**Задание 10.** В пакете прикладных программ *National Instruments* *LabView* выполнить построение ломаной линии, заданной своим изображением по варианту.

Основное требование: использовать структуру *Formula Node* (обёрнутую в циклическую конструкцию *While* или *For*), внутри которой разветвить вычислительный процесс для построения ломаной линии в диапазоне от начального значения (**xn**) до конечного значения (**xk**), с некоторым шагом (**h**), вводимых через графический пользовательский интерфейс.

С синтаксисом Си-образного языка программирования *Formula Node* ознакомиться через раздел *Help* соответствующей структуры.

**Для чётных вариантов:**

Выход влево и вправо за границы диапазона изображения отрисовывать константами (горизонтальными линиями), соответствующими значениям ординат в граничных точках изображения, заданного по варианту.

**Для нечётных вариантов:**

Выход влево и вправо за границы диапазона изображения отрисовывать с прогнозированием звена вида **y = kx + b** или константы в зависимости от того, что подходит на границе диапазона.

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:**

«*20181226. Задание 10и. Иванов.docx*»

«*20181226. Задание 10и. Иванов.vi*»

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

«*20181226. Задание 10и. Иванов (1).docx*»

«*20181226. Задание 10и. 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)**: освоение навыков использования условного оператора в пакете прикладных программ *National Instruments* *LabView*.

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

**Узел формул**

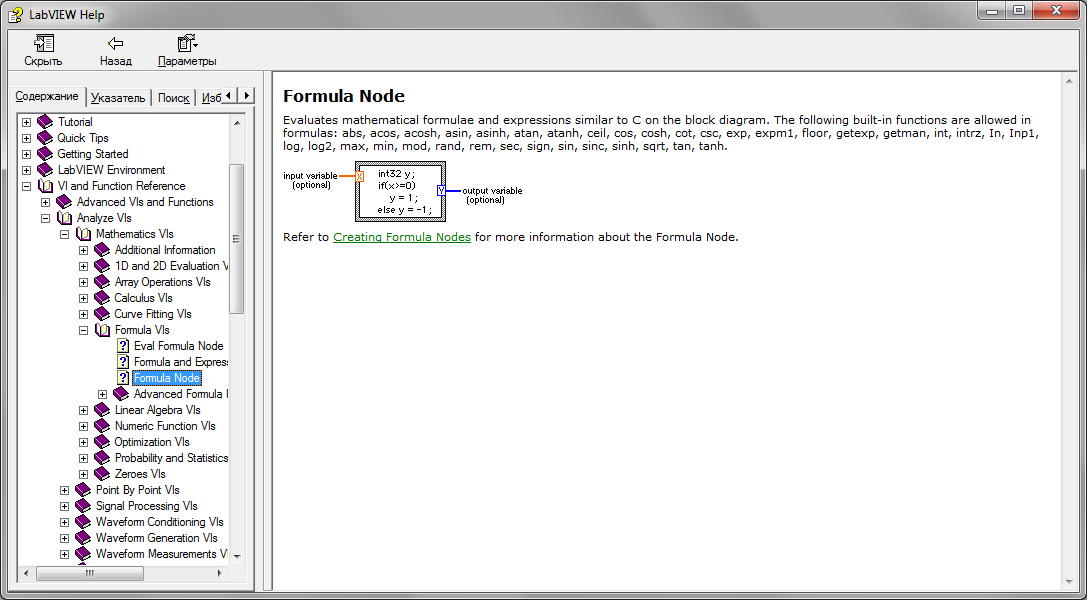


Рисунок 1 – Оригинальный текст подсказки по разделу «Узел формул»

Вычисляет математические формулы и выражения на схожем с языком «Си» языке в рамках блок-диаграммы. Для Узла формул доступны следующие стандартные функции: *abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, In, Inp1, log, log2, max, min, mod, rand, rem, sec, sign, sin, sinc, sinh, sqrt, tan, tanh*.

Более подробная информация по Узлам формул представлена в следующем разделе («Создание Узлов формул»).

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

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

**Создание Узлов формул**

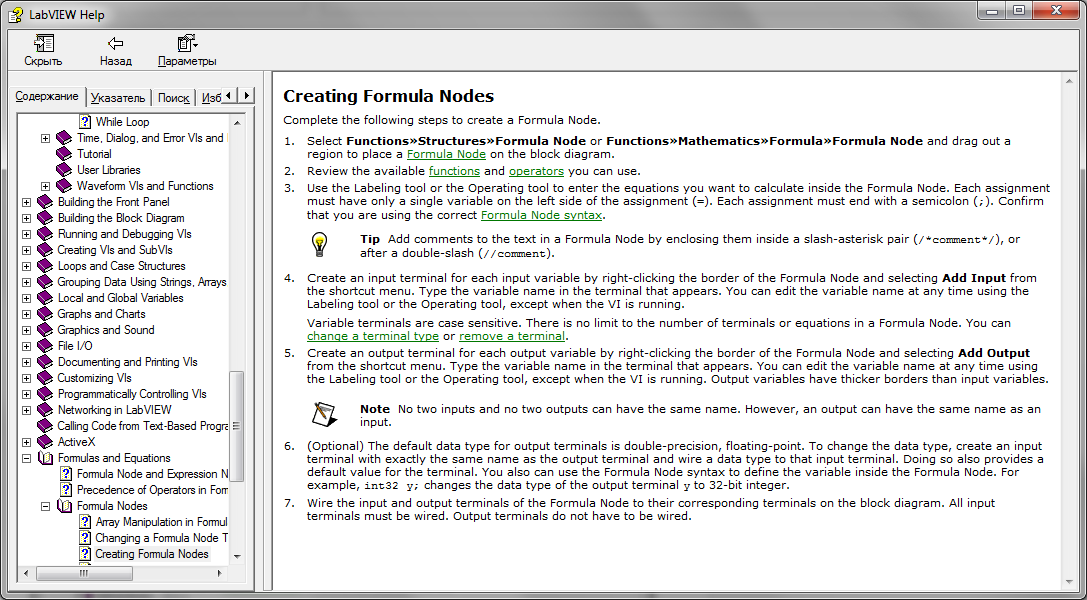


Рисунок 2 – Оригинальный текст подсказки по разделу «Создание Узлов формул»

Следующие шаги позволяют создать Узел формул:

1. Перейдите по разделам «Функции > Структуры > Узел формул» (*Functions > Structures > Formula Node*) или Функции > Математика > Формула > Узел формул (*Functions > Mathematics > Formula > Formula Node*) и перенесите компонент на блок-диаграмму. В соответствующих разделах, расположенных далее по тексту, можно посмотреть перечни функций и операторов, которые могут быть использованы в Узлах формул.

2. При выбранном инструменте ввода текста (или инструменте изменения значения) необходимо ввести выражения или расчётные соотношения внутри Узла формул. Каждое присвоение должно быть выполнено для одной единственной переменной, указанной слева от оператора присвоения (=). Каждое присвоение должно завершаться разделителем (;). В соответствующем разделе можно ознакомиться с синтаксисом Узла формул.

**Внимание:** в Узлах формул уместно добавление комментариев посредство заключения их в пару «косая черта – звёздочка» (/\*комментарий\*/), или вслед за двойной косой чертой (//комментарий).

3. Для каждой переменной, входящей в Узел формул извне, необходимо создать вход-клемму, нажав правой кнопкой мыши на границе Узла формул и выбрав «Добавить вход» (*Add Input***)** в контекстном меню. В появившееся окно клеммы с рыжей рамкой необходимо вписать имя переменной (**Внимание:** предварительно необходимо проверить, чтобы клавиатура находилась в английской раскладке). Редактировать имена клемм возможно в любое время, за исключением ситуаций, когда виртуальный прибор запущен на исполнение.

4. Клеммы являются регистрочувствительными элементами. Количество клемм и выражений в Узлах формул не ограничено. Имеется возможность для изменения типа клемм (вход / выход).

5. Выходная клемма для выходной переменной создаётся следующим образом: нажать правой кнопкой мыши на границе Узла формул и в раскрывшемся контекстном меню выбрать «Добавить выход» (*Add Output*). В возникшем окне напечатать имя выходной клеммы (**Внимание:** предварительно необходимо проверить, чтобы клавиатура находилась в английской раскладке). Изменить имя клеммы можно в любое время за исключением случаев, когда виртуальный прибор запущен на исполнение. Выходные клеммы обладают более толстыми рамками по сравнению со входными.

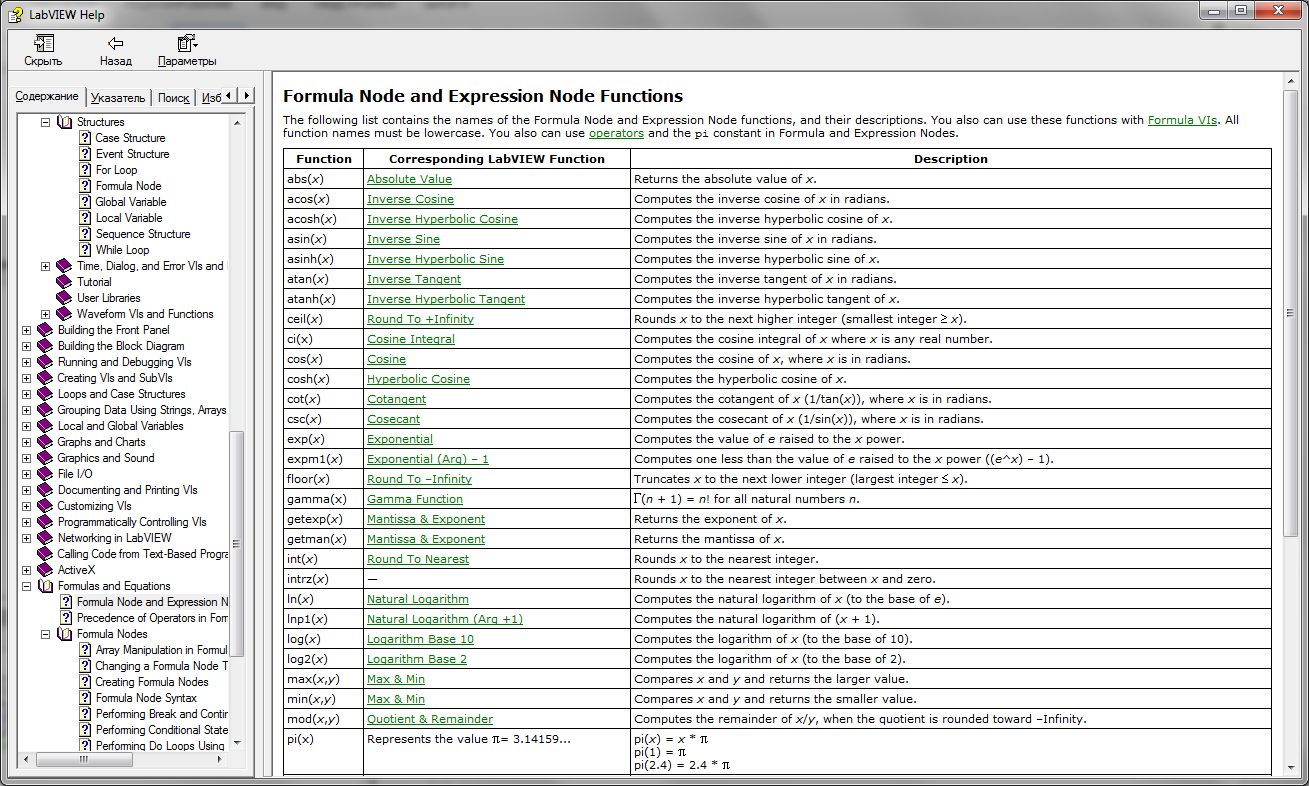
**Внимание:** в Узлах формул не может существовать двух одинаково названных входов и двух одинаково названных выходов, однако, выходы и входы могут иметь одинаковые имена.

