УДК
621.398
В-493

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

ВИНОГРАДОВА Н.А., ЛИСТРАТОВ Я.И. , СВИРИДОВ Е.В.

РАЗРАБОТКА ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В СРЕДЕ LabVIEW

Москва Издательство МЭИ 2005г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

ВИНОГРАДОВА Н.А., ЛИСТРАТОВ Я.И. , СВИРИДОВ Е.В.

РАЗРАБОТКА ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В СРЕДЕ LabVIEW

Учебное пособие по курсам
«Новые информационно-измерительные системы и технологии», «Автоматизированные системы научных исследований в теплофизическом эксперименте», «Технические средства автоматизации и управления»
для студентов, обучающихся по направлениям
«Техническая физика», «Автоматизация и управление»

УДК
621.398
В-493

Утверждено учебным управлением МЭИ в качестве учебного пособия для студентов

Подготовлено на кафедре инженерной теплофизики

Рецензенты: доктор технических наук Г.Ф.Филаретов
кандидат физико-математических наук В.И.Мика

Н.А.Виноградова, Я.И.Листратов, Е.В.Свиридов

Разработка прикладного программного обеспечения в среде LabVIEW: Учебное пособие – М.: Издательство МЭИ, 2005. – с.

ISBN –

Рассмотрена задача проектирования прикладного программного обеспечения для автоматизированных информационно-измерительных систем. Обосновывается эффективность использования стандартных средств разработки, позволяющих получить программный продукт, обеспечивающий выполнение всех основных функций автоматизированной системы, в том числе взаимодействие с измерительной и управляющей аппаратурой, и совместимость с программного обеспечения на разных уровнях. Изложены основные принципы и приемы программирования в рамках графической среды Lab View при выполнении основных этапов разработки.

Предназначено для студентов МЭИ (ТУ), изучающих дисциплины «Новые информационно-измерительные системы и технологии», «Автоматизированные системы научных исследований в теплофизическом эксперименте», «Технические средства автоматизации и управления».

ISBN -

Введение

Современные средства разработки прикладного программного обеспечения предоставляют широкий выбор инструментов, как для опытных программистов, так и для не искушенных в программировании пользователей. Эти средства позволяют создавать пользовательские программы непосредственно на стандартных языках программирования, например C/C++, Basic, а также с помощью специальных библиотек, являющихся основой ряда инструментальных программных средств. Пакеты для разработки прикладного программирования для систем автоматизации по своему основному назначению разделяются на две основные группы:

- пакеты программ LabVIEW, Measurement Studio, LabWindows/CVI, Agilent VEE и т.п. ориентированы, в основном, на использование в системах автоматизации лабораторного эксперимента и испытаний, хотя могут применяться и при создании других приложений, не связанных со взаимодействием с измерительно-управляющим оборудованием;
- пакеты LabVIEW/DSC, Lookout, InTouch, "Трейс Моуд" предназначены для создания прикладного программного обеспечения в автоматизированных системах управления технологическими процессами (АСУТП) и промышленной автоматики (системы SCADA-Supervisory Control And Data Acquisition).

По способу программирования эти пакеты делятся на следующие:

- текстовые или текстово-графические (Pascal, Delphi, LabWindows/CVI, Measurement Studio, Visual Basic, Visual C/C++), использующие элементы визуального текстового программирования для создания пользовательского интерфейса приложения и ориентированные в первую очередь на опытных программистов;
- графические объектно-ориентированные (InTouch, "Трейс Моуд"), основанные на применении графических образов объектов АСУТП в качестве элементов программирования;
- графические функционально-ориентированные (LabVIEW, LabVIEW/DSC, Agilent VEE), использующие функционально-логический принцип конструирования (рисования) и графического представления алгоритмов программ.

Графические пакеты легко осваиваются не только программистами-профессионалами, но и пользователями, не имеющими опыта программирования. С одной стороны современные графические системы позволяют создавать программы, практически не уступающие по

эффективности программам, написанным в текстовых пакетах. С другой стороны в большинстве случаев графические программы более наглядны, легче модифицируются и отлаживаются, быстрее разрабатываются. Несомненным достоинством графических систем программирования является то, что разработчиком приложения может быть сам постановщик задачи – инженер, технолог.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) позволяет разрабатывать прикладное программное обеспечение для организации взаимодействия с измерительной и управляющей аппаратурой, сбора, обработки и отображения информации и результатов расчетов, а также моделирования как отдельных объектов, так и автоматизированных систем в целом. Разработчиком LabVIEW является американская компания National Instruments [1].

В отличие от текстовых языков, таких как C, Pascal и др., где программы составляются в виде строк текста, в LabVIEW программы создаются в виде графических диаграмм, подобных обычным блок-схемам. Иногда можно создать приложение, вообще не прикасаясь к клавиатуре компьютера.

LabVIEW является открытой системой программирования и имеет встроенную поддержку всех применяемых в настоящее время программных интерфейсов, таких как Win32 DLL, COM, .NET, DDE, сетевых протоколов на базе IP, DataSocket и др. В состав LabVIEW входят библиотеки управления различными аппаратными средствами и интерфейсами, такими как PCI, CompactPCI/PXI, VME, VXI, GPIB (КОП), PLC, VISA, системами технического зрения и др. Программные продукты, созданные с использованием LabVIEW, могут быть дополнены фрагментами, разработанными на традиционных языках программирования, например C/C++, Pascal, Basic, FORTRAN. И наоборот можно использовать модули, разработанные в LabVIEW в проектах, создаваемых в других системах программирования. Таким образом, LabVIEW позволяет разрабатывать практически любые приложения, взаимодействующие с любыми видами аппаратных средств, поддерживаемых операционной системой компьютера. Используя технологию виртуальных приборов, разработчик может превратить стандартный персональный компьютер и набор произвольного контрольно-измерительного оборудования в многофункциональный измерительно-вычислительный комплекс.

Несомненным достоинством LabVIEW является то, что разработчику и пользователю доступны функционально идентичные системы программирования для различных операционных систем, таких как Microsoft Windows 95/98/NT/2000/XP, Linux, MacOS. Например программа,

разработанная под Windows будет почти без изменений работать на компьютере с Linux[1].

При установке соответствующих дополнительных модулей можно использовать LabVIEW как среду разработки программ для различных целевых систем и операционных систем (ОС):

- системы на базе ОС реального времени (ОСРВ) LabVIEW-RT;
- карманные компьютеры и другие устройства на базе ОС WindowsCE/PocketPC;
- карманные компьютеры и другие устройства на базе ОС PalmOS;
- многофункциональные программируемые устройства, например FPGA;
- сигнальные процессоры (DSP).

В основе технологии использования LabVIEW лежит комбинированное моделирование систем на ЭВМ, включающее аналитическое, имитационное и натурное.

Для аналитического моделирования характерно то, что алгоритм функционирования системы записывается в виде некоторых аналитических соотношений (алгебраических, интегродифференциальных, конечно-разностных и т.п.) или логических условий.

При имитационном моделировании алгоритм функционирования системы воспроизводится во времени с сохранением логической структуры и последовательности протекания элементарных явлений, составляющих процесс. В настоящее время имитационное моделирование — наиболее эффективный метод исследования систем, а часто и единственный практически доступный метод получения информации о поведении системы, особенно на этапе ее проектирования.

Натурным моделированием называют проведение исследования на реальном объекте с возможностью вмешательства человека в процесс проведения эксперимента и последующей обработки результатов эксперимента на вычислительной технике.

Отличие модельного эксперимента от реального заключается в том, что в модельном эксперименте могут быть реализованы любые ситуации, в том числе "невозможные" и аварийные, что в силу разных причин бывает недопустимо при работе с реальными объектами. Все представленные виды моделирования могут быть реализованы с использованием системы программирования LabVIEW. LabVIEW может успешно применяться в образовательных и научных целях, при промышленной автоматизации, в

[1] В случае использования ряда специфических особенностей различных ОС может потребоваться некоторая модификация исходного кода.

проектных и коммерческих структурах, связанных с тестированием и измерением каких-либо параметров, их анализом, визуализацией результатов, созданием баз данных, использованием компьютерных сетей.

Система LabVIEW включает в себя:

- ядро, обеспечивающее работоспособность программных процессов, разделение аппаратных ресурсов между процессами;
- компилятор графического языка программирования "G";
- интегрированную графическую среду разработки, выполнения и отладки программ;
- набор библиотек элементов программирования в LabVIEW, в том числе библиотеки графических элементов пользовательского интерфейса, библиотеки функций и подпрограмм, библиотеки драйверов, библиотеки программ для организации взаимодействия с измерительно-управляющими аппаратными средствами и т.п.;
- развитую справочную систему;
- обширный набор программ-примеров с возможностью как тематического, так и алфавитного поиска.

Программирование в системе LabVIEW максимально приближено к понятию алгоритм. После того, как вы продумаете алгоритм работы своей будущей программы, вам останется лишь нарисовать блок-схему этого алгоритма с использованием графического языка программирования "G". Вам не потребуется думать о ячейках памяти, адресах, портах ввода-вывода, прерываниях и иных атрибутах системного программирования. Данные будут передаваться от блока к блоку по "проводам", обрабатываться, отображаться, сохраняться в соответствии с вашим алгоритмом. Мало того, сам поток данных будет управлять ходом выполнения вашей программы. Ядро LabVIEW может автоматически использовать эффективные современные вычислительные возможности, такие как многозадачность, многопоточность и т.п.

Процесс программирования в LabVIEW похож на сборку какой-либо модели из конструктора. Программист формирует пользовательский интерфейс программы - "мышкой" выбирает из наглядных палитр-меню нужные элементы (кнопки, регуляторы, графики,..) и помещает их на рабочее поле программы. Аналогично "рисуется" алгоритм - из палитр-меню выбираются нужные подпрограммы, функции, конструкции программирования (циклы, условные конструкции и проч.). Затем также мышкой устанавливаются связи между элементами – создаются виртуальные провода, по которым данные будут следовать от источника к приемнику.

Если при программировании случайно будет сделана ошибка, например какой-то провод будет подключен "не туда", то в большинстве случаев LabVIEW сразу обратит на это внимание программиста. После того, как алгоритм – блок-схема нарисован, программа готова к работе.

Помимо библиотек, входящих в состав комплекта поставки системы LabVIEW, существует множество дополнительно разработанных программ. Многие из них свободно доступны через Internet. Собственные разработки пользователей, накопленные в процессе работы, могут размещаться в новых библиотеках и могут быть многократно использованы в дальнейшем.

Система программирования LabVIEW имеет встроенный механизм отладки приложений. В процессе отладки разработчик может назначать точки останова программы, выполнять программу "по шагам", визуализировать процесс исполнения программы и контролировать любые данные в любом месте программы.

Система LabVIEW позволяет защитить программы от несанкционированного изменения или просмотра их исходного кода. При этом разработчик может либо использовать пароли на доступ к приложениям, либо вовсе удалить исходный код из работающего приложения.

К сожалению, на момент написания данного Пособия не существует локализованной, "русской" версии системы программирования LabVIEW. Поэтому сейчас для эффективного использования LabVIEW разработчику понадобится знание основ технического иностранного языка. В то же время в создаваемых программах разработчик может использовать национальный алфавит без ограничений. Российское представительство National Instruments вместе с многочисленными пользователями ведет активную работу, целью которой является выпуск компанией National Instruments локализованной, русскоязычной версии LabVIEW.

1. Виртуальные приборы (VI – Virtual Instrument)[2]

Традиционные измерительные приборы не позволяют изменять их функциональные возможности, поэтому приходится закупать все приборы, которые необходимы для изучения какого-либо объекта. Технология виртуальных приборов позволяет превратить обычный персональный компьютер в устройство с произвольной функциональностью. Компьютер с подключенными к нему многофункциональными платами может быть и мощной расчетной машиной, и осциллографом, и вольтметром, и коммутатором сигналов, и частотомером, и системой управления технологическим процессом и т.п. Состав библиотек системы LabVIEW

[2] В скобках даются обозначения элементов на английском языке, принятые в LabVIEW.

позволяет в короткие сроки создавать необходимые инструменты для различных этапов исследований, начиная от элементарных приборов и заканчивая управляющими, информационно-поисковыми и аналитическими системами. Это дает основание говорить о принципиальных изменениях, которые вносит технология LabVIEW в создание прикладного программного обеспечения, поскольку эта система позволяет реализовать произвольный набор методов измерения, анализа, отображения и управления в автоматизированных системах различного профиля на базе обычного персонального компьютера [2].

Любая программа, созданная в системе LabVIEW, называется **виртуальный прибор** (ВП) или виртуальный инструмент (ВИ - дословный перевод с английского языка: VI-Virtual Instrument). Компонентами, составляющими ВП являются *передняя панель, блок-диаграмма и пиктограмма/коннектор*. Передняя панель реализует пользовательский интерфейс с ВП, позволяет задавать исходные данные и отображать результаты работы ВП. Блок-диаграмма является аналогом традиционной программы и реализует функциональные возможности ВП. Пиктограмма/коннектор позволяют использовать ВП в качестве подпрограммы (SubVI, виртуальный "подприбор") при построении модульных иерархических программ.

Только самые простые приложения разрабатываются в LabVIEW как один единственный ВП. Серьезные приложения представляют собой иерархию ВП (иногда более тысячи ВП). Такое иерархическое приложение можно разрабатывать методом "сверху вниз", когда исходная сложная, большая задача разбивается на несколько меньших подзадач. Те, в свою очередь, тоже разбиваются на подзадачи и т.д. В конце концов, при таком "дроблении" у разработчика будет набор элементарных задач, которые можно последовательно программировать, отлаживать и создавать из них основное приложение – ВП верхнего уровня. Чем подробнее продумана структура программы, чем лучше описана спецификация исходных данных и результатов работы, тем быстрее приложение будет создано, отлажено и внедрено.

Важно отметить, что система LabVIEW имеет возможность существенно облегчить разработку таких сложных приложений целым коллективом программистов, когда каждый программист реализует свою подзадачу. При этом LabVIEW обеспечивает корректность и актуальность версий различных ВП, отслеживает изменения исходного кода ВП, облегчает отладку ВП.

1.1. Передняя панель (Front Panel)

Передняя панель — это интерактивный интерфейс пользователя. Именно с передней панелью будет работать пользователь программы, поэтому она должна быть удобной, информативной и эргономичной.

На рис.1-а показана передняя панель двухканального виртуального осциллографа. На самом деле самостоятельного прибора – осциллографа нет. В данном случае используются встраиваемая в персональный компьютер многофункциональная измерительная плата и LabVIEW, превращающие компьютер в виртуальный осциллограф. Система LabVIEW предоставляет возможность исследователю работать с тем оборудованием, к которому он привык, даже если аппаратная реализация совсем иная.

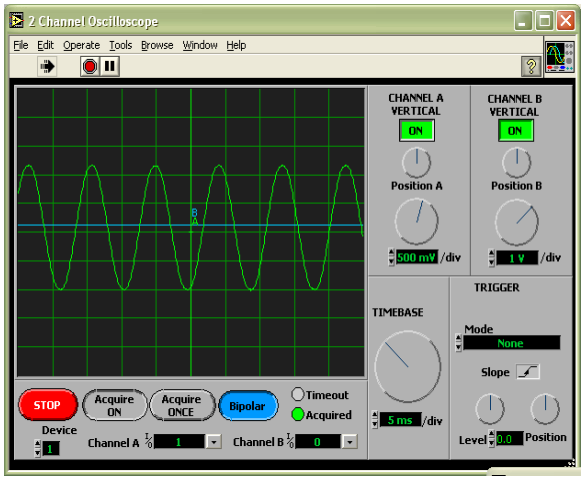

Рис. 1-а
Передняя панель виртуального осциллографа

Рис. 1-б
Передняя панель системы мониторинга и управления ветроэлектростанцией

На рис. 1-б изображена передняя панель ВП системы мониторинга и управления ветроэлектростанцией (Чукотка, г.Анадырь, разработка АОЗТ ЦАТИ), на ней отображаются необходимые параметры, полностью характеризующие работу объекта. Из основного меню ВП можно вызывать и другие панели для более детального мониторинга или управления станцией.

На рис. 2 показана передняя панель ВП, который циклически, от момента запуска до нажатия на кнопку "Стоп", генерирует случайное число от 0 до значения "Диапазон", отображает результат на графике и стрелочном индикаторе. В нижней части рис. 2 показаны некоторые палитры-меню, содержащие элементы пользовательского интерфейса: логические данные, числовые данные, графики. Эти палитры использовались программистом при разработке передней панели ВП [3].

[3] В зависимости от настроек LabVIEW эти палитры-меню могут автоматически отображаться при старте LabVIEW или быть скрыты.

