизвестно какая

Какая из функций выполняется первой: Random Number, Divide или Add?

- Random Number
- Divide
- Add
- **Неизвестно какая**

Какая из функций выполняется последней: Random Number, Subtract или Add?

- Random Number
- **Subtract**
- Add
- Неизвестно какая

Разработка простейшего VI

Самое время попытаться разработать какую-то простейшую программу, чтобы под присмотром преподавателя проделать первые шаги. В качестве варианта можно сделать вот что:

- Простейший калькулятор с выбором арифметического действия

- Мнемосхему условного прибора, который измеряет давление и отображает с помощью красивого индикатора. В случае, когда значение превышает пороговое, переключать лампочку на мнемосхеме