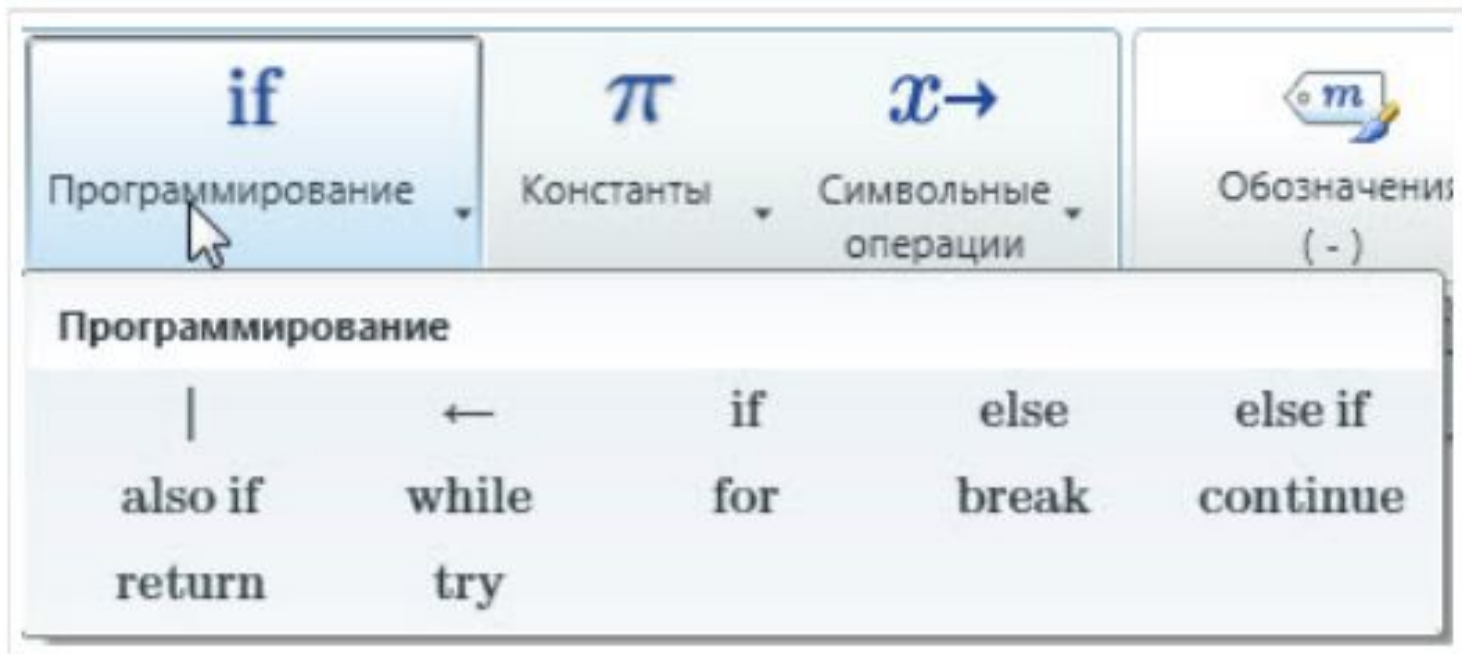


Программирование в Mathcad

Mathcad содержит встроенную среду программирования, что значительно расширяет возможности вычислительного пакета. Операторы программирования находятся в меню Математика → Операторы и символы → Программирование.



Команды:

ы:

«Программирование» – для создания программной структуры.

«Локальное назначение» – знак «равно» для программ.

«Оператор if»– оператор условия.

«Оператор else»– альтернативный выбор.

«Оператор return» – выход из программы.

«Try / On Error» – применяется, если при выполнении программы может возникнуть ошибка.

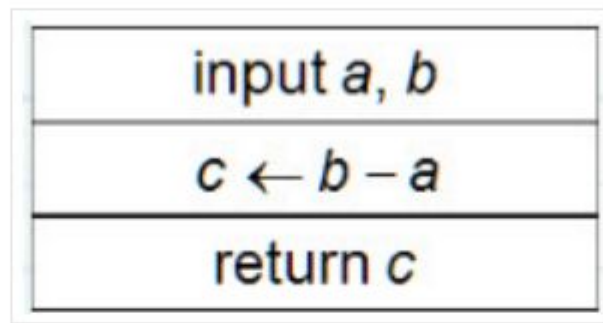
Эти команды можно сочетать с операторами, переменными и функциями Mathcad. Например, Вы можете использовать структуру функции для ввода входных значений:

$$R(a, b) := \begin{cases} c \leftarrow \sqrt{a^2 + b^2} \\ \text{return } c \end{cases} \quad \begin{aligned} R(3, 4) &= 5 \\ R(1, 2) &= 2.236 \end{aligned}$$

«Программирование» и «Локальное определение»

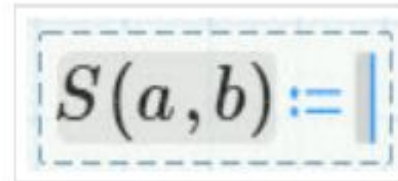
Создадим программу, которая вычисляет разницу между двумя переменными a и b .

