Введение в *Mathematica*

25-янв-2014

Ниже описаны базовые функции и приемы работы в *Mathematica* 9.

Начало работы

Mathematica включает в себя редактор (FrontEnd) [*вы в нем сейчас и читаете этот текст*] и вычислительное ядро (Kernel).

Ячейки

Визуально каждый файл состоит из набора ячеек (синие скобки в правой части). Ячейка создается автоматически когда вы начинаете вводить текст. Какждая ячейка может содержать либо выражение, либо его вывод, либо текст. Стиль ячейки можно выбрать в меню Format > Style.

Основные стили ячеек (вы можете видеть их всех в этом файле):

- Input — выражение для вычисления
- Output — вывод результата вычисления
- Title — заголовок файла
- Section — секция файла

Ячейки можно удалять, переносить, окрашивать в разные цвета и т. д. Все основные операции вы найдете в меню Format и Cell.

Вычисления

Для вычисления выражения нужно стать в его ячейку и нажать SHIFT+ENTER. Ниже сразу же появится ячейка с результатом вычислений.

```
In[1]:= 2 + 2
Out[1]= 4
```

Часто бывает удобно вводить несколько выражений в одну ячейку. Например, создадим матрицу и вычислим ее определитель.

```
In[2]:= Range[9]
        Partition[%, 3]
        Det[%]
Out[2]= {1, 2, 3, 4, 5, 6, 7, 8, 9}
Out[3]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
Out[4]= 0
```

Знак процента (%) означает последний результат. В данном случае мы сначала создали упорядоченный список от 1 до 9, потом разбили его на три части (то есть получили матрицу), а потом вычислили определитель. Применение процента позволяет видеть промежуточные результаты. Иначе все можно записать в одну строку.

In[5]:=
```
Det[Partition[Range[9], 3]]
```
Out[5]= 0

Чтобы не видеть промежуточные результаты, нужно поставить в конце выражения точку с запятой.

In[6]:=
```
Range[9];
Partition[%, 3];
Det[%]
```
Out[8]= 0

Конечно, матрицу можно было бы сохранить в переменную, а уже потом подставить переменную в функцию Det.

In[9]:=
```
m = Partition[Range[9], 3]
```
Out[9]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}

In[10]:=
```
Det[m]
```
Out[10]= 0

Для быстрого прекращения затянувшегося вычисления можно выполнить Evaluaion > Abort Evaluation или нажать ALT+точка.

In[11]:=
$$\text{Integrate}\left[\frac{x}{\text{Cos}[x - \text{Tan}[x]]}, x\right]$$

Out[11]=
$$\int x \, \text{Sec}[x - \text{Tan}[x]] \, dx$$

Вычислительная сессия

Все переменные, как обычно, сохраняются в память пока вы работаете с программой. Но в *Mathematica* есть понятие вычислительной сессии. Обычно сессия начинается при запуске программы и заканчивается при ее закрытии. Однако, вы можете прекратить сессию когда угодно — для этого нужно выбрать меню Evaluation > Quit Kernel > Local. При этом *Mathematica* “забудет” значения всех сохраненных переменных. Это бывает полезно, если посыпались ошибки и вам нужно пересчитать все заново.

Закончите сессию прямо сейчас и проверьте чему равно значени m

In[12]:=
```
m
```
Out[12]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}

Чтобы m получило свое значение нужно перейти в ячейку с *m* = Partition[Range[9], 3] и выполнить ее заново.

Если у вас очень много ячеек в файле, то выполнить их все можно при помощи меню Evaluation > Evaluate Notebook.

Функции

Все функции *Mathematica* имеют более-менее понятные названия на английском языке, например, Length, Join, Reverse, Split, RotateRight, ParametricPlot, поэтому их можно вполне искать в хелпе, даже не зная точного названия, например, “parametric plot”.

Все функции (в том числе созданные вами) вводятся как Название[аргумент1,аргумент2,...], т.е. используя квадратные скобки. Вообще *Mathematica* использует три вида скобок — []

(функции), {} (списки), () (обычная группировка членов выражения).

Почти у всех функций есть не один, а два, три, четыре и больше синтаксисов, то есть способов задания. Например, функция Log[15] по умолчанию вычислит натуральный логарифм числа 15, но можно явно указать основание, например, Log[2,15]. Так что изучайте дополнительные синтаксисы.

Почти у всех функций есть дополнительные опции, которые вводятся как “Опция -> Значение”. Увидеть все опции функции можно вызвав Options[функция]. Например, решая линейную систему при помощи LinearSolve можно явно указать метод решения — Linear-Solve[...,Method->”Cholesky”].

Палитры

В самом начале знакомства с *Mathematica* для ввода выражений очень удобно пользоваться палитрами. Все палитры собраны в меню Palettes.

Базовой является Palettes > Basic Math Assistant, которая позволяет:

- Вводить арифметические выражения, корни, степени, математические константы
- Вводить тригонометрические функции, а также функции для упрощения выражений, суммы, производные, интегралы
- Позволяет создавать списки и матрицы
- Дает доступ к графическим функциям

Со временем вы научитесь обходиться без палитр, так как ВСЁ можно вводить и при помощи клавиатуры.

Работа с числами

N, Rationalize, Im, Re, Abs, Exp, Log, Sin, Cos, Tan, Mod, IntegerDigits, RealDigits, FromDigits, Round, Floor, Ceiling, RandomInteger, RandomReal

Кроме чисел вы можете использовать встроенные константы, вроде e и π (их можно найти на палитре Palettes > Basic Math Assistant).

