**Задание 11.** В пакете прикладных программ *National Instruments* *LabView* разработать виртуальный прибор «Умный калькулятор», в состав которого входят операции, заданные согласно таблице вариантов (Таблица 3).

Операции должны быть размещены на графическом пользовательском интерфейсе строго в том порядке, в котором они упоминаются в таблице исходных вариантов. Определение именно той операции, которую пожелал реализовать пользователь, должно выполняться в соответствии с некоторой логикой, задаваемой булевскими элементами управления, например, кнопками или тумблерами.

В процессе разработки графического пользовательского интерфейса предусмотреть:

* числовые контроллеры в необходимом количестве для ввода значений операндов,
* один единственный числовой индикатор для вывода результата/ответа,
* логические контроллеры в необходимом количестве для однозначного определения системой выполняемой операции.

Результат/ответ выводить на индикатор только при единственном включённом логическом контроллере.

Исключить одновременное включение нескольких логических контроллеров и либо не реагировать на подобный входной сигнал – оставлять на выходе ноль, либо при включении нескольких контроллеров выдавать ошибки, формируемые в виде числового кода, например, «9999», «8888», которые следует расшифровать для пользователя на графическом пользовательском интерфейсе дополнительной таблицей соответствия.

Далее приведены примеры расшифровок кодов, которые могут быть размещены на графическом пользовательском интерфейсе:

* «9999» – недопустимо одновременное умножение и деление,
* «8888» – недопустимо одновременное сложение и вычитание,
* «7777» – недопустимо деление на ноль,
* и так далее.

Автору требуется продумать структуру других возможных ошибок самостоятельно.

На блок-диаграмме все функциональные, то есть значащие пересечения линий связи (узлы) обозначить точками. Для этого в меню перейти к настройкам «*Tools > Options…*», далее в выпадающем списке перейти к настройкам блок-диаграммы (*Block Diagram*) и выставить галочку напротив пункта «*Show dots at wire junctions*».

По итогам выполнения работы сдаются строго два файла:

- отчёт, выполненный в текстовом редакторе *Microsoft Office Word* (*\*.doc* или *\*.docx*);

- файл виртуального прибора *National Instruments LabView* (*\*.vi*) по индивидуальной части работы.

Отправленные поодиночке файлы проверке не подлежат. При отсутствии одного из упомянутых файлов зачёт по заданию не выставляется.

**Требования к именам файлов:**

**Общий вид формата имени файла:** «*Дата. Задание. Фамилия.mcdx*»

**Формат записи даты:** «*ГГГГММДД*», где *ГГГГ* – четыре цифры текущего года, *ММ* – две цифры текущего месяца, *ДД* – две цифры текущего дня.

**Формат записи задания:** «Задание *NNk*», где *NN* – две цифры номера задания, *k* – обозначение «о», если файл содержит общую часть; обозначение «и», если файл содержит индивидуальную часть; обозначение «ои», если файл содержит как общую, так и индивидуальную части.

**Если устранить замечания по работе удаётся в тот же день:** после фамилии ставится пробел и в круглых скобках записывается номер попытки исправления.

**Примеры правильных имён файлов, которые сдаются на проверку впервые:**

«*20181205. Задание 11и. Иванов.docx*»

«*20181205. Задание 11и. Иванов.vi*»

**Примеры правильных имён файлов, которые сдаются на проверку повторно в тот же день:**

«*20181205. Задание 11и. Иванов (1).docx*»

«*20181205. Задание 11и. Иванов (1).vi*»

**Внимание!** Не забудьте выполнить автоматическую нумерацию страниц в отчёте.

Отчёт по выполненной работе должен содержать:

0. Титульный лист.

1. Формулировку цели работы.

2. Описание задачи согласно выданному варианту.

3. Составление блок-схемы алгоритма программы.

4. Подбор и расчёт тестовых примеров.

5. Листинг кода составленного программного обеспечения (блок-диаграммы *LabView*).

6. Графический пользовательский интерфейс программного обеспечения (передняя панель виртуального прибора *LabView*) и его описание.

7. Расчёт тестовых примеров с использованием составленного программного обеспечения.

8. Формулировку вывода о проделанной работе (обезличено – исключить из вывода местоимения, такие как «я», «мы» и другие).

Рекомендации к отчёту, доказывающие самостоятельность выполнения работы и упрощающие процедуру проверки отчёта преподавателем:

1. Выполнение дополнительных скриншотов для случаев, когда текстовое описание проделанных действий становится громоздким или трудным к восприятию.

2. Нумерация рисунков (если есть) с подписями, содержащими названия рисунков, например, «Рисунок 1 – Пользовательский интерфейс *Microsoft Office Excel*».

**Цель работы (одна из возможных формулировок)**: закрепление навыков работы и навыков графического программирования в пакете прикладных программ *National Instruments* *LabView*. Приобретение навыков работы c *CASE*-структурой, входящей в состав пакета прикладных программ *National Instruments* *LabView*.

Необходимо продумать индивидуальность подхода к решению логического наполнения «Умного калькулятора» на базе *CASE-*структуры. Двух одинаковых работ в студенческой группе быть не должно.

----------------------------------------------------------------------------------------------------

**Для выполнения индивидуального задания потребуются следующие функции:**

Раздел числовых элементов содержит следующие графические функции, подразделы и константы, изображённые на Рисунке 1 (чтение ведётся слева направо, сверху вниз):

- операция сложения (*Add*);

- операция вычитания (*Subtract*);

- операция умножения (*Multiply*);

- операция деления (*Divide*);

- операции целочисленного деления и получения остатка от деления (*Quotient & Remainder*);

- подраздел конвертации (*Conversion*);

- операция инкрементирования / добавления единицы (*Increment*);

- операция декрементирования / вычитания единицы (*Decrement*);

- операция суммирования элементов массива (*Add Array Elements*);

- операция перемножения элементов массива (*Multiply Array Elements*);

- элемент, объединяющий арифметические операции (*Compound Arithmetic*);

- подраздел тригонометрии (*Trigonometric*);

- операция взятия абсолютной величины / модуля (*Absolute Value*);

- операция математического округления до ближайшего целого (*Round To Nearest*);

- операция округления в меньшую сторону (*Round To -Infinity*);

- операция округления в большую сторону (*Round To +Infinity*);

- операция выдачи реализации псевдослучайной величины (*Random Number (0-1)*);

- подраздел логарифмов (*Logarithmic*);

- операция взятия квадратного корня (*Square Root*);

- операция инверсии / взятия значения с обратным знаком (*Negate*);

- операция масштабирования по степени двойки / умножение на 2 в степени *N* (*Scale By Power Of 2*);

- операция сигнум (определения знака): пришло любое положительное – в ответе «1», пришло любое отрицательное – в ответе «-1», пришёл ноль – в ответе «0» (*Sign*);

- операция обращения / деления единицы на заданное значение *(Reciprocal*);

- подраздел операций над комплексными числами (*Complex*);

- численная константа (*Numeric Constant*);

- перечислимая константа (*Enum Constant*);

- константа списка значений (*Ring Constant*);

- узел математических выражений (*Expression Node*);

- подраздел дополнительных констант (*Additional Numeric Constants*).

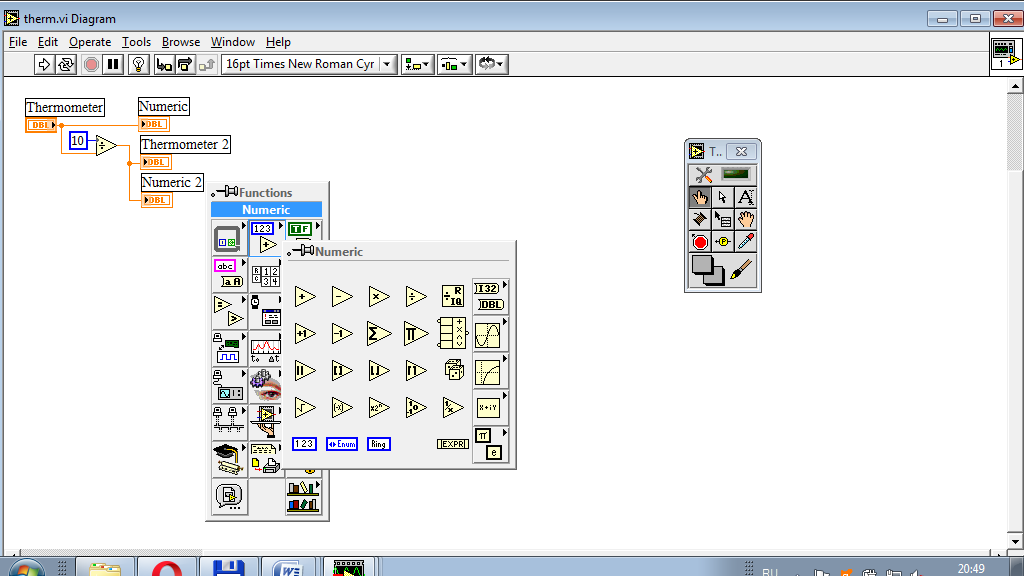


Рисунок 1 – Раздел численных (*Numeric*) функций на блок-диаграмме

Подчёркнутые выше элементы в данной работе использованы не будут. Далее рассмотрен подраздел тригонометрических функций (Рисунок 2):

- синус (*Sine*);

- косинус (*Cosine*);

- тангенс (*Tangent*);

- арксинус (*Inverse Sine*);

- арккосинус (*Inverse Cosine*);

- арктангенс (*Inverse Tangent*);

- гиперболический синус (*Hyperbolic Sine*);

- гиперболический косинус (*Hyperbolic Cosine*);

- гиперболический тангенс (*Hyperbolic Tangent*);

- гиперболический арксинус (*Inverse Hyperbolic Sine*);

- гиперболический арккосинус (*Inverse Hyperbolic Cosine*);

- гиперболический арктангенс (*Inverse Hyperbolic Tangent*);

- косеканс (*Cosecant*);

- секанс (*Secant*);

- котангенс (*Cotangent*);

- объединённых синуса и косинуса (*Sine & Cosine*);

- арктангенс двух аргументов (*Inverse Tangent (2 Input)*);

- отсчётов (*Sinc*).

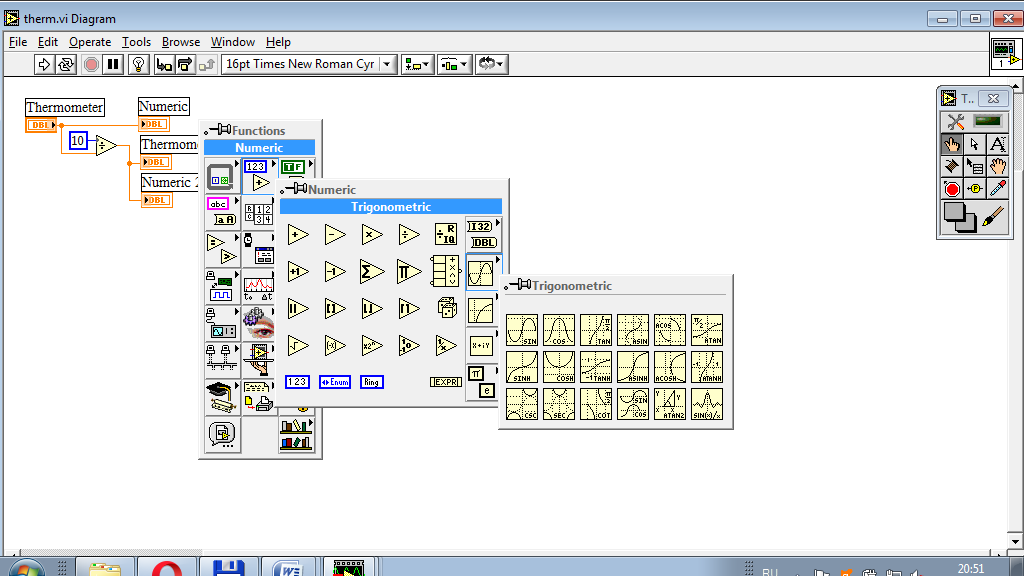


Рисунок 2 – Подраздел тригонометрических (*Trigonometric*) функций раздела численных (*Numeric*) функций на блок-диаграмме

Подчёркнутые выше элементы в данной работе использованы не будут. Далее рассмотрен подраздел логарифмических функций (Рисунок 3):

- экспонента (*Exponential*);

- степень десяти (*Power Of 10*);

- степень двойки (*Power Of 2*);

- возведение *X* в степень *Y* (*Power Of X*);

- экспонента, проходящая через начало координат (*Exponential (Arg) – 1*);

- натуральный логарифм (*Natural Logarithm*);

- логарифм по основанию десяти (*Logarithm Base 10*);

- логарифм по основанию двойки (*Logarithm Base 2*);

- логарифм по основанию *X* (*Logarithm Base X*);

- натуральный логарифм, проходящий через начало координат (*Natural Logarithm (Arg + 1)*).

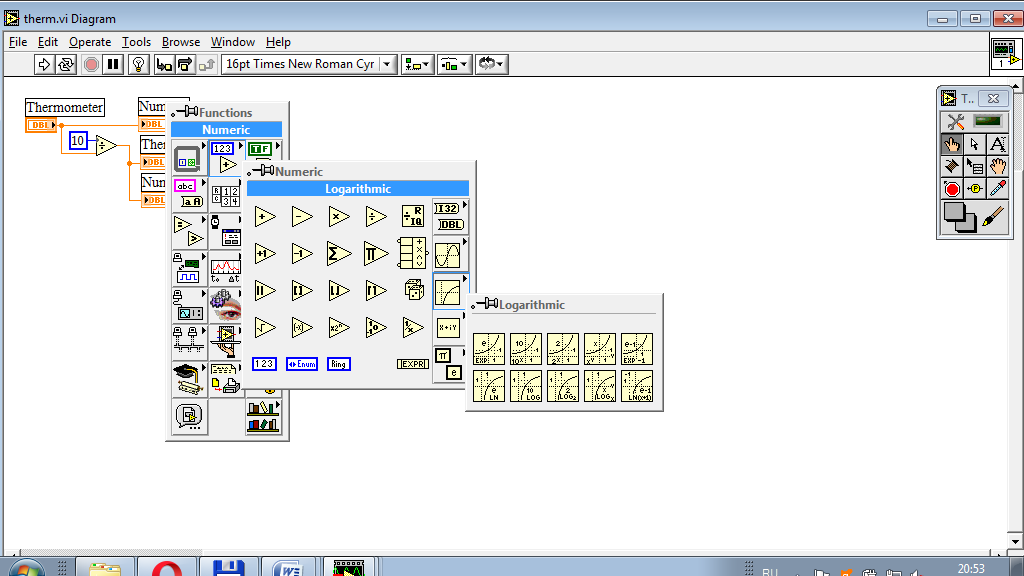


Рисунок 3 – Подраздел логарифмических (*Logarithmic*) функций раздела численных (*Numeric*) функций на блок-диаграмме

Подчёркнутые выше элементы в данной работе использованы не будут. Далее рассмотрен подраздел дополнительных численных констант (Рисунок 4):

- цветовая константа (*Color Box Constant*);

- символьная константа списка значений для интерфейсного элемента «список» (*Listbox Symbol Ring Constant*);

- константа списка ошибок (*Error Ring Constant*);

- Пифагорова константа (*Pi*);

- Пифагорова константа, умноженная на два (*Pi Multiplied By 2*);

- Пифагорова константа, делённая на два (*Pi Divided By 2*);

- обращённая Пифагорова константа (*Reciprocal Of Pi*);

- натуральный логарифм Пифагоровой константы (*Natural Logarithm Of Pi*);

- минус бесконечность (*Negative Infinity*);

- основание натурального логарифма (*Natural Logarithm Base*);

- обращённое основание натурального логарифма (*Reciprocal Of e*);

- логарифм экспоненты по основанию десяти (*Base 10 Logarithm Of e*);

- натуральный логарифм десяти (*Natural Logarithm Of 10*);

- натуральный логарифм двойки (*Natural Logarithm Of 2*);

- плюс бесконечность (*Positive Infinity*);

- постоянная Планка (*Planck Constant (J/Hz)*);

- элементарный заряд (*Elementary Charge (C)*);

- скорость света в вакууме (*Speed Of Light In Vacuum (m/sec)*);

- гравитационная постоянная (*Gravitational Constant (N m2/kg2)*);

- постоянная Авогадро (*Avogadro Constant (1/mol)*);

- Постоянная Ридберга (*Rydberg Constant (/m)*);

- молярная газовая постоянная (*Molar Gas Constant (J / (mol K))*).