На рисунке представлена структурная диаграмма программы:

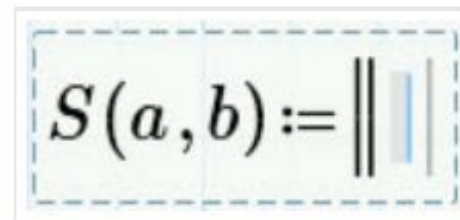


Здесь всего три элемента: **вход, действие и выход**. Вместо структурной диаграммы можно использовать другие способы, помогающие созданию программы, например, блок-схемы или псевдокод.

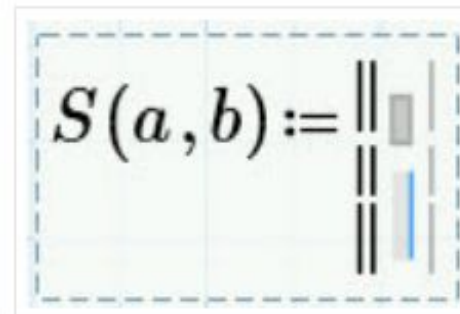
Определите ввод переменных a и b :


$$S(a, b) := |$$

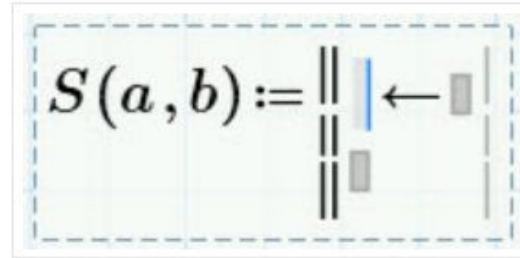
Нажмите оператор «Программирование» на панели Математика → Программирование:


$$S(a, b) := || |$$

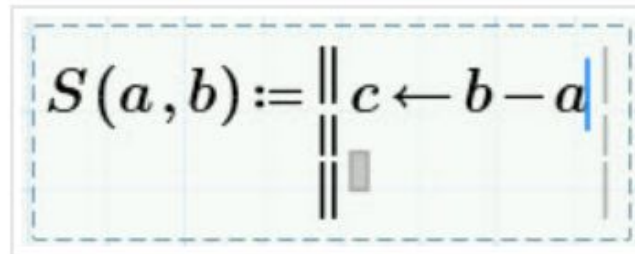
Нажмите [Enter] для создания второй строки:


$$S(a, b) := || |$$

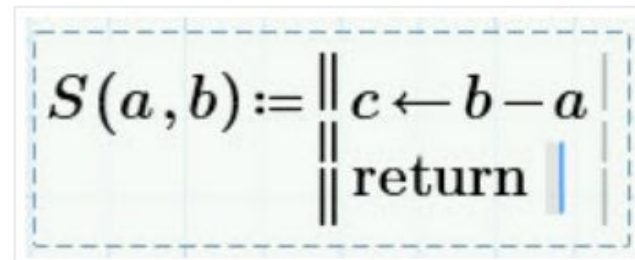
Переместите курсор в верхний местозаполнитель, затем вставьте оператор «Локальное назначение»:



Заполните местозаполнители слева и справа от оператора:



Переместите курсор в нижний местозаполнитель и вставьте оператор «return»:



Важно! Операторы
программирования `return`,
`else`, `if`, `while` и т.д. следует
вставлять из меню
программирования.
Ввод этих команд с
клавиатуры не приведет
к желаемому результату.

Введите переменную в местозаполнитель:

```
S(a, b) := || c ← b - a ||  
           || return c ||
```

Всегда тестируйте программы, потому что при некоторых значениях могут получиться бессмысленные результаты.

В некоторых случаях это могут быть отрицательные числа, ноль или бесконечность.

**У операторов программирования
есть свои горячие клавиши**

**«Программирование» – правая
квадратная скобка]**

**«Локальное назначение» – левая
фигурная скобка {**

«return»– [Ctrl+\\]

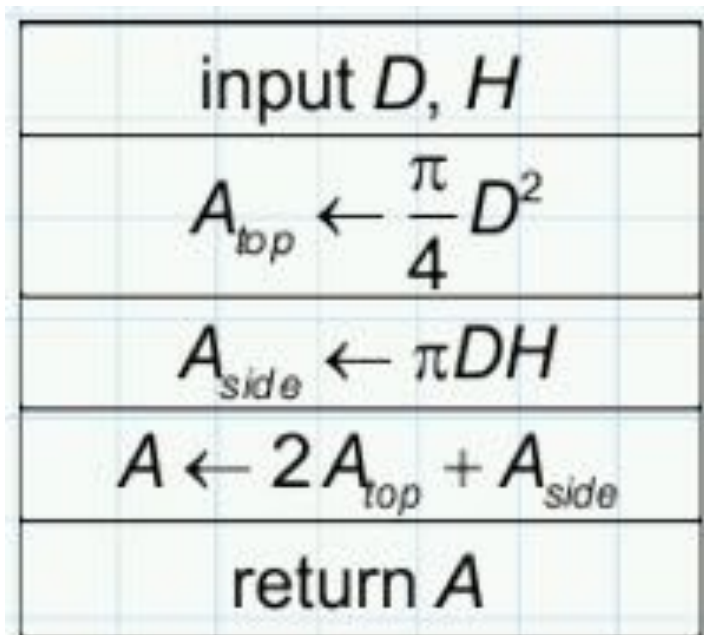
Программе не обязательно задавать входные переменные:

```
|| a ← 7   | = -3  
|| b ← 4   |  
|| c ← b - a |  
|| return c |
```

