

# **ПРАКТИКА №6**

---

## **МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ В ПАКЕТЕ MATHCAD. ПРОЦЕДУРЫ- ФУНКЦИИ**

# Два способа программирования в среде MathCAD:

---

программирование  
без использования  
программных  
модулей

- Безмодульное программирование
- Для решения простых задач

программирование с  
использованием  
программных  
модулей

- Модульное программирование;
- Позволяет реализовывать в программе независимые блоки - подпрограммы-функции;
- Четкая структура программы

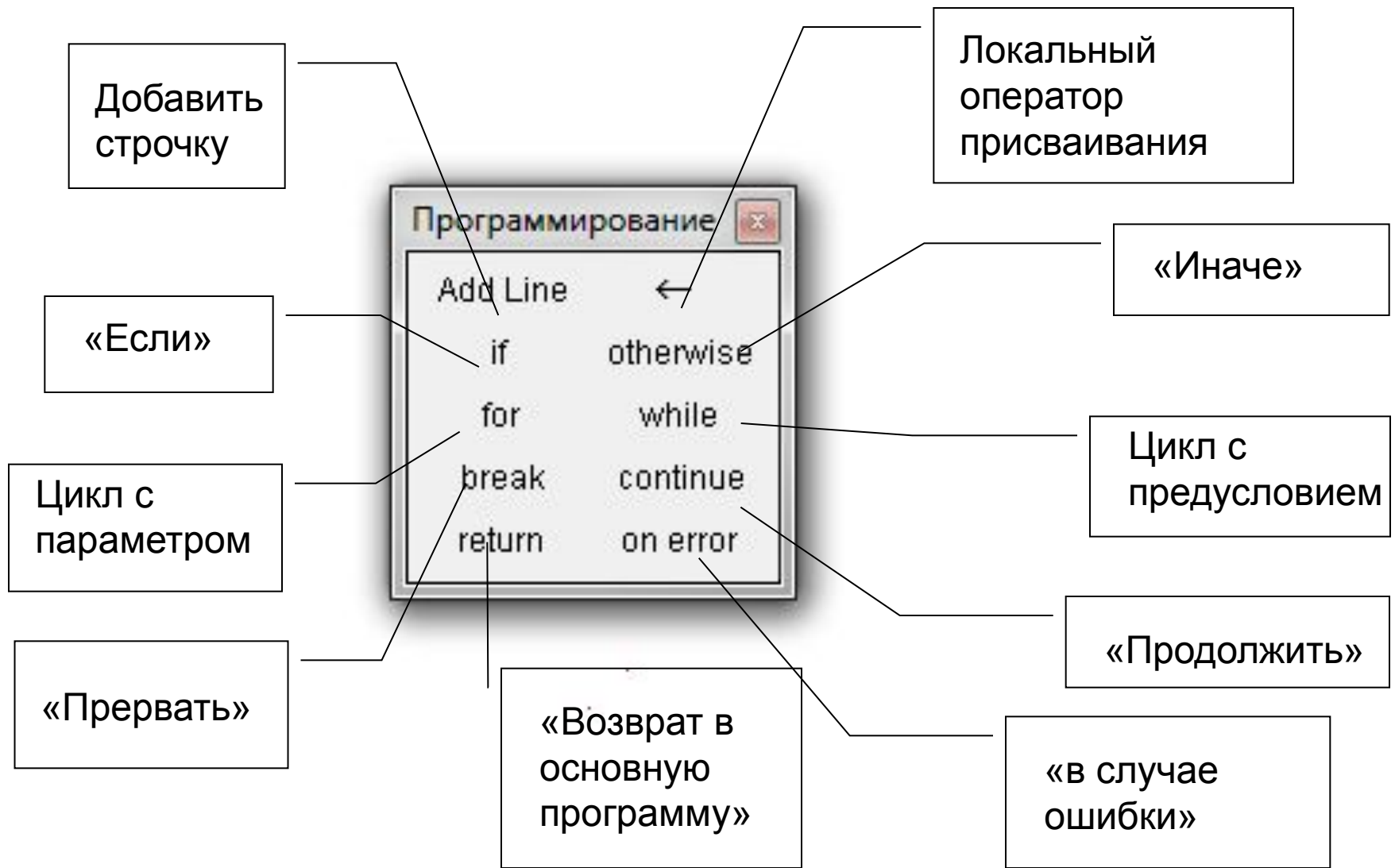
# Модульное программирование

---

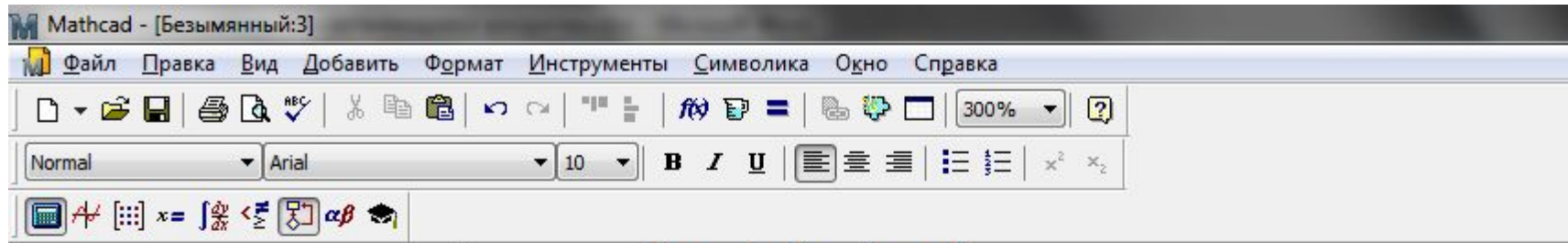
- Используются процедуры – функции (П-Ф)
- Описание П-Ф размещается в рабочем документе перед ее вызовом
- Включает в себя:
  - имя подпрограммы-функции,
  - список формальных параметров (может отсутствовать)
  - тело подпрограммы-функции
- Для ввода конструкций в тело П-Ф используется панель инструментов **ПРОГРАММИРОВАНИЕ**

# Панель ПРОГРАММИРОВАНИЕ

---

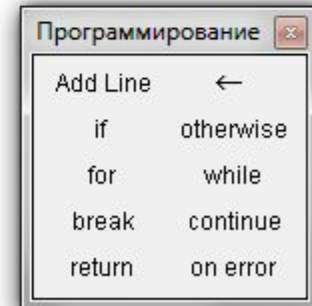
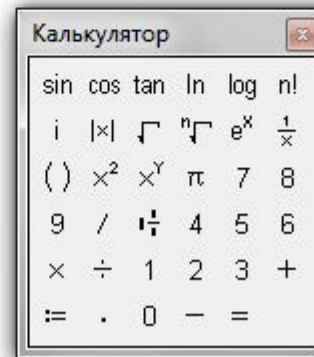


# Программа с использованием П-Ф



$$S(a, b, c) := \begin{array}{l} p \leftarrow \frac{(a + b + c)}{2} \\ S \leftarrow \sqrt{p \cdot (p - a) \cdot (p - b) \cdot (p - c)} \\ S \end{array}$$

$$S(3, 6, 7) = 8.944$$



# Процедура - Функция

---

