

Введение

Программа состоит из 3 областей:

- Основное меню
- Инструментальная панель
- Рабочее поле

1) **Основное меню** состоит из основных команд для работы с документом в целом, такие как вставить, вырезать, открыть, сохранить..., а также содержит математический справочник и набор примеров. Сам справочник набран в Smath Studio и из него легко можно копировать необходимые формулы.

2) **Инструментальная панель разделена по категориям:**

а) Панель «**Арифметика**»

Содержит цифры, математические символы, и основные операции. Рассмотрим 3 из них.

- Оператор присвоения « $:=$ » служит для присвоения переменным каких-либо значений (численных либо символьных). Например $A := 2$ (переменной A присвоили значение 2, теперь программа знает, что есть переменная A , которая равна 2).
- Оператор численного вычисления « $=$ » служит для получения численного результата, он применим как к выражениям, так и к переменным. Например: $A = 2$ или $A+2 = 4$, т.к мы раньше присвоили переменной A значение 2.
- Оператор символьного вычисления « \rightarrow » позволяет вычислять символьный результат, например: $2B+2B \rightarrow$

б) Панель «**Матрицы**»

Содержит команды для работы с матрицами. Позволяет находить определитель матрицы, транспонировать её, находить минор. А также содержит команду векторного умножения (векторы программа рассматривает как матрицу с одним столбцом или строкой).

в) Панель «**Булева**»

Эта панель содержит набор команд для булевой алгебры. А также позволяет задавать логические операции в командах ветвления и циклах.

г) Панель «**Функции**»

Содержит набор часто используемых функций, таких как \sin , \cos , \log и т.п.

д) Панель «**График**»

Эта панель позволяет вращать, перемещать, увеличивать/уменьшать графики функций.

Двумерные графики строятся по переменной x , а трехмерные по 2 переменным x , y (переменные должны вводиться в нижнем регистре).

е) Панель «**Программирование**»

Содержит функции программирования, такие как: ветвление «IF», циклы «WHILE» и «FOR», вспомогательная функция «LINE».

ж) Последние две панели называются одинаково «**Символы**» и содержат греческие символы.

3) **Рабочее поле и первый расчет**

Рабочее поле занимает самую большую часть программы, здесь будем задавать исходные данные. Основным элементом поля является курсор или Фокус ввода (место, где будет набираться выражение), он выглядит как красный крестик.

Произведём нехитрые вычисления, для начала сложим 2 числа:

- Ткните мышкой в свободное место рабочего поля
- Введите «a», затем щёлкните оператор присвоения « $:=$ » на панели «Арифметика» и введите число. Должно получиться так «a:=5» после ввода цифры нажмите Enter, тем самым закончив ввод выражения и переведя курсор на следующую строку.
- Чуть ниже сделайте тоже самое для числа «b». Должно выйти так «b:=23»
- Ещё ниже вводите «a», затем оператор сложения « $+$ » с панели «Арифметика», затем «b» и оператор численного вычисления « $=$ » с той же панели. Что должно получиться см. на рис.1 ниже.

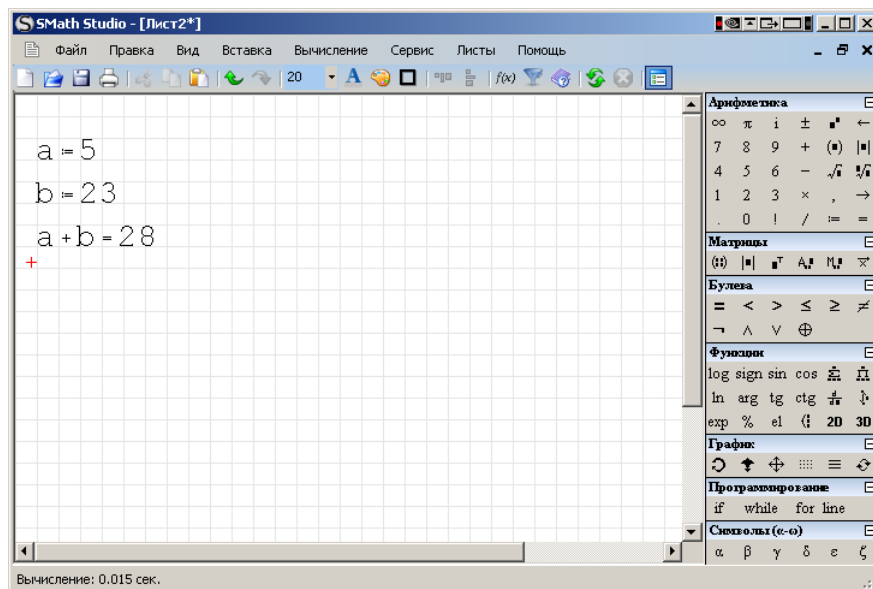


Рис. 1. Первый расчет

4) Об основных правилах написания расчетов в программе

Используемая в расчете переменная/функция должна быть заранее объявлена в программе, т.е. находиться выше либо левее от выражения. При объявлении переменной, можно использовать ранее объявленные переменные, встроенные и ранее объявленные функции, либо их сочетания. Например: $A := \sin(b)$ (где переменная b объявлена выше либо левее).

Для символьных вычислений переменные объявлять не обязательно.

Переменная может содержать в себе не только скалярное значение (численное), но и матрицу/вектор

5) Некоторые горячие клавиши

Сочетание клавиш	Результат
Shift + :	Вставить оператор присвоения «:=» с панели «Арифметика»
Ctrl + .	Вставить оператор символьного вычисления «→» с панели «Арифметика»
=	Вставить оператор численного вычисления «=» с панели «Арифметика»
Enter	Закончить ввод выражения и переместиться на строчку ниже

6) Единицы измерения и автоматическая помощь ввода

Щелкните на свободном поле и наберите «s». Рядом откроется небольшой список со всеми функциями/единицами измерения на букву «s» (*sin*, *sign* и т.п.), см. рис.2. Теперь если два раза кликнуть мышкой на элемент списка, то выбранная функция/единица измерения вставиться в документ (так же можно использовать клавишу **Tab** или сочетание **Ctrl+Enter**).

- В программе можно выбирать функции из списка (**Ctrl + E** или кнопка **fx** в основном меню программы).
- Можно выбрать единицы измерения (**Ctrl+W** или зайти в пункт выпадающего меню **Вставка -> Единица измерения**)

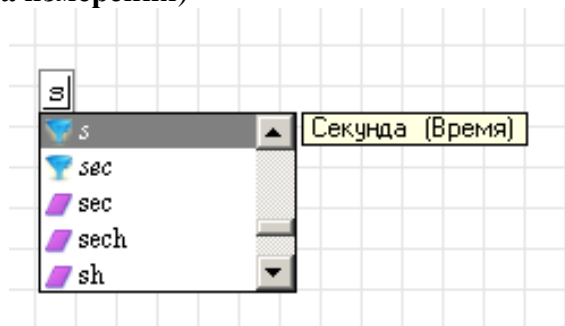


Рис.2. Список предлагаемых функций/единиц измерения

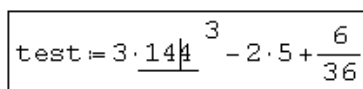
1. Основы работы

1. Курсор и ввод выражений

1.1 Курсор в Smath Studio

В Smath Studio курсор бывает трех типов:

- В виде красного крестика "+" указывает позицию.
- В виде уголка $_|_$ или символа $_|_$


$$\text{test} := 3 \cdot 144^3 - 2 \cdot 5 + \frac{6}{36}$$

Указывает в каком месте математического выражения будет проводиться редактирование (ввод нового символа или удаление неверно введенного символа). Если в данном примере ввести цифру 5, то число 144 изменится на 1454, если же нажать **Delete**, то получим 14.

Используя стрелки «влево» и «вправо» можно передвигать курсор по введенному выражению. Так же для этих целей можно использовать мышку или пробел (нажмите несколько раз пробел и вы увидите, что курсор будет по очереди выделять части выражения). Таким образом, можно расставлять забытые скобки, знаки корня и т.п.

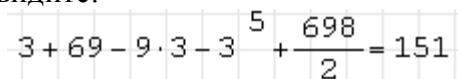
- В виде вертикальной черты |
Указывает позицию в текстовой области, в которой будет производиться редактирование.

1.2 Ввод выражений

Smath Studio можно использовать как простой калькулятор, для этого в любом свободном месте рабочего поля попробуйте ввести пример.

$$3+69-9*3-3^5+698/2=$$

После ввода этой комбинации вы увидите:

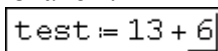

$$3 + 69 - 9 \cdot 3 - 3^5 + \frac{698}{2} = 151$$

- для ввода простых выражений можно пользоваться такими операторами, как "+", "-", "*", "/", "(" и т.п.
- комбинация **3^5** означает **3** в степени **5**.
- для того чтобы получить результат нажмите "=" на клавиатуре или на инструментальной панели.

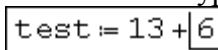
2. Особенности редактирования выражений

2.1 Коррекция знака

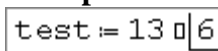
Иногда при вводе математического выражения случаются опечатки. Допустим, при вычислении значения какого-то выражения вы ошиблись знаком.


$$\text{test} := 13 + 6$$

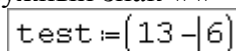
- 1) Для редактирования выражения поместите курсор после знака «+»


$$\text{test} := 13 + 6$$

- 2) Теперь при нажатии клавиши **Backspace** вместо «+» появится пустое поле ввода символа


$$\text{test} := 13 \quad 6$$

- 3) После этого можно вводить нужный знак «-»


$$\text{test} := (13 - 6)$$

Таким же образом можно редактировать любые знаки (:= > < + - и прочие).

2.2 Коррекция неверно введенных скобок

Это исправляется радикальным путем. Последовательность действий:

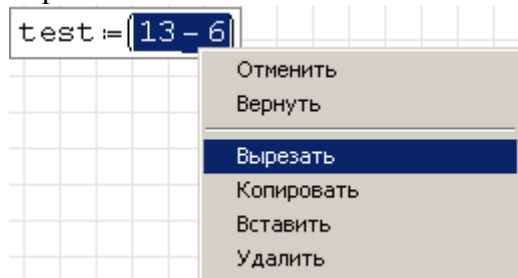
- 1) подвести курсор к открывающей скобке

test := (13 - 6)

- 2) нажать клавишу **Backspace**

test := 13 - 6

- 3) сочетанием **Ctrl+X** или же через выпадающее меню по правой кнопке мыши вырезать выделенную часть выражения



- 4) получатся пустые скобки

test := ()

- 5) нажимая клавишу **Backspace** удалить скобки

test :=

- 6) в пустое поле вставить ранее вырезанное выражение

test := 13 - 6

2.3 Форматирование текста

Введите текст:

Съешь еще этих мягких булочек, да выпей чаю

Обратите внимание: текст можно вводить и без вставки текстовой области. Просто начните набирать нужный текст, при нажатии клавиши «пробел» введенное выражение переведется в текстовую строку.

