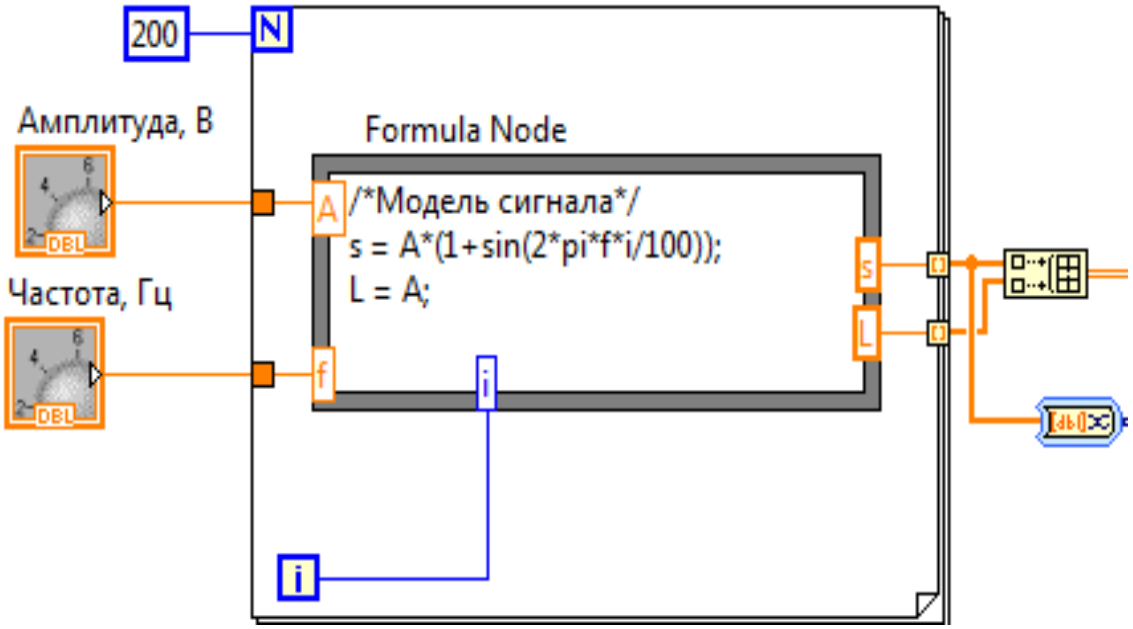
5209

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ОСНОВЫ РАБОТЫ С ПАКЕТОМ LABVIEW

Методические указания



Рязань 2018

УДК 004.4`2

Основы работы с пакетом LabVIEW: методические указания/ Рязан. гос. радиотехн. ун-т; сост. В.В. Карасев. Рязань, 2018. 16 с.

Содержат методический материал к теме, посвящённой аппаратно-программным комплексам фирмы National Instruments. Представлены основные действия в среде LabVIEW, показаны информационные технологии разработки виртуальных приборов в пакете. Практическая часть указаний связана с реализацией простых виртуальных приборов, предназначенных для моделирования сигналов-переносчиков информации и их последующей обработки.

Предназначены для изучения дисциплины «Аппаратно-программные комплексы информационных систем» студентами очной и очно-заочной форм обучения по направлению подготовки 09.03.02 «Информационные системы и технологии» (степень – бакалавр).

Библиогр.: 3 назв.

Блок-диаграмма, виртуальный прибор, лицевая панель, терминал

Печатается по решению редакционно-издательского совета Рязанского государственного радиотехнического университета.

Рецензент: кафедра АСУ РГРТУ (зав. кафедрой канд. техн. наук, доц. С.И. Холопов)

Основы работы с пакетом LabVIEW

Составитель К а р а с е в Виктор Владимирович

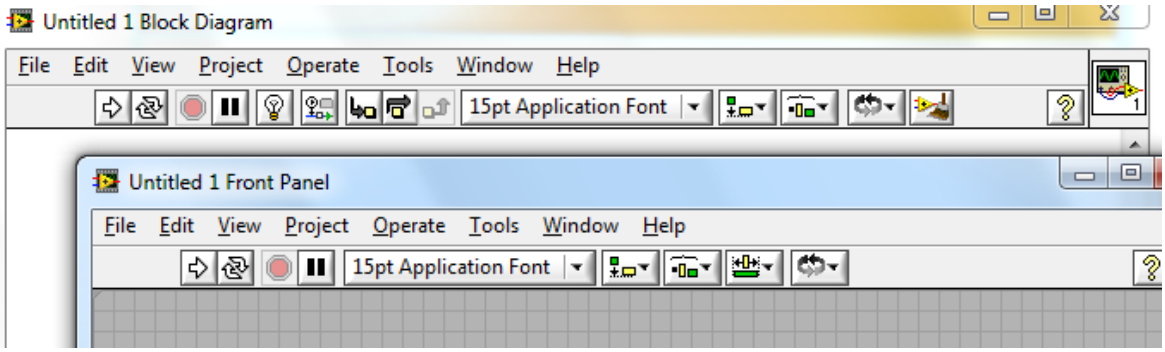
Редактор Н.А. Орлова
Корректор С.В. Макушина
Подписано в печать 29.01.18. Формат бумаги 60 х 84 1/16.
Бумага писчая. Печать трафаретная. Усл.печ.л. 1,0.
Тираж 25 экз. Заказ
Рязанский государственный радиотехнический университет.
390005, Рязань, ул. Гагарина, 59/1.
Редакционно-издательский центр РГРТУ.