(Не обязательная для нормального использования компонента информация) Выходные клеммы по умолчанию выставляются к вещественному типу данных двойной точности. Для изменения типа данных на клеммах, необходимо создать входную клемму с именем, которое в точности совпадает с именем выходной и связать входную клемму с элементом интересующего типа данных. Это действие приводит к изменению типа данных, заданного по умолчанию для всего Узла формул. Также для определения типов данных можно использовать синтаксис Узла формул, выполняя команды внутри самого блока. Например, *int32 y*; изменит тип данных выходной клеммы с именем *y* на 32-битное целое.

Для обеспечения возможности запуска виртуального прибора на исполнение входные и выходные клеммы важно подключить к соответствующим им элементам, расположенным на блок-диаграмме. Все входные клеммы обязательно должны быть соединены проводами. К выходным клеммам жёсткого требования по подсоединению нет.

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

**Функции Узла формул и Узла выражений**



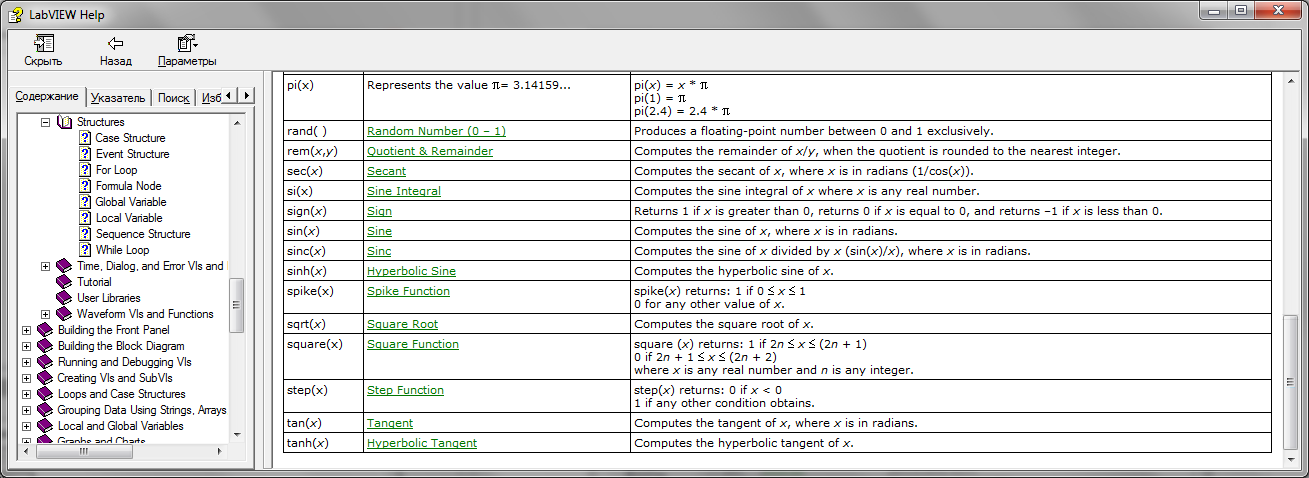


Рисунок 3 – Оригинальный текст подсказки по разделу «Функции Узла формул и Узла выражений»

Нижеследующий список (Таблица 1) содержит наименования функций Узла формул и Узла выражений с их описанием. Эти функции так же могут быть использованы в элементах, расположенных в разделе специальных формул для виртуальных приборов. Все имена функций должны быть строго записаны строчными символами. В Узлах формул и Узлах выражений могут быть использованы операторы, собранные в разделе «Приоритет операторов в Узлах формул и Узлах выражений», а также число Пи.

Таблица 1 – Перечень функций Узла формул

|  |  |  |
| --- | --- | --- |
| **Функция** | **Соответствующая графическая функция LabView** | **Описание** |
| *abs(x)* | [Абсолютная](glang.chm::/Absolute_Value.html) величина | Возвращает абсолютную величину *x*. |
| *acos(x)* | [Значение,](glang.chm::/Inverse_Cosine.html) обратное косинусу | Вычисляет значение, обратное косинусу (арккосинус) *x* в радианах. |
| *acosh(x)* | [Значение,](glang.chm::/Inverse_Hyperbolic_Cosine.html) обратное гиперболическому косинусу | Вычисляет значение, обратное гиперболическому косинусу (гиперболический арккосинус) *x*. |
| *asin(x)* | [Значение,](glang.chm::/Inverse_Sine.html) обратное синусу | Вычисляет значение, обратное синусу (арксинус) *x* в радианах. |
| *asinh(x)* | [Значение,](glang.chm::/Inverse_Hyperbolic_Sine.html) обратное гиперболическому синусу | Вычисляет значение, обратное гиперболическому синусу (гиперболический арксинус) *x*. |
| *atan(x)* | [Значение,](glang.chm::/Inverse_Tangent.html) обратное тангенсу | Вычисляет значение, обратное тангенсу (арктангенс) *x* в радианах. |
| *atanh(x)* | [Значение,](glang.chm::/Inverse_Hyperbolic_Tangent.html) обратное гиперболическому тангенсу | Вычисляет значение, обратное гиперболическому тангенсу (гиперболический арктангенс) *x*. |
| *ceil(x)* | [Округление](glang.chm::/Round_To_posInfinity.html) до целого в большую сторону | Округляет *x* до следующего, большего целого значения (самое маленькое целое >= *x*). |
| *ci(x)* | [Интеграл](gmath.chm::/Cosine_Integral.html) косинуса | Вычисляет интеграл косинуса *x,* где *x* is любое вещественное число. |
| *cos(x)* | [Косинус](glang.chm::/Cosine.html) | Вычисляет косинус *x*, где *x* в радианах. |
| *cosh(x)* | [Гиперболический](glang.chm::/Hyperbolic_Cosine.html) косинус | Вычисляет гиперболический косинус *x*. |
| *cot(x)* | [Котангенс](glang.chm::/Cotangent.html) | Вычисляет котангенс *x* (*1/tan(x)*), где *x* is в радианах. |
| *csc(x)* | [Косеканс](glang.chm::/Cosecant.html) | Вычисляет косеканс *x* (*1/sin(x)*), где *x* в радианах. |
| *exp(x)* | [Экспонента](glang.chm::/Exponential.html) | Вычисляет значение *e*, возведённое в степень *x*. |
| *expm1(x)* | [Экспонента,](glang.chm::/Exponential_Arg_minus_1.html) смещённая в начало координат | Вычисляет значение, на единицу меньшее, чем *e*, возведённое в степень *x* (*(e^x) – 1*). |
| *floor(x)* | [Округление](glang.chm::/Round_To_negInfinity.html) до ближайшего целого в меньшую сторону | Отсекает от *x* вещественную часть до целого значения (самое большое целое <= *x*). |
| *gamma(x)* | [Гамма-функция](gmath.chm::/Gamma_Function.html) | *Г(n + 1) = n!* для всех натуральных значений *n*. |
| *getexp(x)* | [Мантисса](glang.chm::/Mantissa_and_Exponent.html) и экспоненциальная форма | Возвращает *x* в экспоненциальной форме. |
| *getman(x)* | Мантисса и экспоненциальная форма | Возвращает мантиссу *x*. |
| *int(x)* | [Округление](glang.chm::/Round_To_Nearest.html) до ближайшего целого | Округляет *x* до ближайшего целого. |
| *intrz(x)* | [без названия] | Округляет *x* до ближайшего целого между *x* и нулём. |
| *ln(x)* | [Натуральный](glang.chm::/Natural_Logarithm.html) логарифм | Вычисляет натуральный логарифм *x* (по основанию *e*). |

|  |  |  |
| --- | --- | --- |
| *lnp1(x)* | [Натуральный](glang.chm::/Natural_Logarithm_Arg_1.html) логарифм, смещённый в начало координат | Вычисляет натуральный логарифм (*x* + 1). |
| *log(x)* | [Логарифм](glang.chm::/Logarithm_Base_10.html) по основанию 10 | Вычисляет логарифм *x* (по основанию 10). |
| *log2(x)* | [Логарифм](glang.chm::/Logarithm_Base_2.html) по основанию 2 | Вычисляет логарифм *x* (по основанию 2). |
| *max(x,y)* | [Максимум](glang.chm::/Max_and_Min.html) и минимум | Сравнивает *x* с *y* и возвращает большее из значений. |
| *min(x,y)* | [Максимум](glang.chm::/Max_and_Min.html) и минимум | Сравнивает *x* с *y* и возвращает меньшее из значений. |
| *mod(x,y)* | [Целое](glang.chm::/Quotient_and_Remainder.html) от деления и остаток | Вычисляет остаток от деления *x*/*y*, при остатке, округляемом в меньшую сторону. |
| *pi(x)* | Возвращает точное значение числа Пи 3.14159... | pi(*x*)=*x*\*Пи pi(1)=Пи  pi(2.4)=2.4\*Пи |
| *rand( )* | [Случайное](glang.chm::/Random_Number_0_1.html) число из диапазона (0…1) | Выдаёт значение с плавающей точкой между 0 и 1 исключительно. |
| *rem(x,y)* | [Целое](glang.chm::/Quotient_and_Remainder.html) от деления и остаток | Вычисляет целую часть от деления *x*/*y*, при остатке, округляемом до ближайшего целого. |
| *sec(x)* | [Секанс](glang.chm::/Secant.html) | Вычисляет секанс *x*, где *x* в радианах (*1/cos(x)*). |
| *si(x)* | [Интеграл](gmath.chm::/Sine_Integral.html) синуса | Вычисляет интеграл синуса *x*, где *x* любое вещественное число. |
| *sign(x)* | [Сигнум](glang.chm::/Sign.html) | Возвращает  1, если *x* больше 0,  0, если *x* равен 0,  –1, если *x* меньше 0. |
| *sin(x)* | [Синус](glang.chm::/Sine.html) | Вычисляет синус *x*, где *x* в радианах. |
| *sinc(x)* | [Функция](glang.chm::/Sinc.html) отсчётов | Вычисляет синус *x*, делённый на *x* (*sin(x)/x*), где *x* в радианах. |
| *sinh(x)* | [Гиперболический](glang.chm::/Hyperbolic_Sine.html) синус | Вычисляет гиперболический синус *x*. |
| *spike(x)* | Строб-функция | *spike(x)* возвращает:  1, если 0 < *x* < 1,  0 для любого другого *x*. |
| *sqrt(x)* | Квадратный корень | Вычисляет квадратный корень *x*. |
| *square(x)* | Квадратичная функция | *square(x)* возвращает:  1, если 2*n <= x <=* (2*n* + 1),  0, если 2*n* + 1 <= *x <=* (2*n* + 2),  где *x* – любое вещественное и *n* – любое целое. |
| *step(x)* | Ступенчатая функция | *step(x)* возвращает:  0, если *x* < 0,  1, если выполняется любое другое условие. |
| *tan(x)* | Тангенс | Вычисляет тангенс *x*, где *x* задаётся в радианах. |
| *tanh(x)* | Гиперболический тангенс | Вычисляет гиперболический тангенс от *x*. |