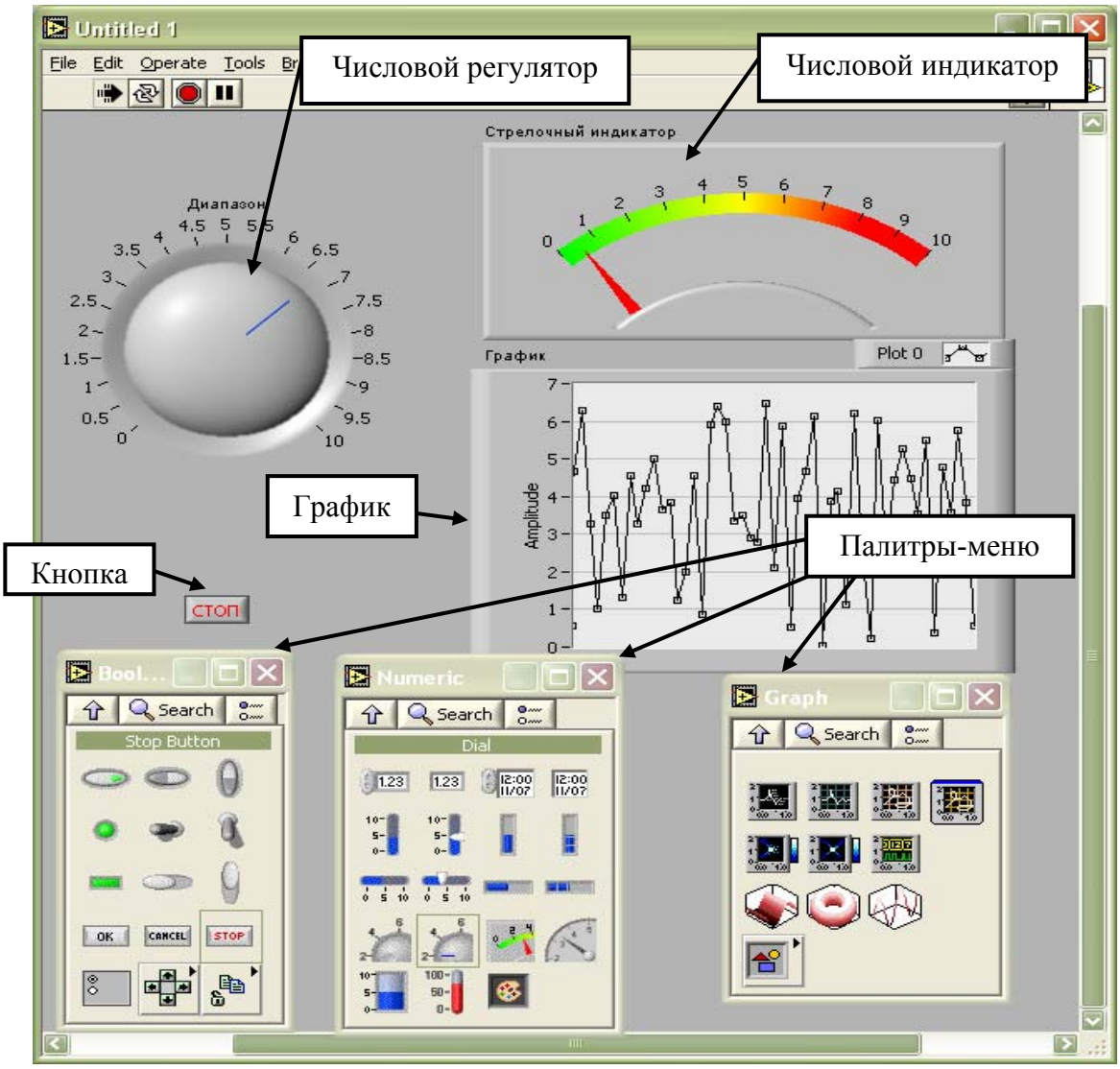

Рис 2. Передняя панель ВП генерации и отображения случайного числа.
Показаны некоторые палитры-меню,
используемые при разработке передней панели ВП.

Передняя панель может содержать необходимые кнопки, тумблеры, регуляторы числовых значений, графики, лампы, внедренные объекты ActiveX (Excel, Word и др.) и т.п. Большинство элементов передней панели могут работать в одном из двух режимов – **регулятор** (Control) или **индикатор** (Indicator). Регуляторы позволяют пользователю задать исходные данные для ВП, а индикаторы отображают результаты работы. При помещении объекта на экран передней панели ВП LabVIEW определяет режим работы, исходя из здравого смысла. Так, например, тумблер будет по умолчанию работать в режиме "регулятор", а термометр – в режиме "индикатор". С помощью меню свойств объекта, вызываемого щелчком

правой кнопки мыши, разработчик ВП может переключать режим работы, а также устанавливать иные свойства объекта [4].

1.2. Блок-диаграмма (Block Diagram)

Функциональные возможности ВП определяются его блок-диаграммой, которая является графической реализацией алгоритма, блок-схемы.

На рис.3 изображена блок-диаграмма ВП генерации и отображения случайного числа, а также представлены некоторые палитры-меню элементов для построения блок-диаграмм (арифметика, булевы операции, базовая математическая обработка данных).

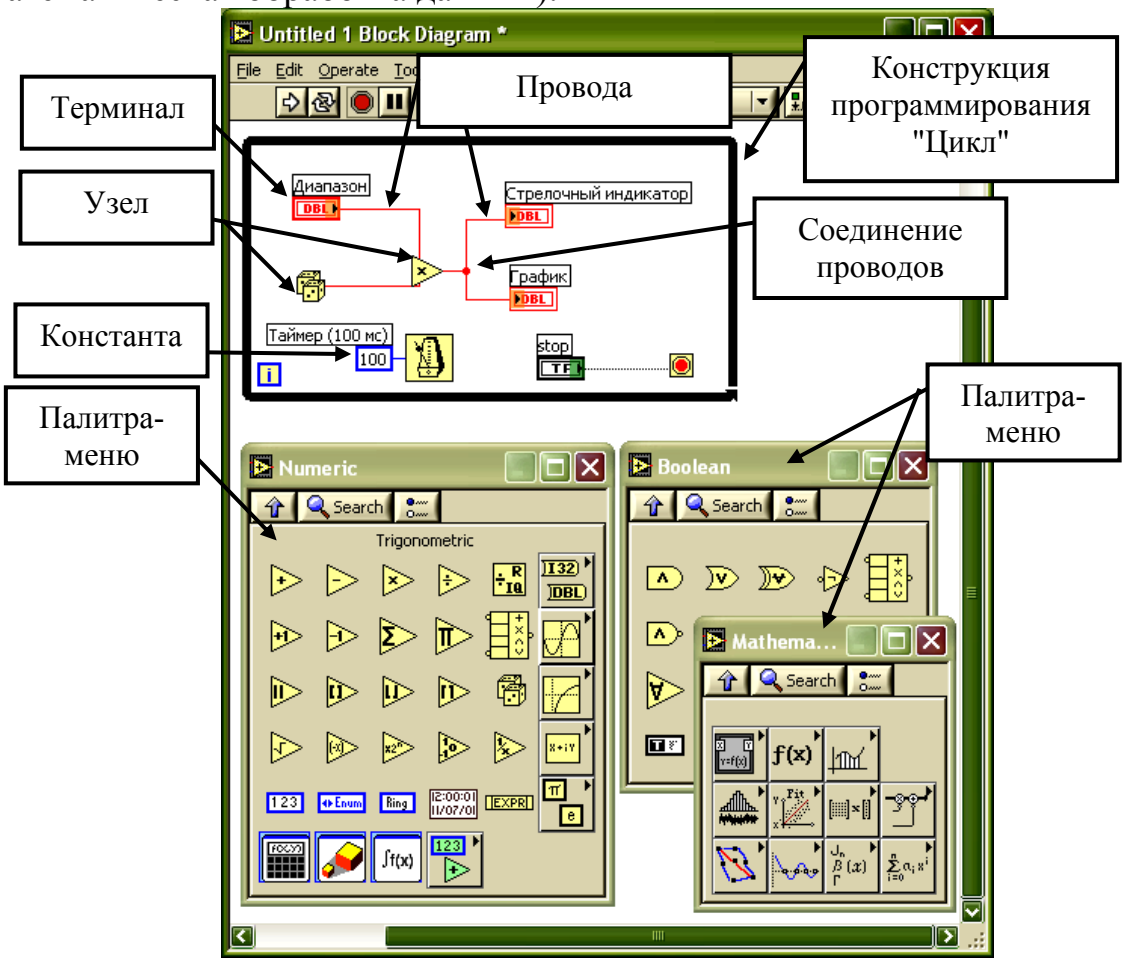

Рис 3. Блок-диаграмма ВП генерации и отображения случайного числа.

Блок-диаграмма состоит из терминалов, узлов, проводов и констант, а также может содержать необходимые текстовые или графические комментарии.

Блок-диаграмма любого ВП может быть произвольного размера и сложности. Однако считается, что если блок-диаграмма не умещается на

[4] Для разных объектов внешний вид всплывающих меню может отличаться. Пример такого меню для числового регулятора приведен на Рис. 9.

одном-двух экранах монитора при удобном для программиста разрешении графической системы, то либо алгоритм ВП недостаточно продуман, либо отдельные части кода такого ВП можно оформить как подпрограммы.

Терминал (Terminal)

Каждому элементу передней панели соответствует один терминал на блок-диаграмме. Терминалы создаются системой LabVIEW на блок-диаграмме автоматически, как только какой-либо элемент создается программистом на передней панели. В зависимости от настроек LabVIEW терминалы отображаются либо как пиктограммы, соответствующие элементам передней панели, либо как цветные прямоугольники разного вида.

Цвет и внешний вид терминала соответствует сопоставленному типу данных, а название (Label) терминала – названию элемента передней панели, см. Рис 4. Контекстное меню (правая кнопка мыши) позволяет быстро найти элемент передней панели, соответствующий выбранному терминалу.

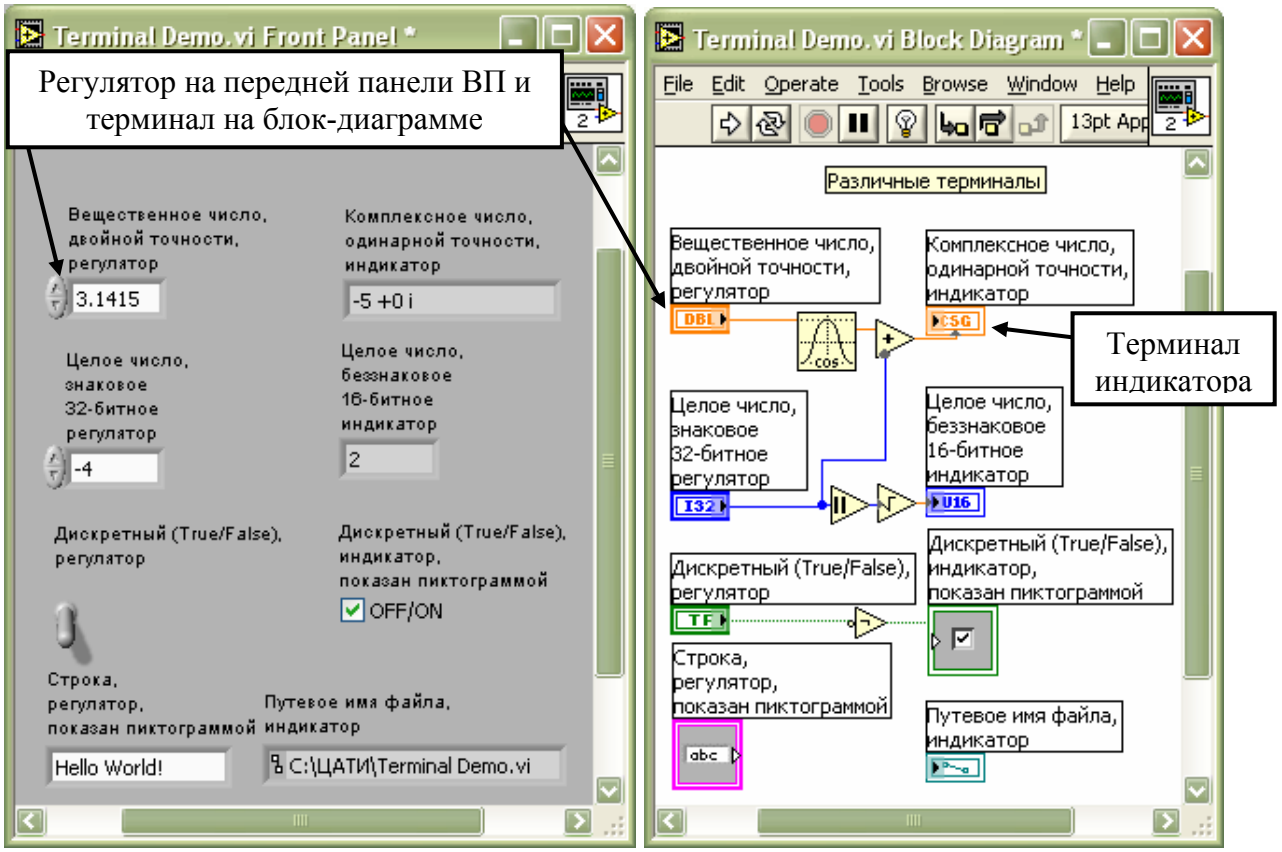

Рис 4. Передняя панель ВП с различными регуляторами и индикаторами и соответствующая блок-диаграмма.[5]

Терминал регулятора позволяет считать данные с передней панели и передать их в программу для дальнейшей обработки; он имеет рамку в виде толстой линии и маленькую стрелку справа, показывающую, что данные

[5] Изображены также некоторые узлы и провода, рассматриваемые ниже.

"выходят из терминала" в блок-диаграмму. Терминал индикатора позволяет отобразить результаты работы ВП на лицевой панели; он имеет тонкую рамку и стрелку слева – данные поступают из блок-диаграммы на терминал.

Узел (Node)

Узел – это аналог понятия "оператор" в текстовом языке программирования. Узлы – все то, что выполняется во время работы ВП: встроенные функции LabVIEW, подпрограммы (виртуальные "подприборы", SubVI). Узлы бывают простые (операторы z=x+y; a=cos(b)) и сложные (конструкции программирования такие, как условия (операторы if, switch, case of), циклы (операторы for, do-while) и т.п. Сложные узлы рассматриваются ниже в разделе *Конструкции программирования LabVIEW*. На Рис. 4 изображены пять простых узлов – расчет косинуса, сложение двух величин, модуль, квадратный корень и логическое отрицание.

Провод (Wire)

Провода – это разноцветные линии на блок-диаграмме, определяющие передачу данных от источника к приемнику во время работы ВП. На рис.4 данные следуют от терминала "Вещественное число двойной точности, регулятор" к узлу "косинус", от узла "модуль" к узлу "квадратный корень", от узла "сумма" к терминалу "Комплексное число, индикатор".

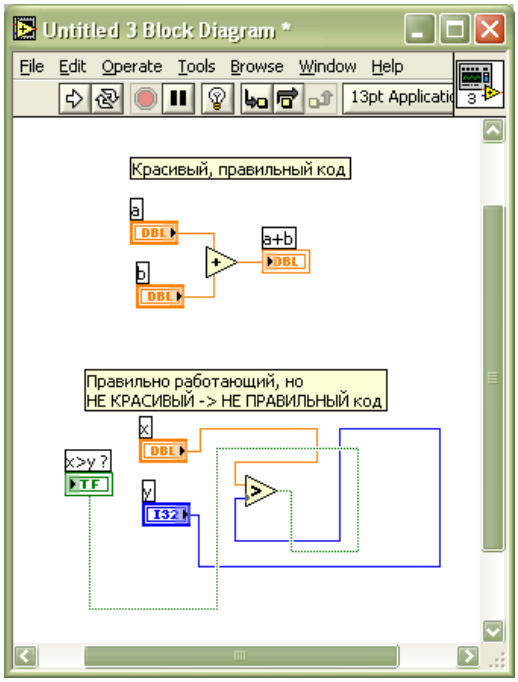

Рис 5. Провода, расположенные "правильно" и "неправильно".

Цвет и внешний вид провода соответствует типу данных, передаваемых по проводу. У любого провода должен быть единственный источник данных, и могут быть несколько приемников. Провод всегда должен быть присоединен к требуемому контакту *коннектора* узла или к *терминалу*, или к

константе или к другому проводу. В месте присоединения одного провода к другому отображается точка (если включен этот режим в меню настроек LabVIEW, меню Tools>>Options, вкладка Block Diagram, пункт "Show dots at wire junction"). Провод может иметь неограниченное число точек поворота, может быть любой длины – эффективность выполнения программы от этого не зависит. Однако программист должен стараться располагать терминалы, узлы и провода так, чтобы блок-диаграмма была наглядной, простой и красивой, смотри рис. 5. Только красиво нарисованная программа будет надежно работать, развиваться и модернизироваться!

1.3. Пиктограмма/коннектор (Icon/Connector)

Пиктограмма – компактное графическое изображение узла. Обычно при создании блок-диаграммы все узлы изображаются в виде пиктограмм.

Коннектор - определенная конфигурация контактов, позволяющих передать узлу исходные данные и получить результаты его работы. Коннектор узла можно отобразить с помощью всплывающего меню свойств узла.

На рис. 6 представлена блок-диаграмма ВП для расчета коэффициентов полинома, аппроксимирующего данные "массив X":"массив Y", по методу наименьших квадратов. Используется готовая подпрограмма, входящая в дистрибутив LabVIEW. В верхней части блок-диаграммы эта же подпрограмма изображена в виде коннектора.

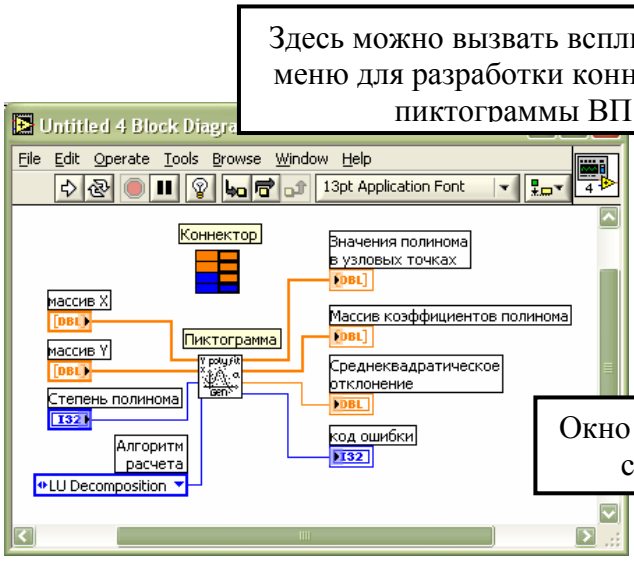

Рис. 6. Пиктограмма и коннектор узла.