Важно: если введенный таким образом текст состоит из одного слова и не содержит пробелов, отформатировать его не удастся. Чтобы это исправить просто добавьте пробел в конце слова.

Съешь еще этих мягких булочек, да выпей чаю

Поместите курсор в любое место этой строки и нажмите сочетание клавиш **Ctrl+B**

Съешь еще этих мягких булочек, да выпей чаю
Съешь еще этих мягких булочек, да выпей чаю

Если еще раз нажать это сочетание, то текст вновь станет не жирным. Точно так же его можно сделать курсивным (комбинация **Ctrl+I**).

Съешь еще этих мягких булочек, да выпей чаю
Съешь еще этих мягких булочек, да выпей чаю

Или подчеркнутым (комбинация **Ctrl+U**)

Съешь еще этих мягких булочек, да выпей чаю
Съешь еще этих мягких булочек, да выпей чаю

Каждый из этих эффектов можно применять к тексту по отдельности, либо все сразу.

2. Переменные

Переменная - участок памяти компьютера, у которого есть **имя**, он может содержать в себе число, матрицу или текст.

Матрица - прямоугольная таблица с элементами, которая может содержать другие матрицы, числа, текст и **переменные**.

Текст - любые символы, заключенные в двойные кавычки.

В Smach Studio переменные могут содержать:

- *Текст*, например: "надпись, или любой другой текст"
- *Число*, например: 5 или 7,68 или -953,25

- *Матрицу*, например: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

При объявлении переменной, можно использовать: ранее объявленные переменные, встроенные и ранее объявленные функции либо их сочетания. Например: $A:=\sin(b)$ (где переменная b объявлена выше либо левее). Для символьных вычислений переменные объявлять не обязательно.

Пример1. Создадим простую переменную с именем **value**, для этого сделайте, как показано ниже:

```
value:= 123
value = 123
```

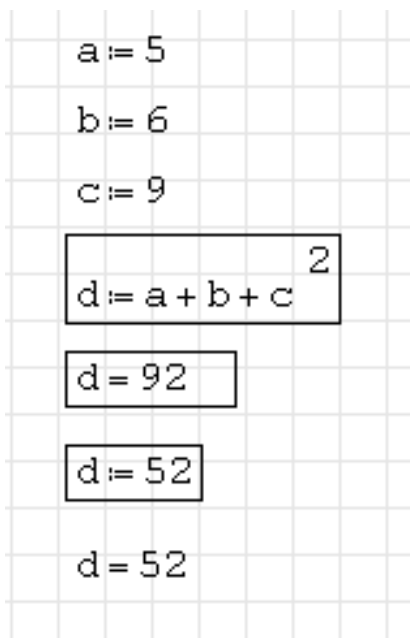
Здесь в первой строчке переменной **value** назначаем значение 123, об этом говорит оператор **присвоения** (**:=**) с панели **Арифметика**. А во второй строчке выводим содержимое переменной при помощи оператора **численного вычисления** (**=**) с той же панели.

Важно: Программа чувствительна к регистру букв, т.е переменные **Value**, **VALUE**, **value** и **VaLuE** воспринимаются ей как совершенно разные, и никак между собой не связанные.

Пример:
value := 123
VALUE := 235
value = 123
VALUE = 235

В этом примере, мы в первой строчке объявляем переменную **value**, а во второй — **VALUE**.

Пример2. Вычислим выражение с использованием переменных, для этого сделайте так, как показано ниже:



В этом примере присвоили переменным **a**, **b** и **c** численные значения, а затем присвоили переменной **d** результат вычисленной формулы. В третьей строчке снизу вывели значение переменной **d=92**. А во второй строчке снизу присвоили переменной новое значение - 52. Таким образом, переменная **d** выше четвертой строчки снизу не определена, между четвертой и второй строчками снизу равна 92, а правее и ниже второй строчки снизу равна 52.

- **Важно:** Переменные перед вычислением должны быть объявлены заранее, т.е выше либо левее вычисляемого выражения.
- **Важно:** Знак **присвоения** (**:=**) позволяет присвоить переменной другое значение. Переменная будет равна этому значению ниже операции присвоения.

```

b := 6
c := 9
d := a + b + c2
d = ■
d := 52
d = 52
a := 5

```

В этом примере показано, что программа не может посчитать выражение и результат присвоить переменной **d** поскольку переменная **a** определена ниже определения переменной **d**.

Пример3. Пример увеличения значения переменной:

```

A:=5
A:=A+2
A=7

```

Вначале присваиваем переменной **A** значение 5, а в следующей строке присваиваем переменной **A** её же значение + 2, и в результате получаем $5+2=7$

3. Булевы операции

Любая логическая операция может вернуть одно из двух значений, **ложь (false)** или **истина (true)**. В Smath Studio значение **ложь(false)** = 0, а значение **истина (true)** = 1.

В дальнейшем особенности работы Smath Studio могут измениться, поэтому будем пользоваться стандартными значениями: 0 - ложь, 1 - истина.

Обозначение	Название операции	Пример	
=	Равно	Оператор выдает истину , если оба выражения слева и справа от оператора равны, в противном случае он выдает ложь .	
		Выражение	Результат
		1 = 1	1
		0 = 1	0
<	Строго меньше	Оператор выдает истину, если число/выражение справа меньше числа/выражения слева.	
		Выражение	Результат
		1 < 1	0
		23 < 25	1
>	Строго больше	Оператор выдает истину, если число/выражение слева больше числа/выражения справа.	
		Выражение	Результат
		1 > 0	1
		23 > 25	0

\leq	Меньше либо равно	<p>Оператор выдает истину, если число/выражение слева меньше либо равно числу/выражению справа.</p> <table><tr><td>Выражение</td><td>Результат</td></tr><tr><td>$1 \leq 0$</td><td>0</td></tr><tr><td>$23 \leq 25$</td><td>1</td></tr></table>	Выражение	Результат	$1 \leq 0$	0	$23 \leq 25$	1				
Выражение	Результат											
$1 \leq 0$	0											
$23 \leq 25$	1											
\geq	Больше либо равно	<p>Оператор выдает истину, если число/выражение слева больше либо равно числу/выражению справа.</p> <table><tr><td>Выражение</td><td>Результат</td></tr><tr><td>$1 \geq 1$</td><td>1</td></tr><tr><td>$23 \geq 25$</td><td>0</td></tr></table>	Выражение	Результат	$1 \geq 1$	1	$23 \geq 25$	0				
Выражение	Результат											
$1 \geq 1$	1											
$23 \geq 25$	0											
\neq	Неравно	<p>Оператор выдает истину, если число/выражение справа, отличается от числа/ выражения слева.</p> <table><tr><td>Выражение</td><td>Результат</td></tr><tr><td>$1 \neq 1$</td><td>0</td></tr><tr><td>$23 \neq 25$</td><td>1</td></tr></table>	Выражение	Результат	$1 \neq 1$	0	$23 \neq 25$	1				
Выражение	Результат											
$1 \neq 1$	0											
$23 \neq 25$	1											
\neg	Отрицание, НЕ	<p>Этот оператор унарный, т.е. ему нужно только одно число/выражение, его приоритет выше приоритета всех остальных логических операций, т.е. он выполняется в первую очередь.</p> <table><tr><td>Выражение</td><td>Результат</td></tr><tr><td>$\neg 1$</td><td>0</td></tr><tr><td>$\neg 0$</td><td>1</td></tr></table>	Выражение	Результат	$\neg 1$	0	$\neg 0$	1				
Выражение	Результат											
$\neg 1$	0											
$\neg 0$	1											
\wedge	И	<p>Операция логическое И (логическое умножение). По аналогии с умножением в алгебре, если любое из 2-х чисел произведения равно нулю, то все произведение равно нулю.</p> <table><tr><td>Выражение</td><td>Результат</td></tr><tr><td>$1 \wedge 1$</td><td>1</td></tr><tr><td>$1 \wedge 0$</td><td>0</td></tr></table>	Выражение	Результат	$1 \wedge 1$	1	$1 \wedge 0$	0				
Выражение	Результат											
$1 \wedge 1$	1											
$1 \wedge 0$	0											
\vee	Или	<p>Операция логическое Или (логическое сложение). По аналогии со сложением в алгебре, если любое из 2-х чисел суммы не равно нулю, то вся сумма не равна нулю.</p> <table><tr><td>Выражение</td><td>Результат</td></tr><tr><td>$1 \vee 0$</td><td>1</td></tr><tr><td>$0 \vee 0$</td><td>0</td></tr></table>	Выражение	Результат	$1 \vee 0$	1	$0 \vee 0$	0				
Выражение	Результат											
$1 \vee 0$	1											
$0 \vee 0$	0											
\oplus	Исключающее Или	<p>Операция Исключающее Или.</p> <table><tr><td>Выражение</td><td>Результат</td></tr><tr><td>$1 \oplus 1$</td><td>0</td></tr><tr><td>$1 \oplus 0$</td><td>1</td></tr><tr><td>$0 \oplus 1$</td><td>1</td></tr><tr><td>$0 \oplus 0$</td><td>0</td></tr></table>	Выражение	Результат	$1 \oplus 1$	0	$1 \oplus 0$	1	$0 \oplus 1$	1	$0 \oplus 0$	0
Выражение	Результат											
$1 \oplus 1$	0											
$1 \oplus 0$	1											
$0 \oplus 1$	1											
$0 \oplus 0$	0											

Договоримся заранее, что логические выражения будем писать в скобках.

Пример1. Проверьте простые логические выражения:

$a := 5$	$c := 7$	
$b := 6$	$d := 7$	
$(a = b) = 0$	$(a \neq b) = 1$	
$(c = d) = 1$	$(c \neq d) = 0$	
$(a < b) = 1$	$(a \leq b) = 1$	
$(c < d) = 0$	$(c \leq d) = 1$	
$(a > b) = 0$	$(a \geq b) = 0$	
$(c > d) = 0$	$(c \geq d) = 1$	
$(a \wedge b) = 1$	$(a \vee b) = 1$	$(a \oplus b) = 0$
$(c \wedge d) = 1$	$(c \vee d) = 1$	$(c \oplus d) = 0$

Обратите внимание на результаты операции «Исключающее или», т.к. оба числа не равны 0, программа воспринимает их как истину.

