

---

# ***Основы создания программ-функций в среде MathCAD***

---

Реализовать тот или иной алгоритм вычисления в среде MathCAD можно двумя способами:

---

1 способ:

размещать обращение к соответствующим переменным, выражениям, операторам, функциям непосредственно в тексте документа MathCAD без какой-либо организационной структуры; данный способ называется *программированием в тексте документа*;

2 способ:

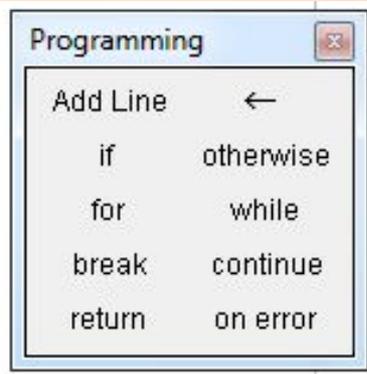
использовать организационные структуры, так называемые программы-функции, которые содержат конструкции, во многом подобные конструкциям таких языков программирования высокого уровня как FORTRAN, Pascal, C++, C# и др.

В данной лекции будем рассматривать второй способ реализации алгоритмов, который называется ***программированием в программе-функции***.

Для создания и использования программ-функций применяется панель инструментов Programming (Программирование).



Кнопка	Название	Назначение кнопки (команды)
Add Line	Add Program Line	позволяет <i>создать и/или удлинить вертикальную линию</i> , справа от которой формируется запись программного блока;
←	Local Definition	позволяет вставить <i>оператор внутреннего присваивания</i> (активен в теле модуля);
if	If Statement	позволяет вставить <i>условный оператор</i> ;
otherwise	Otherwise Statement	позволяет вставить <i>оператор иного выбора</i> ;
for	For Loop	позволяет вставить <i>оператор цикла с заданным числом повторений</i> ;
while	While Loop	позволяет вставить <i>оператор цикла типа «пока»</i> (итерационный цикл);
break	Break Statement	позволяет вставить <i>оператор прерывания</i> ;
continue	Continue Statement	позволяет вставить <i>оператор продолжения</i> ;
return	Return Statement	позволяет вставить <i>оператор-функцию возврата</i> ;
on error	Return Statement	позволяет вставить <i>оператор-функцию возврата</i> ;



# Примеры программ-функций:

Пример программы-функции, вычисляющей сумму ряда:

$$S = 1 + \frac{1}{2^2} - \frac{1}{3^2} + \frac{1}{4^2} - \dots \pm \frac{1}{n^2}$$

```
x ← 1
ε ← 0.001
while 1/x > ε
  s ← s + 1/x2
  x ← x + 1
s
```

= 1.613

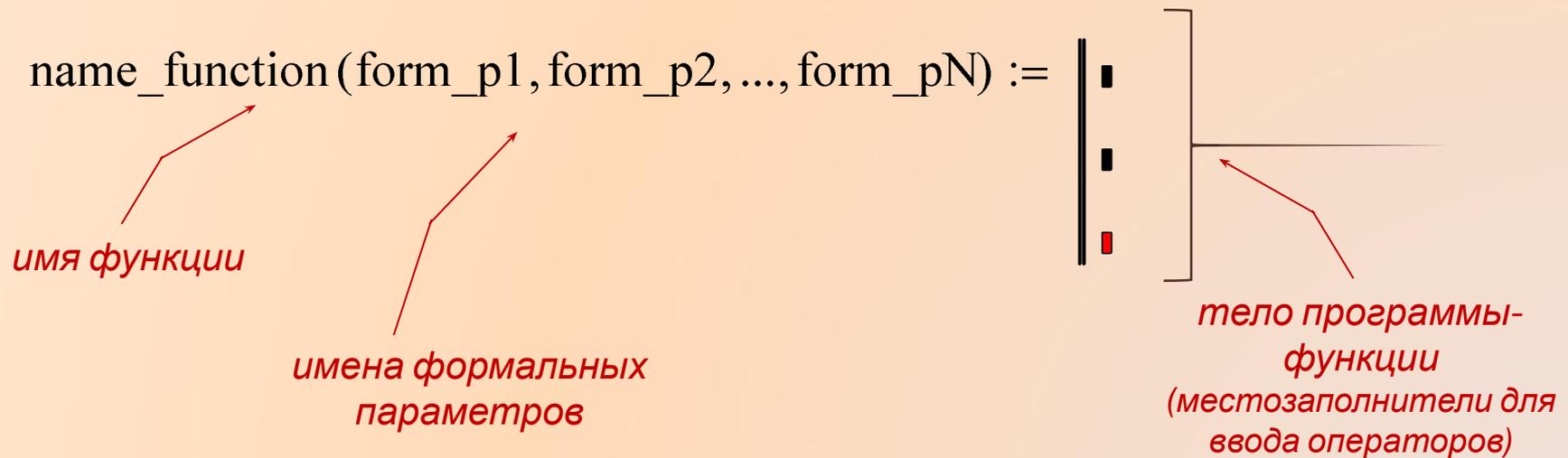
Пример программы-функции, реализующей итерационную процедуру приближенного вычисления корня квадратного из числа a;

```
sqrt(a, eps) :=
  xc ← 10000000000
  xn ← a
  while |xn - xc| > eps
    xc ← xn
    xn ← (xc + a/xc) / 2
  xn
```

