MAPLE

Комплексные числа

Maple может работать с комплексными числами. Мнимая единица в комплексном числе (корень квадратный из -1) обозначается зарезервированным символом ***I (заглавная латинская И)***. Функции **Re(x)** и **Im(x)** возвращают действительную и мнимую части комплексных чисел.

```
> 1.25+Pi*I;
                      1.25 + I π
> Re(1.25+Pi*I);
                         1.25
> Im(1.25+Pi*I);
                           π
>
```

Контроль за числами. Функция ***type(x, numeric)*** позволяет выяснить, является ли ***x*** числом. Если является, то она возвращает логическое значение true (истина), а если нет, то false (ложь).

```
> type(2,numeric);
                        true
> type(2.6,numeric);
                        true
> type(Pi,numeric);
                        false
> type(I,numeric);
                        false
> type(3/7,numeric);
                        true
> type(3^7,numeric);
                        true
> type(x^2,numeric);
                        false
```

Функции ***type(x, integer), type(x, rational)*** и ***type(x, fraction)*** можно использовать для проверки того, имеет ли ***x*** значение соответственно целого числа, рационального числа или простой дроби.

```
> type(123,integer);
                        true
> type(123.,integer);
                        false
> type(123/456,rational);
                        true
> type(1./3,rational);
                        false
> type(1/2,fraction);
                        true
> type(0.5,fraction);
                        false
```

MAPLE

Преобразования чисел с разным основанием

В Maple возможна работа с числами, имеющими различное основание. Помимо десятичных чисел (основание 10 – decimal) используются:

- двоичные (основание 2 — binary),
- восьмеричные (основание 8 — octal),
- шестнадцатеричные (основание 16 — hex).

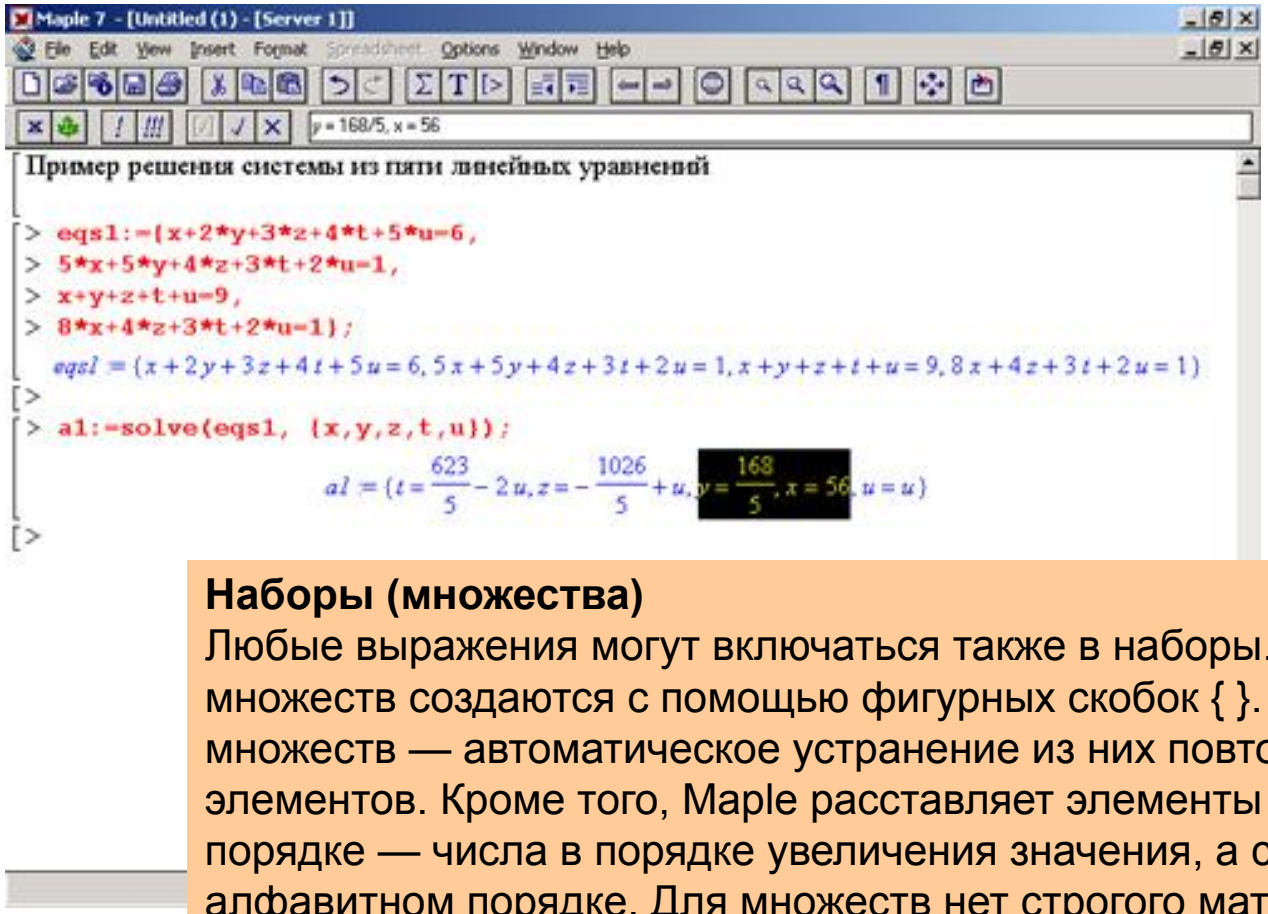
Функция ***convert*** позволяет легко преобразовывать форматы чисел.

```
> convert(12345,binary);
11000000111001
> convert(%,decimal,binary);
12345
> convert(12345,octal);
30071
> convert(123456,hex);
1E240
> convert(%,decimal,hex);
123456
```

