11. ПОДПРОГРАММА-ФУНКЦИЯ: ОПИСАНИЕ И ВЫЗОВ

Описание подпрограммы-функции (П-Ф) и локальный оператор присваивания

Описание П-Ф размещается в рабочем документе перед ее вызовом и включает в себя имя подпрограммы-функции, список формальных параметров (который может отсутствовать) и тело подпрограммы-функции. Для ввода конструкций в тело П-Ф используется палитра инструментов **ПРОГРАММИРОВАНИЕ**, приведенная на рис. 1.

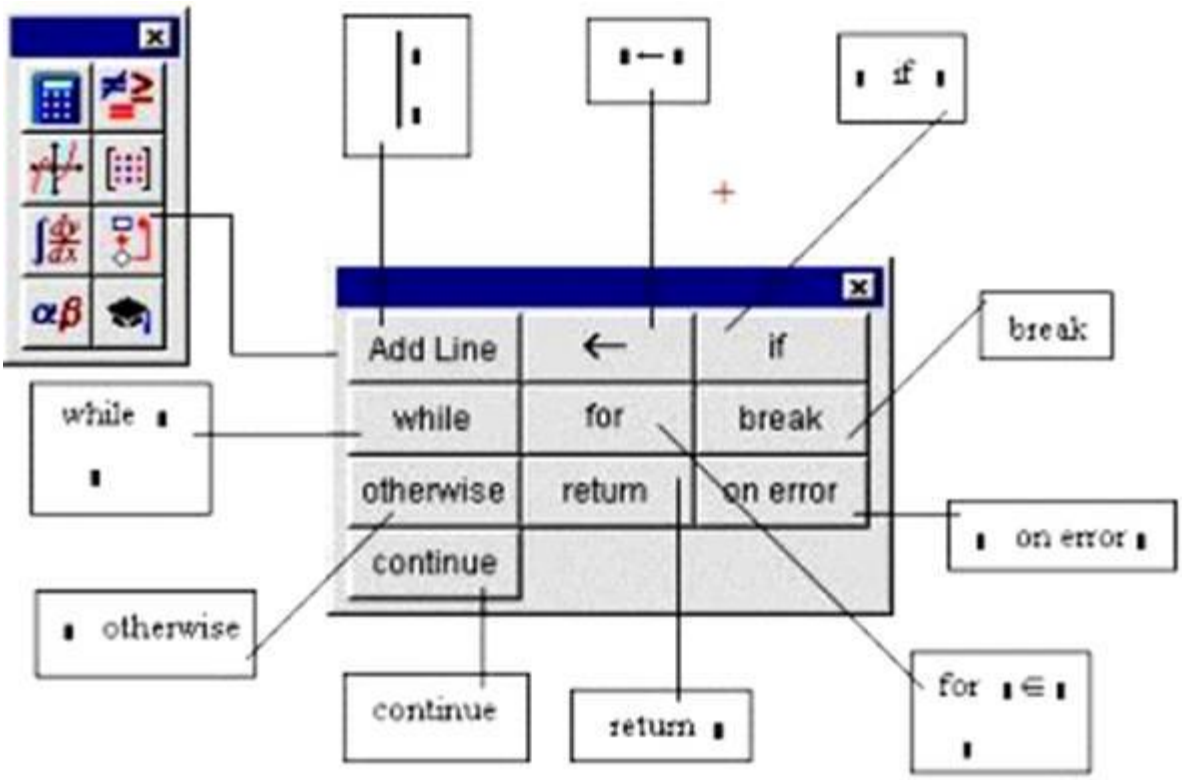


Рис. 1. Палитра **ПРОГРАММИРОВАНИЕ**

Каждая П-Ф MathCAD имеет *оригинальное имя*, посредством которого осуществляется обращение к ней. Через это же имя (***и только через это имя***) «возвращается» результат выполнения П-Ф.

После имени П-Ф идет *список формальных параметров*, заключенный в круглые скобки. Через формальные параметры «внутрь» П-Ф «передаются» данные, необходимые для выполнения вычислений внутри программы, т.е. ***все формальные параметры являются входными***.

В качестве формальных параметров могут использоваться имена простых переменных, массивов и функций. Формальные параметры отделяются друг от друга запятой.

Замечание 1. П-Ф ***может не иметь формальных параметров,*** и тогда данные передаются через имена переменных, определенных выше описания П-Ф.

Тело подпрограммы-функции включает любое число операторов: локальных операторов присваивания, условных операторов и операторов цикла, а также вызов других П-Ф и функций пользователя.

Порядок описания подпрограммы-функции MathCAD.