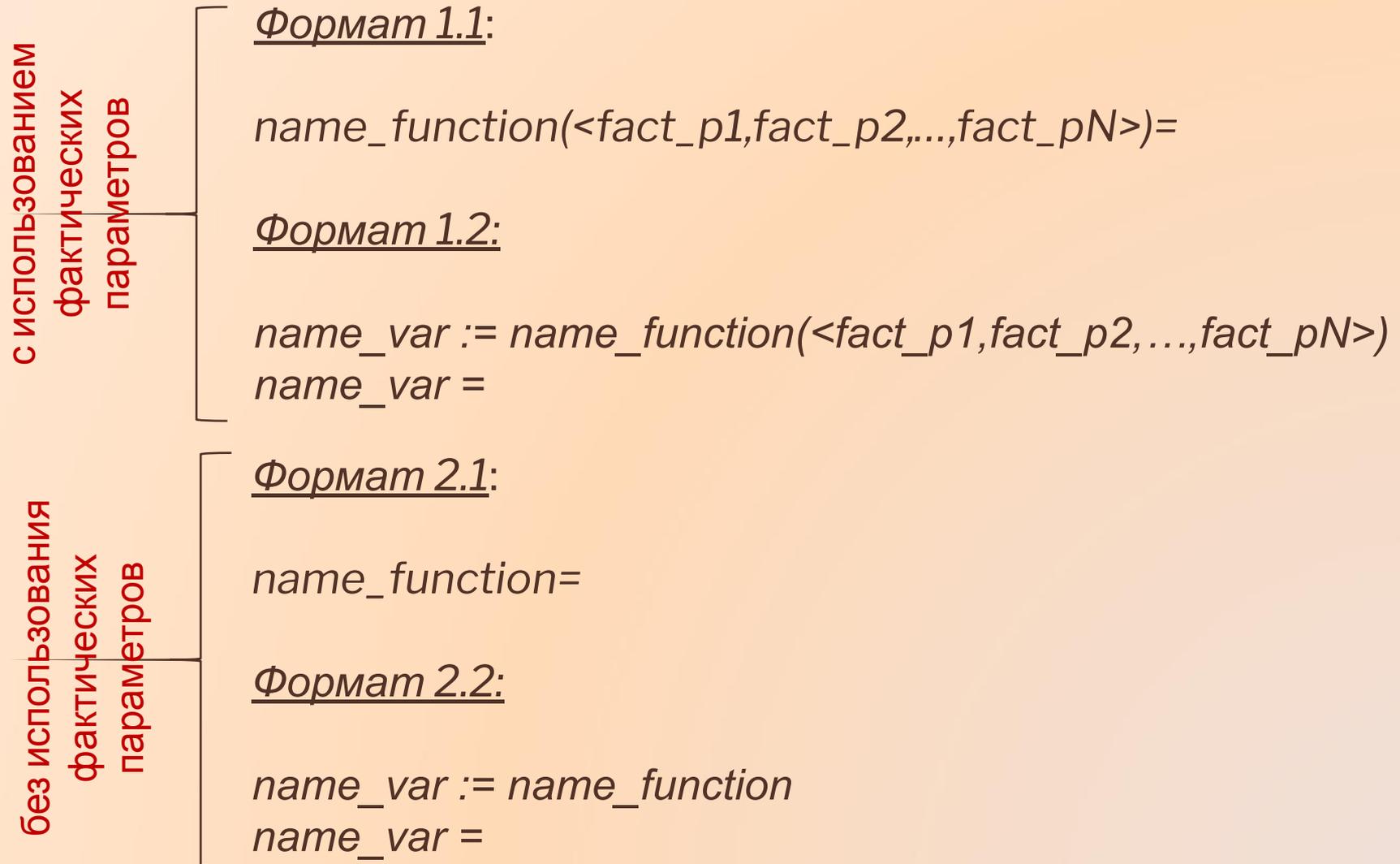
Процесс программирования в программе-функции включает в себя два этапа:

- 1) описание программы-функции;
- 2) обращение к программе-функции (вызов).

Описание программы – функции размещается в рабочем документе перед вызовом программы-функции и имеет следующую структуру:



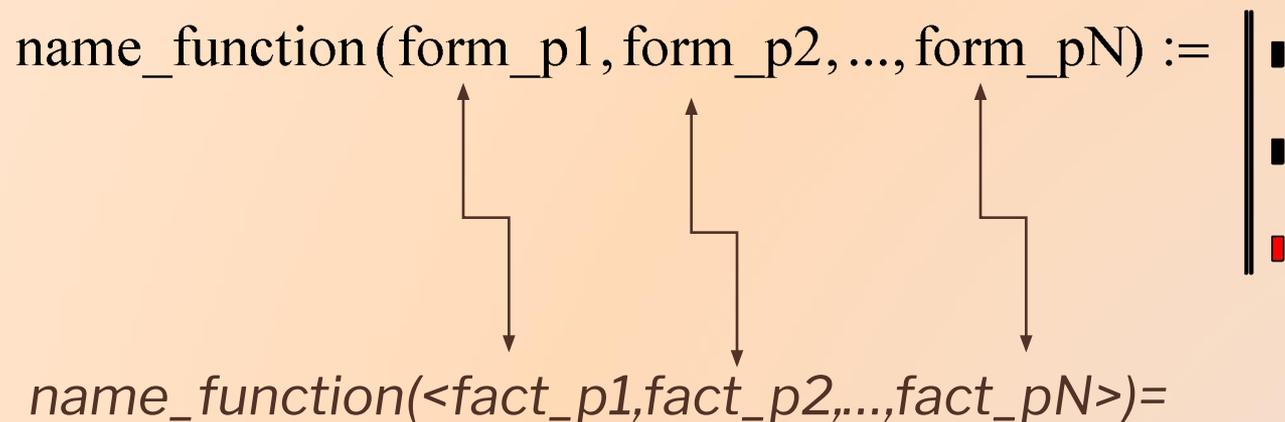
Обращение к программе-функции (вызов) располагается правее или ниже описания программы-функции и может иметь следующую структуру:



Между формальными и фактическими параметрами должно быть установлено взаимно-однозначное соответствие по:

---

- количеству;
- порядку следования;
- типу.



# Варианты обращений к программе-функции, реализующей итерационную процедуру приближенного вычисления корня квадратного из числа a:

1

```
sqrt(a, eps) :=
  xc ← 10000000000
  xn ← a
  while |xn - xc| > eps
    xc ← xn
    xn ← (xc + a/xc) / 2
  xn
```

$\text{sqrt}(25, 0.001) = 5$

$\text{sqrt}(123, 0.000001) = 11.091$

$\text{sqrt}(9, 0.0001) = 3$

2

```
sqrt(a, eps) :=
  xc ← 10000000000
  xn ← a
  while |xn - xc| > eps
    xc ← xn
    xn ← (xc + a/xc) / 2
  xn
```

$y := \text{sqrt}(25, 0.001) \quad y = 5$

$y := \text{sqrt}(144, 0.01) \quad y = 12$

3

```
a := 25      eps := 0.01
sqrt :=
  xc ← 10000000000
  xn ← a
  while |xn - xc| > eps
    xc ← xn
    xn ← (xc + a/xc) / 2
  xn
```

$\text{sqrt} = 5$

**Линейный алгоритм (следование)** - вычислительный процесс, в котором необходимые действия выполняются строго последовательно и каждое один раз.

В реализации алгоритмов линейной структуры в программах-функциях используется **оператор внутреннего присваивания** (локальный оператор присваивания), с помощью которого осуществляется внутри программы-функции присваивание значений (числовых, строковых) переменным.

Формат оператора:

*< имя\_переменной > ← < выражение >*

$f(x) :=$ 

x ← x + 2
z ← x <sup><math>\frac{1}{3}</math></sup>
z

*выражение, определяющее возвращаемое через имя программы-функции значение*

**Замечание:** использование "обычного" оператора присваивания ( := ) в теле программы-функции приводит к синтаксической ошибке.

Варианты обращений к программе-функции, реализующей вычисление значения переменной z:

$$f(x) := \begin{array}{l} x \leftarrow x + 2 \\ \frac{1}{3} \\ z \leftarrow x^3 \\ z \end{array}$$

$z := f(2)$

$z = 1.587$

*различные  
переменные, не  
связанные  
между собой*

$x := 2$

$f := \begin{array}{l} x \leftarrow x + 2 \\ \frac{1}{3} \\ z \leftarrow x^3 \\ z \end{array}$

*используется  
значение  $x=2$*

*значение  
переменной  $x$   
изменяется*

$f = 1.587$

$x = 2$

*вызов  
программы-  
функции*

*вне программы-функции  
переменная **сохраняет**  
свое **значение***

**Пример 1.** Написать программу-функцию, вычисляющую значения корней квадратного уравнения  $ax^2 + bx + c = 0$ .

$$qq1(a, b, c, sig1) := \begin{cases} d \leftarrow b^2 - 4 \cdot a \cdot c \\ x \leftarrow \frac{-b + sig1 \cdot \sqrt{d}}{2 \cdot a} \end{cases}$$

$$qq1(9, -42, 35, 1) = 3.581$$

$$qq1(9, -42, 35, -1) = 1.086$$

$$qq1(9, -4, 5, -1) = 0.222 - 0.711i$$

**Пример 2.** Написать программу-функцию, вычисляющую произведение цифр четырехзначного положительного целого числа N.

**Примечание к примеру 2:** для решения задачи следует использовать функции:

**mod(x,y)** – возвращает остаток при делении числа x на y (x и y – действительные числа, результат наследует знак числа x);