Эта функция может преобразовывать в числа и строковые выражения.

```
> convert("11001111",decimal,binary);
207
> convert("1AF.C",decimal,hex);
431.7500000
> convert("Maple",decimal,36);
37451282
```

MAPLE



Наборы (множества)

Любые выражения могут включаться также в наборы. Такие наборы в виде множеств создаются с помощью фигурных скобок { }. Отличительная черта множеств — автоматическое устранение из них повторяющихся по значению элементов. Кроме того, Maple расставляет элементы множеств в определенном порядке — числа в порядке увеличения значения, а символы и строки в алфавитном порядке. Для множеств нет строгого математического определения, и мы будем считать их наборами, удовлетворяющими перечисленным выше признакам. С помощью множеств, в частности, задаются системы уравнений.

MAPLE Списки выражений

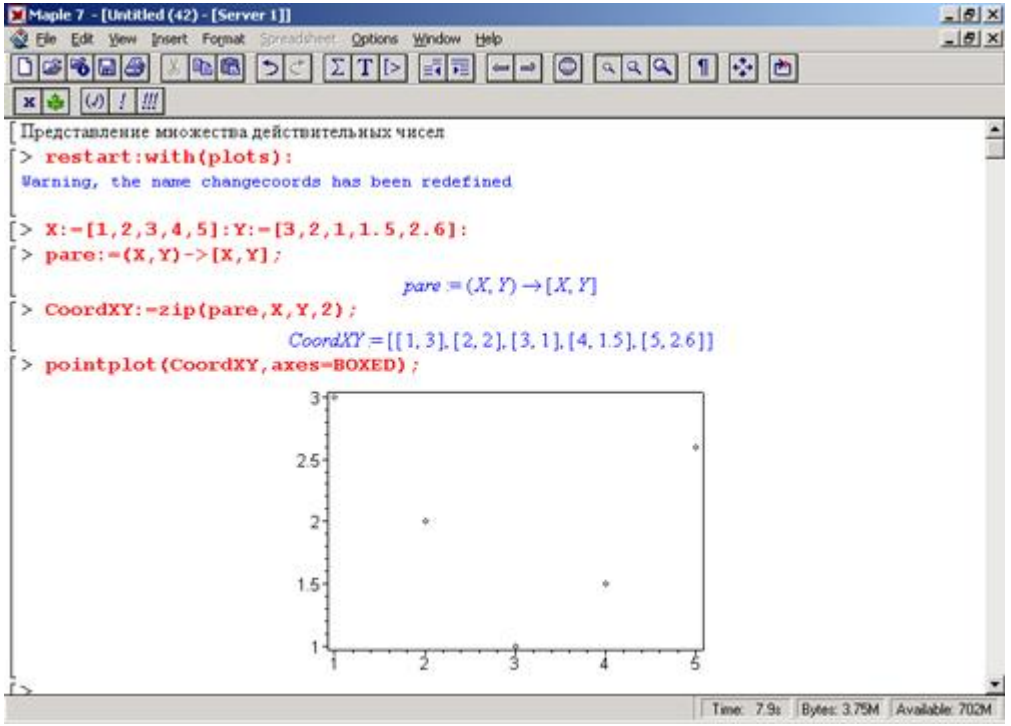
Для создания упорядоченных наборов — списков — служат квадратные скобки [ ]:

```
> [10, 2+3, 4+4, 8, 5, 1]:
[10, 5, 8, 8, 5, 1]
```

Элементы списков преобразуются и выводятся строго в том порядке, в каком они были заданы. Списки широко применяются для задания векторов и матриц.

В ряде случаев, например при подготовке данных для двумерных графиков, возникает необходимость в подготовке парных списков — скажем, координат точек (x, y) графика. Для этого можно использовать функцию ***zip(f, u, v)*** или ***zip(f, u, v, d)***. Здесь ***f*** — бинарная функция, ***u, v*** — списки или векторы, ***d*** — необязательный параметр.

```
> X:=[1,2,3,4,5]:Y:=[3,2,1,1.5,2.5]:
> pare:=(X,Y)->[X,Y];
pare := (X, Y) → [X, Y]
> CoordXY:=zip(pare,X,Y,2);
CoordXY := [[1, 3], [2, 2], [3, 1], [4, 1.5], [5, 2.5]]
> zip((X,Y)->X+Y,X,Y);
[4, 4, 4, 5.5, 7.5]
```



MAPLE

Для создания векторов (одномерных массивов) и матриц (двумерных массивов) служит функция ***array***. Обычно она используется в следующих формах:

- ***array[a. .b,s1]*** — возвращает вектор с индексами от a до b и значениями в одномерном списке s1;
- ***array[a. .b,c. .d,s2]*** — возвращает матрицу с номерами строк от a до b, номерами столбцов от c до d и значениями в двумерном списке s2.

```
array(1..3,[x,y,x+y])- создает вектор с элементами x,y и x+y
```

```
array(1..2,1..2,[[a,b],[c,d]])
```
- квадратная матрица

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Двумерные списки часто путают с матрицами. **Векторы и матрицы создаются с помощью функции *array* и являются отдельным типом данных.** Элементами векторов и массивов могут быть константы, переменные, выражения, списки и иные объекты. Эти элементы являются индексированными переменными и их положение указывается индексами. Размерность массивов, создаваемых списками, не ограничена и массивы могут быть многомерными. Имеется множество функций для работы со списками, массивами и матрицами.