Пример2. Проверьте более сложные логические выражения:

$a := 5$	$c := 7$
$b := 6$	$d := 0$
$(a > b) \wedge (c > d) = 0$	
$(a > b) \vee (c > d) = 1$	
$(a \wedge b) \wedge d = 0$	
$(a \wedge b) \wedge (\neg d) = 1$	
$(a \wedge b) \wedge c = 1$	
$(a \vee b) \vee d = 1$	
$(a \vee b) \vee c = 1$	

Все сложные выражения в примере в конечном итоге сводятся к простым:

Выражение	Примечание
$(a > b) \wedge (c > d)$	<ul style="list-style-type: none"> Подставляем значения переменных: $(5 > 6) \wedge (7 > 0)$ Т.к. скобки устанавливают приоритет, то сначала вычисляется первое выражение в скобках $(5 > 6) = 0$ (ложь) Затем второе $(7 > 0) = 1$ (истина) Теперь подставляем результаты в исходное выражение $(0 \wedge 1) = 0$ (ложь)
$(a \wedge b) \wedge d$	<ul style="list-style-type: none"> Подставляем значения переменных: $(5 \wedge 6) \wedge 0$ Т.к. скобки устанавливают приоритет вычисления, то сначала вычисляем выражение в скобках $(5 \wedge 6) = 1$ (истина) Подставляем результат в исходное выражение $1 \wedge 0 = 0$ (ложь)
$(a \vee b) \vee d$	<ul style="list-style-type: none"> Подставляем значения переменных: $(5 \vee 6) \vee 0$ Вычисляем первое выражение $(5 \vee 6) = 1$ Подставляем результат в исходное выражение $1 \vee 0 = 1$ (истина)

Пример3. Проверьте смешанные сложные выражения:

$$\begin{array}{l} x := 5 \\ y := 2 \\ x^2 + y \cdot 56 - 1036 \cdot ((x > 2) \wedge (x < 5)) = 137 \\ \boxed{(x > 2) \wedge (x < 5) = 0} \end{array}$$

$$\begin{array}{l} x := 3 \\ y := 2 \\ x^2 + y \cdot 56 - 1036 \cdot ((x > 2) \wedge (x < 5)) = -915 \\ \boxed{(x > 2) \wedge (x < 5) = 1} \end{array}$$

Обратите внимание, поскольку истинное выражение в Smath Studio численно равно 1, а ложное 0, логические выражения можно использовать вместе с алгебраическими и др. вычислениями. В этом примере видно, что если $2 < x < 5$ то логическое выражение равно 1, и число 1036 отнимается от остального выражения. А если x выходит за диапазон (2;5), то логическое выражение равно 0, и число 1036 помноженное на 0, тоже обращается в 0!

4. Матрицы, векторы и массивы

В Smath Studio нумерация всех элементов начинается с 1. Если вы будете использовать какую-либо функцию с нулевым элементом, то программа выдаст ошибку.


При описании нескольких аргументов в функции они разделяются символом «;» (на рисунках показаны примеры из предыдущей версии программы, где использовался символ «,»).

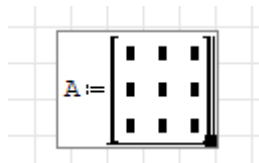
1. Матрицы, создание матриц, элементы матриц (**mat**, **matrix**, **el(2)**, **el(3)**)

Матрица в общем случае это прямоугольный набор элементов, который может содержать:

- числа (целые, вещественные, комплексные)
- строковые значения
- другие матрицы

1.1 Создать матрицу можно несколькими способами:

- а) При помощи команды **mat** или кнопки  «Кубическая матрица» с панели «Матрицы». В результате проделанных действий вы увидите пустую матрицу. Для того чтобы изменить её размеры, ухватитесь мышкой за черный квадратик справа снизу матрицы и растягивайте/сжимайте её.





- б) Командой **matrix(arg1; arg2)**, где **arg1** - число строк, **arg2** - число столбцов. Так **matrix(5;6)** создаст матрицу заполненную нулями, состоящую из 5 строк и 6 столбцов:

$$A := \text{matrix}(5, 6) \quad B := A + \text{random}(10)$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 5 & 1 & 3 & 7 & 6 \\ 2 & 4 & 3 & 4 & 3 & 4 \\ 4 & 9 & 1 & 2 & 5 & 1 \\ 0 & 2 & 2 & 2 & 5 & 5 \\ 1 & 1 & 3 & 1 & 8 & 0 \end{bmatrix}$$

1.2 В Smath Studio существуют две команды, которые позволяют обращаться к элементам матрицы по отдельности. Рассмотрим отличия этих команд в таблице:

el(2)	el(3)
Служит для работы с одномерными матрицами и позволяет получить любой элемент матрицы: $\begin{pmatrix} 11 \\ 21 \\ 31 \end{pmatrix}$	Служит для работы с 2-х мерными матрицами (прямоугольными матрицами), позволяет получить любой элемент матрицы: $\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix}$
После ввода команды появятся два связанных поля:  В верхнее поле записывается матрица, а в нижнее - номер строки. Пример: $\begin{pmatrix} 11 \\ 21 \\ 31 \end{pmatrix}_3 = 31$	После ввода команды появятся три связанных поля:  В верхнее поле записывается матрица, в нижние - номер строки, потом номер столбца. Пример: $\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix}_{2 \ 3} = 23$

2. Матрица-вектор

С точки зрения Smath Studio вектор — это матрица состоящая из одного столбца и трёх строчек (например, командой **matrix(3;1)** можно создать нулевой вектор).

Пример команды векторного умножения векторов  с панели «Матрицы»:

$$\begin{pmatrix} 11 \\ 21 \\ 31 \end{pmatrix} \times \begin{pmatrix} i \\ j \\ k \end{pmatrix} \rightarrow \begin{pmatrix} 21 \cdot k - 31 \cdot j \\ 31 \cdot i - 11 \cdot k \\ 11 \cdot j - 21 \cdot i \end{pmatrix}$$

С векторами можно выполнять те же действия, что и с матрицами.

3. Матрица как массив элементов (max, min, cols, rows)

В Smath Studio можно использовать матрицы и как массив для хранения элементов. Это возможно благодаря трём командам:

- **el(2)** и **el(3)**, которые позволяют обращаться к любому элементу матрицы;
- **matrix** позволяет создавать матрицы во время расчета.

В матрице можно, например, накапливать координаты точек для построения графиков или другие данные, с которыми удобно работать как с массивом.

Далее будут приведены некоторые команды, которые рассматривают матрицы как массив элементов:

Команда	Описание
max	Находит максимальное значение в матрице. $A := \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \quad \max(A) = 33$
min	Находит минимальное значение в матрице. $A := \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \quad \min(A) = 11$
cols	Возвращает количество столбцов в матрице. $\text{cols} \left(\begin{bmatrix} 11 \\ 21 \\ 31 \end{bmatrix} \right) = 1 \quad \text{cols} \left(\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \right) = 3$
rows	Возвращает количество строк в матрице. $\text{rows} \left(\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \right) = 3$

4. Сортировка матриц/массивов (sort, csort, rsort)

Сортировка позволяет упорядочить/отсортировать элементы матрицы/массива по порядку (по возрастанию)

Команда	Описание
sort	Команда применима только к матрицам, состоящим из одного столбца и нескольких строчек, она сортирует элементы в порядке возрастания. Пример: $\text{sort} \left(\begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix}$
csort	Команда позволяет отсортировать всю матрицу, по элементам выбранного столбца. Т.е. матрица сортируется по строчкам, согласно возрастанию элементов указанного столбца. Пример (сортируем матрицу по 1 столбцу): $\text{csort} \left(\begin{bmatrix} 3 & 0 & -5 \\ 1 & 3 & 0 \\ 0 & 2 & 1 \end{bmatrix}, 1 \right) = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 3 & 0 \\ 3 & 0 & -5 \end{bmatrix}$
rsort	Команда позволяет отсортировать всю матрицу, по элементам выбранной строки. Т.е. матрица сортируется по столбцам, согласно возрастанию элементов указанной строки. Пример (сортируем матрицу по 1 строчке): $\text{rsort} \left(\begin{bmatrix} 3 & 0 & -5 \\ 1 & 3 & 0 \\ 0 & 2 & 1 \end{bmatrix}, 1 \right) = \begin{bmatrix} -5 & 0 & 3 \\ 0 & 3 & 1 \\ 1 & 2 & 0 \end{bmatrix}$

5. Объединение матриц (augment, stack)

Команда	Описание
augment	<p>Команда позволяет объединять матрицы по столбцам.</p> $\text{augment} \left(\begin{pmatrix} 3 \\ -2 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 & 0 & -5 \\ 1 & 3 & 0 \\ 0 & 2 & 1 \end{pmatrix} \right) = \begin{pmatrix} 3 & 3 & 0 & -5 \\ -2 & 1 & 3 & 0 \\ 1 & 0 & 2 & 1 \end{pmatrix}$ <ul style="list-style-type: none"> Можно объединить матрицы только с одинаковым числом строк! Команда позволяет объединить несколько матриц, все они отделяются между собой запятыми. Возможно использовать такую конструкцию: A:=augment(A; B), где матрица A должна быть определена перед первым использованием этой конструкции. Она позволяет прибавлять к матрице A, её же значение плюс матрицу B. Этот приём используется в основном в циклических конструкциях.
stack	<p>Команда позволяет объединять матрицы по строкам</p> $\text{stack} \left(\begin{pmatrix} 3 & -2 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 0 & -5 \\ 1 & 3 & 0 \\ 0 & 2 & 1 \end{pmatrix} \right) = \begin{pmatrix} 3 & -2 & 1 \\ 3 & 0 & -5 \\ 1 & 3 & 0 \\ 0 & 2 & 1 \end{pmatrix}$ <ul style="list-style-type: none"> Можно объединить матрицы только с одинаковым числом столбцов! Команда позволяет объединить несколько матриц, все они отделяются между собой запятыми. Возможно использовать такую конструкцию: A:=stack(A; B), где матрица A должна быть определена перед первым использованием этой конструкции. Она позволяет прибавлять к матрице A, её же значение плюс матрицу B. Этот приём используется в основном в циклических конструкциях.