**trunc(x)** – возвращает целую часть действительного числа x.

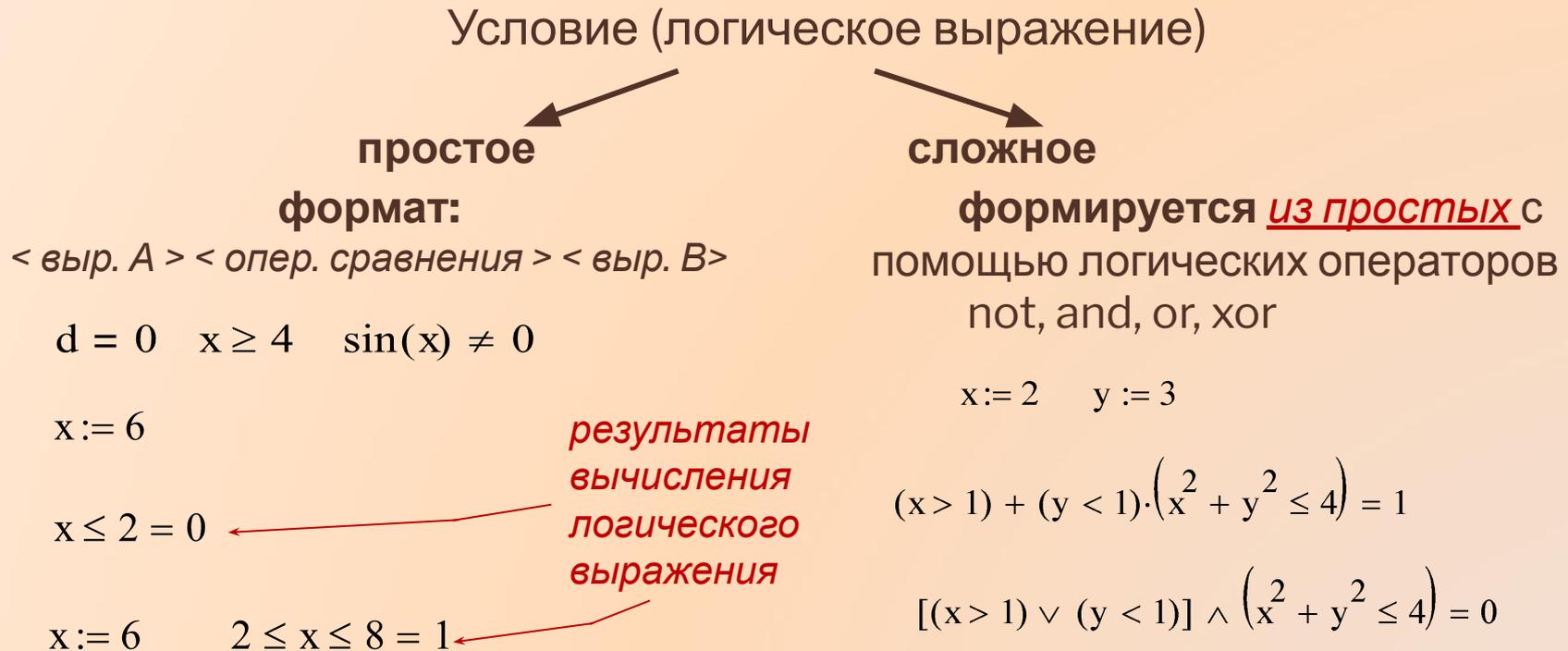
$$g(n) := \begin{cases} n1 \leftarrow \text{mod}(n, 10) \\ n2 \leftarrow \text{mod}\left(\text{trunc}\left(\frac{n}{10}\right), 10\right) \\ n3 \leftarrow \text{mod}\left(\text{trunc}\left(\frac{n}{100}\right), 10\right) \\ n4 \leftarrow \text{trunc}\left(\frac{n}{1000}\right) \\ p \leftarrow n1 \cdot n2 \cdot n3 \cdot n4 \end{cases}$$

$$g(4562) = 240$$



Для реализации ветвлений в программах-функциях используется:

- условная функция **if** (встроенная функция системы MathCAD);
- условный оператор **If Statement** и оператор иного выбора **Otherwise Statement** (с панели инструментов Программирование) .



Следует отметить, что при формировании сложных условий вместо операторов And, Or могут быть также использованы арифметические операторы сложения и умножения:

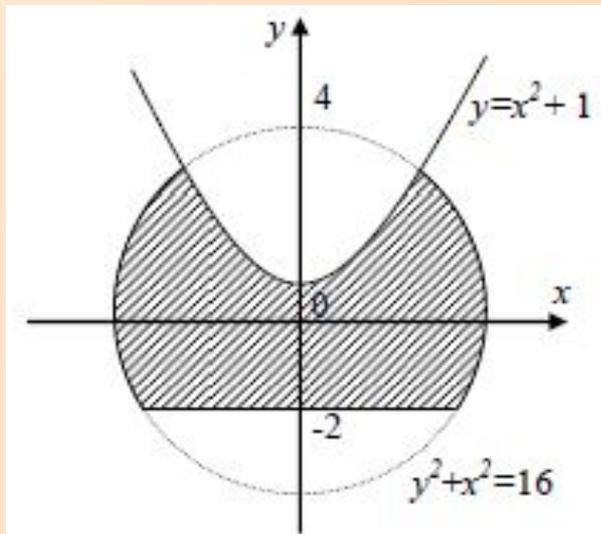
$$(x > 1) + (y < 1) \cdot (x^2 + y^2 \leq 4) = 1$$

## Формат условной функции if:

if ( < логич. выраж. > , < ариф.выраж.1> , < ариф.выраж.2 > )

Внутри программы-функции условная функция используется в арифметических выражениях, стоящих в правой части локального оператора присваивания (оператора внутреннего присваивания).