MAPLE

Таблицы

Еще одним важным типом множественных данных являются таблицы. Они задают данные с произвольной индексацией. Для создания таблиц служит функция ***table***, которая при вызове в простейшем виде ***table[]*** создает шаблон пустой таблицы.
Пустая таблица резервирует память под данные. Когда параметром функции ***table*** является список выражений, он выводится в естественном порядке расположения элементов таблицы, но с произвольным порядком индексации.

Можно выделить отдельные компоненты таблицы и вывести значения и индексы таблицы с помощью функций ***entries*** и ***indices***.

```
> table[];
```

$table_{[\,]}$

```
> T:=table({1,2,Pi,`string`});
```

$T := \mathrm{table}([1 = 1, 2 = 2, 3 = \pi, 4 = string])$

```
> T[3];
```

π

```
> S:=table([(one)=1,(two)=2,(three)=3]);
```

$S := \mathrm{table}([one = 1, two = 2, three = 3])$

```
> S[1];
```

S_1

```
> S[two];
```

2

```
> S[three];
```

3

```
> entries(S);
```

[1], [2], [3]

```
> indices(S);
```

[one], [two], [three]

MAPLE

Строки и комментарии

Строковые данные

Строки как тип данных — это просто цепочки символов. Они обычно используются для создания текстовых комментариев. Строки должны каким-либо образом выделяться, чтобы Maple не отождествляла их с именами констант и переменных. Для этого строки-комментарии имеют внутренний разделительный признак, который устанавливается при их вводе (нажатием клавиши F5, которое приводит к исчезновению знака >).
В других случаях последовательность символов рассматривается как строка, если она заключена в обратные апострофы, то есть в знаки '. Два апострофа подряд формируют апостроф как знак символьной строки, например **`abc``def`** дает строку **abc`def**. Любое математическое выражение может входить в строку и при этом не выполняется:

```
> '2+2 не всегда "четыре"';
2+2 не всегда 'четыре'
```

Неисполняемые программные комментарии

Часто возникает необходимость в задании программных комментариев. Любой текст после знака # рассматривается как невыводимый (неисполняемый) программный комментарий — даже если это математическое выражение. При этом он не вычисляется. Например:

```
> 2+3; #Это пример. А это выражение не вычисляется: 4+5
5
```

Комментарии полезны в программах на Maple-языке и обычно используются для объяснения особенностей реализованных алгоритмов.

MAPLE

Константы

Константы - это простейшие именованные объекты, несущие заранее предопределенные значения. Их имена (идентификаторы) также заранее определены и не могут меняться. Подробную информацию о константах можно найти, исполнив команду ?constant.

В выражениях часто встречаются литералы – это неименованные числовые константы. В выражении 2*sin(1.25) числа 2 и 1.25 являются числовыми литералами. Строковыми константами являются произвольные цепочки символов, заключенные в обратные апострофы, например 'Hello', 'Привет', 'My number‘. Числа, заключенные в апострофы, например '123456', также становятся строковыми константами, которые нельзя использовать в арифметических выражениях.

Встроенные в ядро константы

Есть также ряд констант, которые правильнее считать заведомо определенными глобальными переменными:

false — логическое значение «ложно»;
gamma — константа Эйлера, равная 0.5772156649...;
infinity — положительная бесконечность (отрицательная задается как **-infinity**);
true — логическое значение «истинно»;
Catalan — константа Каталана, равная 0.915965594...;
FAIL — специальная константа (см. справку, выдаваемую по команде ?FAIL);
I — мнимая единица (квадратный корень из -1);
Pi — представляет константу = 3.141...

В этот список не входит основание натурального логарифма — число **e**. В качестве этой константы рекомендуется использовать exp(1). Она отображается как жирная прямая буква E. А exp(1.) выводит 2.71828... (что и следовало ожидать).

Идентификация констант

Функции ***type(x, constant)*** и ***type(x, realcons)*** возвращают логическое значение true, если x представляет целочисленную или вещественную константу, и false, если x не является константой.

MAPLE

Переменные — это объекты, значения которых могут меняться по ходу выполнения документа. Тип переменной может быть задан директивно с помощью конструкции ***name::тип***, а также явно простым присвоением значения. Имена (идентификаторы) переменных – это последовательность допустимых символов алфавита, начинающаяся с буквы. Регистр букв учитывается.

Не допускается использовать в качестве имен переменных зарезервированные слова, имена предопределенных констант и функций. Чтобы проверить уникальность вводимого идентификатора можно выполнить команду **?name**, где name – выбранный идентификатор переменной.