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

**Приоритет операторов в Узлах формул и Узлах выражений**

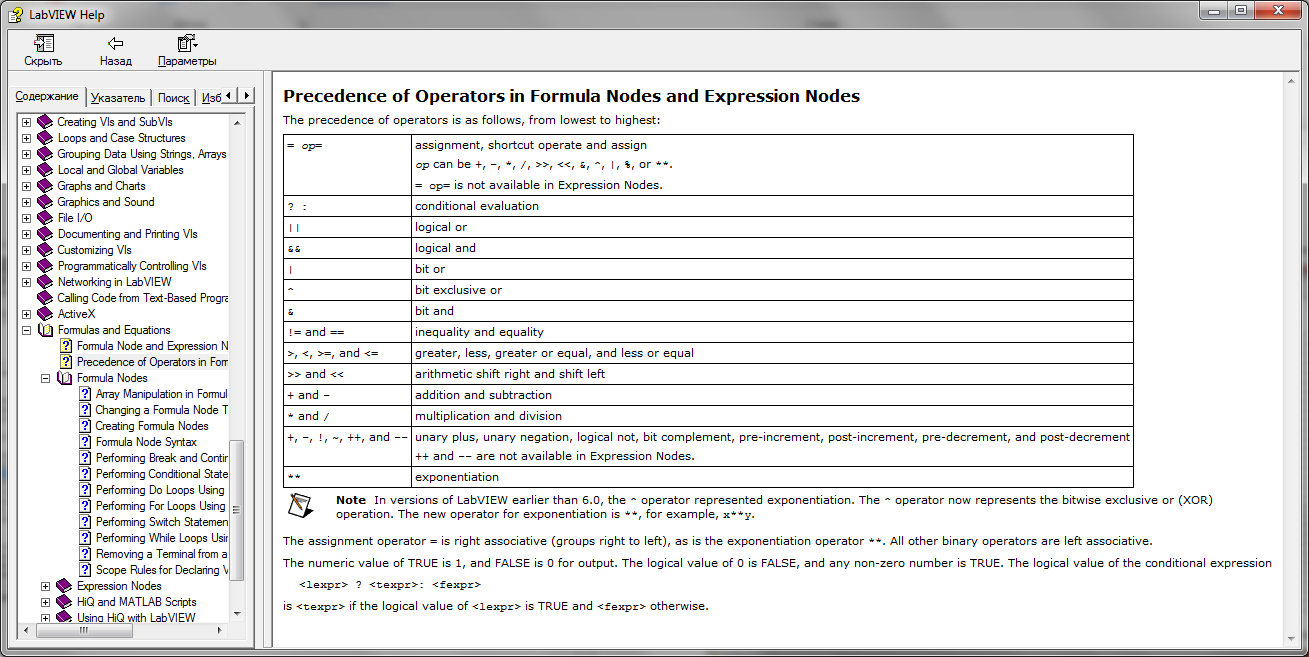


Рисунок 4 – Оригинальный текст подсказки по разделу «Функции Узла формул и Узла выражений»

Приоритет выполняемых операторов изменяется в соответствии с представленным в Таблице 2 перечнем от низкого до высокого.

Таблица 2 – Приоритет операторов в Узлах формул

|  |  |
| --- | --- |
| = op= | переопределение оператора op может быть: «+», «-», «\*», «/», «>>», «<<», «&», «^», «|», «%», или «\*\*». = *op*= не применяется в Узлах выражений. |
| ? : | условное вычисление |
| || | логическое ИЛИ |
| && | логическое И |
| | | битовое ИЛИ |
| ^ | битовое исключающее ИЛИ |
| & | битовое И |
| != и == | неравенство и равенство |
| >, <, >=, и <= | больше, меньше, больше или равно, и меньше или равно |
| >> и << | арифметический сдвиг вправо и арифметический сдвиг влево |
| + и - | сложение и вычитание |
| \* и / | умножение и деление |
| +, -, !, ~, ++, и -- | унарный плюс, унарный минус, логическое НЕ, битовое дополнение, пред инкремент, пост инкремент, пред декремент и пост декремент.  **Внимание:** ++ и -- не применимы для Узлов выражений. |
| \*\* | возведение в степень |

**Внимание:** В версиях *LabView* до 6.0 для возведения в степень используется оператор «^». Начиная с версии 6.1 оператор «^» используется в качестве исключающего ИЛИ (*XOR*). Новым оператором для возведения в степень является «*\*\**», например, *x\*\*y*.

Оператор присвоения (=) является право-ассоциативным (группирует справа налево), как и оператор возведения в степень (\*\*). Все остальные бинарные операторы являются лево-ассоциативными.

Численным значением «ИСТИНЫ» является «1», и для «ЛЖИ» численным значением является «0» на выходе. Логическим значением «0» является «ЛОЖЬ», и любое ненулевое численное значение является «ИСТИНОЙ». Логическое значение условного выражения

*<lexpr> ? <texpr>: <fexpr>*

является *<texpr>* (выражением при истине), если логическое значение *<lexpr>* (логического выражения) является «ИСТИНОЙ» и *<fexpr>* (выражением при лжи) в остальных случаях.

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

**Синтаксис Узла формул**

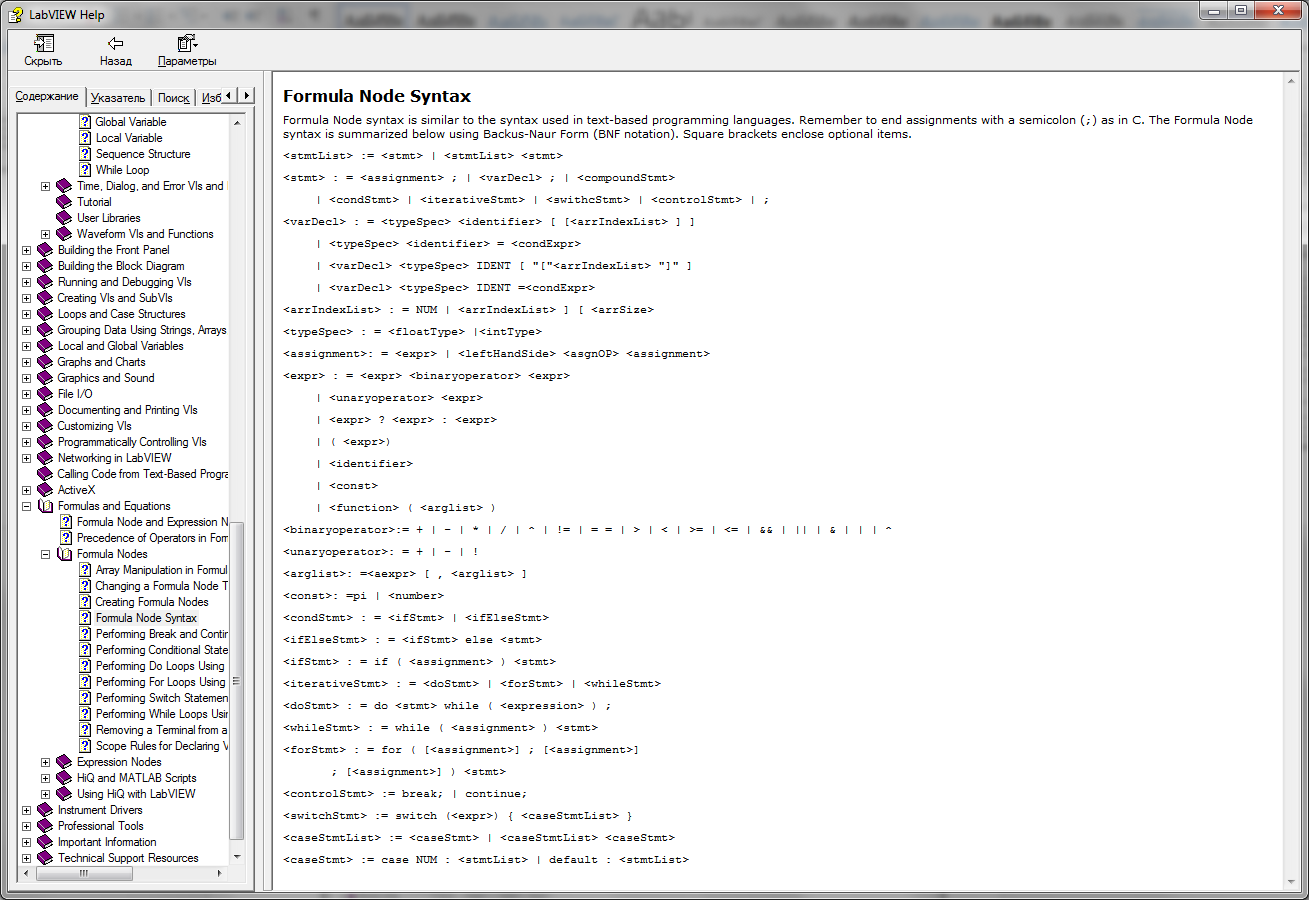


Рисунок 5 – Оригинальный текст подсказки по разделу «Синтаксис Узла формул»

Синтаксис Узла формул схож с синтаксисом, используемым в текстуальных языках программирования. Важно помнить о постановке разделителя по завершении оператора (;) как и на языке Си. Синтаксис Узла формул использует форму Бакуса-Наура (*BNF*-нотацию). В квадратные скобки размещены необязательные элементы.

*<stmtList> := <stmt> | <stmtList> <stmt>*

*<stmt> : = <assignment> ; | <varDecl> ; | <compoundStmt>*

*| <condStmt> | <iterativeStmt> | <swithcStmt> | <controlStmt> | ;*

*<varDecl> : = <typeSpec> <identifier> [ [<arrIndexList> ] ]*

*| <typeSpec> <identifier> = <condExpr>*

*| <varDecl> <typeSpec> IDENT [ "["<arrIndexList> "]" ]*

*| <varDecl> <typeSpec> IDENT =<condExpr>*

*<arrIndexList> : = NUM | <arrIndexList> ] [ <arrSize>*

*<typeSpec> : = <floatType> |<intType>*

*<assignment>: = <expr> | <leftHandSide> <asgnOP> <assignment>*

*<expr> : = <expr> <binaryoperator> <expr>*

*| <unaryoperator> <expr>*

*| <expr> ? <expr> : <expr>*

*| ( <expr>)*

*| <identifier>*

*| <const>*

*| <function> ( <arglist> )*

*<binaryoperator>:= + | - | \* | / | ^ | != | = = | > | < | >= | <= | && | || | & | | | ^*

*<unaryoperator>: = + | - | !*

*<arglist>: =<aexpr> [ , <arglist> ]*

*<const>: =pi | <number>*

*<condStmt> : = <ifStmt> | <ifElseStmt>*

*<ifElseStmt> : = <ifStmt> else <stmt>*

*<ifStmt> : = if ( <assignment> ) <stmt>*

*<iterativeStmt> : = <doStmt> | <forStmt> | <whileStmt>*

*<doStmt> : = do <stmt> while ( <expression> ) ;*

*<whileStmt> : = while ( <assignment> ) <stmt>*

*<forStmt> : = for ( [<assignment>] ; [<assignment>]*

*; [<assignment>] ) <stmt>*

*<controlStmt> := break; | continue;*

*<switchStmt> := switch (<expr>) { <caseStmtList> }*

*<caseStmtList> := <caseStmt> | <caseStmtList> <caseStmt>*

*<caseStmt> := case NUM : <stmtList> | default : <stmtList>*

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

**Изменение типа клемм в Узлах формул**

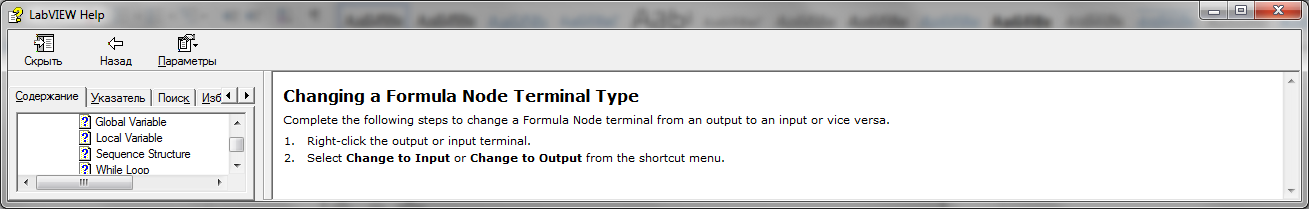


Рисунок 6 – Оригинальный текст подсказки по разделу «Изменение типов клемм в Узлах формул»

Выполните следующие шаги для изменения типа клеммы Узла формул с выхода на вход или со входа на выход:

1. Щёлкните правой кнопкой мыши по выходной или входной клемме.