**Пример 3.** Написать программу-функцию, определяющую принадлежность точки с координатами  $x$  и  $y$  заштрихованной области на рисунке.



**Решение:** заштрихованная область – это точки, удовлетворяющие системе неравенств:

$$\begin{cases} y \geq -2 \\ y \leq x^2 + 1 \\ x^2 + y^2 \leq 16 \end{cases}$$

$$w(x, y) := \begin{cases} z \leftarrow \text{if}[(y \geq -2) \cdot (y \leq x^2 + 1) \cdot (y^2 \leq 16 - x^2)], \text{"точка принадлежит области"} , \text{"точка не принадлежит области"} \\ z \end{cases}$$

$$w(-2, -3.5) = \text{"точка не принадлежит области"} \blacksquare$$

$$w(0, 0) = \text{"точка принадлежит области"} \blacksquare$$

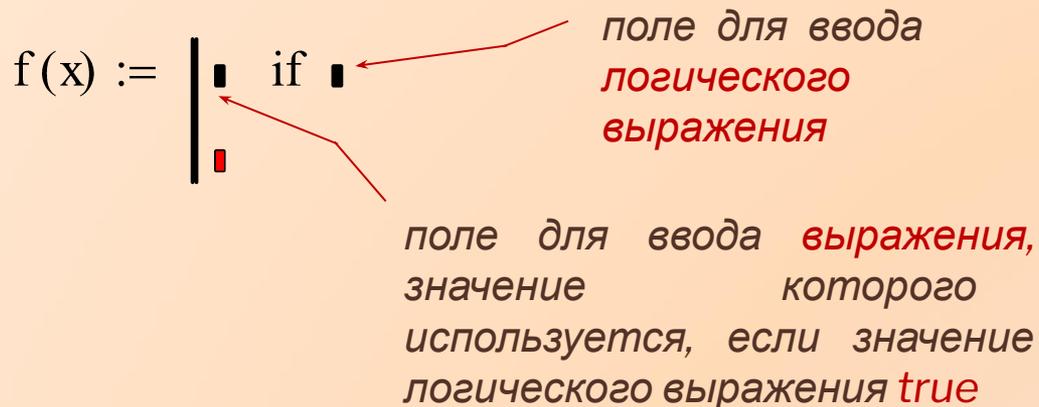
## Формат использования

условного оператора **if** и оператора иного выбора **Otherwise**

---

$f(x) :=$ 

	if	

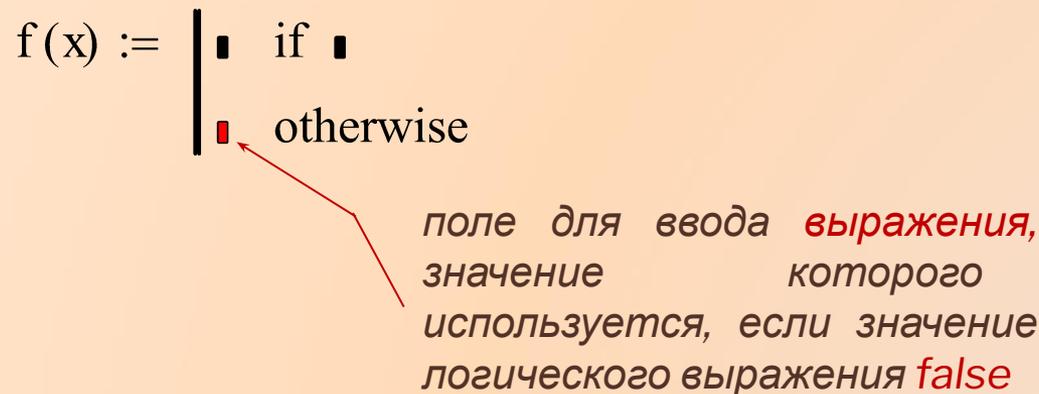
  


поле для ввода логического выражения

поле для ввода **выражения**, значение которого используется, если значение логического выражения **true**