- Имеет **оригинальное имя**, посредством которого осуществляется обращение к ней. Через это же имя «возвращается» результат выполнения П-Ф
- После имени П-Ф идет **список формальных параметров**, заключенный в круглые скобки. Формальные параметры отделяются друг от друга запятой
- П-Ф может не иметь формальных параметров, и тогда данные передаются через имена переменных, определенных выше описания П-Ф
- В качестве формальных параметров могут использоваться имена простых переменных, массивов и функций
- Все формальные параметры являются **входными**. Через формальные параметры «внутри» П-Ф «передаются» данные, необходимые для выполнения вычислений

# Процедура - Функция

---

- **Тело** подпрограммы-функции включает любое число операторов: локальных операторов присваивания, условных операторов и операторов цикла, а также вызов других П-Ф и функций пользователя
- Для задания внутри программы значения какой-либо переменной используется **локальный оператор присваивания**: < имя переменной > ← < выражение >
- Для выполнения П-Ф необходимо обратиться к ее имени с указанием **списка фактических параметров** :  
< имя П-Ф > (< список фактических параметров >)
- Между фактическими и формальными параметрами должно быть соответствие по количеству, порядку следования и типу
- Обращение к П-Ф должно находиться после ее описания, и к моменту обращения фактические параметры должны быть определены

