ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ЮФУ
РОСТОВ-НА-ДОНУ·ТАГАНРОГ

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК
имени И.И. Воровича

Пакеты научных вычислений

Лекция 2

Структуры данных в Maple.
Способы задания функций и замена переменных.
Решение уравнений и неравенств



Наседкина А. А.

Структуры данных в Maple

Структуры данных:

- Последовательность выражений (Expression sequence)
- Множество, или набор (Set)
- Список (List)
- Массив (Array)
- Строка (String)

Преобразование типов и структур

Структуры данных

- Выражения Maple могут быть составлены как из простых объектов, так и из сложных объектов и других выражений Maple. Среди сложных объектов выделяют *структуры данных*.

  - Последовательность выражений (Expression sequence)
  - Множество, или набор (Set)
  - Список (List)
  - Массив (Array)
  - Строка (String)

  - Таблица (Table)
  - Матрица, вектор (Matrix, Vector)

Структуры данных: последовательность выражений (expression sequence)

- **Последовательность выражений (Expression sequence)** – это группа выражений Maple, отделенных друг от друга запятыми

º, º, º, º

- Среди выражений в последовательности могут быть объекты разных типов данных. Тип результирующего объекта - **exprseq**.

```
> s:=2,x,a+b,`a+b`,sin(x^2),x;
```

$$s := 2, x, a + b, a + b, \sin(x^2), x$$

```
> whattype(s);
```

exprseq

1	2	3	4	5	6
2	x	$a + b$	$a + b$	$\sin(x^2)$	x
-6	-5	-4	-3	-2	-1

- Доступ к элементам последовательности

```
> s[-3];whattype(%);
```

$$a + b$$

symbol

```
> s[3];s[4];evalb(s[3]=s[4]);
```

$$a + b$$

$$a + b$$

false

Структуры данных: последовательность выражений (expression sequence)

- Доступ к нескольким элементам последовательности

```
> s[2..4];
```

$$x, a + b, a + b$$

- Добавление элементов в последовательность и приращение последовательности

```
> t:=s[2..4],8;
```

$$t := x, 5 + b, a + b, 8$$

```
> t:=t,s[1],10;
```

$$t := x, 5 + b, a + b, 8, 2, 10$$

- Пустая последовательность

```
> s:=NULL;
```

$$s :=$$

- Оператор формирования последовательности **$**

expr $ *name* = *initial* .. *final*

```
> $1..5;
```

$$1, 2, 3, 4, 5$$

```
> s := i^2 $ i = 2..6;
```

$$s := 4, 9, 16, 25, 36$$

Структуры данных: множество (set)

- **Множество, или набор (Set)** – это группа выражений Maple, записанных в *фигурных* скобках через запятую

{º, º, º, º}

Данный объект имеет все черты математического множества:

- каждый элемент хранится в единственном экземпляре
- заданный порядок элементов не хранится

Среди выражений во множестве могут быть объекты разных типов данных. Тип результирующего объекта «множества» называется **set**.

```
> m:={2,x,x,b,a,a};
```

$$m := \{2, x, a, b\}$$

```
> whattype(m);
```

$$set$$

Структуры данных: множество (set)

- Доступ к элементам множества

```
> m:={2,x,x,b,a,a};
```

$$m := \{2, x, a, b\}$$

```
> m[3];
```

$$a$$

```
> m[5];
Error, invalid subscript selector
```

- Пустое множество

```
> p:={};
```

$$p := \{\ \}$$

- Операции алгебры множеств: объединение (**union**), пересечение (**intersect**), разность (**minus**)

```
> S1:={a,b,c};
```

$$S1 := \{a, b, c\}$$

```
> S2:={b,c,d,e,d};
```

$$S2 := \{b, c, d, e\}$$

```
> SU:=S1 union S2;
```

$$SU := \{a, b, c, d, e\}$$

```
> SU := S1 ∪ S2;
```

$$SU := \{a, b, c, d, e\}$$

Структуры данных: множество (set)

- Операции алгебры множеств

```
> S1:={a,b,c};
```

$$S1 := \{a, b, c\}$$

```
> S2:={b,c,d,e,d};
```

$$S2 := \{b, c, d, e\}$$

```
> SI:=S1 intersect S2;
```

$$SI := \{b, c\}$$

```
> SI := S1 ∩ S2;
```

$$SI := \{b, c\}$$

```
> Sm1 := S1 \ S2;
```

$$Sm1 := \{a\}$$

```
> Sm2:=S2 minus S1;
```

$$Sm2 := \{d, e\}$$

- Добавление элементов во множество и приращение множества

```
> S := SI ∪ {1,x}
```

$$S := \{1, a, b, c, x\}$$

```
> S := S ∪ {t}
```

$$S := \{1, a, b, c, t, x\}$$

Структуры данных: список (list)

- **Список (List)** – это группа выражений Maple, записанных в *квадратных* скобках через запятую

[o, o, o, o]

Данный объект имеет черты, противоположные множеству:

- хранятся все повторяющиеся элементы
- хранится заданный порядок элементов

Среди выражений в списке могут быть объекты разных типов данных.