2. В контекстном меню выберите «Изменить на вход» (*Change to Input*) или «Изменить на выход» (*Change to Output*).

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

**Удаление клемм с Узлов формул**

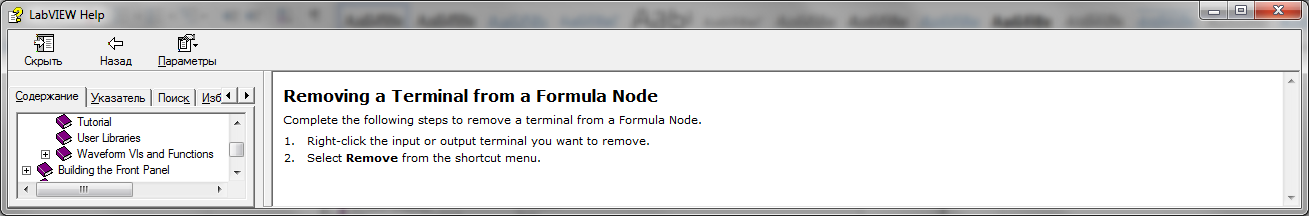


Рисунок 7 – Оригинальный текст подсказки по разделу «Удаление клемм с Узлов формул»

Выполните следующие шаги для удаления клемм с Узлов формул:

1. Щёлкните правой кнопкой мыши на входной или выходной клемме.

2. Выберите пункт контекстного меню удалить (*Remove*) в контекстном меню.

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

**Пример использования Узлов формул**

Рассмотрим пример построения параболы с использованием Узла формул. На Рисунке 8 представлен графический пользовательский интерфейс передней панели виртуального прибора, достаточный для выполнения поставленной задачи.

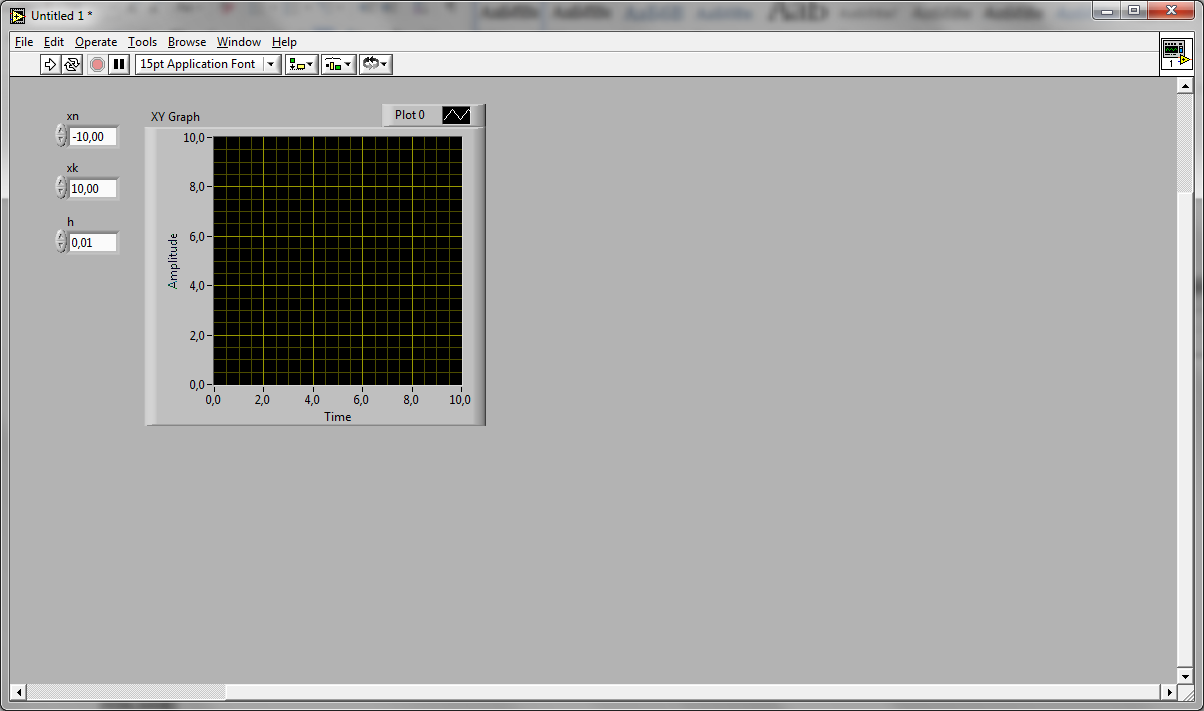


Рисунок 8 – Базовый набор элементов на передней панели виртуального прибора, достаточный для построения графика функциональной зависимости, в частности, ломаной линии

Перейдём на блок-диаграмму, которая в исходном состоянии должна выглядеть так же, как показано на Рисунке 9.

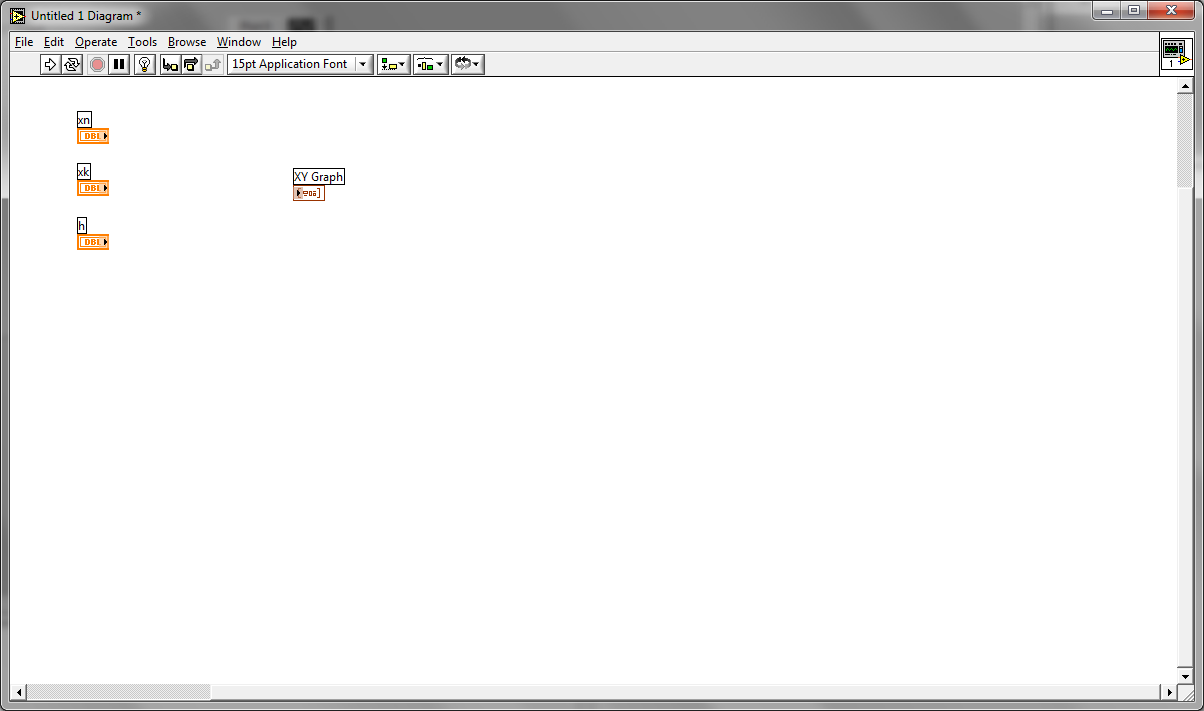


Рисунок 9 – Отображение базового набора элементов, достаточного для построения графика функциональной зависимости, на блок-диаграмме

Подготовим шаблон (Рисунок) для расчёта *N* точек в заданном диапазоне с некоторым шагом, но вместо графического кода, вычисляющего абсциссы и ординаты точек, разместим внутри циклической структуры структуру Узла формул, расположенную в том же разделе структур, где и все использованные в предыдущих задачах структуры.

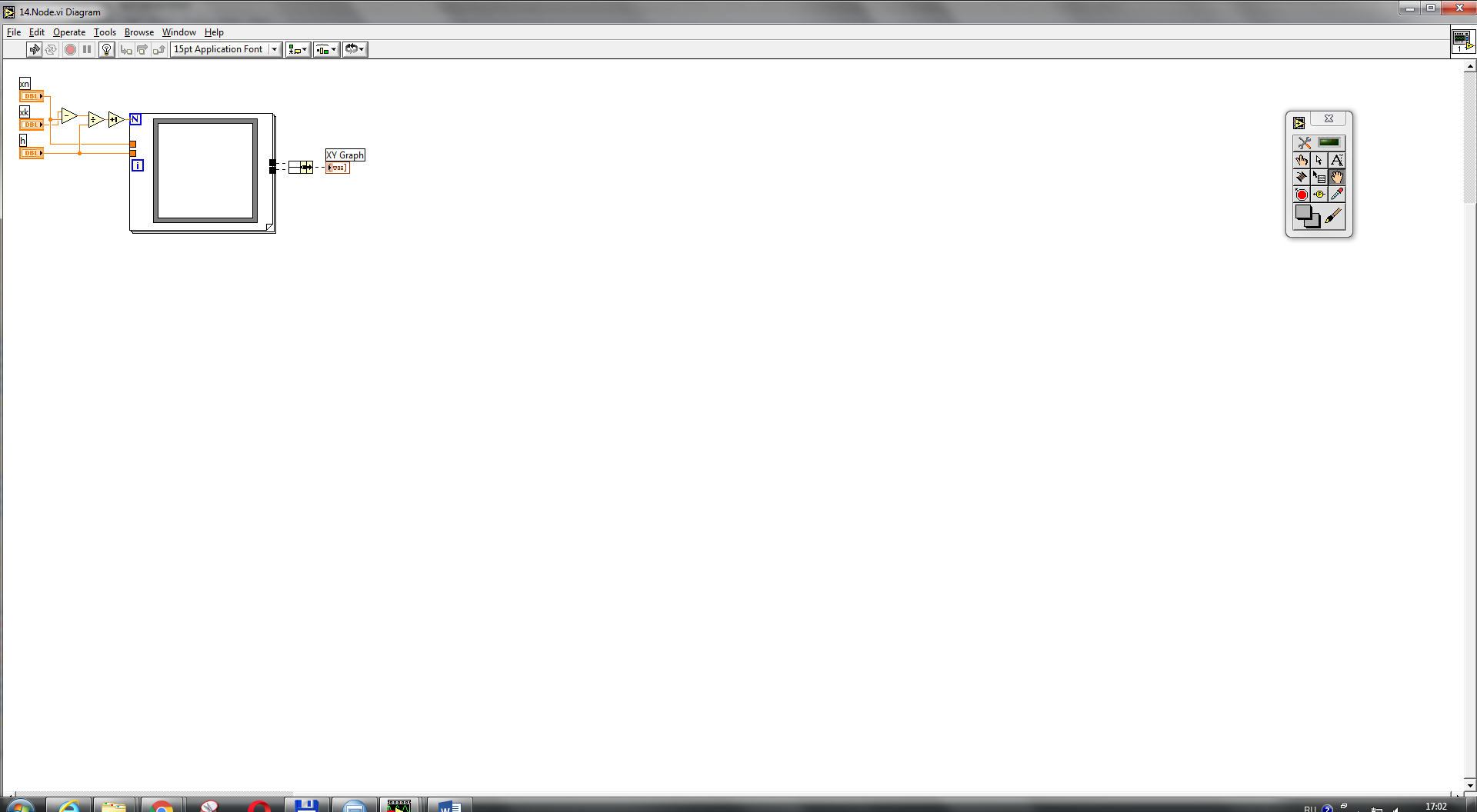


Рисунок 10 – Подготовка кода до состояния настройки Узла формул

Обратимся к контекстному меню структуры Узла формул для размещения на его левой границе входной клеммы (Рисунок 11).

