ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность. В России вот уже несколько лет все чаще заходят разговоры о модернизации предметной области «Технология». Для чего это нужно и почему столько внимания уделяется «второстепенному» по мнению многих учителей, школьников и их родителей предмету?

Дело в том, что в данный момент мы стоим на пороге глобальных мировых изменений, которые открывают все больше новых возможностей. Однако, для нашей страны все эти возможности могут быть осуществимы только в том случае, если мы окажемся способны ими воспользоваться, сумеем кардинально перестроить экономику, в том числе отказаться от нефтяной и газовой зависимости, раздвинуть границы существующих технологий, и тем самым, создавать новые.

А для того, чтобы все это реализовать и не оказаться в тени технического прогресса, необходимы новые, высококвалифицированные специалисты.

Именно поэтому и необходимо менять подходы к преподаванию «Технологии».

"Нам нужно, безусловно, подумать о том - как качественно изменить преподавание школьного предмета "Технология", чтобы ребята могли закреплять базовые знания, полученные при изучении физики, химии и других предметов в практической, проектной деятельности", - сказал глава государства на Съезде Союза машиностроителей в 2016 году.

В настоящее время, реализовать эту идею возможно с помощью графической среды программирования LabView - это среда разработки и платформа для выполнения программ, созданных на графическом языке программирования «G» фирмы National Instruments.

Изучая данную программу учащиеся получают возможность самостоятельно создавать виртуальные приборы, которые подходят для целей вычисления, структурирования, и обработки данных, а также, что немаловажно, для управления техническими устройствами и технологическими процессами.

При чем, обучение в данной среде не будет оторвано от реальности, ведь это не просто так называемый тренажер, подходящий лишь для целей обучения, а инструмент, используемый инженерами во многих развитых странах.

Сейчас обучение языку программирования LabView начинается только в ВУЗах и техникумах, но очевидно, что простота и удобство программирования в этой среде позволяет начать подготовку уже в школе. Это даст возможность одновременно отработать навыки сразу по нескольким смежным дисциплинам: математика, физика, технология и информатика и таким образом сократить разрыв между школьным и профессиональным обучением.

Другими словами, уроки технологии, осуществляемые с применением среды программирования LabView должны позволить моделировать законы реального мира, связать теоретические знания с практическим опытом, развивать наблюдательность, мышление, сообразительность, креативность.

Однако к настоящему времени недостаточно методической литературы: учебников или сборников заданий, которые были бы понятны школьникам. Учебные пособия рассчитаны на учащихся высших и профессиональных учебных заведений, которые обладают большими познаниями в точных областях науки.

Проведенный анализ позволил определить **проблему** исследования: Недостаточная проработанность методического обеспечения, способствующего интеграции в рамках технологического образования некоторых разделов физики, математики и информатики с использованием технологии программирования в среде LabView.

Для того, чтобы решить указанную проблему, для начала необходимо определить, пути интеграции технологии, физики, математики и информатики и определить каких результатов должен достичь ученик восьмого класса в процессе изучения LabView. Таким образом, **объектом** исследования будет: интеграция естественно-научных дисциплин в рамках технологического образования.

Обучение среде LabView должно проходить в форме практических занятий с использованием заданий, смежных с предметами естественно-научного цикла. Такая форма позволит учащимся применять полученные знания, экспериментировать, моделировать, наблюдать, изучать объекты техники, исследовать явления природы.

Исходя из этого **предмет** исследования: использование среды графического программирования LabVIEW для установления межпредметных связей с дисциплинами естественно-научного цикла.

Цель исследования: разработка комплекса практических заданий по программированию, обеспечивающих интеграцию физики, математики и информатики в рамках предметной области «Технология».

Исследование исходит из **гипотезы:**

Задачи по программированию в среде LabVIEW, могут быть включены в курс «Технологии» восьмого класса с целью обеспечения интеграции физики, математики и информатики.

Объект, предмет и цель исследования определили круг **задач:**

1. Провести анализ языков программирования в контексте их использования на уроках технологии.
2. Проанализировать содержание рабочих программ по математике, физике, информатике и технологии.
3. Определить роль и место программирования в среде LabVIEW.
4. На основе анализа рабочих программ, определить круг межпредметных задач, визуализация решения которых возможна с помощью Lab View.
5. Провести апробацию по внедрению межпредметных задач в курс технологии 8 класса.

Работа состоит из введения, двух глав, заключения и списка литературы, состоящего из 28 наименований. Текст иллюстрируют 6 таблиц, 48 рисунков. Объем работы составляет 72 страницы.

Апробация работы проводилась на базе гимназии №13 «Академ» г. Красноярска.

Основные результаты исследования опубликованы в материалах конференции:

– XX Международный научно-практический форум студентов, аспирантов и молодых ученых «МОЛОДЕЖЬ И НАУКА XXI ВЕКА» тема доклада: «Проектирование практических заданий по программированию в среде LabVIEW как результат интеграции технологии, физики, математики и информатики для учащихся восьмых классов» (от 26 апреля 2019 года).

ГЛАВА 1. ГРАФИЧЕСКОЕ ПРОГРАММИРОВАНИЕ И ЕГО РОЛЬ В ИНТЕГРАЦИИ ЕСТЕСТВЕННО-НАУЧНЫХ ДИСЦИПЛИН

1.1 Графические языки программирования

Программирование – это процесс создания компьютерных программ [2]

По сути, под программированием понимается ни что иное, как объяснение машине того, что хочет получить от нее пользователь. А для того, чтобы машина поняла требуемое, необходимо изъясняться на понятном ей языке, который называют языком программирования.

Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под её управлением. [4]

Развитие языков программирования началось сразу же после появления первых компьютеров на электронных лампах. [2]

На начальной стадии развития ЭВМ человеку было необходимо составлять программы на языке, понятном компьютеру, в машинных кодах. Каждая команда состояла из кода операций и адресов операндов, выраженных в виде различных сочетаний единиц и нулей. Итак, любая программа для процессора выглядела на то время как последовательность единиц и нулей. [24]

Использование такого способа программирования на практике позволило выявить ряд недостатков. Пользователь легко мог допустить ошибку, перепутав местами 1 и 0, а найти ее было сложно, ведь последовательность цифр была очень длинной. Выполнение программы плохо контролировалось, программирование требовало много времени и повышенного внимания.

Новым витком в развитии программирования стала автоматизация машинного кода. Для записи программ стали применять язык assembly, который представлял код в более понятной для человека форме: вместо нулей и единиц использовались буквы или сокращенные слова, содержащие основную

суть команды. К таким языкам относились: Autocode, IPL, FLOW-MATIC – языки низкого уровня.

Следующим шагом стало появление языков высокого уровня, наиболее похожих на естественный язык. В них появились некоторые слова из разговорного языка и математические символы.

Языки высокого уровня широко используются, так как позволяют создавать программы длиной до нескольких тысяч строк. Примерами являются COBOL, Pascal, C и другие.

Развитие текстовых языков привело к появлению визуального или графического программирования.

Графический язык программирования — язык, предназначенный для написания программы для компьютера или вычислительного устройства, в котором вместо текстового описания алгоритма работы используется графическое.

Появление визуальных языков программирования приходится на начало 90-х годов. Несмотря на постоянное совершенствование высокоуровневых языков, вопрос повышения удобства и скорости работы остается актуальным по сей день. Это объясняет их популярность и распространение во многие сферы деятельности.

Основное отличие от текстовых языков заключается в том, что программирование осуществляется путем соединения пиктограмм (или, по-другому изображений) элементов языка, в которых уже содержится программный код. Так, например, сложение двух чисел на языке C++ будет выглядеть так:

```
#include <iostream>

using namespace std;

int  main() {
   long a, b;
   cin >> a >> b;
   cout << a + b << endl;
```

```
        return 0;
    }
```

Рис.1

А так на графическом языке:

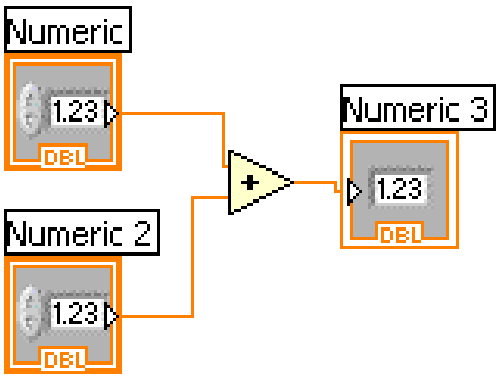


Рис.2

За счет большого количества стандартных функций, библиотек, а также готовых средств визуализации достигается большая наглядность, интерфейс становится интуитивно понятным. Однако, стоит заметить, что из-за этого резко возрастает объем программного кода.

Одним из первых в этом классе был графический язык среды Simulink, входящей в состав Matlab (MathWorks Inc), предназначенный для моделирования динамических систем и устройств, заданных в виде системы блоков, а также язык HP-VEE (Hewlett Packard), ориентированный на работу с измерительными приборами (осциллографами, генераторами, вольтметрами и т.д.). [5]

Платформа с более широким спектром задач была реализована компанией National Instruments (США). В 1986 году Джеффом Кодоски (Jeff Kodosky), Джеймсом Тручардом (James Truchard) и Биллом Новлиным (Bill Nowlin) LabVIEW увидела свет первая версия среды программирования LabVIEW на

графическом языке («G») специально для Apple Macintosh. К сегодняшнему дню существуют версии для Windows, Linux и MacOs.

Расшифровывается LabVIEW как Laboratory Virtual Instrumentation Engineering Workbench, т.е. - лаборатория виртуальных приборов рабочего места инженера. Интерфейс программы («Virtual Instruments» - виртуальный инструмент) подразделяется на две зоны:

1. Front Panel (Передняя панель) – это, по сути, как раз и есть рабочее место, в котором можно изменять входные данные программ, видеть результаты вычислений, показатели приборов и т.д.

2. Block Diagram (Блок-диаграммы) –здесь происходит создание программного кода (визуальное графическое представление).

Так выглядит сложение двух чисел в LabVIEW: слагаемые вводятся переключателем (Knob), а результат выводится на шкалу термометра.

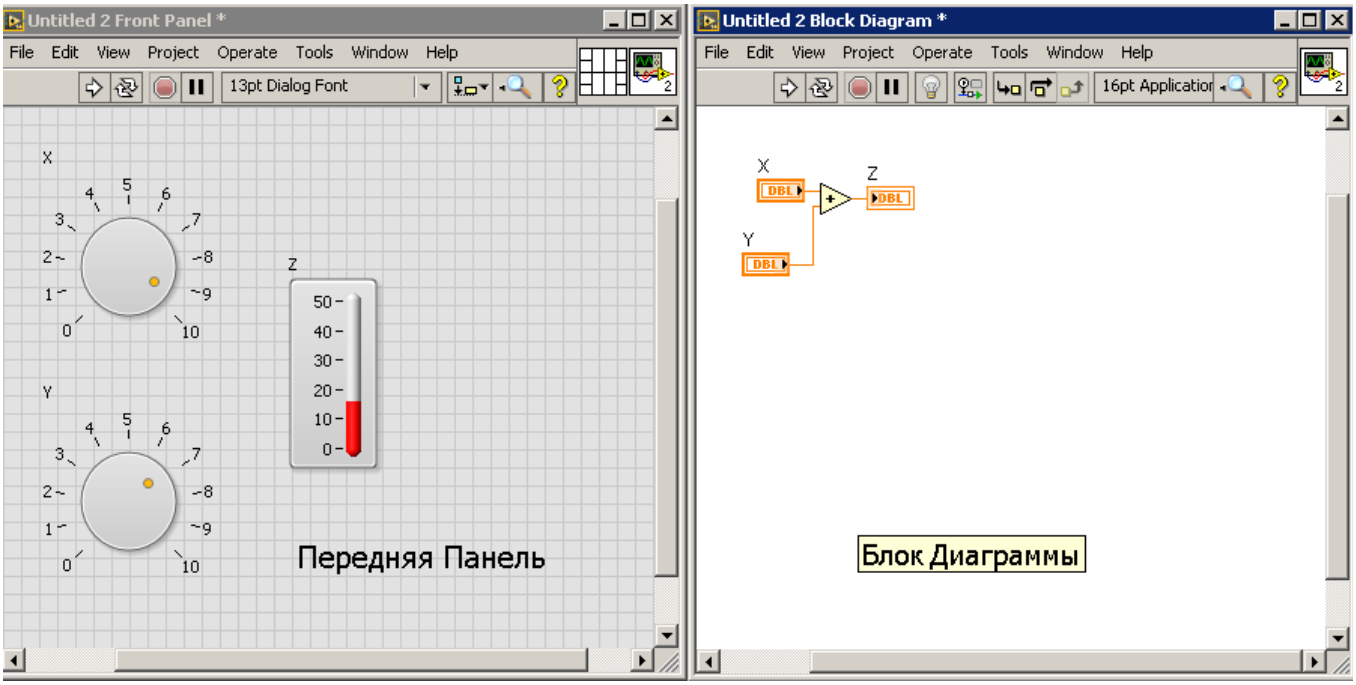


Рис.3

LabVIEW относится к высокоуровневым языкам, хотя имеется возможность включения и «низкоуровневых» модулей в программы. Это может

быть необходимо, например, в случаях, когда требуются сложные или объемные вычисления. Так, используя структуру Formula Node, формула Герона может быть представлена в таком виде:

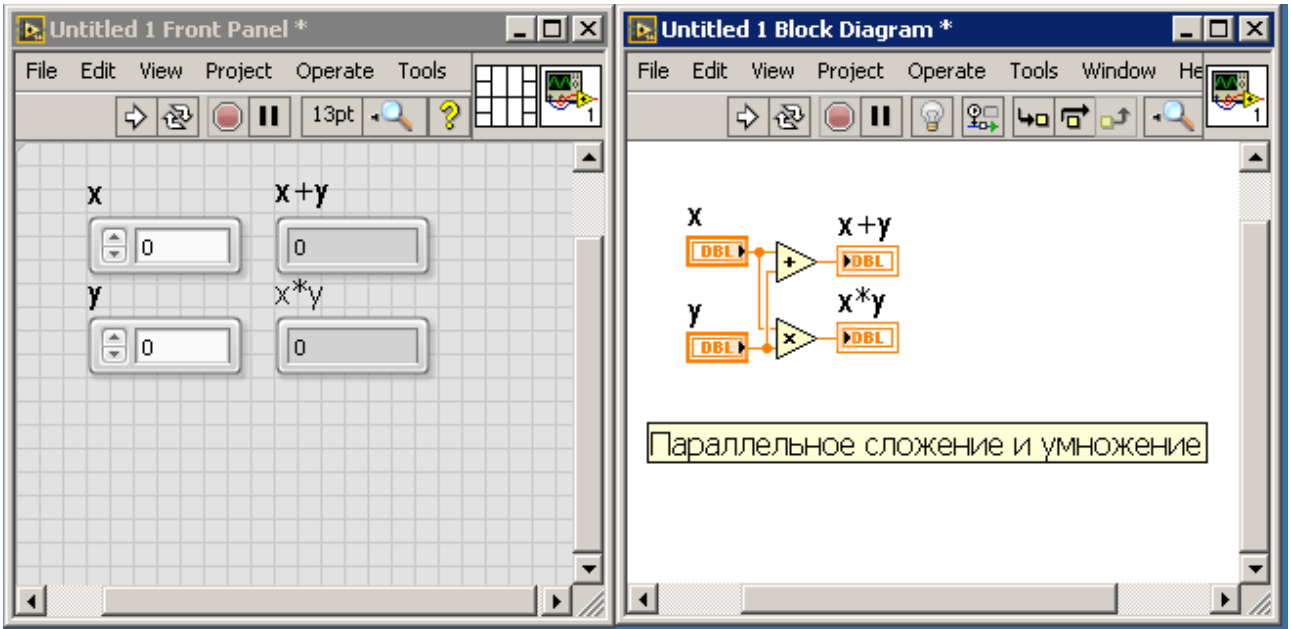
Рис.4

Благодаря возможности согласовывать разные виды представления кодов, Lab VIEW дает пользователю практически безграничные возможности.