In[13]:= $e^{\pi} > \pi^{e}$

Out[13]= True

Эти и многие другие знаки можно ввести при помощи клавиатуры используя Esc. Для этого нажмите Esc, введите код символа, и снова нажмите Esc. Ниже приведены самые известные Escape-последовательности.

esc p esc → π
esc ee esc → e
esc ii esc → i
esc a esc → α
esc b esc → β
esc j esc → φ
esc o esc → ω
esc inf esc → ∞
esc int esc → $\int$
esc S esc → Σ
esc -> esc → →

esc elem esc → ∈

Любое число можно представить в приближенном виде при помощи N (эту функцию можно применять и к спискам значений тоже).

In[14]:=
```
N[e^π]
N[π^e]
```

Out[14]= 23.1407

Out[15]= 22.4592

Обратная операция — превращение вещественного числа в рациональное — осуществляется при помощи Rationalize. Для иррациональных чисел (или даже трансцендентных вроде π) нужно указать точность округления.

In[16]:=
```
Rationalize[9.54]
{Rationalize[π, 0.1], Rationalize[π, 0.01],
 Rationalize[π, 0.001], Rationalize[π, 0.0001]}
```

Out[16]= $\frac{477}{50}$

Out[17]= $\left\{\frac{22}{7}, \frac{22}{7}, \frac{201}{64}, \frac{333}{106}\right\}$

Комплексные числа представляются в виде $a + i\,b$ (комплексную единицу вы найдете на палитре). Знак % указывает на последний вычисленный результат (поэтому его удобно использовать когда вычисляемые выражения находятся в одной ячейке).

In[18]:=
```
√-2
{Re[%], Im[%], Abs[%]}(*Вещественная и мнимая части, а также модуль*)
```

Out[18]= $i\sqrt{2}$

Out[19]= $\left\{0, \sqrt{2}, \sqrt{2}\right\}$

Кстати, символ корня (а также дроби, степени, индекса) можно ввести быстро с клавиатуры. Для этого используйте комбинации клавиш:

$\frac{\square}{\square}$ → Ctrl+/
$\square^{\square}$ → Ctrl+6
$\sqrt{\square}$ → Ctrl+2
$\sqrt[\square]{\square}$ → Ctrl+2, а потом Ctrl+5
$\square_{\square}$ → Ctrl+_

Mathematica знает все основные математические функции (и очень много специальных).

In[20]:=
```
{N[Log[15]], Sin[4/5 π], Cos[7/6 π], Tan[π/2], ArcTan[√3], 5!, Gamma[5]}
```

Out[20]= $\left\{2.70805, \sqrt{\frac{5}{8} - \frac{\sqrt{5}}{8}}, -\frac{\sqrt{3}}{2}, \text{ComplexInfinity}, \frac{\pi}{3}, 120, 24\right\}$

Остаток от деления вычисляется при помощи Mod.

In[21]:=
```
Mod[243, 23]
```

Out[21]= 13

Для того, чтобы получить список цифр целого числа, используется функция IntegerDigits. При ее использовании можно задать основание (по умолчанию оно равно 10), так что с легкостью можно получать цифры в двоичного и любого другого разложения.

```
In[22]:= IntegerDigits[21 436 587]
Out[22]= {2, 1, 4, 3, 6, 5, 8, 7}

In[23]:= IntegerDigits[125, 2]
Out[23]= {1, 1, 1, 1, 1, 0, 1}
```

Обратно собрать число из списка цифр умеет FromDigits (здесь также можно указать основание числа).

```
In[24]:= FromDigits[{4, 3, 2, 1}]
Out[24]= 4321
```

Для округления чисел используются функции Floor (округлить до меньшего целого), Round (округлить до ближайшего целого), Ceiling (округлить до большего целого). Обратите внимание, как они работают для положительных и отрицательных чисел.

```
In[25]:= {Floor[1.45], Round[1.45], Ceiling[1.45]}
Out[25]= {1, 1, 2}

In[26]:= {Floor[-1.45], Round[-1.45], Ceiling[-1.45]}
Out[26]= {-2, -1, -1}
```

Случайные числа можно получить при помощи RandomInteger (целые) и RandomReal (вещественные). Функции позволяют получать отдельные числа, списки и матрицы.

```
In[27]:= RandomInteger[{-10, 10}]
         RandomInteger[{-10, 10}, 10]
         RandomInteger[{-10, 10}, {3, 3}]
Out[27]= -6

Out[28]= {-4, 6, -10, 2, -5, -9, 9, 10, -1, 5}

Out[29]= {{10, 0, 9}, {4, 8, 1}, {4, 7, 0}}
```

Работа со списками

{}, Range, Table, RandomInteger, RandomReal, Part, Span, Position, First, Last, Rest, Take, Insert, Append, Prepend, Drop, Delete, Rest, Length, Count, Total, Join, Union, Intersection, Complement, Sum, Product, Select, MemberQ, FreeQ, Flatten

Создание списка

Создать небольшой список проще всего вручную.

```
In[30]:= {1, 2, 3, 4, 5}
Out[30]= {1, 2, 3, 4, 5}
```

Внутри списка могут быть и **под**списки (и **подпод**списки, и **подподпод**списки, и т.д.).

In[31]:=
```
{1, {2, {3, 4, {5, 6, 7}}}}
```

Out[31]=
```
{1, {2, {3, 4, {5, 6, 7}}}}
```

Если вам нужно создать линейный упорядоченный список, то лучше использовать Range. Эта функция позволяет указать не только начальное и конечное значения, но и шаг.