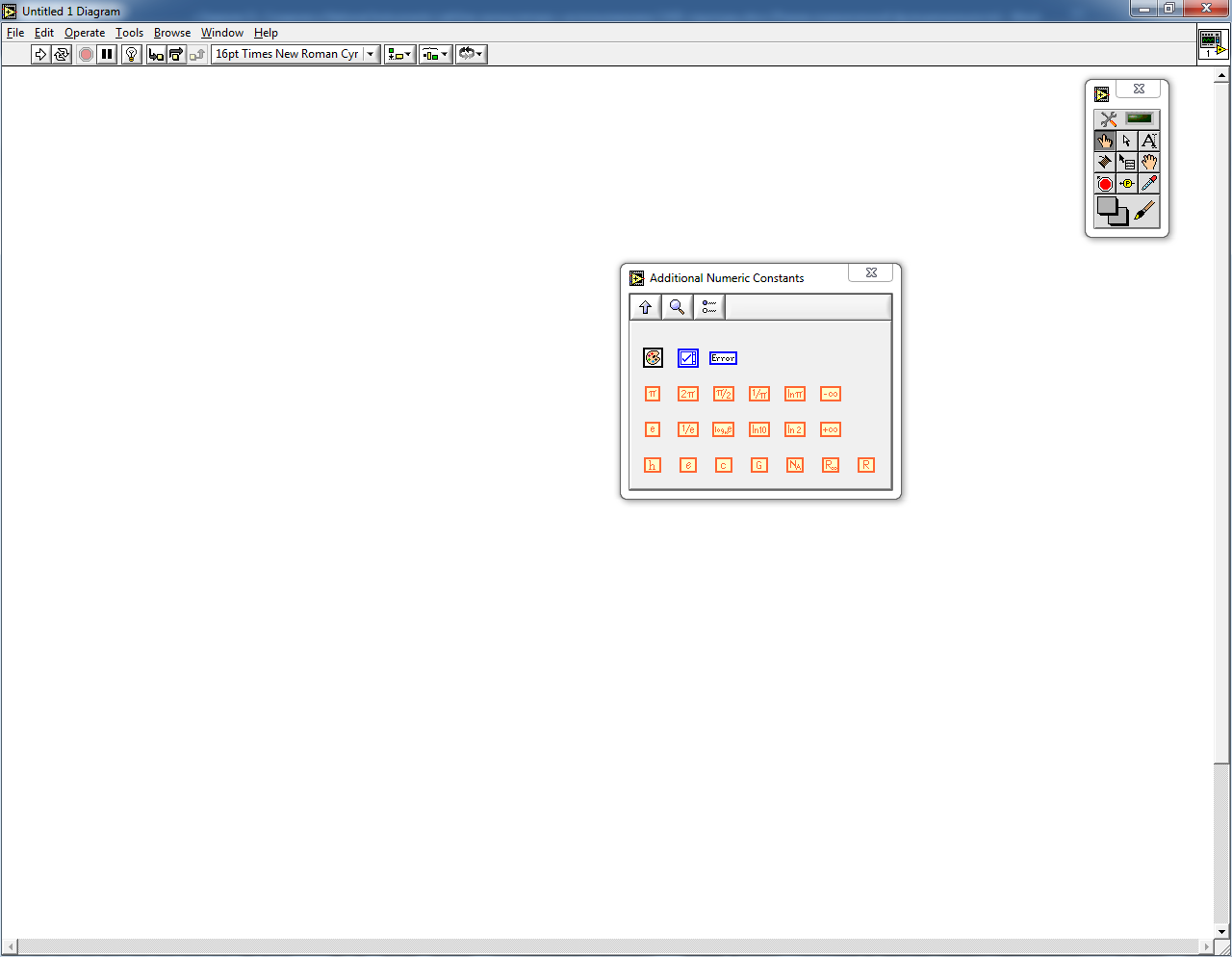


Рисунок 4 – Подраздел дополнительных численных констант (*Additional Numeric Constants*) раздела численных (*Numeric*) функций на блок-диаграмме

Подчёркнутые выше элементы в данной работе использованы не будут.

----------------------------------------------------------------------------------------------------

**Использование объединённой арифметики:**

Объединённая арифметика – функция, относящаяся одновременно к двум разделам: числовому и логическому. Функция объединяет в себе следующие операции: сложения, умножения, логического сложения (ИЛИ), логического умножения (И), исключающего ИЛИ. На Рисунке 5 показано расположение объединённой арифметики в разделе численных функций блок-диаграммы.

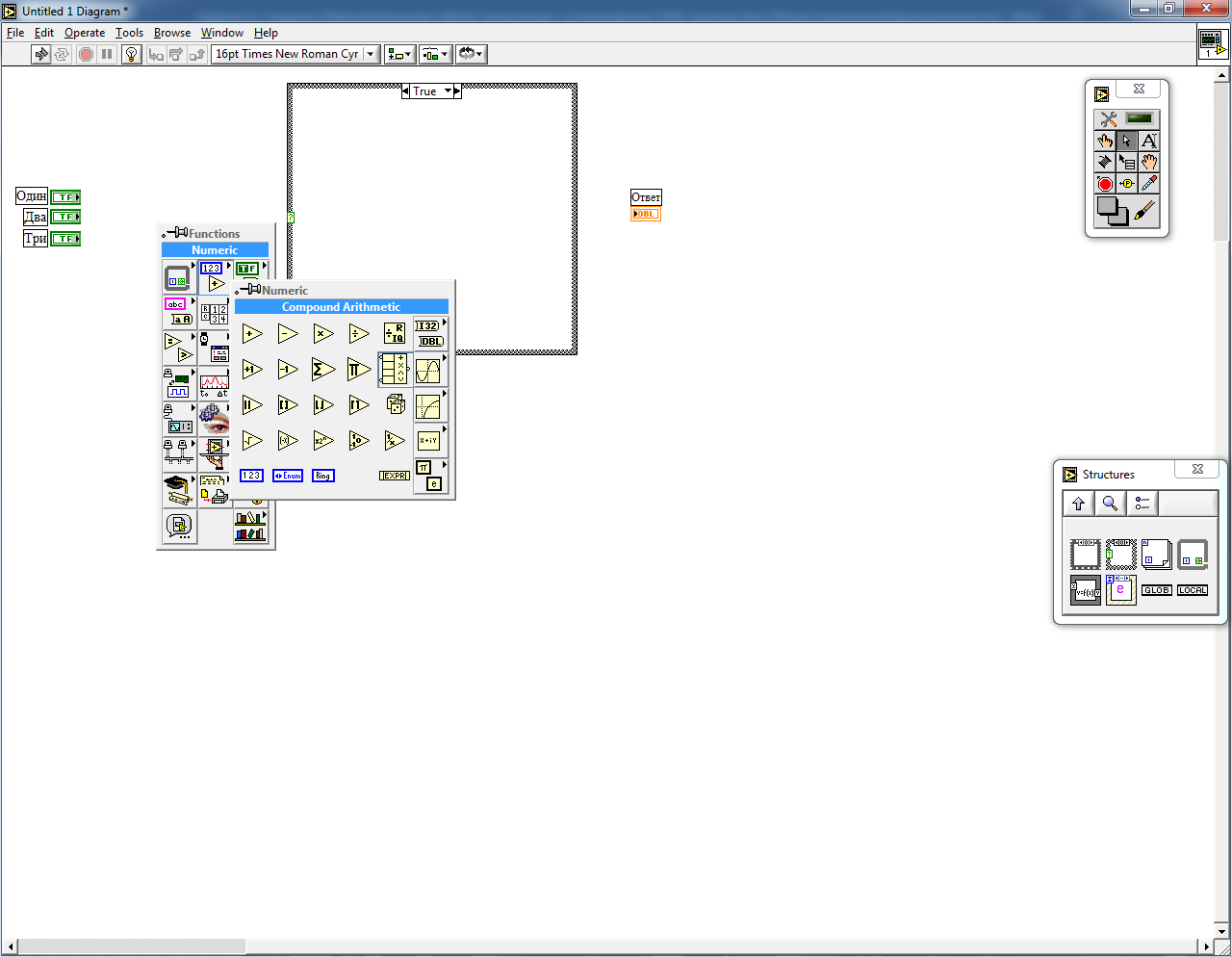


Рисунок 5 – Маркировка расположения функции объединённой арифметики (*Compound Arithmetic*) раздела численных (*Numeric*) функций, размещаемых на блок-диаграмме

При размещении объединённой арифметики на блок-диаграмме по умолчанию доступно только два входа, а также выбрана операция сложения. Для изменения количества входов необходимо выбрать среди инструментов элемент позиционирования / изменения размерности / выбора (*Position / Size / Select*) и, ухватившись за нижнюю границу размещённой на блок-диаграмме функции, вытянуть её вниз на нужное количество входов. Для изменения выполняемой операции функцией объединённой арифметики необходимо вызвать контекстное меню этой функции нажатием на неё правой кнопкой мыши и далее в пункте изменения режима (*Change Mode*) выбрать нужную операцию. В рамках решаемой задачи объединённая арифметика используется в режиме логического умножения (*AND*). Смена режима объединённой арифметики показана на Рисунке 6.

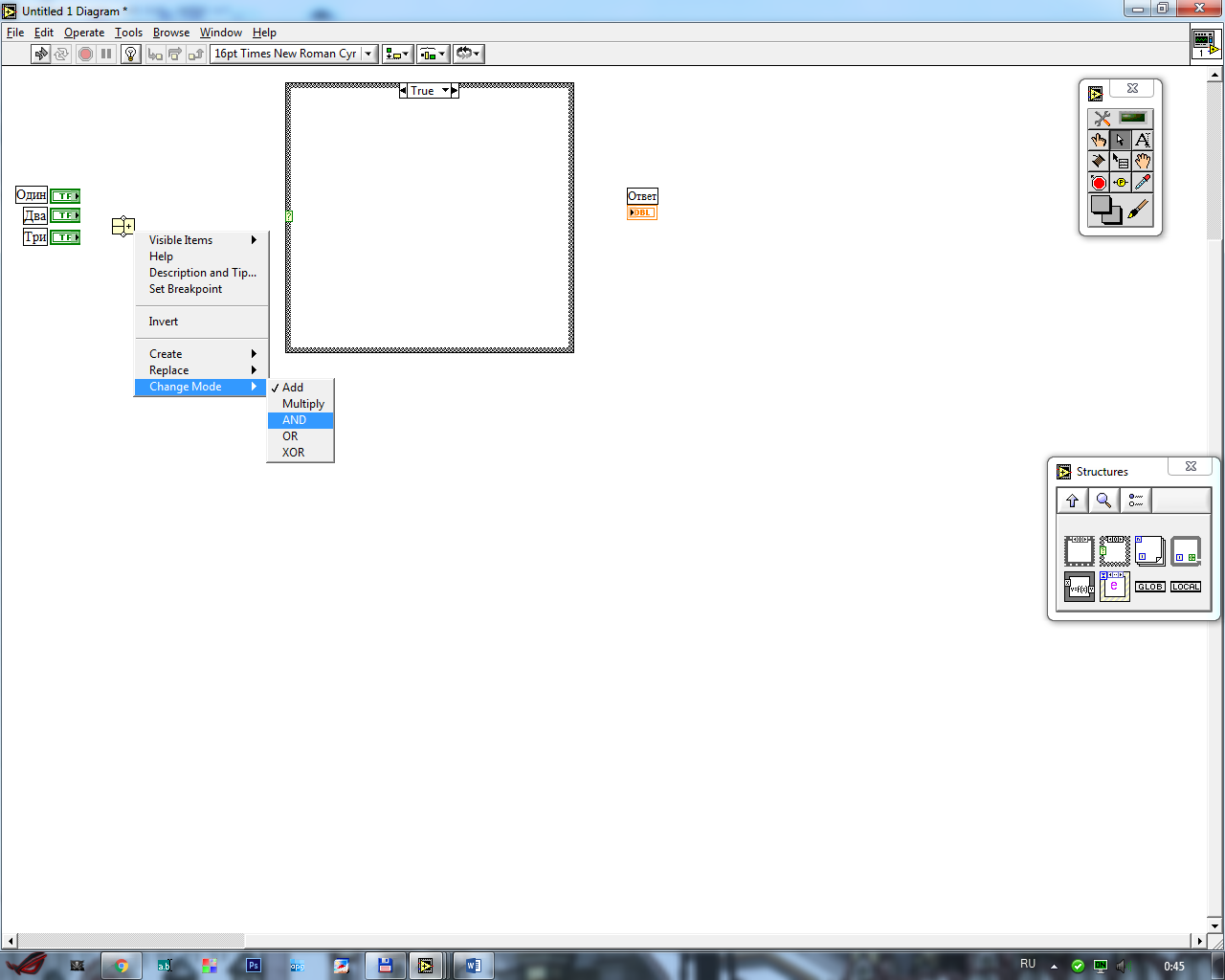


Рисунок 6 – Замена операции в рамках функции объединённой арифметики с численного сложения «+» (*Add*) на логическое умножение «И» («AND»)

С целью сокращения количества применяемых логических элементов на блок-диаграмме можно воспользоваться одной из полезных настроек функции объединённой арифметики. В контекстном меню функции объединённой арифметики присутствует возможность инвертирования (*Invert*) полученного значения. Так, например, в решаемой задаче может быть эффективно использована операция отрицания логического умножения («И-НЕ»). Такая операция доступна посредством размещения на блок-диаграмме одного единственного, но особым образом настроенного блока функции объединённой арифметики. Настройка инверсии логического умножения представлена на Рисунке 7.

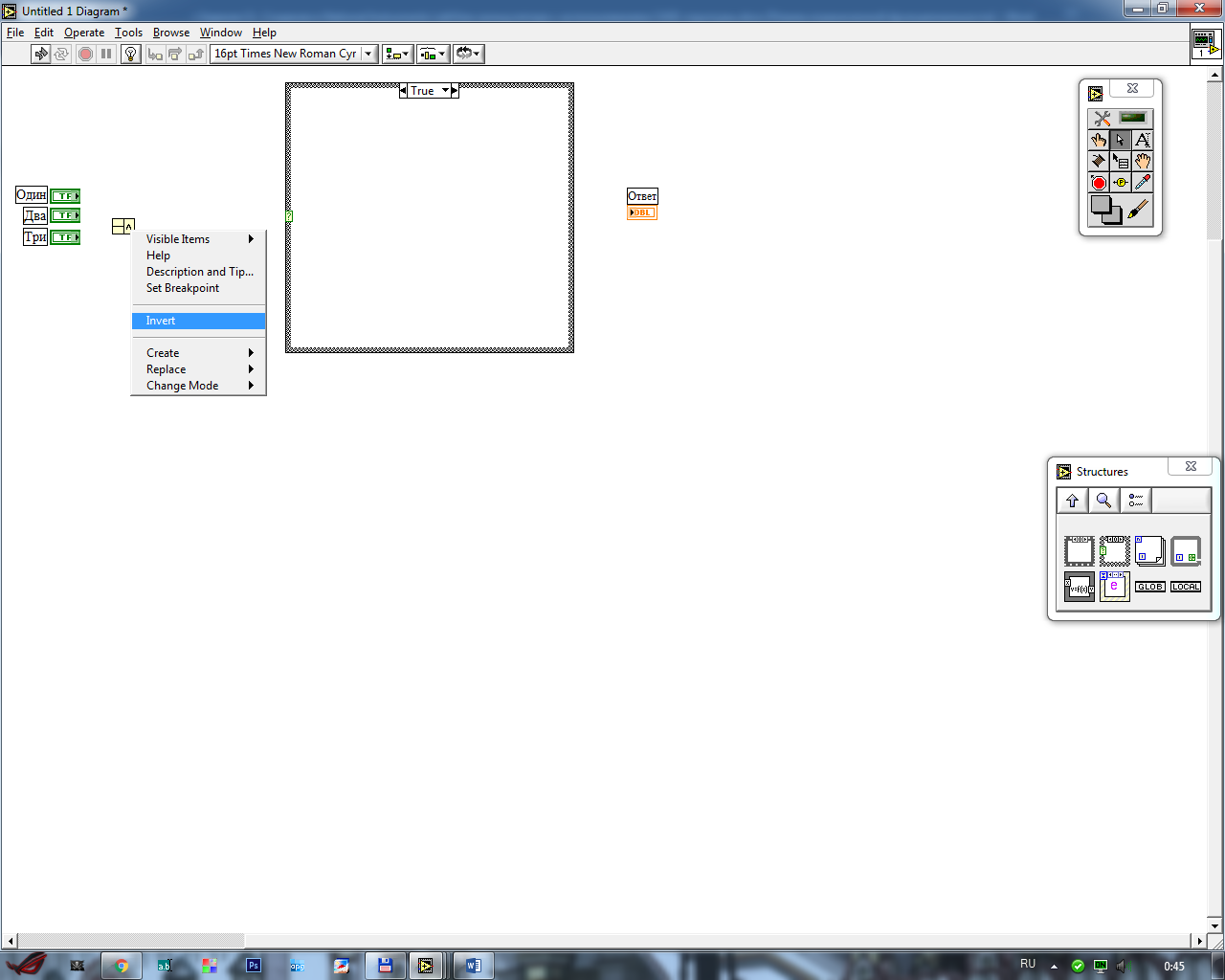


Рисунок 7 – Изменение режима применяемой операции логического умножения с прямой «И» на инверсную «И-НЕ» (*Invert AND*)

По итогам изучения основ настройки функции объединённой арифметики, а также численных и логических элементов, входящих в состав пакета прикладных программ *National Instruments LabView*, можно переходить непосредственно к созданию заготовки для виртуального прибора «Умный калькулятор».