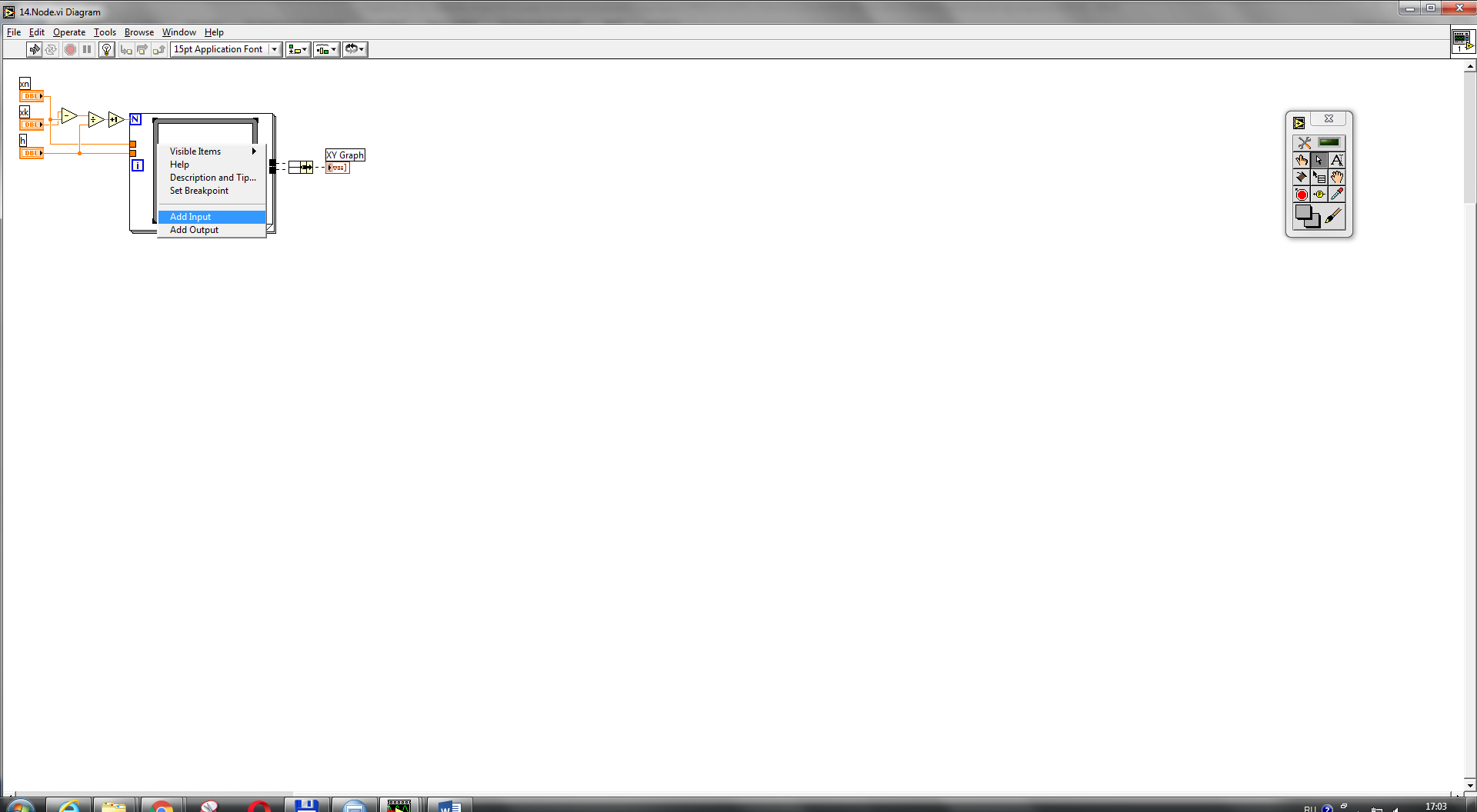


Рисунок 11 – Добавление входной клеммы через контекстное меню Узла формул

На Рисунке 12 показано, как выглядит появляющаяся рамка клеммы.

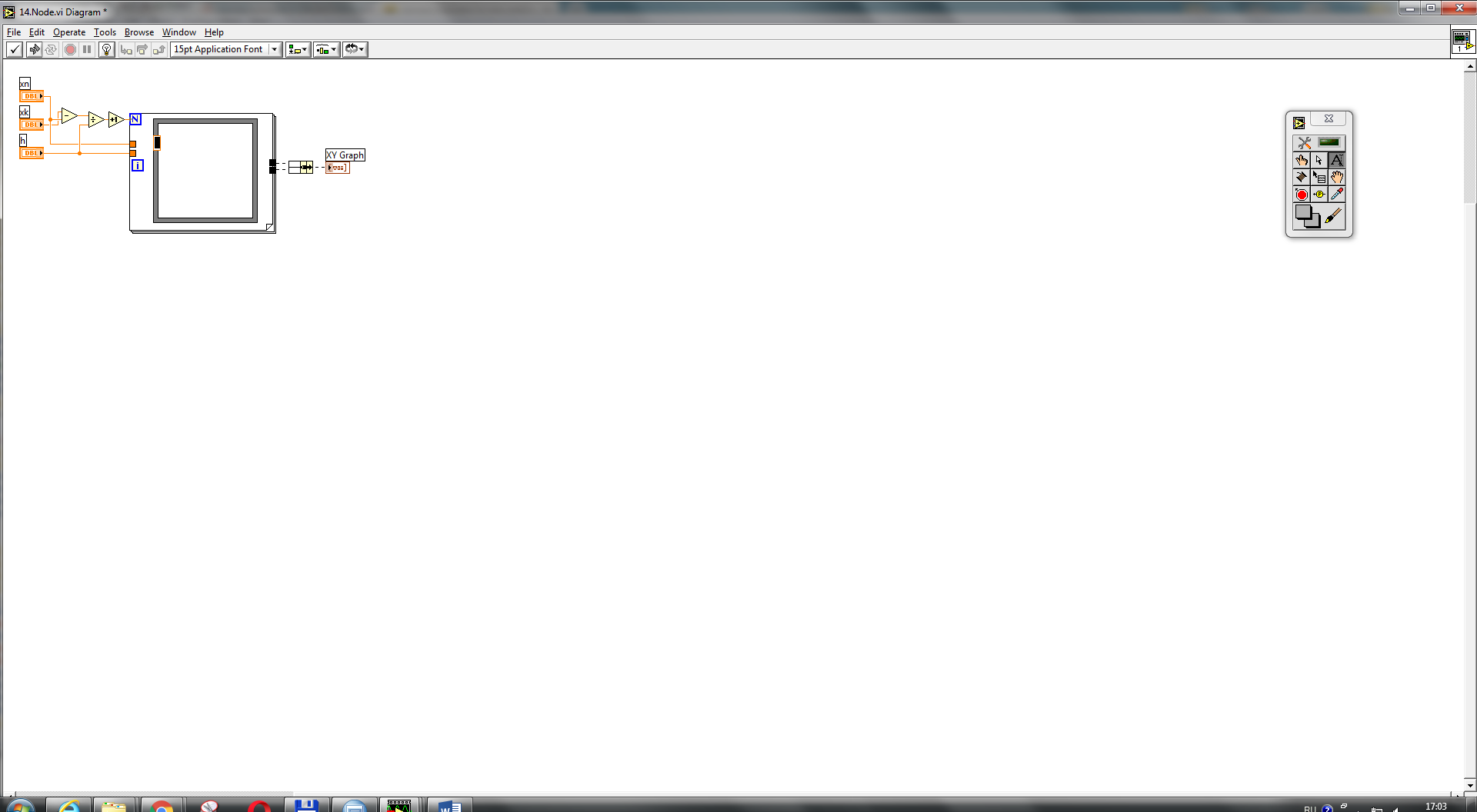


Рисунок 12 – Появление окна входной клеммы, ожидающего ввода имени входной переменной

Для созданной клеммы укажем имя *x*. Данная клемма предназначена для подключения к началу диапазона (*xn*). Имя, не соответствующее входной информации, поступающей с передней панели виртуального прибора, выбрано намеренно, для большей наглядности учебного материала (Рисунок 13). В выполняемых самостоятельных работах обучающимся настоятельно рекомендуется не игнорировать соответствие имён клемм именам входных данных, передаваемых пользователем через переднюю панель виртуального прибора, как это сделано в данном примере.

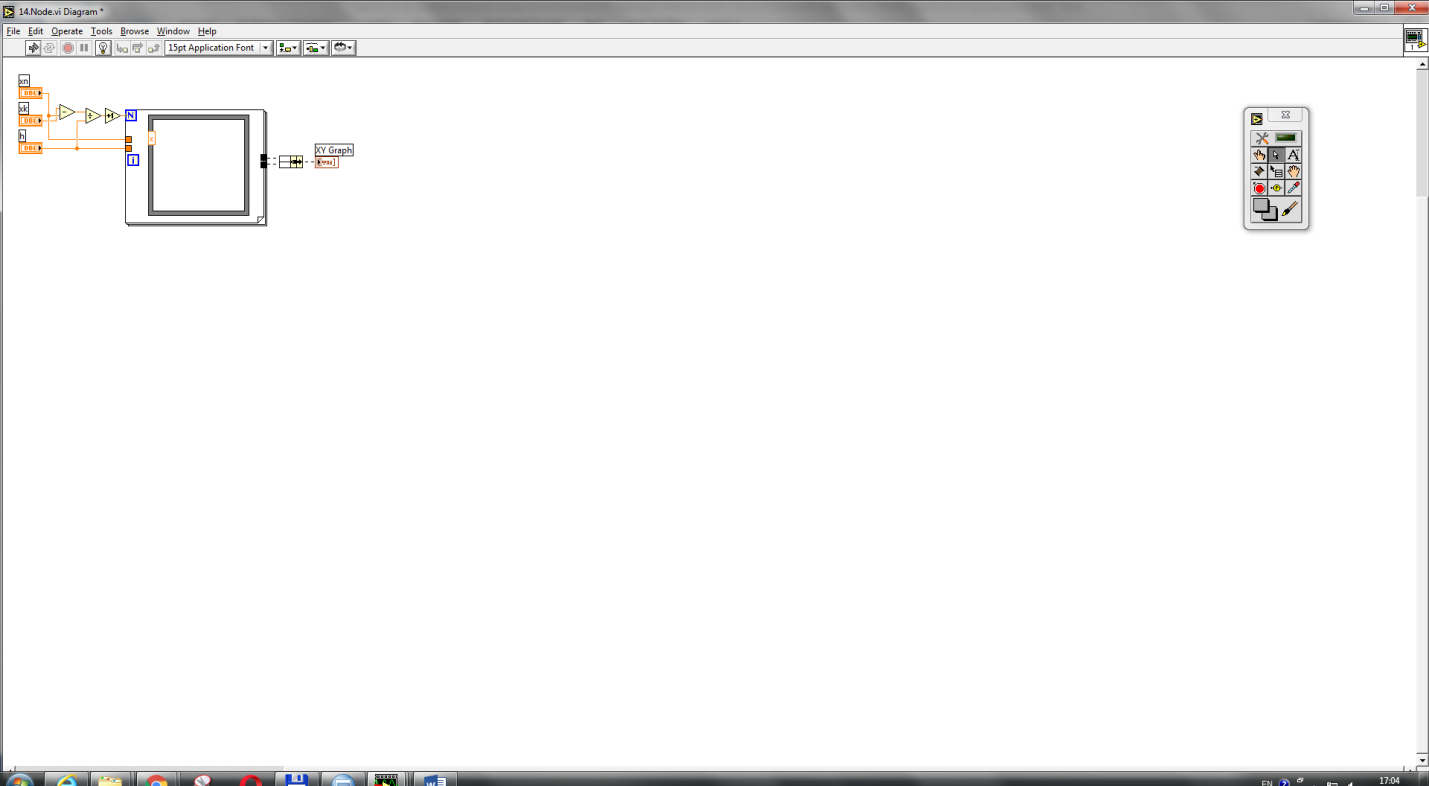


Рисунок 13 – Добавление входной переменной «*x*»

Дополним недостающие входные клеммы и выполним соединение их с графическими элементами блок-диаграммы. Введём код выражений для расчёта абсцисс и ординат параболы (Рисунок 14).