Тип результирующего объекта «множества» называется **list**.

```
> L:=[2,x,x,b,a,a];
```

$$L := [2, x, x, b, a, a]$$

```
> whattype(L);
```

$$list$$

- Пустой список

```
> Lp:=[];
```

$$Lp := [\,]$$

Структуры данных: список (list)

- Доступ к элементам списка

```
> L:=[2,x,x,b,a,a];
```

$$L := [2, x, x, b, a, a]$$

```
> L[5];
```

$$a$$

```
> L[3..5];
```

$$[x, b, a]$$

- Добавление элементов в список и приращение списка

```
> M := [op(L),c];
```

$$M := [2, x, x, b, a, a, c]$$

```
> M := [op(M),sin(x)];
```

$$M := [2, x, x, b, a, a, c, \sin(x)]$$

- Сортировка списка **sort(L,f)** – сортирует список **L,** необязательный аргумент **f** задает порядок сортировки

```
> s:=[2,1,3]:
> sort(s);
```

$$[1, 2, 3]$$

```
> sort(s, `>`);#сортировка по убыванию
```

$$[3, 2, 1]$$

Подсчета и извлечение элементов из списка и множества

nops(x) – выдает количество элементов в объекте **x**
op(i,e) – извлекает элемент, находящийся на позиции **i** в объекте **e**
op(i..j,e) – извлекает элементы, находящиеся на позициях с **i** по **j** в объекте **e** (возвращает последовательность элементов)
op(e) – извлекает все элементы объекта **e** (возвращает последовательность элементов)

```
> s:={a,b,4,-1,f,c,f};nops(s);
```

$$s := \{-1, 4, a, f, b, c\}$$

$$6$$

```
> op(s);
```

$$-1, 4, c, a, b, f$$

```
> op(3,s);op(1..3,s);
```

$$a$$

$$-1, 4, a$$

Команды подсчета и извлечения элементов

```
> s:=[a,b,4,-1,f,c,f];nops(s);
```

$$s := [a, b, 4, -1, f, c, f]$$

$$7$$

```
> op(4,s);op(1..4,s);
```

$$-1$$

$$a, b, 4, -1$$

- Команды **nops** и **op**, могут использоваться и для других объектов

$> g := x^3 + 3x^2 + 5x - 6$

$$g := x^3 + 3x^2 + 5x - 6$$

$> nops(g)$

$$4$$

$> nops(op(1, g))$

$$2$$

$> op(1, op(2, g))$

$$3$$

Команды подстановки значений для операндов: subsop

subsop(i=a,expr) – подстановка значения *a* для *i*-го операнда выражения *expr*

subsop(eq_i1,...eq_i2,expr) – подстановка значений для каждого операнда выражения expr

> $p := x^2 + x - y$

$$p := x^2 + x - y$$

> $op(1, p); op(2, p); op(3, p)$

$$x^2$$

$$x$$

$$-y$$

> $subsop(2 = y, p)$

$$x^2$$

> $op(1, op(1, p)); op(2, op(1, p));$

$$x$$

$$2$$

> $subsop(3 = op(1, p), p)$

$$2x^2 + x$$

Удаления и перестановка элементов в списке и множестве

subsop(i=NULL,s) – удаление *i*-го элемента в списке или множестве s

subsop(i=s[j],j=s[i]) – перестановка *i*-го и *j*-го элемента местами

Список

```
> l := [i·100 $ i = 1..5]
        l := [100, 200, 300, 400, 500]                    (1)
```

Множество

```
> s := {i·100 $ i = 1..5}
        s := {100, 200, 300, 400, 500}                    (2)
```

Последовательность

```
> ss := i·100 $ i = 1..5
        ss := 100, 200, 300, 400, 500                     (3)
```

Удаление элемента под номером 2 из списка *l*

```
> l := subsop(2 = NULL, l);
        l := [100, 300, 400, 500]                         (4)
```

Удаление элемента под номером 4 (по порядку хранения элементов) из множества *s*

```
> s := subsop(4 = NULL, s);
        s := {100, 200, 300, 500}                         (5)
```

Удаление элемента под номером 3 из последовательности *ss*

```
> ss := subsop(3 = NULL, ss);
      # для последовательности не работает
Error, improper op or subscript selector
```

```
> ss := ss[1..2], ss[4..-1]
      #Последний элемент имеет индекс -1
        ss := 100, 200, 400, 500                          (6)
```