$f(x) :=$ 

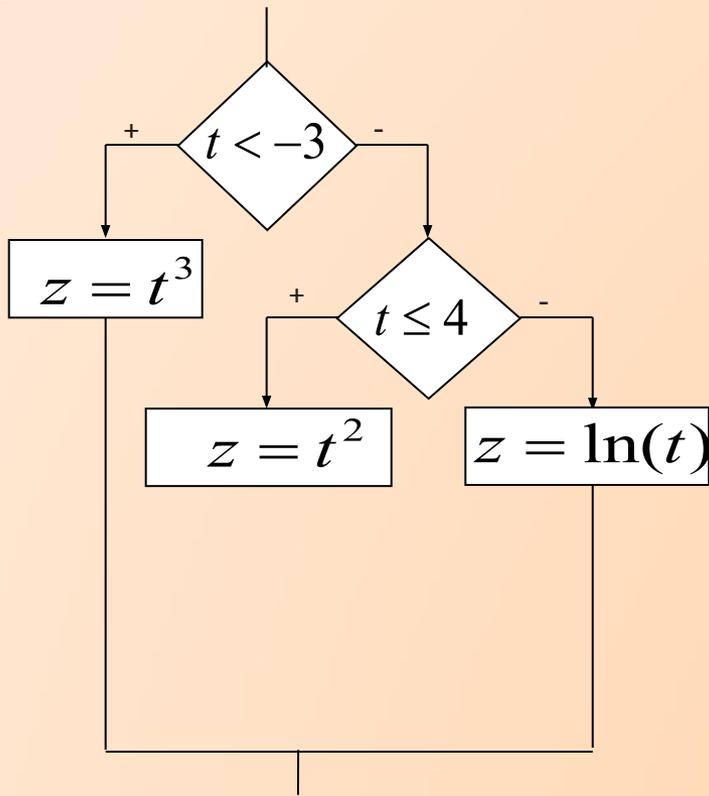
	if	
	otherwise	

поле для ввода **выражения**, значение которого используется, если значение логического выражения **false**

Данные операторы могут находиться только внутри тела программы-функции!

**Пример 4.** Написать программу-функцию, вычисляющую значение кусочно-заданной функции  $z$ :



$$z(t) = \begin{cases} t^3, & t < -3; \\ t^2, & -3 \leq t \leq 4; \\ \ln(t), & t > 4. \end{cases}$$

Предлагаются следующие варианты описания программы-функции:

1  $f(t) := \begin{cases} z \leftarrow t^3 & \text{if } t < -3 \\ z \leftarrow t^2 & \text{if } t \leq 4 \\ z \leftarrow \ln(t) & \text{otherwise} \\ z \end{cases} \quad f(-4) = 16$

2  $f(t) := \begin{cases} z \leftarrow t^3 & \text{if } t < -3 \\ \text{otherwise} \\ \begin{cases} z \leftarrow t^2 & \text{if } t \leq 4 \\ z \leftarrow \ln(t) & \text{otherwise} \end{cases} \\ z \end{cases} \quad f(-4) = -64$

Необходимо выбрать  
верный вариант

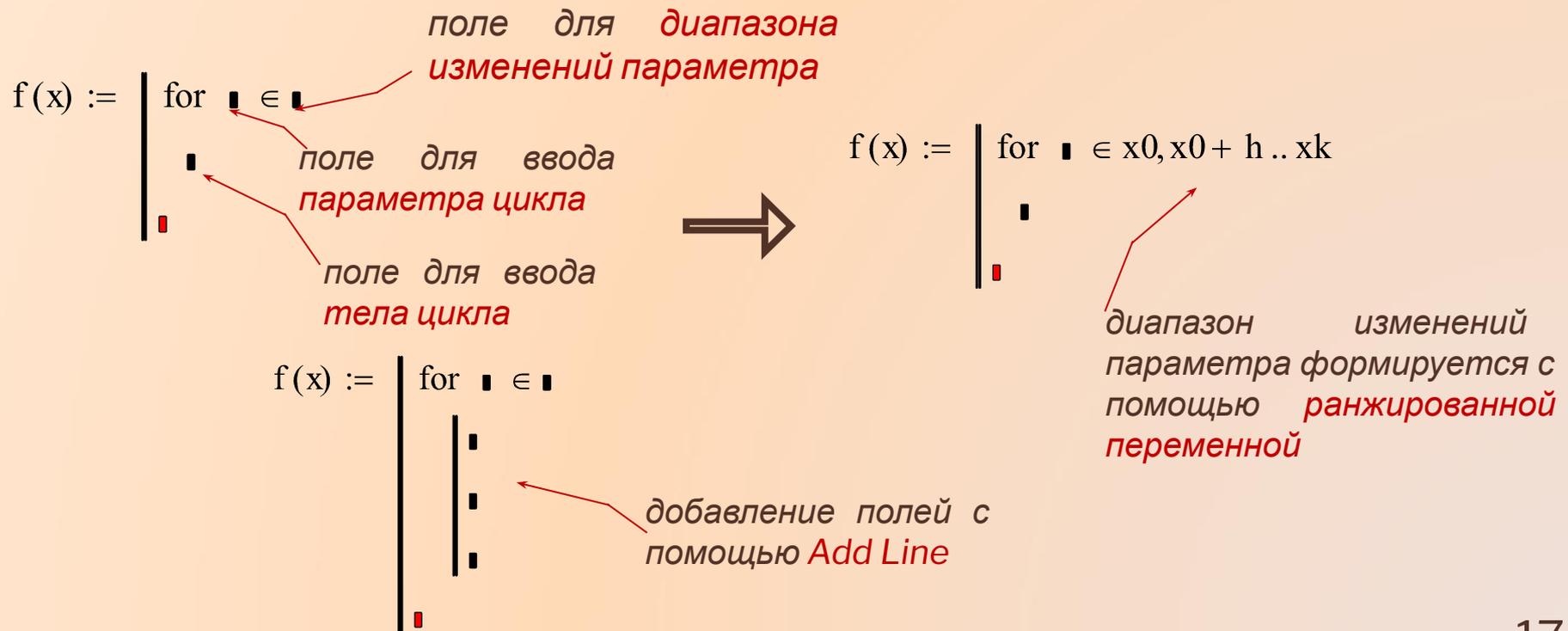


Циклический алгоритм (цикл) - алгоритмическая конструкция, которая содержит упорядоченную совокупность действий, повторяющихся многократно.