Оператор «return» также не обязателен – программа примет за выходное значение последнее вычисление:

```
S(a, b) := || c ← b - a |  
  
S(7, 3) = -4
```

Вычислим площадь поверхности цилиндра диаметром D и высотой H . Структурная диаграмма этой программы:



Программа и некоторые выходные результаты показаны ниже. Возможно, нужно запретить отрицательные входные значения...

$A(D, H) :=$	$\ A_{top} \leftarrow \frac{\pi}{4} \cdot D^2$	$A(2, 1) = 12.6$
	$\ A_{side} \leftarrow \pi \cdot D \cdot H$	$A(0, 3) = 0$
	$\ A \leftarrow 2 \cdot A_{top} + A_{side}$	
	$\ \text{return } A$	$A(-1, 2) = -4.712$

Заметьте, что переменные внутри программы являются локальными.

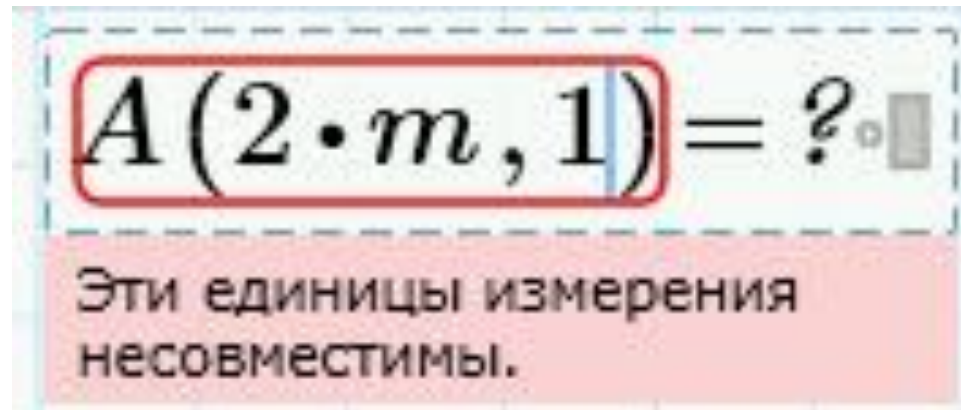
Локальная переменная не определяется вне программы

$A_{side} = ?$

Входным значениям можно дать числа с единицами измерения:

$$A(2 \cdot m, 1 \cdot m) = 12.6 m^2$$

Однако если задать единицу измерения только одной переменной, Mathcad скажет, что единицы не совместимы:


$$A(2 \cdot m, 1) = ?$$

Эти единицы измерения несовместимы.

Поэтому единицы измерения следует использовать единообразно.

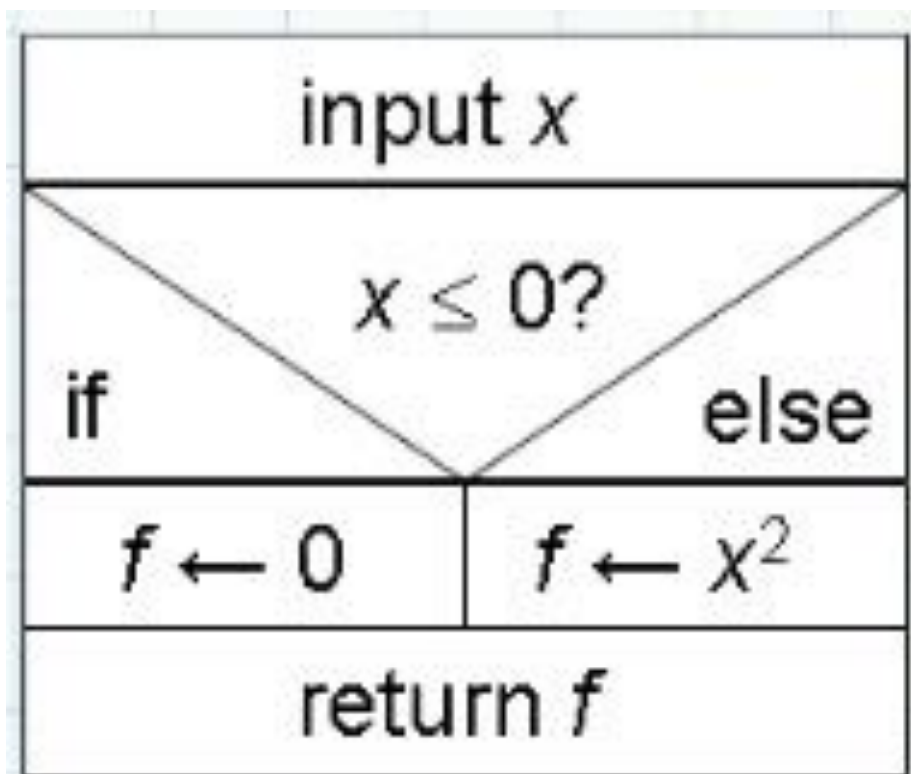
Операторы if и else

Необходимо создать программу функции, которая принимает следующие значения:

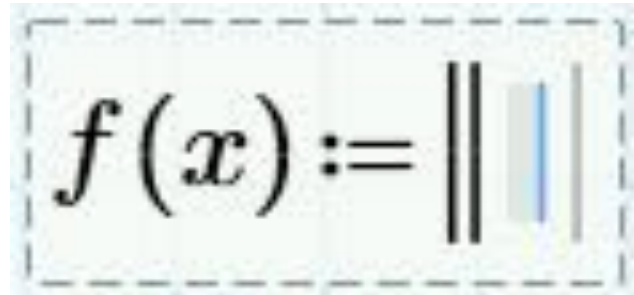
$$\begin{array}{ll} x \leq 0 & f(x) \leftarrow 0 \\ x > 0 & f(x) \leftarrow x^2 \end{array}$$

Операторы if и else

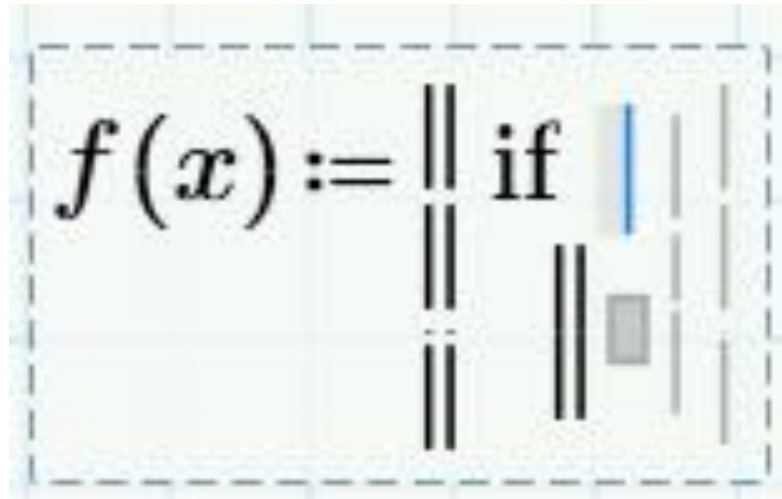
Структурная диаграмма программы представлена ниже. Треугольник означает выбор между двумя или более альтернативными вариантами. Напишем программу, используя операторы if и else.



Введите имя функции и местозаполнитель для первой строки:



Нажмите if в меню программирования или с помощью горячей клавиши }. Появится вторая строка, которая относится к оператору if:



Введите критерий выбора и желаемое значение функции.

Обратите внимание на серые линии

справа:

```
f(x) := || if x < 0
        ||
        || f ← 0
```

Нажмите на внутреннюю серую линию (станет мигающей синей), затем вставьте оператор else.

Появится еще одна строка, относящаяся к else:

```
f(x) := || if x < 0
        ||
        || f ← 0
        || else
        ||
```

Введите необходимую функцию под else.

Внутренняя серая линия удлинится, что указывает на то, что операторы if и else связаны между собой.

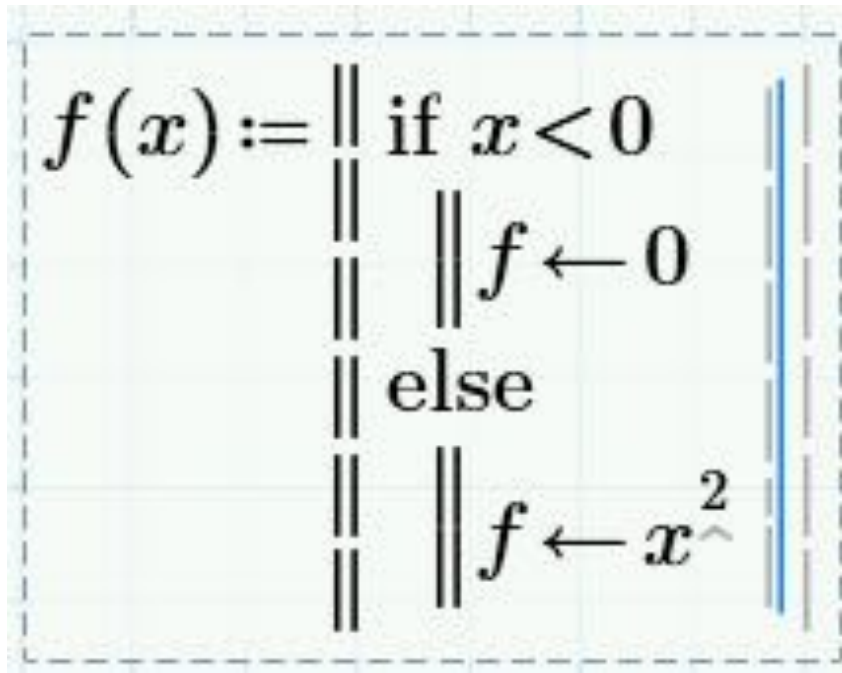
Нажмите на нее, нажмите [Enter], затем вставьте оператор return:

```
f(x) := || if x < 0  
||  
|| f ← 0  
|| else  
|| f ← x2  
|| return f
```

При работе с программой можно добавлять новые строки нажатием клавиши [Enter].