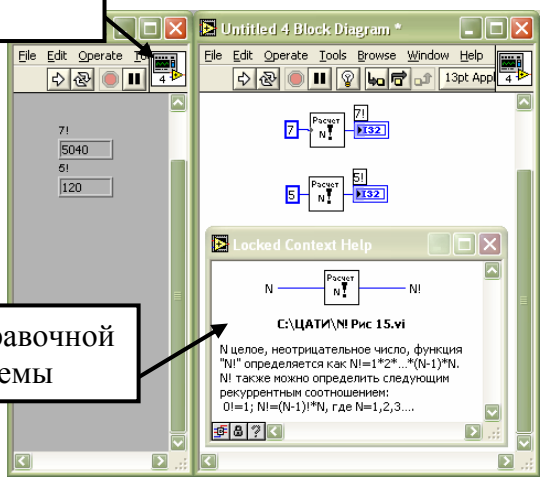

Рис 7. Использование подпрограммы расчета N!, отображение справочной информации о подпрограмме.

При подключении проводов к контактам коннектора LabVIEW производит проверку типов данных, а также подсказывает программисту, к какому именно контакту подключается провод. Поэтому практически невозможно ошибиться с подключением проводов к узлу, программисты редко используют просмотр коннектора.

Для всех виртуальных приборов, которые могут использоваться как подпрограммы (SubVI) [6], следует нарисовать пиктограмму и разработать коннектор. Для задания **всех** исходных данных и получения результатов работы ВП используются регуляторы и индикаторы на передней панели. Именно эти регуляторы и индикаторы можно поставить в соответствие нужным контактам коннектора. Для разработки пиктограммы и коннектора используется всплывающее меню, доступное по правой кнопке мыши на квадратном элементе пиктограмма/коннектор в **правом верхнем углу окна передней панели.** Из меню можно запустить графический редактор для создания пиктограммы, а также проводить необходимые манипуляции с коннектором (выбирать шаблон коннектора, поворачивать его и т.п.). Контакты коннектора ставятся в соответствие элементам передней панели с помощью инструмента "катушка с проводами".

1.4. Документирование ВП

Графический язык программирования "G", используемый в LabVIEW весьма нагляден, программа похожа на традиционную блок-схему алгоритма. Система LabVIEW позволяет разработчику ВП делать любые текстовые пояснения на передней панели или блок-диаграмме, а также записывать основную справочную информацию о виртуальном приборе (меню File>>VI Properties, вкладка Documentation). Кроме того, из всплывающего меню, можно указать поясняющую информацию для любого элемента передней панели. Все эти пояснения и комментарии будут восприняты встроенной справочной системой LabVIEW. Они будут отображаться в окне контекстно-чувствительной справки LabVIEW точно также, как и при использовании стандартных ВП, входящих в дистрибутив LabVIEW (рис 7). Кроме того LabVIEW сможет автоматически сгенерировать документацию на ВП с описанием самого виртуального прибора и всех входных и выходных параметров. Хорошие подробные комментарии позволят облегчить и существенно сократить сроки, требуемые для модернизации программного обеспечения.

[6] Для виртуальных приборов верхнего уровня нет необходимости в подготовке пиктограммы и коннектора, однако авторы рекомендуют сделать это – "делай хорошо-плохо само получится!".

2. Порядок выполнения ВП, технология Dataflow.

Русские пишут слева направо. Арабы пишут справа налево. Древнеегипетские жрецы писали слева направо, справа налево и сверху вниз, при этом направление письма и, соответственно, чтения определялось тем, в какую сторону повернуты головы животных и людей.

Текстовая программа выполняется в порядке следования операторов и в соответствие с тем, что определяют операторы типа goto. Порядок выполнения программы определяется в процессе ее кодирования. А как выполняется программа в LabVIEW!?

Основной технологией, определяющей выполнение виртуального прибора LabVIEW, является технология Dataflow, в соответствие с которой порядок выполнения программы определяет готовность потоков данных, проходящих от одного узла к другому. Общие правила таковы:

1. ни один узел не может выполниться до тех пор, пока на **все** контакты его коннектора, к которым подключены провода, не поступят данные.
2. если данные поступают на несколько узлов "одновременно", то и выполняются эти узлы "одновременно".

Разумеется, в случае однопроцессорного компьютера, несколько действий действительно одновременно выполняться не могут. Поэтому более широкая трактовка второго правила такова:

- Если данные поступают на несколько узлов "одновременно", то **порядок выполнения этих узлов не определен!**

В большинстве случаев при обработке данных технология Dataflow автоматически приводит к корректной последовательности выполнения узлов. Однако иногда требуется вполне определенный порядок действий, который нужно реализовать в момент кодирования программы.

Для примера рассмотрим задачу измерения величины напряжения постоянного тока цифровым вольтметром.

Предположим, что вольтметр подключен к компьютеру по приборному интерфейсу КОП (Канал Общего Пользования, другие названия – GPIB, HP-IB, IEEE-488). Пусть на вольтметре установлен адрес КОП равный 22, пусть команда, которую нужно передать на вольтметр для измерения напряжения постоянного тока – строка "MEAS:VOLT? DC"[7]. Пусть после измерения прибор выдает результат в виде строки, в которой содержится результат измерения, например "+1.2345E+00". Пусть длина этой строки не превышает 20 символов. Тогда алгоритм работы программы может быть такой:

1. передать на прибор с адресом 22 команду "MEAS:VOLT? DC";

[7] Для разных вольтметров эта команда будет разной. Например, в случае отечественного вольтметра В7-34 соответствующая команда будет такой: "F0R7T1M1E"

2. считать с прибора с адресом 22 результат измерений в виде строки, не более 20 символов;
3. преобразовать данные из строки в число с плавающей точкой, которое и является результатом измерений.

Рассмотрим два варианта реализации этого алгоритма на LabVIEW. Блок-диаграммы изображены на рис. 8-а и рис. 8-б.

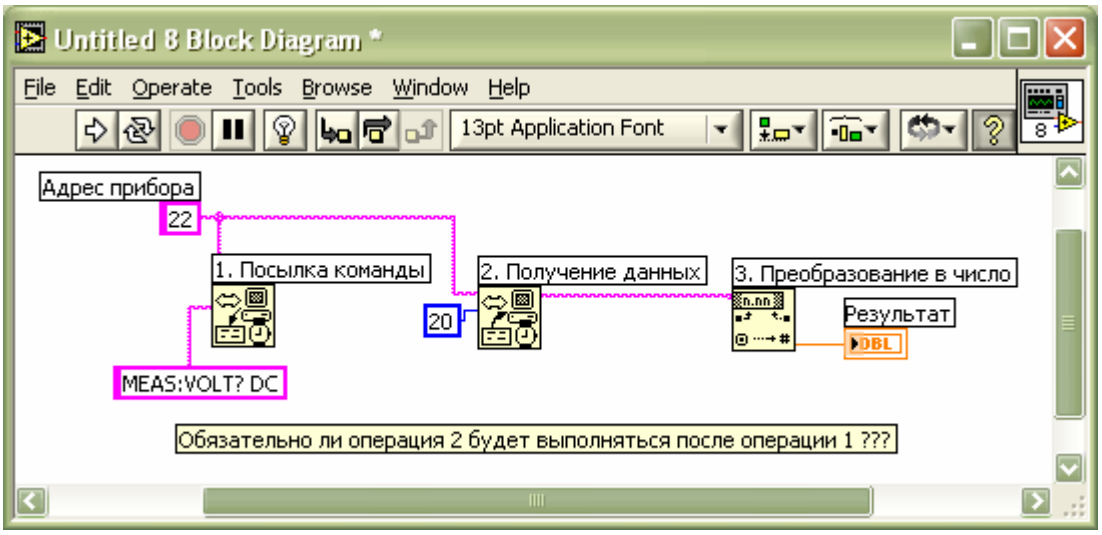

Рис 8-а. Измерение напряжения цифровым вольтметром. ВП может работать неверно (узел №2 может выполниться раньше узла №1).

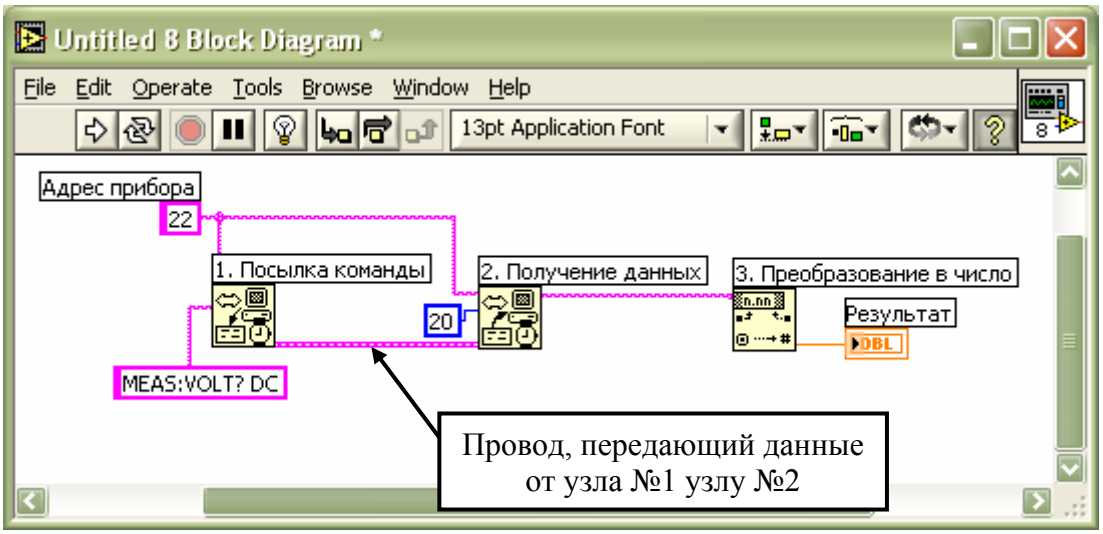

Рис 8-б. Измерение напряжения цифровым вольтметром. Технология Dataflow обеспечивает правильный порядок выполнения узлов!

Блок-диаграмма на рис. 8-а может работать неверно, хотя на первый взгляд она полностью соответствует алгоритму. Разработчик программы может стать жертвой привычки "писать слева направо". Действительно все исходные данные, необходимые для выполнения узлов 1 и 2 (Посылка команды и Получение данных), а именно адрес прибора, команда и количество байт для получения с вольтметра готовы одновременно – заданы

в виде соответствующих констант. Таким образом, порядок выполнения этих узлов не определен - возможно ВП выполнится правильно, а может быть и нет!

На блок-диаграмме на рис. 8-б есть провод, по которому узел №1 "Посылка команды" передает узлу №2 "Получение данных" информацию о коде ошибки (error out-error in). В соответствие с правилами Dataflow выполнение виртуального прибора всегда будет проходить корректно, в нужной последовательности!

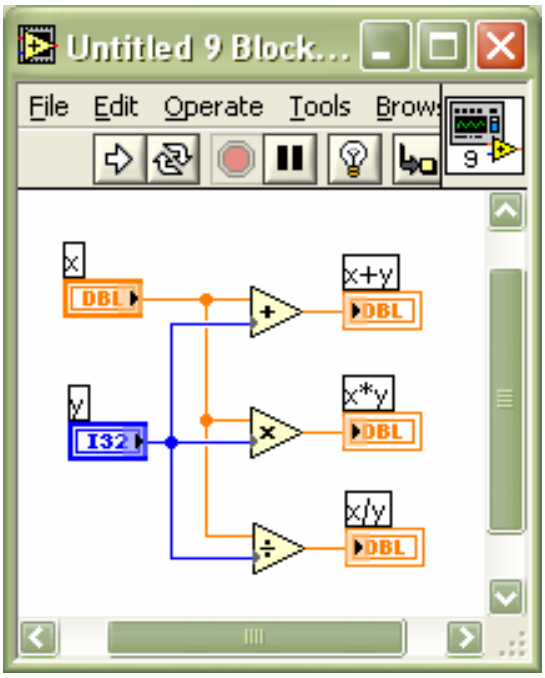

Рис 8-в. Все узлы выполняются "одновременно".

На рис. 8-в приведена блок-диаграмма другого ВП, где порядок выполнения узлов не важен, можно считать, что все три узла выполняются одновременно.

В случае невозможности или нежелательности "проводного" определения порядка выполнения ВП применяется конструкция программирования *последовательность* (Sequence), которая будет рассмотрена ниже.

3. Типы данных в LabVIEW

В LabVIEW используются разнообразные типы данных. Некоторые типы данных соответствуют обычным текстовым системам программирования (целое число, логические данные, строка и пр.). Другие типы данных реализованы только в LabVIEW и предназначены для более надежной и удобной работы в составе автоматизированной системы научных исследований. LabVIEW работает с такими типами данных, как осциллограмма (Waveform), сигнал (Signal), ресурс VISA, измерительный или управляющий канал и т.п. Цветом и внешним видом терминалов и

проводов LabVIEW подсказывает разработчику ВП, как и какие данные обрабатываются блок-диаграммой. При подключении проводов, поэтому работа со всем многообразием типов данных в LabVIEW весьма удобна.

3.1. Простые скалярные типы данных

К числу простых скалярных типов данных относятся:

- вещественные числа (оранжевые терминалы и провода) повышенной точности, двойной точности, одинарной точности – в LabVIEW обозначаются соответственно EXT, DBL, SGL;
- целые числа со знаком (синие терминалы и провода) 32-,16- и 8-разрядные - соответственно I32, I16, I8;
- неотрицательные целые числа без знака (синие терминалы и провода) 32-,16- и 8-разрядные - соответственно U32, U16, U8;
- комплексные числа (оранжевые терминалы и провода) повышенной точности, двойной точности, одинарной точности – соответственно CXT, CBD, CSG

На рис. 9 изображено всплывающее меню свойств числового регулятора. Из этого меню можно выбрать нужный тип данных.

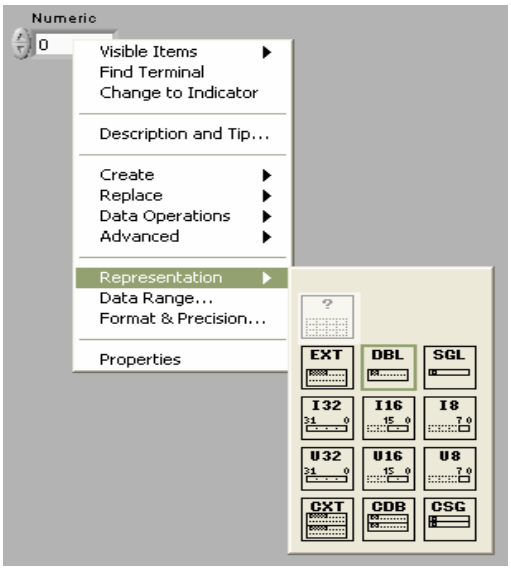

Рис 9. Типы числовых данных, всплывающее меню свойств элемента передней панели (правая кнопка мыши)

- логические данные (зеленые терминалы и провода), имеют одно из двух возможных значений – True/False;
- строковые данные (розовые терминалы и провода) [8].

[8] Строковые данные всегда розовые, но некоторые другие типы данных также могут использовать розовый цвет, например кластеры, содержащие данные определенных типов.

3.2. Кластер (cluster)

Кластер – конечный набор данных различных типов. Этот тип данных соответствует типам struct в C/C++ или record в Pascal. Графически кластер выглядит как прямоугольная область, внутри которой находятся разнотипные данные. Образный пример кластера – плащ с разными карманами. В одном кармане монетка, в другом – связка ключей, в третьем бумажник, в четвертом – телефон. Как правило все элементы кластера имеют отличительные названия - имена (Label).

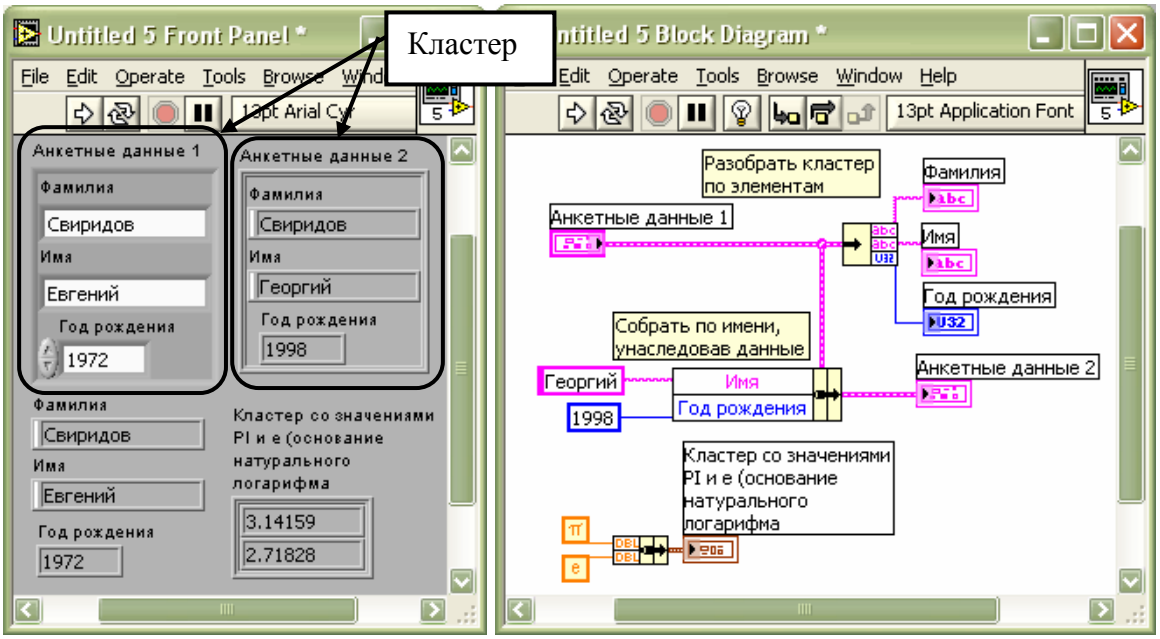

Рис. 10. Пример работы с кластерами (разобрать кластер анкетных данных по элементам, собрать по имени новый кластер, унаследовав элементы и изменив только "Имя" и "Год рождения", собрать кластер из двух математических констант).