In[32]:=
```
Range[1, 10]
Range[2, 20, 3]
```

Out[32]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Out[33]=
```
{2, 5, 8, 11, 14, 17, 20}
```

Самая общая функция для задания списков и матриц — Table. Она позволяет задавать сколько угодно измерений + формулу для общего члена. В качестве формулы можно использовать любое выражение (встроенную функцию, свою функцию, комбинацию функций и т.п.). Например, вот так можно получить таблицу умножения чисел от 1 до 5.

Важно: Функция MatrixForm используется для красивого отображения матриц. Никогда не сохраняйте ее в переменную, так как вы потом не сможете работать с такой переменной как с матрицей/списком. В примере ниже я сначала сохраняю матрицу в M, а потом уже показываю ее.

In[34]:=
```
M = Table[i j, {i, 5}, {j, 5}]
MatrixForm[M]
```

Out[34]=
```
{{1, 2, 3, 4, 5}, {2, 4, 6, 8, 10},
 {3, 6, 9, 12, 15}, {4, 8, 12, 16, 20}, {5, 10, 15, 20, 25}}
```

Out[35]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \\ 4 & 8 & 12 & 16 & 20 \\ 5 & 10 & 15 & 20 & 25 \end{pmatrix}$$

Список случайных чисел

Для создания списка случайных чисел — целых или вещественных — вы можете использовать функции RandomInteger и RandomReal. Первым параметром вы задаете диапазон чисел, а вторым — их количество (вернее, количество строк и столбцов).

In[36]:=
```
RandomInteger[{0, 9}, 9]
```

Out[36]=
```
{8, 1, 8, 2, 7, 0, 2, 5, 1}
```

In[37]:=
```
RandomReal[{0, 1}, {5, 5}]
```

Out[37]=
```
{{0.127123, 0.39729, 0.908859, 0.247528, 0.37284},
 {0.344075, 0.773094, 0.432703, 0.799705, 0.340008},
 {0.210053, 0.505726, 0.00654026, 0.753134, 0.142157},
 {0.783767, 0.441896, 0.809763, 0.99803, 0.363682},
 {0.370729, 0.559712, 0.158531, 0.0503619, 0.318228}}
```

Специальные матрицы — единичная и диагональная

In[38]:=
```
IdentityMatrix[5];
MatrixForm[%]
```

Out[39]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

In[40]:=
```
DiagonalMatrix[{1, 2, 3, 4, 5}];
MatrixForm[%]
```

Out[41]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

Доступ к элементам

Доступ к элементам осуществляется при помощи функции Part (обычно она вводится при помощи Escape-последовательности как esc[[esc и esc]]esc). Индексов должно быть столько, сколько измерений в списке — в данном случае два: строка и столбец.

In[42]:=
```
M[[2, 2]]
```

Out[42]=
```
4
```

Можно за раз брать сразу несколько элементов, для чего используется функция Span (;;). Вы просто указываете, с какого по какой элемент взять в нужно измерении (строке и столбце).

In[43]:=
```
M[[2 ;; 4, 2]]
```

Out[43]=
```
{4, 6, 8}
```

При помощи Span можно брать как отдельные элементы, так и целые подматрицы. Индекс -1 указывает на последний элемент.

In[44]:=
```
M[[2 ;; 4, 3 ;; -1]]
MatrixForm[%]
```

Out[44]=
```
{{6, 8, 10}, {9, 12, 15}, {12, 16, 20}}
```

Out[45]//MatrixForm=

$$\begin{pmatrix} 6 & 8 & 10 \\ 9 & 12 & 15 \\ 12 & 16 & 20 \end{pmatrix}$$

Если вам нужно взять элементы не по порядку (например, сначала 3-ий столбец, потом 1-ый, потом 2-ой), то нужно поместить эти индексы в список.

In[46]:=
```
M[[{3, 1, 2}, ;;]]
```

Out[46]=
```
{{3, 6, 9, 12, 15}, {1, 2, 3, 4, 5}, {2, 4, 6, 8, 10}}
```

Поиск элементов

Для нахождения позиции элемента в списке используется Position. Эта функция выдает

список позиций, так как элемент может встречаться не один раз. В примере, все числа в матрице M различны, поэтому для 16 нашлась только одна позиция.

In[47]:=
```
Position[M, 16]
```
Out[47]=
```
{{4, 4}}
```

Кстати, вы легко можете проверить, есть ли какой-нибудь элемент в списке или матрице, при помощи функции MemberQ

In[48]:=
```
MemberQ[M, 16]
```
Out[48]=
```
False
```

Ответ отрицательный, так как по умолчанию поиск ведется на первом уровне выражения. То есть эту функцию легко применять к спискам, а вот для матриц нужно указать уровень вложенности 2.

In[49]:=
```
MemberQ[M, 16, 2]
```
Out[49]=
```
True
```

Теперь *Mathematica* нашла элемент. Но узнать на какой он позиции вам по-прежнему поможет Position.

Операции над списками

Вообще, работа со списками — одна из центральных тем *Mathematica*. Список является основным типом данных, и зачастую код программы выгоднее строить так, чтобы внутри него использовались списки. Для демонстрации базовых функций создадим список из случайных цифр (т.е. чисел от 1 до 9).

In[50]:=
```
R = RandomInteger[{1, 9}, 10]
```
Out[50]=
```
{1, 8, 7, 2, 9, 9, 8, 2, 7, 7}
```

При помощи следующих функций можно добавлять элементы (в примере это будет 0) в список: Insert (внутрь), Append (в конец), Prepend (в начало).

Замечание: Сам список при этом не меняется. Если вы хотите применить изменения, то используйте код вроде *R* = Append[*R*, 0].