Говоря об отличиях LabVIEW от текстовых языков, выделяют две основных особенности:

1. LabVIEW реализует концепцию графического программирования G, поэтому исходный код представляет собой блок-диаграмму (соединенные друг с другом пиктограммы элементов языка), которая затем компилируется в машинный код. Несмотря на такой подход в языке G используются те же конструкции и методы программирования, что и в других языках: типы данных, циклы, переменные, рекурсия, обработка событий и объектно-ориентированное программирование [9].

2. Поддержка выполнения кода, написанного на языке G, в режиме потока данных (потоковое программирование), в то время как традиционные текстовые языки (например, C и C++) обеспечивают выполнение кода в виде последовательности команд. В основе языков потокового программирования (таких как G, Agilent VEE, Microsoft Visual Programming Language и Apple Quartz Composer) лежит концепция потока данных, который и определяет



последовательность выполнения функциональных узлов программы. Таким образом, как только значения параметров поступает на каждый входной терминал узла, происходит выполнение кода.

Рис.5

Проследив историю развития языков программирования можно заметить, что они становятся все более «дружелюбны» к пользователю. В последнее время широкое распространение получили графические языки программирования, благодаря своей наглядности и простоте в использовании становится возможным их изучение в школьной программе.

1.2 Применение LabView для моделирования и прототипирования

Написанная в среде LabVIEW программа, называется виртуальным прибором (VI). Виртуальный прибор может симулировать различные технологические операции, а также, при необходимости, реальные физические приборы.

В LabVIEW имеется две зоны:

1. Front Panel – лицевая панель, на которой с помощью различных элементов (кнопок, переключателей) пользователь осуществляет управление прибором и отслеживает данные благодаря средствам отображения (светодиоды, графики…).

2. Block Diagram – блок диаграмм. Здесь осуществляется непосредственно программирование объектов из лицевой панели с использованием всевозможных функций, которые представляют собой пиктограммы.

Для организации рабочей зоны на передней панели размещают средства ввода данных, которые являются элементами управления (Numeric Control, Knob и т.д.) и средства отображения данных (Numeric Indicator, Graph, LED и др.). Данные так же различаются по типу содержащихся в них значений. Основные из них:

Numeric – численный тип, отображающийся в виде оранжевого терминала для значений с плавающей запятой и голубого терминала для целочисленных значений.

Boolean – логический тип, принимающий значения 1 (True) и 0 (False). Отображается зеленым терминалом.

String – строковый тип, содержащий текстовые значения. Отображается розовым терминалом.

Все элементы лицевой панели отображаются так же и на блоке диаграмм. Здесь начинается создание программы управления данными. С помощью меню функций (Functions) и палитры Programming на блок диаграмм помещаются

узлы, которые имеют поля для ввода/вывода данных и выполняют алгоритмические операции виртуального прибора. Наиболее часто используются узлы из палитр:

Numeric – узлы для осуществления основных математических операций, таких как сложение (Add), умножение (Multiply), генерация случайных чисел (Random Numeric) и многих других;

Comparison – узлы сравнения;

Boolean – логические операции;

Structures – операторы циклов и вариантов. Применяются для повторения операций, определения последовательности или условий выполнения операций;

Array – узлы работы с массивами, позволяют определять размерность (Array Size), максимальные и минимальные значения (Max & Min), сортировать (Sort 1D Array), менять элементы местами (Reverse 1D Array) и другое;

String – узлы работы со строковыми данными, позволяют определить количество символов в строке (String Length), объединять символы в строке (Concatenate String) и др.

После перемещения всех элементов программы, узлы соединяются со средствами ввода/вывода данных проводниками, окрашенными в цвет, соответствующий передаваемым данным.

Тип проводника данных	Одно значение	Одномерный (1D) массив	Двумерный (2D) массив	Цвет
Численный				Оранжевый (с плавающей точкой), Голубой (целочисленный)
Логический				Зеленый
Строковый				Розовый

Рис. 6

В случае, если элементы обладают разными/не подходящими тапами данных, проводники примут вид прерывистой линии, соответственно значения поступать не будут, запустить программу невозможно.

Для запуска и редактирования виртуальных приборов используют инструментальную панель.

Рис. 7

Run – однократно запускает виртуальный прибор в том случае, если программа составлена верно, в противном случае появится кнопка Error List, нажав на которую левой кнопкой мыши появится подсказка с указанием ошибки.

Run Continuously – позволяет осуществить запуск в непрерывном режиме до принудительной остановки.

Abort Execution – немедленная остановка выполнения программы.

Pause – приостановка программы с указанием места остановки.

Text Setting – редактор текста.

Align Objects – выравнивание объектов по осям.

Distribute Objects – выравнивание в пространстве.

Resize Objects – приведение к одному размеру.

Highlight Execution - позволяет в режиме реального времени отслеживать передачу данных на узлы функций.

Context Help – кнопка вызова справки при наведении курсора на элемент программы.

В большинстве текстовых языков порядок выполнения программы определен расположением функций. В LabVIEW дела обстоят несколько иначе: выполнение функции начинается сразу после поступления на поля ввода данных необходимых элементов независимо от их расположения. Таким образом, при расположении в одном виртуальном приборе двух или более

функций, при условии определенности вводимых данных, выполняться они будут одновременно, так как LabVIEW использует потоковую модель обработки данных. Определить очередность выполнения функций можно программными методами, например, используя циклы.

Выводы по первой главе

Анализ литературы по теме исследования показал, что в ходе истории развития программирования произошли кардинальные изменения. На смену текстовым языкам стали приходить графические, которые по многим параметрам не уступают, а по некоторым и превосходят их. Они были предназначены и успешно использовались для сбора данных, моделирования систем автоматизации, автоматического управления, обработки собранных данных и их визуального представления в виде графиков, таблиц, звука, с помощью компьютерной анимации.

Одним из наиболее популярных языков становится LabVIEW. Это среда графического программирования, которую используют технические специалисты, инженеры, преподаватели и ученые по всему миру для быстрого создания комплексных приложений в задачах измерения, тестирования, управления, автоматизации научного эксперимента и образования.

В основе LabVIEW лежит концепция графического программирования - последовательное соединение функциональных блоков на блок-диаграмме. Отсутствие сложного синтаксиса, наглядный интерфейс, удобные системы отладки позволяют расширить круг пользователей средой, осуществляя тем самым принцип: «Программирование без программистов».

LabVIEW открывает широкие возможности для разработки комплексов тестирования, ввода данных, измерения, анализа, симуляции оборудования, физических, математических и технологических закономерностей.

ГЛАВА 2. МЕЖПРЕДМЕТНЫЕ СВЯЗИ В КОНТЕКСТЕ ПРОГРАММИРОВАНИЯ НА УРОКАХ ТЕХНОЛОГИИ

2.1 Методические рекомендации по планированию и проведению занятий с использованием LabView

Практические задания в среде LabVIEW являются частью рабочей программы образовательной области «Технология», составлены в соответствии с Федеральным законом от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации» (в действующей редакции), Федеральным государственным образовательным стандартом основного общего образования (в действующей редакции), примерной рабочей программой по курсу «Технология» основного общего образования для организаций общего образования, разработанной авторским коллективом Казакевич В.М., Пичугина Г.В., Семенова Г.Ю. (м Казакевич В.М., Пичугина Г.В., Семенова Г.Ю. Технология: программа 5-8 8(+), 9 классы / Москва, Издательский центр «ВЕНТАНА-ГРАФ», 2015), запросом современного общества на обеспечение непрерывности и преемственности образования, и реализующую приоритетные направления развития.

Практические задания могут быть использованы во всех разделах курса технологии.

Практические задания в среде LabVIEW в 8 классе направлены на достижение следующих целей:

- **формирование целостного мировоззрения**, соответствующего современному уровню развития науки и общественной практики, благодаря развитию представлений о новейших средствах производства;
- **понимание роли технологических процессов в современном мире**;
- **совершенствование общеучебных и общекультурных навыков работы с информацией** в процессе систематизации и обобщения имеющихся и

получения новых знаний, умений и способов деятельности в области технологии, информатики и информационно-коммуникационных технологий, математики и физики;

– **развитие навыков самостоятельной учебной деятельности** школьников (учебного проектирования, моделирования, исследовательской деятельности и т.д.);

– воспитание **ответственного и избирательного отношения к информации** с учетом правовых и этических аспектов ее распространения, воспитанию стремления к продолжению образования и созидательной деятельности с применением средств ИКТ;

– освоение и систематизация знаний, относящихся к математическим, физическим объектам технологии; построение описаний объектов и процессов, позволяющих осуществлять их компьютерное моделирование; средствам моделирования; информационным процессам в биологических, технологических и социальных системах;

– овладение умениями строить математические объекты информатики, в том числе логические формулы и программы на формальном языке, удовлетворяющие заданному описанию; использовать общепользовательские инструменты и настраивать их для нужд пользователя;

– развитие алгоритмического мышления, способностей к формализации, элементов системного мышления;

– воспитание культуры проектной деятельности, в том числе умения планировать, работать в коллективе; чувства ответственности за результаты своего труда, используемые другими людьми; установки на позитивную социальную деятельность, недопустимости действий, нарушающих правовые и этические нормы;

– приобретение опыта создания, редактирования, оформления, сохранения, передачи объектов различного типа с помощью современных программных средств; построение компьютерных моделей, коллективной

реализации технологических проектов, преодоление трудностей в процессе интеллектуального проектирования, информационной деятельности в различных сферах, востребованных на рынке труда.

Содержание практических заданий

Таблица 1

Знакомство со средой программирования	
Понятие о среде программирования, назначении, основных функциях; виртуальных приборах, подпрограммах, проектах, знакомство с лицевой панелью, блок-диаграммой. Обзор палитр элементов управления, функций и инструментов.	*Аналитическая деятельность:* – выявлять отличия между лицевой панелью и блоком диаграмм; – выявлять отличия между палитрами элементов управления, функций и инструментов. *Практическая деятельность:* – уметь создавать виртуальные приборы; – уметь добавлять элементы на лицевую панель и блок-диаграмму;
Зависимость силы тока от напряжения	
Понятие графика и диаграммы. Понятие о типах данных, знакомство с операторами, переменными и константами	*Аналитическая деятельность:* – выявлять различия между графиками и диаграммами; – анализировать данные для создания графиков и диаграмм; – анализировать условия задачи для создания графиков и диаграмм; *Практическая деятельность:* – уметь создавать и пользоваться графиками.

Подсчет времени на последовательное и параллельное выполнение технологического процесса	
Понятие типов алгоритмических структур, циклической структуры, условного оператора.	*Аналитическая деятельность:* – выявлять различие структур; – анализировать условия задачи для создания алгоритма; *Практическая деятельность:* – уметь правильно использовать различные алгоритмические структуры; – уметь выбирать нужный тип алгоритмической структуры; – уметь пользоваться справочной системой и режимом отладки.
Графическое решение квадратных уравнений с алгебраической проверкой	
Понятие таблицы и массива в среде программирования, основные характеристики, свойства.	*Аналитическая деятельность:* – выявлять различие таблицы и массива; – выявлять различия между графиками и диаграммами; – анализировать таблицы и массивы для создания графиков и диаграмм; – анализировать условия задачи для создания таблицы и массива; *Практическая деятельность:* – уметь создавать и пользоваться таблицами и массивами;

	– уметь на основании таблиц строить графики и диаграммы;
Расчет суточной нормы потребления калорий	
Понятие линейного алгоритма	– *Аналитическая деятельность:* – выявлять отличия операторов, переменных и констант; – анализировать условия задачи для создания виртуального прибора. – *Практическая деятельность:* – уметь создавать формулы
Расчет энергии	
Понятие блока формул «Formula Node»	*Аналитическая деятельность:* – выявлять отличия между графическими и текстовыми элементами программы *Практическая деятельность:* – уметь создавать подпрограммы и подключать их к виртуальным приборам; – уметь создавать проект и подключать в него виртуальные приборы и подпрограммы;
Перевод числа из десятеричной системы счисления	
Понятие логических типов данных, основные характеристики и свойства	*Аналитическая деятельность:* – анализировать условия задачи; – выявлять наиболее рациональные способы решения задач; – выявлять различие типов данных; *Практическая деятельность:*

	– уметь решать задачу разными способами
Комбинаторика	
Понятия локальные и глобальные переменные	*Аналитическая деятельность:* – выявлять закономерности в условиях задач; – уметь правильно использовать различные алгоритмические структуры; *Практическая деятельность:* – уметь правильно использовать локальные и глобальные переменные;
Составление последовательности Фибоначчи	
Понятие кластер, основные характеристики и свойства	*Аналитическая деятельность:* – выявлять различия между кластером и массивом; – анализировать условия задачи для создания кластера. *Практическая деятельность:* – уметь объединять массивы в кластер.
Анализ прибыльности предприятия	
Понятие массива, основные функции для работы с массивами	– *Аналитическая деятельность:* – выявлять различие таблицы и массива; – анализировать таблицы и массивы для создания графиков и диаграмм; – анализировать условия задачи для создания таблицы и массива;

	– *Практическая деятельность:* – уметь создавать и пользоваться таблицами и массивами;

Место задач в учебном процессе

Таблица 2

№	Название раздела	Практическое задание	Количество часов
1.	Методы и средства творческой и проектной деятельности	Знакомство со средой программирования	1
2.	Производство	Зависимость силы тока от напряжения	1
3.	Технология	Подсчет времени на последовательное и параллельное выполнение технологического процесса	2
4.	Техника	Графическое решение квадратных уравнений с алгебраической проверкой.	2
5.	Технологии обработки пищевых продуктов	Расчет суточной нормы потребления калорий	1
6.	Технологии получения,	Расчет энергии	1

	преобразования и использования энергии		
7.	Технологии получения, обработки и использования информации	Перевод числа из десятеричной системы счисления.	2
8.	Технологии получения, обработки и использования информации	Комбинаторика	1
9.	Технологии растениеводства	Составление последовательности Фибоначчи	1
10.	Социальные технологии	Анализ прибыльности предприятия	2
ИТОГО:			14

Количество часов может варьироваться в зависимости от уровня подготовки обучающихся.