----------------------------------------------------------------------------------------------------

**Заготовка графического пользовательского интерфейса и графической кодовой конструкции для виртуального прибора «Умный калькулятор», функционирующего на базе *CASE*-структуры**

В качестве первого приближения к созданию виртуального прибора «Умный калькулятор» рассмотрим один из его возможных графических пользовательских интерфейсов. Для понимания принципов наращивания (масштабирования) структуры виртуального прибора «Умный калькулятор» достаточно рассмотреть три операции. Каждой операции ставится в соответствие логический контроллер – кнопка. Так на передней панели должны быть размещены три кнопки, которые назовём слева направо «Один», «Два» и «Три», соответственно. В таком случае подразумевается вывод на численный индикатор «Ответ» по нажатию своей, заранее определённой целочисленной константы: либо «1», либо «2», либо «3».

Стоит отметить, что для большинства вариантов данной работы требуется использование операндов для выполнения над ними определённых операций. Под каждый операнд выделяется свой собственный численный контроллер. Заготовка интерфейса, представленная на Рисунке 8, содержит все вышеуказанные элементы, однако, в рассматриваемом примере «Операнд 1» и «Операнд 2» использованы не будут.

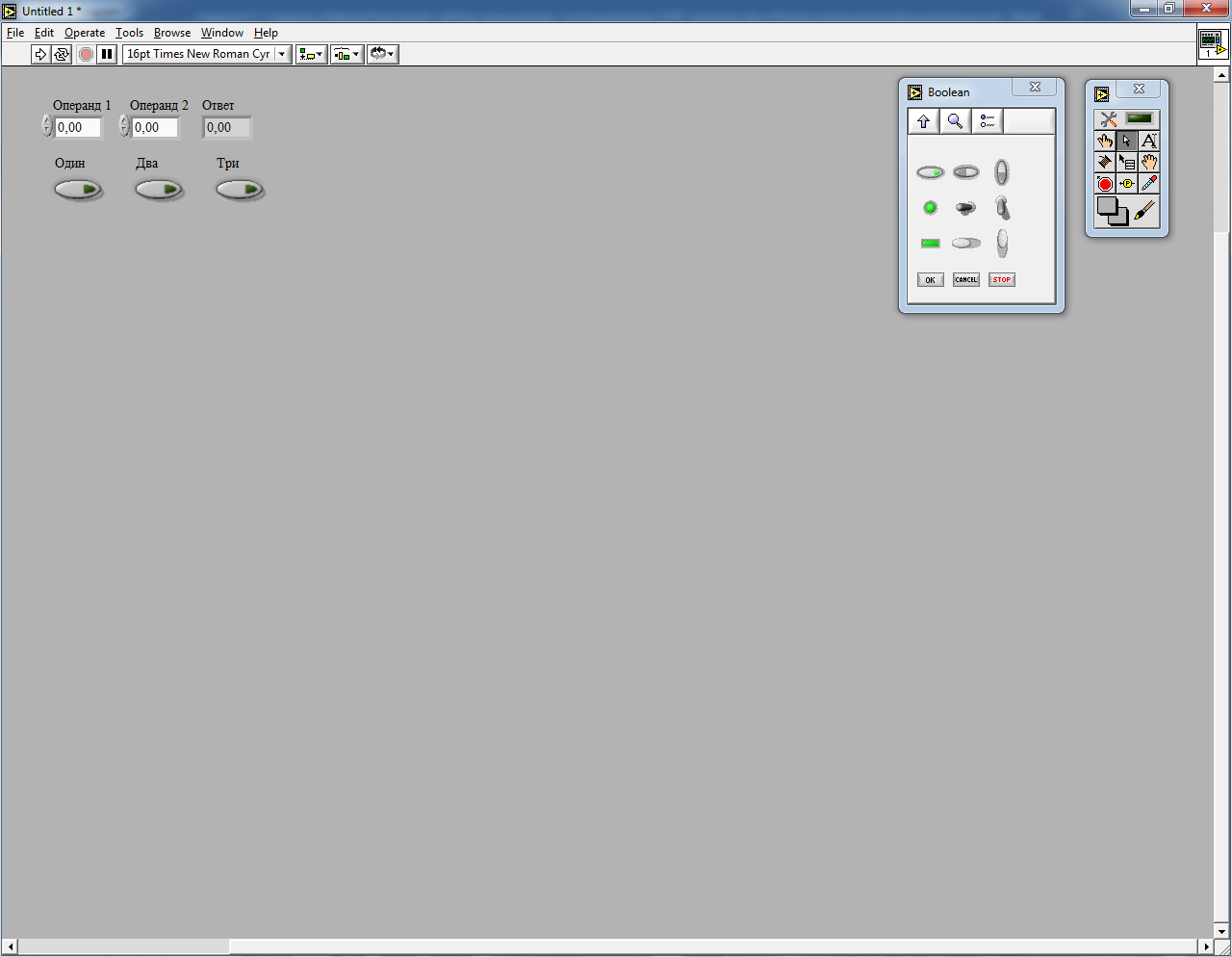


Рисунок 8 – Пример графического пользовательского интерфейса виртуального прибора «Умный калькулятор» с двумя входными операндами, тремя операциями и одним индикатором с численным результатом

При таком наборе интерфейсных элементов блок-диаграмма будет содержать в себе те же элементы, что представлены на Рисунке 9. Помимо них к заготовке добавлена *CASE*-структура. Заметим, что для большей компактности размещения элементов на блок-диаграмме при использовании инструмента позиционирования / изменения размера / выбора (*Position / Size / Select*) наименования этих элементов размещены слева от них.

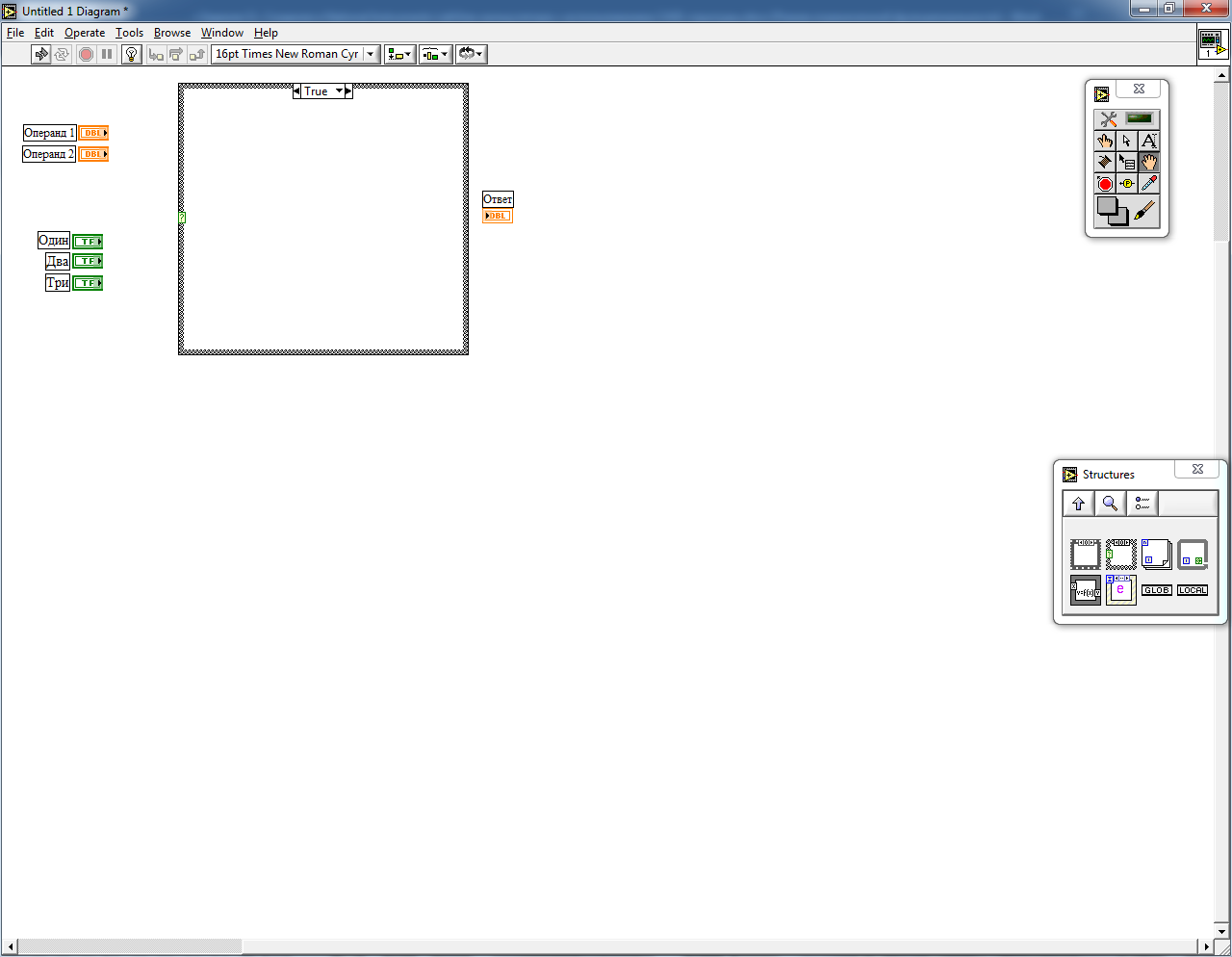


Рисунок 9 – Заготовка блок-диаграммы для последующей настройки рассматриваемого примера

Итого в заготовке использованы элементы из следующих функциональных разделов:

- численные функции;

- логические функции;

- структуры.

Рассмотрение численных и логических уже затрагивалось в более ранних работах Учебной практики, связанных с *National Instruments LabView*. Структуры ранее не использовались. Рассмотрим их расположение в перечне функций, размещаемых на блок-диаграмме (Рисунок 10).

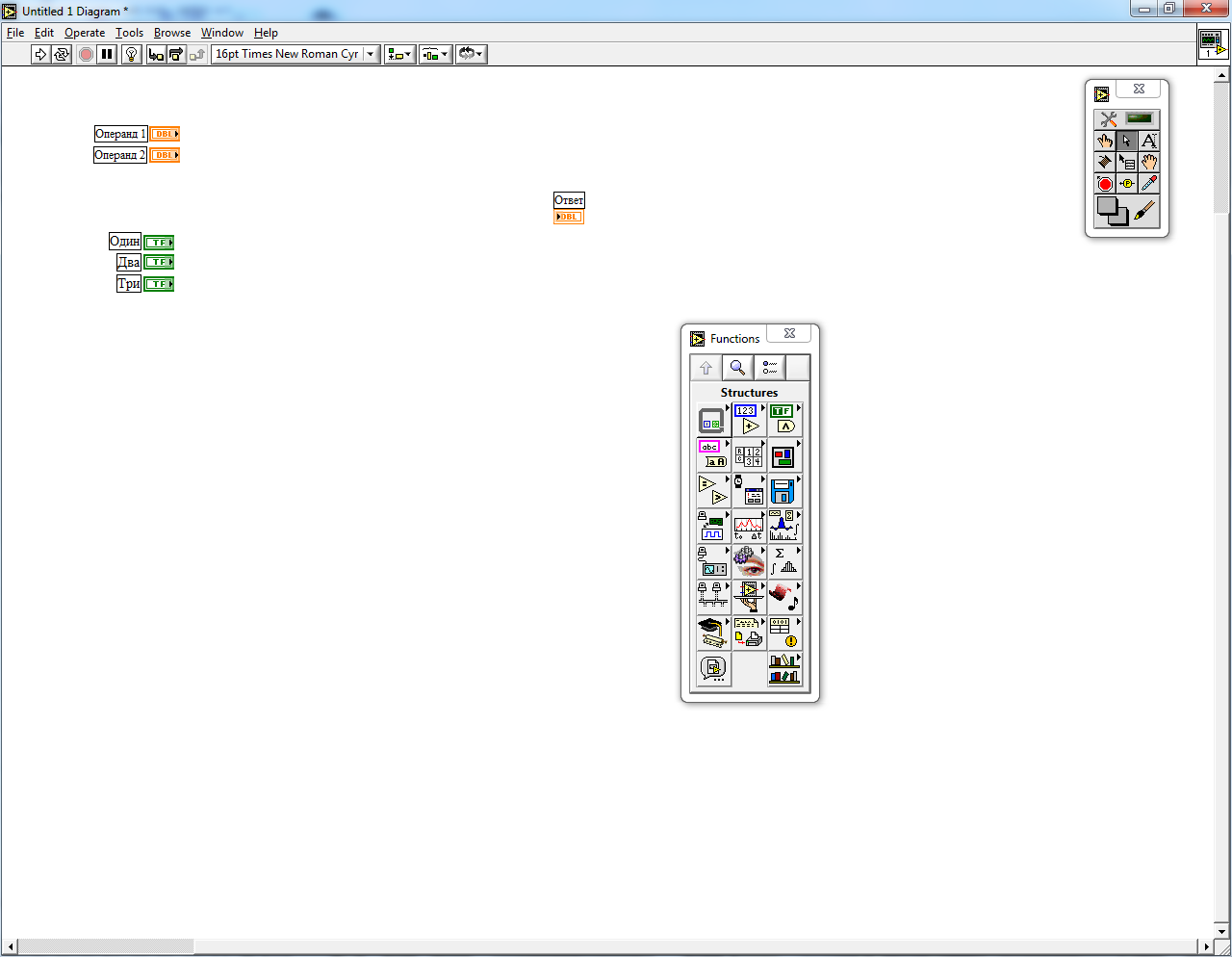


Рисунок 10 – Маркировка раздела структур в перечне функций, размещаемых на блок-диаграмме

Вход в раздел структур сопровождается выводом на экран окна, содержащего все существующие в *National Instruments* *LabView* структуры (их не много), которые при чтении слева направо следующие (Рисунок 11):

- структура последовательности (*Sequence Structure*);

- *CASE*-структура (*Case Structure*);

- структура цикла *For* (*For Loop*);

- структура цикла *While* (*While Loop*);

- узел-формула (*Formula Node*);

- структура событий (*Event Structure*);

- глобальная переменная (*Global Variable*);

- локальная переменная (*Local Variable*).

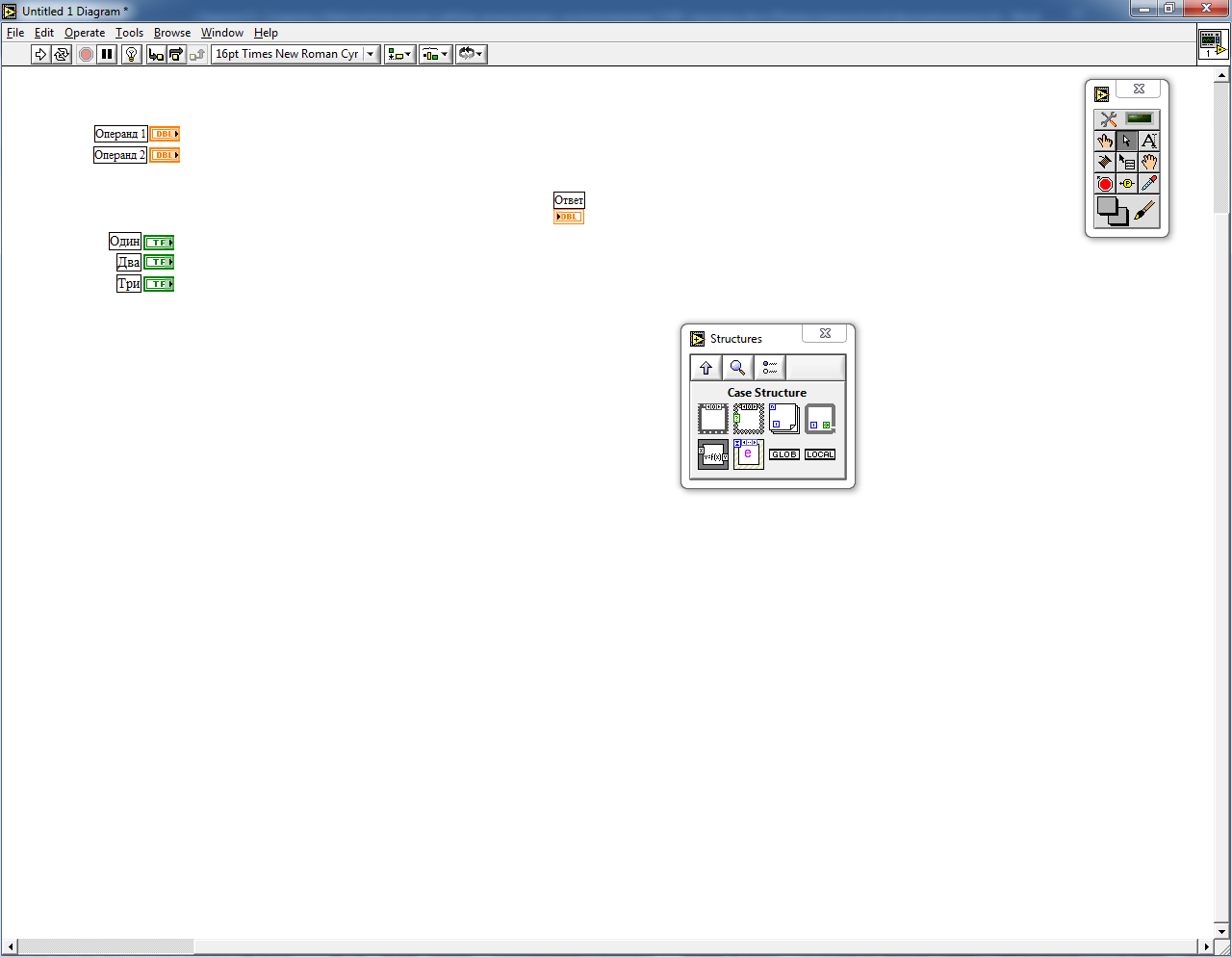


Рисунок 11 – Содержимое раздела структур блок-диаграммы с маркировкой *CASE*-структуры (*Case Structure*)

*CASE*-структура может быть использована в логическом режиме (к знаку вопроса слева подключается связь логического типа), в таком случае её работа аналогична работе условного оператора. Также *CASE*-структура может быть использована в численном режиме (к знаку вопроса слева подключается связь целочисленного типа), и в таком случае структура работает как оператор переключения (*switch* на языке *Visual C#*).