Приведенный ниже код на языке C описывает два разных несовместимых типа данных, поскольку элементы структур "a" и "b" расположены в разном порядке, хотя набор данных одинаков [9]:

```
struct {
    int a;
    double b;
    char c;
}
```

```
struct {
    double b;
    int a;
    char c;
}
```

[9] В языке Pascal для определения аналогичного типа данных используется ключевое слово **record**

Так и в LabVIEW важен порядок следования элементов в кластере. Графическое представление кластера может быть обманчиво, поскольку порядок следования элементов в кластере определяется не их взаимным расположением. Первоначально этот порядок соответствует порядку помещения различных данных внутрь кластера. В последствие этот порядок может быть изменен через всплывающее меню свойств объекта.

Для работы с кластерами в LabVIEW есть соответствующая палитра функций. Функции позволяют разобрать кластер на отдельные элементы (Unbundle, Unbundle by name), или собрать различные данные в кластер (Bundle, Bundle by name). При этом в случае сборки одного кластера можно использовать "наследование" типов и значений данных из другого кластера. Это позволяет, во-первых, избежать ошибок с несовпадением типов, во-вторых, изменить значения только требуемых составляющих кластера, унаследовав другие (Рис 10). Обратите внимание, что в этом примере терминалы и провода кластеров анкетных данных розовые, как и у строковых данных, хотя внешний вид терминалов и проводов отличается.

3.3. Массив (array)

Массив – пронумерованный, непрерывный, неограниченный[10] набор однотипных данных. Каждый элемент массива имеет набор индексов, соответствующий размерности массива: одномерный – 1 индекс, двумерный – 2 индекса и т.д.

Графически массив выглядит как прямоугольная область, через которую можно просматривать элементы массива. Рядом с левым верхним углом этой области отображаются индексы. Значения этих индексов соответствуют элементу массива, показанному в левом верхнем углу. Одномерный массив (вектор) – строка или столбец. Двумерный массив (матрица, таблица) – таблица из нескольких строк и нескольких столбцов. Массивы больших размерностей на плоскости экрана монитора отобразить невозможно, поэтому они выглядят как таблицы, представляющие собой срез по определенным индексам. Массив может содержать данные произвольного типа. Например, может быть массив тумблеров (дискретные регуляторы), или массив целых чисел, или массив кластеров. Для изменения размерности массива можно использовать всплывающее меню свойств индекса(ов) массива.

В LabVIEW элементы массива нумеруются по строкам от нуля. Таким образом элемент двумерного массива с индексами [2;4] находится в третьей строке и пятом столбце.Массив всегда непрерывный набор данных, без пропусков. Это значит, что если в массиве есть элемент с индексом

[10] Формально ограничение одно – индекс (номер) элемента массива – 32-битное число. Поэтому в каждой размерности может быть не более 2^{32} элементов, что на практике равносильно понятию неограниченный.

любой размерности "M", то есть и все элементы с индексами "i" этой размерности, такими, что "0≤i<M". Массив, не содержащий ни одного элемента, называется пустым.

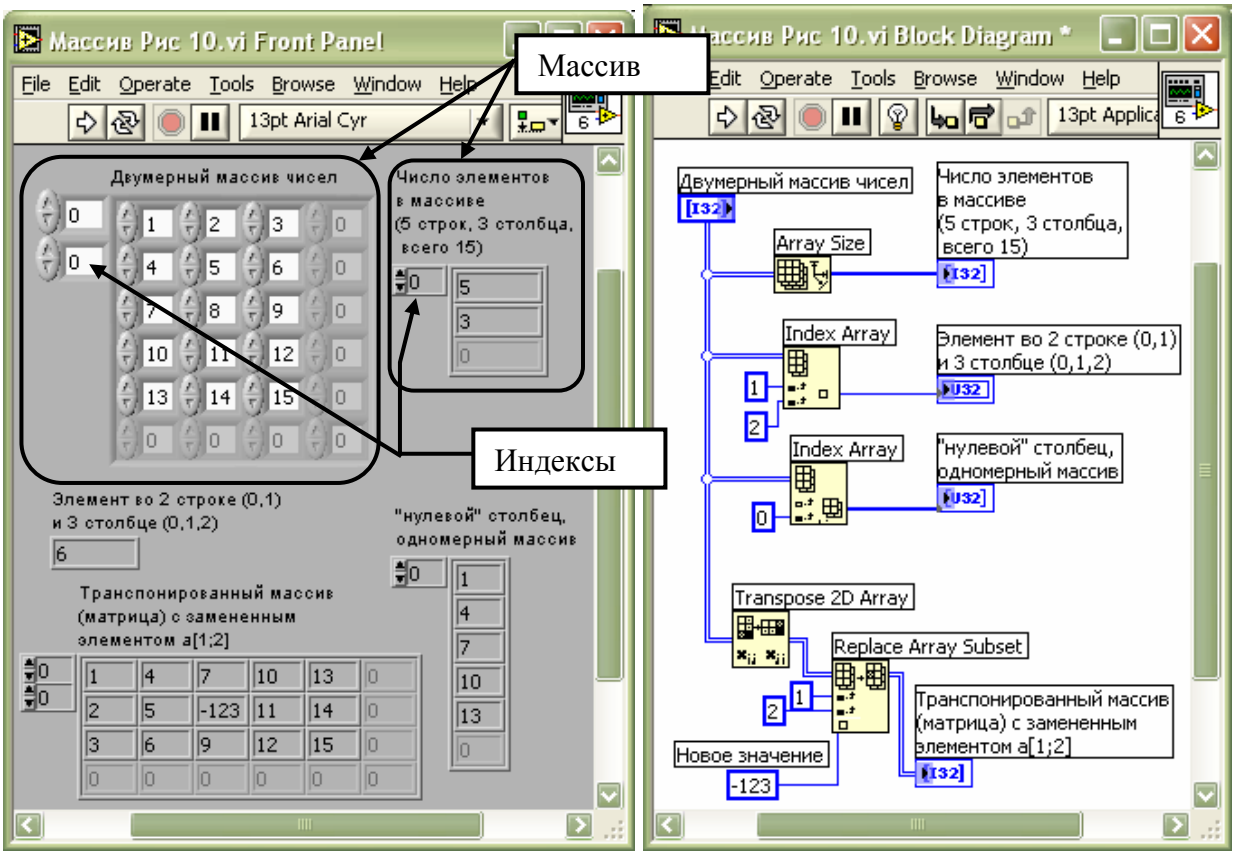

Рис. 11 Пример работы с массивами (определение размера массива, получение значения нужного элемента, получение нужной части многомерного массива, транспонирование двумерного массива, замена элемента массива)

Очень важно отметить, что графическое отображение массива не дает информации о том, сколько элементов содержит массив, т.е. сколько элементов определено. Прямоугольная область, отображающая массив является лишь "просмотровым окном". Подходящий образный пример – бездонный вещмешок или рюкзак с одинаковыми предметами внутри. Мы можем, ослабив тесемки горловины, сделать просмотровое окно шире, или, затянув тесемки, сделать область просмотра уже. Мы можем перемещать горловину мешка влево и вправо, рассматривая нужные предметы.

Все элементы массива имеют один и тот же тип данных, причем в широком смысле. Это значит, что одинаковы как и собственно типы данных, так и их графическое изображение, цвета, размеры графических образов каждого элемента.

В палитре функций есть все необходимые средства для работы с массивами, например определение количества элементов в массиве,

получение элемента по индексу (индексам), сортировка массива, удаление элементов или изменение значений элементов массива. Большинство функций для работы с массивами являются полиморфными (см. ниже) и автоматически подстраиваются под конкретный вариант массива и требуемую задачу. Обратите внимание на разные способы использования и результаты работы "одной и той же" полиморфной функции Index Array. Пример работы с массивом представлен на Рис. 11.

3.4. Другие типы данных

Наряду с традиционными типами данных, рассмотренными ранее, LabVIEW поддерживает работу с рядом типов данных, облегчающих работу в составе автоматизированных систем научных исследований. В рамках данного пособия авторы не имеют возможности подробно рассмотреть каждый тип. Мы ограничимся краткой информацией о некоторых из них.

- Осциллограмма (Waveform) используется для работы с измеренными или программно генерированными данными в случае постоянства периода дискретизации. При этом тип данных "Осциллограмма" включает в себя три компонента – t0 – время начала осциллограммы, dt – период дискретизации, [y] – массив значений. В ранних версиях LabVIEW такого типа данных не было, пользователи создавали кластеры соответствующей структуры. Этот тип данных удобен при отображении на графиках (автоматическая настройка оси абсцисс), для спектральной, корреляционной обработки данных, цифровой фильтрации, получении фазовых или частотных характеристик сигналов.
- Сигнал (Signal) – дальнейшая, по сравнению с осциллограммой, модификация типа данных для работы с измеренными или генерируемыми параметрами. Инкапсулирует названия параметров, режимы измерений или генерации, может содержать в себе несколько параметров [11].
- Сессия VISA (VISA session, VISA resource name) используется при программировании внешних устройств в соответствие со стандартом VISA. Драйверы большинства современных приборов и устройств разрабатываются в соответствие со спецификацией VISA и технологией Plug&Play. Наиболее близкий "родственник" из обычных типов данных и в некоторых случаях заменитель – строка.
- Измерительный или управляющий канал DAQ (DAQ Channel) – включает в себя настройки режимов измерения или управления какого-либо канала многофункциональной измерительно-управляющей платы National Instruments, а также данные для первичной математической обработки.

[11] Многие ExpressVI, входящие в состав LabVIEW 7.0 и более поздних версий используют этот тип данных в качестве исходных данных или результатов работы.

- Ссылка (refnum) – тип данных, аналогичный указателю (pointer) или дескриптору какого-либо объекта в языках текстового программирования. В LabVIEW возможно использование различных объектов и ссылок на них, например ссылка на объект-элемент пользовательского интерфейса, ссылка на внедренный ActiveX объект, ссылка на какой-либо виртуальный прибор и т.п. Использование ссылок позволяет реализовать богатейшие возможности LabVIEW при работе с различными объектами.

3.5. Полиморфизм

Полиморфизм – исключительно удобное свойство многих функций LabVIEW, позволяющее проводить различные операции с данными разных типов с помощью одних и тех же функций. Например, функция "сложение" может сложить два произвольных числа, она же может сложить поэлементно два массива, она же может прибавить к каждому элементу массива какое-либо число. Таким образом, полиморфные функции автоматически могут подстроиться к исходным данным, производя различные действия и получая различные результаты.

Приведем пример одной и той же программы на "C" и на LabVIEW – Рис 12. Эта программа прибавляет число 123 ко всем элементам одномерного массива.

```
void sum_array(void) {
        int i, a[5]={1,2,3,4,5};
        for (i=0;i<5;i=i+1)
                a[i]=a[i]+123;
}
```

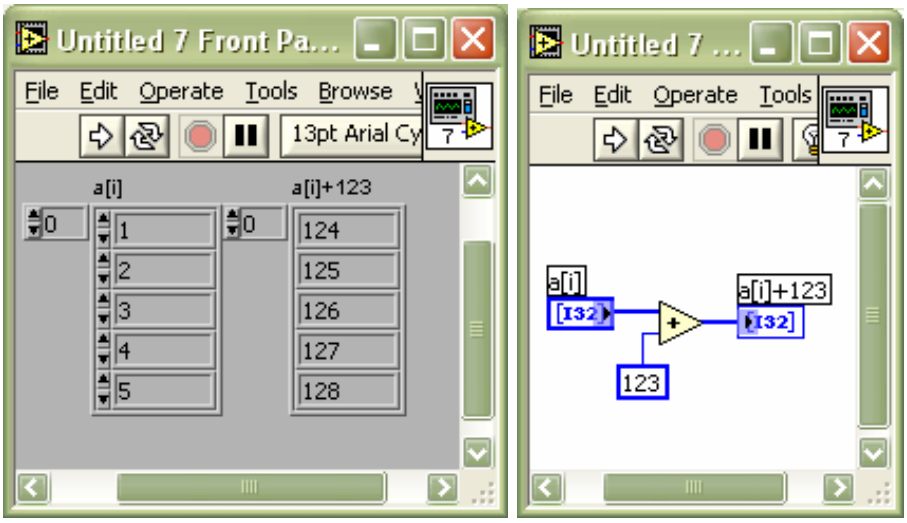

Рис 12. Функция "сумма" в LabVIEW является полиморфной. Она может складывать два числа, может прибавить число ко всем элементам массива и т.п.

4. Конструкции программирования LabVIEW

В системе программирования LabVIEW конструкции программирования называются Structures (структуры). Авторы предпочитают называть эти объекты конструкциями программирования, поскольку термин "структура" имеет совершенно отличный смысл применительно к ряду других систем программирования [12].

Все конструкции программирования LabVIEW представляют собой прямоугольные области блок-диаграммы, ограниченные рамкой. Внутри этой области может быть помещен требуемый программный код. Провода могут пересекать границы конструкций, передавая исходные данные внутрь конструкции, а результаты наружу. В месте пересечения проводами границы автоматически формируются входные и выходные туннели. Через входные туннели данные поступают внутрь конструкции в качестве исходных данных, через выходные - покидают ее в качестве результатов работы.

Каждая конструкция программирования может рассматриваться как "черный ящик" или сложный узел. Это означает, что конструкция начинает выполняться после того, как *на все входные туннели поступят данные*, а значения выходных туннелей будут определены лишь после того, как *вся конструкция закончит свою работу*.

4.1. Последовательность (Sequence)

Конструкция "последовательность" позволяет определить порядок выполнения узлов ВП.

Конструкция "последовательность" может состоять из одной или нескольких страниц программного кода, пронумерованных в порядке возрастания: 0,1,2,…,N. Сначала полностью выполняется код на странице 0, затем полностью выполняется код на странице 1 и так далее, пока не будут полностью выполнены коды на всех страницах. После этого работа конструкции завершается, данные поступают на выходные терминалы и могут выполняться следующие узлы блок-диаграммы. Хорошая аналогия – это книга, которую вы читаете страница за страницей, последовательно переворачивая их. Как нельзя не прочитать до конца интересную книгу, так и конструкция "последовательность" обязательно выполнит все имеющиеся страницы, досрочно завершить выполнение нельзя. Все страницы "лежат в одной стопке", в одном месте блок-диаграммы друг над другом. На верхней границе рамки находится переключатель страниц, с помощью которого можно листать страницы. Нажатием правой кнопки мыши на границе конструкции можно получить всплывающее меню свойств, в котором можно добавлять новые страницы, удалять их, менять местами.

---

[12] В языке СИ ключевое слово *struct* описывает тип данных "структура", аналогичный типу данных кластер в LabVIEW

Конструкция "последовательность" позволяет передавать данные из одной страницы в последующие. Для этого на границе конструкции из всплывающего меню свойств можно поместить один или несколько элементов "Sequence Local". Элемент "Sequence Local" изображается и на страницах, предшествующих той, на которой данные будут определены, но доступ к данным на этих страницах невозможен.