Для ввода в рабочий документ описания П-Ф необходимо выполнить следующие действия:

- ввести имя П-Ф и список формальных параметров, заключенный в круглые скобки
- ввести символ ":" – на экране отображается как ": =";
- открыть палитру **ПРОГРАММИРОВАНИЯ** и щелкнуть кнопкой Add line (см. рис. 1). На экране появится вертикальная черта и вертикальный столбец с двумя полями для ввода операторов, образующих тело П-Ф (см. рис. 2);

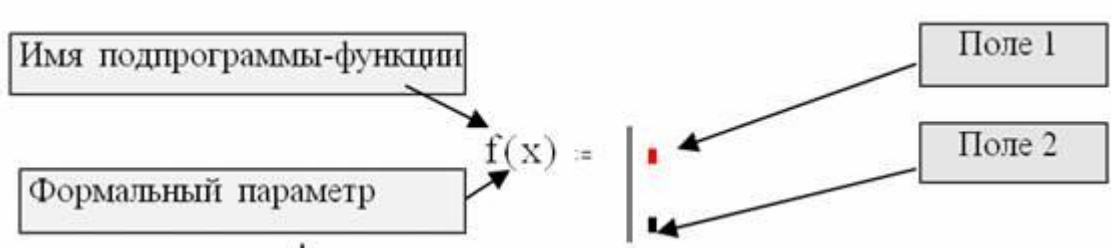


Рис. 2. Структура подпрограммы-функции

- перейти в поле 1 (щелкнув на нем мышью или нажав клавишу [Tab]) и ввести первый оператор тела П-Ф.
- затем ввести второй, третий и т.д. операторы, добавляя пустые поля с помощью щелчка на кнопке Add line палитры **ПРОГРАММИРОВАНИЕ**.

- заполнить самое нижнее поле ввода, введя туда выражение, определяющее возвращаемое через имя П-Ф (см. рис. 3).

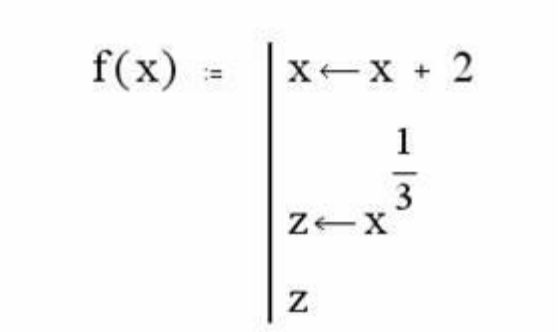


Рис. 3. Окончательная структура подпрограммы-функции

Замечание 2. Если результатом работы П-Ф являются несколько величин, то из них в теле П-Ф необходимо сформировать массив и его имя поместить в последней строке тела П-Ф.

Локальный оператор присваивания

Для задания внутри программы значения какой-либо переменной используется так называемый ***локальный оператор присваивания***, имеющий вид:

< имя переменной > ← < выражение > .

Обращение к подпрограмме-функции MathCAD

Для выполнения П-Ф необходимо обратиться к ее имени с указанием ***списка фактических параметров*** (если в описании программы присутствует список формальных параметров), т.е.:

< имя П-Ф > (< список фактических параметров >) .

Фактические параметры отделяются друг от друга запятой.

 Очевидно, что между фактическими и формальными параметрами должно быть ***соответствие по количеству, порядку следования и типу***.

Замечание 3. Обращение к П-Ф должно находиться после ее описания, и ***к моменту обращения фактические параметры должны быть определены.***

Пример 4. Обращение к программе *f(x),* приведенной на рис. 4.1.4, может иметь следующий вид:

$x := 2 \qquad f(x) = 1.587 \qquad f(-3.23) = 0.536 + 0.928i$

$z := f(x + 2.5) \qquad z = 1.866$

Заметим, что переменная *z* никак не связана с «локальной» переменной *z*, используемой внутри тела П-Ф.

Замечание 4. Передать данные внутрь П-Ф можно, используя внутри подпрограммы переменные, определенные до описания П-Ф (см. пример на рис. 4.2.1).

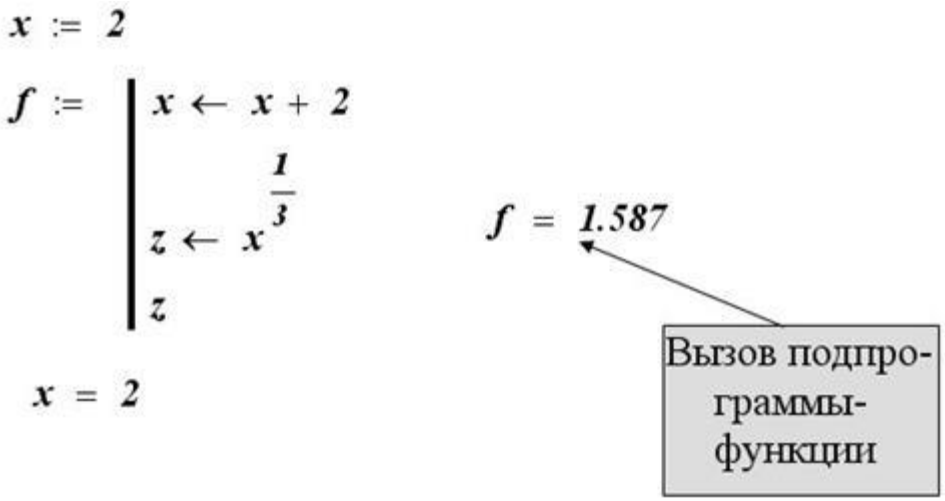


Рис. 4. Подпрограмма-функция без формальных параметров

Программирование АЛГОРИТМОВ в подпрограмме-функции MathCAD

Программирование линейных алгоритмов в подпрограмме-функции

Операторы, реализующие этот алгоритм, в теле П-Ф также размещаются последовательно и выполняются все, начиная с первого и заканчивая последним.