Перестановка второго и последнего элемента в списке *l*

```
> l := [i·100 $ i = 1..5]; # исходный список
        l := [100, 200, 300, 400, 500]                    (7)
```

```
> l := subsop(2 = l[-1], -1 = l[2], l);
      #список с переставленными элементами
        l := [100, 500, 300, 400, 200]                    (8)
```

Перестановка элементов во множестве не имеет смысла, так как порядок не хранится!

```
> s := {i·100 $ i = 1..5}; # исходное множество
        s := {100, 200, 300, 400, 500}                    (9)
```

```
> s := subsop(2 = s[4], 4 = s[2], s);
      #множество с переставленными элементами
        s := {100, 200, 300, 400, 500}                    (10)
```

Перестановка второго и третьего элементов в последотельности *ss*

```
> ss := i·100 $ i = 1..5
      #исходная последовательность\
        ss := 100, 200, 300, 400, 500                     (11)
```

```
> ss := ss[1], ss[3], ss[2], ss[4..-1];
      #последовательность с переставленными
      элементами
        ss := 100, 300, 200, 400, 500                     (12)
```

Структуры данных: массив (Array)

- **Массив (Array)** – это обобщение списка на любую размерность (2, 3 и т. д). Обычный список является одномерным массивом. Для индексов массива можно использовать любые целые числа. Для задания массива используется команда **array** (или новая команда **Array**).

array(indexfunc,dims,init) – создает массив, элементы которого задает (необязательная) индексирующая функция **indexfunc** (задается структура матрицы массива: симметричная, диагональная и т. д.), переменная **dims** – последовательность диапазонов изменения индексов, **init** – список начальных значений массива.

```
> b:=array(1..2,1..3,[[1,2,-1],[-2,3,1]]);
```

$$b := \begin{bmatrix} 1 & 2 & -1 \\ -2 & 3 & 1 \end{bmatrix}$$

```
> c:=array(1 .. 2, 1 .. 2, 1..2,[[[1, 2], [3, 4]],[[5,6],[7,8]]]);
```

$$c := ARRAY([1..2, 1..2, 1..2], [(1,1,1)=1, (1,1,2)=2, (1,2,1)=3, (1,2,2)=4, (2,1,1)=5, (2,1,2)=6, (2,2,1)=7, (2,2,2)=8])$$

Структуры данных: строка (string)

- **Строка (String)** – это любой набор символов, заключенный в двойные кавычки. Длина строки в Maple практически не ограничена.

" "

Тип объекта «строка» называется **string**.

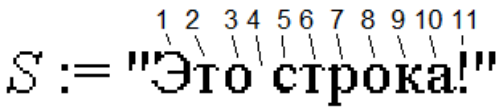
```
> S:="Это строка!";
```

$$S := \text{"Это строка!"}$$

```
> whattype(S);
```

$$string$$

- Доступ к элементам строки




```
> S[9..11];
```

"ка!"

```
> S[14];
```

""

Структуры данных: строка (string)

cat(a,b,c) – конкатенация строк и выражений

```
> cat("program", "ming")
```

"programming"

```
> cat(a, b); cat(a, 1 ..10)
```

ab

a1, a2, a3, a4, a5, a6, a7, a8, a9, a10

Некоторые команды пакета StringTools для работы со строками
Length(s) – выдает длину строки **s**
Split(s) – выдает список строк, составляющих отдельные слова в строке **s** (по умолчанию анализируется расположение пробелов). Можно задать тип разделителя: **Split(s, sep)**
Stem(s) – выдает строку, содержащую основу слова **s** (English)

```
> s:="Impressive string":
> with(StringTools):
> Length(s);
```

17

```
> m:=Split(s);
```

m := ["Impressive", "string"]

```
> Stem(m[1]);
```

"impress"

Использование структур данных в командах Maple

Структуры данных «множество» и «список» широко используются как аргументы различных команд. Некоторые команды допускают использование любой из этих двух структур, если порядок следования элементов не важен. Если должен учитываться порядок элементов в структуре, то надо использовать «список».

```
> solve({x+y=10,x-y=5},{x,y});
```

$$\left\{y = \frac{5}{2}, x = \frac{15}{2}\right\}$$

```
> solve({x+y=10,x-y=5},[x,y]);
```

$$\left[\left[x = \frac{15}{2}, y = \frac{5}{2}\right]\right]$$

```
> f:=x-10*y:eval(f,[x=1,y=2]);
```