Планируемые результаты освоения практических заданий в 8 классе

Таблица 3

Задание	**Содержание**	**Кол-во часов**	**Планируемые предметные результаты (ученик научится)**
Знакомство со средой программирования	Правила техники безопасности. Назначение, основные функции, обзор среды программирования.	1	– выполнять правила ТБ и поведения – различать лицевую панель и блок диаграмм – пользоваться справочной

Задание	**Содержание**	**Кол-во часов**	**Планируемые предметные результаты (ученик научится)**
	Демонстрация примеров. Понятие лицевой панели, блок-диаграммы, узлов, проводников.		системой и режимом отладки
Зависимость силы тока от напряжения	Понятие подпрограммы и проекта. Построение графиков зависимости силы тока от напряжения. Понятие о типах данных, знакомство с операторами, переменными и константами	1	– создавать проект, виртуальный прибор (ВП) и подпрограмму – выявлять отличия операторов, переменных и констант – строить графики зависимости
Подсчет времени на последовательное и параллельное выполнение технологического процесса	Понятие типов алгоритмических структур, циклической структуры, условного оператора.	*2*	– строить математическую модель задачи – выделять исходные данные и результаты, выявлять соотношения между ними – определять основные этапы решения задач – строить алгоритм с использованием циклических алгоритмических конструкций – решать задачи с использованием циклических алгоритмических конструкций
Графическо	Извлечение квадратного	2	– выявлять различия между

Задание	Содержание	Кол-во часов	Планируемые предметные результаты (ученик научится)
е решение квадратных уравнений с алгебраической проверкой	корня. Таблица квадратов чисел и их квадратных корней. График квадратичной функции, свойства.		графиками и диаграммами; – анализировать таблицы и массивы для создания графиков и диаграмм; – создавать и пользоваться таблицами и массивами; – на основании таблиц строить графики и диаграммы.
Расчет суточной нормы потребления калорий	Понятие типов алгоритмических структур, линейного алгоритма.	1	– строить математическую модель задачи – строить алгоритм с использованием линейных алгоритмических конструкций – решать задачи с использованием циклических алгоритмических конструкций
Расчет энергии	Понятие блока формул «Formula Node»	1	– анализировать условия задачи для выбора способа составления формул – использовать синтаксис при составлении формул
Перевод числа из десятеричной системы счисления	Понятие логических типов данных, основные характеристики и свойства. Деление с остатком. Массив логических данных.	2	– выявлять различие типов данных – переводить числа из одной системы счисления в другую программным и математическим способом

Задание	**Содержание**	**Кол-во часов**	**Планируемые предметные результаты (ученик научится)**
Комбинаторика	Понятия локальные и глобальные переменные. Расчет возможного количества комбинаций.	1	– правильно использовать локальные и глобальные переменные – рассчитывать факториал числа
Составление последовательности Фибоначчи	Понятие кластер, основные характеристики и свойства. Создание кластера массивов.	1	– выявлять различия между кластером и массивом – анализировать условия задач для создания кластера
Анализ прибыльности предприятия	Сортировка чисел. Сравнение чисел.	2	– анализировать условия задачи для создания таблицы и массива; – применять функции для работы с массивами

Планируемые личностные результаты:

- понимание роли технологических процессов в современном мире;
- владение первичными навыками анализа и критичной оценки получаемой информации;
- развитие чувства личной ответственности за качество выполняемой работы;
- способность увязать учебное содержание с собственным жизненным опытом;
- готовность к повышению своего образовательного уровня и продолжению обучения;

- способность и готовность к общению и сотрудничеству со сверстниками и взрослыми в процессе образовательной, общественно-полезной, учебно-исследовательской, творческой деятельности;
- способность и готовность к принятию ценностей здорового образа жизни**;**
- становление самоопределения в выбранной сфере будущей профессиональной деятельности, планирование образовательной и профессиональной карьеры, осознание необходимости общественно полезного труда;
- формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками;
- проявление технико-технологического и экономического мышления при организации своей деятельности;
- самооценка готовности к предпринимательской деятельности в сфере технологий, к рациональному ведению домашнего хозяйства;
- планирование образовательной и профессиональной карьеры;
- диагностика результатов познавательно – трудовой деятельности по принятым критериям и показателям;

Планируемые метапредметные результаты:

- владение общепредметными понятиями «объект», «система», «модель», «алгоритм», «исполнитель» и др.;
- владение информационно-логическими умениями: определять понятия, создавать обобщения, устанавливать аналогии, классифицировать, самостоятельно выбирать основания и критерии для классификации, устанавливать причинно-следственные связи, строить логическое рассуждение, умозаключение (индуктивное, дедуктивное и по аналогии) и делать выводы;
- владение умениями самостоятельно планировать пути достижения целей; соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с

изменяющейся ситуацией; оценивать правильность выполнения учебной задачи;

- владение основами самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности;
- владение основными универсальными умениями, такими как: постановка и формулирование проблемы; поиск и выделение необходимой информации, применение методов информационного поиска; структурирование и визуализация информации; выбор наиболее эффективных способов решения задач в зависимости от конкретных условий; самостоятельное создание алгоритмов деятельности при решении проблем творческого и поискового характера;
- владение технологией как основным методом приобретения знаний: умение преобразовывать объект из чувственной формы в пространственно-графическую или знаково-символическую модель; умение строить разнообразные информационные структуры для описания объектов; умение «читать» таблицы, графики, диаграммы, схемы и т. д., самостоятельно перекодировать информацию из одной знаковой системы в другую; умение выбирать форму представления информации в зависимости от стоящей задачи, проверять адекватность модели объекту и цели моделирования;
- ИКТ-компетентность — широкий спектр умений и навыков использования средств информационных и коммуникационных технологий для сбора, хранения, преобразования и передачи различных видов информации, навыки создания личного информационного пространства (обращение с устройствами ИКТ; создание письменных сообщений; создание графических объектов; коммуникация и социальное взаимодействие; поиск и организация хранения информации; анализ информации).

Межпредметные связи:

Таблица 4

№ №	Раздел Технологии	Практическое задание	Межпредметные связи			
			Технология	Математика	Физика	Информатика
1	Методы и средства творческой и проектной деятельности	Знакомство со средой программирования	Современное производство	Переменные, константы		Типы данных, виртуальный прибор
2	Производство	Зависимость силы тока от напряжения	Измерительные приборы	Функции и графики	Закон Ома для участка цепи	Массивы, графики
3	Технология	Подсчет времени на последовательное и параллельное выполнение технологического процесса	Классификация технологий	Переменные, константы	Понятие времени	Технология Dataflow, алгоритмические структуры, строковые типы данных
4	Техника	Графическое решение квадратных уравнений с алгебраической проверкой.		Квадратное уравнение	Траектория	Графики
5	Технологии обработки пищевых продуктов	Расчет суточной нормы потребления калорий	Физиология питания	Переменные, константы		Технология Dataflow
6	Технологии получения, преобразования и использован	Расчет энергии	Ядерная и термоядерная энергия		Ядерная и термоядерная энергия	Включение текстовых элементов программы в LabVIEW

	ия энергии					с помощью Formyla Node
7	Технологии получения, обработки и использования информации	1.Перевод числа из десятеричной системы счисления. 2.Комбинаторика	Кодирование информации	Понятие факториала		Системы счисления Кластер
8	Технологии растениеводства	Составление последовательности Фибоначчи	Числа Фибоначчи и растения	Арифметическая прогрессия		Цикл For, сдвиговый регистр
9	Социальные технологии	Анализ прибыльности предприятия	Маркетинговый анализ	Задачи на проценты		Массивы

2.2. Проектирование межпредметных практических заданий в среде LabView

Задача №1. Знакомство со средой программирования

Цель: создать первое приложение, освоить технологии графического программирования, научиться изменять и редактировать свойства графических элементов управления и индикации, использовать циклы типа While-Do в программе.

Теоретическая часть:

В основу программирования в LabVIEW положено понятие «Виртуальные приборы» (Virtual Instruments, VI). Любая программа представляет собой такой виртуальный прибор, в котором имеется «лицевая панель» (Front Panel) и «схема» (Block Diagram). На лицевой панели, как и положено, располагаются элементы управления программой – кнопки, графики, выключатели и т. д. Блок-схема – это и есть сама программа. При создании программы используется такое понятие, как *«поток данных»* (Data Flow). Суть его в том, что все элементы программы, представленные графически, соединены между собой связями (проводами), по которым и происходит передача данных.

Практическая часть:

Запустите программу National Instruments LabVIEW в результате откроется окно, как на рис. 8

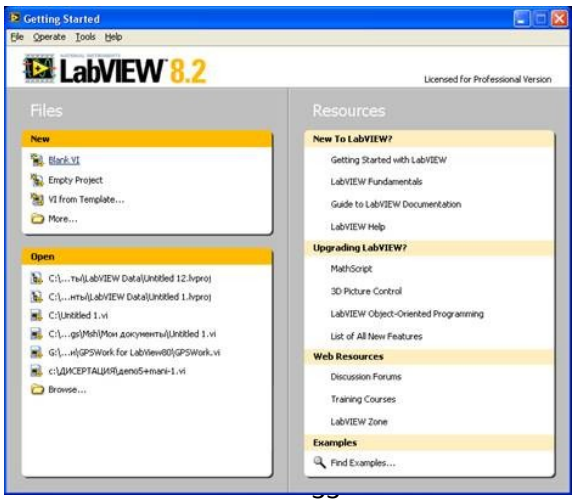


Рис. 8

В появившемся окне выберите опцию **Blank VI.** Если панель управления неактивна, то ее следует вызвать через основное меню **Window** –> **Show Block Diagram**. Для этого проделайте следующие шаги:

1. Используйте указатель мыши в виде стрелки, установите его на интерфейсную панель инструмента **Numeric Control,** который находится на панели управления **(Controls)** –> **Modern** –> **Numeric.**

2. Подпишите его как "Число А". Для этой цели подведите указатель к метке, кликните и просто наберите необходимый текст.

3. Установите, по аналогии с предыдущими двумя шагами, и подпишите, как "Число В" еще один **Numeric Control**. Это будут поля ввода параметров.

4. Для отображения результата поместите на интерфейсную панель **Numeric Indicator**, который также находится на панели управления **(Controls)** –> **Modern** –> **Numeric.** Подпишите его, как "Результат". Должно получиться приблизительно так, как показано на рис. 9

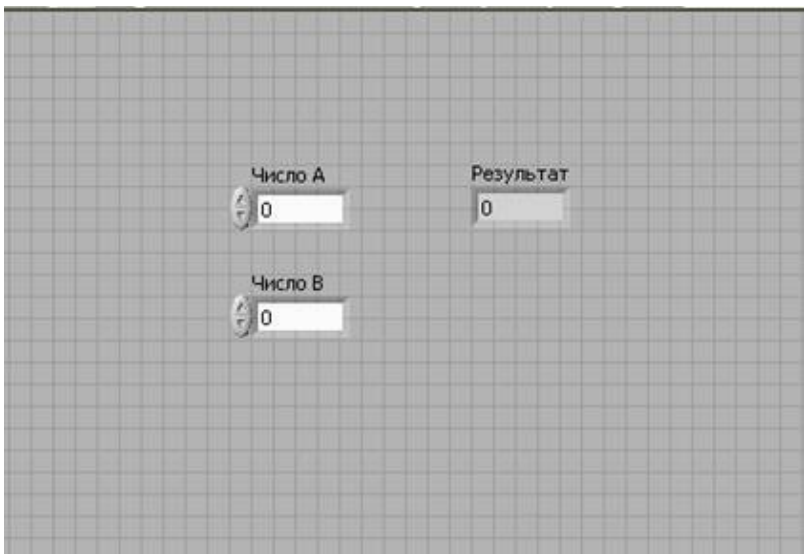


Рис. 9

5. Теперь перейдем к основной части работы, а именно к графическому программированию. В отличие от других языков программирования, таких, как, например, Borland Delphi или Microsoft Visual C++, здесь не нужно писать ни единой строки текстового кода, реализующего определенный алгоритм.

Создав визуальный интерфейс с двумя полями ввода чисел и одним цифровым индикатором, поставим и реализуем задачу, например, перемножения этих чисел. Для этого необходимо перейти в так называемое окно построения диаграмм, где видны три иконки (терминала), которые соответствуют полям ввода чисел и индикатору. Реализация простого или сложного алгоритма будет сводиться к элементарной последовательности действий, т. е. к установке необходимых иконок, которые выполняют ту или иную функцию и служат для связи (соединения) их между собой.

При умножении чисел необходимо вызвать функциональную панель (**Functions**) и перетянуть треугольную иконку, соответствующую операции перемножения, в окно редактирования диаграмм. Она находится в **Functions** –> **Programming** –> **Numeric** –> **Multiply**.

6. Теперь остается соединить необходимые контакты соединительной катушкой (**WiringTool**). Подводим указатель мыши к пиктограмме первого числа до тех пор, пока он не превратится в катушку, затем нажимаем левую клавишу мыши и, не отпуская ее, соединяем второй конец линии с одним из контактов пиктограммы перемножения. Для изменения направления связи потребуется еще один промежуточный щелчок левой клавишей мыши.

7. Повторяем эти действия и для второго числа, аналогично соединяя выход иконки суммирования с входом цифрового индикатора. Должна получиться функциональная диаграмма ("текст" программы) (рис. 10)

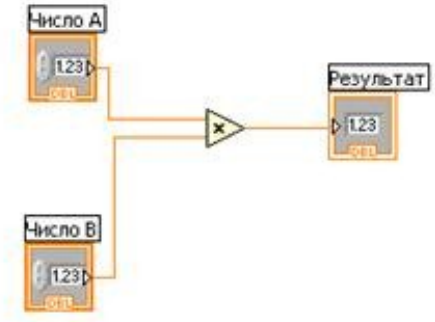


Рис. 10

Итак, программа написана. Теперь остается запустить ее на выполнение и убедиться в ее работоспособности.

8. Переходим на интерфейсную панель, запускаем программу на выполнение в циклическом режиме, нажав левой клавишей мыши на кнопку циклического запуска **(Run Continuously).**

Меняем значения полей ввода чисел, используя клавиатуру или мышь.

Для останова выполнения программы следует воспользоваться кнопкой линейки управления**(Abort Execution).**

Сделаем еще один шаг и сохраним наше первое элементарное приложение на диске. Сохранение LabVIEW-программы поисходит аналогично записи, например документа в Microsoft Word или Excel.