Пример 6. Оформим в виде П-Ф вычисление корней квадратного уравнения $ax^2 + bx + c = 0$ по формуле

$$x_{1,2} = \frac{-b \mp (b^2 - 4ac)^{1/2}}{2a}.$$

Описание П-Ф *root_poly2* и обращение к ней приведено на рис. 5.5.1. П-Ф имеет три входных формальных параметра – коэффициенты квадратного уравнения. Выходом является вектор с двумя компонентами. Заметим, что величины x_1, x_2 являются простыми переменными, а не элементами одномерного массива. Поэтому нижние индексы в их именах вводятся после нажатия клавиши [.] – «десятичная точка».

Подпрограмма-функция реализует ***линейный алгоритм – все операторы выполняются всегда строго последовательно.***

$$root\_poly2(a,b,c) := \left\| \begin{array}{l} d \leftarrow \sqrt{b^2 - 4 \cdot a \cdot c} \\ x_1 \leftarrow \dfrac{-b + d}{2 \cdot a} \\ x_2 \leftarrow \dfrac{-b - d}{2 \cdot a} \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \end{array} \right.$$

$$z := root\_poly2(12,7,10) \qquad z = \begin{pmatrix} -0.292 + 0.865i \\ -0.292 - 0.865i \end{pmatrix}$$

Пример программирования линейного алгоритма

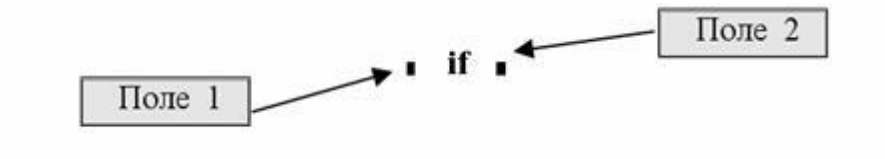
Программирование разветвляющихся алгоритмов в подпрограмме-функции

Для программирования разветвляющихся алгоритмов в подпрограмме-функции MathCAD можно использовать:

- *условную функции*
- *условный оператор if*.

Используя эти конструкции, можно «изменить» последовательное выполнение операторов.

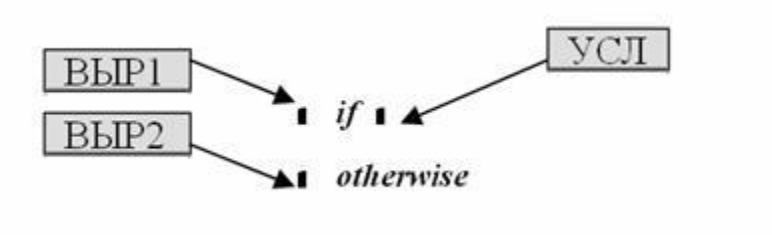
Условный оператор. Этот оператор используется ***только в теле П-Ф*** и для его ввода необходимо щелкнуть на кнопке **if** палитры **ПРОГРАММИРОВАНИЕ**. На экране появляется конструкция с двумя полями ввода, изображенная на рисунке.



Структура условного оператора *if*

В поле 2 вводится логическое выражение УСЛ (в простейшем случае это выражение отношений). В поле 1 вводится конструкция ВЫР1, которая выполняется, если проверяемое логическое выражение принимает значение 1. Если УСЛ = 0, то ВЫР1 не выполняется. Это соответствует условной структуре, называемой **ЕСЛИ – ТО**.

Для получения условной структуры **ЕСЛИ – ТО – ИНАЧЕ** используется оператор ***otherwise***, вводимый с палитры **ПРОГРАММИРОВАНИЕ,** в поле которого размещается конструкция ВЫР2, которая выполняется, если проверяемое логическое выражение принимает значение 0 (см. рисунок). Оператор *otherwise* непосредственно следует после условного оператора *if*.



Реализация структуры **ЕСЛИ – ТО – ИНАЧЕ**

Для ввода ВЫР2 в поле оператора *otherwise* необходимо:

- выделить поле, стоящее после оператора *if*;
- щелкнуть на кнопке otherwise палитры **ПРОГРАММИРОВАНИЕ**;
- в появившееся поле оператора *otherwise* ввести необходимую конструкцию ВЫР2.