----------------------------------------------------------------------------------------------------

**Упрощённый пример настройки «Умного калькулятора». Использование *CASE*-структуры в логическом режиме:**

Как и было заявлено ранее, для демонстрации работы *CASE*-структуры откажемся от операндов «Операнд 1» и «Операнд 2», чтобы сделать акцент на логике работы виртуального прибора (Рисунок 12).

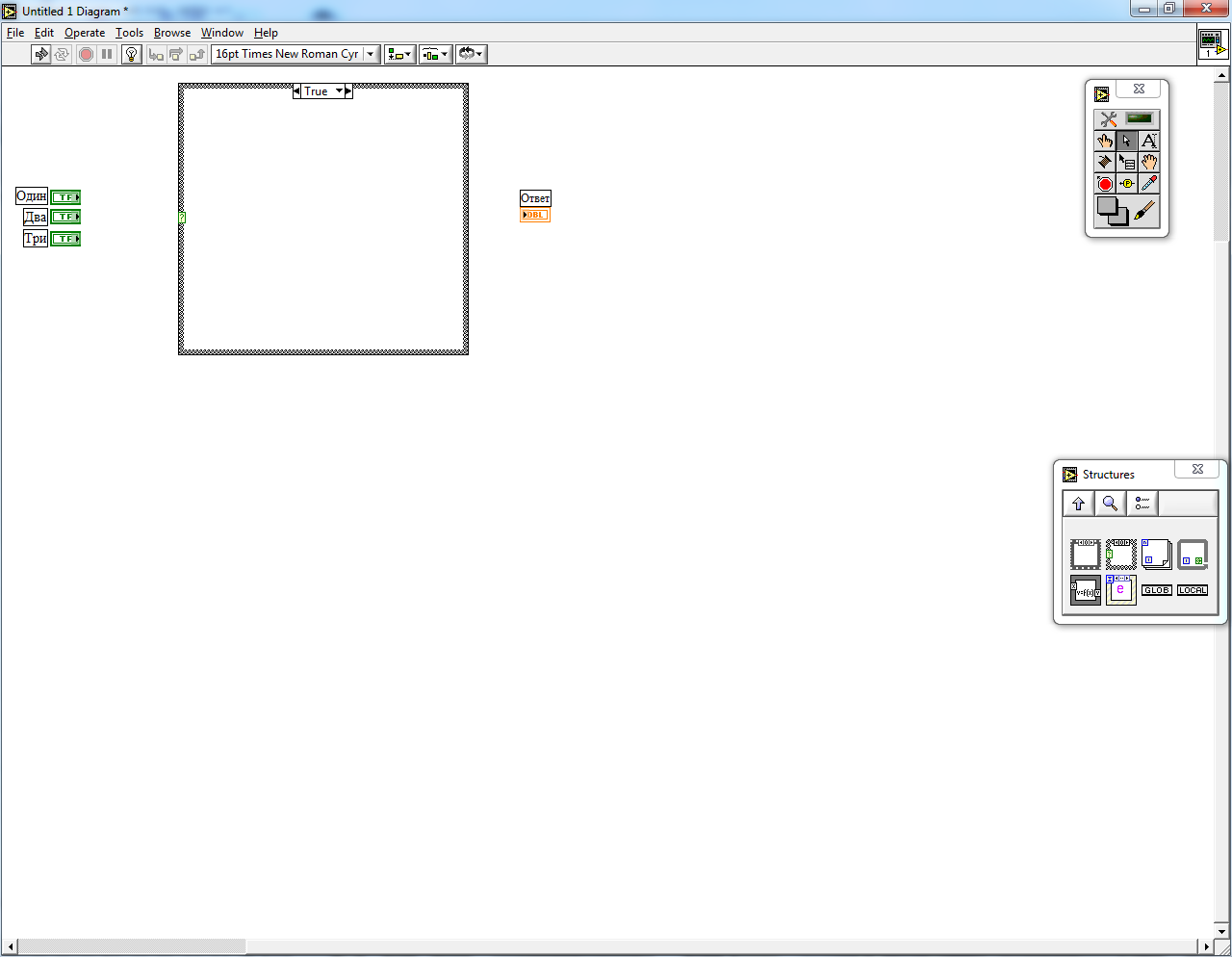


Рисунок 12 – Пример заготовки на блок-диаграмме фрагмента «Умного калькулятора», не требующего участия каких-либо операндов

Для вывода на числовой индикатор только одного из адекватных ответов лишь одна кнопка из трёх должна быть нажата, а другие не нажаты. То есть:

- либо кнопка «Один» нажата, кнопки «Два» и «Три» не нажаты,

- либо кнопка «Два» нажата, кнопки «Один» и «Три» не нажаты,

- либо кнопка «Три» нажата, кнопки «Один» и «Два» не нажаты.

Построим схему на блок-диаграмме в соответствии со сформулированной выше логикой. Начнём с первого условия: «Два» и «Три» логически перемножаются и инвертируются функцией объединённой арифметики и полученный результат логически умножается на «Один» (Рисунок 13).

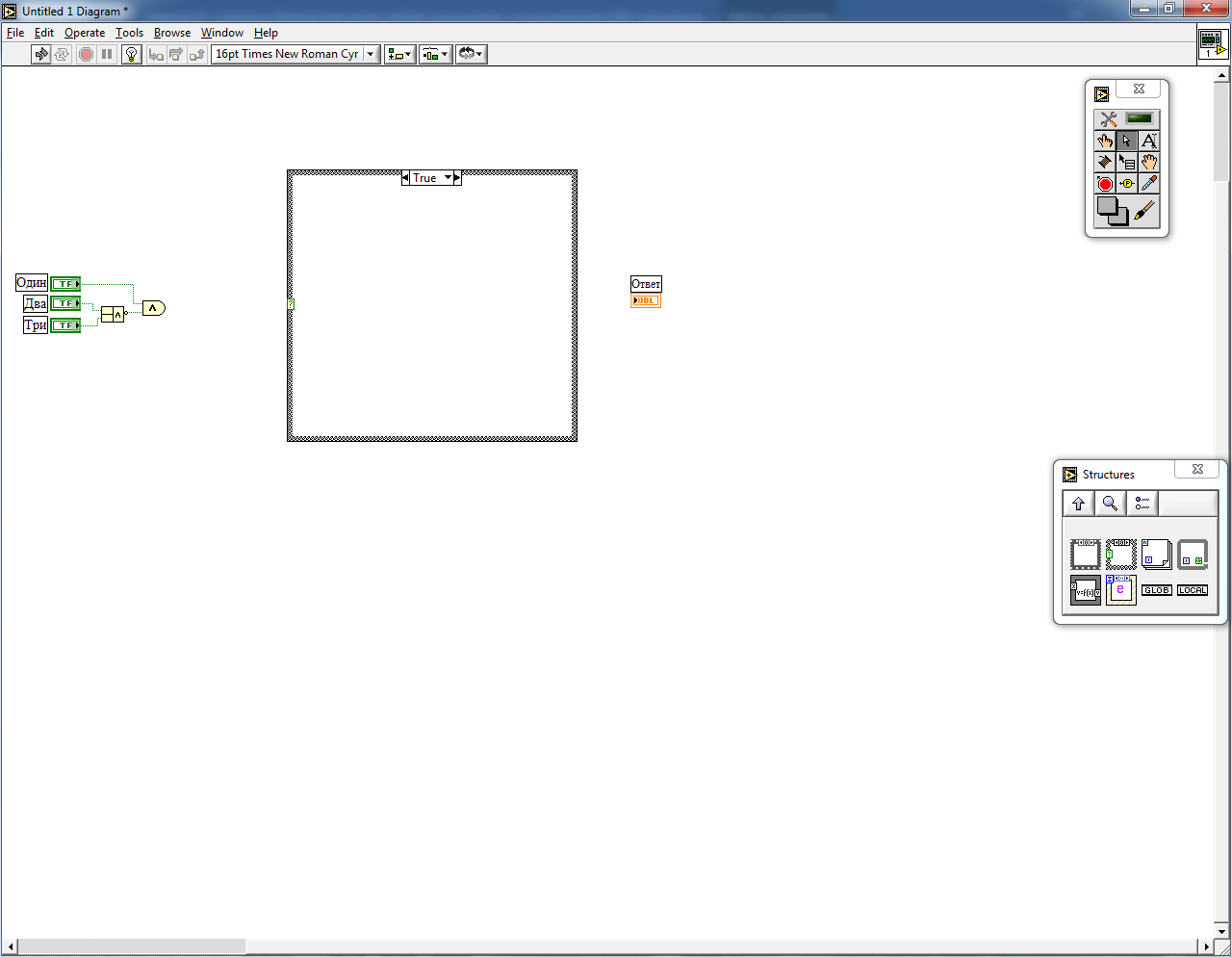


Рисунок 13 – Составление логики для нажатой кнопки «Один» и не нажатых кнопок «Два» и «Три»

По аналогии строится схема для второго условия: «Один» и «Три» логически перемножаются и инвертируются функцией объединённой арифметики и полученный результат логически умножается на «Два» (Рисунок 14).

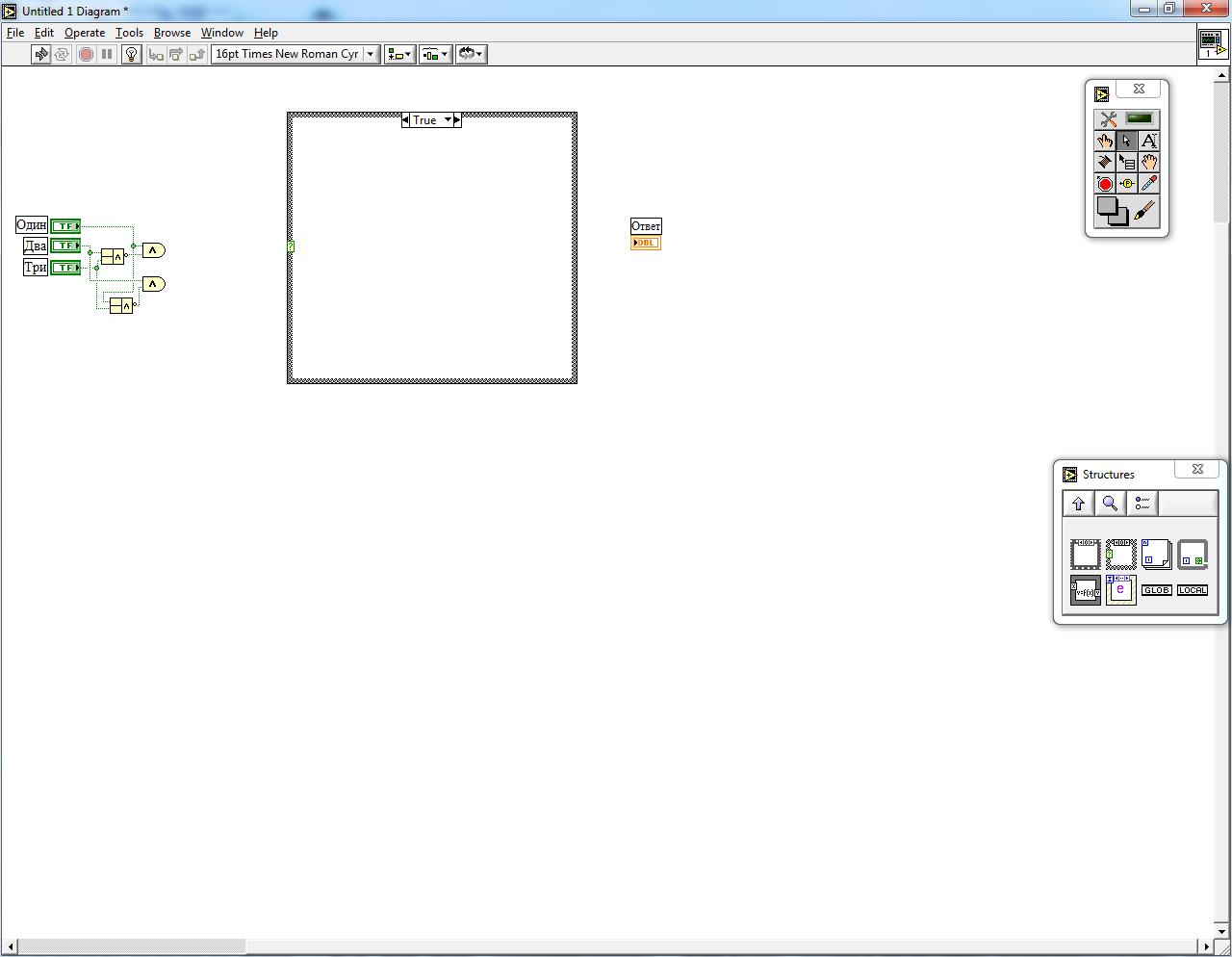


Рисунок 14 – Добавление логики для нажатой кнопки «Два» и не нажатых кнопок «Один» и «Три»

Схема дополняется учётом третьего условия (Рисунок 15).