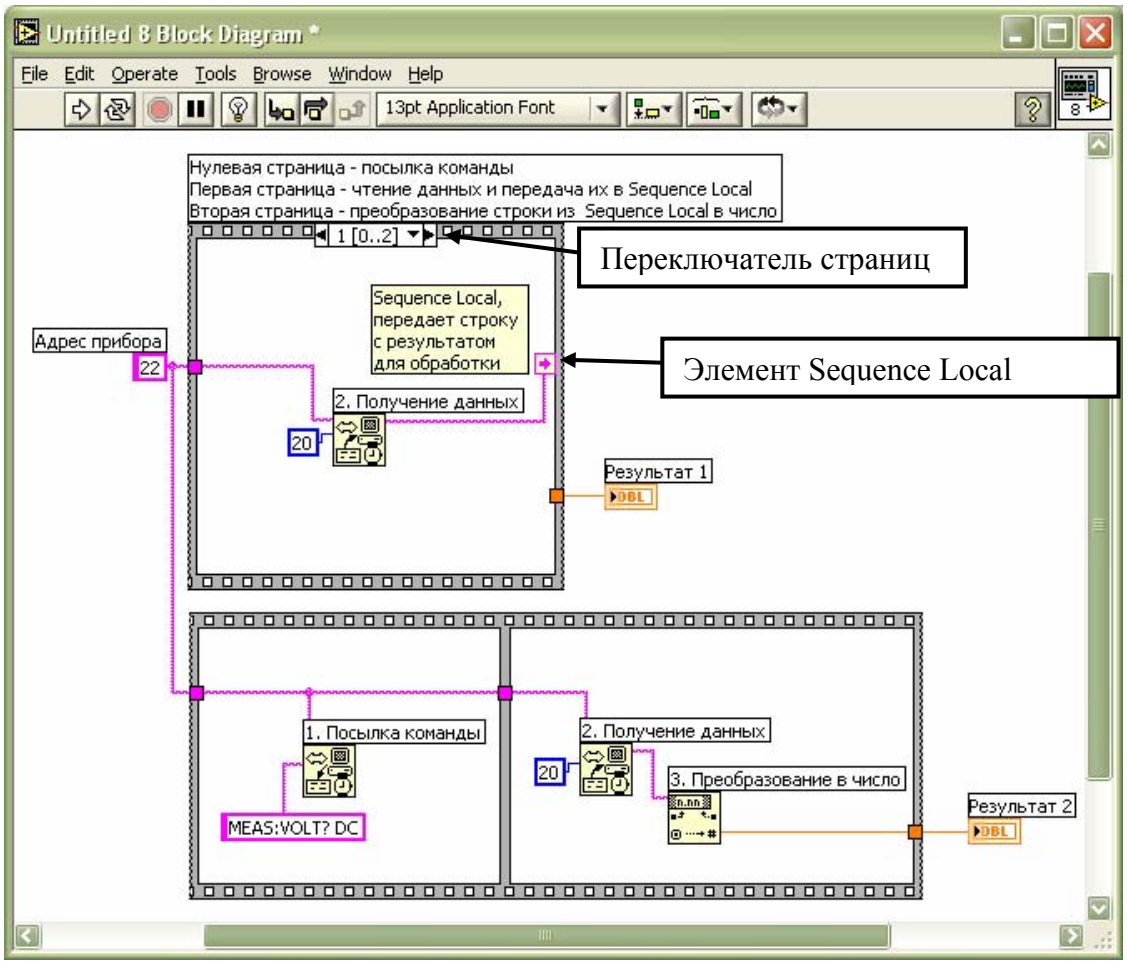

Рис 13. Конструкция "последовательность".
Измерение цифровым вольтметром.

Начиная с LabVIEW версии 7.0, реализован и другой вариант конструкции последовательность, когда страницы лежат не друг над другом, а разложены рядом, слева направо, по аналогии с кадрами кино- или фотопленки. Выполнение такой конструкции осуществляется "кадр за кадром".

На Рис. 13 изображены два варианта использования конструкции "последовательность" для измерения напряжения цифровым вольтметром.[13]

[13] смотри раздел *Порядок выполнения виртуального прибора, технология Dataflow.*

4.2. Условие (Case)

Конструкция "условие" аналогична таким операторам текстового языка, как if-then-else и switch (C)/case of (Pascal). Конструкция "условие" проверяет логический, либо числовой, либо строковый параметр на различные значения и выполняет лишь один из нескольких, соответствующий значению параметра, вариантов кода.

Конструкция "условие" состоит из двух (if-then-else) или более (switch/case of) страниц. На верхней границе конструкции расположен переключатель страниц. Каждой странице соответствует какое-либо значение проверяемой величины. Например, если проверяется логическое значение, то у конструкции условие будут две страницы со значениями "True" и "False". Если проверяется текстовое данное, содержащее название месяца, то у конструкции могут быть страницы со значениями "январь", "февраль", "март",… Если проверяются целые числа, то могут быть страницы со значениями "1", "2", "3:10" (диапазон от 3 до 10), "11,12,13, 15:20" (список и диапазон) и т.п. Менять проверочные значения можно с клавиатуры. Добавлять, удалять, тиражировать нужные страницы можно через всплывающее меню (правая кнопка мыши).

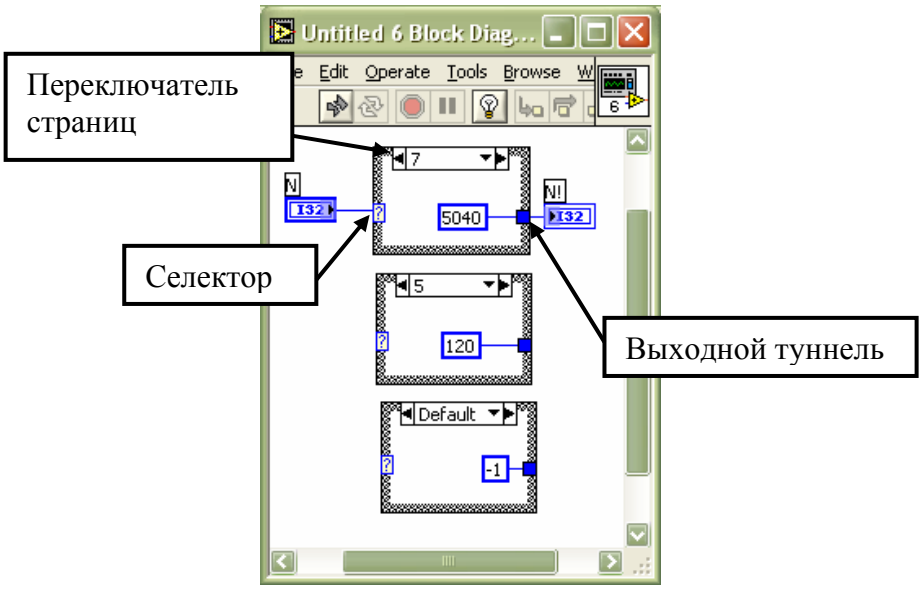

Рис. 14 Табулирование N!.

На левой границе конструкции "условие" расположен селектор (контакт со знаком вопроса), к которому следует проводом подключить проверяемую величину. При выборе из палитры-меню конструкция "условие" соответствует оператору if-then-else и по умолчанию проверяет логическую величину, селектор зеленый. Если требуется проверять строки или числа, то надо лишь подключить нужные данные проводом к селектору - цвет селектора автоматически изменится на розовый или синий.

Необходимо заметить, что в случае проверки строковых или числовых параметров одна из страниц в перечне соответствующих значений должна содержать специальное значение "Default" (по умолчанию). Эта страница будет выполнена в том случае, если ни одна из других страниц не пройдет проверку на соответствие.

На рис. 14 изображена блок-диаграмма определения N! методом табулирования значений (или табличное задание функции), когда интересуют не любые значения N, а лишь некоторые; для них N! рассчитывается предварительно. Каждому такому значению "N" соответствует своя страница конструкции "условие". Для каждого "известного", протабулированного значения "N" конструкция "условие" выдает результат N!. Все остальные, "неизвестные" значения "N" будут обработаны страницей со значением "Default", в данном случае результатом работы конструкции в этих случаях будет значение -1.

Особенности выходных туннелей конструкции "условие".

Данные, получаемые от конструкции "условие" с выходных туннелей должны быть определены независимо от значения проверяемой величины. Следовательно для всех выходных туннелей необходимо определить данные **на всех страницах конструкции "условие"!** – на каждой странице должен быть провод, передающий данные в выходной туннель. В некоторых случаях можно использовать значения туннелей по умолчанию. Для этого используется режим "Use Default If Unwired" всплывающего меню свойств туннеля. При этом для тех страниц, где данные в туннель не передаются, будут использованы значения для любых чисел – 0, для строк - пустая строка, для логических данных – False, для массивов - пустой массив и т.п.

4.3. Циклы (For Loop; While Loop)

В LabVIEW используются два варианта конструкции "цикл":

- цикл с известным числом итераций – "for";
- цикл с неизвестным числом итераций с *постпроверкой условия* продолжения цикла (на языке СИ *do {действие} while(условие)*, на Pascal – *repeat…until(условие)*);

Для реализации алгоритма предпроверки в LabVIEW можно использовать дополнительную конструкцию "условие" [14].

[14] Текстовые языки программирования, как правило, имеют вариант цикла с неизвестным числом итераций с *предроверкой условия* продолжения цикла – while(условие) {действие}

Режимы работы туннелей циклов, работа с массивами [15]

Входные и выходные туннели циклов могут работать в индексирующем режиме, который включается во всплывающем меню свойств туннеля. Эта возможность широко используется для работы с массивами. В общем случае действует правило: входной индексирующий туннель *уменьшает* на 1 размерность данных, т.е. если снаружи цикла двумерный массив (таблица), то внутри индексирующий туннель на каждой итерации будет выбирать отдельную строку – одномерный массив. (на "нулевой" итерации – строка с индексом 0, на первой – строка с индексом 1 и т.п.) Если снаружи одномерный массив, то внутри цикл будет последовательно перебирать отдельные элементы. Выходной индексирующий туннель увеличивает на 1 размерность данных (если внутри скаляр-снаружи одномерный массив, если внутри одномерные массивы (строки)-снаружи двумерный массив (таблица) и т.п.). При этом количество "элементов" в получившемся массиве будет соответствовать числу итераций цикла, см. Рис. 15. Также на Рис.15 изображена блок-диаграмма ВП расчета N! с использованием массива.

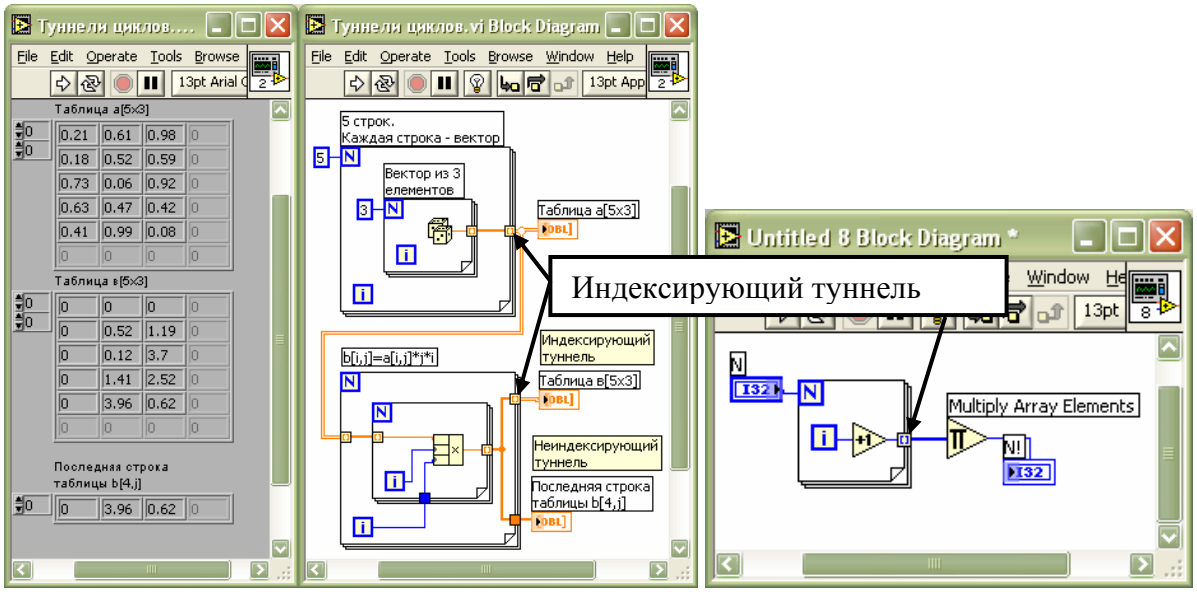

Рис. 15 Индексирующие туннели циклов при работе с массивами.

При использовании цикла For и входных индексирующих туннелей можно не подключать контакт "N" (требуемое число итераций). Входной индексирующий туннель сам определит, сколько элементов есть в массиве и выполнит нужное число итераций, чтобы обработать все элементы в соответствие с вышеуказанным правилом. Если цикл имеет несколько индексирующих туннелей, на которые передаются массивы с разным числом элементов, то количество итераций будет определяться минимальным

[15] В некоторых случаях эффективнее использовать возможности полиморфных функций. См. раздел Полиморфизм.

количеством элементов в массиве. Если какой-либо массив будет пустым, то цикл не выполнится ни разу (число итераций = 0).

Использование результатов предыдущей итерации, регистры сдвига (Shift Register), узлы обратной связи (Feedback Node)

Во многих циклических алгоритмах предполагается передача информации с одной итерации на другую.

Например, рассмотрим алгоритм вычисления факториала. Если "N" целое неотрицательное число, то функция "N!" определяется следующим образом:

N=0: N!=1;
N>0: N!=1*2*…*(N-1)*N

В комбинаторных задачах показывается, что функция N! выражает число перестановок "N" различных объектов. N! можно определить следующим рекуррентным соотношением: 0!=1; N!=(N-1)!*N, где N=1,2,3….

Приведем алгоритм расчета на "C":

```
/* Расчет функции n! Проверку на n>=0 не делаем */
1:      unsigned int fact(unsigned int n)
2:      {
3:              unsigned int f=1, i;
4:              for (i=1; i<=n; i=i+1)
5:                      f=f*i;
6:              return f;
7:      }
```

Запишем иначе пятую строку программы:

5*: $f_{(\text{текущей итерации})}=f_{(\text{с предыдущей итерации})}*i;$

В текстовых языках программирования это означает буквально следующее: надо взять значение из ячейки памяти с именем "f", умножить его на "i" и поместить обратно в ячейку памяти "f". Программируя в LabVIEW, мы не работаем с понятиями "ячейка памяти", мы работаем с потоками данных в соответствие с технологией Dataflow.

Для передачи данных с текущей итерации цикла на следующую используется элемент "регистр сдвига" (Shift Register). Регистр сдвига состоит из двух контактов на левой и правой сторонах рамки конструкции "цикл". Правый контакт позволяет запомнить значение на текущей, "i-ой" итерации, а левый – вспомнить его на следующей, "i плюс первой". Для выполнения "нулевой" итерации левый контакт регистра сдвига можно инициализировать нужным значением.[16]

На рис. 16 изображен алгоритм расчета N! в LabVIEW. Обратите внимание, что этот ВП можно использовать в других ВП в качестве подпрограммы (SubVI), см. рис 7-б.

[16] Начиная с LabVIEW 7.0 реализована аналогичная конструкция Feedback Node. Внешнее представление конструкции несколько иное, работа аналогична регистрам сдвига.

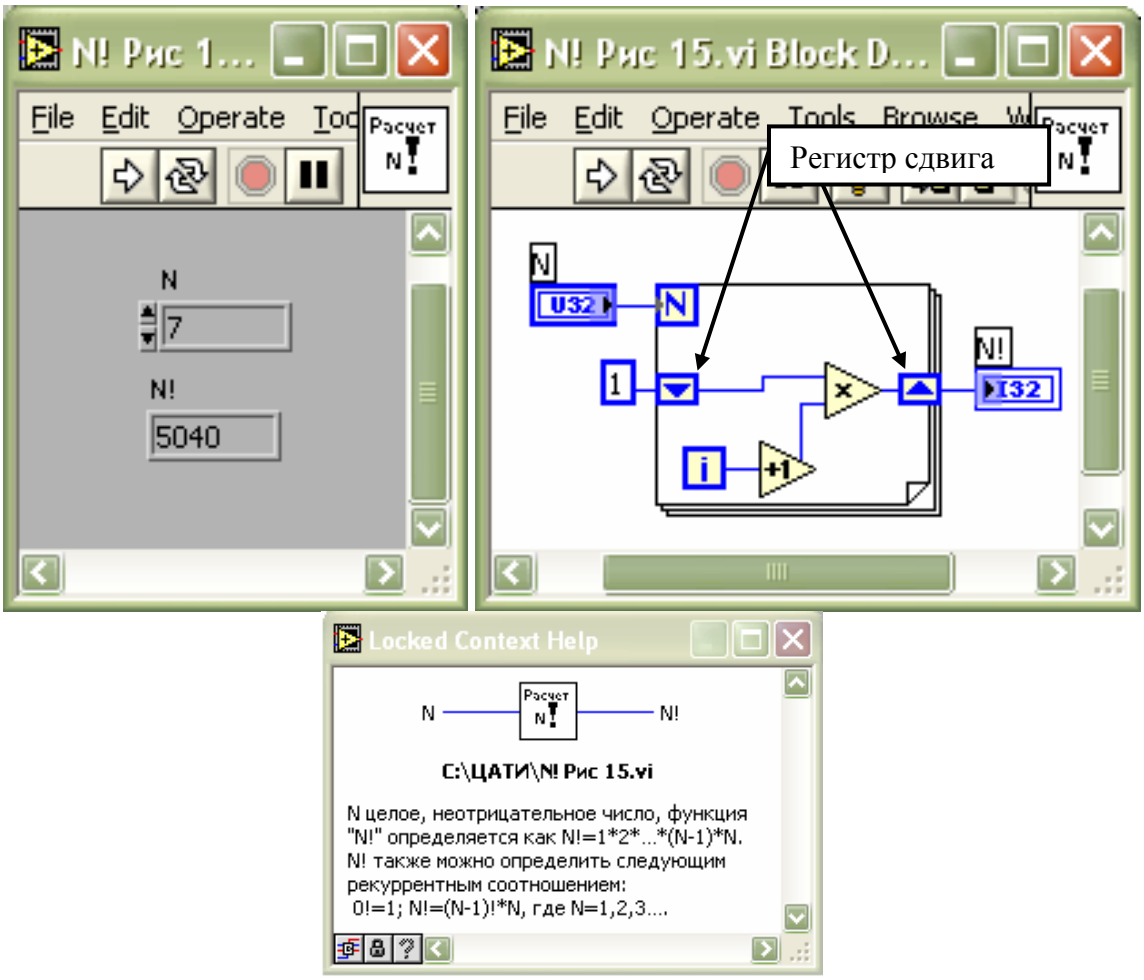

Рис. 16. ВП расчета N! с использованием регистра сдвига. Встроенная справочная система LabVIEW показывает описание этого ВП.