По умолчанию любые переменные рассматриваются как объекты символьного типа. Благодаря этому такие переменные могут фигурировать в математических выражениях (таких, как ***sin(x)/x***) без их предварительного объявления – это не влечет за собой появления сообщений об ошибках и является более естественным. Для присваивания переменным конкретных значений используется комбинированный символ присваивания **«:=».**

Если значение переменной определено с помощью оператора присваивания, то такие переменные не могут использоваться в символьных вычислениях. Для придания статуса неопределенных переменных необходимо либо вызвать команду **restart** (тогда все ранее определенные переменные станут неопределенными), либо выполнить оператор ***x:='x';***, либо использовать функцию ***x:=evaln(x);***.

```
> x:=10;
x := 10
> x;
10
> int(x^2,x);
Error, (in int) wrong number (or type) of arguments
```

```
> x:='x';
x := x
> int(x^2,x);
1/3 x^3
```

```
> x:=123;
x := 123
> x:=evaln(x);
x := x
> int(x^n,x);
x^(n+1)/(n+1)
```

```
> x:=5;
x := 5
> x^2;
25
> restart;
> x;
x
> x^2;
x^2
```

MAPLE Операторы

Имеется пять основных типов операторов:
- **binary** — бинарные операторы (двумя операндами);
- **unary** — унарные операторы (с одним операндом);
- **nullary** — нульарные операторы (без операнда — это одна, две и три пары кавычек);
- **precedence** — операторы старшинства (включая логические операторы);
- **functional** — функциональные операторы.

```
> 2+3-(-4);
9
> [2^3,2**3];
[8,8]
> 7 mod 5;
2
> [3@2,3@@2];
[3,3^(2)]
> [x@x,x@@x];
[x^(2),x^(x)]
> [x$3,x$4];
[x,x,x,x,x,x,x]
> int(x^2,x=1..4);
21
> S:=`Hello`||`my`||`friend!`;
S := Hellomyfriend!
```

Бинарные операторы:

«+» - Сложение; «-» - Вычитание; «*» - Умножение; «/» - Деление; «**» или «^» - Возведение в степень; **«mod» - Остаток от деления; «$» - Оператор последовательности;** «.» - Разделительная точка; **«@» - Оператор композиции; «@@» - Повторение композиции;** «,» - Разделитель выражений; «:=» - Присваивание; **«..» - Задание интервала; «||» - Конкатенация (объединение) строк.**

MAPLE Операторы

```
> f:=a->1/(1+a);(f@@3)(a);
```

$$f := a \to \frac{1}{1+a}$$

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+a}}}$$

Оператор композиции @@ может использоваться для создания сложных функций, содержащих цепные дроби.

В примере используется функциональный оператор «->», задающий пользовательскую функцию.

```
> {a,a,b,c,c,d} union{e,e,f,g};
{f, g, a, b, c, d, e}
> {a,a,b,c,c,d} intersect {a,c,e,e,f,g};
{a, c}
> {a,a,b,c,c,d} minus {a,d};
{b, c}
```

Для данных типа «множество» определены следующие бинарные операторы:
union — включает первый операнд (множество) во второй;
intersect — создает множество, содержащее общие для операндов элементы;
minus — исключает из первого операнда элементы второго операнда.
В любом случае, в результирующем множестве устраняются повторяющиеся элементы.

Специальный оператор % обеспечивает подстановку в строку ввода (или в выражение) последнего результата операции, %% — предпоследнего и %%% — третьего с конца.

MAPLE Операторы

Логические операторы:

«<» - Меньше; «<=» - Меньше или равно; «>» - Больше; «>=» - Большее или равно; «=» - Равно; «<>» - Не равно; «and» - Логическое «и»; «or» - Логическое «или».

Результатом вычисления выражения с логическими операторами являются логические константы true или false, если оно не выполняется.

Кроме того, к логическим операторам относится **унарный оператор «not»** — он представляет логическое «нет».

Для возврата логических значений выражений с этими операторами используется функция ***evalb(условие)***.

```
> 5<2;
5 < 2
> evalb(%);
false
> evalb(4=2+2);
true
> evalb(3<>3);
false
> evalb(not(%));
true
> evalb(3=3 or 2<0);
true
> evalb(3=3 and 4>2);
true
> evalb(x*y=y*x);
true
```

Логические операторы часто используются в управляющих структурах программ, составленных на языке программирования Maple.

MAPLE Операторы

Специальные типы операторов:

- композиционные («@»);
- функциональные («var -> result» или «<result | var>»);
- неопределенные и инертные («&name»);
- заданные с помощью команды **define(name, свойство1, свойство2,….)**

```
(x,y) -> x^2 + y^2
x-> (2*x, 3*x^4)
(x,y,z) -> (x*y, y*z)
```

```
> restart;
> define(fib,fib(0)=1,fib(1)=1,fib(n::posint)=fib(n-1)+fib(n-2));
> fib(6);
13
> fib(10);
89
> fib(20);
10946
> time(fib(20));
1.672
```

time() – системная функция, показывающая время в секундах, затраченное на выполнение указанного в скобках оператора.

MAPLE Математические функции

Понятие о встроенных функциях

Maple имеет полный набор элементарных математических функций. Все они, кроме арктангенса двух аргументов, имеют один аргумент x, например sin(x). Он может быть целым, рациональным, дробно-рациональным, вещественным или комплексным числом. В ответ на обращение к ним элементарные функции возвращают соответствующее значение. Поэтому они могут быть включены в математические выражения. Все описанные здесь функции называются встроенными, поскольку они реализованы в ядре системы.