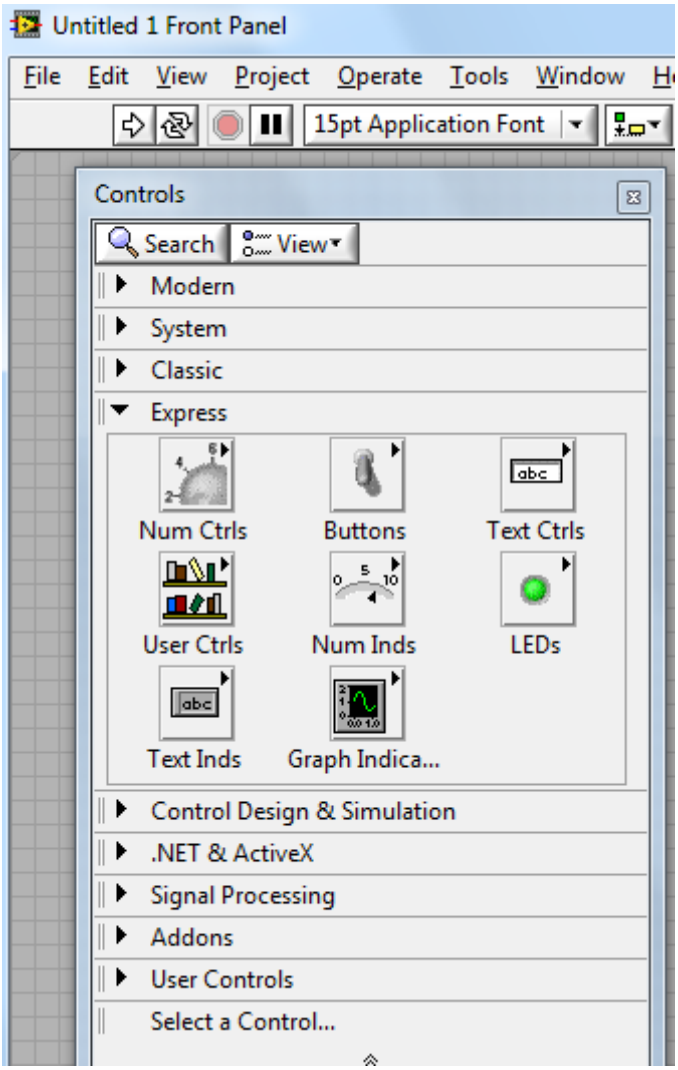
Введение в среду пакета LabVIEW

Пакет LabVIEW создан разработчиками фирмы National Instruments (США). Его первая версия вышла в 1986 г. для Mac OS, а с 1992 г. версии работают и под управлением Windows. LabVIEW предполагает создание программы в форме виртуального прибора (**ВП** – **VI** от Virtual Instrument) [1]. Каждый ВП представлен в среде разработки своей лицевой панелью (**Front Panel**), которая определяет пользовательский интерфейс, и блок-диаграммой (**Block Diagram**), описывающей функциональные возможности ВП. Кроме этого, каждый ВП имеет компактное представление в виде пиктограммы (**Icon**) с соединительной панелью (**Connector**). Иконка позволяет использовать ВП как подпрограмму (как **ВПП** – виртуальный подприбор, **SubVI**). Пакет содержит богатейшие библиотеки ВП для различных областей применения, предполагающих регистрацию, обработку и анализ сигналов и данных, моделирование процессов и систем, взаимодействие с оборудованием систем реального времени, работу с базами данных и многое другое.

Графический язык программирования "G", используемый в LabVIEW, позволяет создавать программу в стиле разработки традиционной блок-схемы алгоритма. Лицевая панель ВП создаётся своими инструментальными средствами пакета, а блок-диаграмма – своими. Названные средства сгруппированы на соответствующих графических инструментальных панелях, которые автоматически отслеживают действия разработчика. Ниже показано окно среды пакета после её запуска.

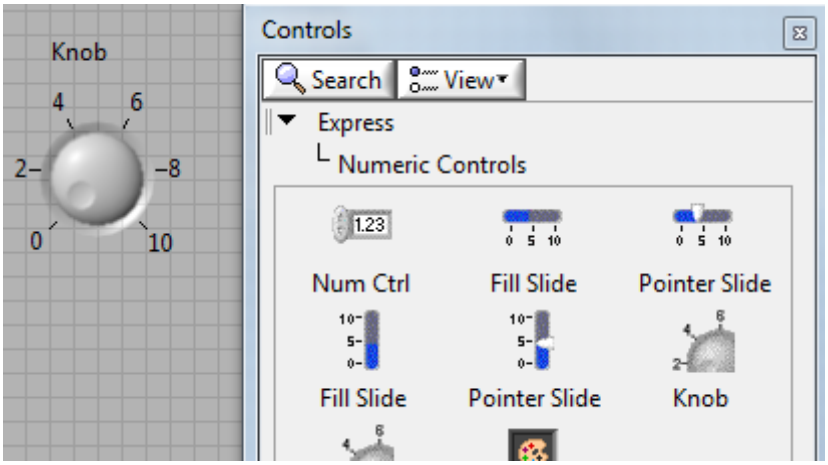
Для создания нового ВП выполняют команду **New/Blank VI**. В результате на экране появляются окна (следующий рисунок) для разработки операторского интерфейса (Front Panel) и блок-диаграммы (Block Diagram). Активность (видимость) окон меняется сочетанием клавиш **Ctrl + E** (см. секцию Window главного меню) или мышью.



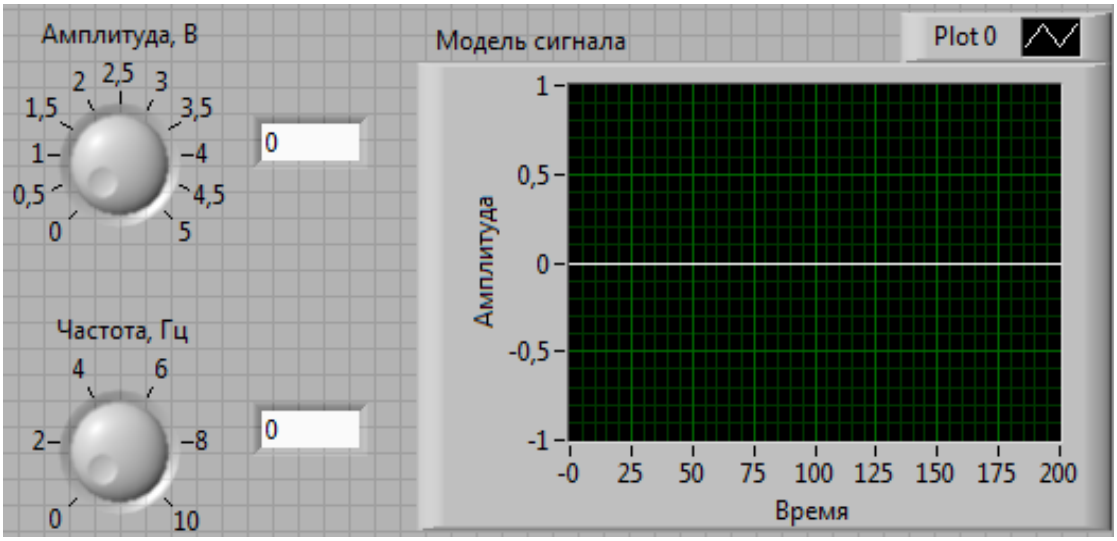


Для создания операторского интерфейса вызывают правой кнопкой мыши (ПКМ) инструментальные панели (рисунок слева), на которых представлены в основном органы управления (**Controls**) и элементы индикации (**Indicators**). Из рисунка на примере панели Express видно, что инструментальные панели устроены иерархически, т.е. содержат инструменты на палитрах по категориям. Разместим на лицевой панели виртуального прибора органы управления величиной амплитуды и фазы некоторого гармонического сигнала. Для этого наведем указатель мыши на палитру **Numeric**

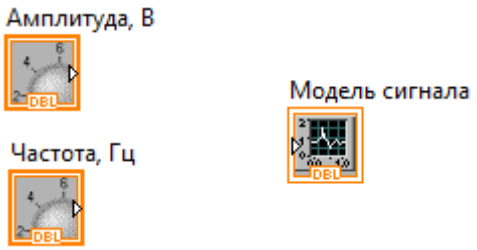
Controls панели Express, она раскроет своё содержимое, и выберем нажатием левой клавиши мыши (ЛКМ) элемент управления (регулятор) **Knob**. После этого вставляем его на нужное место лицевой панели щелчком ЛКМ (рисунок ниже), он будет задавать амплитуду сигнала.



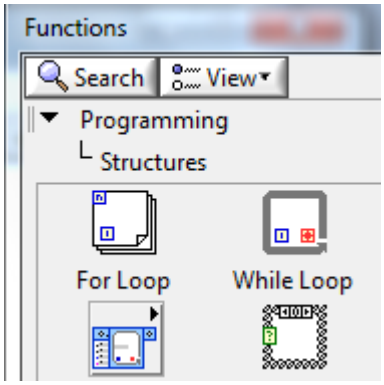
Щелчком ПКМ по элементу открывают контекстное меню и вызывают окно диалога для настройки его свойств (**Properties**). Затем аналогично создают элемент управления частотой сигнала и настраивают его свойства. Для графического представления сигнала на лицевой панели размещают элемент **Waveform Graph**, взятый из палитры **Express/Graph Indicators**. В результате описанных действий лицевая панель ВП приобретает вид, показанный ниже.