Для первого сохранения программы необходимо выбрать в меню **File** пункт **Save.** В появившемся диалоговом окне необходимо выбрать или создать желаемый каталог (папку), ввести имя файла и подтвердить ввод. Записанный нами файл сохранился с расширением **.vi**(Virtual Instrument - виртуальный инструмент) и имеет вид **<имя файла>.vi**.

Файлы с расширением **vi** переносимы между различными платформами, будь то Windows или Unix/Linux.

Изменим внешний вид наших графических объектов. Для этого подводим указатель в виде стрелки на объект, соответствующий "Числу A", и нажимаем правую кнопку мыши.

В появившемся контекстном меню выбираем опцию замены **(Replace).** Далее входим в подменю **Numeric** и там выбираем шарообразную ручку управления **(Knob).**

Изменим размер ручки. Меняя положение указателя, увидим, что в четырех точках он меняет вид со стрелки на окружности. В этот момент, нажав и удерживая левую кнопку мыши, изменяем вид ручки.

Теперь разместим метку "Число А", выделив и переместив ее в необходимую позицию.

Изменим атрибуты ручки "Число А". В выпадающем меню, нажав правую клавишу мыши, выберем изменение визуальных свойств объекта (**Visible Items**), а в них **Ramp**.

Для точного позиционирования ручки или отображения значения выберем еще и свойство**Numeric Display**, которое также находится в **Visible Items**. Разместите его на панели по вашему усмотрению.

Проделайте аналогичные шаги и для остальных элементов интерфейса. В результате должен получиться такой внешний вид, как на рис. 11.

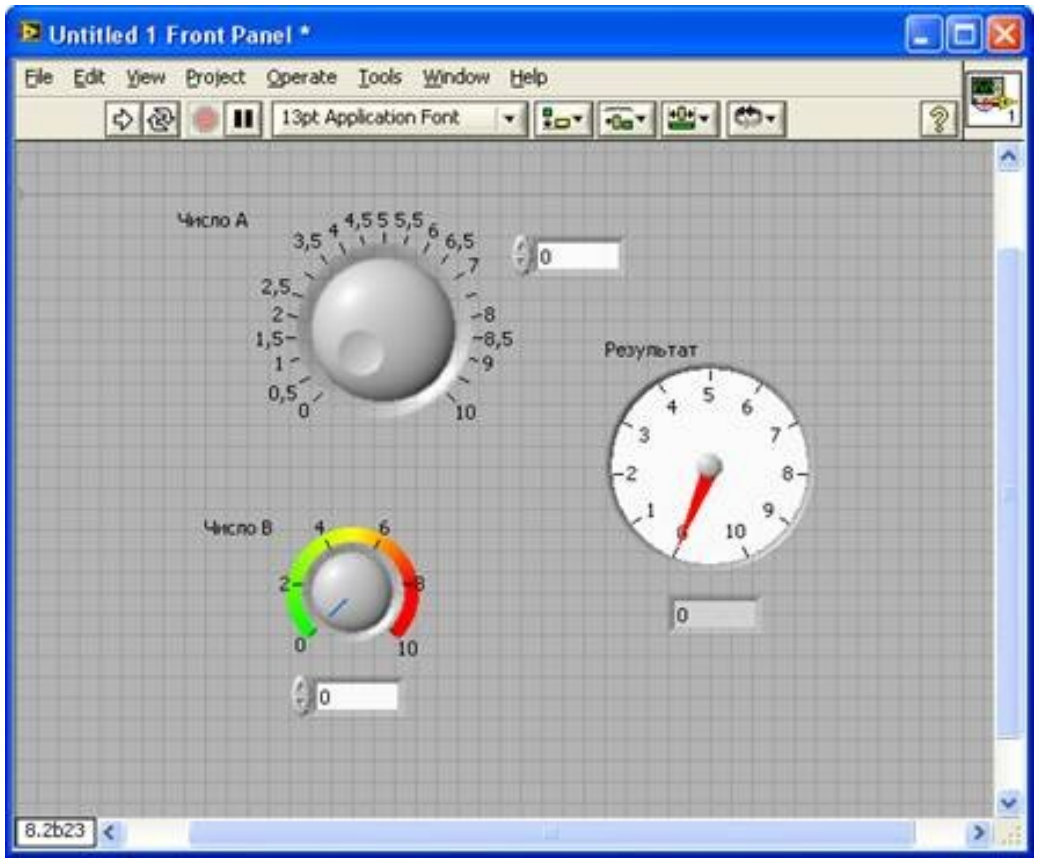


Рис. 11

Контрольные вопросы:

1. Как начать создавать приложение?
2. Как выполняются арифметические операции?
3. Как создаются индикаторы и управляющие элементы?

Задача №2. Зависимость силы тока от напряжения

Цель: Алгебраически проверить зависимость между силой тока, напряжением на однородном участке электрической цепи и сопротивлением этого участка, построить график зависимости.

Теоретическая часть:

Закон Ома лежит в основе науки под названием электротехника.

Т. к. напряжение в законе рассматривается на концах проводника и учитывается сопротивление самого проводника, то закон применим именно к участку цепи, т. е. к какой-либо его части.

$$I = \frac{V}{R};$$ где:

I – сила тока, А;

V – напряжение, В;

R – сопротивление, Ом.

При работе с законом Ома следует понимать, что он выполним отдельно для каждого рассматриваемого участка цепи с различными значениями входящих в него параметров.

Практическая часть:

В предложенной программе будем изменять напряжение U и наблюдать за изменением значений силы тока I при неизменном сопротивлении R.

Для подачи напряжения создадим на лицевой панели не Numeric Control, а Knob, для наглядности. Следующим шагом поместим XY Graph, в котором будет отображаться зависимость, по оси абсцисс напряжение U, по оси ординат

ток I. Оси необходимо подписать. И последним этапом работы с лицевой панелью будет установка Stop Button, при нажатии на которую программа должна завершить работу.

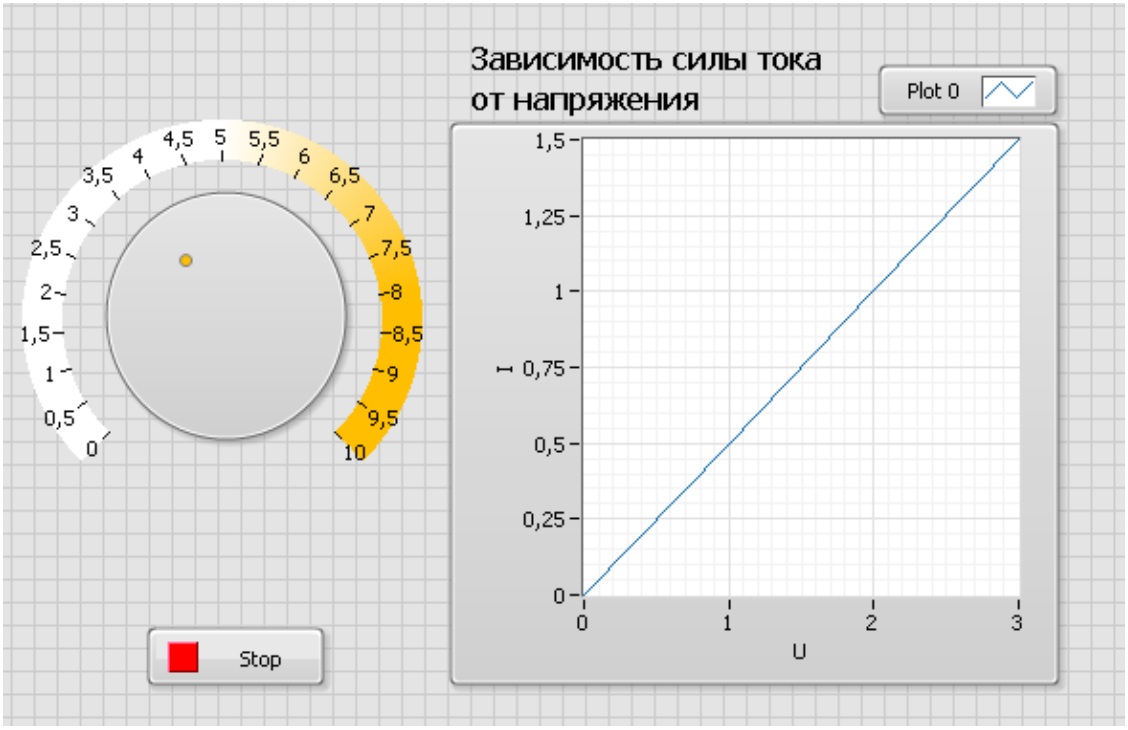


Рис. 12

Перейдем на панель диаграмм. Во первых необходимо реализовать формулу: $I=\frac{U}{R}$. Работу будем выполнять в цикле For Loop. Так как сопротивление будет неизменным, на нижнем входе деления создадим константу. Для возможности изменения напряжения без прерывания цикла, количество на вход N подадим Knob, а счетчик итераций I на первый вход деления. К выходам цикла подведем значения U и результата от деления I. Объединим оба значения функцией Bundle и подадим значения на график.

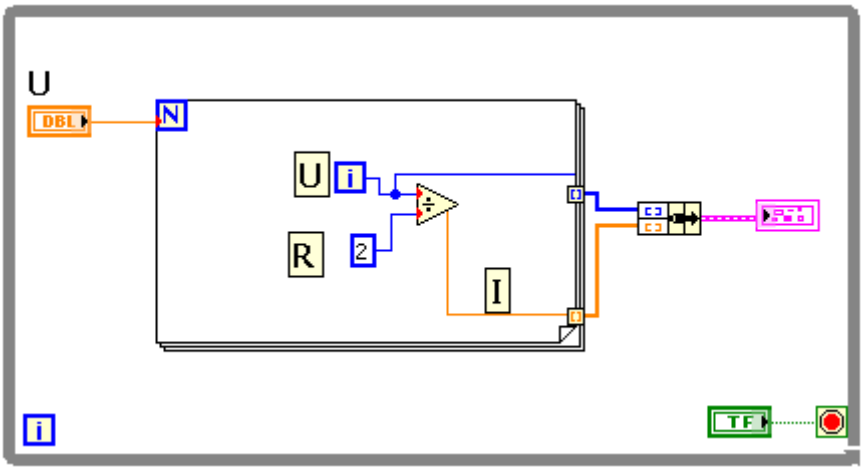


Рис. 13

Контрольные вопросы:

1. Как зависит сила тока в проводнике от напряжения на концах проводника?
2. Какой вид имеет график зависимости силы тока от напряжения?
3. Измените программу таким образом, чтобы на графике отображалась зависимость тока I от сопротивления R.

Задача №3. Подсчет времени на последовательное и параллельное выполнение технологического процесса

Цель: Смоделировать технологический процесс при параллельном и последовательном выполнении операций. Рассчитать временные затраты.

Теоретическая часть:

При последовательной форме движения обработка партии деталей на каждой последующей операции начинается лишь после того, как вся партия прошла обработку на предыдущей операции.

При параллельном движении передача предметов труда (деталей) на последующую операцию осуществляется поштучно, либо транспортной партией сразу после обработки на предыдущей операции.

Практическая часть:

Представим, что для изготовления определенного изделия к детали A необходимо присоединить деталь B, а затем C. На выходе должно получаться изделие ABC.

Рассмотрим последовательное выполнения процесса. Для этих целей в LabView используется структура Flat Sequence, по своему виду напоминающая кадры кинопленки. До тех пор, пока не выполнятся все операции в одном «кадре», не начнется выполнения следующего.

Условия задачи требуют соединения деталей A, B и C, для их ввода на лицевой панели создадим String Control, складывать которые мы будем функцией Concatenate String. Так как технологический процесс предполагает производства нескольких изделий, каждое сложение поместим в цикл и зададим количество итераций.

Для расчета времени, поместим всю программу в цикл While Loop. Затраченное время будет равно разнице времени окончания и начала процесса. В качестве часов будем использовать Tick Count из палитры Timing.

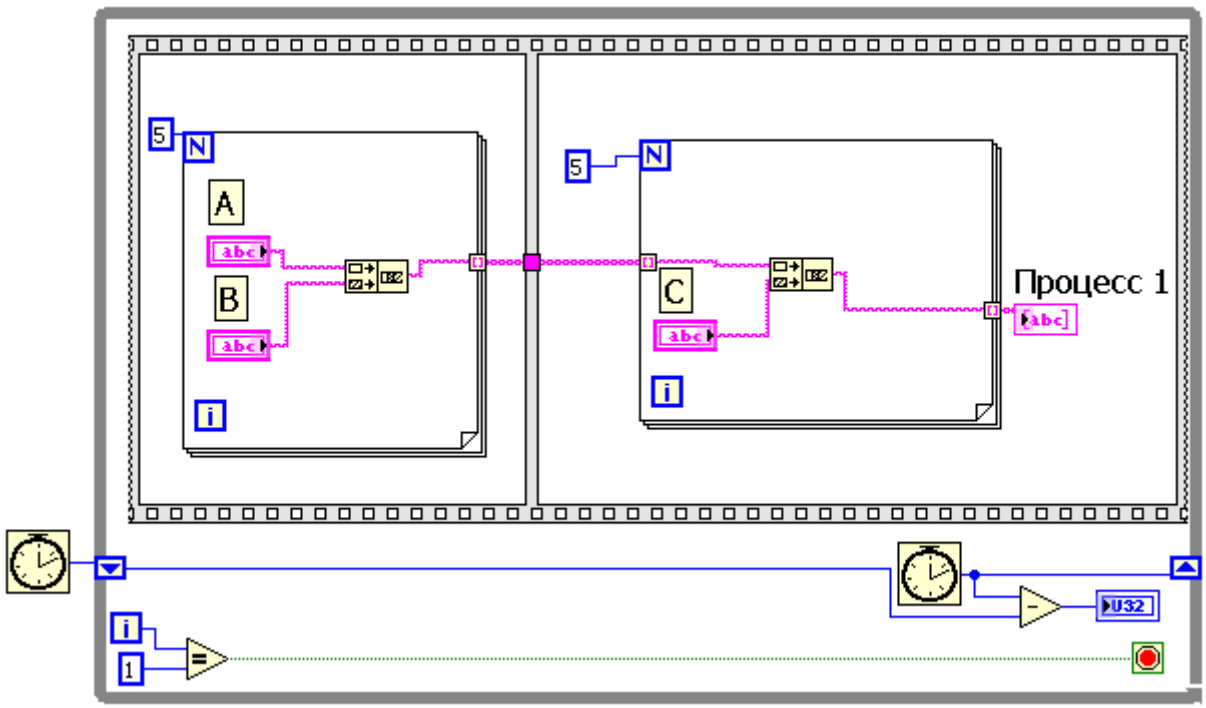


Рис. 14

Аналогичным образом смоделируем второй технологический процесс, только во втором случае передача значений на второй этап будет проходить сразу после выполнения первого, без задержек.