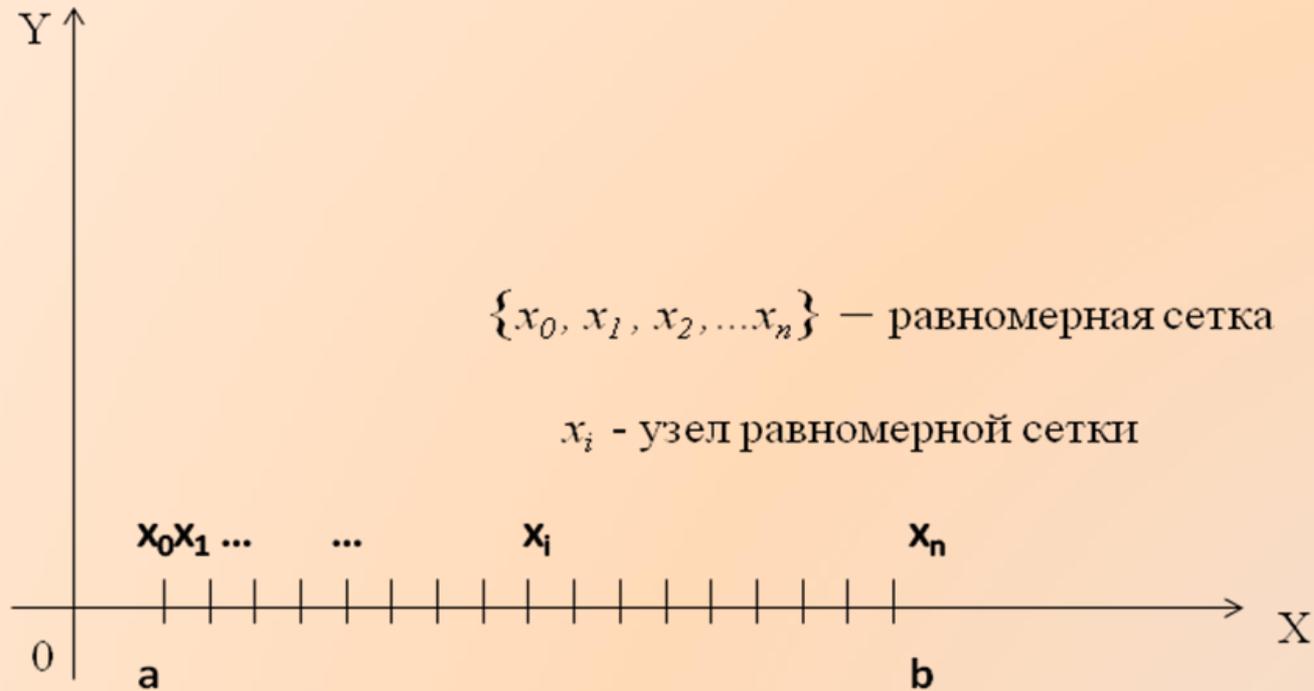
Циклы можно условно разделить на две группы:

- циклы типа арифметической прогрессии (с заданным числом повторений), реализуются *с помощью оператора for*;
- итерационные циклы, реализуются *с помощью оператора while*.

### Формат использования оператора for



**Пример 5.** Табулирование функции одной переменной  $y=f(x)$  на отрезке  $[a;b]$  с шагом  $h$ .



$$\begin{aligned} X_0 &= a \\ X_1 &= X_0 + h \\ X_2 &= X_1 + h \\ &\dots \\ X_i &= X_{i-1} + h \end{aligned}$$

Для решения задачи необходимо найти значения заданной функции в каждом узле получившейся совокупности точек и представить их в виде таблицы.

**Конкретизируем постановку задачи:** на отрезке от -2 до 2 с шагом 0,5 протабулировать функцию  $f(x) = e^{-x} \cdot \cos(2x)$ .

```

form_tab(x0,xk,d) :=
  i ← 1
  for x ∈ x0,x0+d..xk
    z ← e-x · cos(2·x)
    yi ← z
    i ← i + 1
  y
  