Особенности регистров сдвига

1. Левый, входной, контакт регистра сдвига можно растянуть по вертикали. В этом случае в текущей, "i-ой" итерации цикла можно использовать данные с "i-1", "i-2", и т.п. итераций.
2. Если на левый, входной, контакт регистра сдвига извне цикла не передаются данные (неинициализированный регистр сдвига), то регистр сдвига "запоминает" свое значение и сохраняет его для последующего запуска ВП. В языке "C" аналогичную роль выполняет ключевое слово "static". Один из способов использования этой, весьма полезной, особенности – реализация механизма Function Global. Этот механизм может использоваться для обмена данными, как между отдельными участками кода ВП, так и между различными ВП.

Особенности цикла "For"

1. Цикл "For" в LabVIEW нельзя завершить досрочно, до истечения заданного числа итераций. Можно блокировать участки кода от

выполнения на "нежелательных" итерациях с помощью конструкции "условие".

2. Цикл "For" может ни разу не выполниться, т.е. число итераций будет равно 0 (например в случае пустого массива, переданного на входной индексирующий туннель). В этом случае значениями данных **на всех выходных туннелях** будут **"значения по умолчанию"**.
3. Если данные поступают в цикл на вход регистра сдвига (Shift Register), и покидают цикл с выхода регистра сдвига, то значения сохраняются даже в том случае, если цикл ни разу не выполнится.

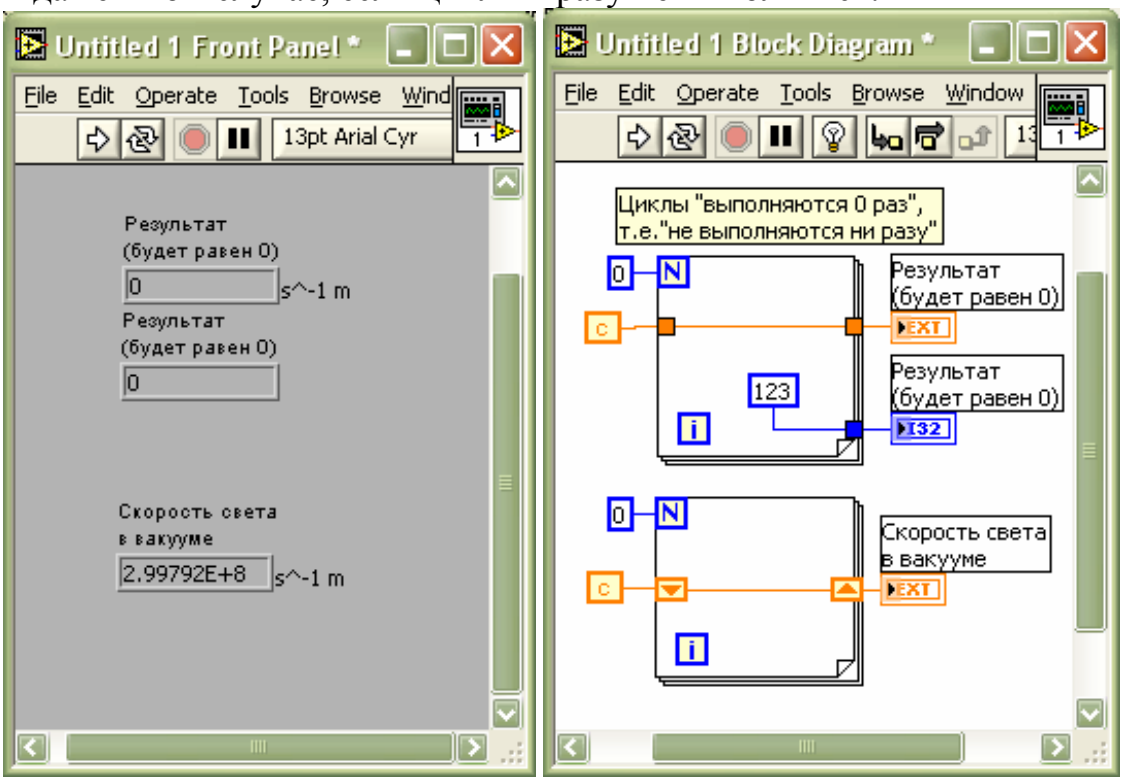

Рис 17. Особенности цикла For при "нулевом" числе итераций

Особенности цикла "While"

1. Цикл "While" в LabVIEW выполняется по крайней мере 1 раз, поскольку является циклом с **постпроверкой** условия.
2. В современных версиях LabVIEW с помощью всплывающего меню можно менять правило проверки продолжения-останова. Всплывающее меню позволяет выбрать один из двух режимов – "Stop if True" или "Continue if True" ("остановись, если истина" или "продолжай работу, если истина"). В ранних версиях LabVIEW был реализован только режим "Continue if True".

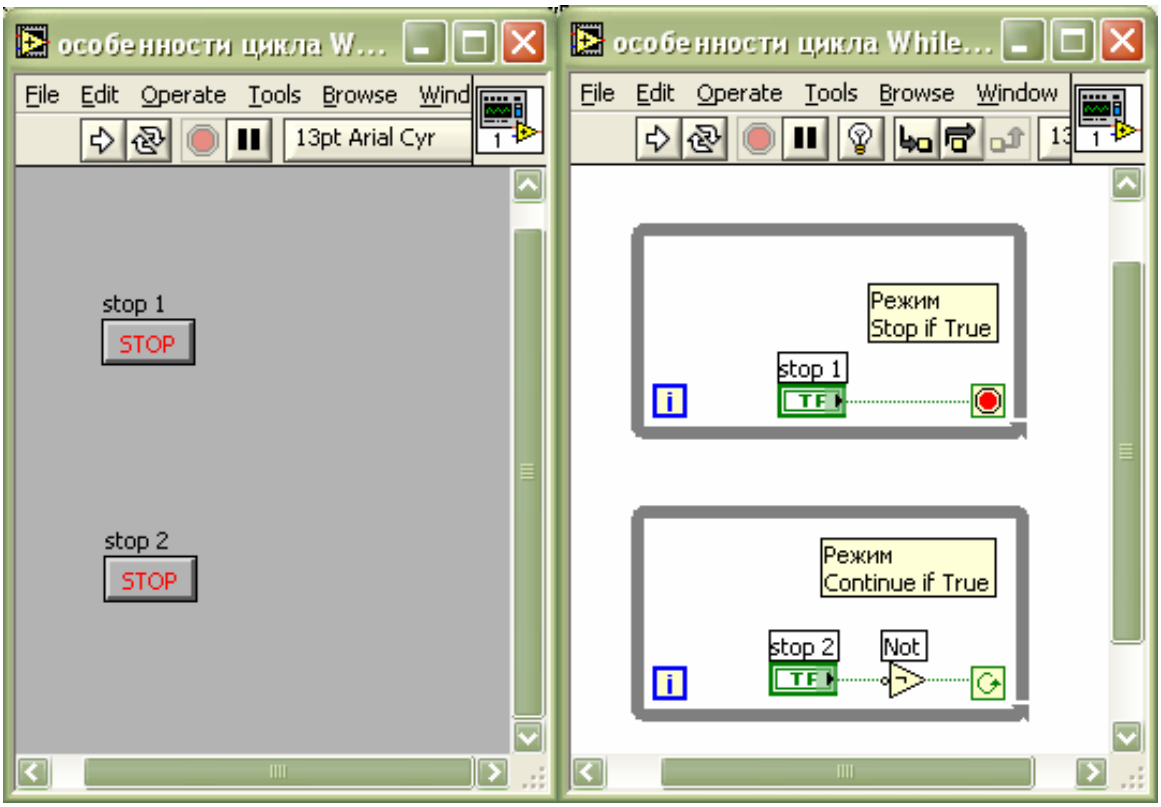

Рис 18. Различные режимы останова цикла While (пока кнопка на передней панели не нажата она выдает значение False, при нажатии - True)

4.4. Формула (Formula Node)

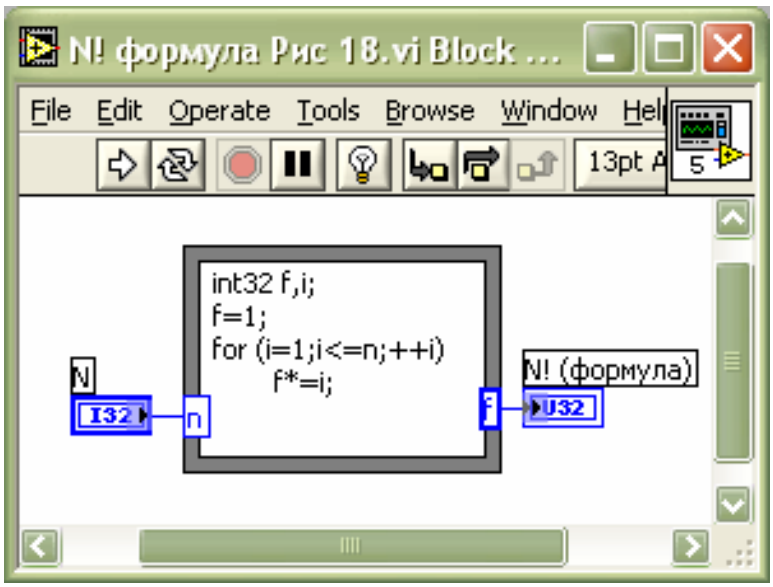

Рис. 19. Конструкция "Формула" при расчете N!

Конструкция "формула" позволяет компактно и наглядно кодировать сложные, громоздкие выражения. Во всех случаях такие выражения можно рассчитывать, используя встроенные функции LabVIEW, однако блок-

диаграмма в этом случае может быть весьма громоздкой и трудно воспринимаемой "на глаз".

Конструкция "формула" предоставляет возможность реализовывать небольшие фрагменты исходного кода "C" непосредственно в LabVIEW. На рис. 19 представлен вариант расчета функции N! с использованием этой конструкции.

4.5. Локальные (Local Variable) и глобальные (Global Variable) данные

Конструкция "глобальные данные" и соответствующая технология LabVIEW позволяют создать данные, которые будут использоваться различными ВП. Использовать эту технологию следует с большой осторожностью, чтобы не допустить спонтанного изменения данных виртуальными приборами. Опытные программисты достаточно редко используют "глобальные данные" при разработке виртуальных приборов, поскольку есть более безопасные и не менее эффективные методы передачи данных между ВП, такие как: Function Global, Буфер (Queue), технология Data Socket и др.

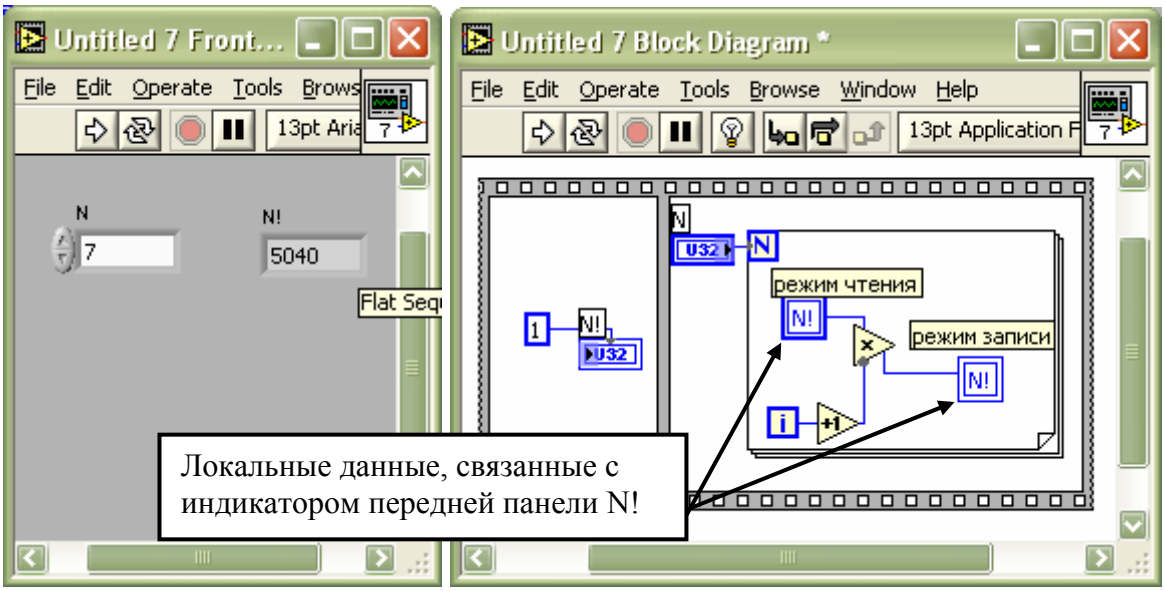

Рис. 20. Расчет N! с помощью конструкции "локальные данные" (Local Variable)

Гораздо чаще используется конструкция "локальные данные" (Local Variable). Ключевые аспекты использования локальных данных таковы:

1. Local Variable можно рассматривать как копию терминала **какого-либо элемента передней панели**. Таким образом, можно использовать один и тот же элемент передней панели в различных частях кода ВП. Например,

можно остановить два параллельно работающих цикла While нажатием на одну кнопку "Стоп" [17].

2. Использование конструкции "локальные данные" позволяет преодолеть разницу между режимами работы элемента пользовательского интерфеса "регулятор" и "индикатор". Т.е. можно программно изменять значения регуляторов и считывать значения с индикаторов. Local Variable может работать в режиме чтения или записи данных, режим переключается через всплывающее меню. На рис. 20 приведен пример расчета N! с использованием локальных переменных. Недостатком такого приема является низкая скорость работы, поскольку на каждой итерации требуется обращение к передней панели.

5. Пример создания виртуального прибора

Постановка задачи.

Пусть требуется разработать виртуальный прибор для циклической генерации случайного числа в заданном диапазоне с проверкой на допустимость значения и записью измеренных значений в файл.

Исходные данные для разработки ВП могут быть представлены следующей таблицей.

Количество параметров	1
Диапазон значений	диапазон задается в темпе эксперимента регулятором на передней панели
Период дискретизации	500 мс
Проверка на допустимость значений параметра	в темпе эксперимента, проверка на максимально допустимое значение (уставку)
Индикация аварии	лампа на передней панели
Визуализация в темпе эксперимента	"бегущий" график
Отображение результатов после эксперимента	таблица время-значение
Запись результатов в файл	в темпе эксперимента. Имя файла запрашивается у оператора в начале работы ВП
Формат файла	"текстовый", построчный
Формат строки	время, диапазон, уставка, значение параметра, разделитель полей– символ "табуляция"
Начало измерений	кнопка "Пуск" на передней панели
Завершение работы	кнопка "Стоп" на передней панели

[17] Может потребоваться изменение "механического аналога" кнопки через всплывающее меню свойств кнопки (Mechanical Action)

Разработка передней панели ВП.

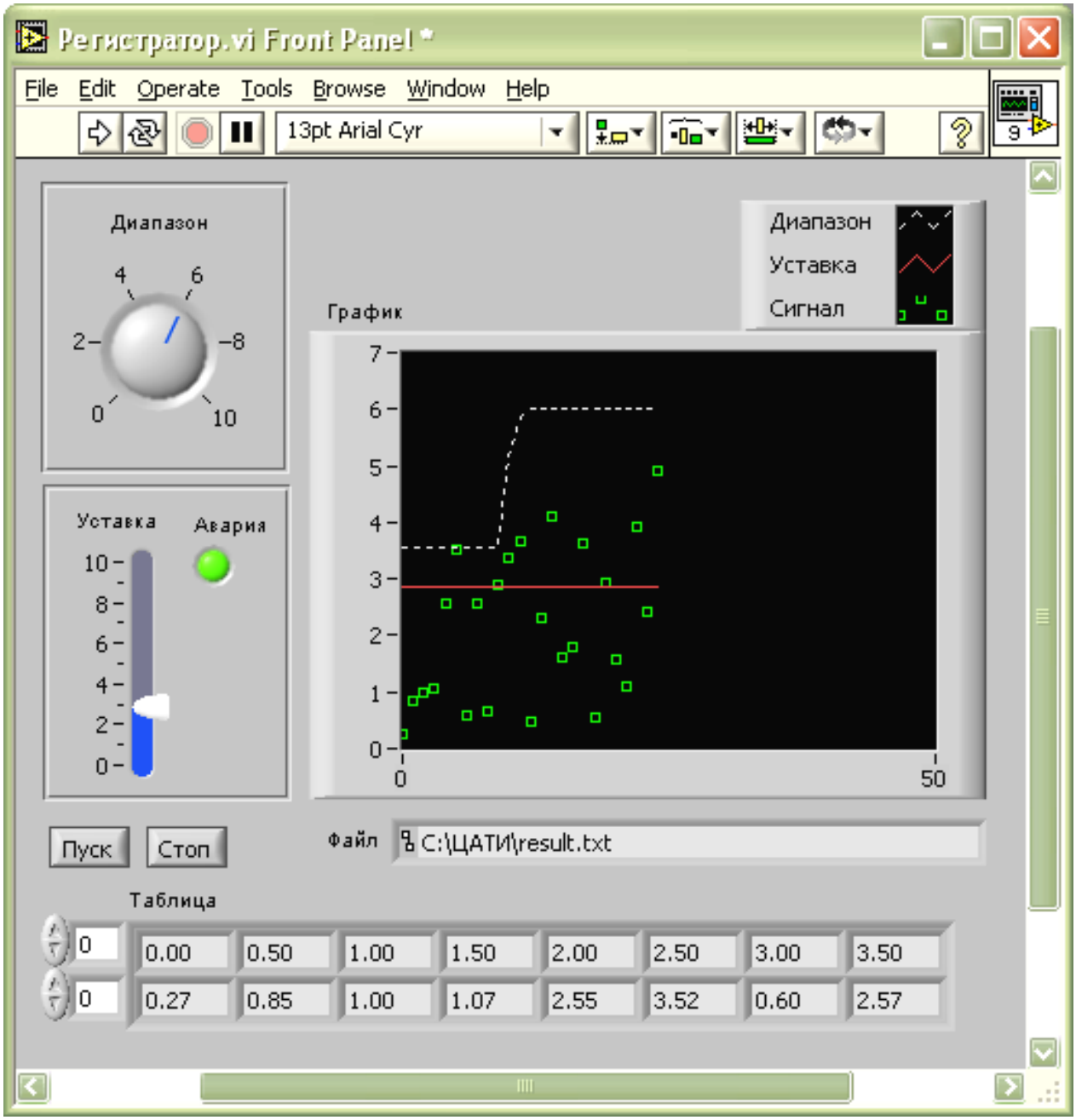

Рис. 21. Передняя панель одноканального регистратора.