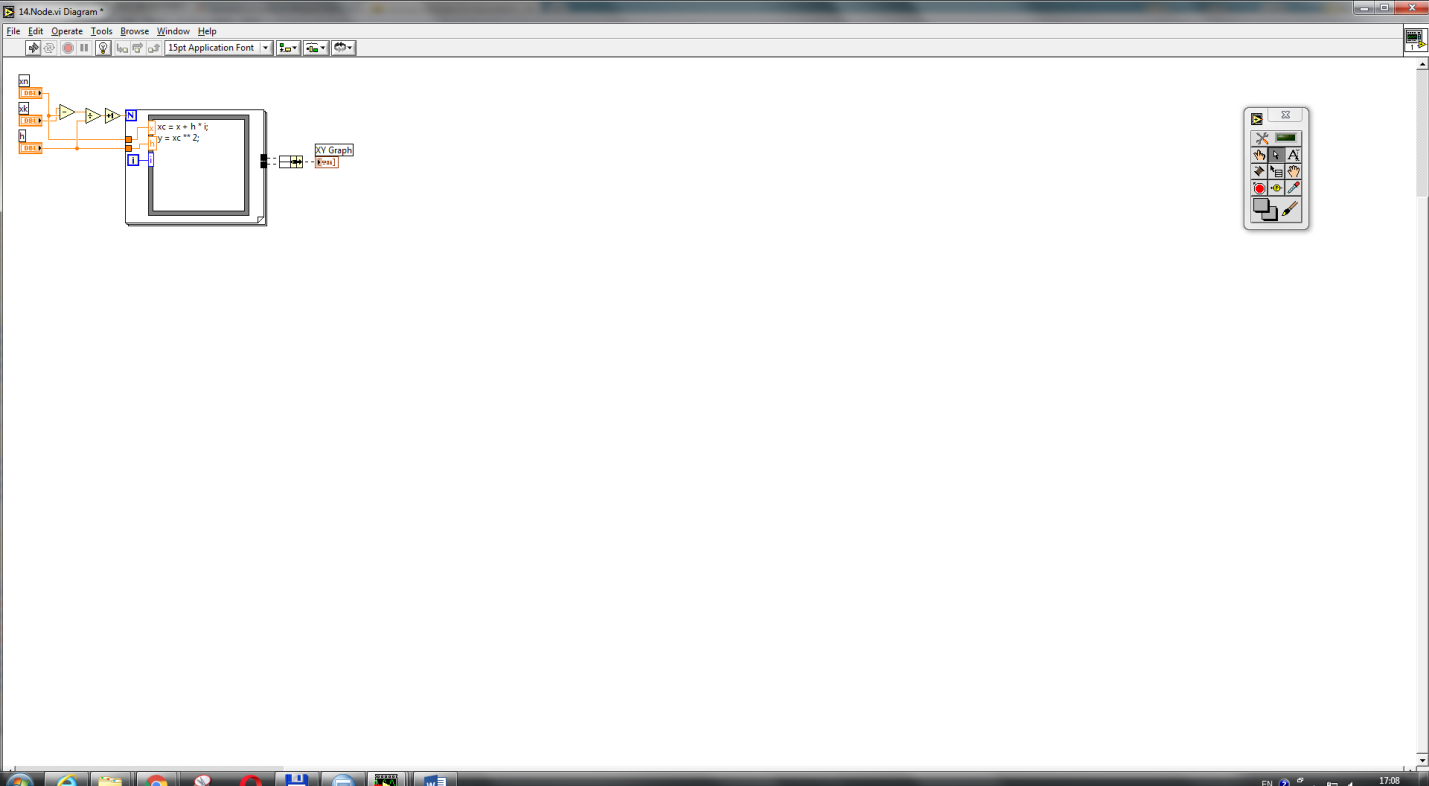


Рисунок 14 – Подключение переменных, приходящих извне, ко входным клеммам и ввод расчётных соотношений для вычисления абсциссы и ординаты точек параболы

Добавим выходные клеммы (Рисунок 15).

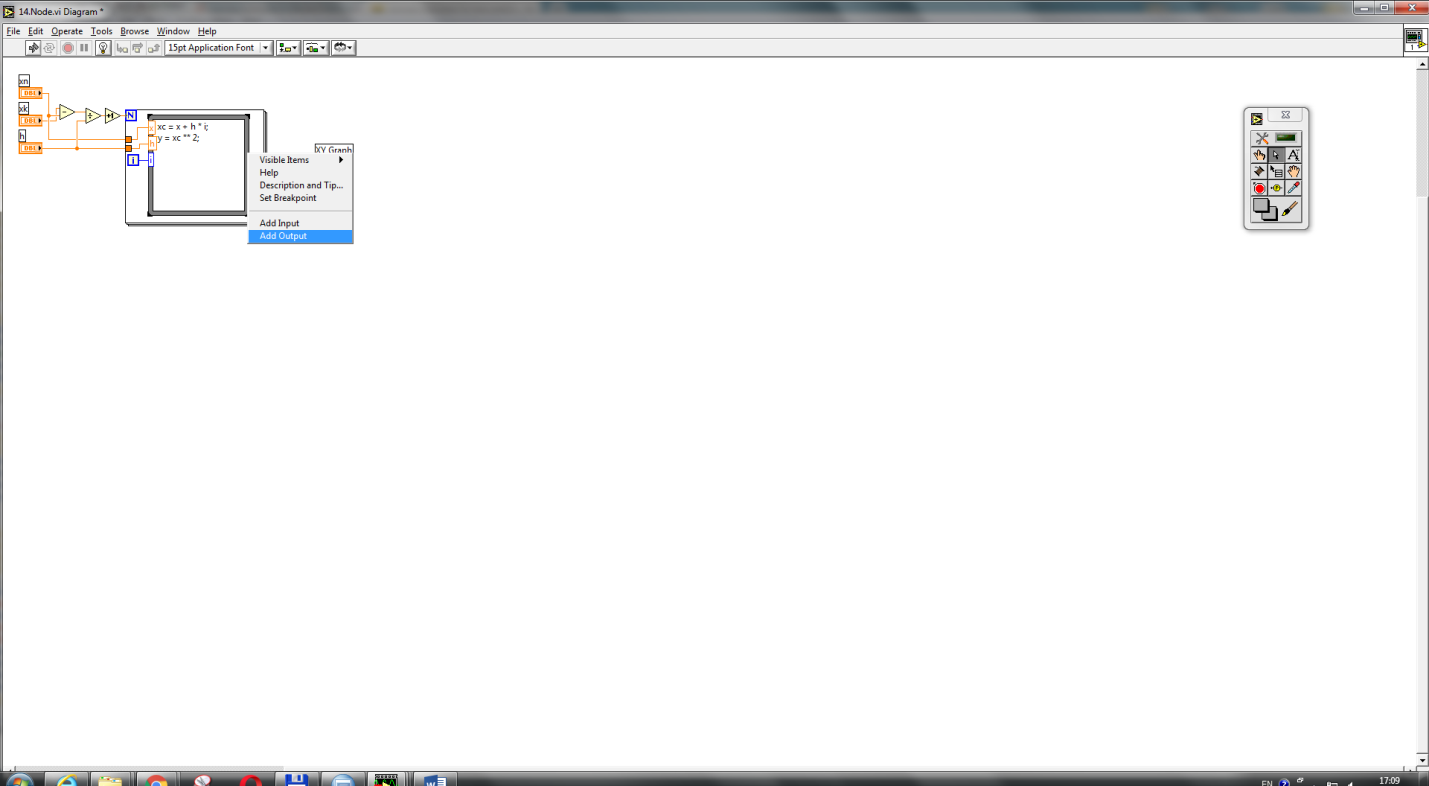


Рисунок 15 – Добавление выходной клеммы через контекстное меню Узла формул

На Рисунке 16 хорошо видно, что выходные клеммы более жирные по сравнению со входными. Выведем значения с выходны клемм на границу циклической конструкции, а затем на область для построения графиков функциональных зависимостей через «сцепку» в кластер типа «точка».

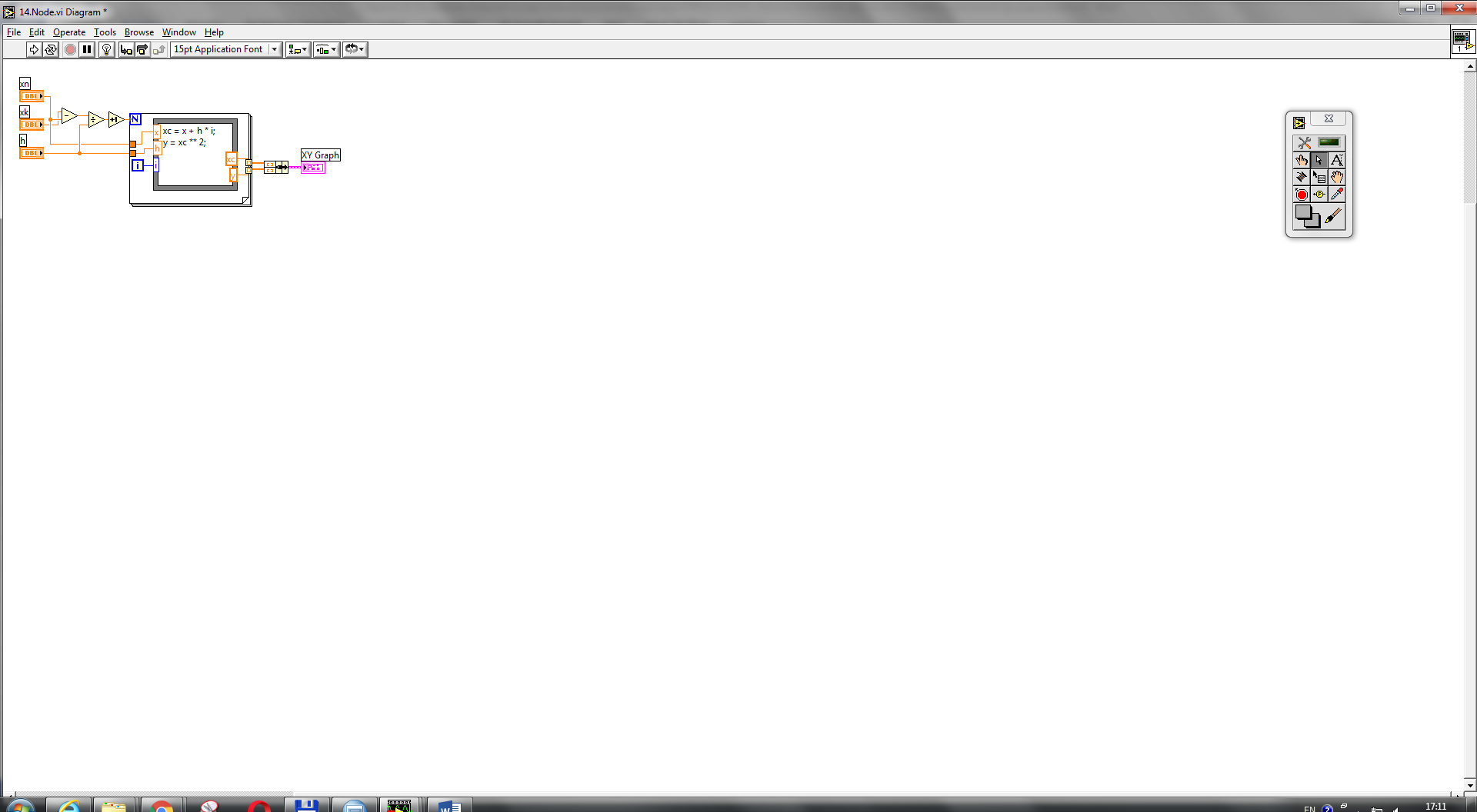


Рисунок 16 – Подключение переменных, исходящих из Узла формул во внешний код, к границам цикла и оттуда через «сцепку», объединяющую координаты в точку на область построения графиков функциональных зависимостей

Полученный результат, действительно, является параболой, как это и ожидалось (Рисунок 16).