6. Субматрицы (col, row, submatrix)

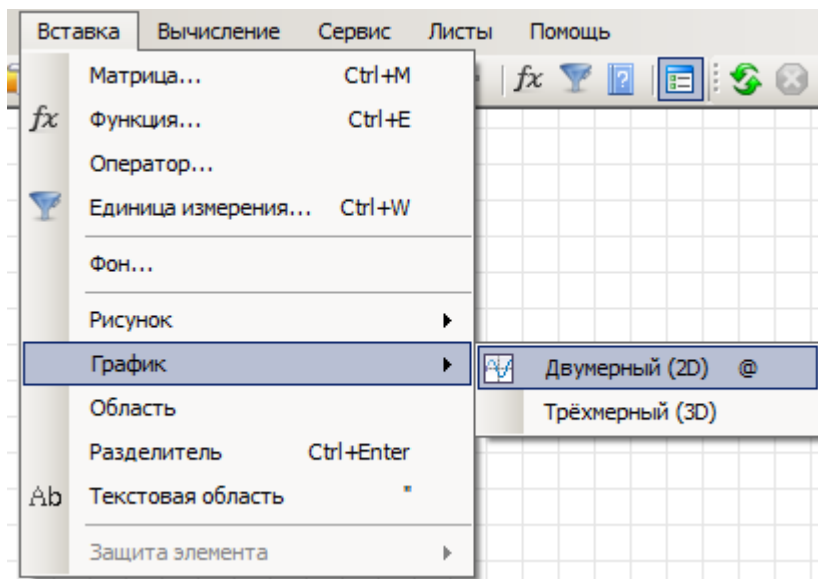
Команда	Описание
col	<p>Команда позволяет извлечь из матрицы любой указанный столбец:</p> $\text{col} \left(\begin{pmatrix} 3 & 0 & -5 & -2 \\ 1 & 3 & 0 & 0 \\ 0 & 2 & 1 & 5 \\ 3 & 4 & 3 & 68 \end{pmatrix}, 3 \right) = \begin{pmatrix} -5 \\ 0 \\ 1 \\ 3 \end{pmatrix}$ <p>Номер столбца должен быть меньше либо равен количеству столбцов в матрице.</p>
row	<p>Команда позволяет извлечь из матрицы любую указанную строчку:</p> $\text{row} \left(\begin{pmatrix} 3 & 0 & -5 & -2 \\ 1 & 3 & 0 & 0 \\ 0 & 2 & 1 & 5 \\ 3 & 4 & 3 & 68 \end{pmatrix}, 2 \right) = \begin{pmatrix} 1 & 3 & 0 & 0 \end{pmatrix}$ <p>Номер требуемой строки должен быть меньше либо равен количеству строк в матрице.</p>
submatrix	<p>Команда позволяет извлечь из матрицы указанную прямоугольную область:</p> $\text{submatrix} \left(\begin{pmatrix} 3 & 0 & -5 & -2 \\ 1 & 3 & 0 & 0 \\ 0 & 2 & 1 & 5 \\ 3 & 4 & 3 & 68 \end{pmatrix}, 1, 4, 2, 3 \right) = \begin{pmatrix} 0 & -5 \\ 3 & 0 \\ 2 & 1 \\ 4 & 3 \end{pmatrix}$ <p>Мы вычленили из основной матрицы подматрицу, которая включает в себя строки с 1 по 4 и колонки со 2 по 3 из основной матрицы!</p>

5. Графики

Графики функции в Smath Studio бывают 2-х видов: двухмерные и трехмерные. Для работы с ними есть специальная панель «График».

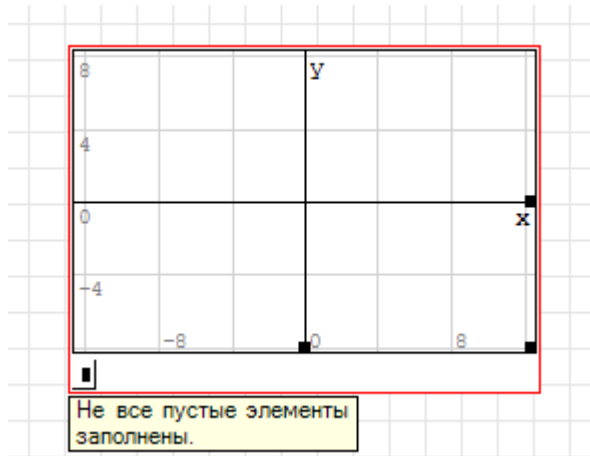
1) Вставить график в расчет можно несколькими способами:

- 2D-график можно вставить комбинацией Shift + @.
- При помощи меню Вставка.



2) Работа с графиком

Для начала редактирования графика, необходимо его выделить и поместить курсор в поле ввода данных.



В нижней части графика есть пустое поле, куда нужно вставить:

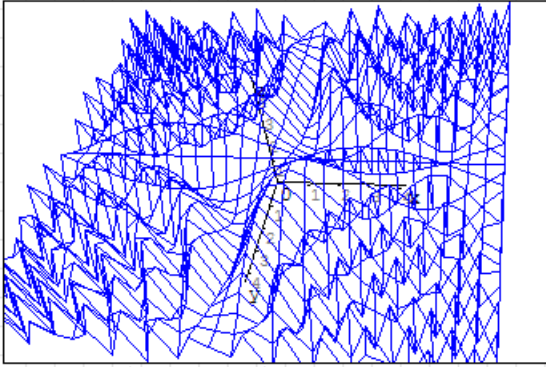
- для 2D графиков — функцию от x (например, $\sin(x)$ или $\cos(x)$)
- для 3D графиков — функцию от x и y (например, $\sin(x+y)$), но можно и просто $\sin(x)$ или $\cos(x)$.

В объявлении функции можно использовать любые имена переменных, но график строиться только относительно « x » и « y ». Т.е. для графика функция должна быть вызвана с использованием этих переменных.

Описываем пользовательскую функцию

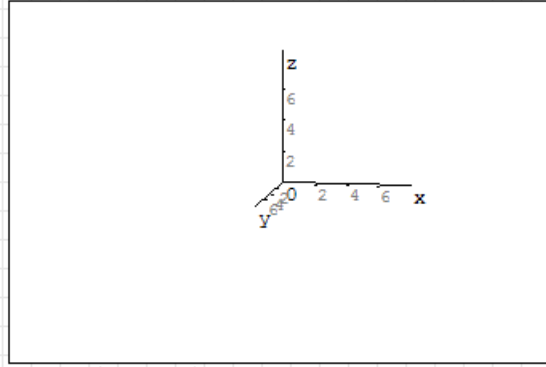
```
testFn(a, b, c) := sin(b · c) + a
```

И Вставляем ее в 3d график




```
testFn(1, x, y)
```

Это правильно!

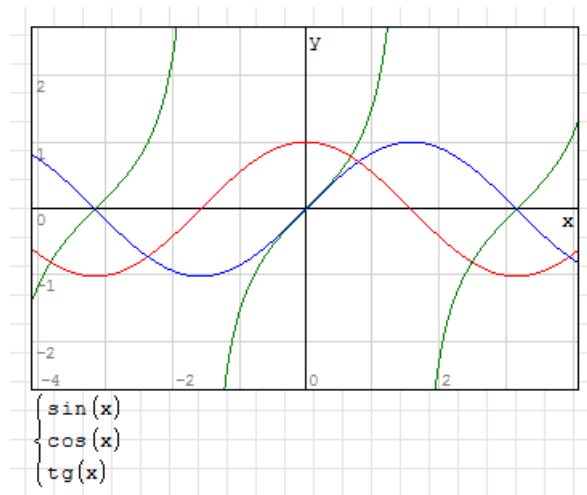


```
testFn(1, b, c)
```

Это не правильно!

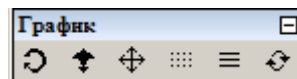
Если нужно построить графики сразу нескольких функций, то можно воспользоваться функцией **sys** (алгебраическая система), введя ее с клавиатуры или нажав кнопку  с панели «Функции».

Например:



При этом разные графики рисуются линиями разного цвета.

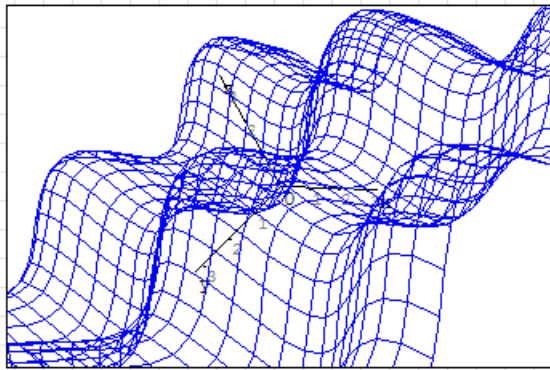
3) Панель «График»



Панель содержит 6 кнопок для работы с графиками:

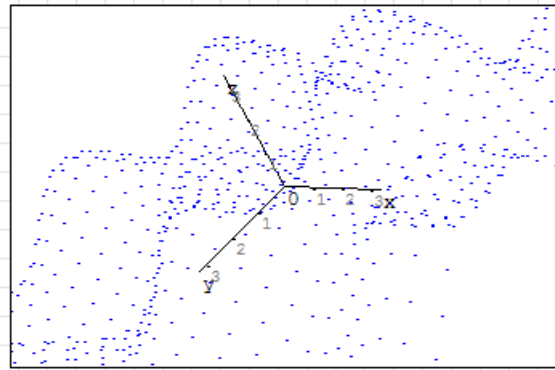
- Вращать — позволяет вращать мышкой 3D-график.
- Масштабировать — позволяет масштабировать график (можно колесиком мышки).
- Перемещать — позволяет перемещать изображение внутри графика.
- График точками и График линиями — переключает режим отображения графиков точками/линиями.
- Обновить — позволяет вернуть исходное положение графика.