Используя палитру-меню элементов пользовательского интерфейса конструируем переднюю панель. Возможный вид передней панели изображен на рис. 21. Для этого используются следующие элементы:

Название	**Палитра, элемент, режим работы**
Диапазон	Numeric, Dial, круглый регулятор
Уставка	Numeric, Vertical Slide, ползунковый регулятор
Авария	Boolean, лампа, индикатор

График	Graph, Waveform chart (бегущий график), индикатор, мышкой растягиваем объект Legend (легенда, над верхним правым углом графика), задаем способ отображения и названия кривых через всплывающее меню. Для отображения на графике нескольких значений эти значения следует собрать в кластер.
Пуск	Boolean, OK Button, регулятор, меняем надпись на кнопке
Стоп	Boolean, OK Button, регулятор, меняем надпись на кнопке
Файл	String & Path, File path indicator, индикатор
Таблица	Array, двумерный массив чисел, внутрь массива помещаем Numeric Indicator

Кроме того, для визуальной группировки различных исходных данных используем две прямоугольные рамки из палитры Decorations и окружаем ими регулятор "Диапазон" и параметры проверки на допустимость значений.

Разработка блок-диаграммы

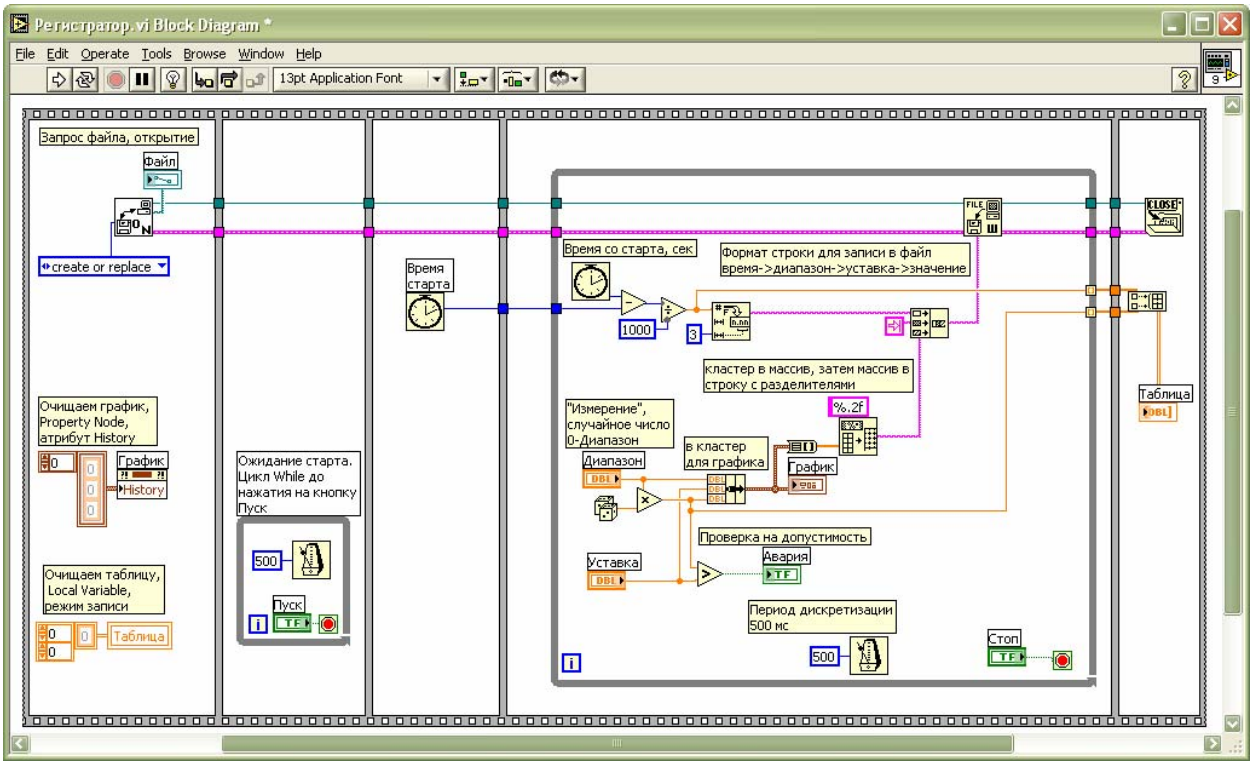

Рис. 22 Блок-диаграмма одноканального регистратора.

Алгоритм работы ВП может быть такой:

1. *начало (запуск виртуального прибора)*
2. *запрашиваем у оператора имя файла для сохранения результатов, открываем выбранный файл. Удаляем с графика результаты предыдущего эксперимента. Удаляем элементы из таблицы.*
3. *ожидаем нажатие на кнопку "Пуск"*
4. *определяем время старта, переходим в режим "измерений"*

5. *циклически, через dt=500 мс. выполняем следующие действия:*
   - *определяем время со старта;*
   - *генерируем случайное число 0-1. Умножаем на значение регулятора "Диапазон", получаем число 0-Диапазон;*
   - *определяем текущее значение уставки;*
   - *сравниваем результат измерения с уставкой, при превышении зажигаем лампу "Авария";*
   - *отображаем на график следующие данные: Диапазон, Уставка, измеренное значение;*
   - *формируем строку со значением времени, для чего преобразуем число (время в секундах) в строковый вид – формат числа с плавающей точкой, 3 знака после запятой;*
   - *формируем строку с полями Диапазон, Уставка, измеренное значение, разделенными табуляцией, в конце символ перевода строки. Формат преобразования – число с плавающей точкой, 2 знака после запятой;*
   - *формируем итоговую строку для записи в файл в виде время – табуляция – строка с предыдущего этапа;*
   - *записываем итоговую строку в файл;*
   - *передаем текущее время и измеренное значение на выходные индексирующие туннели цикла, чтобы можно было после завершения работы построить таблицу время-значение;*
   - *проверяем условие завершения цикла – нажатие оператором на кнопку "Стоп"*
6. *после завершения цикла на шаге №4 закрываем файл, получаем одномерные массивы с выходных индексирующих туннелей цикла, формируем двумерный массив время-значение и отображаем на переднюю панель.*
7. *конец:*

Для реализации этого алгоритма можно использовать конструкцию программирования "Последовательность", состоящую в данном случае из пяти страниц, расположенных слева направо. Помещаем на блок-диаграмму конструкцию *"Flat Sequence Structure"* через всплывающее меню добавляем нужное количество страниц. Затем формируем блок-диаграмму в соответствие с нашим алгоритмом. Один из вариантов построения блок-диаграммы изображен на рис. 22.

Сохраняем виртуальный прибор на диск под именем "Регистратор.vi".

Наш виртуальный готов к работе!

На рис. 23 показан фрагмент файла результатов в текстовом редакторе, а также возможность обработки полученных данных в других программах.

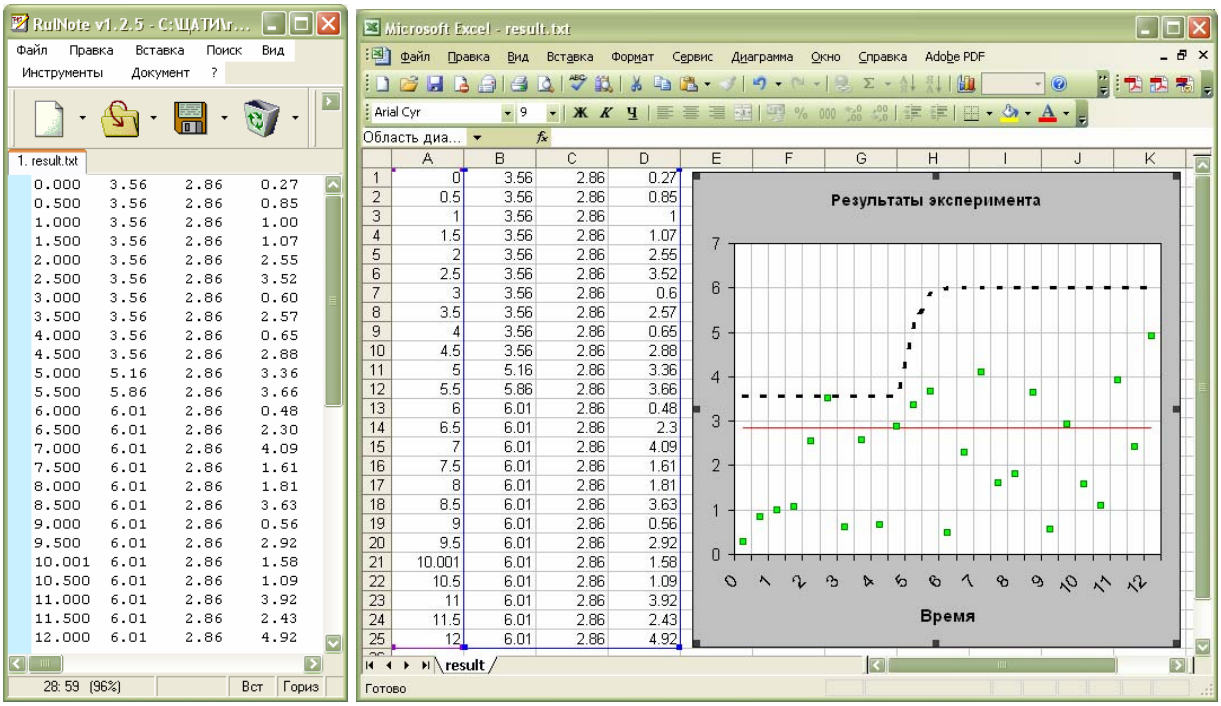

Рис. 23 Файл результатов работы регистратора,
обработка полученных данных в Microsoft Excel

6. Контрольные вопросы для самопроверки:

1. Какие пакеты для разработки программ вы знаете?
2. Каковы отличительные особенности среды графического программирования LabVIEW, какова структура этого инструментального средства?
3. Поясните содержание и назначение понятий «виртуальный прибор», «передняя панель», «блок-диаграмма», «пиктограмма», «коннектор».
4. Что находится на передней панели ВП? Объясните разницу между режимами работы элементов передней панели "Регулятор" и "Индикатор"?
5. Каким образом можно отобразить результаты работы ВП на элементе передней панели с режимом "Регулятор"? Как считать данные с "Индикатора"?
6. Что находится на блок-диаграмме? Что такое узел, терминал, конструкция программирования?
7. Чем определяется порядок выполнения виртуального прибора?
8. Что такое полиморфизм?
9. Как в LabVIEW огранизовать цикл с неизвестным числом итераций с предпроверкой условия?

Приложения

Приложение 1. Краткое описание интерфейса LabVIEW - меню, тулбары, горячие клавиши[18]

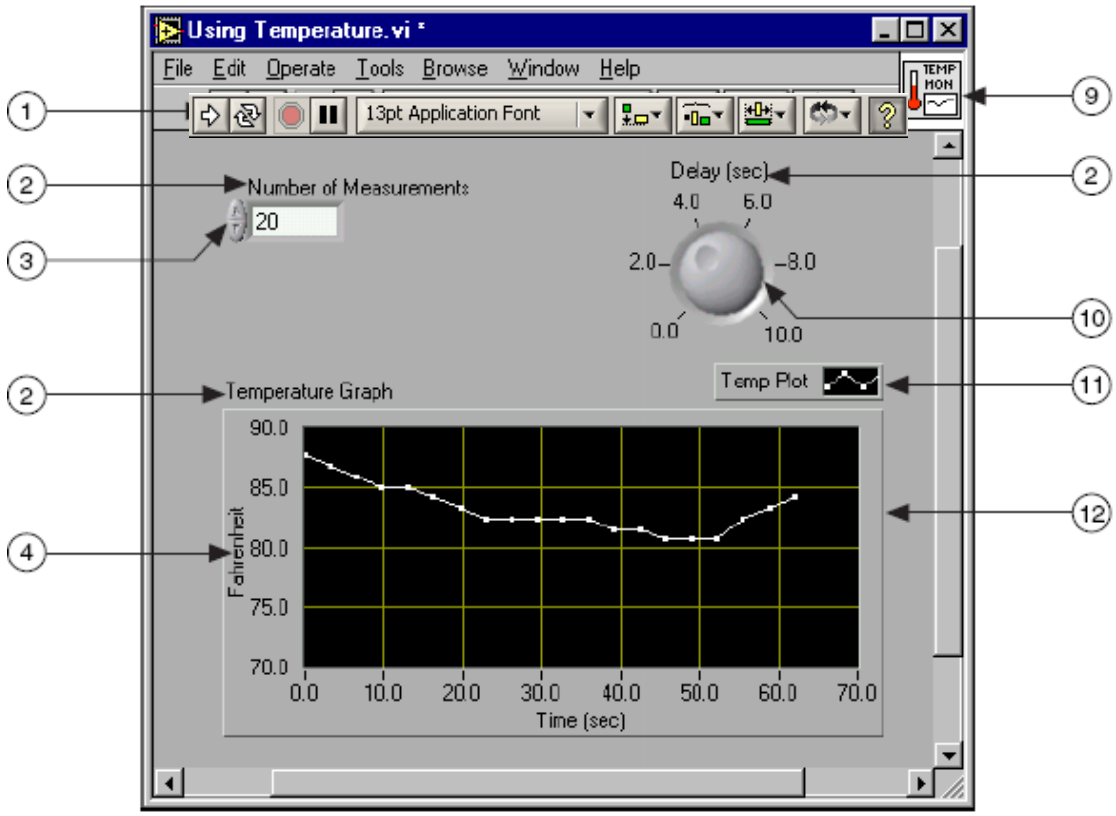

Рис. П1. Передняя панель виртуального прибора.

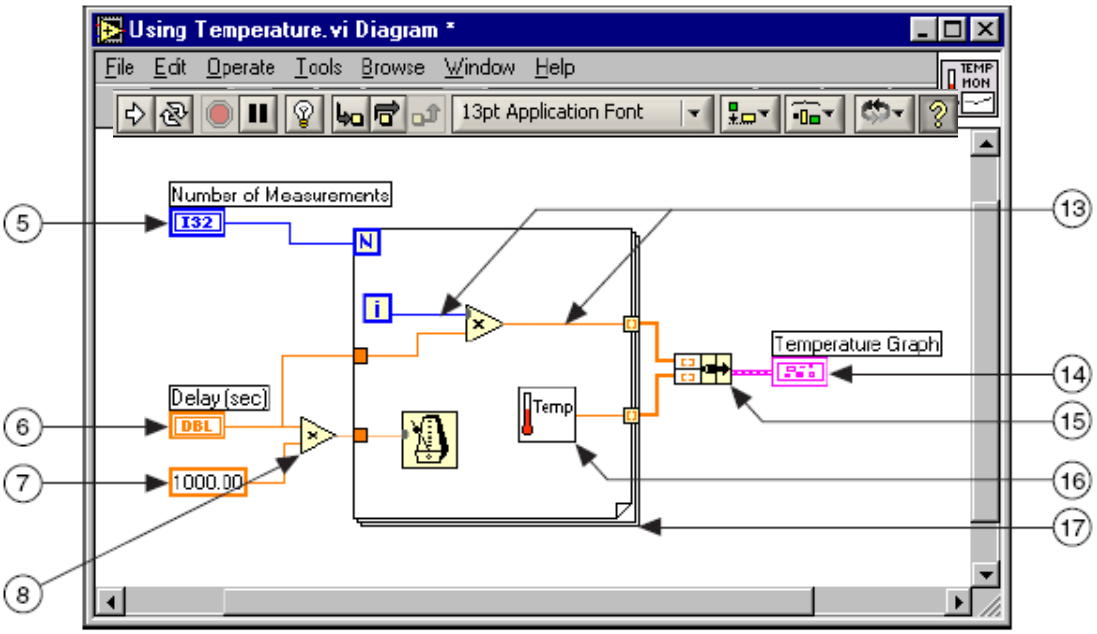

Рис. П2. Блок-диаграмма виртуального прибора.

[18] Здесь приводится краткое описание интерфейса системы LabVIEW версии 7.1. На момент издания данного пособия текущей версией LabVIEW является версия 8.0, в которой меню и внешний вид палитр-меню несколько отличается.