```

ORIGIN:= 1

<описание программы-функции>

form\_tab(-2,2,0.5) =

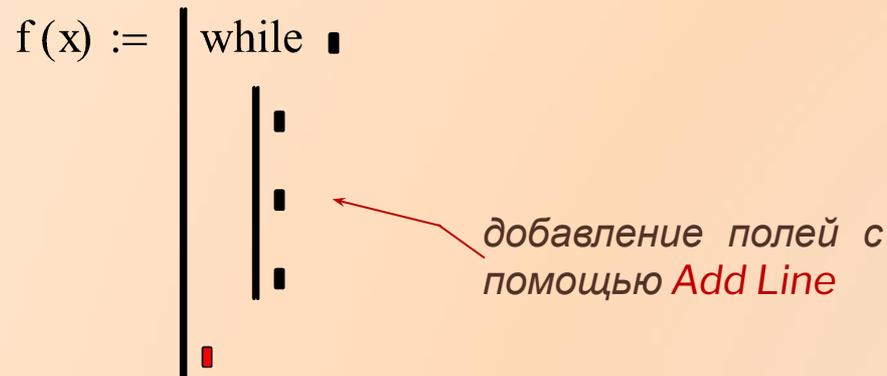
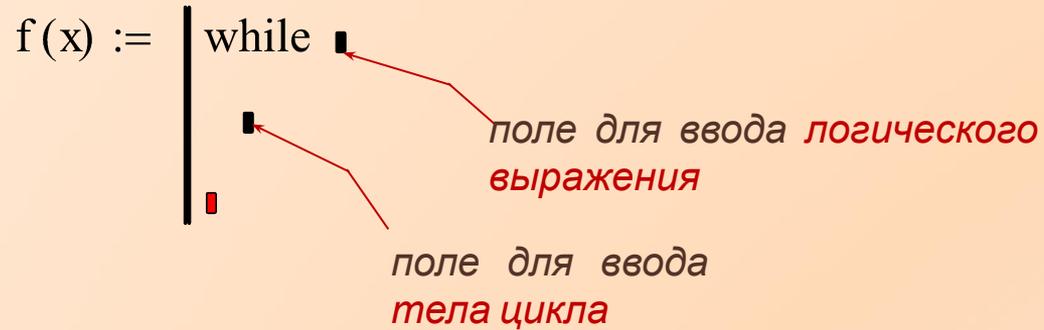
form\_tab(-2,2,0.5) =

-4.83
-4.437
-1.131
0.891
1
0.328
-0.153
-0.221
-0.088

	0
0	0
1	-4.83
2	-4.437
3	-1.131
4	0.891
5	1
6	0.328
7	-0.153
8	-0.221
9	-0.088

отображается значение  $y_0=0$ , т.к. значение системной переменной ORIGIN по умолчанию равно 0

## Формат использования оператора while:

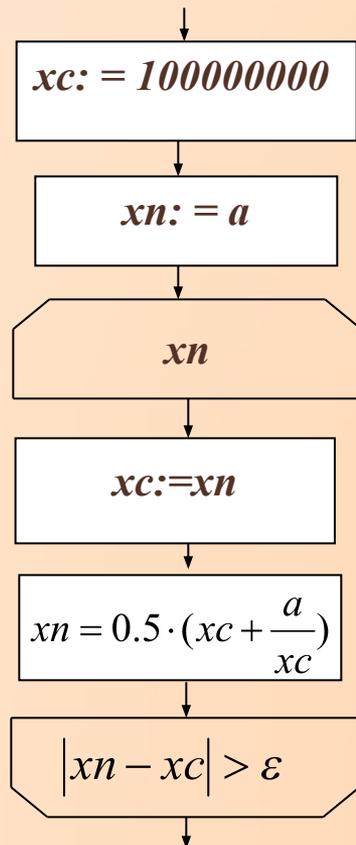


Для итерационных циклов нельзя априори определить количество повторений тела цикла. Это обусловлено тем, что окончание таких циклов определяется не выходом параметра цикла за конечное значение, а более сложными условиями.

**Пример 6.** Вычислить приближенное значение корня квадратного  $x = \sqrt{a}$ , используя итерационную процедуру:

$$x_n = 0.5 \cdot \left( x_{n-1} + \frac{a}{x_{n-1}} \right), \text{ где } n = 1, 2, 3, \dots, x_0 = a$$

Известно, что в качестве приближенного значения корня квадратного берется такое значение  $x_n$ , которое удовлетворяет условию  $|x_n - x_{n-1}| \leq \varepsilon$   
 $\varepsilon$  – заданная точность.



```

sqrt ( a, eps ) :=
  xc ← 10000000000
  xn ← a
  while |xn - xc| > eps
    xc ← xn
    xn ← (xc + a/xc) / 2
  xn
  
```

$$\text{sqrt} ( 9, 0.001 ) = 3 \qquad \text{sqrt} ( 49, 0.01 ) = 7$$

$$\text{sqrt} ( 123, 0.0001 ) = 11.091$$

$$\text{sqrt} ( -9, 0.001 ) = \blacksquare$$

## Формат использования оператора прерывания break:

$f(x) :=$  **break if**  $\left[ \begin{array}{l} \text{поле для ввода логического выражения} \\ \text{прерывание цикла, если значение логического выражения true} \end{array} \right.$

```

sqrt ( a, eps ) :=
  xc ← a
  for i ∈ 1.. 1000
    xn ←  $\frac{xc + \frac{a}{xc}}{2}$ 
    break if |xn - xc| ≤ eps
    xc ← xn
  ierr ←  $\begin{cases} 1 & \text{if } i = 1000 \\ 0 & \text{otherwise} \end{cases}$ 
  x0 ← xn
  x1 ← ierr
  x
  
```

поле для ввода **оператора**, выполняемого если значение лог. выражения **false**

$$\text{sqrt}(-9, 0.001) = \begin{pmatrix} -0.909 \\ 1 \end{pmatrix}$$

$$\text{sqrt}(25, 0.001) = \begin{pmatrix} 5 \\ 0 \end{pmatrix}$$

значение **ierr**

---

***Спасибо за внимание***