# Описание подпрограммы-функции

---

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

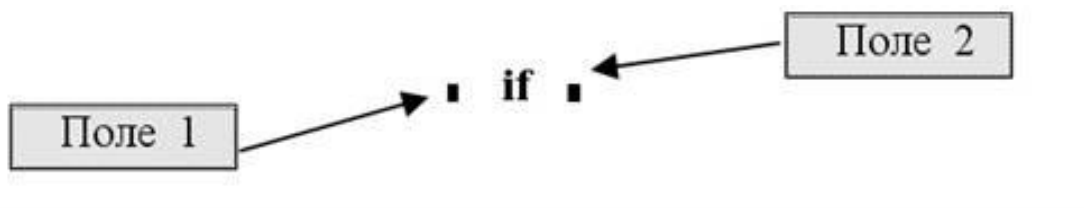
- ввести имя П-Ф и список формальных параметров (в круглых скобках)
- ввести символ “:” – на экране отображается как :=
- открыть палитру **ПРОГРАММИРОВАНИЕ** и щелкнуть кнопкой Add line
- На экране появится вертикальная черта и вертикальный столбец с двумя полями для ввода операторов, образующих тело П-Ф



# Условный оператор if

---

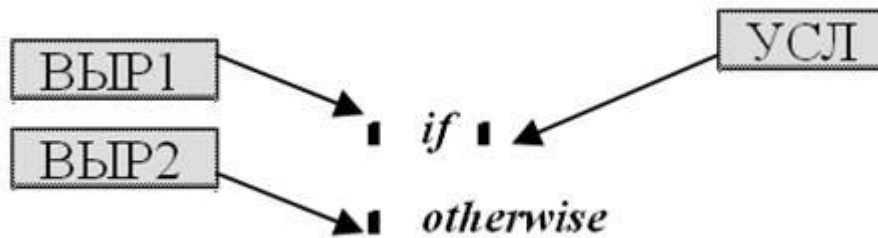
- Используется только в теле П-Ф для программирования условной структуры **ЕСЛИ – ТО**
- Для ввода необходимо щелкнуть на кнопке if палитры **ПРОГРАММИРОВАНИЕ**.
- На экране появляется конструкция с двумя полями ввода



- В Поле 2 вводится логическое выражение (УСЛОВИЕ)
- В Поле 1 вводится конструкция ВЫР1, которая выполняется, если УСЛ = 1 (истинно)
- Если УСЛ = 0 (ложь), то ВЫР1 не выполняется

# Оператор *otherwise*

- Для программирования условной структуры **ЕСЛИ – ТО – ИНАЧЕ** используется оператор *otherwise*, вводимый с палитры **ПРОГРАММИРОВАНИЕ**



- в поле ВЫР2 оператора *otherwise* помещается конструкция, которая выполняется, если проверяемое логическое выражение УСЛ ложно

# ПРИМЕР

---

Составить описание П-Ф, вычисляющей функцию

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

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

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

# ПРИМЕР

Составить описание П-Ф, вычисляющей функцию

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

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

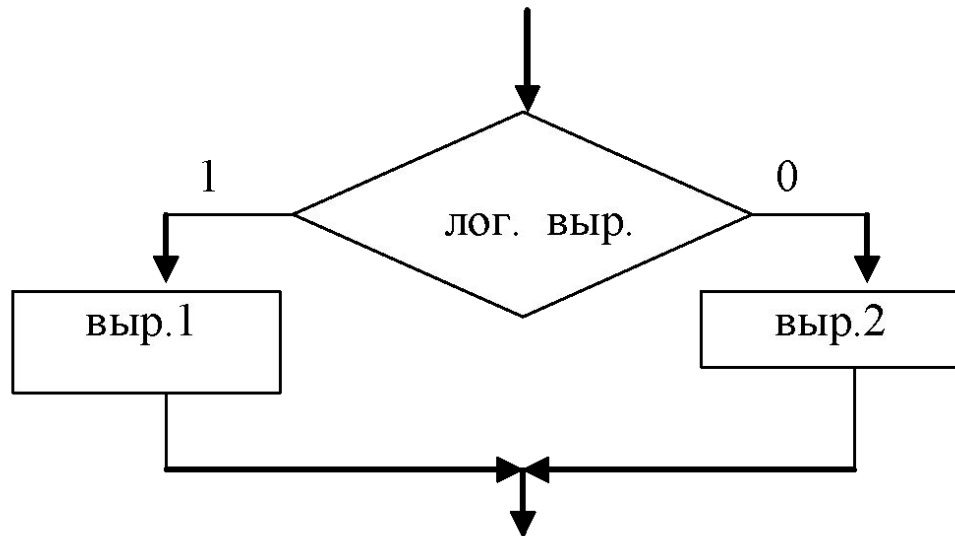
- Функция  $z(t)$  получит значение  $\ln(t)$  только тогда, **когда не выполняются условия, записанные в двух вышестоящих строках тела П-Ф**
- Если в строке 3 ввести просто  $\ln(t)$ , то это выражение **будет вычисляться всегда** вне зависимости от выполнения заданных выше условных операторов.