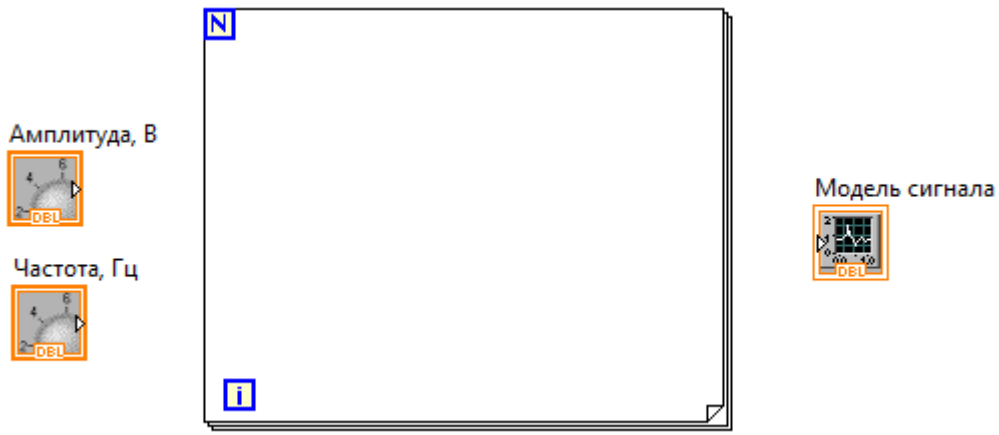


Среда пакета произвела размещение на блок-диаграмме ВП иконок так называемых терминалов, которые соответствуют элементам его лицевой панели (следующий рисунок). Иконки элементов ввода (управления) имеют жирную рамку, а иконки элементов вывода (индикации) – двойную. Кроме этого, на пиктограммах присутствуют надписи **DBL** (от слова **Double**), характеризующие тип данных действующих лиц (переменных) строящегося алгоритма (блок-диаграммы).



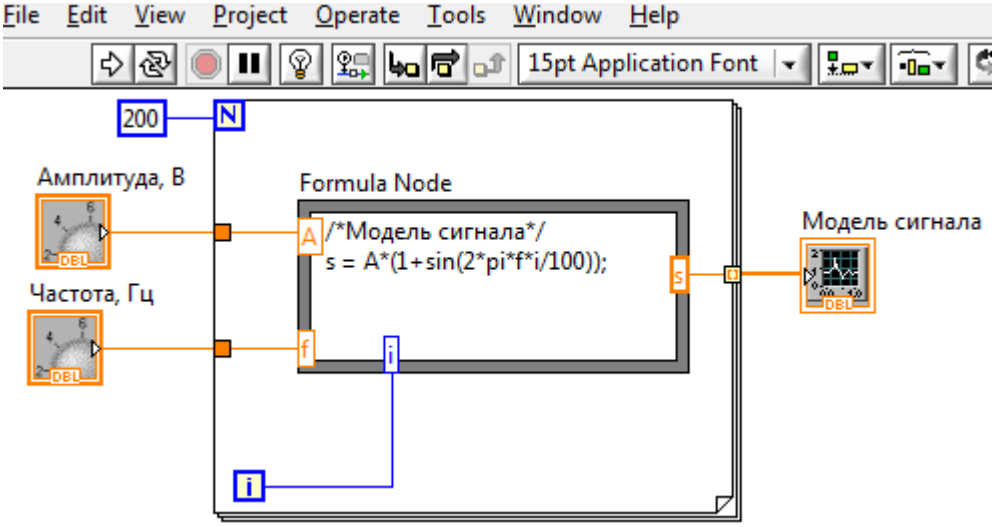
Для математического описания сигнала потребуется узел формул – **Formula Node**, а для получения отсчетов сигнала – цикл. Начнём с размещения на блок-диаграмме цикла с параметром – **For Loop**. Это делается в результате последовательных переходов в инструментальных средствах: панель **Functions** (функции) – палитра **Programming** (программирование) – подпалитра **Structures** (структуры) – цикл **For Loop** (следующий рисунок слева). Щёлкнув по структуре цикла, растягивают его при нажатой ЛКМ на блок-диаграмме между терминалами ввода (Амплитуда и Частота) слева и терминалом индикатора справа (рисунок ниже). Количество **N** повторений тела цикла задают числовой константой, терминал которой находят, пройдя путь: Program-





ming/Numeric/Numeric Constant. Терминал константы (синего цвета − так маркируются целые числа) размещают напротив терминала N в структуре цикла. После этого подводят указатель мыши к терминалу константы справа так, чтобы на терминале появился элемент вывода, а указатель превратился в катушку с соединительным проводом (**Wire**). Щёлкнув ЛКМ по элементу вывода и не отпуская нажатой ЛКМ, протаскивают мышью соединительную линию до элемента ввода на терминале N. Отпустив ЛКМ, убеждаются в создании требуемой связи. Двойной щелчок ЛКМ по созданной линии позволяет выделить её, а клавишей **Del** − удалить. Изгиб линий связи реализуют отпусканием ЛКМ в нужном месте и продолжением движения в выбранном направлении при нажатой ЛКМ.

Следующий шаг связан с размещением внутри цикла узла формул. Путь к нему проходит через **Mathematics/Scripts & Formulas**. Выбрав щелчком ЛКМ **Formula Node**, растягивают его до размера, позволяющего записать в нём нужные формулы. Внутри узла создают модель сигнала (рисунок ниже). Формульный узел содержит один или несколько операторов в Си-подобном формате, в том числе и комментарии



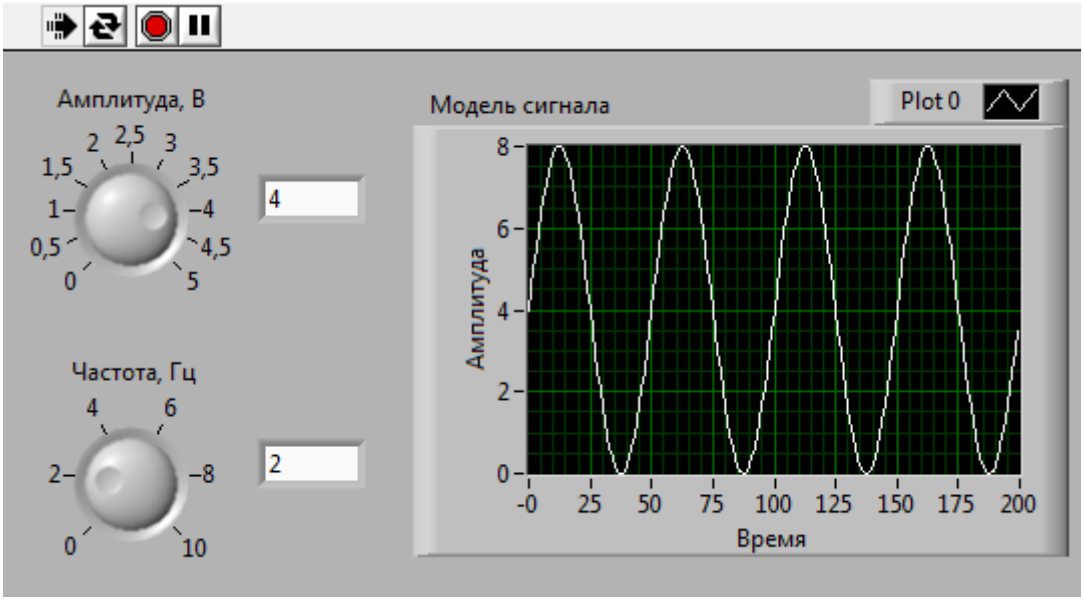
(/*comment*/). Для ввода в узел формул амплитуды **A** и частоты **f** сигнала командой **Add Input** (с этой целью производят щелчок ПКМ на границе Formula Node) создают соответствующие входы узла. Анало-

гично командой **Add Output** вызывают позицию для ввода имени **s** выходной переменной узла. После этого соединяют терминалы ввода амплитуды и частоты со входами узла формул. При этом на границе цикла автоматически создаются туннели (**Loop Tunnel**), представленные на диаграмме маленькими квадратами, имеющими цвет, соответствующий типу вводимых данных. При соединении выходной переменной **s** узла со входом терминала виртуального осциллографа также создается тунель (**Auto-Indexed Tunnel**). В цикле for *по умолчанию включена автоматическая индексация* элементов одномерного массива отсчётов сигнала s, формируемых на его выходе. Линия передачи массива на вход осциллографа жирная. Основополагающий для LabVIEW принцип *потока данных* (dataflow), согласно которому функции выполняются лишь тогда, когда они получают на вход необходимые данные, однозначно определяет порядок исполнения алгоритма [2]. Ниже в таблице представлены органы управления работой созданного ВП [3], а на следующем рисунке – работающий ВП.