In[51]:=
```
Insert[R, 0, 5](*вставляем 0 на позицию 5*)
Append[R, 0](*добавляем 0 в конец*)
Prepend[R, 0](*добавляем 0 в начало*)
```
Out[51]=
```
{1, 8, 7, 2, 0, 9, 9, 8, 2, 7, 7}
```
Out[52]=
```
{1, 8, 7, 2, 9, 9, 8, 2, 7, 7, 0}
```
Out[53]=
```
{0, 1, 8, 7, 2, 9, 9, 8, 2, 7, 7}
```

Для удаления элементов можно использовать: Drop (убрать несколько последовательных элементов), Delete (убрать элемент на позиции), Rest (убрать первый элемент).

In[54]:=
```
Drop[R, 5](*убираем первых пять элементов*)
Delete[R, 5](*убираем 5-ый элемент*)
First[R](*берем первый элемент*)
Last[R](*берем последний элемент*)
Rest[R](*убираем только 1-ый элемент*)
```

Out[54]= {9, 8, 2, 7, 7}

Out[55]= {1, 8, 7, 2, 9, 8, 2, 7, 7}

Out[56]= 1

Out[57]= 7

Out[58]= {8, 7, 2, 9, 9, 8, 2, 7, 7}

Удаление скобок {}

Очень полезной функцией является Flatten, которая убирает внутренние скобки {}, то есть превращает матрицу в список.

In[59]:=
```
RandomInteger[{0, 9}, {3, 3}]
Flatten[%]
```

Out[59]= {{3, 0, 2}, {3, 4, 4}, {8, 4, 0}}

Out[60]= {3, 0, 2, 3, 4, 4, 8, 4, 0}

Количество и сумма элементов

Очень важная функция — длина списка Length.

In[61]:=
```
Length[R]
```

Out[61]= 10

А как узнать, сколько не всех элементов, а только чисел 5 в нашем списке? Для этого можно использовать функцию Count.

In[62]:=
```
Count[R, 5]
```

Out[62]= 0

Используя Count и Table очень легко получить количество каждого числа от 1 до 9 в списке. Сгенерируем список пар (число, количество таких чисел в списке).

In[63]:=
```
Table[{i, Count[R, i]}, {i, 9}]
```

Out[63]= {{1, 1}, {2, 2}, {3, 0}, {4, 0}, {5, 0}, {6, 0}, {7, 3}, {8, 2}, {9, 2}}

Помните старую легенду про шахматы и зерна пшеницы? Создатель шахмат попросил в награду положить 1 зерно на первую клетку, 2 — на вторую, 4 — на третью и т.д. Каково же общее количество зерен на 64 клетках? Для этого нам понадобится просуммировать элементы списка. Делается это при помощи Total.

Получившее число огромно — оно превышает весь объем пшеницы, собранной за всю историю человечества.

In[64]:=
```
Table[2^(i-1), {i, 64}]
Total[%]
```

Out[64]=
```
{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16 384, 32 768,
 65 536, 131 072, 262 144, 524 288, 1 048 576, 2 097 152, 4 194 304, 8 388 608,
 16 777 216, 33 554 432, 67 108 864, 134 217 728, 268 435 456, 536 870 912,
 1 073 741 824, 2 147 483 648, 4 294 967 296, 8 589 934 592, 17 179 869 184,
 34 359 738 368, 68 719 476 736, 137 438 953 472, 274 877 906 944, 549 755 813 888,
 1 099 511 627 776, 2 199 023 255 552, 4 398 046 511 104, 8 796 093 022 208,
 17 592 186 044 416, 35 184 372 088 832, 70 368 744 177 664, 140 737 488 355 328,
 281 474 976 710 656, 562 949 953 421 312, 1 125 899 906 842 624, 2 251 799 813 685 248,
 4 503 599 627 370 496, 9 007 199 254 740 992, 18 014 398 509 481 984,
 36 028 797 018 963 968, 72 057 594 037 927 936, 144 115 188 075 855 872,
 288 230 376 151 711 744, 576 460 752 303 423 488, 1 152 921 504 606 846 976,
 2 305 843 009 213 693 952, 4 611 686 018 427 387 904, 9 223 372 036 854 775 808}
```

Out[65]=
```
18 446 744 073 709 551 615
```

Кстати, этот же список можно было сгенерировать и при помощи Range.

In[66]:=
```
2^Range[0,63]
Total[%]
```

Out[66]=
```
{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16 384, 32 768,
 65 536, 131 072, 262 144, 524 288, 1 048 576, 2 097 152, 4 194 304, 8 388 608,
 16 777 216, 33 554 432, 67 108 864, 134 217 728, 268 435 456, 536 870 912,
 1 073 741 824, 2 147 483 648, 4 294 967 296, 8 589 934 592, 17 179 869 184,
 34 359 738 368, 68 719 476 736, 137 438 953 472, 274 877 906 944, 549 755 813 888,
 1 099 511 627 776, 2 199 023 255 552, 4 398 046 511 104, 8 796 093 022 208,
 17 592 186 044 416, 35 184 372 088 832, 70 368 744 177 664, 140 737 488 355 328,
 281 474 976 710 656, 562 949 953 421 312, 1 125 899 906 842 624, 2 251 799 813 685 248,
 4 503 599 627 370 496, 9 007 199 254 740 992, 18 014 398 509 481 984,
 36 028 797 018 963 968, 72 057 594 037 927 936, 144 115 188 075 855 872,
 288 230 376 151 711 744, 576 460 752 303 423 488, 1 152 921 504 606 846 976,
 2 305 843 009 213 693 952, 4 611 686 018 427 387 904, 9 223 372 036 854 775 808}
```

Out[67]=
```
18 446 744 073 709 551 615
```

А еще ту же задачу можно решить при помощи функции Sum, которая как раз создана для суммирования элементов

In[68]:=
```
Sum[2^(i-1), {i, 64}]
```

Out[68]=
```
18 446 744 073 709 551 615
```

Объединение, пересечение и дополнение списков

Конечно, есть в *Mathematica* и функции, которые позволяют работать сразу с несколькими списками: Join (объединить списки как есть), Union (объединение с удалением повторов), Intersection (пересечение), Complement (дополнение, разность).