Где появится местозаполнитель, зависит от положения курсора.

Выбрана внутренняя серая линия:



```
f(x) := || if x < 0  
||  
|| || f ← 0  
||  
|| else  
||  
|| || f ← x^2  
||
```

The image shows a code editor window with a dashed border. Inside, a function definition is written in a monospaced font. The function is $f(x) :=$ followed by an if-else statement. The if-else statement is enclosed in a vertical bar structure. The first line is $f(x) :=$ followed by a vertical bar, then $if\ x < 0$. The second line is a vertical bar, then a vertical bar, then $f \leftarrow 0$. The third line is a vertical bar, then $else$. The fourth line is a vertical bar, then a vertical bar, then $f \leftarrow x^2$. The fifth line is a vertical bar. A blue vertical line is positioned to the right of the code, and a gray vertical line is positioned to the left of the code, indicating the current cursor position.

Функции в программах

Векторы и матрицы

Откройте Функции → Все функции и откройте раздел Векторы и матрицы.

Найдите функции `last()` и `length()`.

Это функции для определения некоторых свойств вектора:

$$v := \begin{bmatrix} 17 \\ -3 \\ 8 \\ 24 \end{bmatrix} \quad \text{length}(v) = 4$$
$$v_3 = 24 \quad \text{last}(v) = 3$$

Функция `length()` определяет длину вектора, т.е. количество элементов в нем, а функция `last()` выводит индекс последнего элемента. По умолчанию в Mathcad нумерация элементов вектора начинается с нуля, поэтому у четвертого элемента массива индекс

Теория чисел/комбинаторика

Наибольший общий делитель:

$$\gcd(2, 4, 7) = 1 \quad \gcd(12, 4) = 4$$

Наименьшее общее кратное:

$$\text{lcm}(2, 4, 7) = 28 \quad \text{lcm}(12, 4) = 12$$

Остаток от деления x на y :

$$\text{mod}(6, 3) = 0 \quad \text{mod}(7, 3) = 1 \quad \text{mod}(8, 3) = 2 \quad \text{mod}(9, 3) = 0$$

Строковые функции

Строки в Mathcad заключаются в двойные кавычки:

```
“Это строка.”
```

Строки можно задавать в качестве переменных, но их нельзя использовать в вычислениях. (Строку, содержащую только числа, можно преобразовать в константу.)

Конкатенация строк:

```
a := “Это”    b := “ ”    c := “строка.”  
concat(a, b, c) = “Это строка.”
```

Длина строки (включая пробелы):

```
strlen(a) = 3
```

Строки могут быть полезны для задания в программах сообщений об ошибках.

Усечение и

Наименьшее целое число, большее ...

$$\text{ceil}(5.73) = 6 \quad \text{ceil}(-2.81) = -2$$

Наибольшее целое число, меньшее ...

$$\text{floor}(5.73) = 5 \quad \text{floor}(-3.81) = -4$$

Округление:

$$\text{round}(\pi, 2) = 3.14 \quad \text{round}(173, -2) = 200$$

Усечение

е:

$$\text{trunc}(5.73) = 5 \quad \text{trunc}(-3.81) = -3$$

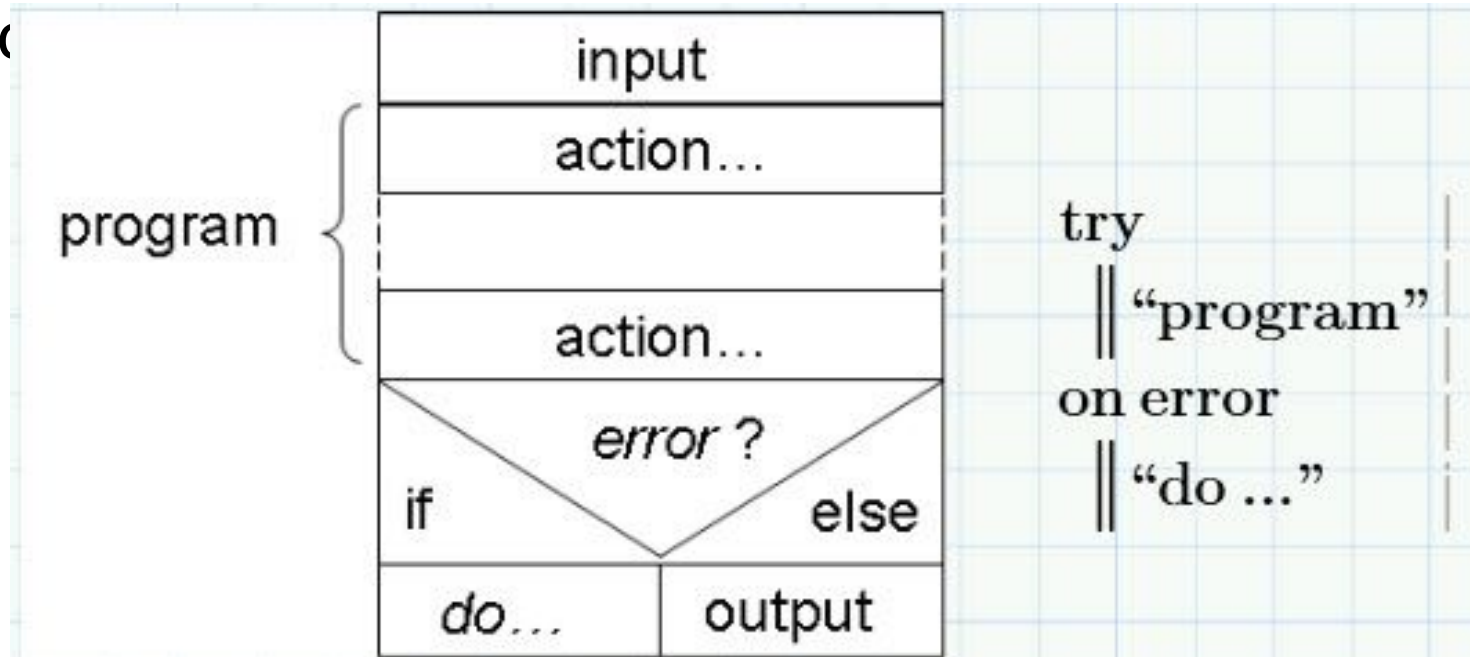
Подведя указатель мыши к имени функции в списке, Вы увидите ее полное название и краткое описание.

Если Вы вставите функцию в рабочую область, а затем нажмете [F1], Вы получите расширенное описание функции

Try / On Error

Последняя команда, которую мы изучим в этом уроке, используется для указания, что должно быть сделано, если при выполнении программы возникает ошибка (например, деление на ноль).

Если при выполнении программы в блоке `try` возникает ошибка, программа выполняет действия в блоке `on error`.



программа с тремя
операторами if внутри блока
try:

```
 $f(x) := \text{try}$   
  || || if  $x < 0$   
  || ||   ||  $f \leftarrow 0$   
  || || if  $0 \leq x < 1$   
  || ||   ||  $f \leftarrow x^2$   
  || || if  $1 \leq x$   
  || ||   ||  $f \leftarrow 1$   
  || || return  $f$   
  
on error  
  || return “wrong input”
```

При неверном вводе появится сообщение об ошибке. Таким образом, можно отследить большую часть ошибок, но не все:

$$f(2) = 1$$

$$f(\sqrt{-1}) = \text{"wrong input"}$$

$$f(\text{abc}) = ?$$

Эта переменная не определена. Проверьте правильность установки обозначения.

Поскольку переменная abc не определена, функция не вычисляется.

Резюме

Мы изучили следующие элементы программирования:

1. Входные данные – обычно вводятся как параметры функции.
2. Первая строка программы – вводится с помощью `]`. Больше линий – с помощью `[Enter]`.
3. Оператор локального определения – вводится с помощью `{`.
4. В конструкциях выбора с помощью оператора `if` применяются операторы сравнения.
5. `If` вводится с помощью `}`. За `if` вводится логическое выражение, например `x<0`. Под оператором записывается алгоритм, который должен быть выполнен, если выражение после `if` верно.

Резюме

1. После if может следовать оператор else или другой оператор if.
2. [Enter] добавляет новую строку в программу. Место появления новой строки зависит от позиции курсора до нажатия на [Enter].
3. Вывод переменной осуществляется с помощью оператора return. Переменной может быть одиночная переменная, вектор или матрица, которые могут содержать как числовые значения, так и текст.
4. Mathcad содержит большое число встроенных функций, которые могут быть полезны при написании программ. Список функций с подсказками можно открыть по команде Функции³⁰ →

Программирование в Mathcad

—

ЦИКЛЫ

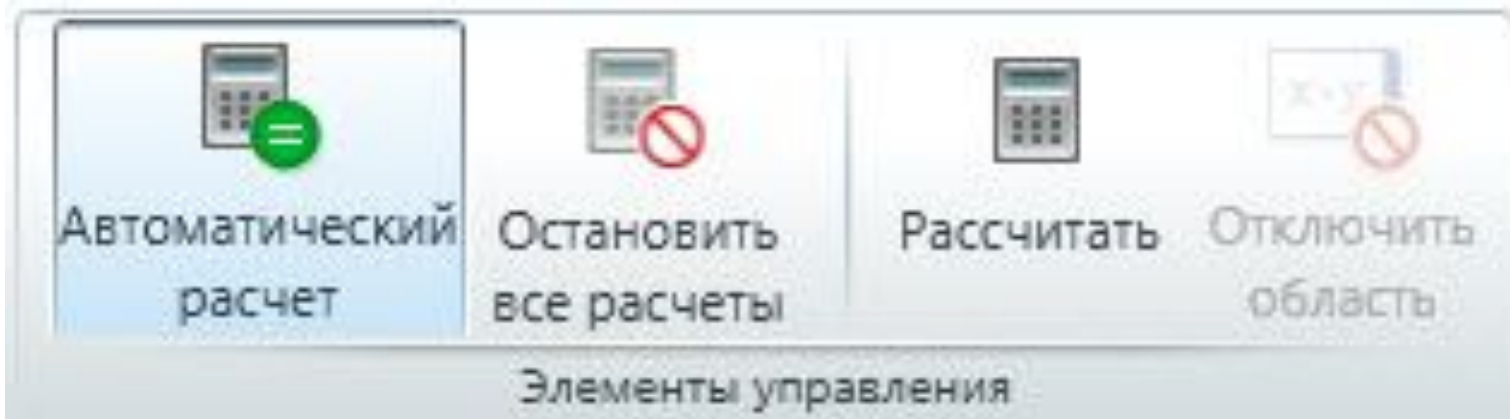
КОМАНДЫ:

`for` – для циклов `for`.

`while` – для циклов `while`.

`return` – как команду для отслеживания ошибок.

Операции Mathcad, которые мы рассматривали ранее, достаточно безвредны – они не заставят Ваш компьютер «зависнуть». Но с циклами это не так. Поэтому для начала следует изучить команды в меню



Кнопка «Автоматический расчет» обычно включена. Она отключается при нажатии на кнопку «Остановить все расчеты». Зеленый индикатор в левом нижнем углу становится серым. «Остановить все расчеты» служит для прекращения всех расчетов в документе на случай, если что-то пошло не так. При автоматическом расчете вычисления производятся лишь в том случае, когда происходят какие-либо изменения. С помощью кнопки «Рассчитать» можно сделать пересчет всего документа. Кнопка «Отключить область» прекращает вычисления в тех математических областях, которые Вы выбрали.

Циклы for

Циклы for применяются, когда заранее известно число повторений вычислений.

Программа ниже формирует вектор из $n+1$ элементов. Значения начинаются с нуля и имеют шаг 1.

введите n (целое число)
вычислить $n + 1$ элементов
вектора N со значениями,
равными значению счетчика
вернуть вектор N

```
input  $n$ 
for  $i \in 0..n$ 
   $N_i \leftarrow i$ 
return  $N$ 
```

Составим

программу

Задайте имя программы-функции, вставьте программную структуру (вертикальная линия) и определение цикла for из меню Математика → Операторы и символы → Программирование или с помощью сочетания клавиш [Ctrl+Shift+”]:

```
f(n) := || for | ∈ | |
```

Определите имя переменной-счетчика:

```
f(n) := || for i | ∈ | |
```

Определите диапазон счетчика:

```
f(n) := || for i ∈ 0 .. n | |
```

Введите команды тела цикла и оператор return:

```
f(n) := || for i ∈ 0..n ||  
        ||   || N ← i ||  
        ||   ||   □i ||  
        || return N ||
```

Проверим
программу:

$$f(2) = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \quad f(2.9) = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \quad f(0) = [0] \quad f(-1) = ?$$

Как видно, использование дробных или отрицательных чисел – не лучшая идея.

Вы можете изменить точку начала, но этого лучше избегать:

```
f(a, b) := || for i ∈ a..b |  
           ||   || N_i ← i |  
           ||   ||  
           || return N |
```

$$f(2, 4) = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Видите нули, которые появились в начале вектора? Причиной появления этих нулей является то, что если не определить некоторые элементы вектора, то им автоматически присво

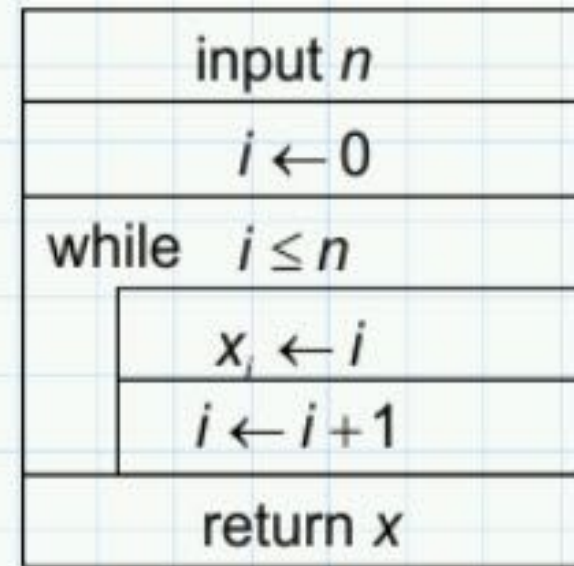
0:

```
y_3 := 6  
  
y = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 6 \end{bmatrix}
```

While делает то же самое, что и предыдущий цикл for

```
f(n) := || i ← 0  
|| while i ≤ n  
||   || xi ← i  
||   || i ← i + 1  
|| return x
```

$f(3) = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}$



- До цикла необходимо создать строку, содержащую определение начального значения счетчика.
- Следующую строку можно прочитать как «Выполнять цикл, пока соблюдается условие $i \leq n$ ».
- После определения элемента вектора нужно задать команду на увеличение переменной-счетчика, так как в цикле while это не происходит автоматически.