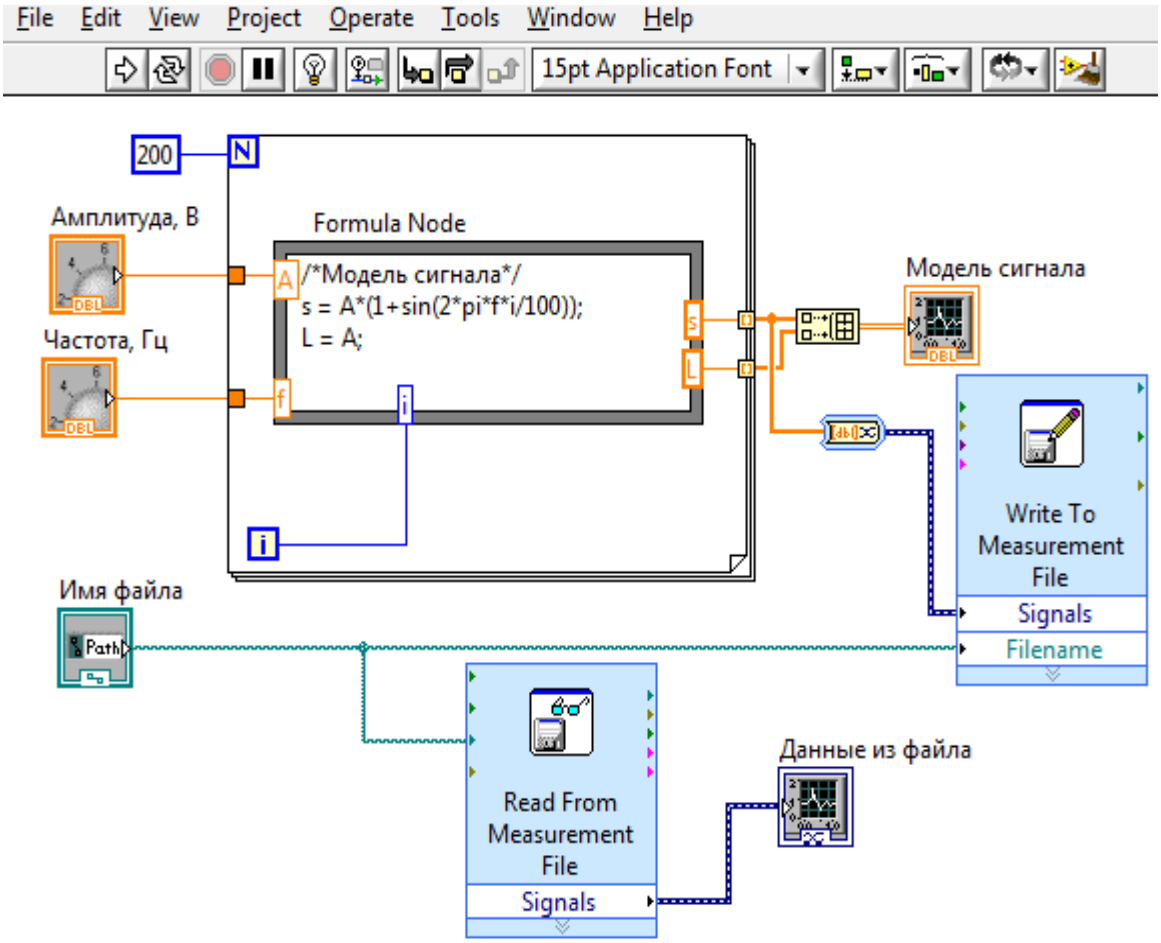
	Кнопка *Запуск* (Run) работоспособного ВП
	Кнопка *Непрерывный запуск* (Run Continuously) вызывает непрерывную работу ВП, остановить которую можно кнопкой *Прервать*
	Кнопка *Прервать* (Abort Execution) вызывает остановку выполняющегося ВП
	Кнопка *Пауза* (Pause) временно останавливает выполнение ВП

Дополним алгоритм созданного ВП некоторыми полезными действиями. Источниками данных в информационных системах часто являются файлы. Разместим на блок-диаграмме инструменты записи данных в файл и чтения их из него. С этой целью перейдём с помощью контекстного меню на палитру **Express/Output** (или палитру **Programming/File I/O**) и выберем соответствующие инструменты − **Write To Measurement File** и **Read From Measurement File** (рисунок с ними представлен вслед за рисунком работающего ВП). Инструменты

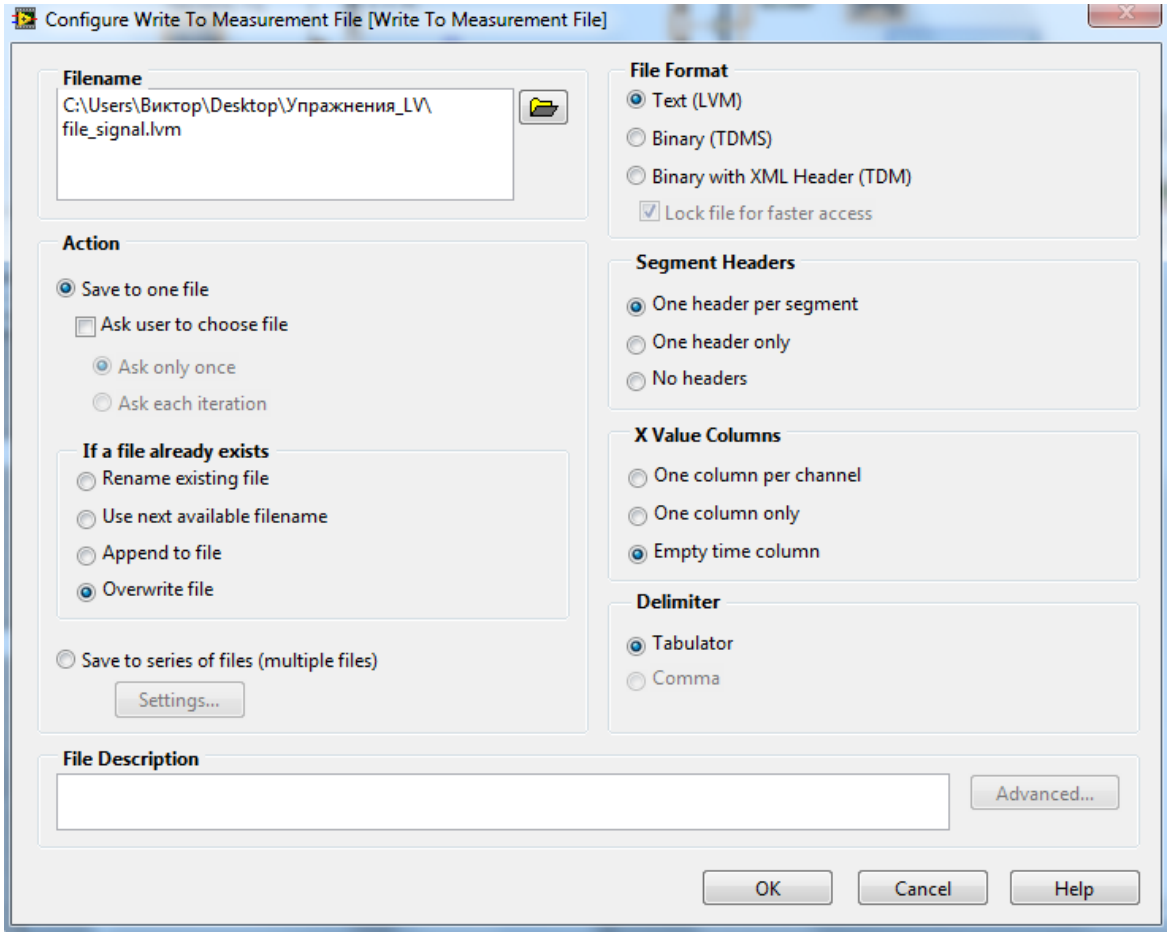
взаимодействуют с данными, полученными в результате измерений или программной генерации, как в нашем случае. С каждым из названных