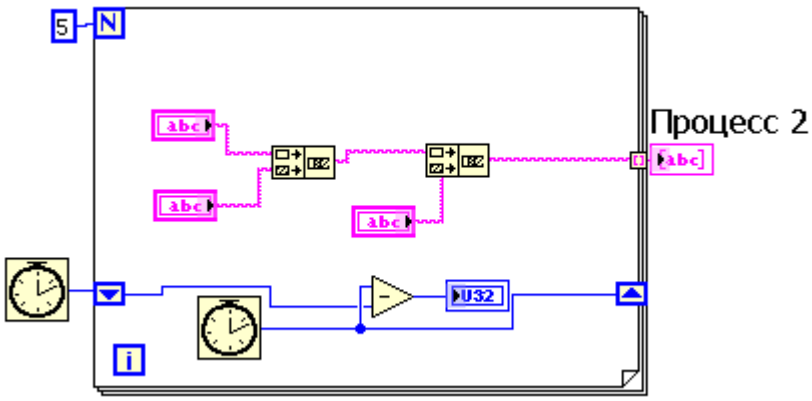


Рис. 15

Для проверки программы и большей наглядности запустим ее, используя функцию Highlight Execution.

Как вы видите, результаты процесса, записанные в массив идентичны в обоих вариантах, различно лишь время выполнения. Важно понимать, что время отображается в миллисекундах.

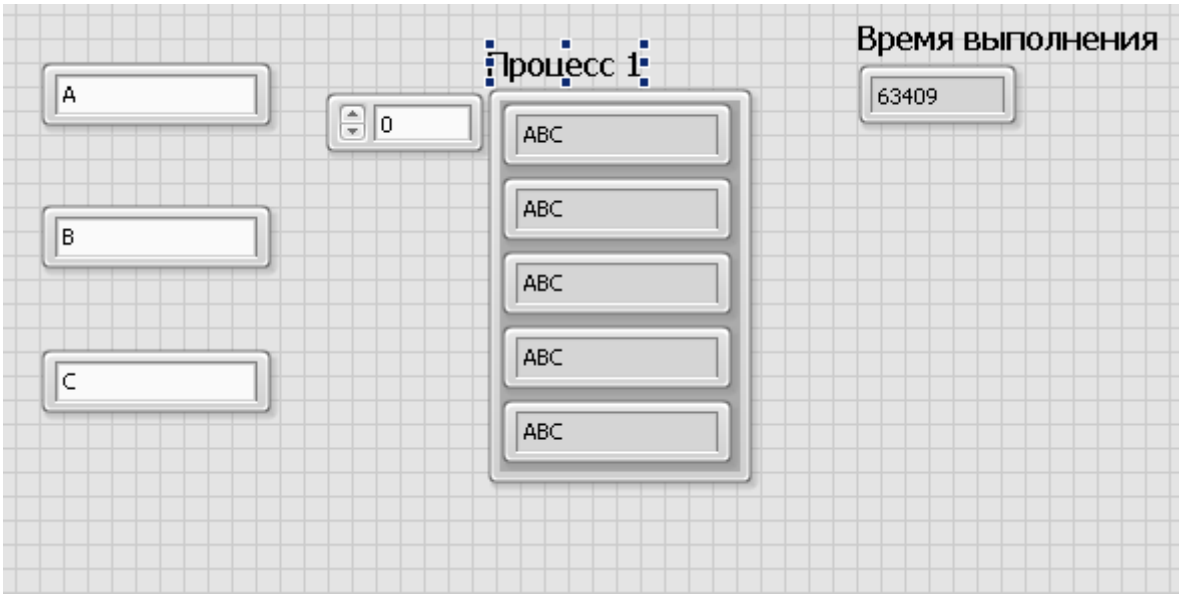


Рис. 16

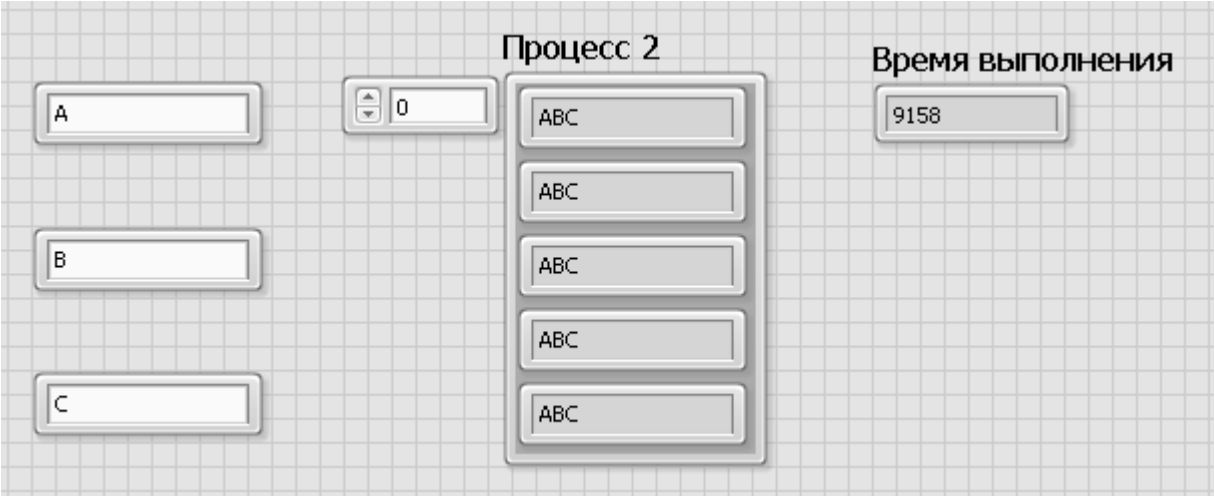


Рис. 17

Контрольные вопросы:

1. Какой из производственных процессов выполнился быстрее? Почему?
2. Приведите примеры производственных процессов, для которых наиболее рациональным будет последовательный тип?
3. Опишите ключевые особенности структуры Flat Sequence.

Задача №4. Графическое решение квадратных уравнений с алгебраической проверкой.

Цель: Создать программы для алгебраического и геометрического решения квадратных уравнений.

Теоретическая часть:

Квадратным уравнением называют уравнение вида $ax^2+bx+c=0$, где a, b, c — любые числа (коэффициенты), причём $a\neq 0$.

Используя знания о некоторых функциях и их графиках, возможно решать квадратные уравнения. При том, существует как минимум два вида решения: алгебраическим и графическим методом. Рассмотрим эти методы на примере одного квадратного уравнения: $x^2-2x-3=0$.

1) Для начла приведем данное уравнение к виду: $x^2=2x+3$. Затем, в одной системе координат построим графики функций: $y=x^2; y=2x+3$.

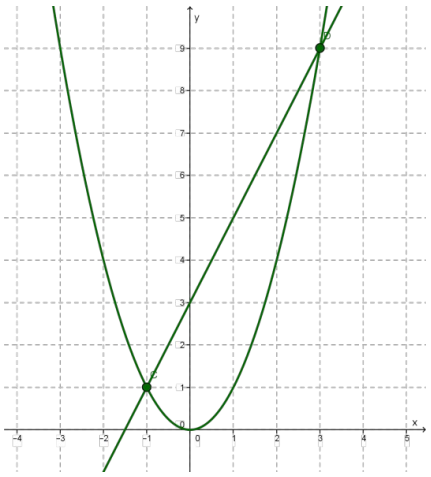


Рис. 18

Графики пересекаются в двух точках: C(−1;1) и D(3;9). Корнями уравнения служат абсциссы точек C и D, значит, $x_1=-1; x_2=3$.

2) Чтобы решить уравнение алгебраическим способом, можно воспользоваться формулой дискриминанта:

$D=b^2-4ac$.

Однако сам дискриминант не является корнем уравнения, а лишь указывает на их количество:

$D>0$ – уравнение имеет два корня;

$D=0$ – уравнение имеет один корень;

$D<0$ – уравнение не имеет корней.

Сами корни уравнения находятся по формуле:

$$x_{1,2}=\frac{-b\pm\sqrt{b^2}}{2a}$$

Практическая часть:

Чтобы построить графики функций: $y=x^2; y=2x+3$ необходимо вместо x подставлять любые числа. Для записи чисел создадим массивы $x.1$ и $x2$в которые вручную запишем числа от -3 до 3. Можно выбрать и любой другой числовой ряд, главное использовать и отрицательные и положительные значения для большей точности отображения графиков. Затем создадим массивы $y1 u y2$, в которые вставим числа, получившиеся от подстановки. В этом варианте решения просчитаем самостоятельно.

После, с помощью элемента Bandle объединим $x u y$, относящиеся к одним и тем же функциям и подадим значения на *XY Graph*.

Блок диаграмма будет выглядеть так:

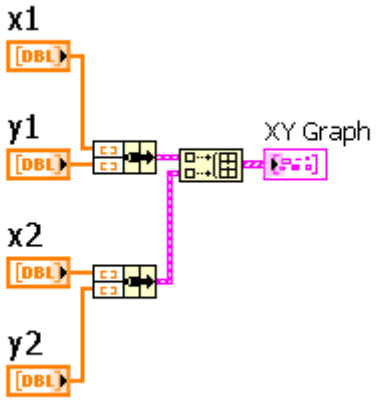


Рис. 19

Оба графика должны отображаться на одной диаграмме:

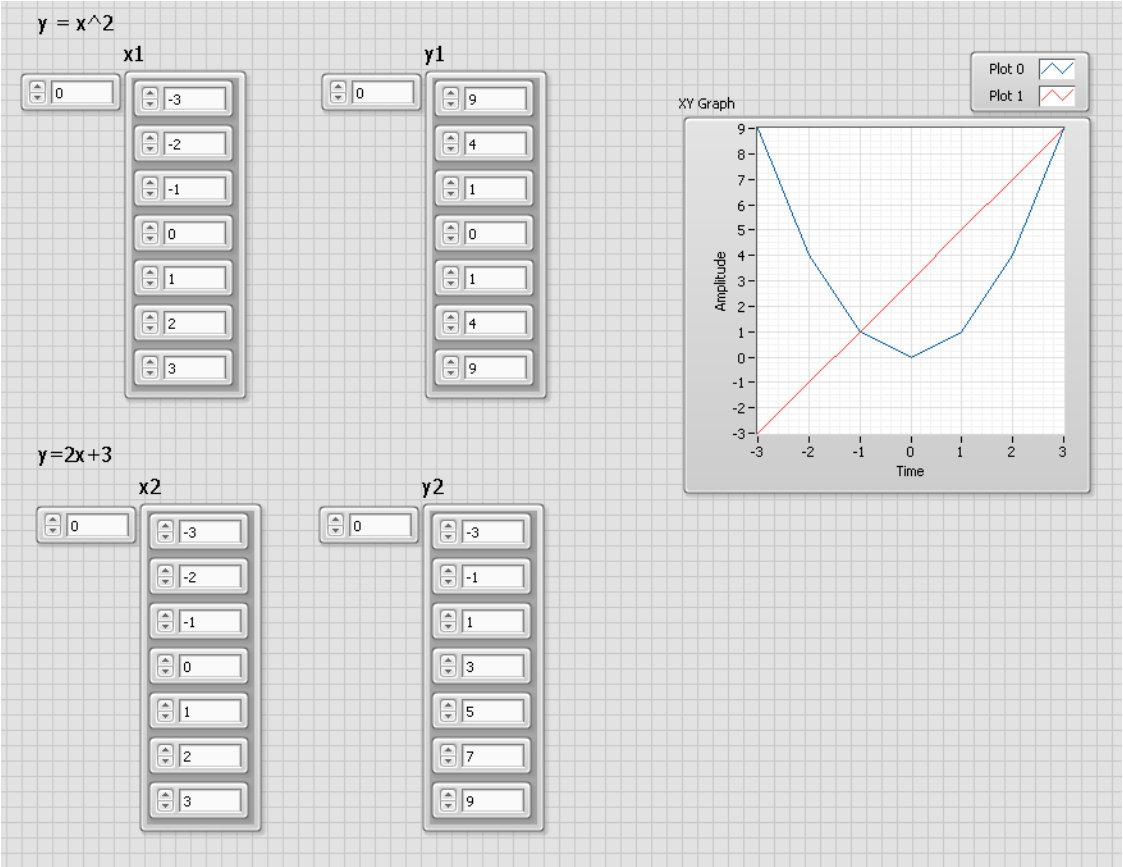


Рис. 20

Обратим внимание на то, что графики пересекаются, а точки пересечения и будут корнями уравнения $x^2-2x-3=0$.

В нашем случае уравнение имеет два корня: $x_1=-1; x_2=3$

Для решения уравнения алгебраическим методом создадим на лицевой панели три Numeric Cjntrol, которые будут соответствовать переменным $a, b u c$. Три Numeric Indicator, в которые будем записывать результаты вычисления дискриминанта (D) и корней $x_1 u x_2$.

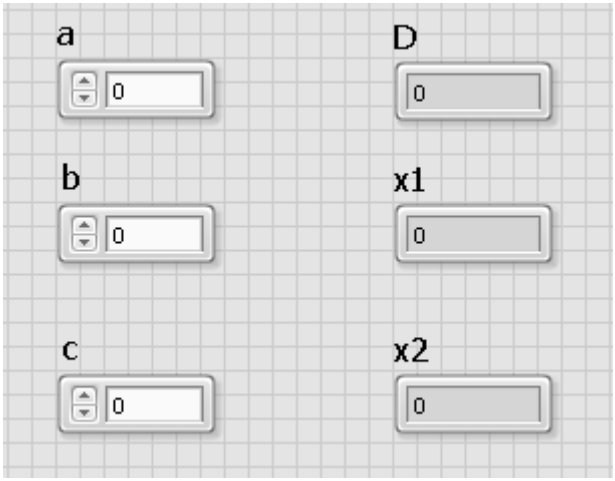


Рис. 21

Перейдем к составлению уравнений. На панели Block Diagram с помощью палитры Numeric составляем уравнение: $D=b^2-4ac$.

На этом этапе можно создать локальные переменные для значений a,b,c и D. Они будут нужны при нахождении корней, чтобы не загромождать программу переплетениями проводников, однако, этот этап не обязателен.

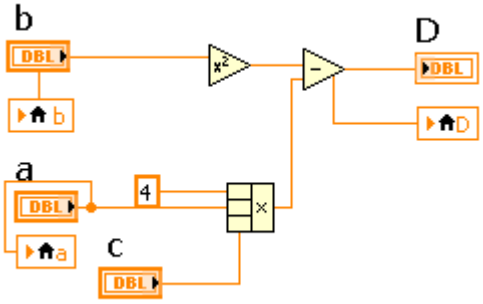


Рис. 22

Вычислив дискриминант, приступаем к нахождению корней, тем же образом составляя уравнения: $x_{1,2}=\frac{-b\pm\sqrt{b^2}}{2a}$.

