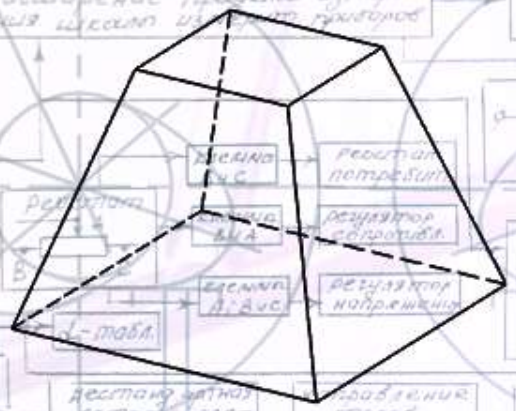
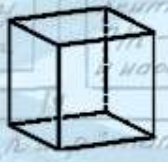
Лекция 4
Элементы программирования в Mathcad

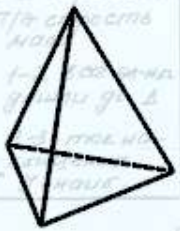


Лектор

Ст. преподаватель Купо А.Н.



1. Структура программных блоков в системе Mathcad. Палитра «Программирование» и ее элементы.
2. Правила применения программных операторов Mathcad.
3. Программирование алгоритмов обработки векторов и матриц.
4. Логические переменные и функции



С точки зрения программирования вообще, MathCAD–программа представляет собой **подпрограмму–функцию**, которая может возвращать в качестве результата **число, вектор или матрицу.**

MathCAD–программа создается в виде **программного модуля.** Это самостоятельный блок, который отличает жирная вертикальная черта.

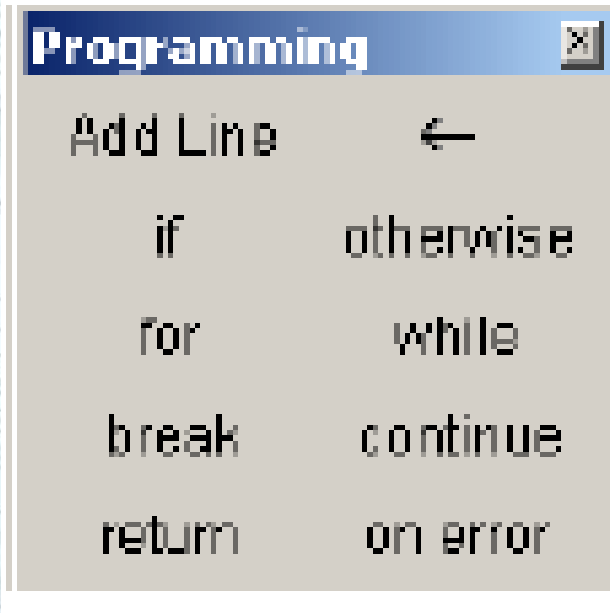
Программный модуль может существовать в рабочем документе как функция без имени.

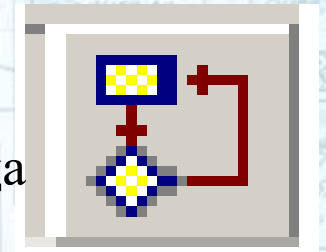
$$S(a,b,c) := \left| \begin{array}{l} P \leftarrow \dfrac{(a+b+c)}{2} \\ S \leftarrow \sqrt{P\cdot(P-a)(P-b)\cdot(P-c)} \end{array} \right.$$

$$S(2,8,7) = 6.437 \qquad S(5,4,3) = 6$$

Пример программного модуля «Вычисление площади треугольника по формуле Герона»

Для создания MathCAD–программы следует воспользоваться панелью инструментов **Programming**, которая вызывается кнопкой математической палитры.