инструментов связано окно конфигурирования, всплывающее через некоторое время после размещения компонента на диаграмме. В этом

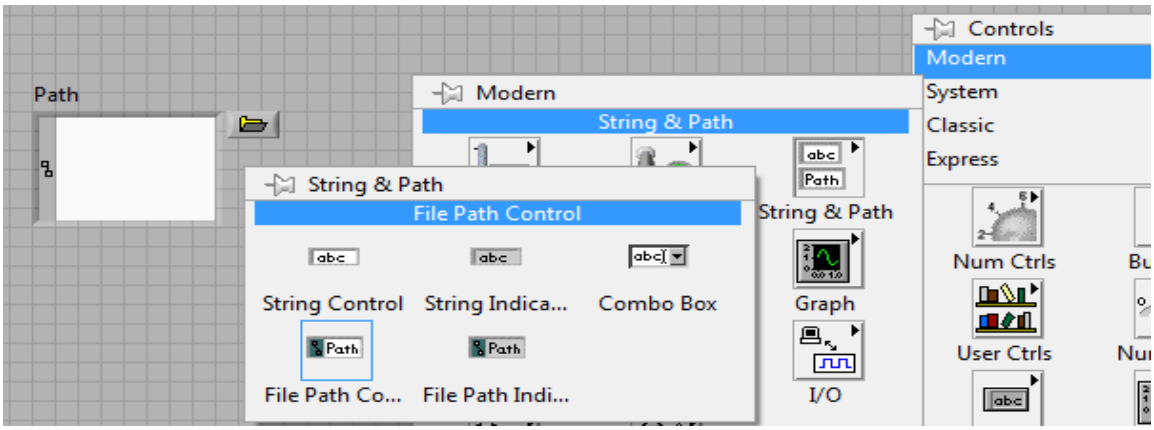


окне (следующий рисунок) в группе зависимых переключателей **File Format** выбирается формат создаваемого (читаемого) файла. Текстовые файлы (файлы по умолчанию, имеют расширение .lvm) допускают просмотр их содержимого в Блокноте, а также обработку в приложениях типа Microsoft Excel. Двоичные файлы (Binary, имеют расширение .tdms) имеют преимущество перед текстовыми по скорости их записи/чтения и по объёму занимаемого пространства на диске [1]. По умолчанию файл создаётся в папке **LabVIEW Data** под именем **test.lvm**.



Нужное месторасположение файла задают в окне диалога, вызываемого с помощью инструмента справа от окна ввода имени файла (**Filename**). Запись может осуществляться как в один файл, так и в несколько файлов в режимах переименования существующего файла (файлов) (**Rename existing file**), дозаписи в существующий файл (**Append to file**), перезаписи существующего файла (**Overwrite file**) или использования следующего доступного имени файла (**Use next available filename**). Для

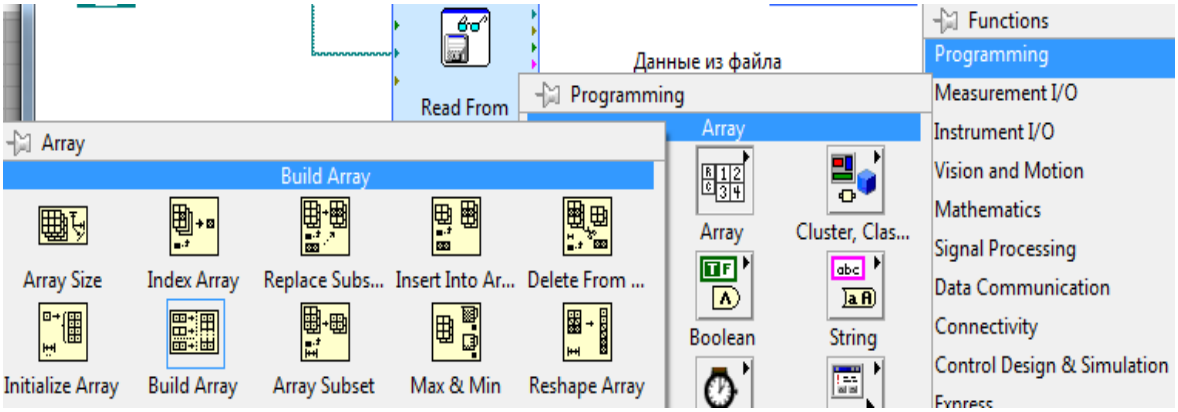
двоичных файлов доступны дополнительные настройки (**Advanced…**). Терминалы имеют определённое их назначением количество входов и выходов. Если терминалы растянуть мышкой по вертикали, то маленькие треугольники входов и выходов превратятся в соответствующие им полоски в составе терминалов с их обозначениями. Для ввода имени файла (Filename) на одноимённый вход терминала удобно использовать инструмент **File Path Control**, расположенный на палитре **String & Path** вкладки **Modern** (рисунок ниже), вызываемой с помощью ПКМ на лицевой панели ВП. Справа от многострочного редактора ввода инструмента находится иконка для открытия окна диалога выбора пути к файлу и задания его имени. Остаётся изменить данную по умолчанию метку (Path) компонента и произвести соединение на блок-диаграмме его выхода со входами Filename инструментов Write To Measurement File и Read From Measurement File, если приёмник и источник данных один.



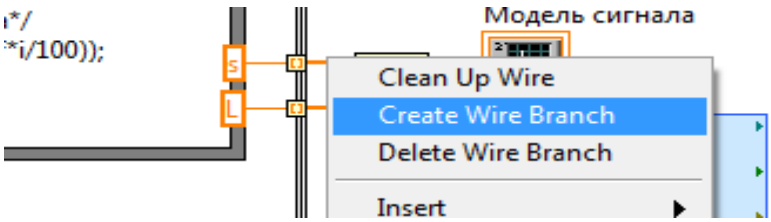
Для отображения содержимого файла необходимо разместить на лицевой панели ВП второй виртуальный осциллограф Waveform Graph, взятый из палитры Express/Graph Indicators. Вход осциллографа подключают к выходу **Signals** инструмента Read From Measurement File. Среда выбирает при этом характерный для передачи сигналов тип соединительной линии (динамический тип данных – Dynamic Data Type) .