Целочисленные функции, используемые в теории чисел:
factorial(n) — функция вычисления факториала (альтернатива — оператор !);
iquo(a.b) — целочисленное деление a на b;
irem(a,b) — остаток от деления a на b;
igcd(a b) — наибольший общий делитель;
lcm(a,b) — наименьшее общее кратное.

```
> [factorial(10),10!];
[3628800, 3628800]
> iquo(234,5);
46
> irem(234,5);
4
> lcm(124,3);
372
> [3!!,(3!)!];
[720, 720]
```

MAPLE

Математические функции

В ядре Maple реализован полный набор **тригонометрических и гиперболических функций**, а также их обратных функций: **sin** — синус; **cos** — косинус; **tan** — тангенс; **sec** — секанс; **csc** — косеканс; **cot** — котангенс; **arcsin** — арксинус; **arccos** — арккосинус; **arctan** — арктангенс; **arcsec** — арксеканс; **arccsc** — арккосеканс; **arccot** — арккотангенс; **sinh** — гиперболический синус; **cosh** — гиперболический косинус; **tanh** — гиперболический тангенс; **sech** — гиперболический секанс; **csch** — гиперболический косеканс; **coth** — гиперболический котангенс; **arcsinh** — гиперболический арксинус; **arccosh** — гиперболический арккосинус; **arctanh** — гиперболический арктангенс; **arcsech** — гиперболический арксеканс; **arccsch** — гиперболический арккосеканс; **arccoth** — гиперболический арккотангенс.

К степенным и логарифмическим относятся следующие функции Maple: **exp** — экспоненциальная функция; **ilog10** — целочисленный логарифм по основанию 10 (возвращает целую часть от логарифма по основанию 10); **ilog** — целочисленный логарифм (библиотечная функция, возвращающая целую часть от натурального логарифма); **ln** — натуральный логарифм; **log** — логарифм по заданному основанию (библиотечная функция); **log10** — логарифм по основанию 10; **sqrt** — квадратный корень.

MAPLE Математические функции

Функции с элементами сравнения

В алгоритме вычисления ряда функций заложено сравнение результата с некоторым опорным значением. К таким функциям относятся:

abs — абсолютное значение числа;
ceil — наименьшее целое, большее или равное аргументу;
floor — наибольшее целое, меньшее или равное аргументу;
frac — дробная часть числа;
trunc — целое, округленное в направлении нуля;
round — округленное значение числа;
signum(x) — знак x (-1 при x < 0, 0 при x = 0 и +1 при x > 0).

Для комплексного аргумента x эти функции определяются следующим образом:

tranc(x) = trunc(Re(x)) + I*trunc(Im(x));
round(x) = round(Re(x)) + I*round(Im(x));
frac(x) - frac(Re(x)) + I*frac(Im(x)).
floor(x) = floor(Re(x)) + I*floor(Im(x)) + X, где
ceil(x) = -floor(-x)

$$X=\begin{cases} 0, & a+b<1, \\ 1, & a+b\geq 1 \text{ и } a\geq b, \\ I, & a+b\geq 1 \text{ и } a<b. \end{cases}$$

Специальные математические функции обычно являются решениями линейных дифференциальных уравнений различного типа и выражаются в виде интегралов, не представимых через элементарные функции. Maple имеет практически полный набор таких функций. Их представления можно найти в справочной базе данных Maple.

MAPLE

Функции работы с массивами, векторами и матрицами

Вызов элемента и присваивание ему значений:
V[i]:=x — присваивание нового значения x i-му элементу вектора V;
M[i,j]:=x — присваивание нового значения x элементу матрицы M.

Векторы и матрицы похожи на списки, но не полностью отождествляются с ними. Используя функцию преобразования данных ***convert***, можно преобразовывать одномерные списки в векторы, а двумерные списки — в матрицы.
Функция ***type(V, vector)*** — тестирует аргумент V и возвращает true, если V — вектор, и false в ином случае; ***type(M,matrix)*** — тестирует аргумент M и возвращает true, если M — матрица, и false в ином случае.

```
> M1:=[1,2,3,4];
```

$$M1 := [1, 2, 3, 4]$$

```
> type(M1,vector);
```

false

```
> V:=convert(M1,vector);
```

$$V := [1, 2, 3, 4]$$

```
> type(V,vector);
```

true

```
> M2:=[[1,2],[3,4]];
```

$$M2 := [[1, 2], [3, 4]]$$

```
> type(M2,matrix);
```

false

```
> M:=convert(M2,matrix);
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> type(M,matrix);
```

true

Для проведение вычислений с массивами и матрицами используется функция ***evalm(M)***

MAPLE

Функции работы с массивами, векторами и матрицами

Примеры матричных вычислений:

```
> V:=array(1..4,[1,2,3,4]);
```

$$V := [1, 2, 3, 4]$$

```
> [V[1],V[2],V[4]];
```

$$[1, 2, 4]$$

```
> V[1]:=a:V[3]:=b:
> evalm(V);
```

$$[a, 2, b, 4]$$

```
> evalm(V+2);
```

$$[a+2, 4, b+2, 6]$$

```
> evalm(2*V);
```

$$[2a, 4, 2b, 8]$$

```
> evalm(V**V);
```

$$[a, 2, b, 4]^{V}$$

```
> evalm(a*V);
```

$$[a^2, 2a, ab, 4a]$$

```
> M:=array(1..2,1..2,[[1,2],[3,4]]);
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> evalm(M-M);
```

$$0$$

```
> evalm(M+M);
```

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

```
> evalm(M*M);
```

$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

```
> evalm(M/M);
```

$$1$$

```
> evalm(M^0);
```

$$1$$

```
> M:=array(1..2,1..2,[[1,2],[3,4]]);
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> evalm(2*M);
```