График линиями



$$\cos(x) + \sin(y)$$

График точками



$$\cos(x) + \sin(y)$$

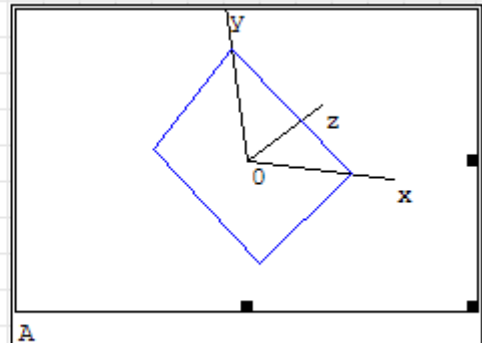
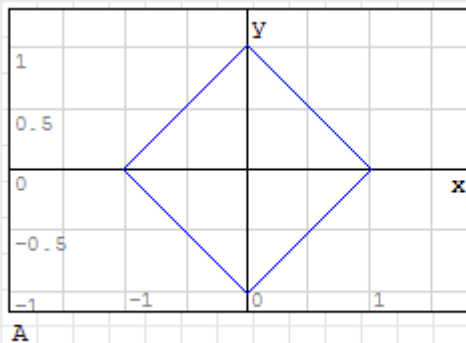
4) Построение графика по точкам

Для построения графика по точкам нужно использовать **матрицы**. В таблице показан формат матриц для 2D и 3D графиков:

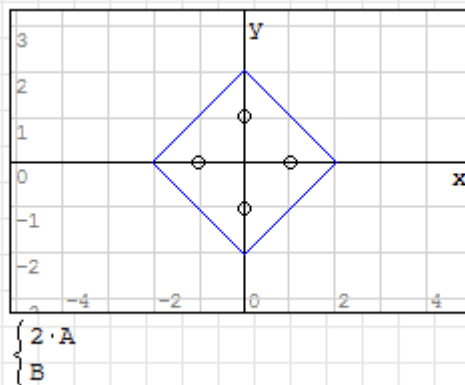
2D	3D
$\begin{bmatrix} x1 & y1 \\ x2 & y2 \\ \vdots & \vdots \\ xn & yn \end{bmatrix}$ <p>Эта матрица содержит пары координат X и Y. Матрицу этого формата можно вывести и на 2D и на 3D-графике (на 3D графике изображение построится в плоскости XY).</p>	$\begin{bmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ \vdots & \vdots & \vdots \\ xn & yn & zn \end{bmatrix}$ <p>Эта матрица содержит тройки координат X, Y, Z. Матрица такого формата способна отображаться лишь на 3D-графике.</p>
Такой подход к построению графиков как правило требует программирования. Для работы с матрицами могут понадобиться функции augment , el(2) , el(3) , stack .	

Пример1. Постройте 2D-графики по точкам:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{bmatrix}$$

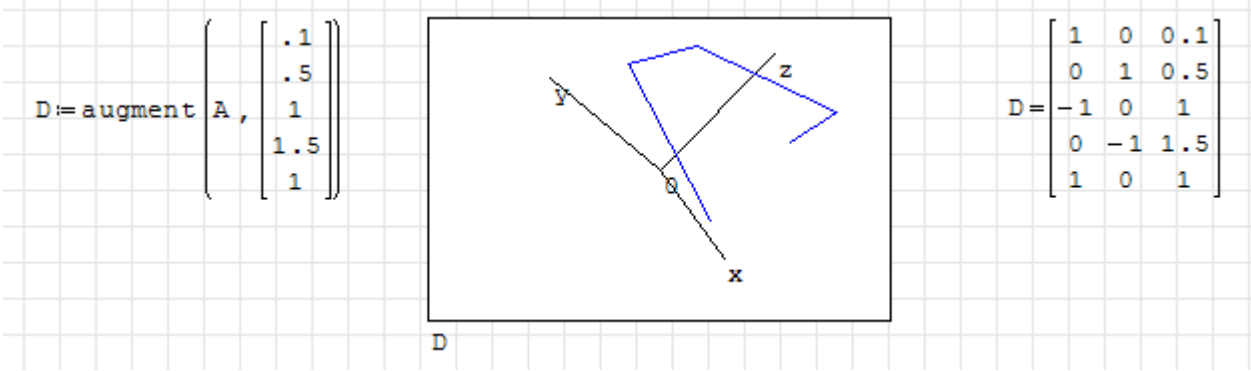


$$B = \text{augment} \left(A, \begin{bmatrix} "o" \\ "o" \\ "o" \\ "o" \\ "o" \end{bmatrix} \right)$$



$$B = \begin{bmatrix} 1 & 0 & "o" \\ 0 & 1 & "o" \\ -1 & 0 & "o" \\ 0 & -1 & "o" \\ 1 & 0 & "o" \end{bmatrix}$$

Пример2. Постройте 3D-график по точкам:



5) Вывод надписей на графике

Программа позволяет выводить на графике, как надписи, так и разного вида «метки».

Формат матрицы для вывода надписей:

$$\begin{bmatrix} x1 & y2 & "text" & txtSize & "txtColor" \\ x2 & y2 & "text" & txtSize & "txtColor" \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ xn & yn & "text" & txtSize & "txtColor" \end{bmatrix}$$

где:

Параметр	Описание
X1...Xn	Координата X точки
Y1...Yn	Координата Y точки
"text"	Сама надпись в кавычках
txtSize	Размер шрифта
"txtColor"	Цвет текста (пишется в кавычках)

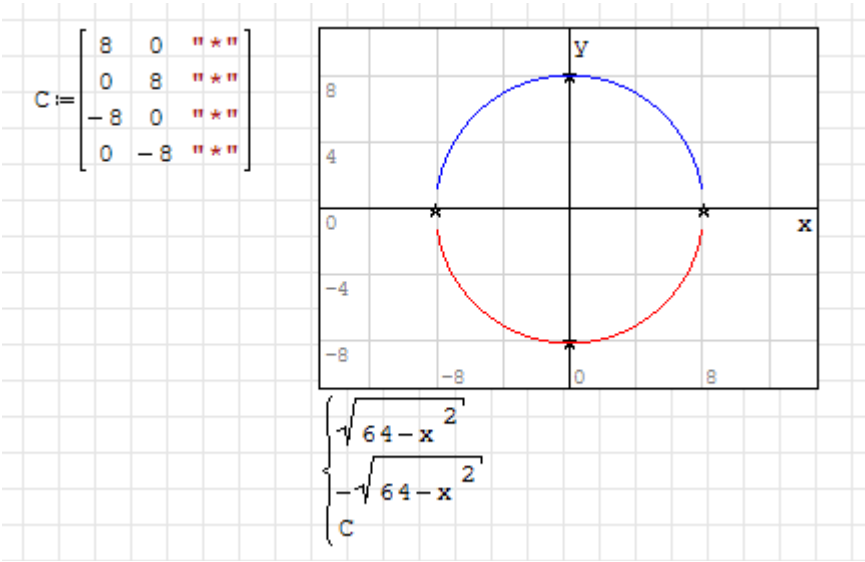
Обратите внимание: Последние два столбца матрицы являются не обязательными, их можно не использовать, при этом программа сама назначит цвет и размер шрифта.

Параметр *"text"* может содержать как текст, так и специальные символы. *Обычный текст* отображается на графике правее указанной координаты и может содержать почти любые символы. *Специальные символы* - символы, которые можно наносить на график как точки.

Программа воспримет спецсимвол только в том случае, если он записан в кавычках один и без пробелов.

Спецсимвол	Описание
+	знак "+" отображается как крестик
*	знак "*" отображается в виде пятиконечной звезды
o	знак "o" (маленькая буква О в английской раскладке) отображается как окружность
x	знак "x" (икс маленькая в английской раскладке) отображается как повернутый крестик
.	знак "." (точка) отображается как жирная точка.

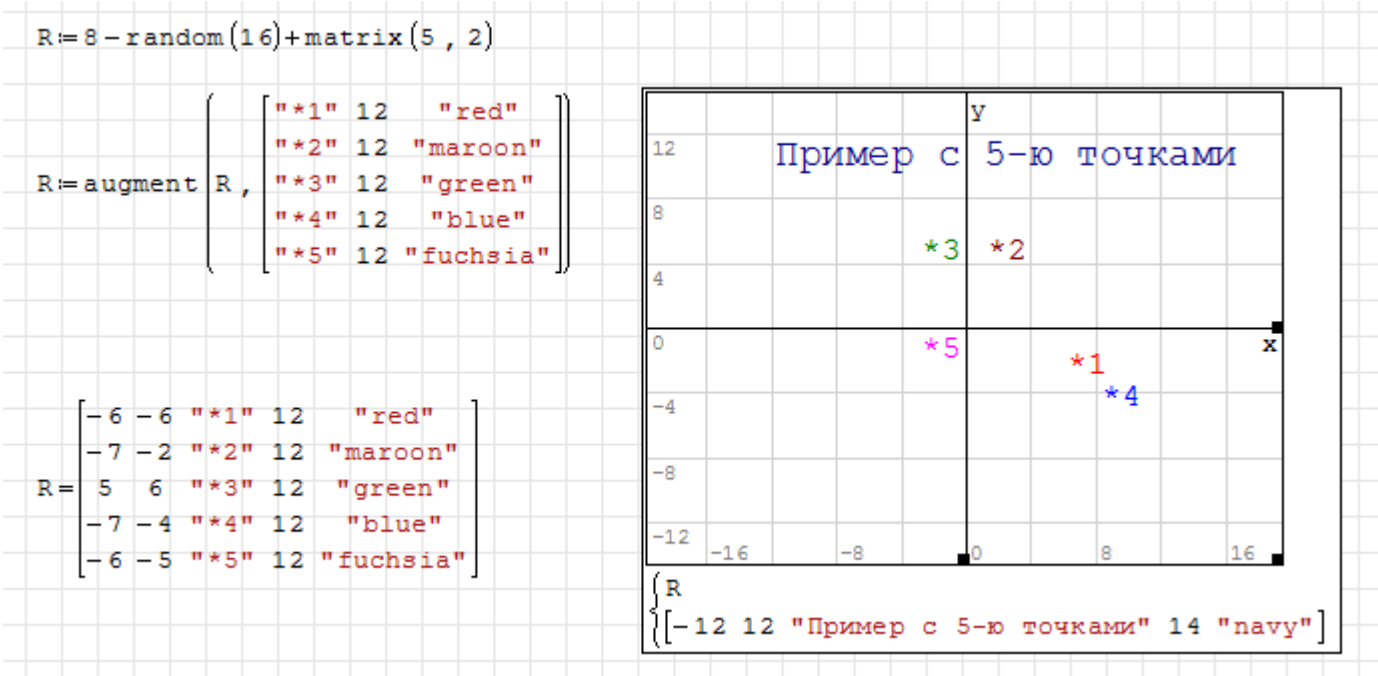
Пример 3. График с использованием спецсимволов:



Параметр `"txtColor"` позволяет задавать цвет надписи. Цвет задается в кавычках, некоторые возможные значения приведены в таблице:

Значение	Цвет	Значение	Цвет
"aqua"	<div></div>	"navy"	<div></div>
"black"	<div></div>	"olive"	<div></div>
"blue"	<div></div>	"purple"	<div></div>
"brown"	<div></div>	"red"	<div></div>
"fuchsia"	<div></div>	"silver"	<div></div>
"gray"	<div></div>	"teal"	<div></div>
"green"	<div></div>	"violet"	<div></div>
"lime"	<div></div>	"white"	<div></div>
"maroon"	<div></div>	"yellow"	<div></div>

Пример 4. График с использованием символов разного цвета:



Составьте следующий отчётный лист с описаниями:

1. Простые арифметические действия

$$a:=1 \quad b:=5 \quad c:=88$$

$$a+b \cdot c - a^2 + \ln(a) - a \sqrt{a \cdot b \cdot c + 10} = -10$$

2. Находим корни квадратного уравнения

$$\text{solve}(x^2 + 2 \cdot x - 4, x) = \begin{bmatrix} -3.2361 \\ 1.2361 \end{bmatrix}$$

3. Находим производную (diff)

по X и по Y для функции:

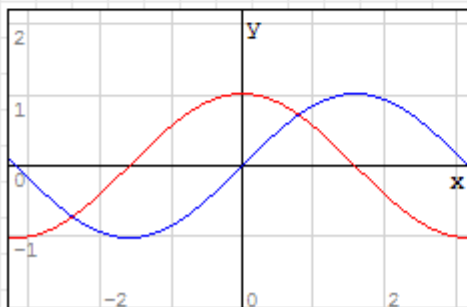
$$(x^5 + 3 \cdot x^2)^3$$

$$\frac{d}{dx} (x^5 + 3 \cdot x^2)^3 = 3 \cdot x^5 \cdot (3 + x^3)^2 \cdot (2 \cdot (3 + x^3) + 3 \cdot x^3)$$

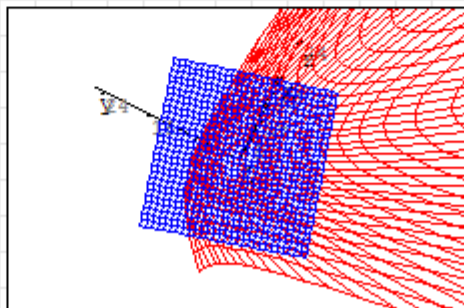
$$\frac{d}{dy} (x^5 + 3 \cdot x^2)^3 = 0$$

Здесь использован оператор символьного вычисления (стрелка вправо на панели Арифметика)

4. Строим графики функций



$$\begin{cases} \sin(x) \\ \cos(x) \end{cases}$$



$$\begin{cases} 1 \\ x^2 + y^2 - x \cdot y \end{cases}$$

6. Функции пользователя

В Smath Studio можно создавать свои функции, которые могут использоваться в расчёте.

Синтаксис описания функции:

<имя функции>(<параметр1> ; <параметр2>; .. ; <параметрN>) := <выражение>

где:

<имя функции> — название функции;

<параметр1> , **<параметр2>**, .. , **<параметрN>** — параметры, с которыми будет вызываться функция;

<выражение> — любое правильно написанное выражение, которое может использовать параметры функции, описывает что именно будет делать функция.

Пример1.

$$\text{sum2}(a; b) := a + b + 1$$

В этом примере объявлена простейшая функция **sum2**, которая складывает два числа **a** и **b**, и прибавляет к ним единицу. Использовать такую функцию можно так же, как и стандартную функцию Smath Studio:

$$\text{sum2}(5; 2) = 8$$

Функции пользователя очень гибкий и удобный инструмент, они позволяют существенно уменьшить объем расчета путем замены часто повторяющихся участков.

- **Важно:** числу параметров, указанному при описании функции должно соответствовать число параметров при вызове. Например:
sum2(5) — неправильно
sum2(5;3;5;8) — неправильно
sum2(5,3) — правильно
- **Важно:** параметрами функции могут быть матрицы, числа и строки. Но при вызове функции должен быть учтен порядок записи переменных. Т.е. если в описании функции 1-я переменная — матрица, 2-я — число, а 3-я — строка, то и при вызове функции 1-й параметр должен быть матрицей, 2-й — числом, 3-й — строкой.

Пример2. В этом примере показано как можно создать массив функций:

```
func_list:= $\begin{bmatrix} x+y \\ x \cdot y \\ x-y \end{bmatrix}$ 

i:=1

test_func(x,y):=func_list_i

test_func(2,3)=5

i:=2

test_func(x,y):=func_list_i

test_func(2,3)=6

i:=3

test_func(x,y):=func_list_i

test_func(2,3)=-1
```

Подробно рассмотрим как это работает:

Строки	Описание
$func_list:=\begin{bmatrix} x+y \\ x \cdot y \\ x-y \end{bmatrix}$	func_list - это матрица (3 строки, 1 столбец), в которой содержатся функции.
$i:=1$ $test_func(x,y):=func_list_i$	Объявляем переменную i Объявляем функцию test_func(x;y) и присваиваем ей значение элемента матрицы func_list_i с номером i
$test_func(2,3)=5$	вызываем функцию test_func , где параметр x=2, y=3 и получаем соответственно ответ: первый элемент матрицы func_list равен x+y — значит функция test_func(x;y) = x+y , а значит 2+3=5

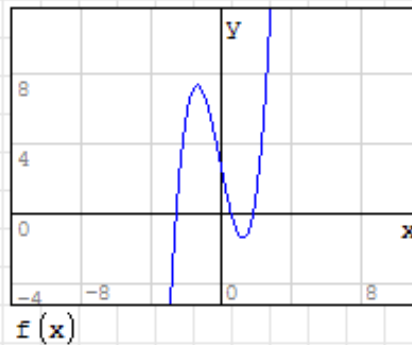
Важно: числу параметров, указанному при описании функции (т.е. в матрице) должно соответствовать число параметров при вызове и при объявлении функции.

Составьте следующий отчётный лист с описаниями:

Пример решения уравнения

$$x^3 - 5 \cdot x = -3$$

$$f(x) := x^3 - 5 \cdot x + 3$$



Уравнение имеет три корня

1 способ:

$$\text{solve}(f(x), x) = \begin{bmatrix} -2.4909 \\ 0.6566 \\ 1.8342 \end{bmatrix}$$

2 способ:

$$v := \begin{bmatrix} 3 \\ -5 \\ 0 \\ 1 \end{bmatrix} \quad \text{polyroots}(v) = \begin{bmatrix} 0.6566 \\ 1.8342 \\ -2.4909 \end{bmatrix}$$

где v - вектор коэффициентов полинома

Пример решения системы линейных уравнений

$$\begin{cases} 3 \cdot x + 4 \cdot y = 5 \\ 5 \cdot x + 4 \cdot y = 3 \end{cases}$$

1 способ (матричный):

$$A := \begin{bmatrix} 3 & 4 \\ 5 & 4 \end{bmatrix} \quad B := \begin{bmatrix} 5 \\ 3 \end{bmatrix}$$

$$A^{-1} \cdot B = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

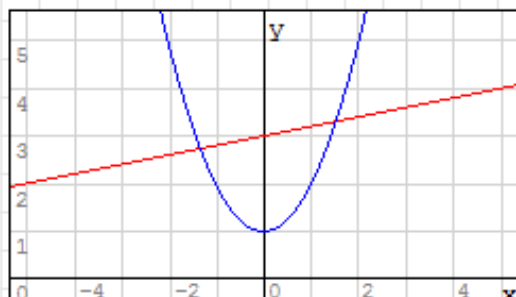
2 способ:

$$S(x, y) := \begin{bmatrix} 3 \cdot x + 4 \cdot y - 5 \\ 5 \cdot x + 4 \cdot y - 3 \end{bmatrix}$$

$$\text{roots}(S(x, y), \begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

Пример решения системы нелинейных уравнений

$$\begin{cases} y = x^2 + 1 \\ y = 0.2 \cdot x + 3 \end{cases}$$



$$\begin{cases} x^2 + 1 \\ 0.2 \cdot x + 3 \end{cases}$$

$$S(x, y) := \begin{bmatrix} x^2 - y + 1 \\ .2 \cdot x - y + 3 \end{bmatrix}$$

$$\text{roots}(S(x, y), \begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} -2 \\ 3 \end{bmatrix}) = \begin{bmatrix} -1.3177 \\ 2.7365 \end{bmatrix}$$

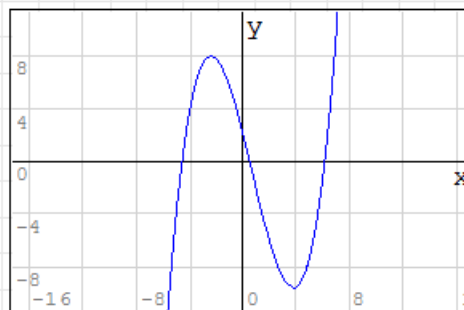
$$\text{roots}(S(x, y), \begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix}) = \begin{bmatrix} 1.5177 \\ 3.3035 \end{bmatrix}$$

В последнем примере на основе построенного графика было решено вести поиск корней от точек с координатами $(-2, 3)$ и $(2, 3)$. Для более удобного отображения результата использована Численная оптимизация (через контекстное меню).

Проверьте следующие прикладные задачи, в которых необходимо решить уравнение или систему уравнений.

Пример нахождения экстремумов (минимумов/максимумов) функции

$$f(x) := 0.15 \cdot x^3 - 0.3 \cdot x^2 - 4 \cdot x + 2$$



$f(x)$

$$d(x) := \frac{d}{dx} f(x)$$

$$x1 := \text{solve}(d(x), x, -8, 0)$$

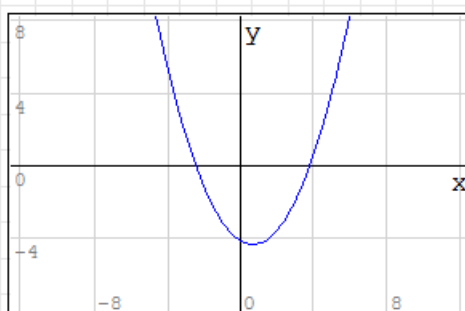
$$x1 = -2.3884$$

— 1-й локальный экстремум на отрезке $(-8, 0)$

$$x2 := \text{solve}(d(x), x, 0, 8)$$

$$x2 = 3.7217$$

— 2-й локальный экстремум на отрезке $(0, 8)$



$d(x)$

Знак производной в окрестности точки $x1$ меняется с плюса на минус, поэтому в этой точке функция достигает максимума.

В окрестности точки $x2$ знак производной поменялся с минуса на плюс, поэтому в этой точке функция достигает минимума.

$$\frac{d^2}{dx^2} f(x1) = -2.7495$$

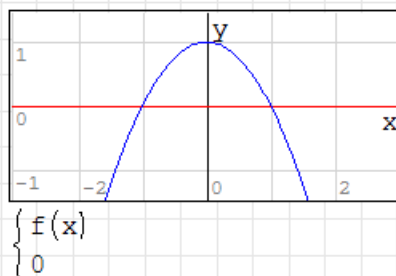
В точке $x1$ вторая производная меньше нуля, значит, точка $x1$ соответствует максимуму функции.

$$\frac{d^2}{dx^2} f(x2) = 2.7495$$

В точке $x2$ вторая производная больше нуля, значит, точка $x2$ соответствует минимуму функции.