Пример 6. Составим описание П-Ф, вычисляющей функцию *y(x),* заданную выражением

$$y = \begin{cases} x^2, & \textit{если } x \le 0; \\ \sqrt{x}, & \textit{в противном случае}. \end{cases}$$

Описание и вызов П-Ф приведены на рисунке

$$y(x) := \begin{cases} x^2 & \text{if } x \le 0 \\ \sqrt{x} & \text{otherwise} \end{cases}$$

$y(2) = 1.414 \qquad y(-4) = 16$

Реализация разветвляющегося алгоритма

Пример 7. Составим описание П-Ф для вычисления переменной *z(t)* по формуле

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

Описание П-Ф и ее вызов приведены на рисунке

$$z(t) := \begin{cases} t^3 & \text{if } t \le -3 \\ t^2 & \text{if } -3 \le t \le 4 \\ \ln(t) & \text{otherwise} \end{cases}$$

$z(-8) = -512 \qquad z(2) = 4 \qquad z(7) = 1.946$

Реализация разветвляющегося алгоритма

Из описания видно, что функция *z(t)* получит значение *ln(t)* только тогда, ***когда не выполняются условия, записанные в двух вышестоящих строках тела П-Ф.***

Внимание! Если в строке 3 ввести просто *ln(t)*, то это выражение ***будет вычисляться всегда*** вне зависимости от выполнения заданных выше условных операторов.

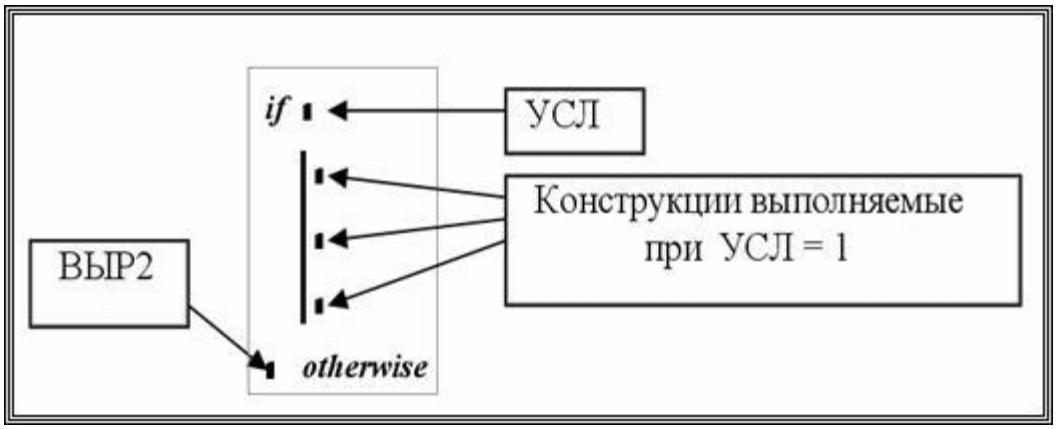
Задание 2. Составьте описания П-Ф, реализующих следующие разветвляющиеся алгоритмы:

$$y(x)=\begin{cases}\dfrac{1}{2\sqrt{x}}, \text{если } x \ge 1;\\ \dfrac{1}{3}\sqrt[3]{x}, \text{если } 0 \le x \le 1;\\ \dfrac{1}{4}\sqrt[4]{|x|}, \text{если } x < 0.\end{cases}$$

$$\mu(x,\varepsilon)=\begin{cases}\dfrac{1}{2\sqrt{x+1}}, \text{если } |x-y| < \varepsilon,\\ \dfrac{1}{3}\sqrt[3]{x+y}, \text{если } |x-y| \ge \varepsilon,\end{cases}$$

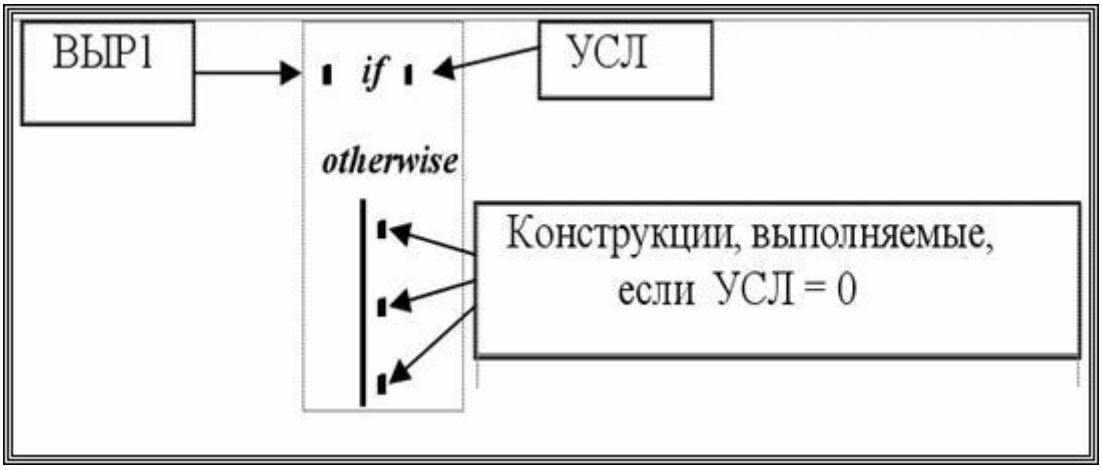
где $y=|x|$.

Вариант 1. ***При выполнении заданного условия УСЛ необходимо выполнить несколько конструкций MathCAD.***



Вариант 2. ***При невыполнении заданного условия УСЛ необходимо выполнить несколько конструкций MathCAD.***

В этом случае необходимо выделить *поле* оператора *otherwise*, щелкнуть на кнопке Add line палитры **ПРОГРАММИРОВАНИЕ** нужное число раз и заполнить появившиеся поля.