$$-19$$

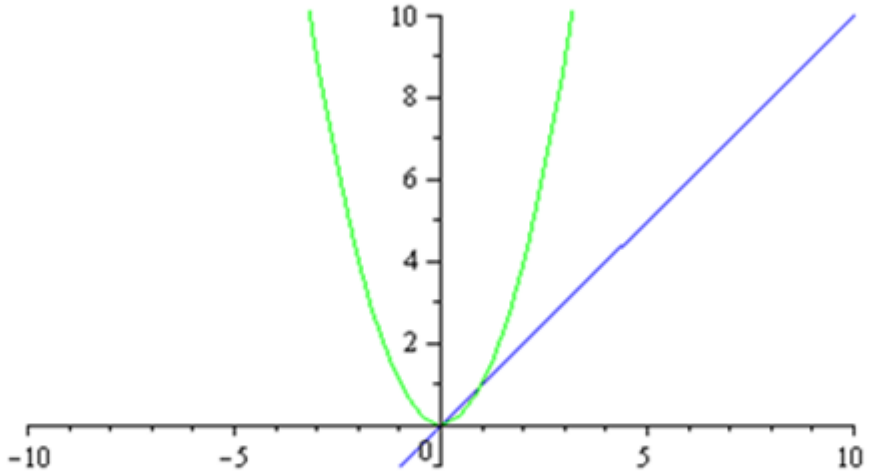
```
> eval(f,{x=1,y=2});
```

$$-19$$

Использование структур данных в командах Maple: пример для команды plot

Требования к аргументам команды и их типам всегда можно прочитать в справочной информации по данной команде.

```
> plot([x,x^2],x=-10..10,
  -1..10,scaling=CONSTRAINED,color=[blue,green]);
```




```
> plot({x,x^2},x=-10..10,-
1..10,scaling=CONSTRAINED,color={blue,green});
Error, (in plot) invalid color specification: {blue, green}
```

Преобразование типов и структур: convert

- **convert** – преобразует выражение в другой вид, может использоваться для преобразования типа выражения или структуры данных

  convert(expr, form, optional arguments)

> $convert([1, 2, 3], vector)$

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

> $convert(x^2 + 2 \cdot x \cdot y + y^2, string)$

"x^2+2*x*y+y^2"

> $convert(9, binary)$

1001

> $convert(1.23456, rational)$

$$\frac{3858}{3125}$$

> $convert\left(\frac{1}{8}, float, 3\right)$

0.125

> $convert([1, 2, 3, 4], `+`)$

10

Способы задания функций и замена переменных

- Команды подстановки выражений
- 1. Определение функции как выражения Maple
- 2. Определение функции как функционального оператора
- 3. Определение функции с помощью команды unapply
- Определение кусочно-непрерывных функций

Команды подстановки выражений: subs

subs(x=a, expr) – подстановка x=a в выражение expr, зависящее от x

subs(eq1,...eqn, expr) – выполнение подстановок, записанных в виде уравнений eq1,...,eqn, в выражение expr

> $subs(x=-2, x^2+x+1)$

$$3$$

> $subs\left(\sin(x)=t, \frac{\sin(x)}{\sqrt{1-\sin(x)}}\right)$

$$\frac{t}{\sqrt{1-t}}$$

> $subs(x=1, y=2, x-5\cdot y)$

$$-9$$

> $subs([x=1, y=2], x-5\cdot y)$

$$-9$$

Выполняется только подстановка, но не вычисление выражения!

> $subs\left(x=\frac{\mathrm{Pi}}{2}, y=\mathrm{Pi}, \sin(x)+\cos(y)\right)$

$$\sin\left(\frac{1}{2}\pi\right)+\cos(\pi)$$

> $simplify(\%)$

$$0$$

1. Определение функции как выражения Maple

- Определение функции как выражения Maple с помощью оператора присваивания (**:=**)
- Данный способ не является *способом задания функции* в терминах Maple.

$> f := \sin(x) + x^2 :$

Вычислим значение f при заданном x

$> x := \frac{\pi}{4} : f$

$$\frac{1}{2}\sqrt{2} + \frac{1}{16}\pi^2$$

$> x := 10 : f$

$$\sin(10) + 100$$

$> g := 3 \cdot a + 5 \cdot b^3 : a := 1 : b := -1 : g$

$$-2$$

$> g1 := 3x + 5y^3 : x := 1 : y := -1 : g1$

$$25$$

Почему такой результат?

1.1 Определение функции как выражения Maple: функция одной переменной

Чтобы вычислить значение выражения с одной переменной, нужно использовать

- команду **eval** в виде **eval(expr,x=a)**
- либо команду **subs** в виде **subs(x=a,expr)**

```
> restart
> f := sin(x) + x^2 :
> eval(f, x = π)
```

$$\pi^2$$

```
> subs(x = π, f)
```

$$\sin(\pi) + \pi^2$$

```
> simplify(%)
```

$$\pi^2$$

```
> evalf(%)
```

$$9.869604404$$

1.2 Определение функции как выражения Maple: функция многих переменных

Чтобы вычислить значение выражения со многими переменными, нужно использовать

- команду **eval** в виде **eval(expr,[x1=a1,…,xn=an])**
- либо команду **subs** в виде **subs([x1=a1,…,xn=an],expr)** или **subs(x1=a1,…,xn=an,expr)**

```
> g := sin(x) + cos(y) :
> eval(g, [x = π/3, y = π/6])
```