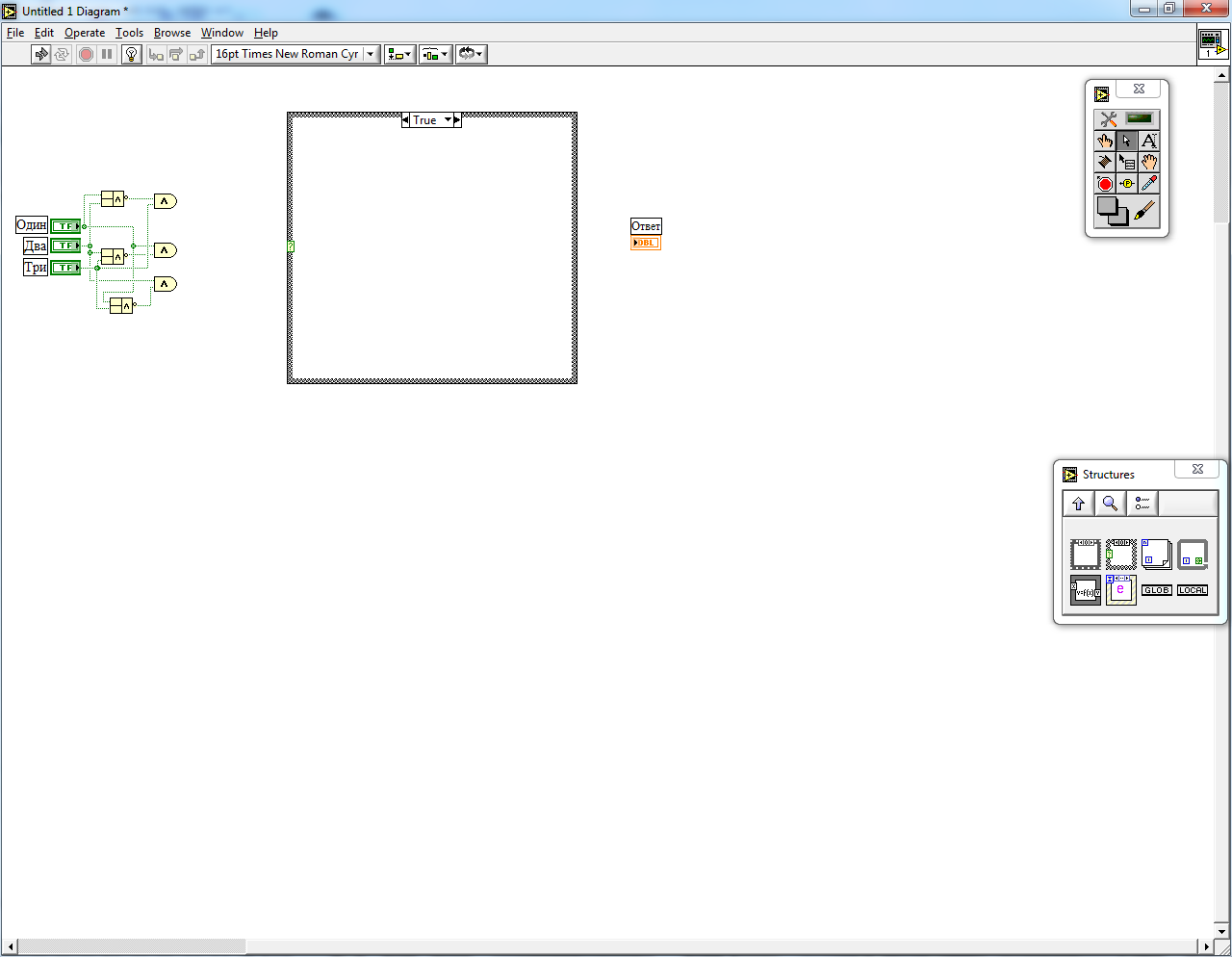


Рисунок 15 – Добавление логики для нажатой кнопки «Три» и не нажатых кнопок «Один» и «Два»

Решение задачи контроля нажатия одной единственной кнопки из трёх при составлении данной логической цепочки – это тот редкий случай, когда уместно применение операции исключающего ИЛИ. Это трудно принять с первого прочтения, поскольку само по себе в рассуждениях напрашивается обыкновенное логическое сложение, но это не так. Для доказательства составим таблицу истинности по логическому сложению (Таблица 1) и таблицу истинности для исключающего ИЛИ (Таблица 2).

Таблица 1 – Истинность объединения трёх операндов по ИЛИ

|  |  |  |  |
| --- | --- | --- | --- |
| **Один** | **Два** | **Три** | **OR** |
| false | false | false | **false** |
| false | false | true | **true** |
| false | true | false | **true** |
| false | true | true | **true** |
| true | false | false | **true** |
| true | false | true | **true** |
| true | true | false | **true** |
| true | true | true | **true** |

Таблица 2 – Истинность объединения трёх операндов по исключающему ИЛИ

|  |  |  |  |
| --- | --- | --- | --- |
| **Один** | **Два** | **Три** | **XOR** |
| false | false | false | **false** |
| false | false | true | **true** |
| false | true | false | **true** |
| false | true | true | **false** |
| true | false | false | **true** |
| true | false | true | **false** |
| true | true | false | **false** |
| true | true | true | **false** |

Хорошо видно, что объединение по ИЛИ – это истинность хотя бы одного из трёх. То есть если хотя бы один элемент отвечает истиной, то всё выражение отвечает истиной – логика не подходит. Исключающее ИЛИ гласит: только один должен ответить истиной, тогда всё выражение ответит истиной.

Объединение ранее сформулированных условий по исключающему ИЛИ объединённой арифметикой представлено на Рисунке 16. Вместе с тем для тестирования виртуального прибора в блок истинного результата всего выражения устанавливается плюс бесконечность.

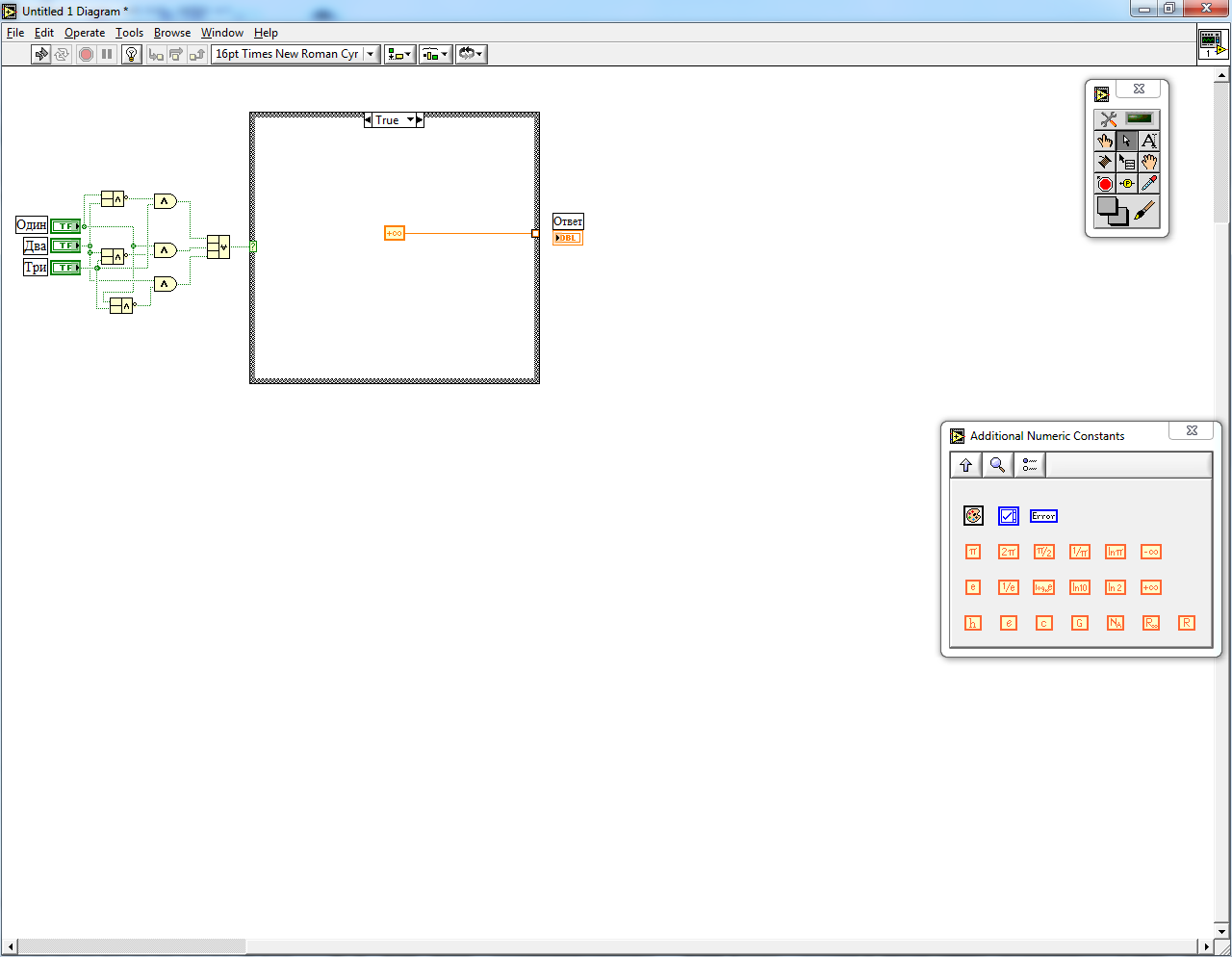


Рисунок 16 – Объединение условий по исключающему ИЛИ (*XOR*). Проверка работоспособности составленной схемы: если условие выполняется – выводится «бесконечность» (*Inf*)

В блоке ложного результата (Рисунок 17) всего выражения выставляется константа не числового значения (*NaN*). Для создания константы не числового значения на блок-диаграмме размещается обыкновенная целочисленная константа и в ней вручную записывается с клавиатуры *NaN* при строгом учёте верхнего и нижнего регистра символов (две заглавных «*N*» объединяются между собой строчной «*a*»).

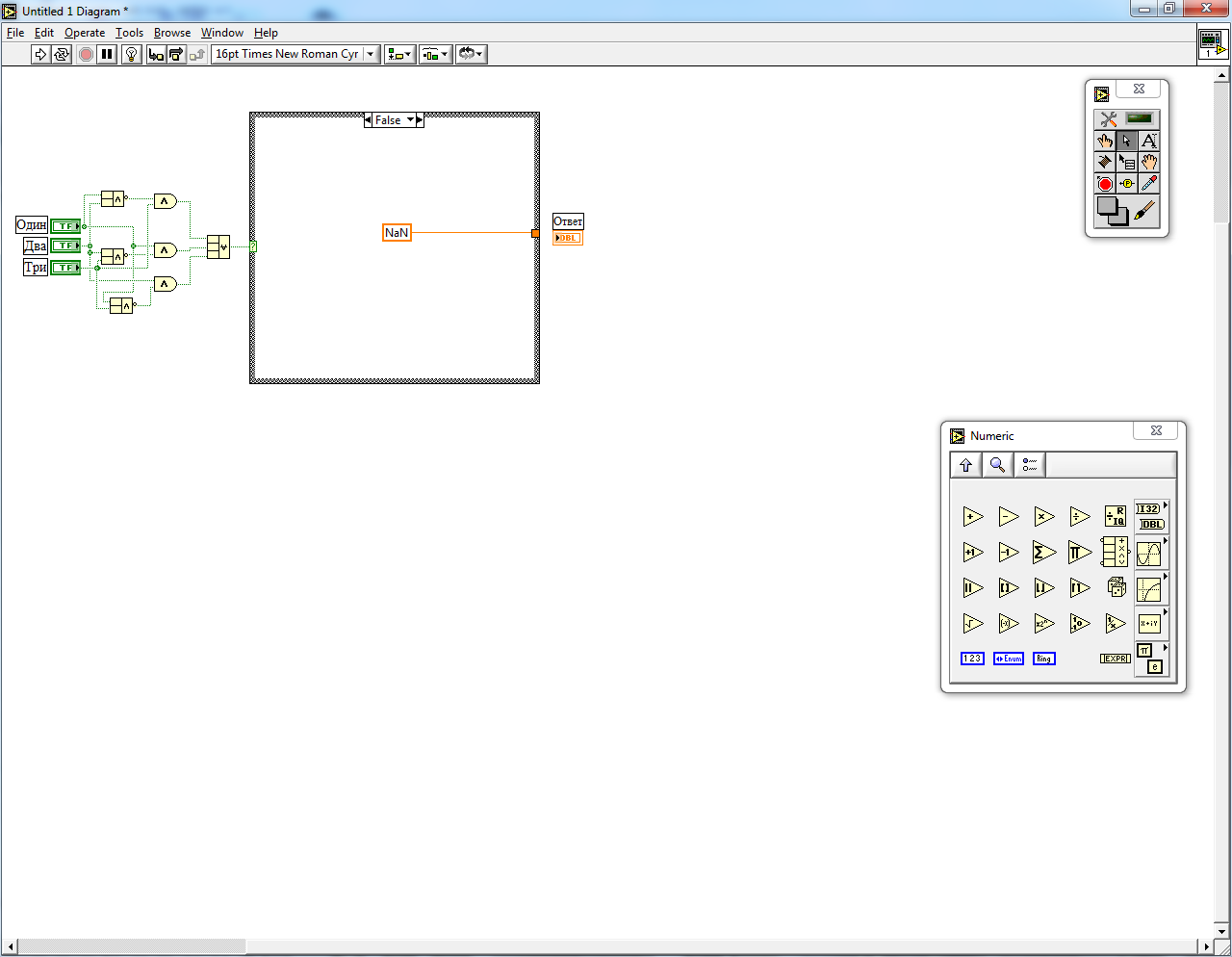


Рисунок 17 – Проверка работоспособности составленной схемы: если условие не выполняется – выводится «не числовое значение» (*NaN*)

После настройки содержимого всех блоков *CASE*-структуры на забудьте соединить её объединённый выход с численным индикатором «Ответ», как это показано на Рисунке 18. Это довольно частая ошибка, которая препятствует запуску виртуального прибора на исполнение.

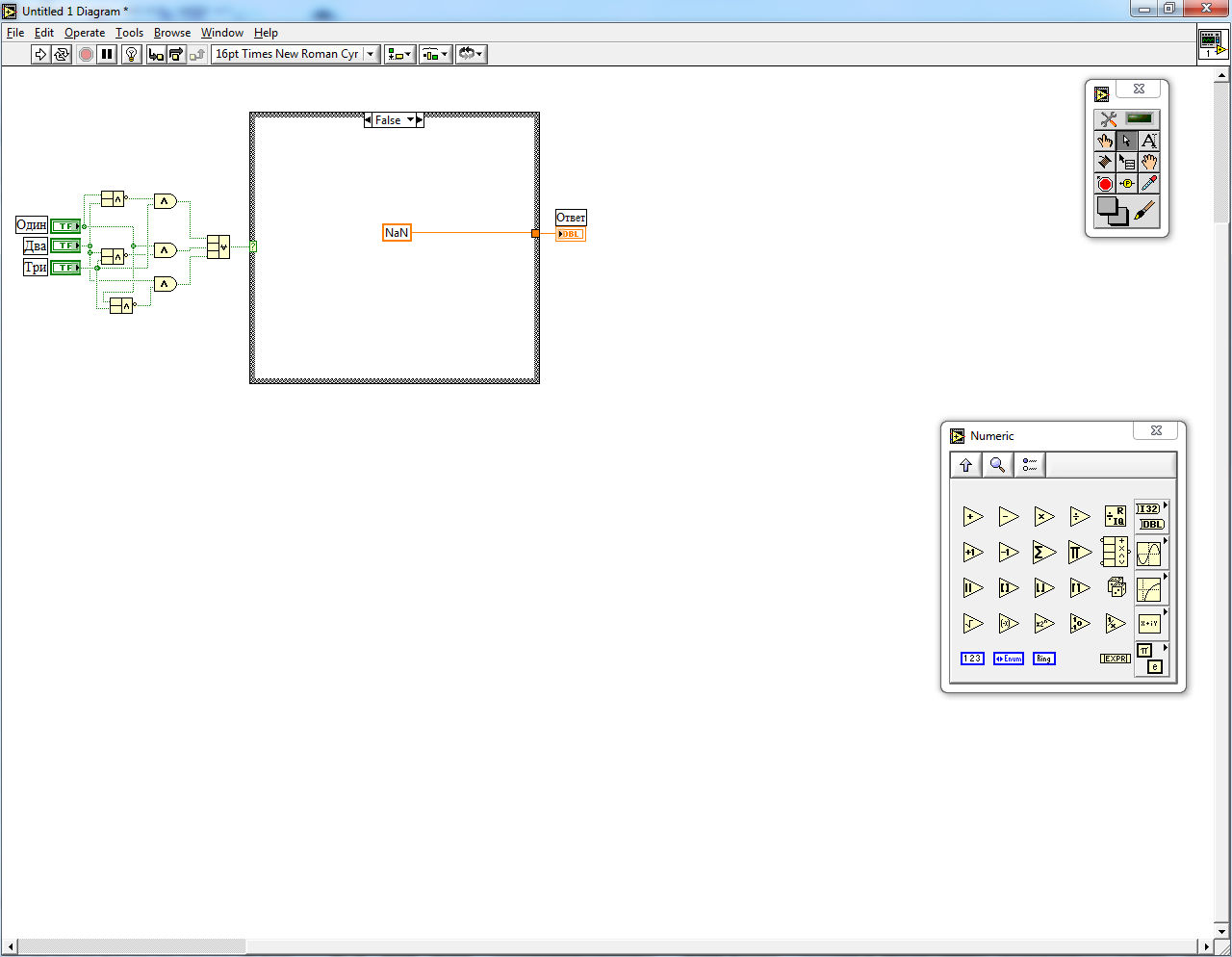


Рисунок 18 – Передача сформированного *CASE*-структурой результата на интерфейсный элемент, выводящий ответ

Запуск виртуального прибора при всех не нажатых кнопках даёт в результате не числовое значение – произошло попадание в блок ложного результата. Согласно составленной ранее таблице истинности так и должно быть (Рисунок 19).