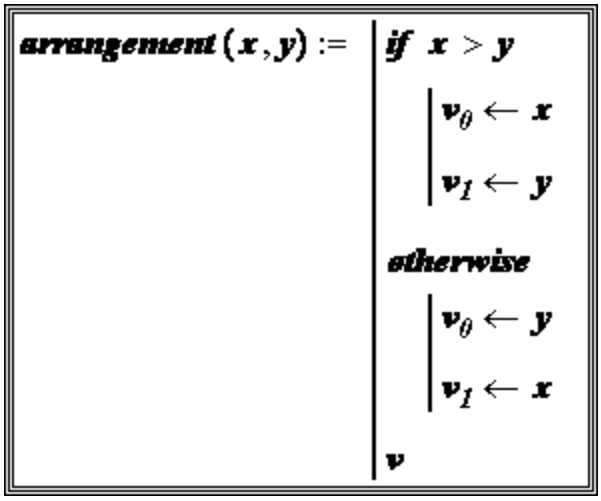


Пример 8. Составьте описание П-Ф, вычисляющей значения двух полиномов *x(t), y(t)* нулевой или первой степени. Порядок полиномов задается переменной *n*. Если $n < 0$ или $n > 1$, то значения полиномов равны 0. Описание П-Ф приведено на рисунке.

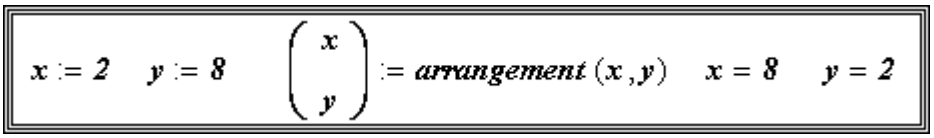
$$
\begin{array}{l|l}
poly(n, a, b, t) := & \textbf{if } n = 0 \\
 & \quad x \leftarrow a_0 \\
 & \quad y \leftarrow b_0 \\
 & \textbf{if } n = 1 \\
 & \quad x \leftarrow a_0 + a_1 \cdot t \\
 & \quad y \leftarrow b_0 + b_1 \cdot t \\
 & \textbf{otherwise} \\
 & \quad x \leftarrow 0 \\
 & \quad y \leftarrow 0 \\
 & \begin{pmatrix} x \\ y \end{pmatrix}
\end{array}
$$

Реализация алгоритма примера

Пример 9. Даны два числа *x, y*. Составить описание П-Ф, которая переменной *x* присваивает максимальное значение из этих двух чисел, а *y* – минимальное. Описание приведено на рисунке.



Реализация разветвляющегося алгоритма



Вызов подпрограммы-функции *arrangement*

Задание 3. Даны три числа *a, b, c*. Составить П-Ф, реализующую следующий алгоритм. Если $a \le b \le c$, то все числа заменить их квадратами, если $a > b > c$, то каждое число заменить максимальным значением из этих трех чисел, в противном случае — сменить знаки у чисел.

Задание 4. Координаты точки на плоскости задаются двумя числами *x, y*. Составить П-Ф, вычисляющую номер четверти на плоскости, в которую попала точка.

Задание 5. Длина сторон треугольника задается числами *a, b, c*. Составить П-Ф, вычисляющую значение целой переменной *n* по следующему правилу: *n* = 3, если три стороны равны; *n* = 2, если любые две стороны равны; *n* = 1, если все три стороны имеют разную длину.

Программирование циклических алгоритмов в подпрограмме-функции

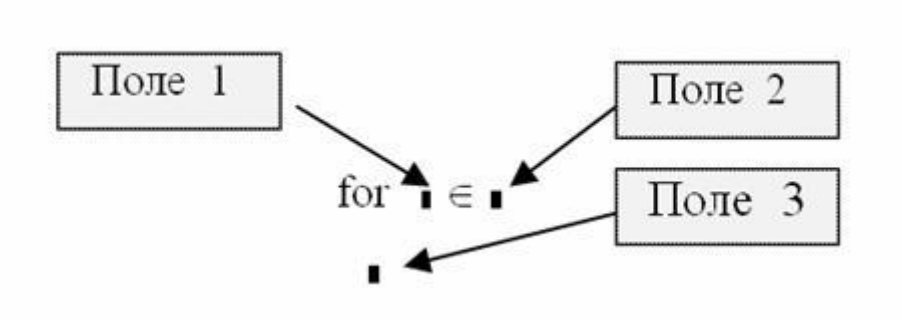
Циклы можно условно разделить на две группы:

- *циклы типа арифметической прогрессии;*
- *итерационные циклы.*

Программирование цикла типа арифметической прогрессии

Для программирования таких циклов используется оператор цикла *for* (часто называемый оператором цикла с параметром). Для ввода такого оператора необходимо выполнить следующие действия:

- щелкнуть на кнопке for палитры **ПРОГРАММИРОВАНИЕ**. На экране появятся поля ввода, изображенные на рисунке



Поля оператора цикла ***for***

- в поле ввода 1 ввести имя переменной, являющейся параметром цикла;
- в поле 2 — закон изменения параметра цикла, используя для этого описание дискретной переменной или *описание массива*;
- в поле 3 — операторы, составляющие тело цикла. Если одной строки недостаточно, то дополнительные поля ввода (дополнительные строки) создаются щелчком на кнопке Add line палитры **ПРОГРАММИРОВАНИЕ,** и тогда слева от тела цикла появляется вертикальная черта, охватывающая тело цикла.