Для отображения на первом виртуальном осциллографе центральной линии, относительно которой происходит изменение смоделированного синусоидального сигнала, вставим в узел формул её описание –

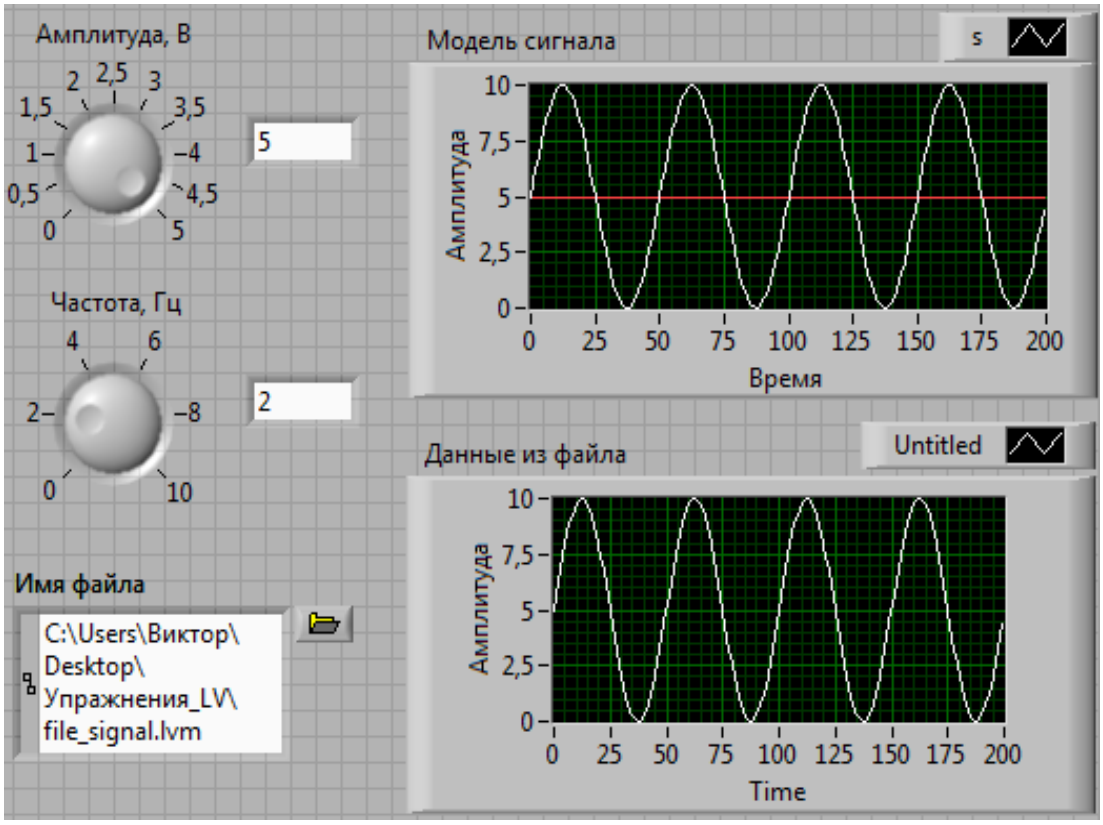
L = A и создадим соответствующий выход (L) для узла. Удаляем провод, связывающий туннель сигнала s со входом осциллографа. Чтобы можно было отображать два (или более) сигналов на одном осциллографе, разместим на блок-диаграмме элемент **Build Array**, взятый с панели Array, как показано на рисунке ниже. Элемент растягивают по вертикали мышью с целью получения необходимого числа входов, в нашем случае двух. На первый вход подают сигнал s, на второй – L. Выход элемента Build Array соединяют проводом со входом осциллографа.



Остаётся подключить сигнал s к входу Signals терминала инструмента Write To Measurement File. Для этого подводят указатель мыши к проводу, соединяющему выход сигнала из туннеля цикла со входом элемента Build Array, нажимают ПКМ и вызывают окно диалога, в котором выбирают ЛКМ команду **Create Wire Branch** (рисунок ниже). В результате на диаграмме появится пунктирная линия начала создаваемого соединения, которую следует довести до входа Signals, производя щелчки ЛКМ в местах её изгиба и завершения. Среда автоматически вставит в разрыв созданного провода конвертер массива отсчётов сигнала в данные динамического типа (Convert to Dynamic Data), который можно сконфигурировать в одноимённом окне диалога после двойного клика ЛКМ по конвертеру.



В результате описанных действий был создан ВП. На следующем рисунке показана его лицевая панель в режиме выполнения кода.

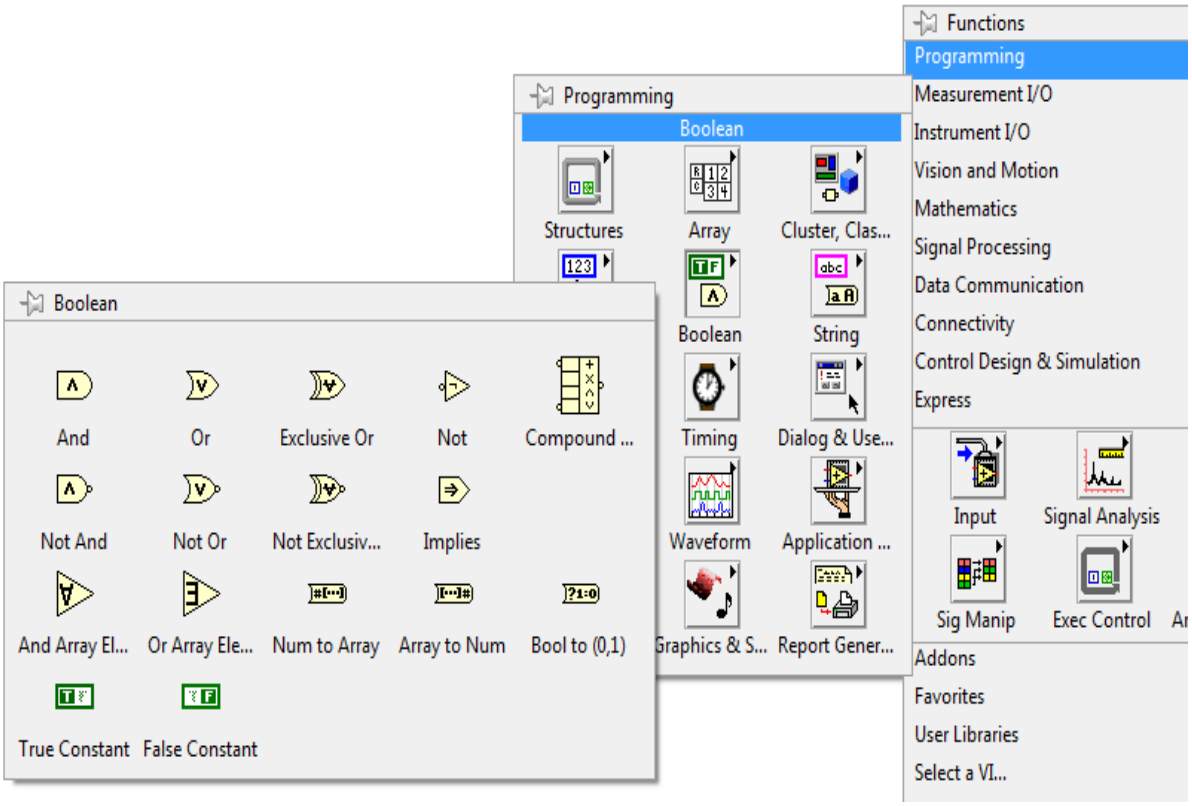


Первые шаги, предпринятые при разработке несложного ВП, показали, что язык G пакета LabVIEW – типизированный и предполагает работу как с простыми, так и со структурированными типами данных. Среда пакета, дружественно относящаяся к пользователю, автоматически выполняет некоторые преобразования типов, добавляя в блок-диаграмму соответствующие конверторы, в чём мы убедились ранее. Рассмотрим другие широко применяемые в ВП типы данных.