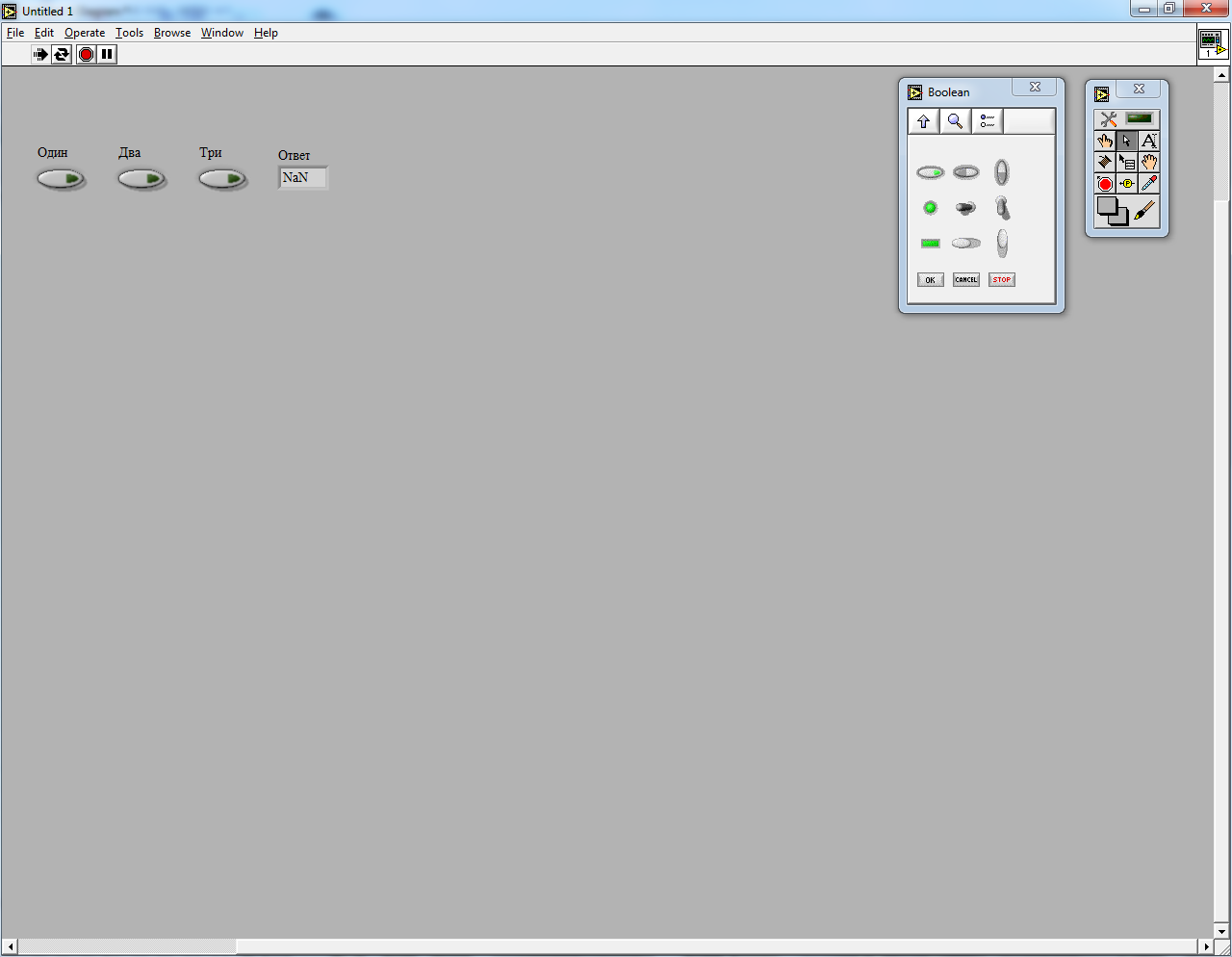


Рисунок 19 – Проверка работоспособности составленной логики: ничего не нажато – результат отрицательный

Запуск виртуального прибора при всех нажатых кнопках тоже даёт в результате не числовое значение в связи с попаданием в блок ложного результата *CASE*-структуры по результату выполнения объединённых логических операций. Согласно составленной ранее таблице истинности, опять же, так и должно быть (Рисунок 20).

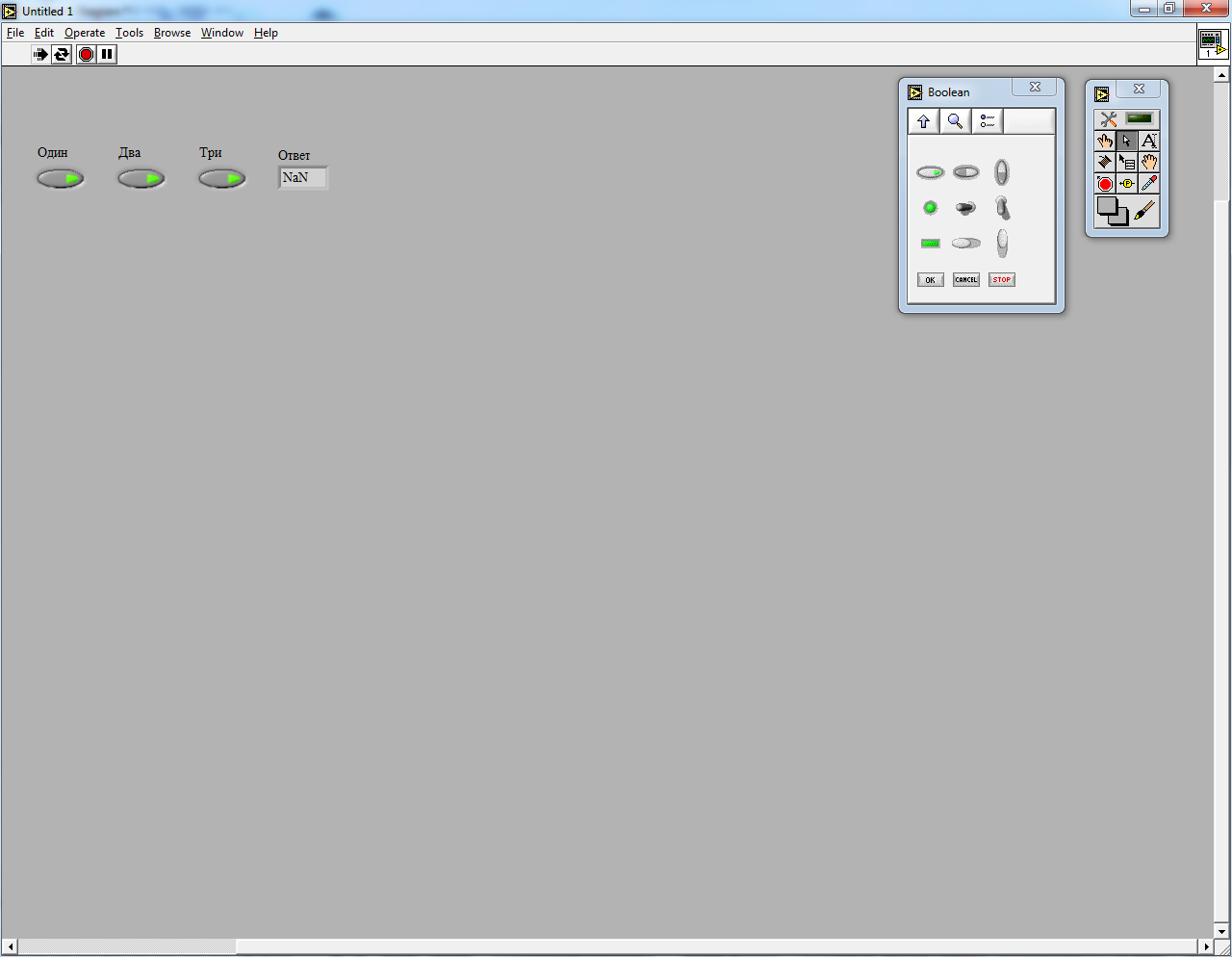


Рисунок 20 – Проверка работоспособности составленной логики: нажаты все кнопки сразу – результат отрицательный

Запуск с нажатыми «Один» и «Три» – шестая строчка таблицы (не считая заголовка). В ответе ложный результат, но так и должно быть (Рисунок 21).

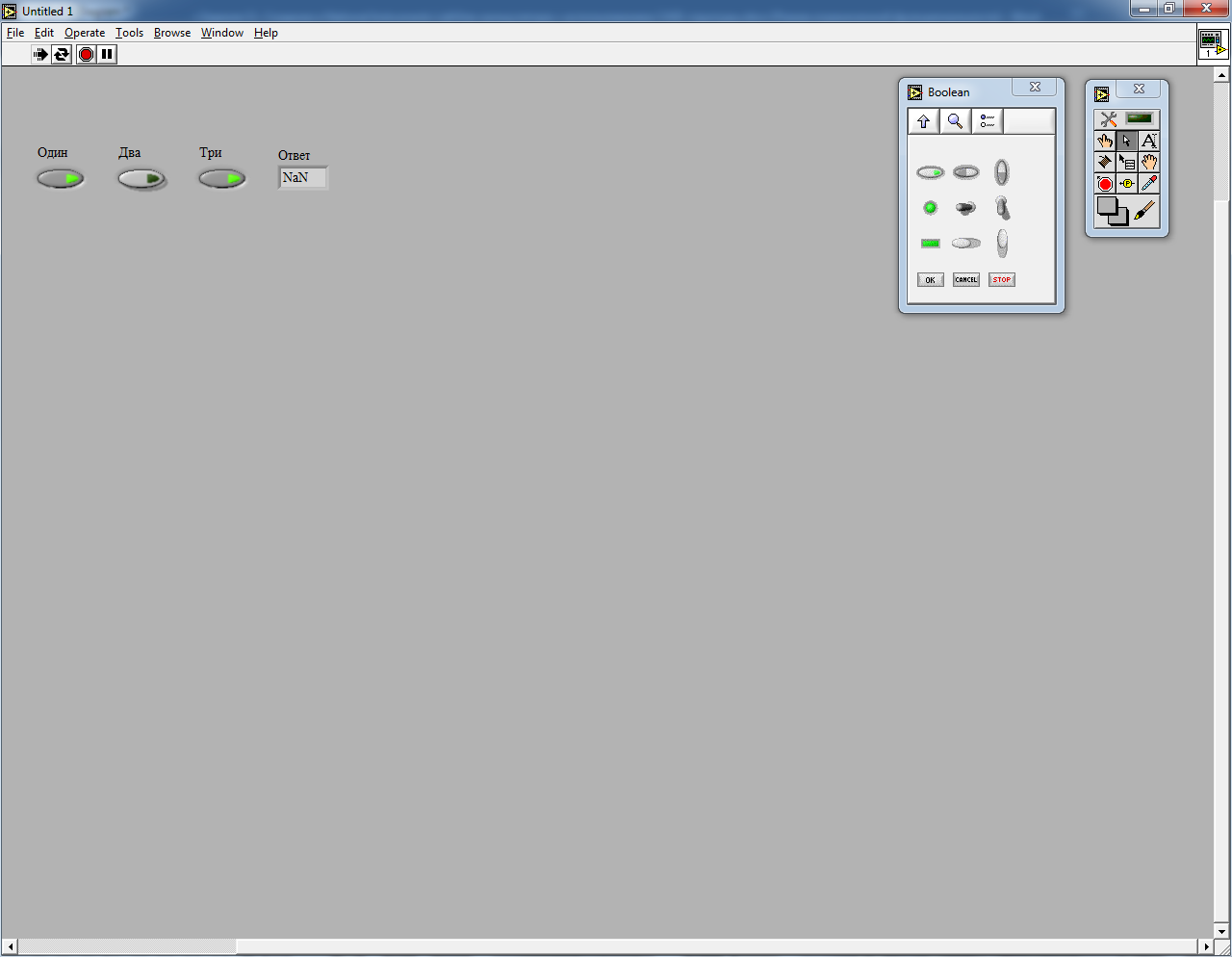


Рисунок 21 – Проверка работоспособности составленной логики: нажаты сразу две кнопки – результат отрицательный

Запуск с нажатыми «Один» и «Два» – седьмая строчка таблицы (не считая заголовка). В ответе ложный результат. Адекватно (Рисунок 22).

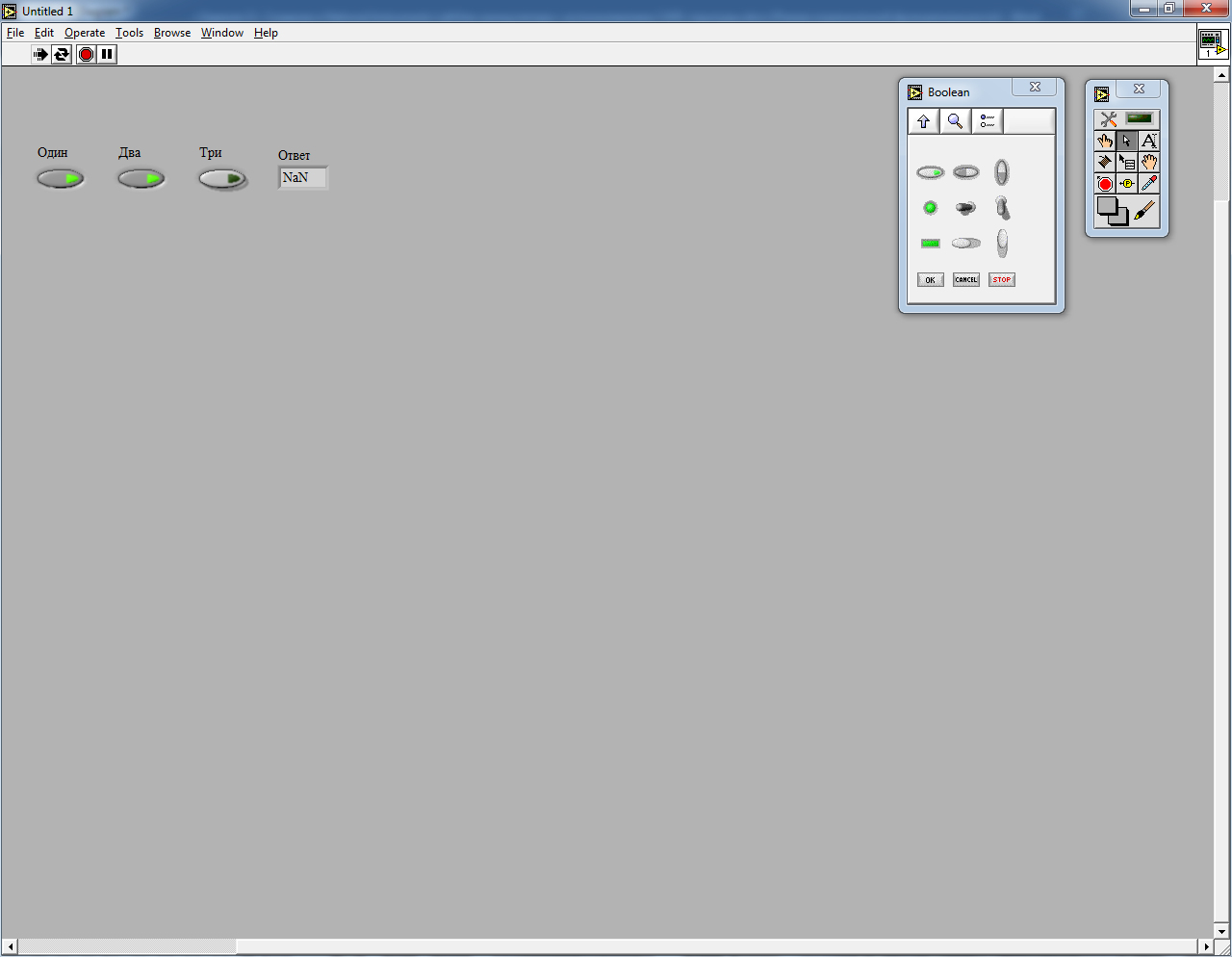


Рисунок 22 – Проверка работоспособности составленной логики: нажаты сразу две другие кнопки – результат отрицательный

Наконец, встречается только «Один» – это пятая строчка таблицы. В ответе плюс бесконечность вместо не числового значения – попали в блок истинного результата. Адекватно (Рисунок 23).