In[69]:=
```
A = RandomInteger[{1, 9}, 5]
B = RandomInteger[{1, 9}, 5]
```

Out[69]=
```
{7, 3, 8, 3, 2}
```

Out[70]=
```
{2, 8, 9, 9, 8}
```

In[71]:=
```
Join[A, B](*простое сцепливание списков*)
Union[A, B]
(*при объединении элементы упорядочиваются + удаляются дубликаты*)
Intersection[A, B](*пересечение списков*)
Complement[A, B](*вычитание списка B из списка A*)
```

Out[71]= {7, 3, 8, 3, 2, 2, 8, 9, 9, 8}

Out[72]= {2, 3, 7, 8, 9}

Out[73]= {2, 8}

Out[74]= {3, 7}

Работа с матрицами

Det, Transpose, Inverse, MatrixRank, Eigenvalues, Eigenvectors, Dot (.), LinearSolve

Матрицы — это просто списки с вложенными подсписками, поэтому работать с ними можно так же, как с обычными списками. Но есть несколько математических операций над матрицами, которые нужно знать.

Сначала зададим матрицу 6x6 из случайных целых чисел.

In[75]:=
```
M = RandomInteger[{-5, 5}, {6, 6}];
MatrixForm[M]
```

Out[76]//MatrixForm=

$$\begin{pmatrix} -2 & 2 & -5 & 2 & -4 & 1 \\ 0 & -2 & -5 & 2 & -2 & 0 \\ -2 & -5 & -5 & 5 & 4 & -5 \\ 1 & -5 & -1 & 5 & 2 & 4 \\ -3 & -3 & 5 & 3 & -4 & 0 \\ 1 & 4 & -1 & 4 & 5 & 4 \end{pmatrix}$$

Определитель матрицы:

In[77]:=
```
Det[M]
```

Out[77]= 23 616

Операция транспонирования — элемент *i*, *j* переходит в *j*, *i* (т.е. строки меняются местами с столбцами).

In[78]:=
```
MatrixForm[Transpose[M]]
```

Out[78]//MatrixForm=

$$\begin{pmatrix} -2 & 0 & -2 & 1 & -3 & 1 \\ 2 & -2 & -5 & -5 & -3 & 4 \\ -5 & -5 & -5 & -1 & 5 & -1 \\ 2 & 2 & 5 & 5 & 3 & 4 \\ -4 & -2 & 4 & 2 & -4 & 5 \\ 1 & 0 & -5 & 4 & 0 & 4 \end{pmatrix}$$

Обратная матрица:

In[79]:=
```
MatrixForm[Inverse[M]]
```

Out[79]//MatrixForm=

$$\begin{pmatrix} -\frac{325}{492} & \frac{1021}{984} & -\frac{61}{369} & -\frac{2195}{5904} & \frac{1201}{5904} & \frac{325}{984} \\ -\frac{55}{656} & \frac{251}{1312} & -\frac{4}{123} & -\frac{1469}{7872} & \frac{559}{7872} & \frac{219}{1312} \\ -\frac{6}{41} & \frac{4}{41} & -\frac{4}{123} & -\frac{19}{246} & \frac{29}{246} & \frac{3}{41} \\ -\frac{195}{656} & \frac{711}{1312} & -\frac{1}{41} & -\frac{603}{2624} & \frac{601}{2624} & \frac{359}{1312} \\ \frac{25}{164} & -\frac{129}{328} & \frac{11}{123} & \frac{295}{1968} & -\frac{269}{1968} & -\frac{25}{328} \\ \frac{157}{492} & -\frac{469}{984} & -\frac{8}{369} & \frac{1787}{5904} & -\frac{889}{5904} & -\frac{157}{984} \end{pmatrix}$$

Ранг матрицы, т.е. количество линейно независимых строк (столбцов).

In[80]:=
```
MatrixRank[M]
```

Out[80]= 6

Списки собственных значений и собственных векторов:

In[81]:=
```
N[Eigenvalues[M]]
N[Eigenvectors[M]]
```

Out[81]= {-9.7057, 8.76816, 7.78223, -4.83241 + 2.62113 i, -4.83241 - 2.62113 i, -1.17988}

Out[82]=
```
{{0.345434, 2.26978, 5.96111, 1.51325, -4.64443, 1.},
 {0.445687, 0.285556, -0.253352, 0.827475, -0.0766017, 1.},
 {0.894279, 0.572119, -0.673477, 0.611004, -0.503603, 1.},
 {-0.955817 + 0.623813 i, -1.23237 + 0.349264 i, -0.9188 + 0.628708 i,
  -0.990354 - 0.162313 i, 0.0190985 + 0.375645 i, 1.},
 {-0.955817 - 0.623813 i, -1.23237 - 0.349264 i, -0.9188 - 0.628708 i,
  -0.990354 + 0.162313 i, 0.0190985 - 0.375645 i, 1.},
 {-1.79465, -0.539168, -0.513898, -1.03018, 0.475655, 1.}}
```

Умножение матриц выполняется при помощи оператора Dot (.), т.е. между матрицами просто ставится точка. Например, решим матричное уравнение $A \cdot x = b$. Для этого домножим слева и справа на обратную матрицу A^{-1}: получим $A^{-1} \cdot A \cdot x = A^{-1} \cdot b \Rightarrow$ так как $A^{-1} \cdot A = E \Rightarrow x = A^{-1} \cdot b$.