$$\sqrt{3}$$

```
> subs(x = π/3, y = π/6, g)
```

$$\sin\left(\frac{1}{3}\pi\right) + \cos\left(\frac{1}{6}\pi\right)$$

```
> subs([x = π/3, y = π/6], g)
```

$$\sin\left(\frac{1}{3}\pi\right) + \cos\left(\frac{1}{6}\pi\right)$$

```
> simplify(%)
```

$$\sqrt{3}$$

Несколько особенностей

```
> restart
> f := sin(x) + x^2 : x := 2 : f
```

$$\sin(2) + 4$$

Пытаемся применить eval и subs при присвоенном значении x

```
> eval(f, x = π/4)
```

$$\frac{1}{2}\sqrt{2} + 4$$

```
> eval(f, 2 = π/4); eval(f, 2 = π)
```

$$\frac{1}{2}\sqrt{2} + 4$$

$$4$$

```
> subs(x = π/4, f);
```

$$\sin\left(\frac{1}{4}\pi\right) + 4$$

```
> unassign('x') : eval(f, x = π/4)
```

$$\frac{1}{2}\sqrt{2} + \frac{1}{16}\pi^2$$

```
> subs(x = 3, f);
```

$$\sin(3) + 9$$

2. Определение функции как функционального оператора

- Определение функции с помощью оператора-стрелки
- Обращение к пользовательской функции происходит как к команде Maple

Символ → в режиме ввода 2D Math-input автоматически преобразуется из двух введенных с клавиатуры символов ->.

$$> \quad f := x \to \sin(x) + x^2$$

$$f := x \to \sin(x) + x^2$$

$$> \quad f\left(\frac{\pi}{4}\right); f(10);$$

$$\frac{1}{2}\sqrt{2} + \frac{1}{16}\pi^2$$

$$\sin(10) + 100$$

$$> \quad g := (x, y) \to \sin(x) + \cos(y)$$

$$g := (x, y) \to \sin(x) + \cos(y)$$

$$> \quad g\left(\frac{\pi}{3}, \frac{\pi}{6}\right)$$

$$\sqrt{3}$$

3. Преобразование выражения в функциональный оператор

- С помощью команды **unapply(expr,x1,x2,…)**, где **expr** – выражение, **x1,x2,…** – набор переменных, от которых оно зависит, можно преобразовать *выражение* **expr** в *функциональный оператор*.

```
> restart
> f1 := sin(x) + x^2 :
> f := unapply(f1, x)
```

$$f := x \to \sin(x) + x^2$$

```
> f(π/4)
```

$$\frac{1}{2}\sqrt{2} + \frac{1}{16}\pi^2$$

```
> g1 := sin(x) + cos(y) :
> g := unapply(g1, x, y)
```

$$g := (x, y) \to \sin(x) + \cos(y)$$

```
> g(π/3, π/6)
```

$$\sqrt{3}$$

Определение кусочно-непрерывной функции

В *Maple* имеется возможность определения неэлементарных функций вида

$$f(x)=\begin{cases} f_1(x), x<a_1 \\ f_2(x), a_1<x<a_2 \\ \ldots\ldots\ldots\ldots \\ f_n(x), x>a_n \end{cases}$$

с помощью команды

`piecewise(cond_1,f_1,cond_2,f_2,...,cond_n,f_n,f_otherwise)`

- В 2D-Math Input можно использовать шаблон из палитры Expression $\begin{cases} -x & x<a \\ x & x\geq a \end{cases}$
- Можно задавать как выражения, так и функциональные операторы

Выражение Maple

```
> f1 := piecewise(0 < x, x)
```

$$f1:=\begin{cases} x & 0<x \\ 0 & otherwise \end{cases}$$

```
> eval(f1, x=-5); eval(f1, x=3.5)
```

$$0$$

$$3.5$$

Функциональный оператор, использование логического оператора

```
> f2 := x -> piecewise(x < -1, x, -1 <= x and x < 1, -x^2, x >= 1, -x)
```

$$f2 := x \to piecewise\left(x < -1, x, -1 \le x \text{ and } x < 1, -x^2, 1 \le x, -x\right)$$

```
> f2(-10); f2(0.5); f2(3/2)
```

$$-10$$

$$-0.25$$

$$-\frac{3}{2}$$

Для корректной работы функции piecewise в Maple 11 лучше записывать все неравенства в одну сторону (знак меньше)

```
> f1 := piecewise(-x^2  x < 0,
                  x^4   x >= 5)
```

$$f1 := \begin{cases} -x^2 & x < 0 \\ x^4 & 5 \le x \end{cases}$$

```
> eval(f1, x=-2); eval(f1, x=2); eval(f1, x=5)
```

$$-4$$

$$0$$

$$625$$

Решение уравнений и неравенств

- Аналитическое решение алгебраических уравнений и неравенств, команда solve
- Специальные опции для команды solve
- Численное решение уравнений и другие специальные команды
- Поиск корней