Логический тип данных и связанные с ним действия

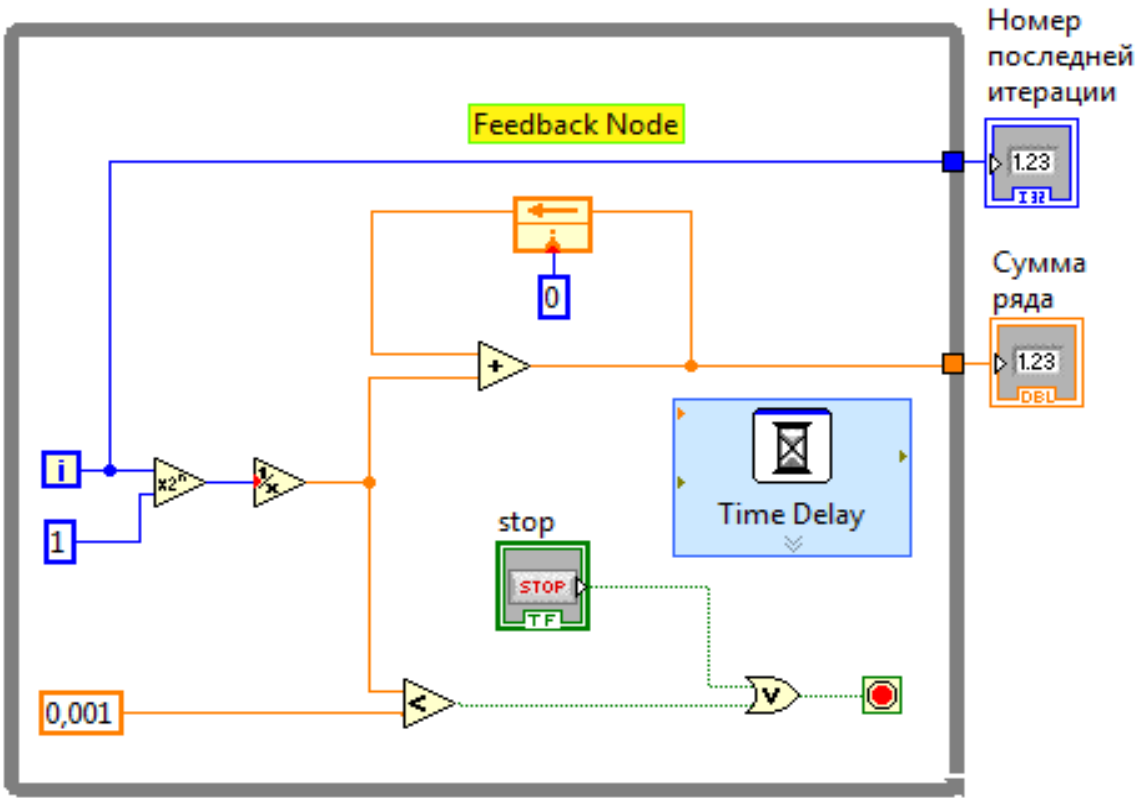
Логический тип Boolean имеет всего два значения: False и True, используемые для выполнения действий по некоторому условию. Набор реализованных в пакете логических операций гораздо шире, чем в популярных языках программирования высокого уровня. На следующем рисунке представлена палитра таких операций. Её можно вызвать, пройдя путь Programming/Boolean. Логическим данным соответствуют

терминалы и проводники зелёного цвета. Для ввода значений логического типа на передней панели размещают элементы управления, например с палитры Modern/Boolean. По умолчанию логические терминалы инициализируются значением False. Рассмотрим особенности взаи-

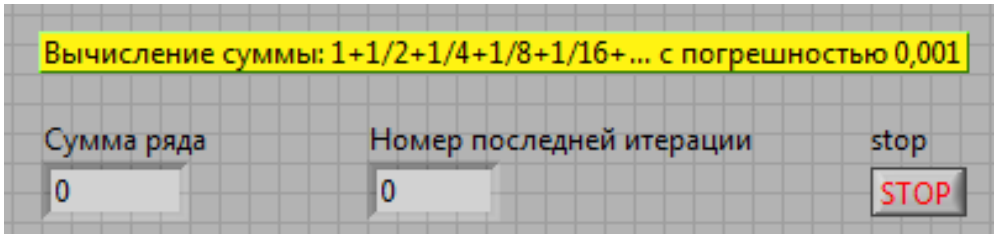


модействия с логическими данными, решая популярную математическую задачу, посвящённую вычислению суммы бесконечного ряда с заданной погрешностью. Разместим на блок-диаграмме цикл **While Loop** с панели Programming/Structures. Этот цикл управляется терминалом **Loop Condition**, расположенным в его правом нижнем углу. Этот терминал имеет два варианта представления – **Stop if True** (остановиться, если истина) и **Continue if True** (продолжить, если истина). В первом случае цикл While Loop будет выполняться до тех пор, пока на терминал не поступит значение True, а во втором – пока на терминал не поступит значение False. Если вход терминала условия не соединить с выходом элемента управления исполнением цикла, то ВП окажется в неисправном состоянии, а стрелка его запуска примет вид .

Разместим в центре структуры цикла, как показано ниже, элемент **Add** с палитры Express Numeric. Он реализует функцию добавления в сумму очередного слагаемого. Для получения слагаемого с номером i (нумерация итераций в цикле от нуля) размещаем элементы **Scale By Power Of 2** ($x2^n$) и **Reciprocal** (1/x). В качестве x подаём на первую функцию единицу, а показателем n степени двойки будет служить значение счётчика i итераций цикла, терминал которого для удобства размещения смещён вверх. Выход элемента 1/x соединяем с нижним входом сумматора, а выход сумматора – с его верхним входом. При этом среда автоматически вставит узел **Feedback Node** обратной связи (на рисунке над ним соответствующая надпись). Стрелка в узле показывает направление передачи данных – в нашем случае суммы предыдущих i-1 слагаемых. Узел инициализируется нулем (на **Initializer Terminal** узла подано значение исходной суммы), иначе на выход узла при запуске ВП проходит последнее записанное в него значение. Когда очередное слагаемое становится меньше погрешности 0,001, работа цикла и ВП останавливается. Покажем и другой способ останова программы.

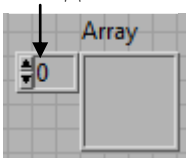


Расположим на лицевой панели кнопку **Stop** с палитры Express/Buttons & Switches, а на блок-диаграмме соединим её выходной терминал с одним входом элемента **Or**, на второй вход которого подадим логический сигнал с выхода элемента **Less?** *«Меньше»* (палитра Express/Ariphmetic & Comparison). Для туннелей, через которые выводится номер последнего учтённого в сумме слагаемого и сама сумма ряда, щелчком ПКМ по ним вызывают окно диалога и выбирают в нём команду Create/Indicator. В результате этих действий на схеме появятся терминалы соответствующих индикаторов. Добавим в схему также элемент **Time Delay** (Express/Exec Control) задержки времени, задав требуемый интервал, например 0,5 с, в окне диалога. Лицевая панель созданного ВП представлена на следующем рисунке.