Пример 9. Составить описание П-Ф, реализующей алгоритм формирование вектора.

Заметим, что значение системной переменной *ORIGIN* (начальное значение индексного выражения) задается равным 1.

$$form\_vec1(n) := \left| \begin{array}{l} \text{for } i \in 1..n \\ \quad z_i \leftarrow \dfrac{1}{i^2 + 1} \\ z \end{array} \right. \qquad form\_vec1(5) = \begin{pmatrix} 0.5 \\ 0.2 \\ 0.1 \\ 0.059 \\ 0.038 \end{pmatrix}$$

Подпрограмма-функция формирования вектора

Пример 9. Для *x* меняющегося от -2 до 2 с шагом 0.5 вычислить значение $f(x) = e^{-x} \cdot Cos(2x)$ и сформировать из этих значений вектор y, т.е. $y_1 = f(-2)$, $y_2 = f(-1.5)$ и т.д.

В этом примере количество повторений тела цикла определяется по формуле

$$\left[\frac{x_k - x_0}{d} \right] + 1,$$

где x_k, x_0 – конечное и начальное значения параметра цикла, d – шаг его изменения. Подставив значения, получаем (2–(–2))/0.5+1=9.

Следовательно, сформированный вектор *y* будет содержать 9 элементов.

Описание П-Ф и ее вызов приведены на рисунке. Видно, что в теле цикла выполняется два оператора. Первый оператор формирует элемент массива *y*, а второй изменяет на 1 значение индекса.

$$form\_vec2(x_0, x_k, d) := \left| \begin{array}{l} i \leftarrow 1 \\ \text{for } x \in x_0, x_0 + d .. x_k \\ \quad \left| \begin{array}{l} y_i \leftarrow e^{-x} \cdot \cos(2 \cdot x) \\ i \leftarrow i + 1 \end{array} \right. \\ y \end{array} \right.$$

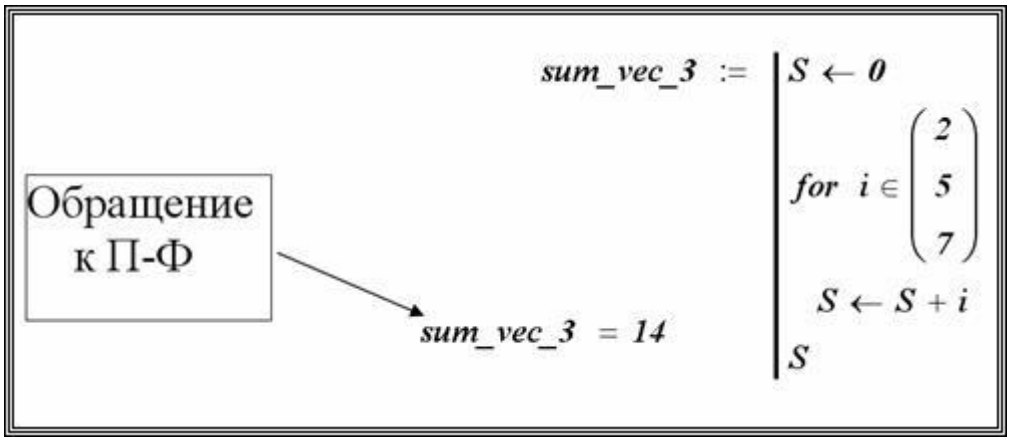
$$z := form\_vec2(0, 0.8, 0.1)$$

$$z^T = (\ 1 \quad 0.887 \quad 0.754 \quad 0.611 \quad 0.467 \quad 0.328 \quad 0.199 \quad 0.084 \quad -0.013\ )$$

Формирование вектора примера 9

Пример 10. Составить описание П-Ф, где значения параметра цикла задаются вектором.

На рисунке приведено описание такой П-Ф.

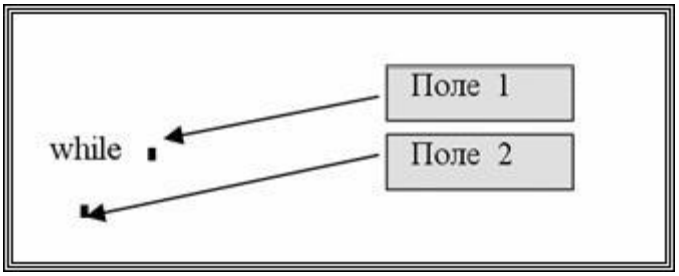


Задание 5. Составьте описание П-Ф формирования вектора *y* примера 9, приняв в качестве параметра цикла переменную *i.*

Программирование итерационных циклов

Для программирования таких циклов используется оператор цикла *while*. Для ввода этого оператора необходимо выполнить следующие действия:

- щелкнуть на кнопке ***while*** палитры **ПРОГРАММИРОВАНИЕ.** На экране появляются элементы, показанные на рисунке



Структура оператора цикла *while*

- в поле 1 ввести условие выполнения цикла;
- в поле 2 ввести операторы тела цикла. В теле цикла должны присутствовать операторы, которые *могут изменить значение условия цикла*, иначе цикл будет продолжаться бесконечно.