Решение алгебраических уравнений: solve

solve(equations, variables) – решение уравнения (системы уравнений) **equations** относительно переменной (набора переменных) **variables**
solve (eq) – набор переменных можно не указывать

> $solve(x+5)$

$$-5$$

> $solve(x^2 - x = 2025, x)$

$$\frac{1}{2} + \frac{1}{2}\sqrt{8101},\ \frac{1}{2} - \frac{1}{2}\sqrt{8101}$$

> $solve(2y - x^2 = 4, y)$

$$\frac{1}{2}x^2 + 2$$

Решить квадратное уравнение и найти сумму корней

> $s := solve(x^2 - 5x + 6)$

$$s := 3, 2$$

> $s[1] + s[2]$

$$5$$

Решение систем уравнений и присваивание решений переменным (команда assign)

- Набор уравнений и набор неизвестных могут задаваться в виде списка [] или множества{}, обычно набор уравнений – в {}
- Для присваивания найденных корней переменным, применяется команда **assign**

Результат - последовательность множеств

$> s := solve(\{x^2 - 5x + 6\})$

$$s := \{x = 3\}, \{x = 2\}$$

$> solve(\{x^2 y^2 = 0, x - y = 1, x \neq 0\})$

$$\{x = 1, y = 0\}, \{x = 1, y = 0\}$$

$> restart$

$> s1 := solve([x + y = 1, y - x = 10], [y, x])$

$$s1 := \left[\left[y = \frac{11}{2}, x = -\frac{9}{2}\right]\right]$$

$> assign(s1); x, y, x + y$

$$-\frac{9}{2}$$

$$\frac{11}{2}$$

$$1$$

Примеры: решение относительно присвоенных переменных (ошибка!), проверка решения

```
> s2 := solve( {x + y = 1, x - y = 10}, {y, x})
Warning, solving for expressions other than names or functions is not
recommended.
Error, (in solve) a constant is invalid as a variable, -9/2, 11/2
```

Два различных способа отмены присваивания

```
> unassign('x'); y :='y';
```

$$y := y$$

```
> s2 := solve( {x + y = 1, x - y = 10}, {y, x})
```

$$s2 := \left\{x = \frac{11}{2}, y = -\frac{9}{2}\right\}$$

```
> assign(s2); x - y
```

$$10$$

Можно не указывать набор неизвестных

```
> eqs := {u + v + w = 1, 3 u + v = 3, u - 2 v - w = 0} :
> sls := solve(eqs)
```

$$sls := \left\{u = \frac{4}{5}, v = \frac{3}{5}, w = -\frac{2}{5}\right\}$$

Проверка

```
> subs(sls, eqs)
```

$$\{0 = 0, 1 = 1, 3 = 3\}$$

Решение неравенств

- Аналогично решению уравнений, используется команда solve
- Можно получить решение в виде интервального множества или ограничения по искомой переменной

Решение в виде интервала

$> solve\left(\sqrt{x+3} < \sqrt{x-1} + \sqrt{x-2}, x\right)$

$$RealRange\left(Open\left(\frac{2}{3}\sqrt{21}\right), \infty\right)$$

$> solve\left(5 \le x^2 + x, x\right)$

$$RealRange\left(-\infty, -\frac{1}{2} - \frac{1}{2}\sqrt{21}\right), RealRange\left(-\frac{1}{2} + \frac{1}{2}\sqrt{21}, \infty\right)$$

Решение в виде ограничения по искомой переменной

$> solve\left(\sqrt{x+3} < \sqrt{x-1} + \sqrt{x-2}, \{x\}\right)$

$$\left\{\frac{2}{3}\sqrt{21} < x\right\}$$

Решение системы неравенств

$> solve\left(\{x + y \ge 2, x - 2\cdot y \le 1, x - y \ge 0, x - 2\cdot y \ge 1\}, \{x, y\}\right)$

$$\left\{y = \frac{1}{2}x - \frac{1}{2}, \frac{5}{3} \le x\right\}$$

$> solve\left(\{x + y < 10, x^2 = 9\}, [x, y]\right)$

$$[[x = 3, y < 7], [x = -3, y < 13]]$$

Специальные опции для команды solve: решение тригонометрических уравнений

- Команда **solve**, примененная для решения тригонометрического уравнения, выдает только главные решения, то есть решения в интервале [0,2π]. Для того, чтобы получить все решения, следует предварительно ввести дополнительную опцию (будет действовать до команды restart) **_EnvAllSolutions:=true**

В 2D Math при ввод символа _ может осуществляться переход в нижний регистр, если это происходит, следует набирать _
В 1D Math-input символ _ набирается обычным образом.