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

```
> evalm(2+M);
```

$$\begin{bmatrix} 3 & 2 \\ 3 & 6 \end{bmatrix}$$

```
> evalm(M^2);
```

$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

```
> evalm(M^(-1));
```

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

MAPLE

Функции работы с массивами, векторами и матрицами

```
> M1:=array(1..2,1..2,[[a1,b1],[c1,d1]]);
```

$$M1 := \begin{bmatrix} a1 & b1 \\ c1 & d1 \end{bmatrix}$$

```
> M2:=array(1..2,1..2,[[a2,b2],[c2,d2]]);
```

$$M2 := \begin{bmatrix} a2 & b2 \\ c2 & d2 \end{bmatrix}$$

```
> evalm(M1+M2);
```

$$\begin{bmatrix} a1+a2 & b1+b2 \\ c1+c2 & d1+d2 \end{bmatrix}$$

```
> evalm(M1-M2);
```

$$\begin{bmatrix} a1-a2 & b1-b2 \\ c1-c2 & d1-d2 \end{bmatrix}$$

```
> evalm(M1&*M2);
```

$$\begin{bmatrix} a1\,a2+b1\,c2 & a1\,b2+b1\,d2 \\ c1\,a2+d1\,c2 & c1\,b2+d1\,d2 \end{bmatrix}$$

```
> evalm(M1/M2);
```

$$\begin{bmatrix} -\frac{a1\,d2}{-a2\,d2+b2\,c2}+\frac{b1\,c2}{-a2\,d2+b2\,c2} & \frac{a1\,b2}{-a2\,d2+b2\,c2}-\frac{b1\,a2}{-a2\,d2+b2\,c2} \\ -\frac{c1\,d2}{-a2\,d2+b2\,c2}+\frac{d1\,c2}{-a2\,d2+b2\,c2} & \frac{c1\,b2}{-a2\,d2+b2\,c2}-\frac{d1\,a2}{-a2\,d2+b2\,c2} \end{bmatrix}$$

```
> evalm(M1^2);
```

$$\begin{bmatrix} a1^2+b1\,c1 & a1\,b1+b1\,d1 \\ c1\,a1+d1\,c1 & b1\,c1+d1^2 \end{bmatrix}$$

```
> evalm(sin(M1));
```

$$\begin{bmatrix} \sin(a1) & \sin(b1) \\ \sin(c1) & \sin(d1) \end{bmatrix}$$

```
> evalm(M1*z);
```

$$\begin{bmatrix} z\,a1 & z\,b1 \\ z\,c1 & z\,d1 \end{bmatrix}$$

```
> evalm(M1/z);
```

$$\begin{bmatrix} \frac{a1}{z} & \frac{b1}{z} \\ \frac{c1}{z} & \frac{d1}{z} \end{bmatrix}$$

```
> evalm(M1+z);
```

$$\begin{bmatrix} a1+z & b1 \\ c1 & d1+z \end{bmatrix}$$

```
> evalm(M1-z);
```

$$\begin{bmatrix} a1-z & b1 \\ c1 & d1-z \end{bmatrix}$$

Примеры символьных вычислений с матрицами.

Функция ***map(fname, M, x)*** позволяет выполнить функцию fname над всеми элементами матрицы.

```
> M:=array(1..2,1..2,[[x,x^2],[x^3,x^4]]);
```

$$M := \begin{bmatrix} x & x^2 \\ x^3 & x^4 \end{bmatrix}$$

```
> map(diff,M,x);
```

$$\begin{bmatrix} 1 & 2x \\ 3x^2 & 4x^3 \end{bmatrix}$$

```
> map(int,%,x);
```

$$\begin{bmatrix} x & x^2 \\ x^3 & x^4 \end{bmatrix}$$

MAPLE

Работа со строковыми данными

Обработка строк

Имеется ряд функций для работы со строками. Из них наиболее важны следующие:

lenght(str) — возвращает число символов, содержащихся в строке str;

substring(str, a, b) — возвращает подстроку строки str от a-го символа до b-го;

cat(strl,str2....) — возвращает строку, полученную объединением строк str1, str2,... (альтернатива — оператор конкатенации в виде точки .);

SearchText(s, str) — производит поиск подстроки s в строке str и при его успехе возвращает номер позиции s в строке str (при отсутствии s в str функция возвращает 0).

```
> length(str);
6
> substring(str,1..3);
Hel
> substring(str,4..6);
lo!
> s:=cat(`Hello`,` my`,` friend!`);
s := Hello my friend!
> SearchText(my,s);
7
> ss:=`Hello`||`my friend!`;
ss := Hellomy friend!
> seq(Name||i,i=1..4);
Name1, Name2, Name3, Name4
```

```
> y:=readstat(`Введите выражение `);
Введите выражение a*x^2+b;
y := a x^2 + b
```

Интерактивный ввод математических выражений с запросом:
функция ***readstat(promt)***, где promt — строка с текстовым комментарием.

```
> s:=`2+3*5`;
s := 2+3*5
> evaln(s);
s
> parse(%);
17
```

Ввод строкового выражения с последующим преобразованием его в математическое выражение с помощью функции ***parse***

MAPLE

Пользовательские функции и средства программирования

Для повышения эффективности работы с Maple возникает необходимость в создании собственных функций.

Самый простой способ – присвоить некоторой переменной выражение и в дальнейшем его использовать, как функцию.

```
> m:=sqrt(x^2+y^2);
```

$$m := \sqrt{x^2+y^2}$$

```
> x:=3:y:=4:m;
```

$$\sqrt{25}$$

```
> evalf(m);
```

5.000000000

Заданный таким образом объект все же не является полноценной функцией пользователя, поскольку в нем используются только глобальные переменные (x и y) и нет объявленного списка параметров, от которых зависит значение функции. При этом значения переменных функции приходится заведомо задавать отдельно, используя операции присваивания.