Оператор цикла *while* **выполняется следующим образом**: обнаружив оператор while, MathCAD проверяет указанное в операторе условие. Если оно равно 1 (т.е. выполняется), то выполняется тело цикла, и снова проверяется условие. Если условие принимает значение 0, то цикл заканчивается.

Пример 11. Составим П-Ф, реализующую итерационную процедуру приближенного вычисления корня квадратного $\sqrt{a}$, используя итерационную процедуру:

$$x_{n+1} = 0.5 \cdot (x_n + \frac{a}{x_n}),\ n = 0,\ 1,\ \dots\ ;\ x_0 = a.$$

В качестве приближенного значения $\sqrt{a}$ принимается x_{n+1} , удовлетворяющее условию:

$$\left| x_{n+1}^2 - a \right| \le \varepsilon ,$$

где ε – заданная точность вычисления корня квадратного.

Нет необходимости хранить в памяти все приближенные решения $x_0, x_1, x_2, \dots$ Достаточно хранить предыдущее («старое») значение (обозначим его как *xc*) и последующее («новое») значение *xn*.

```
sqroot(a, ε) := | xn ← a
                | while |xn² − a| > ε
                |     | xc ← xn
                |     | xn ← (xc + a/xc) / 2
                | xn
```

$$sqroot(9, 0.1) = 3.00009$$

$$sqroot(9, 0.0001) = 3.00000$$

К сожалению, организация итерационного цикла с помощью оператора *while* без дополнительных средств контроля может привести к зацикливанию, т.е. повторению тела цикла «бесконечное» число раз. Например, задав при обращении к П-Ф $\varepsilon < 0$, получаем зацикливание.

Поэтому в MathCAD имеется специальный оператор *break*, который позволяет выйти из цикла или приостановить исполнение программы при выполнении заданного в операторе *break* условия.

Оператор *break* используется в левом поле ввода условного оператор *if*, а в правом размещается условие, при выполнении которого происходит прекращение работы цикла или программы. Поэтому *первоначально вводится оператор if***,** а затем заполняются поля этого оператора. Следующий пример показывает написание подпрограммы без «зацикливания» с использованием оператора *break*.

Пример 12.1. Составим П-Ф, реализующую итерационную процедуру вычисления корня квадратного без «зацикливания»**.**

$$\begin{aligned}
&sqroot_new(a, \varepsilon) := \\
&\quad xn \leftarrow a \\
&\quad ierr \leftarrow 1 \\
&\quad \text{for } i \in 1..10000 \\
&\quad\quad \text{if } |xn^2 - a| < \varepsilon \\
&\quad\quad\quad ierr \leftarrow 0 \\
&\quad\quad\quad \text{break} \\
&\quad\quad xc \leftarrow xn \\
&\quad\quad xn \leftarrow \frac{xc + \frac{a}{xc}}{2} \\
&\quad xn \leftarrow \text{"not solve"} \quad \text{if } ierr = 1 \\
&\quad \begin{pmatrix} xn \\ ierr \end{pmatrix}
\end{aligned}$$

$$sqroot\_new(9, 0.0001) = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

$$sqroot\_new(9, -0.0001) = \begin{pmatrix} "not\ solve" \\ 1 \end{pmatrix}$$

Рис. 5.3.7. Реализация итерационного цикла без «зацикливания»

Пример 12.2 Составить П-Ф, осуществляющую суммирование ряда с бесконечным числом слагаемых. Накопление суммы прекращается, как только очередное слагаемое по абсолютной величине становится меньше заданной погрешности ε.

Описание П-Ф и ее вызов показаны на рисунке. Заметим, что вторым формальным параметром является имя функции пользователя, определяющей зависимость величины члена ряда от его номера. При вызове этот формальный параметр заменяется фактическим – именем функции пользователя, описанной до обращения к П-Ф.

$$b(i) := \frac{1}{i^2}$$

$$d(i) := \frac{1}{\sqrt{i}}$$

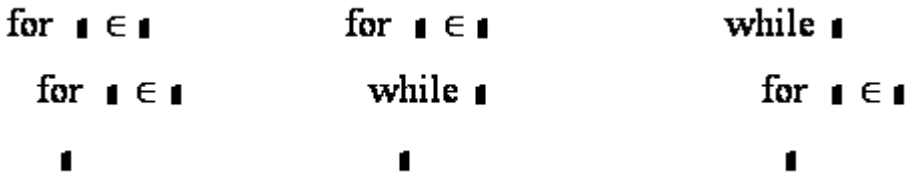
```
sum_conv(ε, a) :=
    sum ← 0
    ierr ← 1
    for i ∈ 1..10000
        if |a(i)| < ε
            ierr ← 0
            break
        sum ← sum + a(i)
    sum ← "not convergence"  if ierr = 1
    (sum)
    (ierr)
```