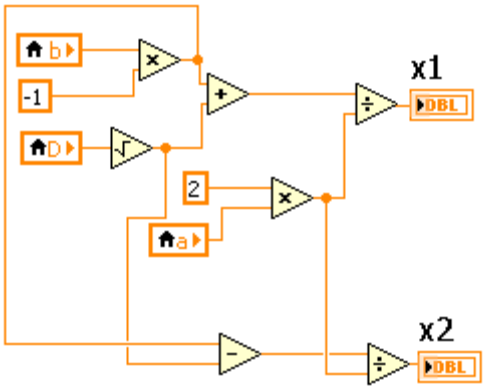


Рис. 23

Осталось подставить значения переменных в программу и сравнить результаты.

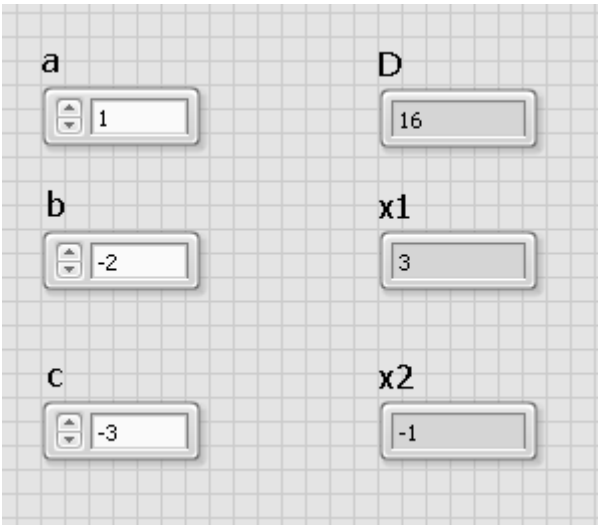


Рис. 24

Контрольные вопросы:

1. В чем заключается графический и аналитический методы решения квадратных уравнений?
2. Для каких целей используются локальные переменные?
3. Будет ли работать программа для аналитического решения предложенного уравнения при других значениях переменных?

Задача №5. Суточная потребность в калориях

Цель: Составить программу для расчета суточной нормы калорий, исходя из индивидуальных особенностей человека.

Теоретическая часть:

Под калорийностью в физиологии питания понимают меру обеспечения организма энергией.

Калории служат нашему организму для реализации множества функций: дыхания, перекачивания крови, умственной или физической работы и даже для отдыха и сна.

Как недостаток, так и переизбыток калорий может навредить организму, особенно в подростковый период, так как организм еще не сформирован в полной мере.

В разных можно встретить противоречивые рекомендации, касающиеся нормы калорий. Чаще всего в них указываются усредненные показатели, но согласитесь, не могут два человека одного пола и возраста, но с разным родом деятельности довольствоваться одним и тем же запасом энергии. Поэтому, высчитывать калорийность необходимо индивидуально. Для этого существует формула:

$$Ккал = \left((вес * 10) + (рост * 6{,}25) - (возраст * 5) - \frac{+161\,ж}{-5\,м} \right) * A;$$ где

A- коэффициент активности (приведен в таблице)

Таблица 5

Коэффициент активности	Физическая активность
1.2	Нагрузка отсутствует или минимальная
1.38	Тренировки средней тяжести 3 раза в неделю
1.46	Тренировка средней тяжести 5 раз в неделю
1.55	Интенсивные тренировки 5 раз в неделю
1.64	Тренировки каждый день
1.73	Интенсивные тренировки каждый день или по 2 раза в

	день
1.9	Ежедневная физическая нагрузка + физическая работа

Практическая часть:

Используя математические функции, составьте формулу для расчета индивидуальной потребности в калориях.

Контрольные вопросы:

1. Для чего необходимо знать суточную норму калорий?
2. Для составления формулы, какими значениями вы пользовались Numeric Control или Numeric Constant. Почему?
3. Рационально ли использование только Numeric Control?

Задача №5. Расчет энергии

Цель: Актуализация знаний об энергии из курса физики. Знакомство с функцией Formula Node.

Теоретическая часть:

На занятиях по физике, математике и некоторым другим предметам постоянно приходится сталкиваться с разнообразными формулами, так же, как и инженерам на предприятиях. В настоящее время, в условиях развития компьютерной техники, самостоятельно каждый раз вычислять тот или иной показатель является нецелесообразным. Мы уже убедились, что любую формулу, подходящую под разные параметры, можно реализовать в Lab VIEW, используя инструменты палитры Numeric. Однако, бывают случаи, когда формулы в таком виде занимают достаточно большое пространство рабочего поля и мешают наглядности и удобству работы с программой.

На примере всем известной формулы проведем способ оптимизации рабочего пространства.

Из школьного курса физики вы узнали, что энергия – это физическая величина, показывающая, какую работу может совершить тело.

$$E = mc^2;$$ где

E – Энергия [Дж]

m – масса [кг]

c – скорость света [м/с].

Практическая часть:

Для вычисления формулы на лицевой панели создадим два Numeric Control, в которых будем менять значения для m и c, а также Numeric Indicator, куда будет записываться результат вычисления.

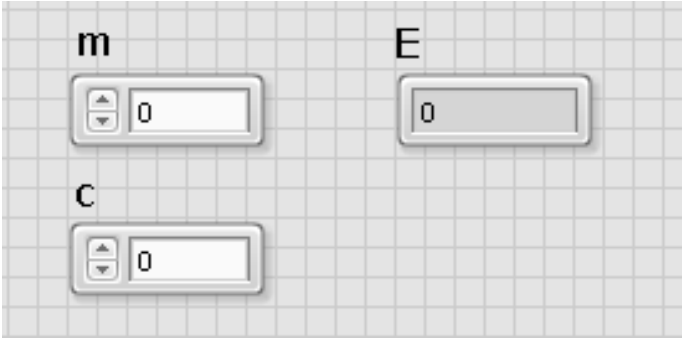


Рис. 25

Перейдем на панель программирования. В меню выберем палитру Structures и функцию Formula Node помещаем в рабочую зону. Для записи формулы будем использовать тот же синтаксис, что и в языке Си для арифметических операций, вот некоторые из них:

Таблица 6

сложение	+
вычитание	-
умножение	*
деление	/
возведение в степень	pov(x/n)
квадратный корень	sqrt

На левой границе структуры создадим терминалы для ввода данных, для этого необходимо нажать правой кнопкой мыши в любом месте границы и выбрать Add Input. В появившемся окошке прописываем название переменной, которая будет подаваться на вход (m или c) и соединяем с соответствующим

Numeric Control. Для вывода данных проделываем те же операции, выбрав Add output. Меняем параметры переменных и запускаем программу.

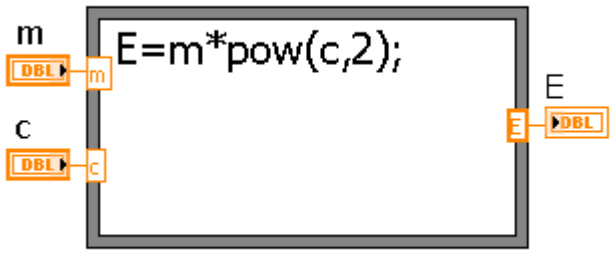


Рис. 26

Контрольные вопросы:

1. В каких сферах человеческой жизни используется понятие энергия?
2. В каких случаях требуется применение структуры Formula Node?
3. Любую ли формулу можно записать, пользуясь данной структурой?

Задача №6. Перевод числа из десятеричной системы счисления

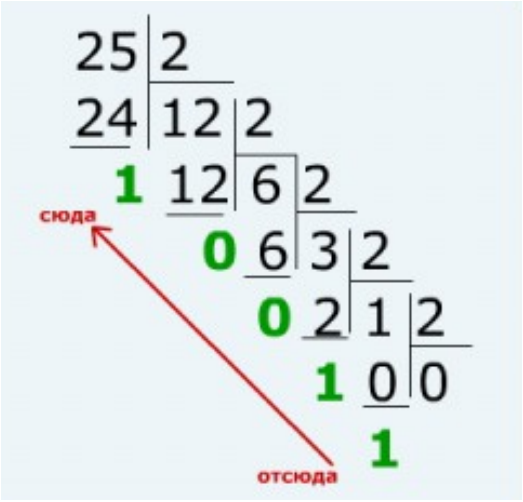
Цель: Рассмотреть способы перевода десятичного числа в двоичное. Ознакомиться с циклами While Loop и For Loop.

Теоретическая часть:

Система счисления – это способ отображения числа.

Наиболее распространенными являются десятеричная, которой мы пользуемся в повседневной жизни, и двоичная, используемая в вычислительной технике, представляющая собой последовательность 0 и 1.

Вводимое число на компьютер в десятичной форме, программным способом преобразуется в двоичный код. Одним из способов такого преобразования является деление десятичного числа на 2 до тех пор, пока



деление нацело будет невозможно, то есть, пока делимое не будет равно 1. Остатки от деления, записанные в обратной последовательности и будут являться двоичным кодом. Так, десятичное число 25_{10} равно двоичному 11001_2.

Рис. 27

Практическая часть:

Для реализации перевода чисел из одной системы счисления в другую, на передней панели создадим Numeric Control, в поле которого будем вводить десятичное число. На панели диаграмм в палитре Numeric выберем Quotient & Remainder. Данная функция позволяет не только делить число, но и выводить остаток от деления, что нам и нужно. Делить всегда будем на 2, поэтому к нижнему входу подсоединяем Numeric Constant, а к верхнему Numeric Control. Так как делить нам придется не один раз, создадим несколько Quotient & Remainder, на вход каждого последующего будем подавать второй выход предыдущего, а для первого выхода создадим Numeric Indicator, для записи остатков от деления.

Запустим программу для уже известного нам числа 25. Так как для получения двоичного числа нам пришлось поделить 5 раз, то 5 Quotient & Remainder нам и потребуется.

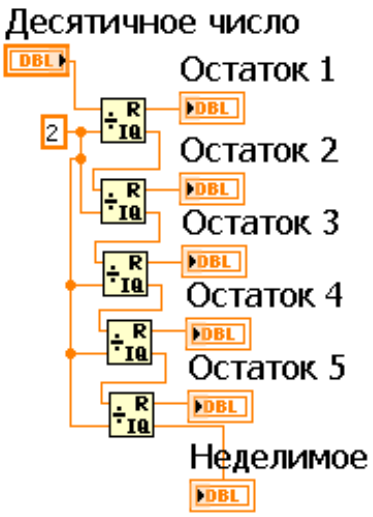


Рис. 28

Результат перевода мы видим на индикаторах лицевой панели, которые для наглядности можно сразу переставить в обратном порядке.

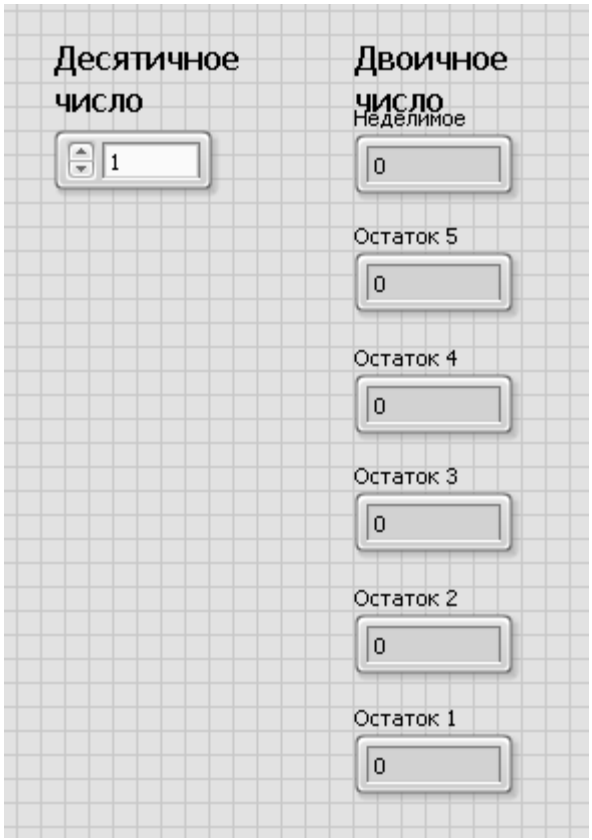


Рис. 29

Однако, что будет, если число будет слишком большим, а последовательность Quotient & Remainder будет слишком длинной и не удобной для просмотра?

В таком случае логичнее всего воспользоваться циклами. Преобразуем исходную программу так, как показано на рисунке:

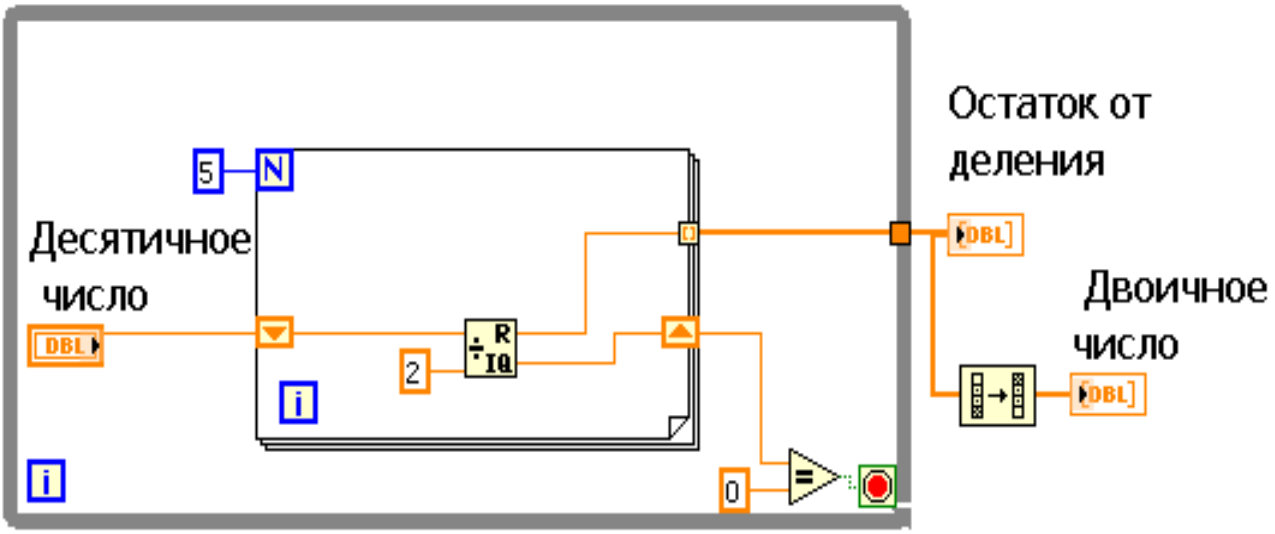


Рис. 30

Деление в цикле может происходить любое количество раз. На лицевой панели остаток от деления будет представлен в виде массива, перевернув который с помощью функции Reverse 1D Array отобразится двоичное число. Но, к сожалению, не зная точное количество операций придется самостоятельно смотреть длину последовательности 1 и 0 в остатке от деления (последняя единица всегда будет концом деления).