Для задания полноценных функций пользователя применяется функциональный оператор. При этом используется следующая конструкция: ***name:=(x,y,...)->expr.***

После этого вызов функции осуществляется в виде name(x.y,...), где (x. y,...) — список формальных параметров функции пользователя с именем name. Переменные, указанные в списке формальных параметров, являются локальными. При подстановке на их место фактических параметров они сохраняют их значения только в теле функции (expr). За пределами этой функции переменные с этими именами оказываются либо неопределенными, либо сохраняют ранее присвоенные им значения.

```
> restart;
> x:=0;y:=0;
```

$x := 0$

$y := 0$

```
> m:=(x,y)->sqrt(x^2+y^2);
```

$$m := (x,y) \rightarrow \sqrt{x^2+y^2}$$

```
> m(3,4);
```

5

```
> m(3.,4);
```

5.000000000

```
> [x,y];
```

[0,0]

MAPLE

Пользовательские функции и средства программирования

Еще один способ задания функции пользователя базируется на применении функции unapply:
name:=unapply(expr,var1,var2,...)

При таком задании функции пользователя, помимо численных, можно проводить и символьные вычисления.

```
> restart;
> fm:=unapply(sqrt(x^2+y^2),x,y);
```

$$fm := (x,y) \to \sqrt{x^2+y^2}$$

```
> fm(4.,3.);
```

5.000000000

```
> fe:=unapply(x^2+y^2,x,y);
```

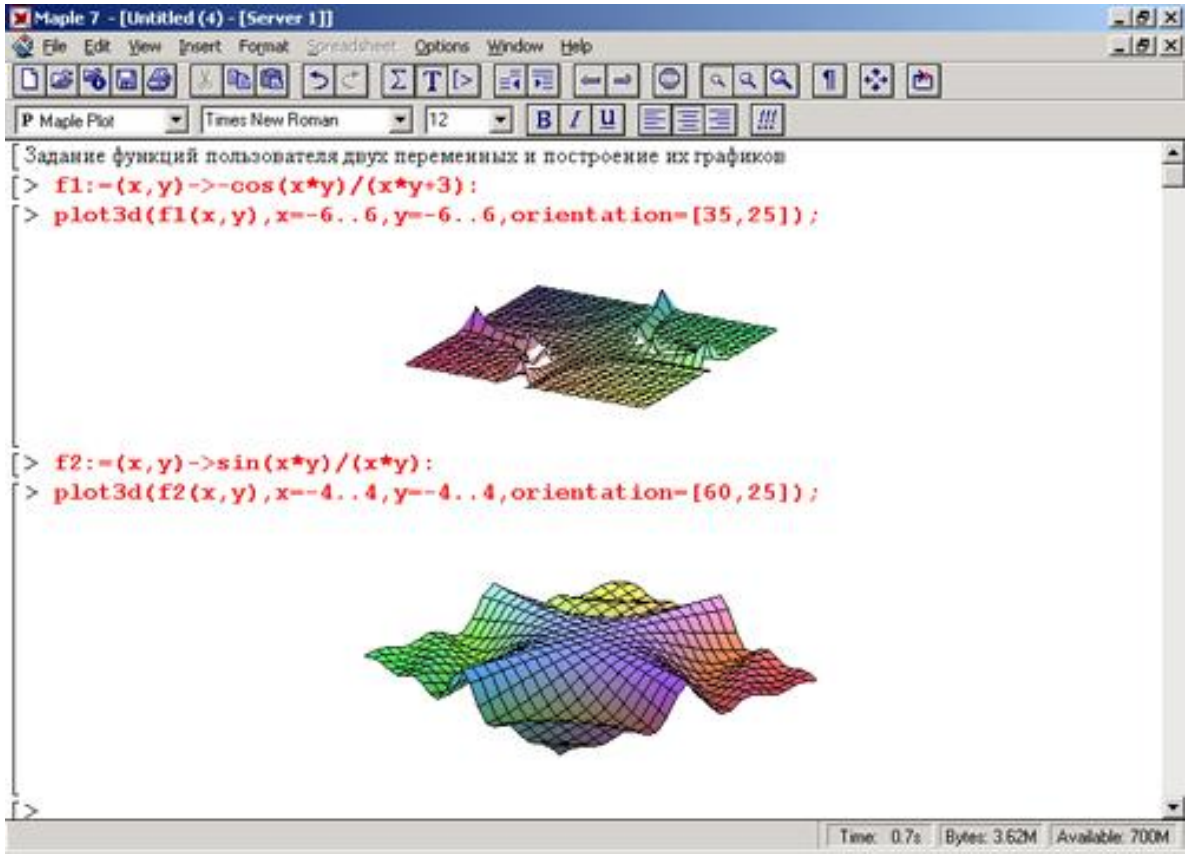
$$fe := (x,y) \to x^2+y^2$$

```
> fe(sin(x),cs(x));
```

$$\sin(x)^2 + cs(x)^2$$

```
> simplify(fe(sin(x),cos(x)));
```

1



Для графической визуализации заданных функций, как правило используются функции ***plot*** и ***plot3d***

MAPLE

Пользовательские функции и средства программирования

Важный класс функций – импликативные, в которых связь между переменными задана неявно, в виде какого-либо выражения. Самый характерный пример такой функции — это выражение для задания окружности радиуса r:

x^2 + y^2 = r^2.