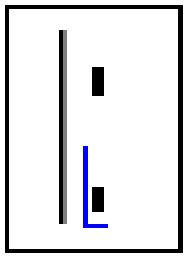


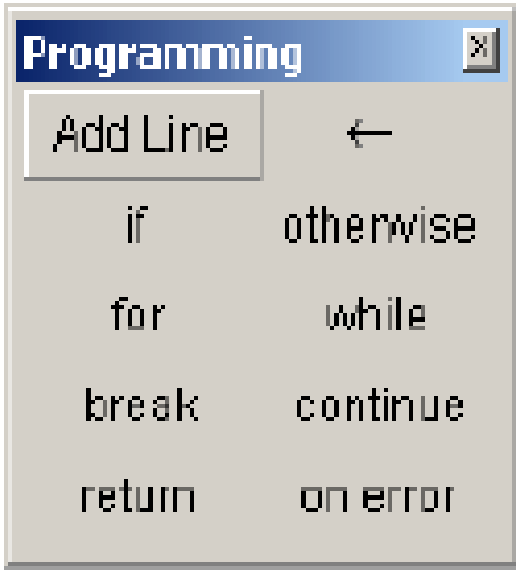


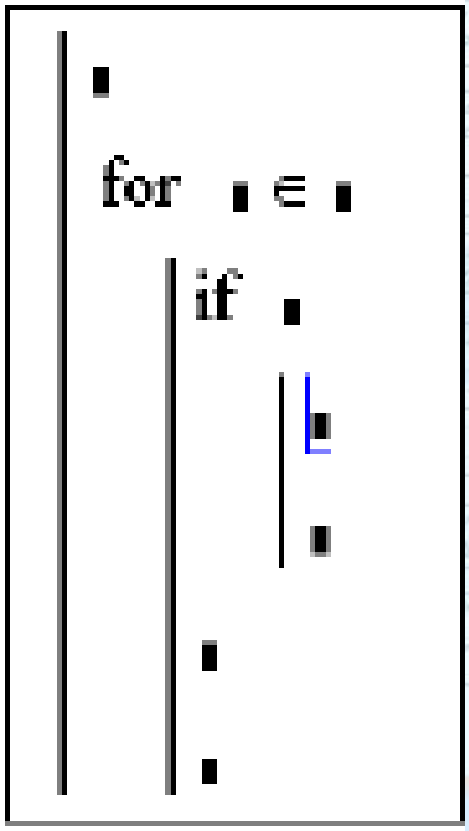
- **Add Line** – создает блок, для ввода команд MathCAD–программы;
- ← – присваивание;
- **if** – условный оператор;
- **otherwise** – оператор альтернативного выбора, применяется вместе с условным;
- **for** – цикл с известным числом повторений;
- **while** – условный цикл;
- **break** – прерывание;
- **continue** – продолжение;
- **return** – возврат;
- **on error** – обработка ошибок.

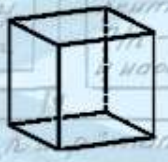
Создание программного модуля

Оператор **Add Line** вставляет в рабочий документ конструкцию, показанную. Блок ограничен жирной вертикальной линией, справа от которой расположены поля ввода команд программного модуля. Расширить программный блок можно, добавляя новые поля ввода, повторным вызовом оператора **Add Line**, но уже внутри программного модуля. Причем расширять блок можно не только в длину, но и создавать всевозможные вложенные структуры, например, такие как на рисунке.









Оператор, обозначенный на панели инструментов Programming стрелкой ←, выполняет **операцию присваивания**.

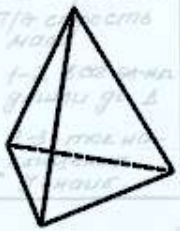
Условный оператор применяется, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие. Условный оператор имеет следующую структуру: *оператор* **if** *условие*, где *оператор* это действие, которое выполняется, в случае если *условие* истинно. Если *условие* ложно, управление передается следующему за **if** *оператору*.

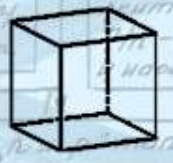
if Конструкция условного оператора

$$F(a,b) := \begin{vmatrix} c \leftarrow 0 \\ c \leftarrow a + b \ \text{if} \ a < b \\ c \leftarrow c + 12 \end{vmatrix}$$

$F(3,6) = 21$

$F(8,1) = 12$





Оператор альтернативного выбора применяется, если необходимо запрограммировать условную конструкцию вида

Оператор1 **if** *условие*

Оператор2 **otherwise**.

Конструкция оператора

if

otherwise

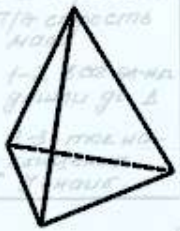
$$F(a,b) := \begin{cases} \text{"a>b"} & \text{if } a > b \\ \text{"a<b"} & \text{otherwise} \end{cases}$$

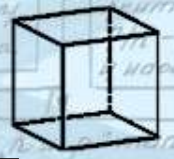
$F(3,7) = \text{"a<b"}$

$F(8,1) = \text{"a>b"}$

$$F(a,b) := \begin{cases} \text{"a>b"} & \text{if } a > b \\ \text{otherwise} & \\ \quad \begin{cases} \text{"a<b"} & \text{if } a < b \\ \text{"a=b"} & \text{otherwise} \end{cases} & \end{cases}$$