пример цикла `while` вычисляет экспоненту отрицательного числа, используя разложения в ряд:

$$\exp(-x) = \sum_{k=0}^{\infty} \frac{(-1)^k \cdot x^k}{k!}$$

- Суммирование будем производить с помощью цикла `while`.
- Будем проверять, насколько изменяется общая сумма S при каждом увеличении k .
- Если абсолютное значение этого изменения достаточно мало, цикл завершится.

Чтобы начать цикл, необходимо определить первые два элемента век

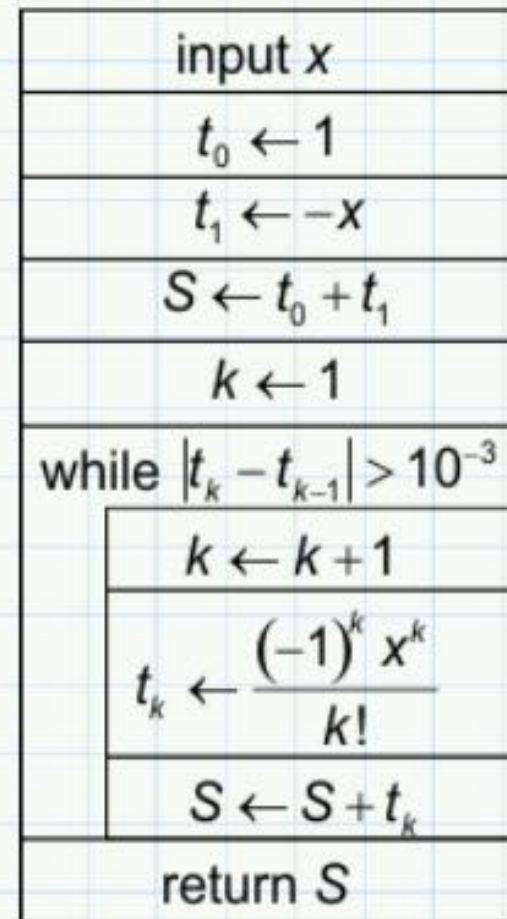
$$t_0 = 1 \quad t_1 = -x$$

Кроме того, мы определили начальное значение суммы S и счетчика k .
Дальнейшие вычисления производятся в цикле:

```

 $E_-(x) :=$ 
||  $t_0 \leftarrow 1$ 
||  $t_1 \leftarrow -x$ 
||  $S \leftarrow t_0 + t_1$ 
||  $k \leftarrow 1$ 
|| while  $|t_k - t_{k-1}| > 10^{-3}$ 
||   ||  $k \leftarrow k + 1$ 
||   ||  $t_k \leftarrow \frac{(-1)^k \cdot x^k}{k!}$ 
||   ||  $S \leftarrow S + t_k$ 
||   ||
|| return  $S$ 

```



Проверк

а:

$$E_(5) = 0.007$$

$$E_(1) = 0.368$$

$$E_(-1) = 2$$

$$\exp(-5) = 0.007$$

$$\exp(-1) = 0.368$$

$$\exp(1) = 2.718$$

С положительными числами программа работает хорошо, но для работы с отрицательными она не предназначена.

Отладк

Одна из простых технологий отладки программ – вывод промежуточных результатов вычислений и их сравнение с тем, какие значения должны быть.

Пример на цикле while:

$$f(n) := \begin{array}{|l} \parallel i \leftarrow 2 \\ \parallel \text{while } i \leq n \\ \parallel \quad \parallel x_i \leftarrow i \\ \parallel \quad \parallel i \leftarrow i + 1 \\ \parallel \text{return } x \end{array} \quad f(3) = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 3 \end{bmatrix}$$

Число элементов вектора верное, но второй элемент неправильный.

Похоже, что последние элементы нашего вектора получили правильный индекс, а второй

Отладк

а Мы можем проверить, какой индекс получил второй элемент, вставив «return i» в первую строку цикла while.

Программа остановит вычисление и вернет значение i:

```
n := 3    i1 := || i ← 2 |           i1 = 2    должно быть 1
              || while i ≤ n | | |
              ||   || return i |
              ||   || xi ← i |
              ||   || i ← i + 1 |
              || return x |
```

Ошибка в первой строке.

Замените 2 на 1, удалите дополнительную строку return, и получите верный результат.

После отладки всегда
следует удалять
дополнительные строки,
которые Вы вводите, так как
программа всегда
прекращает работу после
первого оператора `return`.

Резюме

1. Отключайте вычисления при написании или редактировании программы (Вычисления → Остановить все расчеты).
2. Цикл `for` – повторяет вычисления определенное количество раз. Цикл `for` обязательно включает в себя счетчик и число повторений.
3. Цикл `while` – выполняется, пока соблюдается определенное условие. Начальное значение счетчика следует задавать до цикла. В теле цикла изменение счетчика задается вручную.
4. Программы почти всегда требуют отладки. Полезная команда для этого – `return`, с помощью которой можно вывести промежуточное значение.
5. Важно подготовиться к написанию программы, например, составив структурную диаграмму до написания непосредственно кода.