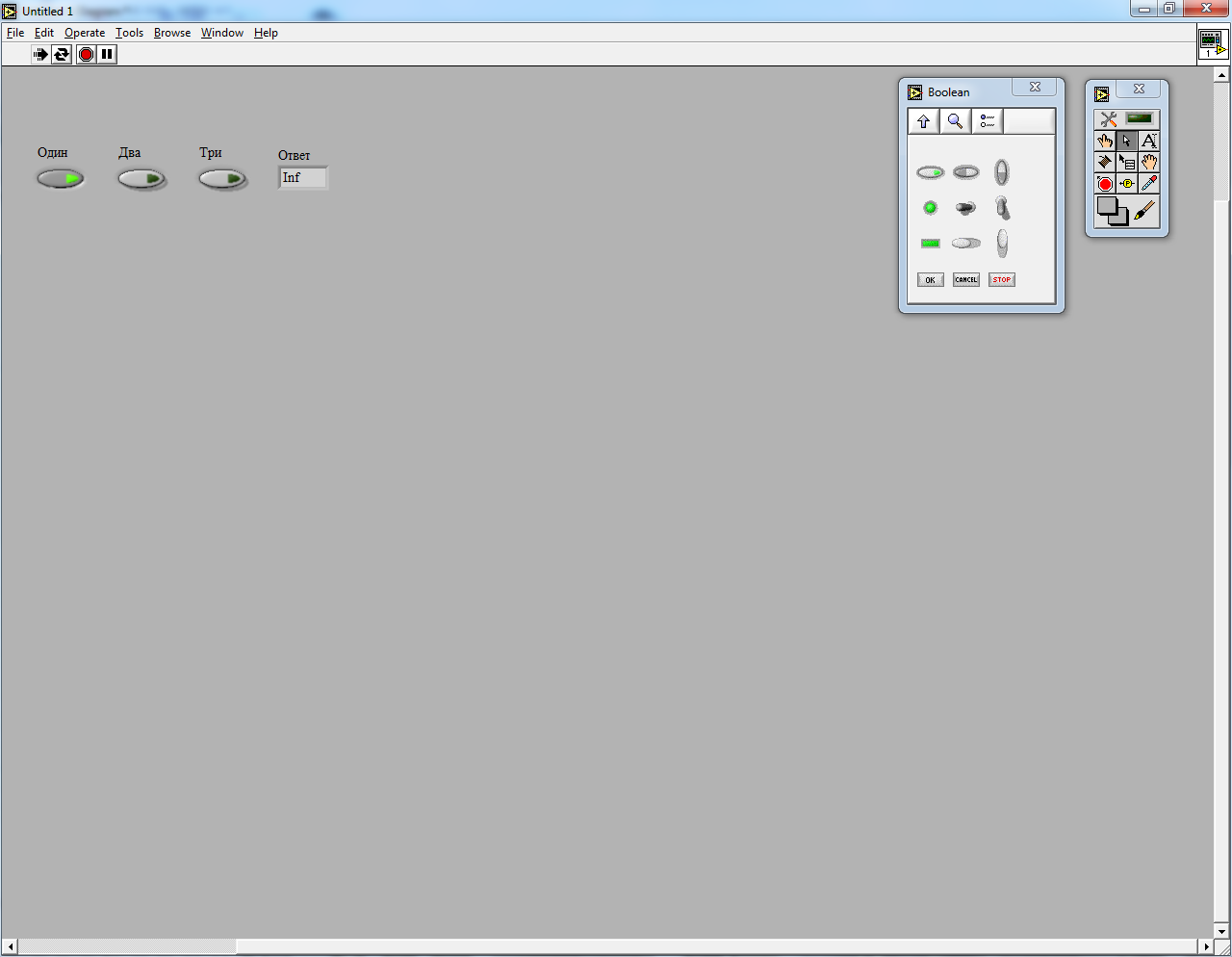


Рисунок 23 – Проверка работоспособности составленной логики: нажата только одна кнопка – результат положительный

После нехитрого тестирования продолжим развитие кода заготовки виртуального прибора «Умный калькулятор» в направлении желаемого результата. В блоке ложного результата контрольной (внешней) *CASE*-структуры оставляем константу не числового значения, а в блоке истинного результата размещаем ещё одну, вложенную *CASE*-структуру. На логический вход вложенной *CASE*-структуры подаём ветвь логики, отвечающей за нажатие «Один» и не нажатие «Два» и «Три». Итого, если нажата кнопка «Один», в блоке истинного результата должна зарождаться численная константа, равная «1» (Рисунок 24). Но если «Один» не нажата, то остаётся ещё два возможных варианта, которые необходимо проверить (либо нажата «Два», либо нажата «Три»). Это ещё одна вложенная *CASE*-структура.

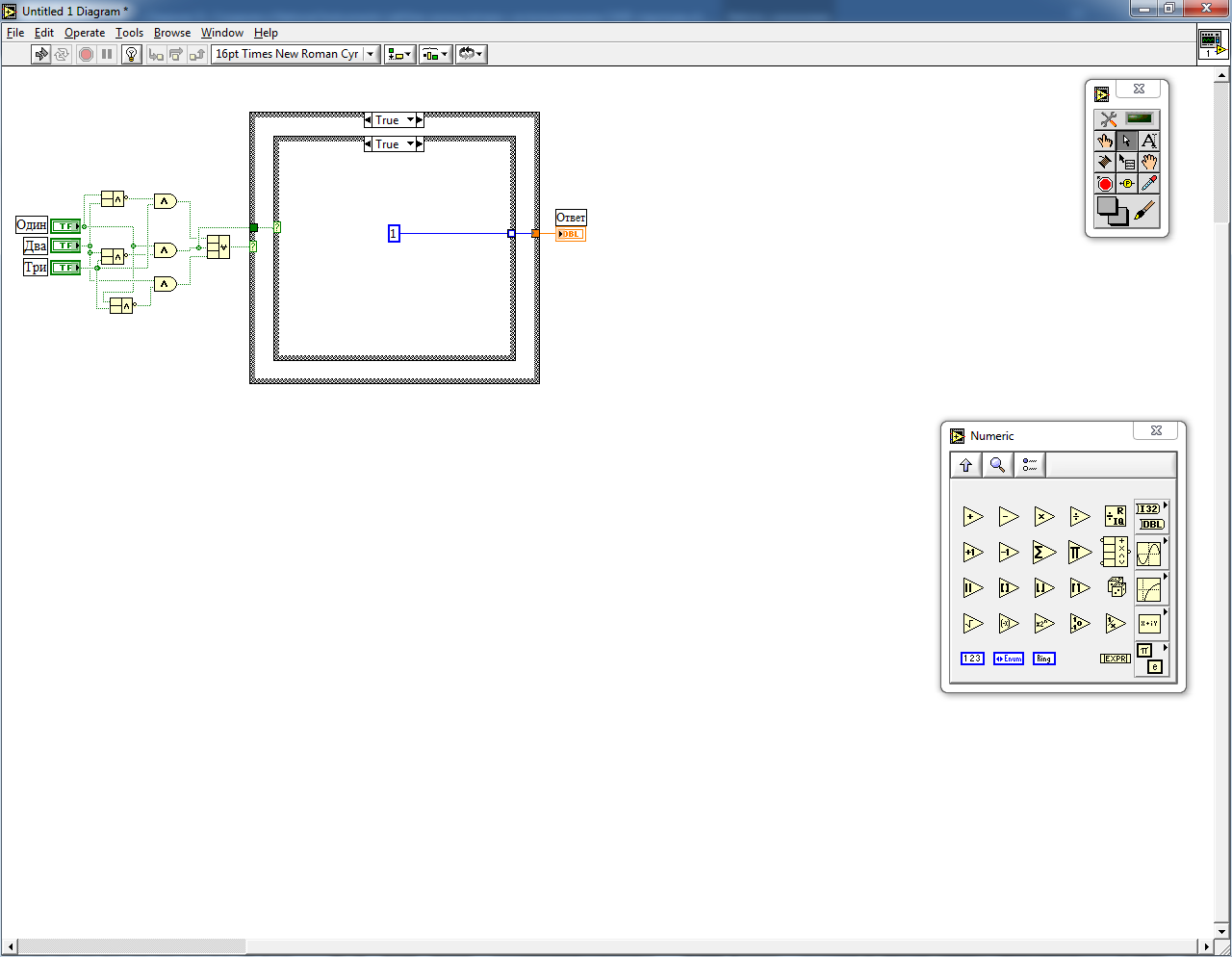


Рисунок 24 – Развитие структуры программы: в случае, если нажата только одна кнопка, и эта кнопка – «Один» должно выводиться в качестве ответа целочисленное значение, равное «1»

На логический вход новой *CASE*-структуры подаём проверку условия нажатия на «Два» (Рисунок 25). Если, действительно, нажата «Два», то должна генерироваться константа, равная «2».

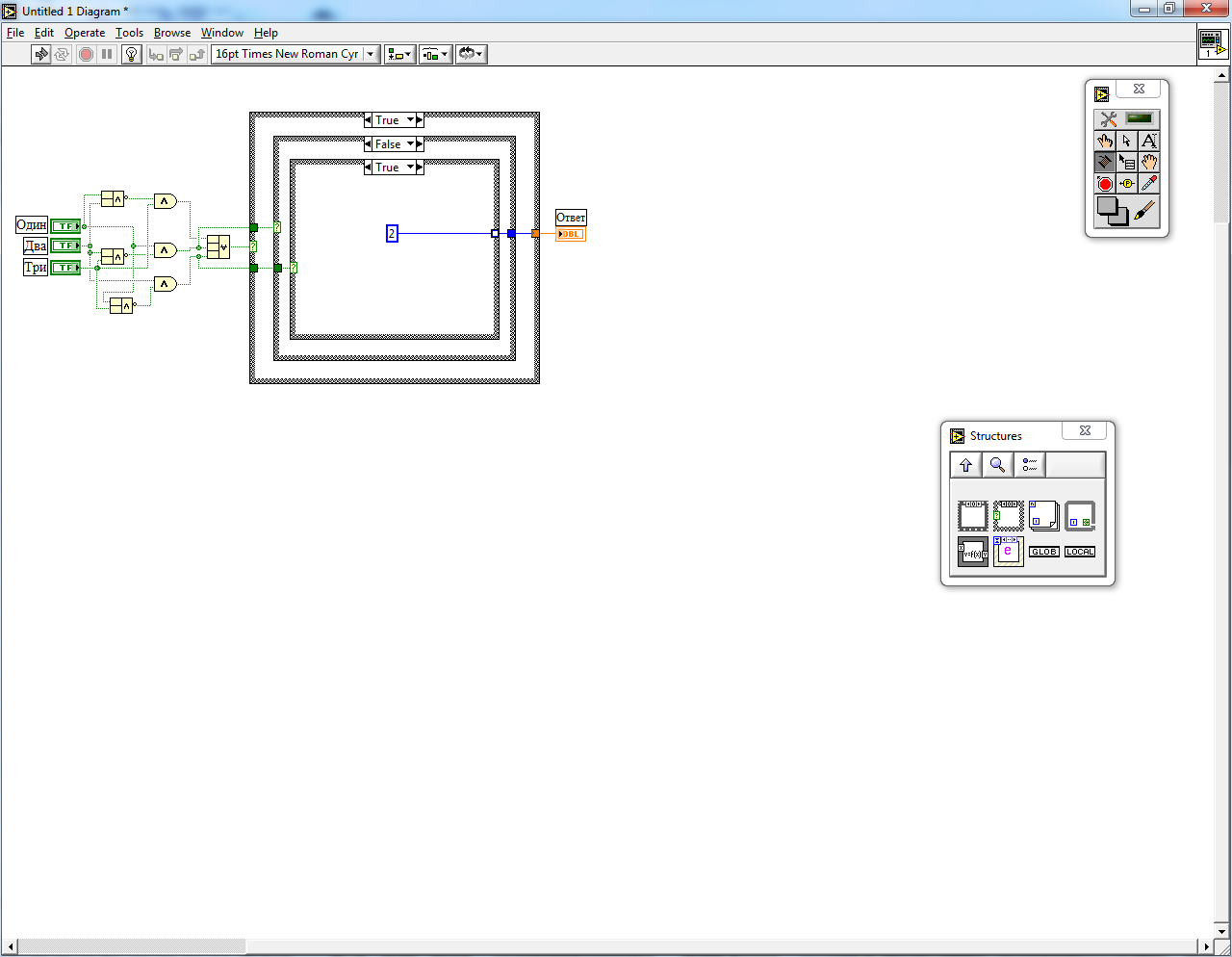


Рисунок 25 – Развитие структуры программы: в случае, если нажата только одна кнопка, и эта кнопка не «Один», а «Два» – должно выводиться в качестве ответа целочисленное значение, равное «2»

Но если не «Два», то остаётся один единственный вариант, что нажата «Три», потому его дополнительной *CASE*-структурой контролировать не нужно, константу со значением «3» генерирует оставшийся незаполненным блок ложного результата (Рисунок 26). Неопытные программисты могут захотеть надстроить ещё одну *CASE*-структуру с целью проверки условия для ветки «Три» по аналогии. Это не будет ошибкой, но в этом случае останется пустой блок ложного результата, в который никогда, ни при каких условиях курсор виртуального прибора не попадёт. Это случай так называемой кодовой избыточности, с которой необходимо бороться.

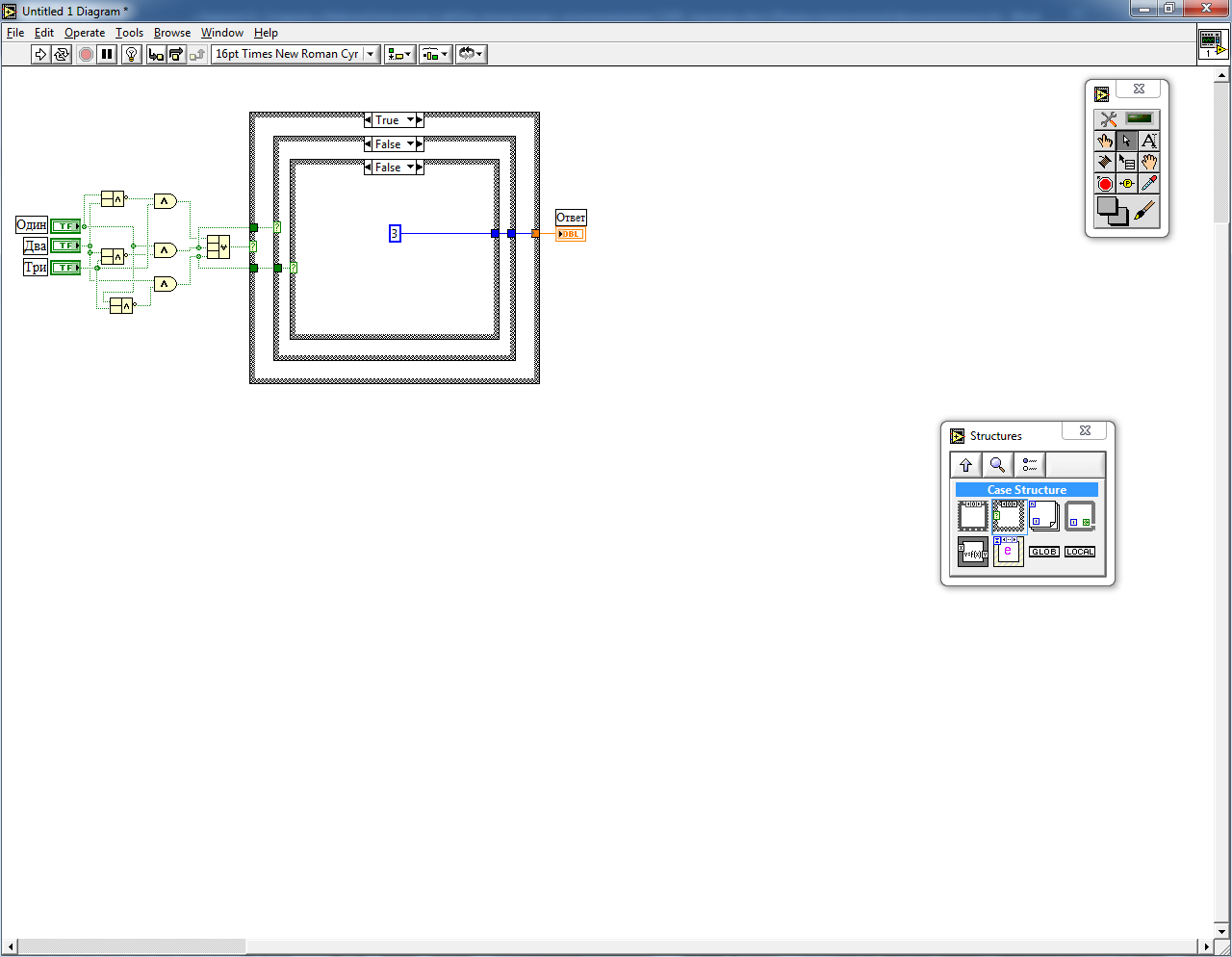


Рисунок 26 – Развитие структуры программы: в случае, если нажата только одна кнопка, и эта кнопка не «Один» и не «Два», то остаётся заключительный вариант – нажата кнопка «Три» и, соответственно, должно выводиться в качестве ответа целочисленное значение, равное «3»

Нижеследующие иллюстрации (Рисунок 27-29) демонстрируют адекватность работы созданного фрагмента виртуального прибора «Умный калькулятор» с учётом ранее проведенных тестов по защите от одновременного выбора нескольких операций.