$F(3,7) = \text{"a<b"}$ $F(8,1) = \text{"a>b"}$ $F(1,1) = \text{"a=b"}$





Цикл с известным числом повторений

Оператор **for** предоставляет возможность организовать цикл по некоторой переменной, изменяющейся в заданном диапазоне.

for *переменная* ∈ *начальное значение*, *шаг* **..** *конечное значение*
оператор

for ■ ∈ ■ Конструкция оператора

■

Сумма цифр от 1 до 10

SUM := | S ← 0
for i ∈ 1 .. 10
S ← S + i

SUM = 55

Сумма нечетных цифр от 1 до 10

SUM_N := | S ← 0
for i ∈ 1 ,3 .. 10
S ← S + i

SUM_N = 25

Сумма чисел от 1 до 10 с шагом 0.5

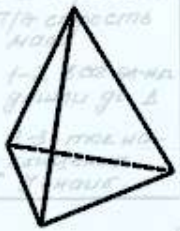
SUM := | S ← 0
for i ∈ 1 ,1.5 .. 10
S ← S + i

SUM = 104.5

Сумма четных цифр от 10 до 2

SUM_N := | S ← 0
for i ∈ 10 ,8 .. 2
S ← S + i

SUM_N = 30



Цикл с известным числом повторений

Границы диапазона могут быть заданы:

- конкретными значениями;
- переменными;
- в виде вектора.

Сумма цифр от m до n

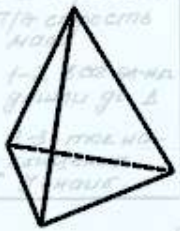
```
Sum(m,n) := | S ← 0
            | for i ∈ m..n
            |   S ← S + i
```

Sum(10,20) = 165

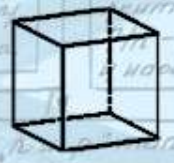
Сумма цифр, заданных вектором

```
Sum_V(v) := | S ← 0
            | for i ∈ v
            |   S ← S + i
```

v := (1 7 2 5 6) Sum_V(v) = 21



Цикл с известным числом повторений



Программный модуль с применением цикла, в теле которого выполняются два оператора. Результатом работы функции является вектор, первый элемент которого сумма чисел от 1 до n, второй – произведение.

```
P(n) := | S ← 0
        | P ← 1
        | for i ∈ 1..n
        |     | S ← S + i
        |     | P ← P·i
        | V0 ← S
        | V1 ← P
        | V
```

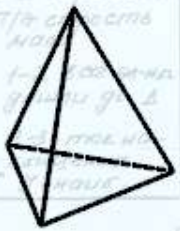
Присвоение начальных значений

Вычисляется значение суммы чисел и произведения от 1 до n.

Формируется вектор из значений суммы и произведения

$$P(5) = \begin{pmatrix} 15 \\ 120 \end{pmatrix}$$

Результат работы функции



Цикл с условием

Оператор **while** применяют для создания цикла, количество повторений которого неизвестно, но предусмотрен выход из него по некоторому логическому **условию**. Запись цикла имеет вид:

while *условие*

оператор.

Это означает, что оператор **будет выполняться** в цикле до тех пор, **пока истинно условие**.

while ▪ Конструкция оператора

▪

Сумма цифр от m до n

$$\text{Sum}(m,n) := \left| \begin{array}{l} S \leftarrow 0 \\ i \leftarrow m \\ \text{while } i \le n \\ \quad \left| \begin{array}{l} S \leftarrow S + i \\ i \leftarrow i + 1 \end{array} \right. \\ S \end{array} \right.$$

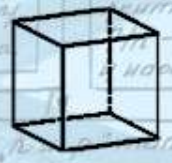
$\text{Sum}(10,20) = 165$

Определение наибольшего общего кратного двух чисел по алгоритму Евклида

$$P(a,b) := \left| \begin{array}{l} \text{while } a \ne b \\ \quad \left| \begin{array}{l} a \leftarrow a - b \text{ if } a > b \\ b \leftarrow b - a \text{ otherwise} \end{array} \right. \\ a \end{array} \right.$$

$P(5,15) = 5 \quad P(21,18) = 3 \quad P(1,13) = 1$

Оператор прерывания



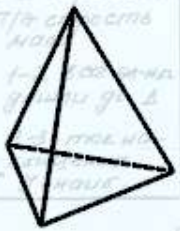
Иногда бывает необходимо завершить цикл досрочно. Для этого предназначен оператор **break**.