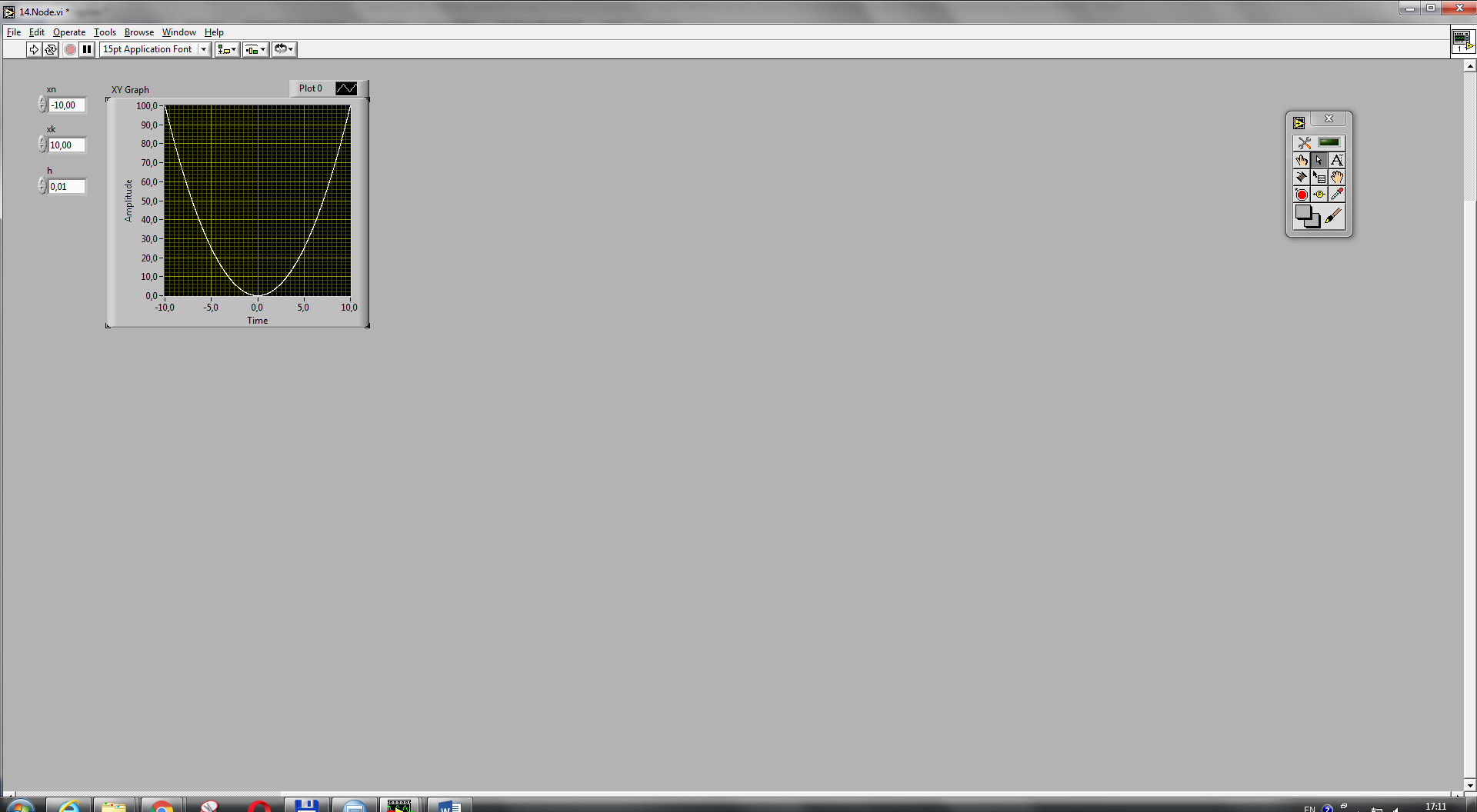


Рисунок 16 – Результат построения параболы с использованием Узла формул

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

**Варианты ломаных линий, предлагаемых для построения с использованием Узла формул:**

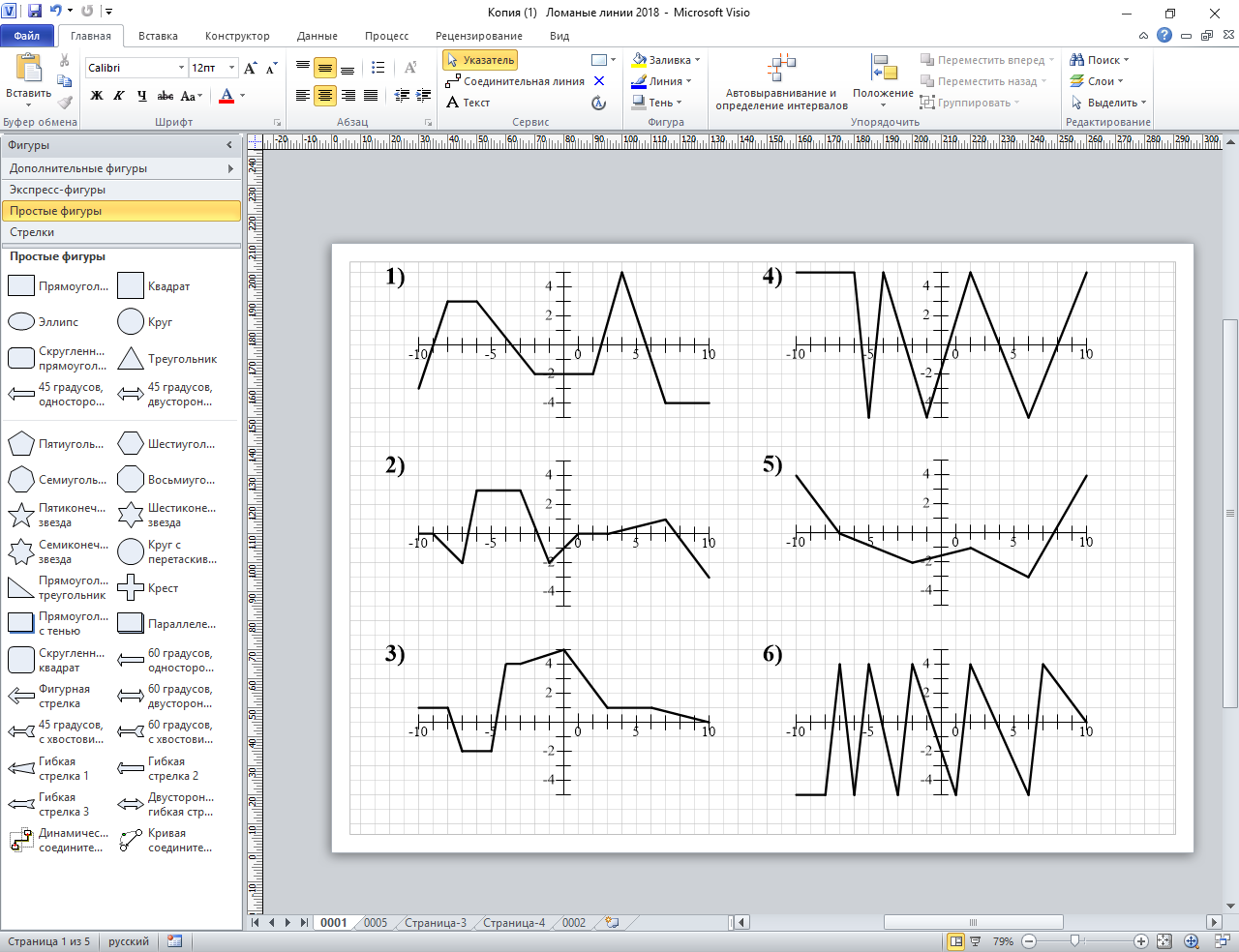


Рисунок 17 – Варианты ломаных линий (1-6)

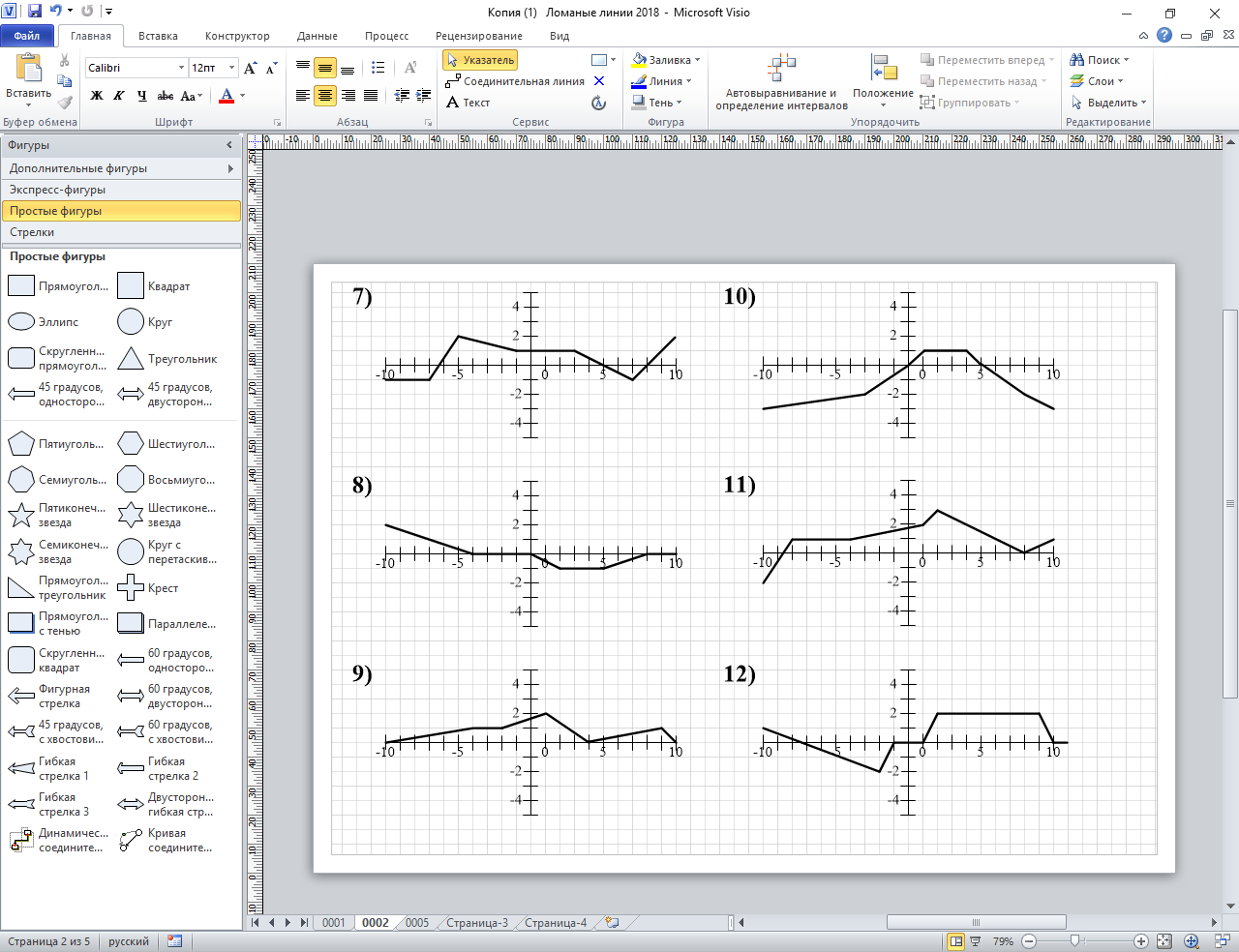


Рисунок 18 – Варианты ломаных линий (7-12)