Нахождение площади фигуры, ограниченной линиями

$$f(x) := 1 - x^2$$

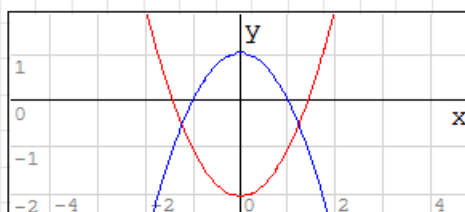


$$\text{solve}(f(x), x) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$S := \int_{-1}^1 f(x) dx$$

$$S = 1.3333$$

$$f1(x) := 1 - x^2 \quad f2(x) := x^2 - 2$$



$$X := \text{solve}(f1(x) - f2(x), x)$$

$$X_1 = -1.2247 \quad X_2 = 1.2247$$

$$S := \int_{X_1}^{X_2} |f1(x) - f2(x)| dx$$

$$S = 4.899$$

Построение кривых по заданным точкам

Пример1. Построение прямой, проходящей через точки A(-2,-4) и B(5,7).

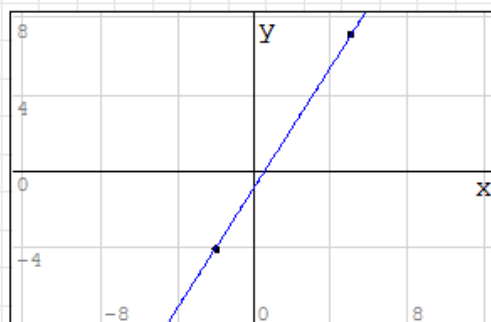
Подставим в уравнение прямой координаты данных точек и получим систему:

$$\begin{cases} -4 = -2 \cdot a + b \\ 7 = 5 \cdot a + b \end{cases}$$

$$A := \begin{bmatrix} -2 & 1 \\ 5 & 1 \end{bmatrix} \quad B := \begin{bmatrix} -4 \\ 7 \end{bmatrix} \quad X := A^{-1} \cdot B \quad X = \begin{bmatrix} 1.5714 \\ -0.8571 \end{bmatrix} \quad a := X_1 \quad b := X_2$$

Уравнение прямой будет иметь вид $y = ax + b$, где $a = \frac{11}{7}$ $b = -\frac{6}{7}$

$$f(x) := a \cdot x + b$$



$$\begin{cases} f(x) \\ [-2 \ -4 \ "."] \\ [5 \ 7 \ "."] \end{cases}$$

Пример2. Построение параболы, проходящей через точки A(-1,-4), B(1,-2) и C(3,16)

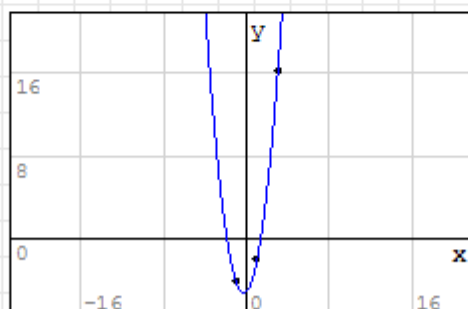
Подставляем в уравнение параболы заданные координаты точек и получаем систему:

$$\begin{cases} -4 = a \cdot (-1)^2 + b \cdot (-1) + c \\ -2 = a \cdot 1^2 + b \cdot 1 + c \\ 16 = a \cdot 3^2 + b \cdot 3 + c \end{cases}$$

$$A := \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & 1 \\ 9 & 3 & 1 \end{bmatrix} \quad B := \begin{bmatrix} -4 \\ -2 \\ 16 \end{bmatrix} \quad X := A^{-1} \cdot B \quad X = \begin{bmatrix} 2 \\ 1 \\ -5 \end{bmatrix}$$

Получаем уравнение параболы $y = 2 \cdot x^2 + x - 5$

$$f(x) := 2 \cdot x^2 + x - 5$$



$$T := \begin{bmatrix} -1 & -4 & ". " \\ 1 & -2 & ". " \\ 3 & 16 & ". " \end{bmatrix}$$

$$\begin{cases} f(x) \\ T \end{cases}$$

Пример3. Построение окружности, проходящей через три точки A(2,4), B(5,0.99) и C(2,-2)

Подставим в уравнение окружности заданные координаты точек и получим систему:

$$\begin{cases} (2-x_0)^2 + (4-y_0)^2 = R^2 \\ (5-x_0)^2 + (0.99-y_0)^2 = R^2 \\ (2-x_0)^2 + (-2-y_0)^2 = R^2 \end{cases}$$

$$S(x_0, y_0, R) := \begin{bmatrix} (2-x_0)^2 + (4-y_0)^2 - R^2 \\ (5-x_0)^2 + (0.99-y_0)^2 - R^2 \\ (2-x_0)^2 + (-2-y_0)^2 - R^2 \end{bmatrix} \quad \text{roots} \left(S(x_0, y_0, R), \begin{bmatrix} x_0 \\ y_0 \\ R \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

Подставим полученные координаты центра окружности и радиус в уравнение окружности:

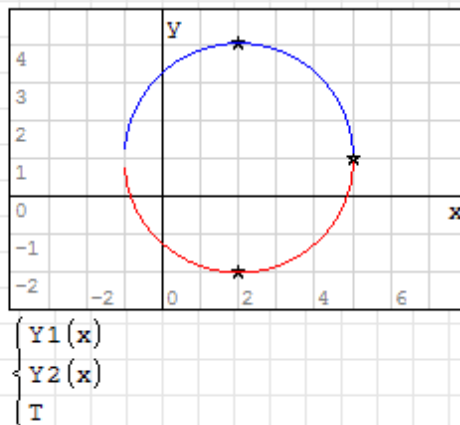
$$(x-2)^2 + (y-1)^2 = 3^2$$

Выразим Y и построим окружность:

$$Y1(x) := 1 + \sqrt{9 - (x-2)^2}$$

$$Y2(x) := 1 - \sqrt{9 - (x-2)^2}$$

$$T := \begin{bmatrix} 2 & 4 & "*" \\ 5 & .99 & "*" \\ 2 & -2 & "*" \end{bmatrix}$$



7. Функции программирования

1. Условный переход if-else

Оператор условного перехода **if-else** служит для выполнения определенного набора команд, в зависимости от выполнения условия.

Пример1:

```

a:=5
b:=6
if a>b
    max:=a
else
    max:=b
max=6
    
```

В начале функция **if** проверяет условие **a>b** и если это истина, то выполняется следующее после **if** выражение **max:=a**. В противном случае выполняется выражение **max:=b** следующее после **else**.

А что если нужно выполнить только одно условие после **if** или **else**?

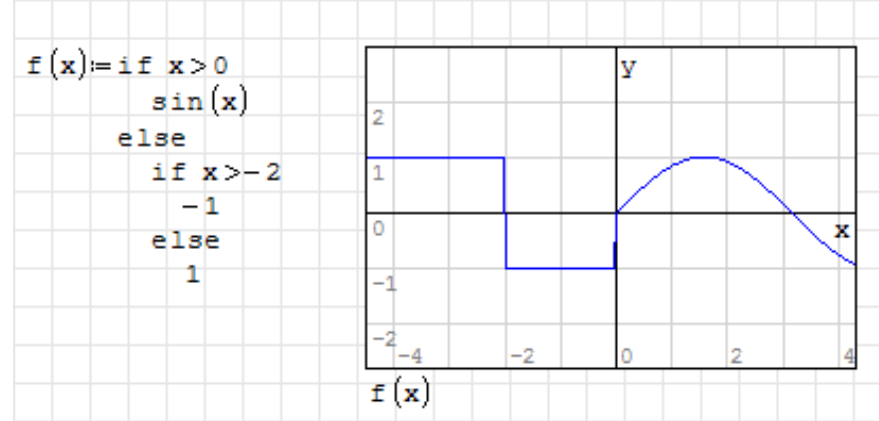
В таком случае можно записать на месте ненужного выражения любое число, например 1.

Пример2:

```
a:=5
b:=4
if a>b
  1
else
  a:=b+1
  a=5
```

Здесь если выражение **a>b** истинно, то программа ничего не будет делать и приступит к вычислению следующего оператора после блока **if-else**.

Пример3:



2. Циклы for, while

Цикл — специальная конструкция, предназначенная для многократного повторения набора инструкций. Количество повторений может задаваться заранее или зависеть от выполнения определенного условия.

Цикл со счетчиком for

Пример1:

```
a:=0
cntr:=8
for i:=2, i<cntr, i:=i+2
  a:=a+1
a=3
```

Как это работает:

Выражение	Описание
a:=0	Присваиваем переменной a число 0. Она нужна для того чтобы узнать количество повторений цикла
cntr:=8	Эта переменная нужна для условия цикла
for i:=2, i<cntr, i:=i+2	Здесь задаем цикл: i:=2 это счетчик цикла, он показывает на каком шаге находится цикл i<cntr логическое условие, когда оно станет ложным — цикл закончится i:=i+2 эта конструкция задает шаг цикла, т.е i будет изменяться с шагом 2
a:=a+1	Переменная позволяет узнать сколько повторений выполнил цикл. В первой строке a=0 , при первом проходе мы задаем переменной a её же значение +1
a=3	Поскольку a=3 , команды в цикле выполнились 3 раза

- Важно:** если условие построено таким образом, что выдает всегда истину — цикл никогда не остановится.

cntr:=1 for i:=2, i>cntr, i:=i+1	cntr:=2 for i:=2, i=cntr, i:=i+1 cntr:=i
-------------------------------------	--

- **Важно:** если счетчик цикла не изменяется или изменяется неправильно, то цикл может работать бесконечно:

<pre>cntr:=8 for i:=2, i<cntr, i:=i a:=a+1</pre>	<pre>cntr:=8 for i:=2, i<cntr, i:=i-1 a:=a+1</pre>
---	---

- **Важно:** если условие цикла не выполняется изначально, то цикл ни разу не выполнится.

```
a:=0
cntr:=1
for i:=2, i<cntr, i:=i+2
  a:=a+1
a=0
```

- **Обратите внимание:** для выхода из цикла достаточно изменить счетчик цикла так, чтобы условие цикла стало **ЛОЖНЫМ**. Например, следующий цикл завершит свою работу на 3 шаге:

```
a:=0
cntr:=50
for i:=1, i<cntr, i:=i+2
  if i = 3
    i:=cntr
  else
    1
```

Пример2:

```
j := 0
t := 1 .. 5

for x ∈ t
  j := j + x
j = 15
x = 5
```

В этом примере счетчик цикла **x** меняется сам и принадлежит диапазону **t**.