# ПРИМЕР

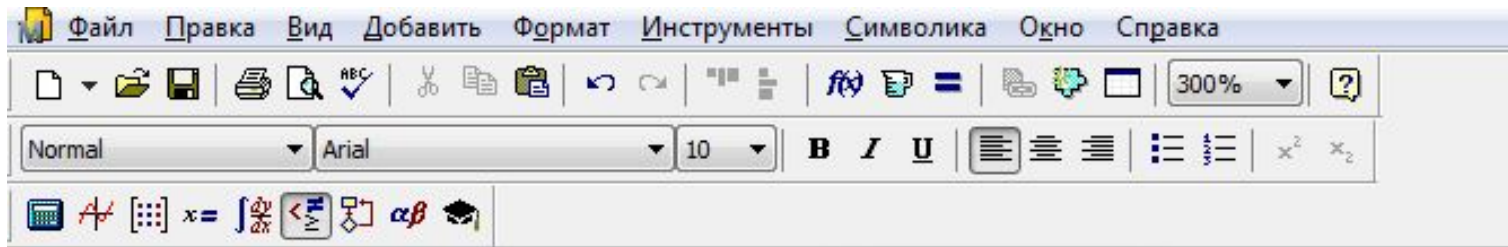
Используя условную функцию *if* запрограммировать вычисление функции

$$g(x) = \begin{cases} 2, & \text{если } x \leq 2; \\ x, & 2 \leq x \leq 8; \\ 10, & \text{если } x > 8. \end{cases}$$

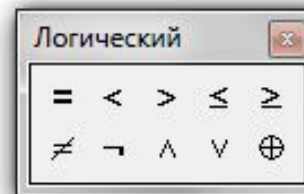
*if* (<логическое выражение>, <выр. 1>, <выр. 2>),



# Безмодульная программа



$x := 3$



$y := \text{if}(x \leq 2, 2, \text{if}(2 < x \leq 8, x, 10))$

$y = 3$

# Модульная программа

---

$$g(x) := \begin{cases} 2 & \text{if } x < 2 \\ x & \text{if } 2 \leq x \leq 8 \\ 10 & \text{if } x > 8 \end{cases}$$

$$g(3) = 3$$

ИЛИ

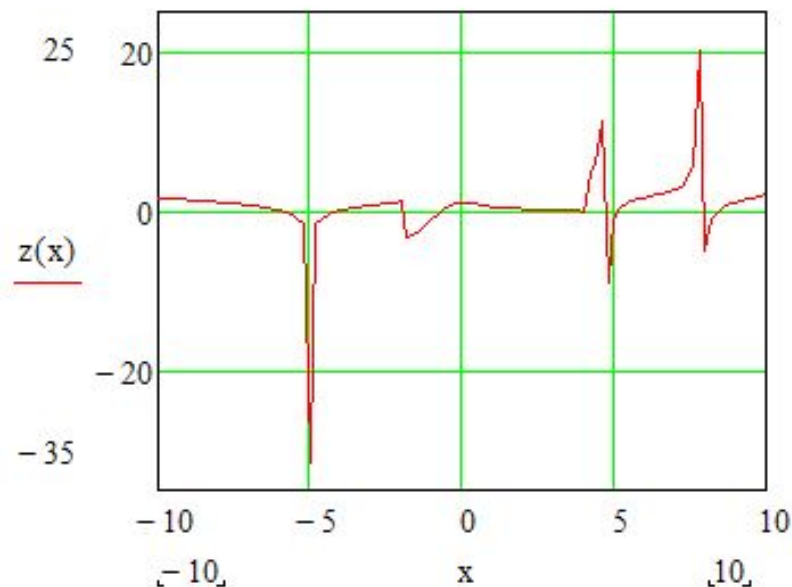
$$g(x) := \begin{cases} 2 & \text{if } x < 2 \\ x & \text{if } 2 \leq x \leq 8 \\ 10 & \text{otherwise} \end{cases}$$

$$g(3) = 3$$

Составить программу для вычисления функции  $z(x)$  на интервале -10 до 10 с шагом 0,2 при помощи