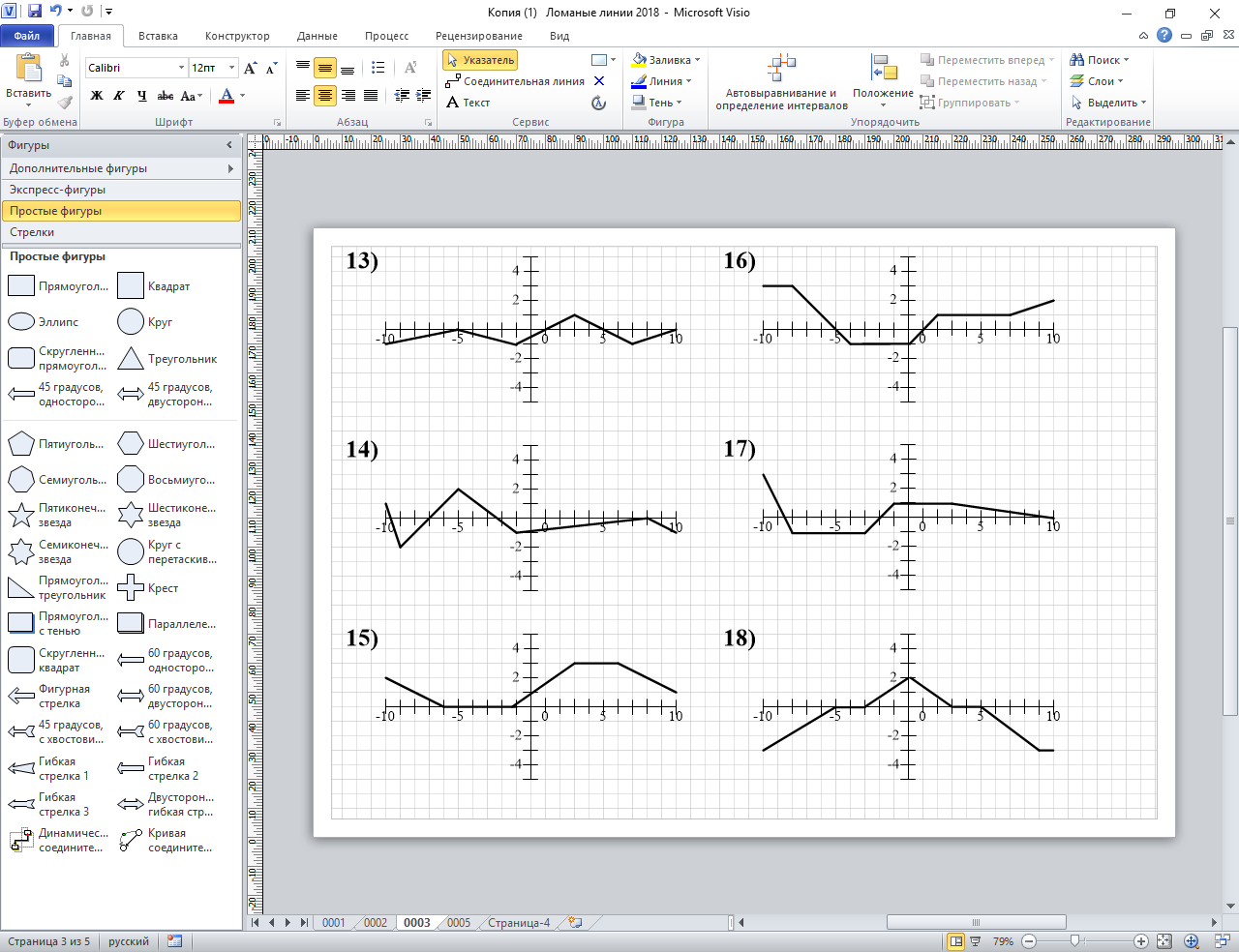


Рисунок 19 – Варианты ломаных линий (13-18)

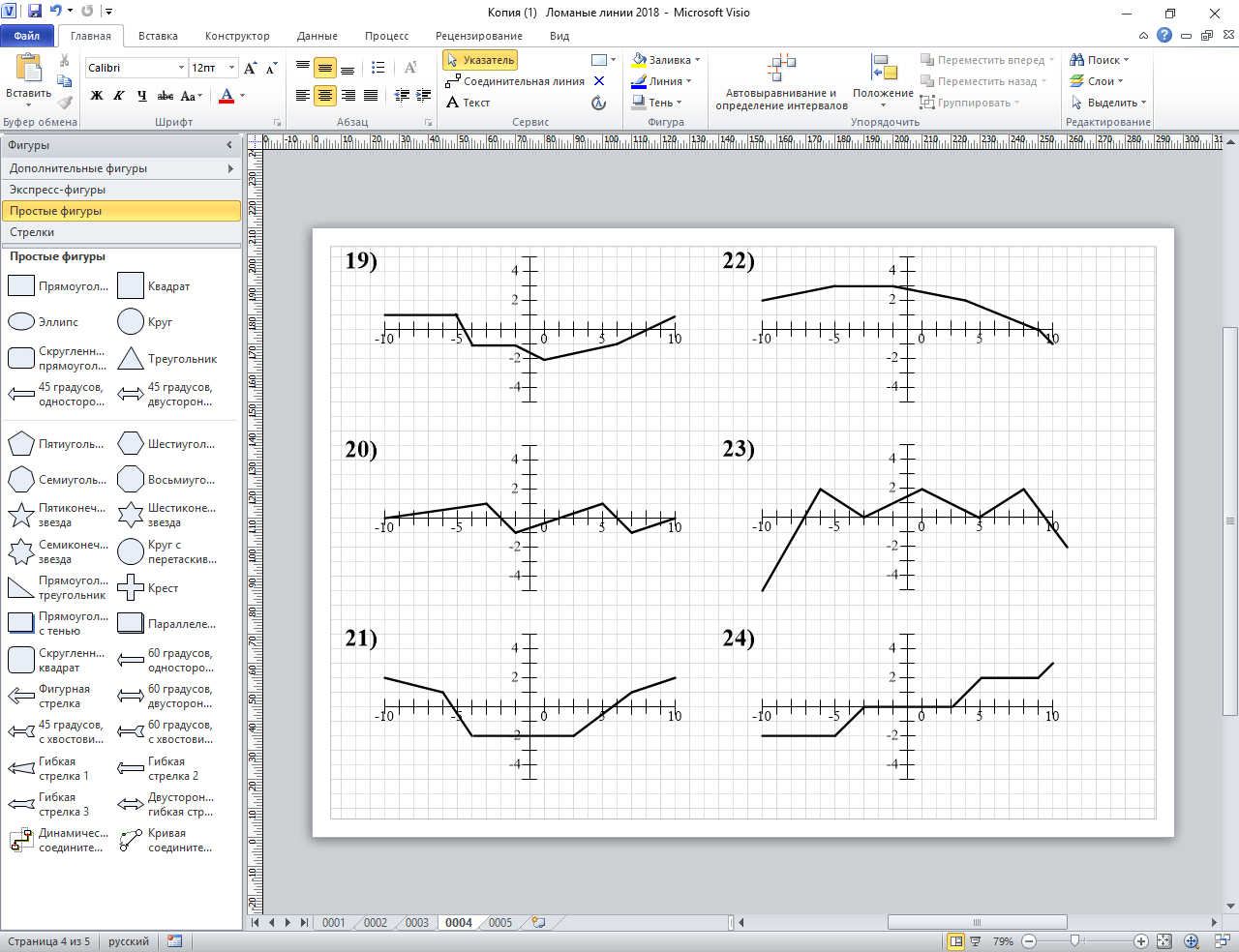


Рисунок 20 – Варианты ломаных линий (19-24)

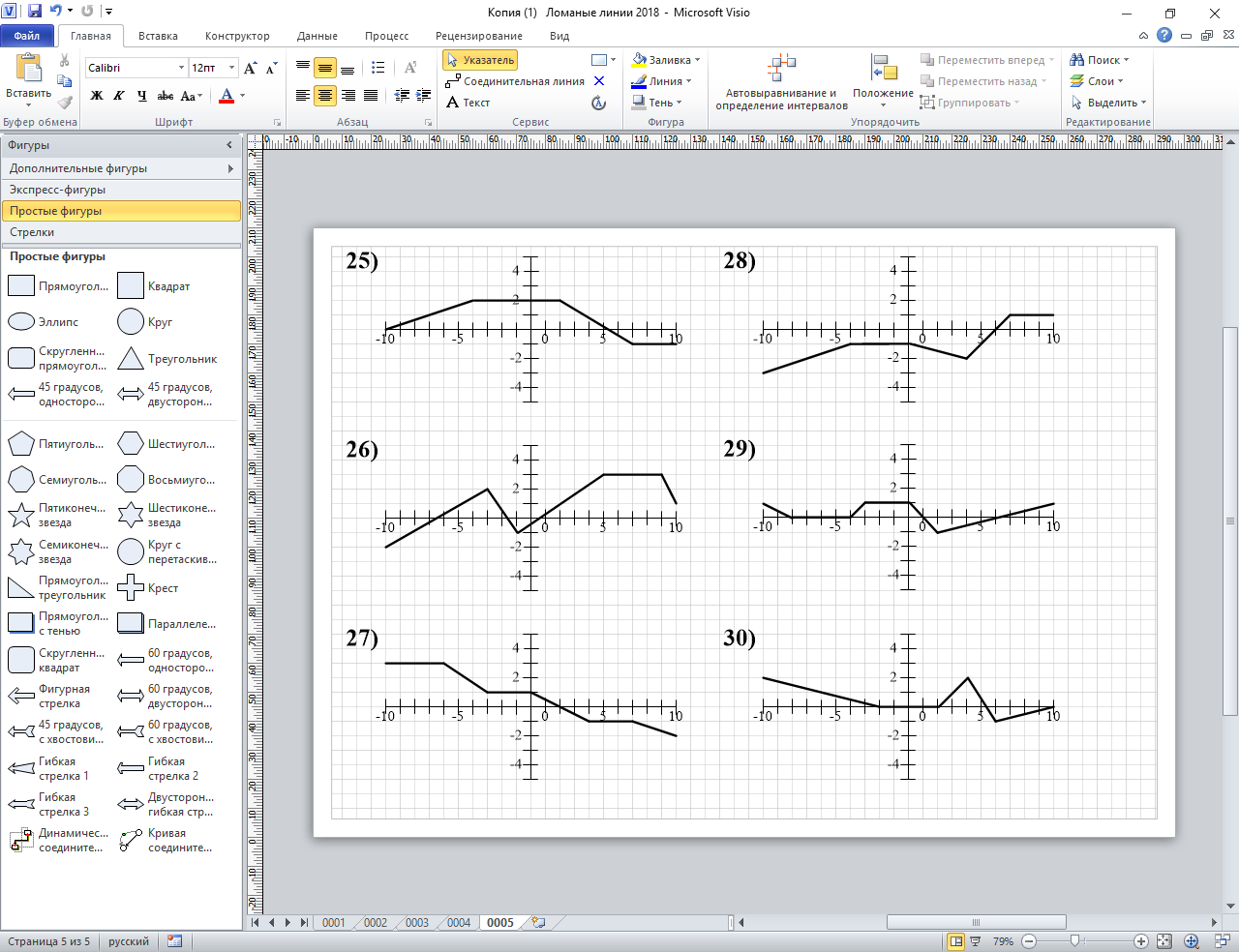


Рисунок 21 – Варианты ломаных линий (25-30)