Пример: определение положения первого нулевого элемента в массиве чисел.

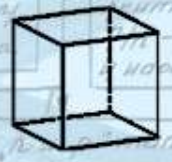
```
ORIGIN := 1

G(X) := | k ← 0
        | for i ∈ 1..rows(X)
        |   if X_i = 0
        |     | k ← i
        |     | break
        | k
```

$$Y := \begin{pmatrix} 1 \\ 2 \\ 0 \\ 3 \\ 5 \end{pmatrix} \qquad Z := \begin{pmatrix} 1 \\ 8 \\ 9 \end{pmatrix}$$

$G(Y) = 3$ $G(Z) = 0$



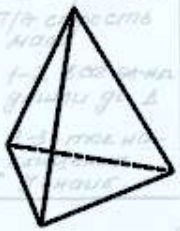
Оператор продолжения



Оператор **continue** – это так же оператор прерывания, но в отличие от break, который прерывает цикл, он прерывает выполнение только текущей операции.

Пример: вычисления произведения ненулевых элементов массива

$$F(x) := \left| \begin{array}{l} P \leftarrow 1 \\ \text{for } i \in 0..\text{rows}(x) - 1 \\ \quad \left| \begin{array}{l} \text{continue if } x_i = 0 \\ P \leftarrow P \cdot x_i \end{array} \right. \\ P \end{array} \right. \qquad X := \begin{pmatrix} 1 \\ 5 \\ 0 \\ 3 \\ 0 \\ 1 \end{pmatrix} \qquad F(X) = 15$$



Оператор прерывания программного модуля

Оператор **return** прерывает выполнение программного модуля в любой точке и возвращает значение выражения, переменной или текстовое сообщение, стоящее следом за ним.

Пример : функция определяет, является ли вектор единичным. При наличии хотя бы одного элемента вектора отличного от единицы программный модуль прерывается.

$$E(V) := \left| \begin{array}{l} \text{for } i \in 0..\text{rows}(V) - 1 \\ \quad \text{return "NO" if } V_i \neq 1 \\ \text{"YES"} \end{array} \right.$$

$$a := \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad b := \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

$$E(a) = \text{"YES"} \qquad E(b) = \text{"NO"}$$

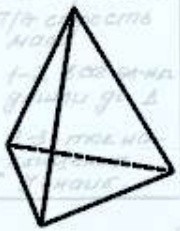
Оператор обработки ошибок

Если при составлении программного модуля предполагается, что какая либо команда может вызвать ошибку (например, деление на ноль), то можно воспользоваться оператором **on error** для перехвата этой ошибки.

Оператор перехвата ошибок имеет следующую конструкцию:

выражение1 **on error** *выражение2*,

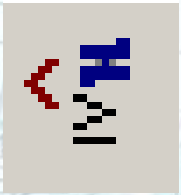
где *выражение2* это действие которое должно выполниться в данной строке программного модуля, *выражение1* – выражение, которое будет выполнено вместо *выражение2*, если при выполнении последнего произойдет ошибка.

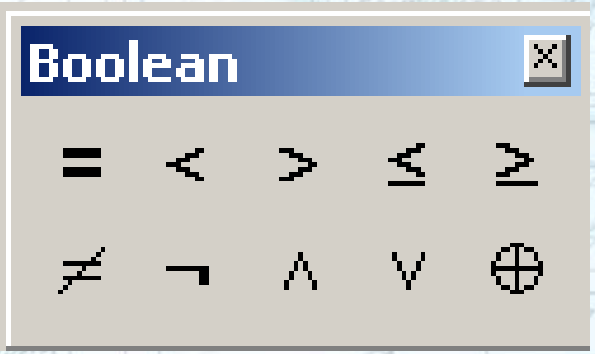


$F(a,b) := \infty \text{ on error } \frac{a}{b}$ $\quad F(0,3) = 0$ $\quad F(3,0) = 1 \times 10^{307}$

$F(a,b) := 1 \text{ on error } \frac{a}{b}$ $\quad F(0,3) = 0$ $\quad F(3,0) = 1$

На панели инструментов **Boolean** которая управляется кнопкой математической палитры, расположены логические операторы и операторы отношения.





- и (And) ^ (**Ctrl+Shift+7**);
- или (Or) ∨ (**Ctrl+Shift+6**);
- исключающее или (Exclusive or) ⊕ (**Ctrl+Shift+5**);
- отрицание (Not) ¬(**Ctrl+Shift+1**).

A	B	¬ A	A ^ B	A ∨ B	A ⊕ B
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0