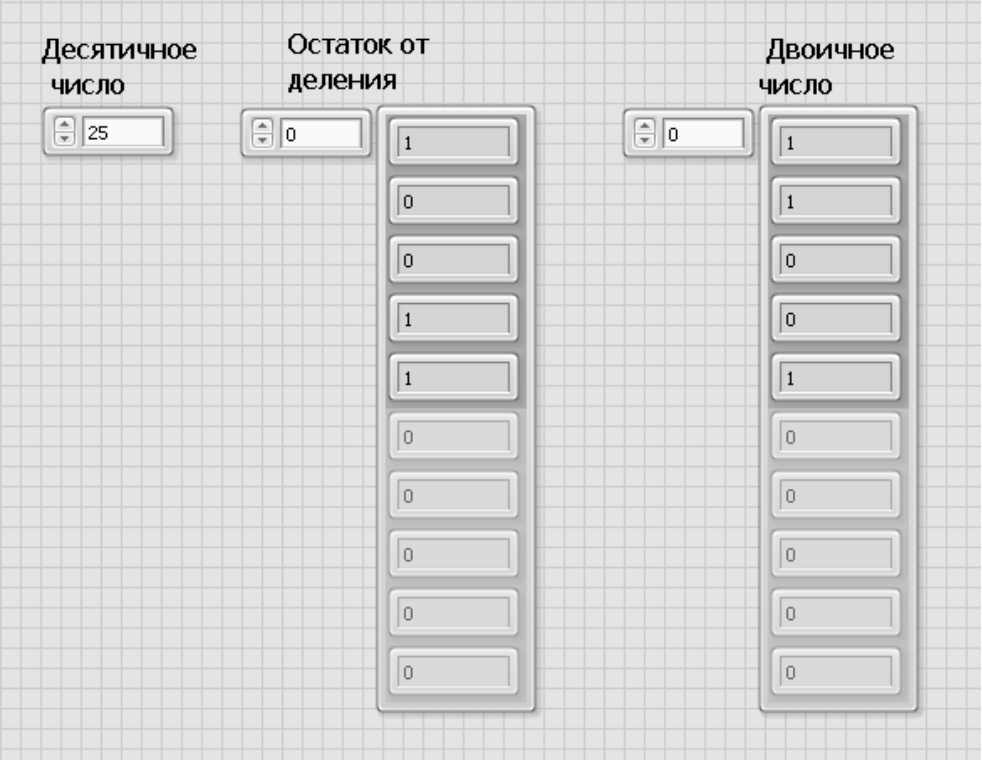


Рис. 40

Существует и третий способ преобразования. Для его реализации выберем функцию Num to Array в палитре Boolean. На вход подадим двоичное число, а на выходе создадим индикатор, который будет представлять собой массив, состоящих из «лампочек» (булевых значений), включенное состояние которых = 1, а выключенное = 0. Полученное значение, как и в предыдущем варианте необходимо перевернуть для реального отображения двоичного кода.

Рис. 41

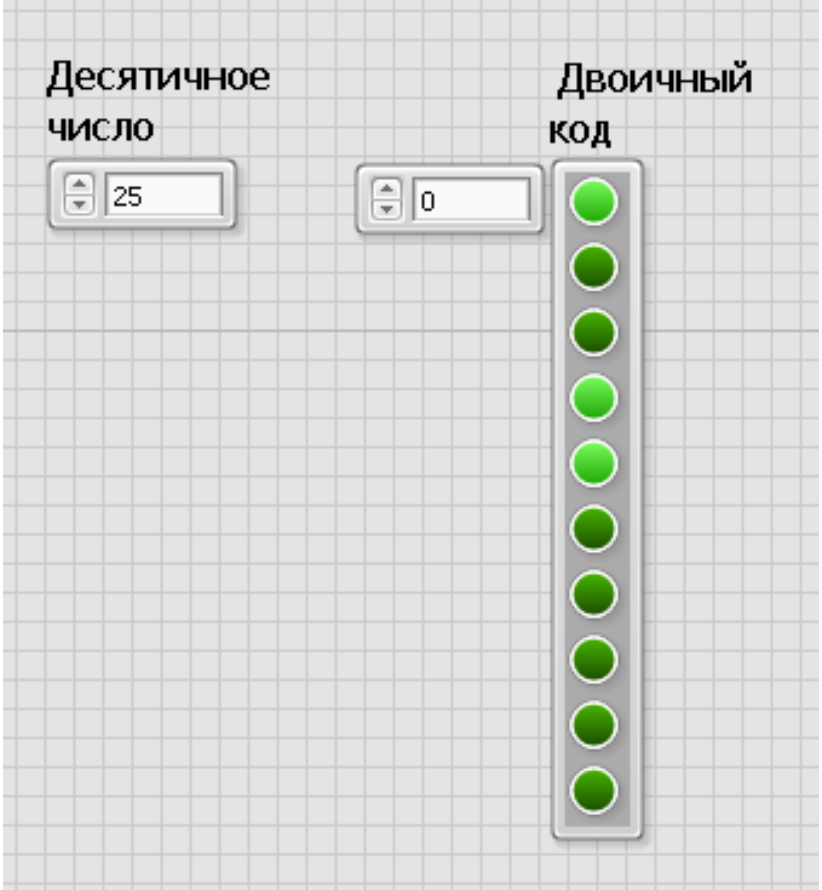


Рис. 42

Контрольные вопросы:

1. Для чего необходимо умение переводить десятичные числа в двоичную систему счисления?
2. Какой из предложенных способов наиболее рационален.
3. Как можно дополнить программу, чтобы она стала удобней.

Задача №7. Комбинаторика

Цель: Составить программу для автоматического расчета факториала числа.

Теоретическая часть:

Комбинаторика – область математики, в которой изучаются вопросы о количестве различных комбинаций из определенного числа элементов.

Используется комбинаторика в математике, информатике, биологии, а также дизайне и творчестве.

Для определения количества комбинаций можно перебирать все элементы. Например, определяя сколько трехзначных чисел можно составить из цифр 1,2 и 3, не сложно составить комбинации: 123, 132, 213, 231, 312, 321. Всего получилось 6 комбинаций. Однако в случаях, когда количество элементов достаточно большое, простая перестановка займет много времени. В таких случаях используют факториал.

$$n! = 1 * 2 * 3 \ldots * n;$$

где n – количество элементов.

Таким образом, если нам требуется узнать, сколько существует комбинаций для получения полотна из 4 цветных лент, то необходимо рассчитать 4!, то есть 1*2*3*4 = 24 комбинации.

Практическая часть:

Факториал числа предполагает многократное умножения последовательности чисел. Для этой цели удобно использовать цикл For Loop. Второй множитель с каждым циклом должен увеличиваться на 1,. По этому, к счетчику итераций i прибавим единицу и подадим на вход умножения. На второй ход изначально подадим так же единицу, так как с нее начинается любое вычисление факториала, зададим ее с помощью константы (Numeric Constant). Затем, для изменения ее значений в процессе выполнения циклов, воспользуемся функцией Shift Register.

Количество циклов N будет равно тому числу, факториал которого требуется рассчитать. Блок диаграмм должен выглядеть так:

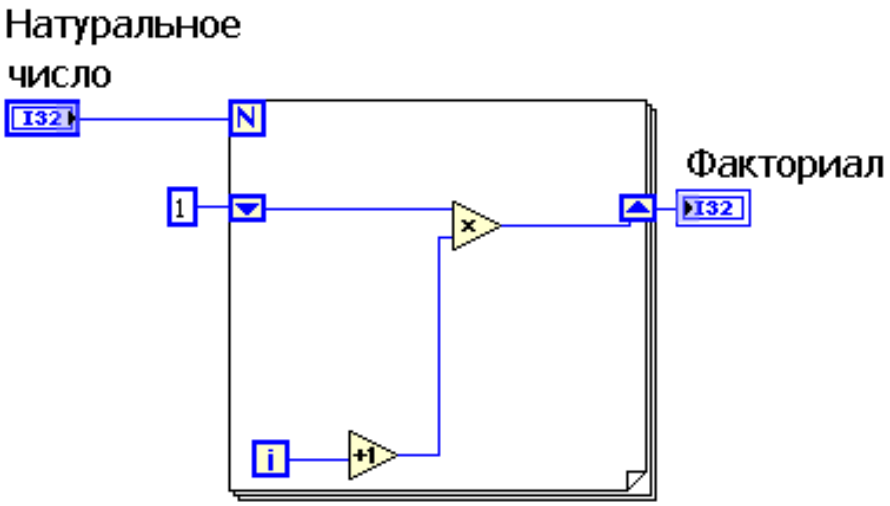


Рис. 43

Лицевая панель:

Рис. 44

Контрольные вопросы:

1. Приведите примеры из жизни, в которых применимы способы расчета комбинаций факториалом.

2. Факториал какого наибольшего числа возможно рассчитать в программе?

3. Каким образом можно изменить программу для получения больших значений.

Задача №8. Составление последовательности Фибоначчи

Цель: Реализовать систему для автоматического расчёта последовательности Фибоначчи.

Теоретическая часть:

Последовательность Фибоначчи, представляет собой последовательность чисел, при чем каждое последующее число является суммой двух предыдущих. Начинается последовательность обычно с 1 и 0, либо с двух единиц.

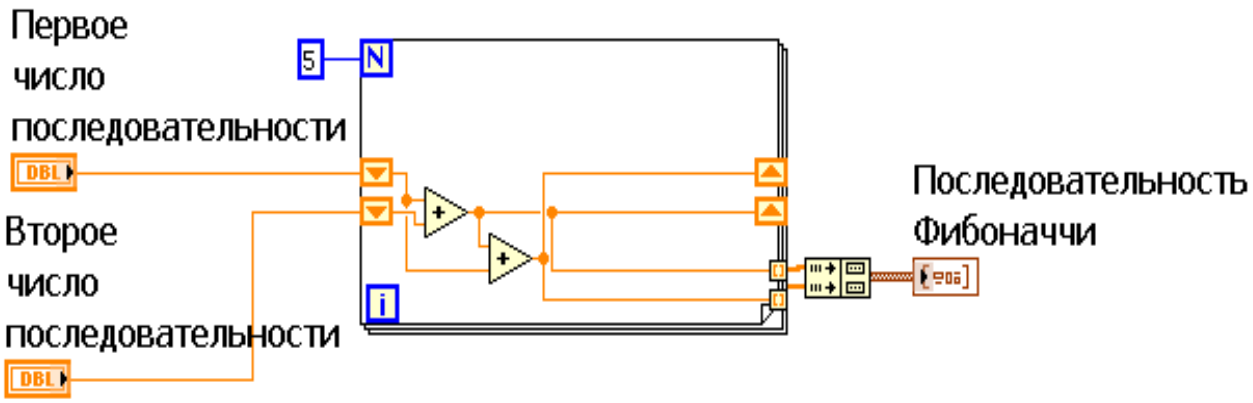
Как вы уже знаете, последовательность используется во многих сферах человеческой деятельности и даже в самой природе, однако составление числового ряда занятие довольно утомительное, гораздо удобнее составить программу вычисления, которая будет автоматически строить последовательность, начиная с любых ее элементов.

При составлении программы важно не запутаться со слагаемыми: ответ от сложения первых двух чисел будет являться первым слагаемым для последующего сложения, а вторым слагаемым будет первое число первого сложения.

Практическая часть:

Исходя из того, что сложение будет выполнятся многократно, создадим на панели диаграмм цикл For Loop. Количество повторений цикла будет зависеть от требуемого количества чисел в ряду, деленого на 2. Например, если требуется отобразить 10 чисел, количество повторений N будет равно 5.

В поле цикла пропишем первые два этапа сложения. Для автоматического изменения слагаемых воспользуемся функцией Add Shift Register, которая



вызывается нажатием правой кнопкой мыши по границе цикла. На входе цикла создадим переменные, необходимые для ввода начальных слагаемых, а выходные данные соединим с функцией Index & Bundle Cluster Array, находящейся в палитре Cluster, Class, & Variant и позволяющей объединить ячейки массива с одинаковым индексом в кластер. Программа должна выглядеть так:

Рис. 45

На лицевой панели введем значения начальных чисел последовательности и запустим программу.

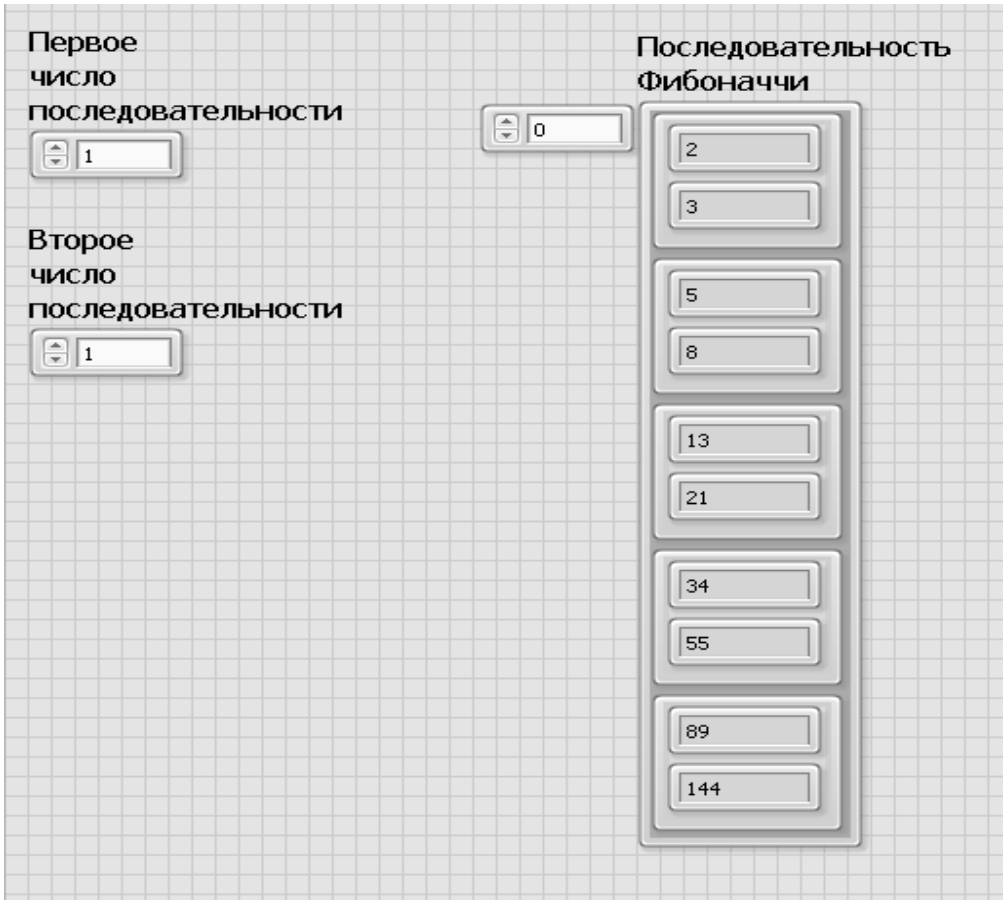


Рис. 46

Контрольные вопросы:

1. Перечислите известные вам примеры последовательности Фибоначчи, встречающиеся в природе.
2. Реализованная программа может вычислять последовательность до бесконечности? Ответ обоснуйте.
3. С помощью программы составьте пропорции золотого сечения.