Вызовы подпрограммы-функции

$$sum\_conv(0.00001, b) = \begin{pmatrix} 1.64177 \\ 0 \end{pmatrix},$$

$$sum\_conv(0.00001, d) = \begin{pmatrix} "not\ convergence" \\ 1 \end{pmatrix}.$$

Программирование двойных циклов



Варианты вложений операторов цикла

Составить описание П-Ф формирующей матрицу по следующему правилу:

$$B_{i,j} = \frac{1}{i+j+1}, \quad i = 1,\ldots,n, \quad j = 1,\ldots,m$$

.

Описание и вызов П-Ф приведены на рисунке. В этой П-Ф параметром внешнего цикла является переменная *i*, а параметром внутреннего — переменная *j*.

$$ORIGIN := 1$$

$$form\_mat1(n,m) := \left| \begin{array}{l} \textbf{for } i \in 1..n \\ \quad \textbf{for } j \in 1..m \\ \quad\quad B_{i,j} \leftarrow \dfrac{1}{i+j+1} \\ B \end{array} \right.$$

$$form\_mat1(4,2) = \begin{pmatrix} 0.333 & 0.25 \\ 0.25 & 0.2 \\ 0.2 & 0.167 \\ 0.167 & 0.143 \end{pmatrix}$$

Реализация двойного цикла

Дополнительные операторы, используемые при программировании циклов

Оператор *continue*. Обычно используется для продолжения выполнения цикла путем возврата в начало тела цикла. Следующий пример поясняет работу этого оператора.

Пример 12.3 Составить описание П-Ф, формирующей новый вектор из положительных проекций исходного вектора.

Описание приведено на рисунке

$$form\_vec2(v) := \left| \begin{array}{l} i \leftarrow 0 \\ j \leftarrow 0 \\ \text{while } i < last(v) \\ \quad \left| \begin{array}{l} i \leftarrow i + 1 \\ \text{continue if } v_i \leq 0 \\ j \leftarrow j + 1 \\ w_j \leftarrow v_i \end{array} \right. \\ w \end{array} \right.$$

$$x := \begin{pmatrix} 2 \\ -4 \\ 3 \\ 2 \\ -11 \end{pmatrix}$$

$$z := form\_vec2(x) \qquad z = \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix}$$

Оператор *return*. Прерывает выполнение П-Ф и возвращает значение операнда, стоящего в поле 1 (см. рисунок).

Структура оператора ***return***

Пример 13. Составить описание П-Ф, находящей первую положительную проекцию исходного вектора. Возможны два варианта программной реализации алгоритма решения этой задачи. Вариант В представляется более простым и «элегантным».

$$pol\_1(v) := \left| \begin{array}{l} i \leftarrow 1 \\ \text{while } v_i \leq 0 \\ \quad i \leftarrow i + 1 \\ v_i \end{array} \right.$$

Вариант А

$$pol\_2(v) := \text{for } i \in 1..last(v)$$
$$\text{return } v_i \text{ if } v_i > 0$$

Вариант В

Оператор *on error*. Этот оператор является обработчиком возникающих при выполнении тех или иных вычислений ошибок и записывается в виде:

< *конструкция 1* > on error < *конструкция 2* >.

Оператор выполняется следующим образом. Если при выполнении <*конструкция 2*> возникает ошибка, то выполняется <конструкция 1>. Если ошибка не возникает, то выполняется <конструкция 2>.

Пример 14. Используем оператор *on error* для предотвращения появления ошибки «деление на ноль» при вычислении функции *angl(x,y).* Фрагмент представлен на рисунке

$$angl(x,y) := \frac{x}{y} \qquad angl(2,0) = \text{Ошибка «деление на ноль»}$$

$$angl(x,y) := 10^{10} \text{ on error } \frac{x}{y}$$

$$angl(2,0) = 1 \times 10^{10} \qquad angl(2,2) = 1$$

Функция error. Используется для вывода диагностических сообщений при возникновении в вычислениях ошибки и записывается в виде:

error ("< *диагностическое сообщение* >").

Имя функции вводится с клавиатуры. Функция используется в левом поле условного оператора *if***,** как показано в следующем примере.

Пример 16. Запрограммируем вывод диагностического сообщения при попытке спроецировать вектор *v* на нулевой вектор *w*. Описание П-Ф и ее вызовы приведены на рисунке.

$$proj(v,w) := \begin{cases} error("You\ \ cannot\ onto\ the\ 0\ vector") & if\ |w| < 10^{-10} \\ \frac{w}{|w|} \cdot (v \cdot w) & otherwise \end{cases}$$

$$x := \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \qquad w := \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} \qquad proj(x,w) = \begin{pmatrix} 16.036 \\ 10.69 \\ 5.345 \end{pmatrix}$$

$$x := \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \qquad w := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

proj(x,w) = ▮▮

You cannot onto the 0 vector