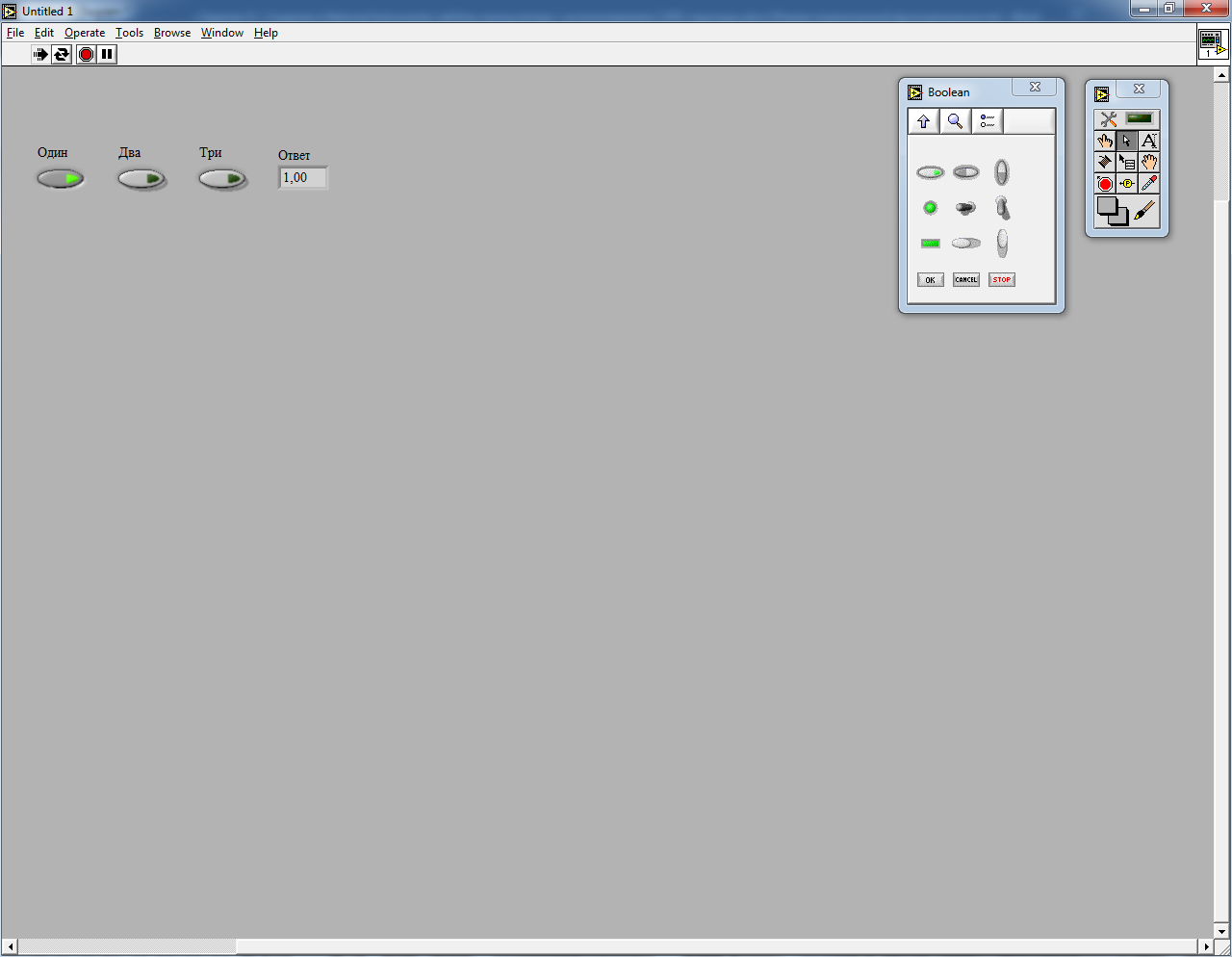


Рисунок 27 – Проверка работоспособности программного обеспечения: нажата кнопка «Один» и в качестве ответа выдаётся значение, равное «1»

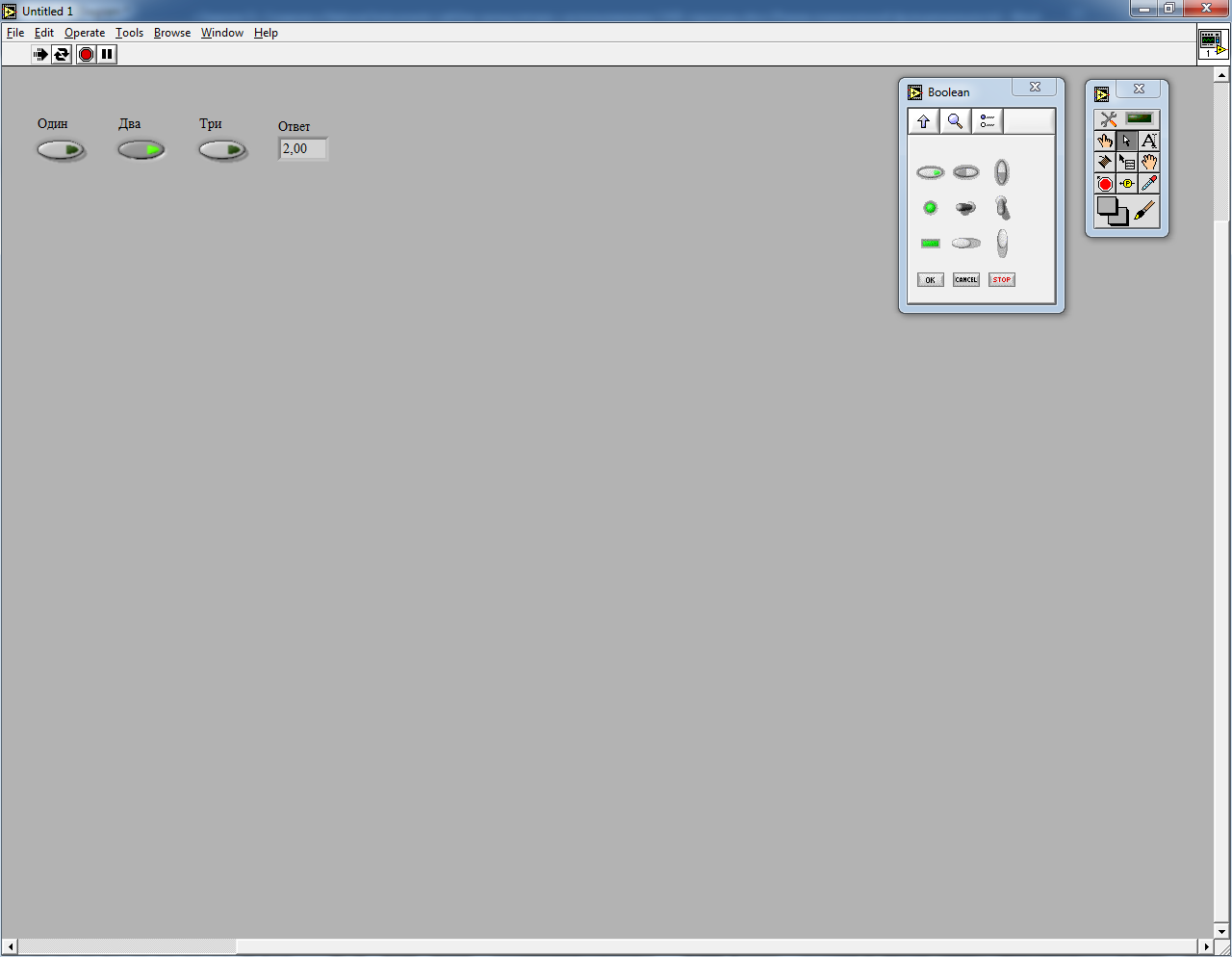


Рисунок 28 – Проверка работоспособности программного обеспечения: нажата кнопка «Два» и в качестве ответа выдаётся значение, равное «2»

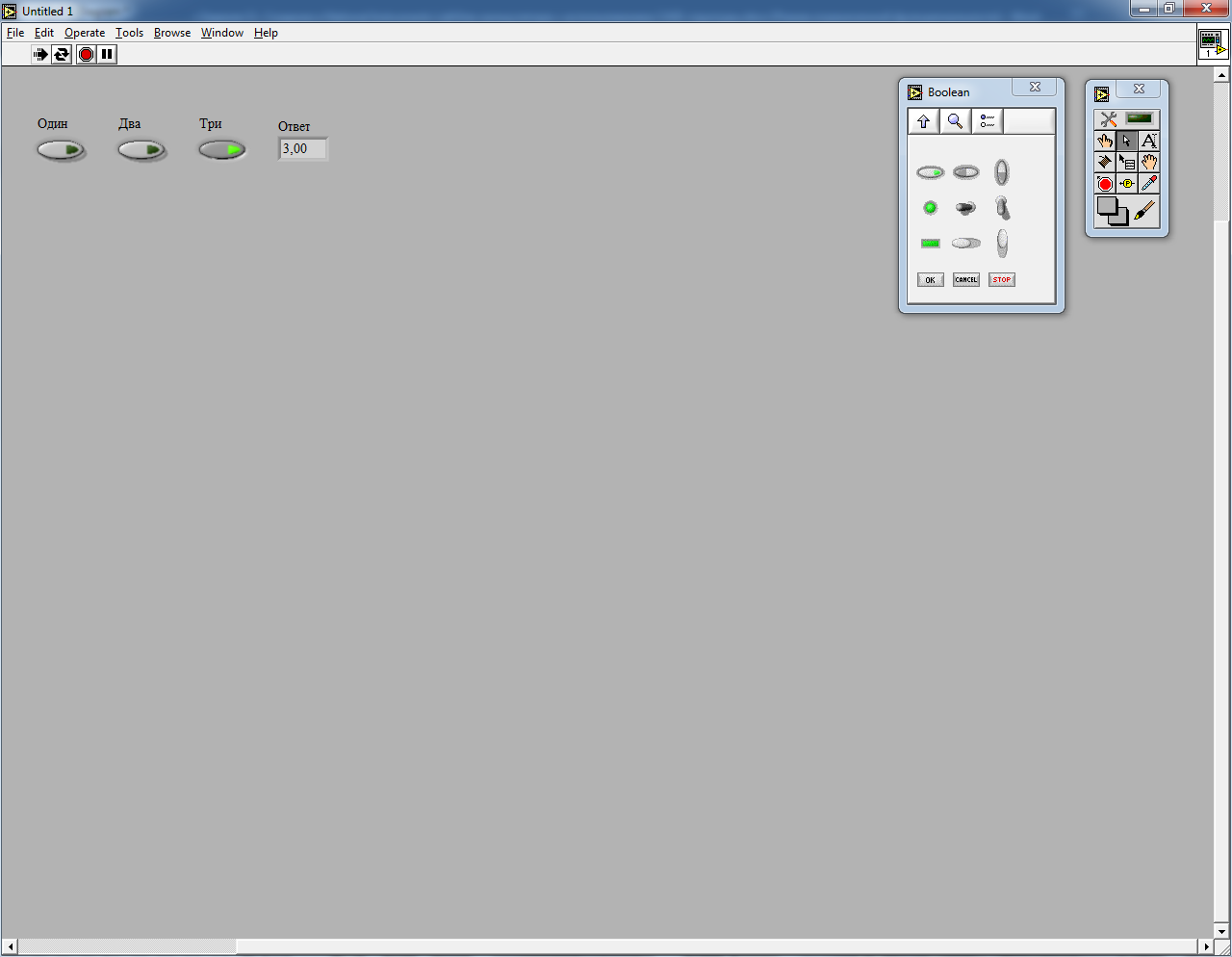


Рисунок 29 – Проверка работоспособности программного обеспечения: нажата кнопка «Три» и в качестве ответа выдаётся значение, равное «3»

**Индивидуальное задание:**

Таблица 3 – Исходные данные для выполнения индивидуального задания

|  |  |
| --- | --- |
| **№ вар.** | **Содержание работы** |
| **1.** | Использовать **три** операнда.  Предусмотреть операции:  - сложения;  - инверсии суммы;  - инверсии суммы объединённой арифметикой;  - сложения объединённой арифметикой. |
| **2.** | Использовать **два** операнда.  Предусмотреть операции:  - умножения;  - деления первого на второй;  - деления второго на первый;  - масштабирования на степень двойки;  - логарифма второго по основанию первого. |
| **3.** | Использовать **один** операнд.  Предусмотреть операции:  - абсолютного значения;  - сигнума;  - синуса;  - экспонирования. |
| **4.** | Использовать **два** операнда.  Предусмотреть операции:  - деления по модулю первого;  - деления по модулю второго;  - инкрементирования первого. |
| **5.** | Использовать **один** операнд.  Предусмотреть операции:  - секанса;  - натурального логарифма;  - декрементирования;  - инверсии;  - получения Пифагоровой константы. |
| **6.** | Использовать **четыре** операнда.  Предусмотреть операции:  - инверсии суммы объединённой арифметикой;  - инверсии разности объединённой арифметикой. |

|  |  |
| --- | --- |
| **7.** | Использовать **два** операнда.  Предусмотреть операции:  - арктангенса с двумя параметрами;  - арктангенса первого;  - арктангенса второго;  - тангенса первого;  - тангенса второго. |
| **8.** | Использовать **один** операнд.  Предусмотреть операции:  - экспонирования;  - взятия логарифма по основанию 2;  - натурального логарифма;  - десятичного логарифма. |
| **9.** | Предусмотреть вывод констант:  - Пифагоровой;  - основания натурального логарифма;  - гравитационной;  - нечислового значения;  - плюс бесконечности;  - минус бесконечности. |
| **10.** | Использовать **два** операнда.  Предусмотреть операции:  - остатка от деления первого на второй;  - остатка от деления второго на первый. |
| **11.** | Использовать **три** операнда.  Предусмотреть операции:  - квадратного корня из суммы квадратов;  - квадратного корня из модуля разности квадратов;  - квадратного корня из модуля разности;  - квадратного корня из модуля суммы. |
| **12.** | Использовать **один** операнд.  Предусмотреть операции:  - арккосинуса;  - округления до ближайшего целого;  - квадратного корня. |
| **13.** | Использовать **три** операнда.  Предусмотреть операции:  - нахождения периметра треугольника;  - нахождения площади треугольника;  - нахождения высоты треугольника. |

|  |  |
| --- | --- |
| **14.** | Использовать **два** операнда.  Предусмотреть операции:  - обращение суммы;  - корень квадратный из суммы;  - деление второго на первый;  - масштабирование на степень двойки;  - логарифм второго по основанию первого. |
| **15.** | Использовать **один** операнд.  Предусмотреть операции:  - нахождения диаметра окружности;  - нахождения площади круга;  - нахождения длины окружности. |
| **16.** | Использовать **четыре** операнда.  Предусмотреть операции:  - нахождения периметра прямоугольника;  - нахождения площади прямоугольника;  - нахождения диагонали прямоугольника. |
| **17.** | Использовать **три** операнда.  Предусмотреть операции:  - нахождения объёма фигуры;  - нахождения площади поперечного сечения фигуры;  - нахождения площади продольного сечения фигуры. |
| **18.** | Использовать **один** операнд.  Предусмотреть операции:  - взятия квадратного корня;  - тангенса;  - обращения;  - возведения двойки в степень. |
| **19.** | Использовать **один** операнд.  Предусмотреть операции:  - математического округления;  - округления в большую сторону;  - округления в меньшую сторону. |
| **20.** | Использовать **два** операнда.  Предусмотреть операции:  - вывода численной константы, равной «8»;  - вычитания второго из первого;  - вычитания первого из второго;  - суммы синуса первого и косинуса второго (синус и косинус в одной операции). |
| **21.** | Использовать **один** операнд.  Предусмотреть операции:  - вычисления значения функции отсчётов;  - декрементирования. |

|  |  |
| --- | --- |
| **22.** | Использовать **три** операнда.  Предусмотреть операции:  - произведения трёх;  - разности суммы первого и третьего из второго;  - суммы трёх;  - частного произведения второго на третье на первое. |
| **23.** | Использовать **один** операнд.  Предусмотреть операции:  - косинуса;  - степени десяти;  - вывода постоянной Планка;  - инкрементирования;  - декрементирования. |
| **24.** | Предусмотреть вывод констант:  - нечисловое значение;  - Авогадро;  - обращённой Пифагорову;  - натурального логарифма десяти. |
| **25.** | Использовать **три** операнда.  Предусмотреть операции:  - вычисления периметра треугольника;  - вычисления высоты треугольника;  - вычисления площади треугольника. |
| **26.** | Использовать **один** операнд.  Предусмотреть операции:  - возведения двойки в степень;  - котангенса;  - вывода обращённого основания натурального логарифма. |
| **27.** | Использовать **два** операнда.  Предусмотреть операции:  - синуса суммы;  - экспонирования суммы;  - обращения суммы;  - возведения двойки в степени суммы. |
| **28.** | Использовать **шесть** операндов.  Предусмотреть операции:  - произведения объединённой арифметикой;  - суммирования объединённой арифметикой. |
| **29.** | Использовать **один** операнд.  Предусмотреть операции:  - косеканса;  - натурального логарифма, проходящего через начало координат. |

|  |  |
| --- | --- |
| **30.** | Использовать **два** операнда.  Предусмотреть операции:  - косинуса разности;  - вычитания из первого квадрата тангенса второго. |