- Безмодульного программирования
- С использованием П-Ф
- Построить ее график

$$z(x) = \begin{cases} e^x + \ln|x + 5|, & x < -2 \\ \frac{\sin x + \cos^2 x}{2^x}, & -2 \leq x \leq 4 \\ \operatorname{tg}x + \frac{x + 2}{x - 2}, & x > 4 \end{cases}$$





# Алгоритмы циклической структуры

- Последовательность действий, повторяющаяся в зависимости от выполнения какого-либо условия, называется **телом цикла**
- **Вложенным** называется цикл, находящийся внутри тела другого цикла
- **Итерационным** называется цикл, число повторений которого не задается, а определяется в ходе выполнения цикла. В этом случае одно повторение цикла называется **итерацией**

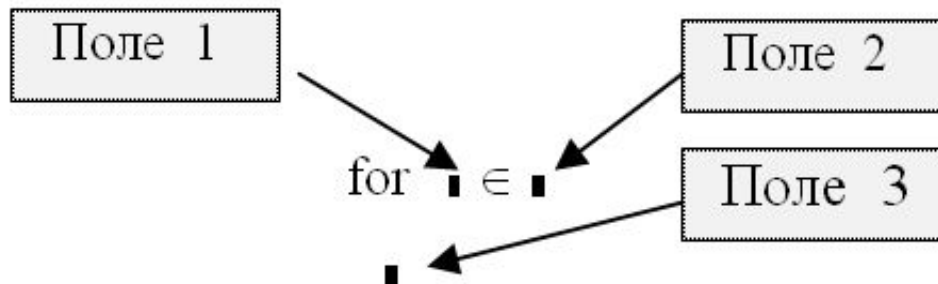
# Цикл For

## Цикл с параметром

**Поле 1** — имя переменной, являющейся параметром цикла

**Поле 2** — закон изменения параметра цикла, используется дискретная переменная или *массив*

**Поле 3** — операторы, составляющие тело цикла



# ПРИМЕР1: заполнение

## вектора

Составить П-Ф, формирующую вектор из  $n$  компонент по заданной формуле:

$$y[i] = \frac{3^i}{i}, \quad \text{где } i = 1, 2, \dots, 10$$

Form\_vec1(n) :=  $\left\{ \begin{array}{l} \text{for } i \in 1..n \\ y_i \leftarrow \frac{3^i}{i} \\ y \end{array} \right.$

Form\_vec1(10) =

	0
0	0
1	3
2	4.5
3	9
4	...

# ПРИМЕР2: поиск суммы, произведения и количества элементов вектора

Найти сумму, произведение и количество элементов вектора:

$$X = (2, -4, 0.1, -3)$$

ORIGIN := 1

$$x := \begin{pmatrix} 2 \\ -4 \\ 0.1 \\ -3 \end{pmatrix}$$

```
sum(x) := | sum ← 0
           | for i ∈ 1..4
           |   sum ← sum + xi
           | sum
```

$$\text{sum}(x) = -4.9$$

```
pr(x) := | pr ← 1
          | for i ∈ 1..4
          |   pr ← pr · xi
          | pr
```

$$\text{pr}(x) = 2.4$$

```
kol(x) := | kol ← 0
           | for i ∈ 1..4
           |   kol ← kol + 1
           | kol
```

$$\text{kol}(x) = 4$$

# ПРИМЕР 3: поиск максимального и минимального элементов вектора

Найти максимальный и минимальный элементы вектора:

$$X = (2, -4, 0.1, -3)$$

```
max_el(x) := |
               max ← -106
               for i ∈ 1..4
                 max ← xi if xi > max
               |
               max
```

$$\text{max\_el}(x) = 2$$

```
min_el(x) := |
               min ← 106
               for i ∈ 1..4
                 min ← xi if xi < min
               |
               min
```

$$\text{min\_el}(x) = -4$$

# ПРИМЕР 4: поиск порядковых номеров максимального и минимального элементов вектора

Найти порядковые номера максимального и  
минимального элементов вектора:  $X = (2, -4, 0.1, -3)$

```
max_el(x) :=  
  max ← -106  
  nmax ← 1  
  for i ∈ 1..4  
    if xi > max  
      | max ← xi  
      | nmax ← i  
  ( max )  
  ( nmax )
```

$$\text{max\_el}(x) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

```
min_el(x) :=  
  min ← 106  
  nmin ← 1  
  for i ∈ 1..4  
    if xi < min  
      | min ← xi  
      | nmin ← i  
  ( min )  
  ( nmin )
```

$$\text{min\_el}(x) = \begin{pmatrix} -4 \\ 2 \end{pmatrix}$$