Задача №9. Анализ прибыльности предприятия

Цель: Составить программу для наглядного отображения анализа эффективности предприятия, используя функции массивов.

Теоретическая часть:

Анализ прибыльности, или рентабельность позволяет оценить экономическую эффективность работы предприятия.

Обычно из доходов за расчетный период вычитают расходы, получая тем самым прибыль, которая соответственно должна быть больше расходов. В упрощенном виде это выглядит так:

$$Д - Р = П;$$

П >0 – предприятие является рентабельным;

П ≤ 0 – предприятие нерентабельно.

После анализа можно принимать меры по улучшению ситуации, которая может зависеть от разных факторов, от местоположения до сезонных особенностей.

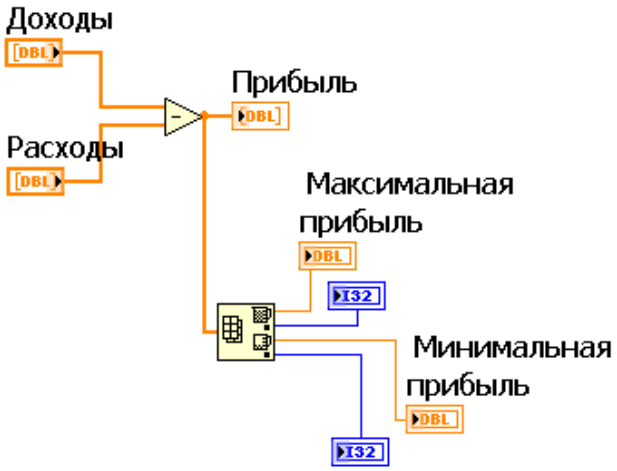
Практическая часть:

На лицевой панели необходимо создать два массива, в которые будут вписаны доходы и расходы за 6 месяцев. На панели диаграмм добавим функцию вычитание и на выходе создадим массив, в котором будет отображаться прибыль.

Воспользовавшись стандартной математической функцией можно рассчитать разность ячеек массивов с соответствующими индексами. Если требуется вычитание не всех элементов массива, или элементов с разными индексами, необходимо воспользоваться функцией Index Array.

В случае с большим количеством данных, самостоятельно определять наибольшее и наименьшее значение не удобно. Для решения этой проблемы можно воспользоваться функцией Max & Min. Она не только отображает максимальное и минимальное значение, но и выводит соответствующие им индексы, если это требуется по условию задачи.

В нашем случае, функцию можно использовать для ответа на вопрос: в каком месяце прибыль была наибольшей/наименьшей.



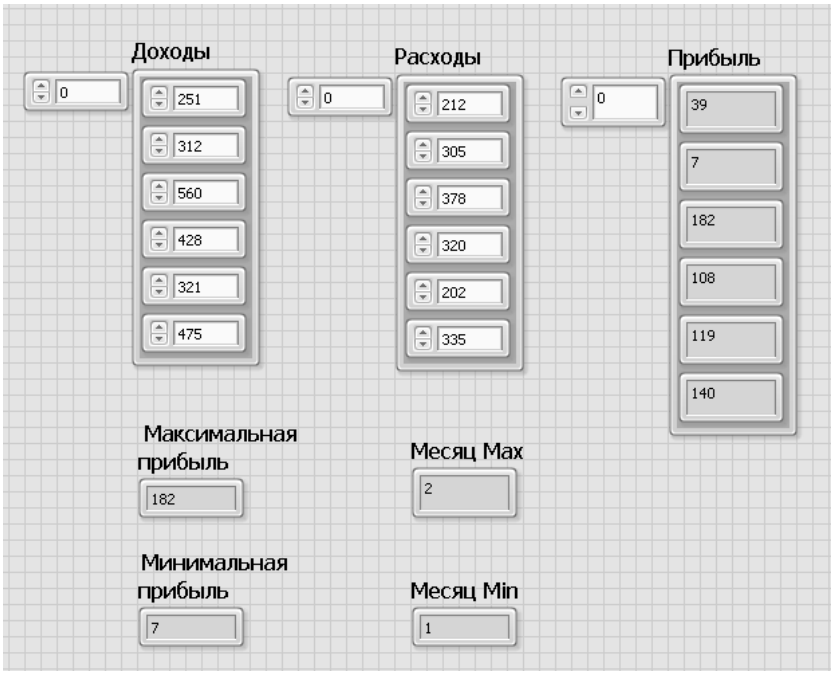


Рис. 47

Рис. 48

Контрольные вопросы:

1. Для чего нужен анализ деятельности предприятия?
2. Как еще в экономике можно использовать подобную программу? Приведите примеры.

3. Почему, месяц с максимальной прибыльностью имеет индекс имеет второй индекс, хотя находится на третьей позиции?

2.3 Педагогический эксперимент

Педагогический эксперимент проводился на базе МАОУ Гимназия №13 «Академ».

В рамках описанного курса было проведено десять практических заданий с решением межпредметных задач. Каждая задача соответствовала определенному разделу курса «Технологии». Время, отведенное на выполнение задачи, варьировалось от 45 до 90 минут, в зависимости от степени сложности и объема заданий. В работе были задействованы все шесть классов из восьмой параллели. Выборки по направлениям обучения не было, так как планируется ввести изучение среды LabVIEW в основную рабочую программу по технологии.

Перед решением задач следовал блок теоретического материала: часть теории объяснялся учителем, часть учащиеся прорабатывали самостоятельно по средствам подготовки рефератов и докладов по теме раздела. Структура теоретических сведений имела три составляющих: сначала к рассмотрению предлагался «Технологический» блок, который помогал ответить на вопрос: - «Зачем мне это нужно, где я смогу это применить?», затем Физико-Математическая составляющая, позволяющая определить принципы и закономерности работы, а в завершении блок Информатики, описывающий необходимые структуры для реализации программы в среде LabVIEW.

В начале работы, учащимся было объявлено об изменениях в структуре работы на уроке «Технология». При этом классы условно разделились на две группы. Одна часть учащихся с энтузиазмом восприняла идею об использовании среды программирования LabVIEW в решении задач. К этой группе относились учащиеся, которые хотели бы связать свою жизнь с программированием, посещали дополнительные занятия, имели хорошие оценки по информатике или же попросту не могли в полной мере реализовать себя в классических предметных заданиях. В следующую группу вошли учащиеся, знания школьного курса информатики были ниже среднего, и они

увидели в изменившихся условиях опасность снижения оценок и по технологии.

Первое практическое занятие, направленное на знакомство со средой программирования LabVIEW, помогло развеять некоторые сомнения. Учащиеся смогли на практике убедиться в простоте ее использования. Для решения задач не требовалось досконально разбираться в сложном синтаксисе, вместо этого – простые пиктограммы, значения которых интуитивно понятны, расположение, форма, цвет приборов и инструментов на рабочем поле визуально воспринимается легче, нежели сплошной текст. Таким образом, учащиеся из условно второй группы смогли почувствовать себя в равных условиях с другими учениками.

Большая роль учителя в освещении теоретических аспектов задач была в начале занятий: многие попросту не привыкли к такой структуре занятия и требовалось время на осмысление. В последствии учащиеся стали понимать требования и могли самостоятельно, или обращаясь к дополнительной литературе, анализировать все предложенные стороны решения задач, выступали с докладами. Некоторые учащиеся, стали обращаться к учителям-предметникам по математике, информатике и физике за информацией об истории, сферах применения, способах решения тех или иных задач. Учителя, в свою очередь, отметили повышение интереса к своим дисциплинам.

К концу года, несмотря на опасения некоторых учеников, успеваемость не снизилась. Часть учащихся приняли решение продолжить изучение среды программирования LabVIEW в рамках дополнительного образования.

Подводя итоги, можно сказать, что в ходе работы была достигнута цель исследования, практические задания разработаны и успешно внедрены в курс «Технологии» восьмого класса, но, что самое главное – учащиеся смогли взглянуть на изучаемые дисциплины не с разных позиций, а с точки зрения их взаимосвязи и применимости к реальным жизненным обстоятельствам.

Выводы по второй главе

Практические занятия играют одну из наиболее важных ролей в современном учебно-образовательном процессе. Процент усвоенного материала зависит не только от его качества, но и от способов проведения занятий.

В рамках предметной области «Технология» важно показать учащимся значимость теоретических сведений, полученных на других предметах, в решении практических задач стремительно меняющегося мира. В связи с этим необходимо искать новые средства для организации занятий. Практические задания должны соответствовать современному уровню развития производственной, общественной и других видов деятельности.

На основании проведенного эксперимента по внедрению заданий с использованием среды программирования LabVIEW в курс «Технологии» восьмого класса, можно говорить о том, что такая организация работы мотивирует учащихся на достижение учебных задач, способствует целостному мировоззрению, позволяет наглядно представить степень и характер влияния физических, математических, информационных и технологических параметров на различные системы.

ЗАКЛЮЧЕНИЕ

В ходе работы над магистерской диссертацией были проанализированы языки программирования. С ходом времен они изменялись, переходя от текстового способа написания программ к графическому. Одной из наиболее удобных сред графического программирования на сегодняшний день является LabVIEW. Компоненты среды позволяют создавать ничем не уступающие текстовым языком программы, а наглядный интерфейс позволяет осуществлять работу не только программистам, но и людям, не имеющим специального образования в области программирования.

Анализ рабочих программ по математике, физике, информатике и технологии восьмого класса позволил определить тематические блоки, на основе которых были составлены межпредметные задачи, визуализация решения которых возможна с помощью среды программирования LabVIEW.

Проведенный эксперимент показал, что учащиеся восьмого класса, вне зависимости от профиля и степени подготовки способны справиться с предложенными заданиями, более того, в процессе работы наблюдался повышенный интерес не только к новому виду деятельности на уроках технологии, но и к другим предметам естественно-научного цикла.

Подводя итоги, можно говорить о том, что использование на уроках технологии среды программирования LabVIEW позволяет учащимся закреплять знания, полученные на занятиях по математике, физике, информатике и технологии, а также проследить их взаимосвязь с современным производством и жизнью в целом.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Белиовская Л.Г. Узнайте, как программировать на LabVIEW. – ДМК Пресс, 2014. – 138 с.
2. Босова Л.Л., Босова А.Ю. Информатика.: учеб. для 8 кл. – М. Бином, 2014. – 160 с.
3. Виноградова Н. А., Листратов Я. И., Свиридов Е. В. Разработка прикладного программного обеспечения в среде LabVIEW. – М. МЭИ, 2015. – 50 с.
4. Вольфенгаген В. Э. Конструкции языков программирования. Приёмы описания. — М.: Центр ЮрИнфоР, 2001. — 276 с.
5. Дорофеев Г. В., Суворова С.Б., Бунимович Е.А. Алгебра 8 кл.: учеб. для общеобразоват. организаций. – М.: Просвещение, 2016. – 312 с.
6. Доуринг Э. NI myRIO: Базовое руководство по проектам. - National Technology and Science Press, 2014. – 243 с.
7. Казакевич В. М., Пичугина Г. В. Технология 8 – 9 кл.: учеб. пособие для общеобразоват. организаций. – М.: Просвещение, 2018. – 255 с.
8. Казакевич В. М., Пичугина Г. В. Технология. Методическое пособие. 5 – 9 кл.: учеб. пособие для общеобразоват. организаций. – М. : Просвещение, 2017. – 81 с.
9. Михеев П. М., Крылова С. И., Лукьянченко В. А. Учебный курс. LabVIEW. Основы. – М. МГУ им. М.В. Ломоносова, 2007. – 365 с.
10. Романенко Р. Н. Разработка приложений в среде LabVIEW. учеб. пособие. – СПб, 2014. – 117 с.
11. Перышкин А. В. Физика. 8 кл.: учеб. для общеобразоват. учреждений. – М.: Дрофа, 2013. – 237 с.
12. Пейч Д. И., Точилин Д. А., Поллак Б. П. LabVIEW для новичков и специалистов. – Горячая линия – Телеком, 2004. – 384 с.
13. Соболь А. Заметки по практическому программированию на VEE // Компоненты и технологии №3, 2009г.

14. Соучек Б. Микропроцессоры и микро-ЭВМ /; Пер. с англ. под ред. А. И. Петренко. -М.: Сов. радио 1979. - 517 с

15. Суранов А. Я. LabVIEW 8.20: Справочник по функциям. – М.: ДМК Пресс, 2007. – 536 с.

16. Тревис Дж. LabVIEW для всех / Пер. с англ. Клушин Н. А.- М.: ДМК Пресс; 2005. - 544 с.

17. Ф. Бьянкуцци, Ш. Уорден. Пионеры программирования. Диалоги с создателями наиболее популярных языков программирования. — СПб.: Символ-Плюс, 2010. — 608 с.

18. Федосов И.В. Основы программирования в LabVIEW. – Саратов, 2010. – 53 с.

19. Академик [Электронный ресурс]. – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/1465

20. Быстрая разработка [Электронный ресурс]. – Режим доступа: http://www.labview.ru/labview/what_is_labview/rapid_development.php

21. Графическое программирование [Электронный ресурс]. – Режим доступа:https://megaobuchalka.ru/10/10217.html

22. Изучение элементов и систем автоматики, а также специального программного обеспечения [Электронный ресурс]. – Режим доступа: http://mc-plc.ru/index.htm

23. История языков программирования [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/История_языков_программирования

24. Краткая история языков программирования [Электронный ресурс]. – Режим доступа: https://www.programm-school.ru/raznoe/kratkaya-istoriya-yazykov-programmirovaniya.html

25. ПиКАД – информационно-технический журнал [Электронный ресурс]. – Режим доступа: http://www.picad.com.ua/index.htm

26. Физические измерения в системе LabVIEW. Практикум. Введение в технику эксперимента [Электронный ресурс]. – Режим доступа: http://genphys.phys.msu.ru/rus/lab/vtek/VTEK-Zadacha_35.pdf

27. Черных И.В. Система моделирования динамических систем Simulink [Электронный ресурс]. – Режим доступа: http://bourabai.ru/cm/simulink.htm

28. LabVIEW Portal [Электронный ресурс]. – Режим доступа: http://labviewportal.org/