1. Тулбар (Toolbar)
2. Метка-название элемента передней панели (Label)
3. Элемент «числовой регулятор» (Digital Numeric Control)
4. Метка-название оси на элементе «график» (Scale Label)
5. Терминал элемента «числовой регулятор» (поз.3) (Digital Numeric Control Terminal)
6. Терминал элемента «числовой регулятор» (поз.10) (Knob Terminal)
7. Константа на панели блок-диаграммы (Numeric Constant)
8. Узел-функция «умножения» (Multiply Function)
9. Пиктограмма ВП (Icon)
10. Элемент «числовой регулятор» (Knob Control)
11. «Легенда» графика (Graph Legend)
12. График (XY Graph)
13. Провод на панели блок-диаграммы
14. Терминал элемента График (поз.12) (XY Graph Terminal)
15. Функция сборки элементов в кластер (Bundle Function)
16. Подпрограмма (Sub VI)
17. Конструкция программирования «Цикл For» (For Loop Structure)

Тулбары

Для окон переденей панели и блок-диаграммы вид тулбаров несколько отличается. Тулбар панели блок-диаграммы имеет дополнительные кнопки для отладки ВП

Рис. П3. Тулбар окна передней панели (Front Panel Toolbar)

Рис. П4. Тулбар окна блок-диаграммы (Block Diagram Toolbar)

	запустить ВП
	 вид кнопки запуски ВП в процессе работы
	 компилятор обнаружил ошибки в коде ВП, нажатие на кнопку откроет список ошибок.
	циклический непрерывный запуск ВП, отмена циклического запуска
	аварийный останов ВП
	приостанов выполнения ВП, переключение пошагового режима выполнения ВП
	настройка атрибутов отображения текста
	выравнивание и распределение элементов на панели
	показать/спрятать окно контекстно-чувствительной справочной системы (версии LabVIEW 7.0 и выше)

Дополнительные кнопки тулбара окна блок-диаграммы	
	замедленное выполнение ВП с режимом визуализации, показ текущих данных, передаваемых каждым проводом
	пошаговое исполнение с заходом «внутрь» конструкции программирования или подпрограммы (SubVI)
	пошаговое исполнение без захода «внутрь» конструкции программирования или подпрограммы (SubVI)
	выполнение программы ВП до завершения текущего блока исходного кода, после - останов

Палитры-меню (Palette)[19]

Палитра-меню Controls Palette используется для создания передней панели ВП. Она содержит в иерархической форме все элементы пользовательского интерфейса (числовые регуляторы и индикаторы, логические, графики, кнопки и т.п.).

Палитра-меню элементов блок-диаграммы (Functions Palette) используется для создания блок-диаграммы ВП. Содержит в иерархической форме все встроенные функции и подпрограммы (SubVI), распределенные по функциональному назначению (арифметика, работа со строками, массивами, кластерами, файловый ввод-вывод и т.п.).

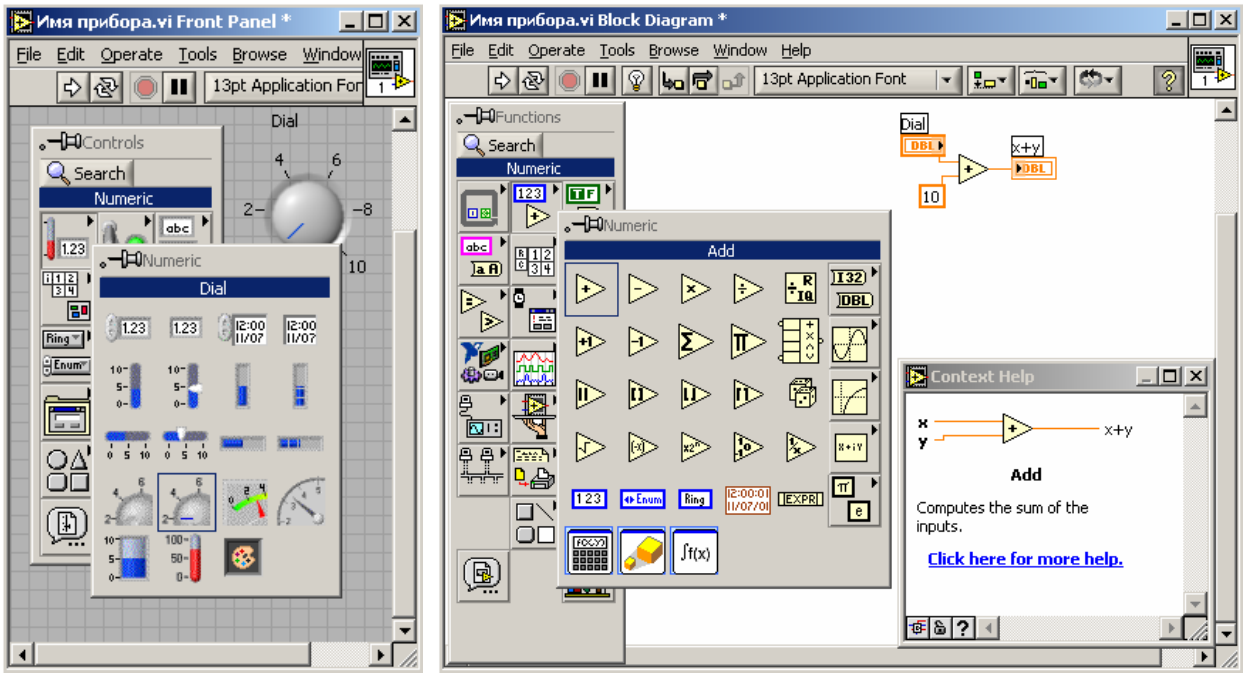

Рис. П5. Палитра-меню *Controls Palette* на окне передней панели (слева) и палитра-меню *Functions Palette* окне блок-диаграммы (справа).

Для отображения палитр на экране используются пункты меню *Window>>Show Controls Palette* окна передней панели ВП и *Window>>Show Functions Palette* окна блок-диаграммы ВП. Также палитры-меню могут быть вызваны щелчком правой кнопки мыши на свободном месте экрана передней панели или блок-диаграммы ВП. Меню Tools>>Options позволяет настроить LabVIEW так, чтобы палитры-меню отображались на экране автоматически при запуске LabVIEW.

[19] Внешний вид палитр-меню можно настроить через меню Tools>>Options

Палитра-меню инструментов (Tools Palette).

Содержит инструменты, использующиеся для разработки ВП. Отобразить палитру можно с помощью пункта меню *Window>>Show Tools Palette*, либо комбинацией клавиш Shift+правая кнопка мыши.

Рис. П6. Палитра-меню инструментов.

	«указательный палец» (Operate Value) – изменение значений регуляторов, констант, нажатие кнопок. Этот инструмент автоматически включается во время запуска ВП.
	«стрелка» (Position/Size/Select) – основной инструмент редактирования передней панели и блок диаграммы ВП. Позволяет менять расположение, размеры элементов, группировать их вместе.
	«текст» (Edit Text) – позволяет менять текстовые поля в любом месте рабочей области LabVIEW, например, менять названия элементов, настраивать оси координат графиков, создавать свободные текстовые комментарии на передней панели и блок-диаграмме ВП и т.п.
	«катушка» (Connect Wire) – используется для прокладки и подключения проводов на блок диаграмме, а также при создании коннектора подпрограммы (SubVI) – указывает соответствие контакта коннектора ВП и элемента передней панели ВП.
	«всплывающее меню свойств» (Object Shortcut Menu) – аналогичен правой кнопке мыши. Отображает всплывающее меню свойств объекта или палитры-меню элементов передней панели или блок-диаграммы ВП.
	«ладонь» (Scroll Window) – перемещает рабочее поле передней панели или блок-диаграммы в любом направлении относительно окна программы. Используется в случае больших размеров передней панели/блок-диаграммы или недостаточном разрешении видеосистемы компьютера.
	«точка останова» (Set/Clear Breakpoint) – используется при отладке ВП. Назначает/снимает точку останова для любого узла ВП. Точкой останова можно управлять из всплывающего меню свойств узла.
	«пробник» (Probe Data) – используется для просмотра значений данных, передаваемых по любому проводу блок-диаграммы. Пробником можно управлять из всплывающего меню свойств нужного провода.
	«пипетка» (Get Color) – запоминает цвет выбранного объекта.
	«палитра цветов» и «кисть» (Set Color) – позволяет выбрать цвет фона и текста, а также раскрасить выбранный объект.

Меню[20]

Большинство пунктов меню являются стандратными как и для большинства Windows-приложений. Отметим лишь некоторые пункты меню, характерные для LabVIEW.

Раздел меню	Пункт меню	Горячие клавиши	Назначение
File			
	New VI	Ctrl+N	Создать новый ВП
	Save All		Сохранить все ВП (включая всю иерархию ВП)
	Save with Options		Сохранить с опциями (например, защитить паролем или записать в бибилиотеку ВП)
	VI Properties	Ctrl+I	Настроить свойства ВП
Edit			
	Undo	Ctrl+Z	Отменить последнее действие (количество шагов «отката» можно настроить в меню *Tools>>Options*)
	Customize Control		Редактировать внешний вид элемента передней панели, позваляет создавать собственные элементы передней панели
	Remove Broken Wires	Ctrl+B	Удалить с блок-диаграммы все «плохие», разорванные провода
	Create SubVI		Создать подпрограмму на основе выделенного на блок-диаграмме кода
	Run-Time Menu		Создание иедактирование меню ВП, которое будет отображаться и работать во время выполнения ВП
Operate			
	Run	Ctrl+R	Запуск ВП из оболочки LabVIEW
	Stop	Ctrl+ «.»	Останов ВП
Tools			
	Measurement & Automation Explorer		Запуск интерактивного драйвера оборудования National Instruments (MAX)
	Instrumentation		Подключение драйверов приборов к LabVIEW
	Data Acquisition		Интерактивный помощник создания приложений для работы с многофункциональными измерительно-управляющими платами National Instruments
	Build Application or Shared Library (DLL)		Компиляция исходного кода, создание EXE-файла или библиотеки динамической компоновки DLL
	Edit VI Library		Редактировать библиотеку ВП.
	Options		Настройки LabVIEW.

[20] Расположение пунктов меню может несколько отличаться в разных версиях LabVIEW.

Browse			
	Show VI Hierarchy		Показать полную иерархию ВП (ВП верхнего уровня и все связанные подпрограммы)
Window			
	Show Block Diagram/Show Front Panel	Ctrl+E	Открыть блок-диаграмму/переднюю панель текущего ВП (удобно при одновременном редактировании нескольких ВП)
	Show Controls Palette (в меню окна передней панели)		Открыть палитру-меню элементов построения передней панели ВП
	Show Functions Palette (в меню окна блок-диаграммы)		Открыть палитру-меню построения блок-диаграммы ВП
	Show Tools Palette		Открыть палитру инструментов
	Show Error List	Ctrl+L	Показать список предупреждений и ошибок компилятора LabVIEW
Help			
	Show Context Help	Ctrl+H	Показать/спрятать окно контекстно-чувствительной справочной системы
	VI, Function, & How – To Help	Ctrl+?	Открыть справочник по функциям LabVIEW
	Search The LabVIEW Bookshelf		Поисковая система по электронной документации LabVIEW
	Find Examples		Поиск примеров ВП по тематике задачи или по имени
	Explain Error		Расшифровать числовое значение кода ошибки

Наиболее часто употребляемые «горячие» клавиши LabVIEW (Hot Keys)

Ctrl+B	Удалить все "плохие" провода с блок-диаграммы
Ctrl+E	Найти и показать окно передней панели или блок-диаграммы ВП
Ctrl+S	Сохранить ВП [21]
Ctrl+R	Запуск ВП
правая кнопка мыши	1. Щелчок на объекте показывает всплывающее меню свойств объекта. 2. Щелчок на пустом месте экрана отображает палитру-меню элементов пользовательского интерфейса, или палитру-меню конструкций программирования, функций и подпрограмм
Shift+правая кнопка	Показ всплывающего меню для выбора инструмента

[21] В LabVIEW нет механизма автоматического сохранения программ через заданный промежуток времени.

Приложение 2. Различные способы расчета функции N!

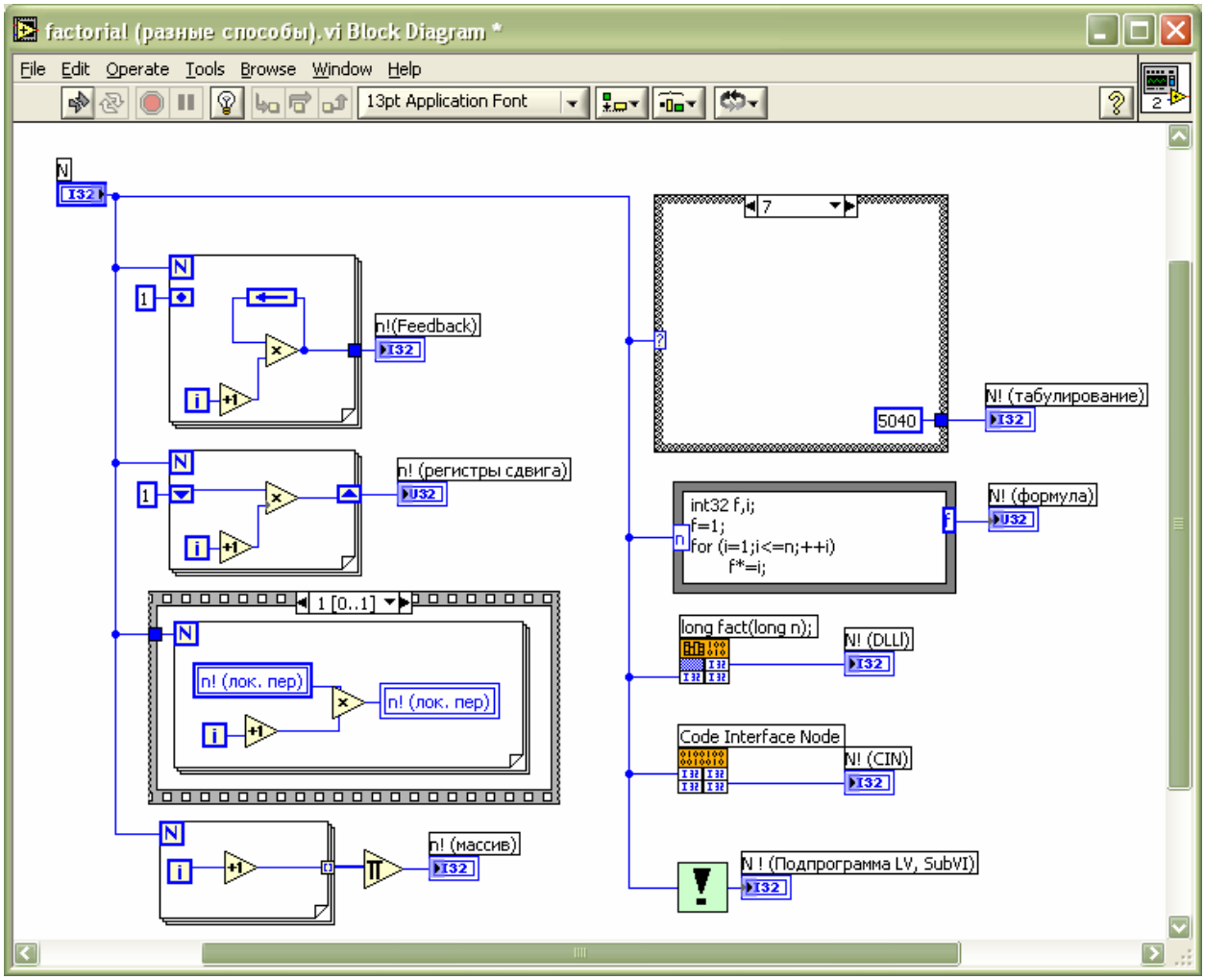

Рис П7. Различные способы расчета значения N!

На Рис.П7 приведено девять разных способов определения значения N!.

- использование конструкции Feedback Node;
- использование регистра сдвига (Shift Register);
- использование конструкций "последовательность" и "локальные данные";
- формирование массива чисел [1,2,3...N] в цикле с использованием индексирующего туннеля, затем перемножение элементов
- табличное задание функции, использование конструкции "условие";
- использование конструкции "формула";
- вызов функции библиотеки динамической компоновки DLL
- интегрирование кода программы на "СИ" в LabVIEW (Code Interface Node);
- использование ранее созданной подпрограммы (SubVI).

Библиографический список

1. Пейч Л.И., Точилин Д.А., Поллак Б.П. LabVIEW для новичков и специалистов. М.: Горячая линия-Телеком, 2004.-268 с.
2. Теоретические основы теплотехники. Справочник./Под ред. А.В.Клименко и В.М.Зорина. 3-е изд.. М.: Издат. МЭИ, 2001г. Раздел 8 «Теплотехнический эксперимент».

Оглавление

Учебное издание

Наталия Александровна Виноградова,

Ярослав Игоревич Листратов, Евгений Валентинович Свиридов

РАЗРАБОТКА ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В СРЕДЕ LabVIEW

Учебное пособие по курсам
«Новые информационно-измерительные системы и технологии», «Автоматизированные системы научных исследований в теплофизическом эксперименте», «Технические средства автоматизации и управления»
для студентов, обучающихся по направлениям
«Техническая физика», «Автоматизация и управление»

Редактор издательства

Темплан издания МЭИ	учебн	Печать офсетная
Подписано к печати	Формат бумаги 60x80	Физ.печ.л.
Тираж 300	Изд.№	Заказ

Издательство МЭИ, 112250, Москва Красноказарменная ул., д.14

Отпечатано в типографии ЦНИИ «Электроника»,
117415, Москва, просп. Вернадского, д.39