> $solve(\sin(x) = \cos(x), x)$

$$\frac{1}{4}\pi$$

$$_EnvAllSolutions := true :$$

> $solve(\sin(x) = \cos(x), x)$

$$\frac{1}{4}\pi + \pi _Z2\sim$$

Специальные опции для команды solve: представление решений в явном виде

- Если при решении сложных уравнений со степенями или систем уравнений в ответе появляется функция **RootOf,** это значит, что Maple не может выразить корни в радикалах

$$> \; eq := x^4 - 2\cdot x^3 + 2 = 0 :$$

$$> \; solve(eq, x)$$

$$\mathrm{RootOf}\left(_Z^4 - 2\,_Z^3 + 2,\ index = 1\right),\ \mathrm{RootOf}\left(_Z^4 - 2\,_Z^3 + 2,\ index = 2\right),$$
$$\mathrm{RootOf}\left(_Z^4 - 2\,_Z^3 + 2,\ index = 3\right),\ \mathrm{RootOf}\left(_Z^4 - 2\,_Z^3 + 2,\ index = 4\right)$$

- В этом случае (для представления решения в явном виде) можно использовать следующие способы:

1) перед командой **solve** следует ввести дополнительную опцию (будет действовать до команды restart) **_EnvExplicit:=true;**
2) использовать команду **allvalues** к результату решения (невычисляемому корню)
3) использовать команду **convert** с параметром **radical** к результату решения (невычисляемому корню)

Представление решений в явном виде: примеры

> $eq := x^4 - 2\cdot x^3 + 2 = 0$:

Решение не может быть представлено в явном виде

> $solve(eq, x)$

$$RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 1\right), RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 2\right), \quad (1)$$
$$RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 3\right), RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 4\right)$$

1) Использование опции _EnvExplicit

> $restart;\ _EnvExplicit := true$:

> $eq := x^4 - 2\cdot x^3 + 2 = 0 : solve(eq, x)$

$$\frac{1}{2} + \frac{1}{2}I + \frac{1}{2}\sqrt{4 - 2I}, \frac{1}{2} + \frac{1}{2}I - \frac{1}{2}\sqrt{4 - 2I}, \frac{1}{2} - \frac{1}{2}I + \frac{1}{2}\sqrt{4 + 2I}, \quad (2)$$
$$\frac{1}{2} - \frac{1}{2}I - \frac{1}{2}\sqrt{4 + 2I}$$

2) Использование команды allvalues

> $restart : eq := x^4 - 2\cdot x^3 + 2 = 0 : s := solve(eq, x)$:

> $allvalues(s)$ *#так будет ошибка*

```
Error, (in allvalues) redundant option RootOf(_Z^4-2*
_Z^3+2, index = 3), if there is more than one RootOf to
avoid, they should be put in a set or list
```

Ошибка: корни уравенения содержат более одной функции RootOf, нужно использовать список или множество.

Таким образом, можно применить allvalues к одному из корней

```
> s := solve(eq, x) : allvalues(s[1])
```

$$\frac{1}{2}+\frac{1}{2}I+\frac{1}{2}\sqrt{4-2I} \qquad (3)$$

Либо в команде solve задать список или множество корней

```
> ss := solve(eq, [x])
```

$$ss := \left[\left[x = RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 1\right)\right], \left[x = RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 2\right)\right], \left[x = RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 3\right)\right], \left[x = RootOf\left(_Z^4 - 2\,_Z^3 + 2, index = 4\right)\right]\right] \qquad (4)$$

```
> allvalues(%)
```

$$\left[\left[x=\frac{1}{2}+\frac{1}{2}I+\frac{1}{2}\sqrt{4-2I}\right], \left[x=\frac{1}{2}+\frac{1}{2}I-\frac{1}{2}\sqrt{4-2I}\right], \left[x=\frac{1}{2}-\frac{1}{2}I-\frac{1}{2}\sqrt{4+2I}\right], \left[x=\frac{1}{2}-\frac{1}{2}I+\frac{1}{2}\sqrt{4+2I}\right]\right] \qquad (5)$$

3) Использование команды convert с параметром radical

```
> restart;
> eq := x^4 - 2·x^3 + 2 = 0 : s := solve(eq, x) :
> convert(s[1], radical)
```

$$\frac{1}{2}+\frac{1}{2}I+\frac{1}{2}\sqrt{4-2I} \qquad (6)$$

```
> ss := solve(eq, [x]) : convert(ss, radical)
```

$$\left[\left[x=\frac{1}{2}+\frac{1}{2}I+\frac{1}{2}\sqrt{4-2I}\right], \left[x=\frac{1}{2}+\frac{1}{2}I-\frac{1}{2}\sqrt{4-2I}\right], \left[x=\frac{1}{2}-\frac{1}{2}I-\frac{1}{2}\sqrt{4+2I}\right], \left[x=\frac{1}{2}-\frac{1}{2}I+\frac{1}{2}\sqrt{4+2I}\right]\right] \qquad (7)$$