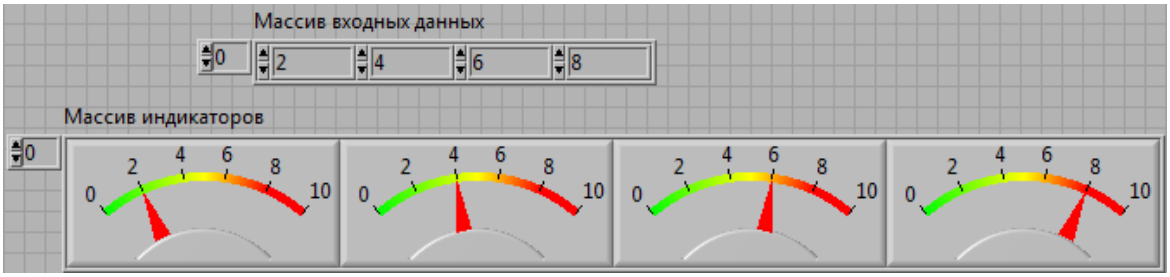
Массивы и кластеры

Массив состоит из элементов и описателей размерности. Массив может иметь одну или более размерностей, каждая из которых может содержать до 2^{31}-1 элементов. Элементы могут быть следующих типов: числовой (Numeric), булев (Boolean), путь (Path), строка (String), ссылка (Refnum) и кластер (Cluster). Нельзя создать массив из массивов. Однако можно создать массив кластеров, где каждый кластер содержит один или несколько массивов. Кластер – это совокупность логически связанных элементов разного типа, т.е. записи в терминах языка Pascal. Отсчёт индексов элементов массива ведётся от нуля.

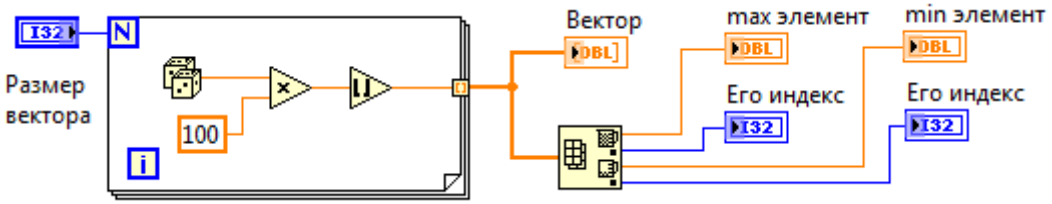


На лицевой панели размещают заготовку будущего массива – элемент **Array** (Controls/Classic/Array, Matrix & Cluster), показанный справа. По умолчанию массив одно-

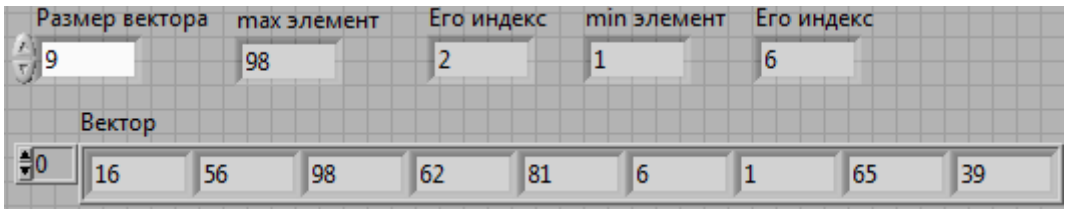
мерный. Размерность можно увеличить/уменьшить командами **Add Dimension/Remove Dimension** окна диалога, вызываемого щелчком ПКМ по дисплею индексов массива. Следующим шагом будет определение типа данных элементов массива. С этой целью с выбранной панели перетаскивают нужный элемент на поле массива. Ниже на рисунке показан одномерный массив входных числовых значений (2, 4, 6, 8) и связанный с ним проводником на блок-диаграмме массив индикаторов.



На следующем рисунке изображена блок-диаграмма ВП, создающего вектор из вводимого количества случайных чисел и выводящего максимальный и минимальный его элементы с их индексами. Элемент ввода *Размер вектора*, а также все индикаторы представлены своими терминалами (в их свойствах снят флажок View As Icon). Для определения максимального и минимального элементов вектора и их индексов используется функция Array Max & Min палитры Programming/Array.



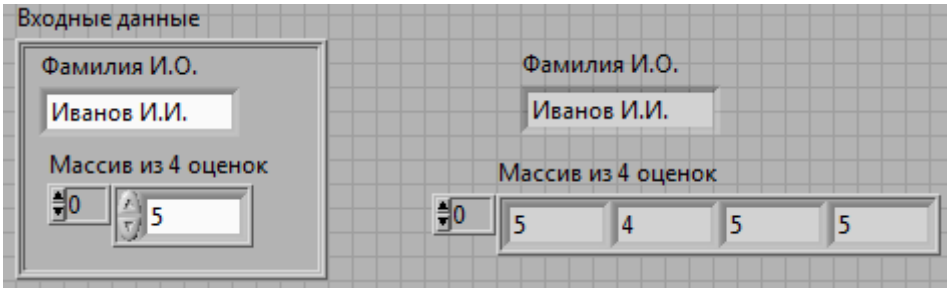
Ниже показан вид лицевой панели этого ВП.

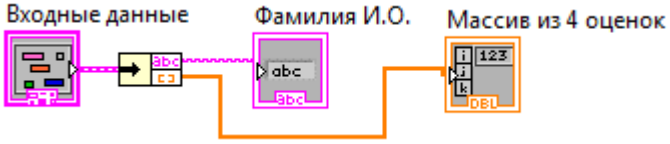


Аналогичным образом сначала создаётся заготовка для кластера, которая затем заполняется или элементами (типами) ввода (управления), или индикаторами (смешивать их в кластере недопустимо). Со-

единительная панель ВП может содержать максимум 28 терминалов. Если лицевая панель содержит больше 28 элементов управления и индикаторов, которые нужно использовать программно, то их следует сгруппировать в кластер и назначить этот кластер терминалу соединительной панели ВП. Элементы кластера имеют логический порядок независимо от их положения в оболочке кластера. Объект, помещенный в кластер первым, будет являться элементом 0, помещенный вторым – элементом 1 и т.д. При удалении элемента порядок автоматически корректируется. Упорядоченность кластера определяет порядок, в котором элементы появляются на терминалах функций Bundle и Unbundle на блок-диаграмме. Порядок элементов можно изменить пунктом **Reorder Controls In Cluster** контекстного меню [1].

Ниже приведена часть лицевой панели ВП, использующего кластер, содержащий поле с фамилией студента и поле, представляющее массив оценок. Справа от кластера показаны названные поля, извлечённые из него с помощью функции **Unbundle** (Programming/Cluster Class & Variant). На последнем рисунке представлена соответствующая блок-диаграмма ВП. Другие функции для работы с кластерами, в том числе для преобразования кластера в массив и наоборот находят на палитре, указанной выше.





Библиографический список

1. LabVIEW. Руководство пользователя/ пер. с англ. С.В. Николаева. – ni.com/Russia, 2007. − 370 с.

2. LabVIEW для всех/ Джеффри Тревис: пер. с англ. Н. А. Клушина. − М.: ДМК Пресс; ПриборКомплект, 2005. − 544 с.

3. http://labview-ifit.narod.ru/page12.html. Дата обращения 02.10.2017.