Импликативные функции записываются как уравнения. Соответственно их можно решать с помощью функции ***solve***.

```
> impf:=x^2+y^2=r^2;
```

$$impf := x^2 + y^2 = r^2$$

```
> subs(x=a,impf);
```

$$a^2 + y^2 = r^2$$

```
> solve(%);
```

$$\{a = \sqrt{-y^2 + r^2}, y = y, r = r\}, \{a = -\sqrt{-y^2 + r^2}, y = y, r = r\}$$

```
> impf1:=x^2+y^2=25;
```

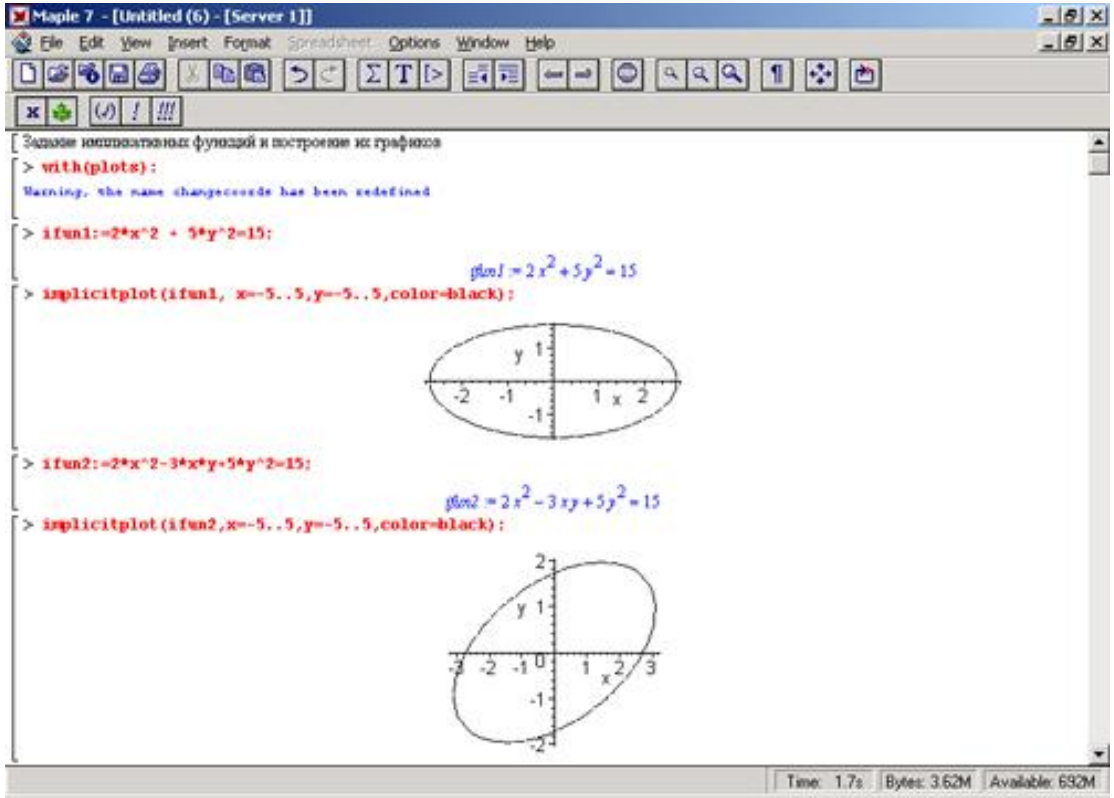
$$impf1 := x^2 + y^2 = 25$$

```
> subs(x=4,impf1);
```

$$16 + y^2 = 25$$

```
> solve(%);
```

$$3, -3$$



Визуализация импликативных функций осуществляется функцией ***implicitplot*** пакета ***plots***

Пользовательские функции и средства программирования

Как и любой другой язык программирования Maple-язык содержит **набор операторов**, с помощью которых можно организовывать сложные вычисления. К ним относятся:

- **Условный оператор**: **if** (условие) **then** *Элементы_1* **else** *Элементы_2* ***fi;***
- **Оператор цикла**: **for** <name> **from** <expr1> **to** <expr3> **by** <expr2> **while** <expr4> **do** <Выполняемые элементы> **od;** и его различные модификации, в том числе упрощенный цикл **while** <expr4> **do** <Выполняемые элементы> **od;**

Процедуры

Можно задать некоторую последовательность действий в виде отдельной процедуры:

name := proc(Формальные_параметры) Тело процедуры end;

Вызов осуществляется в любом месте: ***name(Фактические параметры);***

Используя оператор ***return(Значение),*** можно явно указать значение, возвращаемое процедурой. Если ***return*** опущен, то процедура возвращает значение, полученное в последнем выражении тела.

```
> modc:=proc(z)
> evalf(sqrt(Re(z)^2+Im(z)^2))
> end;
modc := proc(z) evalf(sqrt(ℜ(z)^2 + ℑ(z)^2)) end proc
```

```
> modc:=proc(z)
> evalf(sqrt(Re(z)^2+Im(z)^2)):
> RETURN(%)
> end;
modc := proc(z) evalf(sqrt(ℜ(z)^2 + ℑ(z)^2)); RETURN(%) end proc
> modc(3.+I*4);
5.000000000
```

MAPLE