In[83]:=
```
A = RandomInteger[{-5, 5}, {5, 5}]
B = RandomInteger[{-5, 5}, 5]
Inverse[A].B
```

Out[83]=
```
{{5, 1, 0, 4, 5}, {4, 0, 2, -4, 4},
 {-4, 2, 4, 2, -3}, {4, 5, -4, 4, 2}, {-3, -3, -3, 5, 2}}
```

Out[84]= {3, -3, -5, 2, 2}

Out[85]= $\left\{\frac{3727}{2503}, -\frac{3162}{2503}, -\frac{2247}{5006}, \frac{3833}{5006}, -\frac{3126}{2503}\right\}$

Кстати, то же самое (решить линейную систему) можно сделать при помощи LinearSolve.

In[86]:=
```
LinearSolve[A, B]
```

Out[86]= $\left\{\frac{3727}{2503}, -\frac{3162}{2503}, -\frac{2247}{5006}, \frac{3833}{5006}, -\frac{3126}{2503}\right\}$

Структура выражений

FullForm, TreeForm, Part, AtomQ, Head

Любое выражение в *Mathematica* (**абсолютно любое!**) на самом деле является конструкцией вида *f*[expr1, expr2, ...]. Здесь:
1) f — название функции или голова выражения
2) expr1, expr2, ... — выражения, которые также имеют вид *f*[expr1, expr2, ...] и т.д.

Вы можете писать *a* + *b*, но *Mathematica* все равно понимает этот код как Plus[*a*, *b*]. Увидеть настоящую форму выражения можно при помощи FullForm.
Примеры:

In[87]:=
```
FullForm[Sin[x + 1]]
FullForm[{a, b, c}]
FullForm[a + b c (e + f)]
```

Out[87]//FullForm=
```
Sin[Plus[1, x]]
```

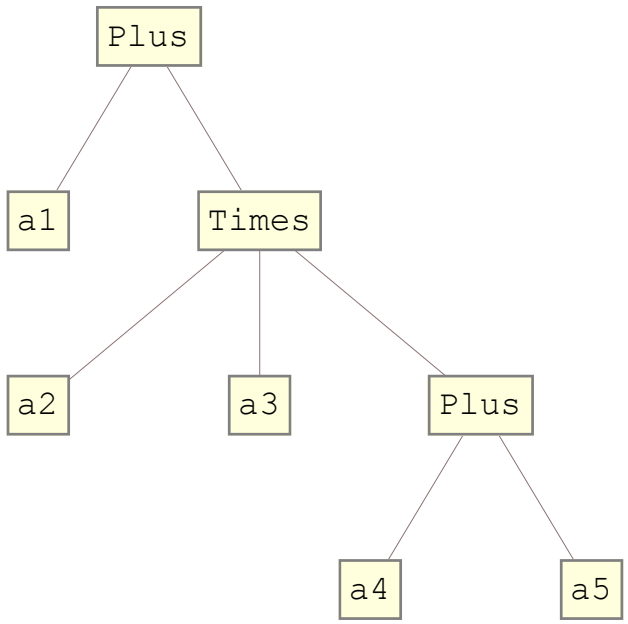
Out[88]//FullForm=
```
List[a, b, c]
```

Out[89]//FullForm=
```
Plus[a, Times[b, c, Plus[e, f]]]
```

Визуально это можно представить в виде дерева, что и делает функция TreeForm.

In[90]:=
```
TreeForm[a1 + a2 a3 (a4 + a5)]
```

Out[90]//TreeForm=



Понимая как устроено выражение, вы можете обращаться к любому его элементу (и изменять его). Например, выражение выше состоит из 4 уровней — 0, 1, 2, 3. На нулевом уровне всегда находится голова выражения. Получить каждый элемент этого выражения можно при помощи уже упоминавшейся ранее функции Part ([[]]).

Замечание: Обратите внимание, что выражения expr[[2]], expr[[2, 3]] являются составными выражениями (т.е. у них тоже есть голова, которая также находится на 0 уровне).

In[91]:=
```
expr = a1 + a2 a3 (a4 + a5);
```

In[92]:=
```
expr[[0]]
expr[[1]]
expr[[2, 0]]
expr[[2, 1]]
expr[[2, 2]]
expr[[2, 3, 0]]
expr[[2, 3, 1]]
expr[[2, 3, 2]]
```

Out[92]= Plus

Out[93]= a1

Out[94]= Times

Out[95]= a2

Out[96]= a3

Out[97]= Plus

Out[98]= a4

Out[99]= a5

Голову выражения можно получить и при помощи функции Head.

In[100]:=
```
{Head["строка"], "строка"[[0]]}
{Head[{a, b, c}], {a, b, c}[[0]]}
{Head[1.65], 1.65[[0]]}
{Head[a + b], (a + b)[[0]]}
```

Out[100]= {String, String}

Out[101]= {List, List}

Out[102]= {Real, Real}

Out[103]= {Plus, Plus}

Вообще, может получиться бесконечная вложенность — выражения состоят из выражений, которые состоят из выражений, которые состоят из выражений, ...
Такого не случается, так как есть атомарные выражения, которые уже не делятся. К атомарным выражениям относятся: числа, символы, строки.