Обратите внимания на строчку **t:=1..5**, где использована функция **range**, которая присваивает переменной **t** диапазон от 1 до 5 с шагом 1.

Здесь цикл повторяется пять раз, что видно из значения **j=15** ($j=0+1+2+3+4+5=15$). Значение **x** внутри цикла берется из диапазона. Если диапазон **t = (1,2,3,4,5)**, то при каждом следующем шаге **x** будет равен следующему значению, взятому из этого диапазона.

Пример3:

```
t:=1..3

for j ∈ t
  for i ∈ t
    Ai j := i - j

A =  $\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$ 
```

Цикл с предварительной проверкой условия **while**

Пример4:


```
i := 2
a := 1
while i < 5
  i := i + 1
  a := a + 1
i = 5
a = 4
```

Обратите внимание на вертикальную линию под командой **while**. Здесь использована команда **line**.

Как это работает:

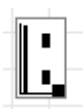
Выражение	Описание
i:=2	Присваиваем переменной i число 2, эта переменная будет счетчиком цикла
a:=1	Эта переменная позволяет посчитать, сколько повторений выполнил цикл
while i<5	Здесь задаём цикл: i<5 означает, что цикл будет выполняться пока это условие верно
i:=i+1	Изменяем счетчик цикла на единицу
a:=a+1	Переменная позволяет узнать, сколько повторений выполнил цикл. При каждом проходе увеличиваем a на 1
a=4	Поскольку a=4 цикл повторился 3 раза, т.к. начальное значение a=1

- **Важно:** все замечания по циклу **for** также относятся и к циклу **while**.
- **Важно:** переменная, которая участвует в условии цикла, должна быть задана, иначе вы не сможете предсказать как будет работать цикл.
- **Важно:** в теле цикла должна быть предусмотрена конструкция, которая изменяет счетчик цикла, иначе цикл будет выполняться бесконечно.

Если Вы запустили расчёт с бесконечным циклом (или хотите прервать цикл), то для остановки расчета можно использовать кнопку  (Вычисление – Прервать вычисление).

3. Составная команда **Line**

Команда **Line** позволяет сделать составную команду, т.е. одну команду из нескольких. Все команды циклов, а также оператор ветвления позволяют записывать внутри себя лишь одну команду. Чтобы внутри этих команд можно было выполнить целый ряд действий, используют **line**. На рисунке ниже показана команда **line**:



По умолчанию она вставляется с двумя пустыми полями ввода, туда можно записать нужные команды. Чёрный квадратик справа внизу служит для растягивания линии, с его помощью линию можно растянуть/сжать на столько команд, на сколько нужно.

Внутри оператора **line** можно вставлять еще несколько операторов **line**, вложений может быть много, но на практике больше 2-3 обычно не используется. Если навести мышку на оператор **line**, то появится всплывающая подсказка, где показано состояние всех переменных.

Пример1:

```
s:=2
for i:=1, i≤4, i:=i+1
  if s>0
    s:=s+1
    p:=s-i
  else
    0
  p:=i
```

i = 5
s = 6
p = 4

Пример2:

```
t := -π, -π +  $\frac{\pi}{12}$  .. π
```

```
i := 1
```

```
for x ∈ t
```

```
  F1i := x
```

```
  F2i := sin(x)
```

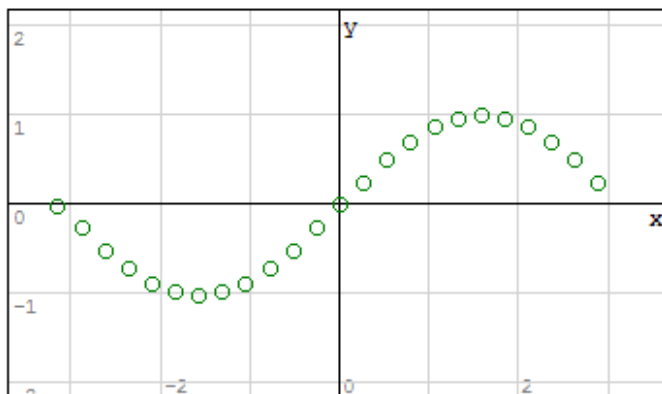
```
  F3i := "o"
```

```
  F4i := 12
```

```
  F5i := "green"
```

```
  i := i + 1
```

```
F := augment(F1, F2, F3, F4, F5)
```



```
F = [ -3.1416  -3.2311·10-15  "o"  12  "green"
      -2.8798  -0.2588  "o"  12  "green"
      -2.618   -0.5     "o"  12  "green"
      -2.3562  -0.7071  "o"  12  "green"
      -2.0944  -0.866   "o"  12  "green"
      -1.8326  -0.9659  "o"  12  "green"
      -1.5708  -1       "o"  12  "green"
      -1.309   -0.9659  "o"  12  "green"
      -1.0472  -0.866   "o"  12  "green"
      -0.7854  -0.7071  "o"  12  "green"
      ⋮
```

Пример3. Пример по преобразованию арабских чисел в римские:

Римские цифры

Функция преобразования Арабских чисел в Римские:

```
ArabicToRoman(arabic) = relation = (1000 900 500 400 100 90 50 40 10 9 5 4 1)
                                ("M" "CM" "D" "CD" "C" "XC" "L" "XL" "X" "IX" "V" "IV" "I")
if (arabic ≤ 0) ∨ (arabic ≥ 4000)
  roman := "Value must be in the range 1-3999."
else
  roman := ""
  for j = 1, j ≤ cols(relation), j = j + 1
    while arabic ≥ relation1 j
      arabic = arabic - relation1 j
      roman = concat(roman, relation2 j)
  roman
```

Пример использования:

```
ArabicToRoman(1) = "I"
```

```
ArabicToRoman(8) = "VIII"
```

```
ArabicToRoman(58) = "LVIII"
```

+

Описание работы функции ArabicToRoman

Строка	Описание						
ArabicToRoman(arabic):=	Здесь объявляем функцию. Объявление функции может состоять из 2 частей: 1. Имя функции, должно быть обязательно указано. 2. Используемые переменные перечисляются в скобках через запятую.						
Как видно, в этой функции активно используется оператор line	Необходим для выполнения несколько команд.						
1 строка <pre>relation = (1000 900 500 400 100 90 50 40 10 9 5 4 1) ("M" "CM" "D" "CD" "C" "XC" "L" "XL" "X" "IX" "V" "IV" "I")</pre>	Это матрица 2x13, в ней содержатся арабские числа и соответствующие им римские.						
2 строка <pre>if (arabic ≤ 0) v(arabic ≥ 4000) roman := "Value must be in the range 1-3999."</pre>	Т.к. в нашем случае римское число должно быть не больше 3999, и не меньше 1. <ul style="list-style-type: none"> если логическое выражение вернёт истину, то программа выполнит следующую строчку после if (см. строчку 3), а затем перейдёт к следующей после блока if- else инструкции (см. строчку 8) если - ложь, то программа приступит к выполнению команд после оператора else 						
3 строка <pre>roman := "Value must be in the range 1-3999."</pre>	Эта строка выполнится, только если логическое выражение после if в строке 2 вернет истину . В этой строке переменной roman присваиваем текст «Число должно быть в диапазоне 1-3999.»						
4 строка <pre>roman := ""</pre>	Весь блок команд после else выполнится, только если логическое выражение после if в строке 2 не вернёт истину . В этой строчке указываем программе, что переменная roman будет содержать текстовое значение, т.к. по умолчанию все переменные численные .						
5 строка <pre>for j:=1, j ≤ cols(relation), j:=j+1,</pre>	Объявляем цикл, для того, чтобы пройти по всем элементам матрицы relation . <table border="1" data-bbox="820 1415 1505 1765"> <tr> <td>j:=1</td><td>счетчик цикла</td></tr> <tr> <td>j ≤ cols(relation)</td><td>логическое условие (цикл будет выполняться пока условие не станет ложным) Буквально означает: пока j меньше либо равно кол-ву столбцов в матрице relation</td></tr> <tr> <td>j:=j+1</td><td>увеличиваем счетчик цикла на 1 при каждом проходе</td></tr> </table> <p>В нашем случае цикл повториться 13 раз, переменная j изменится с шагом 1 от 1 до 13</p>	j:=1	счетчик цикла	j ≤ cols(relation)	логическое условие (цикл будет выполняться пока условие не станет ложным) Буквально означает: пока j меньше либо равно кол-ву столбцов в матрице relation	j:=j+1	увеличиваем счетчик цикла на 1 при каждом проходе
j:=1	счетчик цикла						
j ≤ cols(relation)	логическое условие (цикл будет выполняться пока условие не станет ложным) Буквально означает: пока j меньше либо равно кол-ву столбцов в матрице relation						
j:=j+1	увеличиваем счетчик цикла на 1 при каждом проходе						
6 строка (внутри цикла for объявляем цикл) <pre>while arabic ≥ relation_1 j</pre>	Означает: повторять пока выполняется условие <code>arabic ≥ relation_1 j</code> Дословно условие читается так: пока arabic больше либо равно значению ячейки матрицы relation с координатами (1 строка, j столбец). Где j — счетчик цикла из 5 строки.						

	<table> <tr> <td>j=1</td><td>relation_{1 j} = 1000</td></tr> <tr> <td>j=2</td><td>relation_{1 j} = 900</td></tr> <tr> <td>j=3</td><td>relation_{1 j} = 500</td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td>j=12</td><td>relation_{1 j} = 4</td></tr> <tr> <td>j=13</td><td>relation_{1 j} = 1</td></tr> </table>	j=1	relation _{1 j} = 1000	j=2	relation _{1 j} = 900	j=3	relation _{1 j} = 500	j=12	relation _{1 j} = 4	j=13	relation _{1 j} = 1
j=1	relation _{1 j} = 1000												
j=2	relation _{1 j} = 900												
j=3	relation _{1 j} = 500												
...	...												
j=12	relation _{1 j} = 4												
j=13	relation _{1 j} = 1												
7 и 8 строчки (внутри цикла while) <pre>arabic:=arabic-relation_{1 j} roman:=concat(roman, relation_{2 j})</pre>	Здесь вначале переменной arabic присваиваем её же значение минус значение 1-й строки j столбца матрицы relation до тех пор пока логическое условие в 6 строчке не станет ложным. Затем в строковой переменной roman накапливаем значение 2-й строки j столбца матрицы relation . См. команду concat .												
9 строчка roman	Функции в Smath Studio устроены так, что возвращают значение последней операции. Функция вернёт значение переменной roman .												