Численное решение уравнений и другие специальные команды

fsolve(eq) – численное решение уравнения, поиск вещественных корней
isolve(eq) – целочисленное решение уравнения
rsolve(eq) – решение рекуррентного уравнения

Численное решение уравнения, которое не имеет аналитического решения

```
> fsolve(cos(x) = x)
```

$$0.7390851332$$

```
> solve(cos(x) = x)
```

$$RootOf(_Z - \cos(_Z))$$

```
> evalf(solve(cos(x) = x))
```

$$0.7390851332$$

Поиск целочисленного решения. При отсутствии целочисленного решения выдается NULL (без вывода на экран)

```
> isolve({3 x - 4 y = 7, x + y = 14})
```

$$\{x = 9, y = 5\}$$

```
> isolve({4 x - y = 7, x + 2 y = 8})
> solve({4 x - y = 7, x + 2 y = 8})
```

$$\left\{x = \frac{22}{9}, y = \frac{25}{9}\right\}$$

Пример рекуррентного уравнения

```
> rsolve(f(n) = -3 f(n - 1) - 2 f(n - 2), f(k))
```

$$(-f(0) - f(1))\,(-2)^k + (2 f(0) + f(1))\,(-1)^k$$

Поиск вещественных корней с помощью fsolve

fsolve(eq, x=a..b) – поиск вещественных корней на интервале [a,b]

```
> eqn:=x^3+3-exp(x)=0;
```

$$eqn := x^3 + 3 - e^x = 0$$

```
> solve(eqn,x);
```

$$\text{RootOf}\left(-e^{-Z} + _Z^3 + 3\right)$$

Не всегда можно найти корни в явном виде

```
> _EnvExplicit:=true:solve(eqn,x);
```

$$\text{RootOf}\left(-e^{-Z} + _Z^3 + 3\right)$$

Численное решение

```
> fsolve(eqn);
```

4.623116246

Все ли это корни?

fsolve без параметров находит только один корень

```
> fsolve(eqn);
```

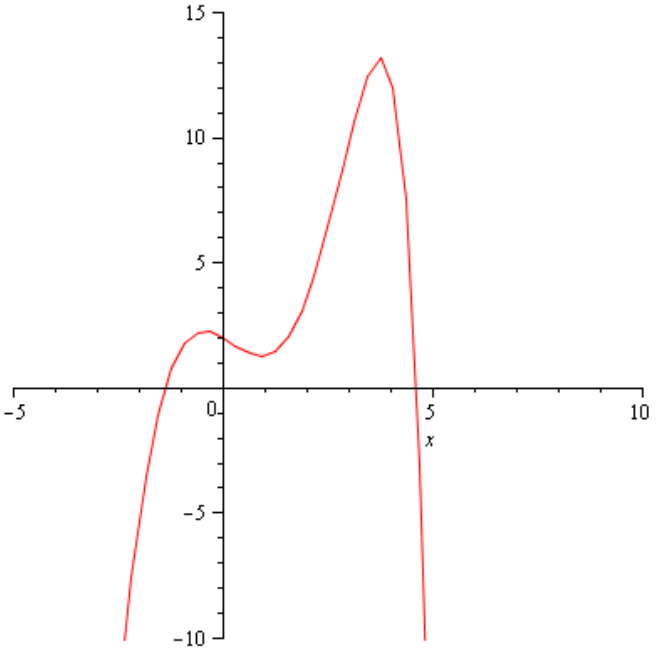
4.623116246

Первый вещественный корень

```
> fsolve(eqn,x=-2..-1);
```

−1.401667024

```
> plot(lhs(eqn),x=-5..10,-10..15);
```



Второй вещественных корень

```
> fsolve(eqn,x=-4..5);
```

4.623116246

Других вещественных корней нет

```
> fsolve(eqn,x=-1..4);
```

$$fsolve\left(x^3 + 3 - e^x = 0, x, -1..4\right)$$

Поиск корней с помощью пакета RootFinding

with(RootFinding):
Analytic(eq, re=a..b, im=c..d) – поиск корней в заданных интервалах для вещественной и мнимой части

```
> eqn:=x^3+3-exp(x)=0;
```

$$eqn := x^3 + 3 - e^x = 0$$

```
> with(RootFinding); Analytic(eqn, re = -10 .. 10,
  im = -10 .. 10);
```

[*Analytic*, *AnalyticZerosFound*, *BivariatePolynomial*, *Homotopy*, *Isolate*, *NextZero*]

$$7.3410916840555 - 8.9305861665020\,I,\ 4.6231162462848,$$
$$1.0348073103420 - 0.85239542694970\,I,\ -1.4016670238656,$$
$$1.0348073103420 + 0.85239542694965\,I,\ 7.3410916840555 + 8.9305861665020\,I$$