Примеры атомарных выражений (проверить на атомарность можно при помощи AtomQ):

In[104]:=
```
AtomQ[1.65](*число*)
AtomQ[2 + 3 i](*комплексное число*)
AtomQ["строка"](*число*)
```

Out[104]= True

Out[105]= True

Out[106]= True

А практически любые другие выражения уже не будут атомами.

In[107]:=
```
AtomQ[a + b]
AtomQ[Sin[x^2]]
AtomQ[{1, 2, 3}]
```

Out[107]= False

Out[108]= False

Out[109]= False

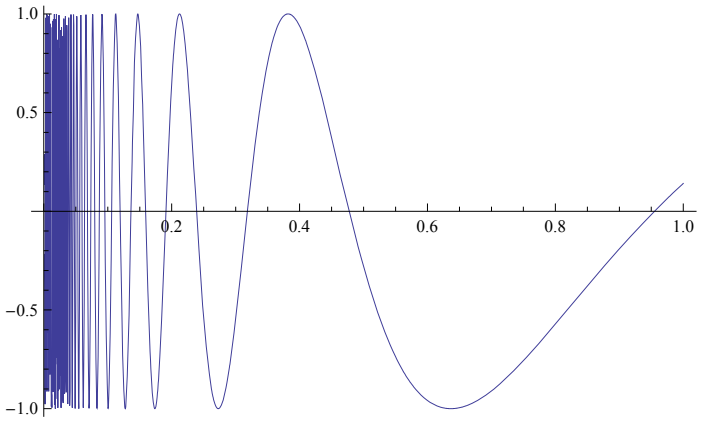
Графика

Plot, PolarPlot, ParametricPlot, ListPlot, опции графики PlotStyle, PlotRange

Для построения графиков функций используется Plot. У нее два параметра — функция (или список функций), диапазон значений переменной.

In[110]:=
```
Plot[Sin[3 / x], {x, 0.001, 1}]
```

Out[110]=



Список всех опций этой функции.

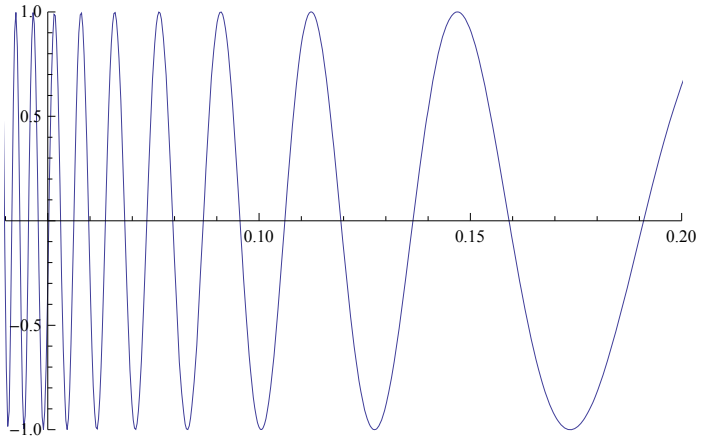
In[111]:=
```
Options[Plot]
```

Out[111]=
```
{AlignmentPoint → Center, AspectRatio → 1/GoldenRatio, Axes → True,
 AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
 BaselinePosition → Automatic, BaseStyle → {}, ClippingStyle → None,
 ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
 ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
 DisplayFunction :→ $DisplayFunction, Epilog → {}, Evaluated → Automatic,
 EvaluationMonitor → None, Exclusions → Automatic, ExclusionsStyle → None,
 Filling → None, FillingStyle → Automatic, FormatType :→ TraditionalForm,
 Frame → False, FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic,
 FrameTicksStyle → {}, GridLines → None, GridLinesStyle → {},
 ImageMargins → 0., ImagePadding → All, ImageSize → Automatic,
 ImageSizeRaw → Automatic, LabelStyle → {}, MaxRecursion → Automatic,
 Mesh → None, MeshFunctions → {#1 &}, MeshShading → None, MeshStyle → Automatic,
 Method → Automatic, PerformanceGoal :→ $PerformanceGoal, PlotLabel → None,
 PlotLegends → None, PlotPoints → Automatic, PlotRange → {Full, Automatic},
 PlotRangeClipping → True, PlotRangePadding → Automatic, PlotRegion → Automatic,
 PlotStyle → Automatic, PreserveImageOptions → Automatic, Prolog → {},
 RegionFunction → (True &), RotateLabel → True, TargetUnits → Automatic,
 Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}
```

Одна из самых полезных опций — PlotRange — позволяет явно задать область графика (иначе *Mathematica* сама решит, как ей его выводить). В примере указана область $0.04 \le x \le 0.2$, $-1 \le y \le 1$.

In[112]:=
```
Plot[Sin[3 / x], {x, 0.001, 1}, PlotRange → {{0.04, 0.2}, {-1, 1}}]
```

Out[112]=



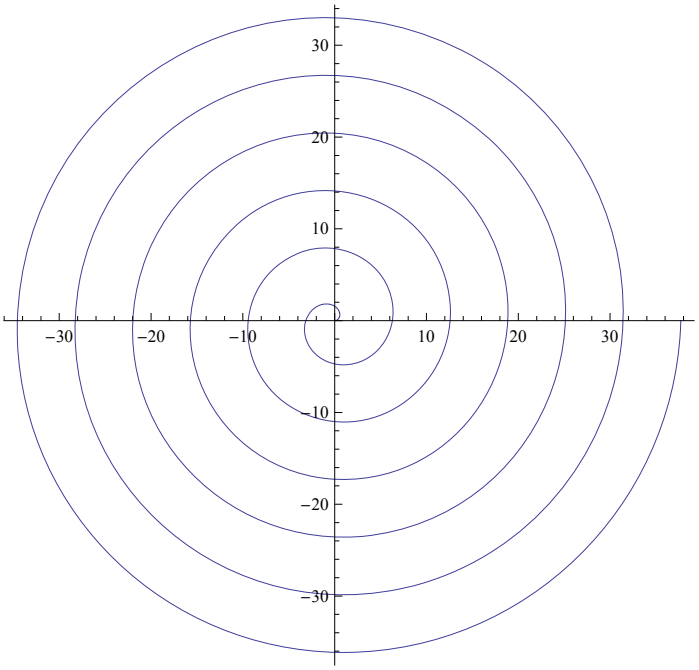
Базовые опции графики:

AspectRatio — отношение сторон графика (по умолчанию используется золотое сечение)
Axes — показывать оси координат или нет (по умолчанию показывать)
AxesOrigin — центр осей (по умолчанию *Mathematica* сама принимает решение, не всегда это будет (0,0))
ImageSize — размер рисунка в пикселях (по умолчанию *Mathematica* сама принимает решение)
PlotRange — видимая область (по умолчанию *Mathematica* показывает все иксы, а уже игреки подбираются автоматически, поэтому бывают обрезки)
PlotStyle — стили (по умолчанию *Mathematica* сама принимает решение, но вы можете настроить стиль каждой линии в отдельности)

Для вывода полярных графиков используется PolarPlot.

In[113]:=
```
PolarPlot[t, {t, 0, 12 π}]
```

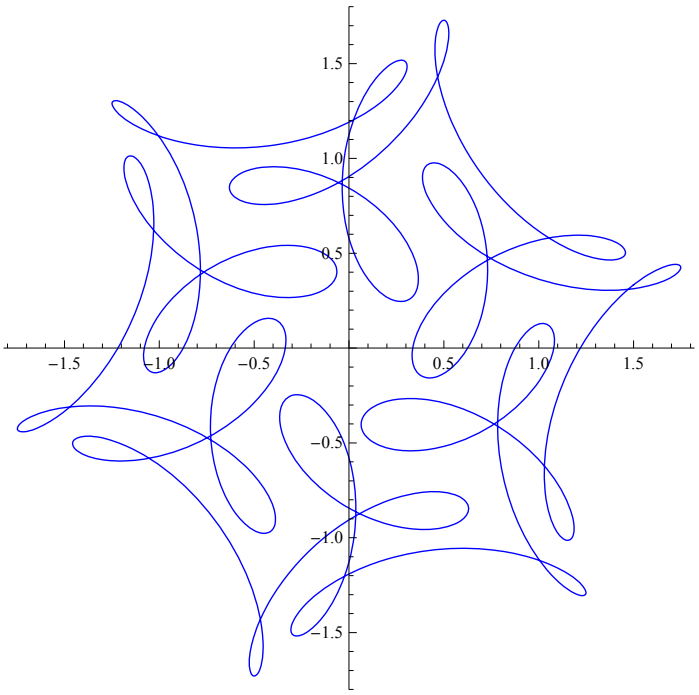
Out[113]=



Если функция задана параметрически, то для ее вывода используется ParametricPlot. Опция PlotStyle (которая есть и у Plot, и у PolarPlot) позволяет изменить цвет и толщину линий (и много другое).

In[114]:=
```
ParametricPlot[
 {Cos[t], Sin[t]} + 1/2 {Cos[7 t], Sin[7 t]} + 1/3 {Cos[-17 t + π/2], Sin[-17 t + π/2]},
 {t, 0, 2 π}, PlotStyle → {Thickness[0.002], Blue}]
```

Out[114]=



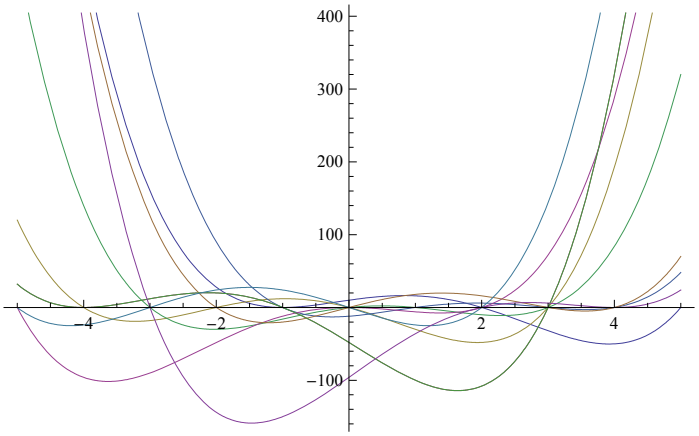
Никто не мешает сначала подготовить список функций, а потом уже нарисовать их. В примере генерируются 10 полиномов вида $(x - a)(x - b)(x - c)(x - d)$, где a, b, c, d —

случайные числа от -5 до 5. То есть эти полиномы имеют по четыре корня на промежутке $-5 \le x \le 5$, что и видно на картинке.

In[115]:=
```
Table[Product[x - RandomInteger[{-5, 5}], {4}], {10}]
Plot[%, {x, -5, 5}]
```

Out[115]=
```
{(-5 + x) (-2 + x) (1 + x)^2, (-3 + x) (1 + x) (4 + x)^2, (-3 + x) x (2 + x) (4 + x),
 (-3 + x) (-1 + x) x (3 + x), (-4 + x) (-3 + x) (-1 + x) (1 + x),
 (-2 + x) x^2 (5 + x), (-4 + x) (-3 + x) x (2 + x), (-3 + x) (1 + x) (4 + x)^2,
 (-2 + x) x (3 + x) (5 + x), (-4 + x)^2 (-2 + x) (3 + x)}
```

Out[116]=

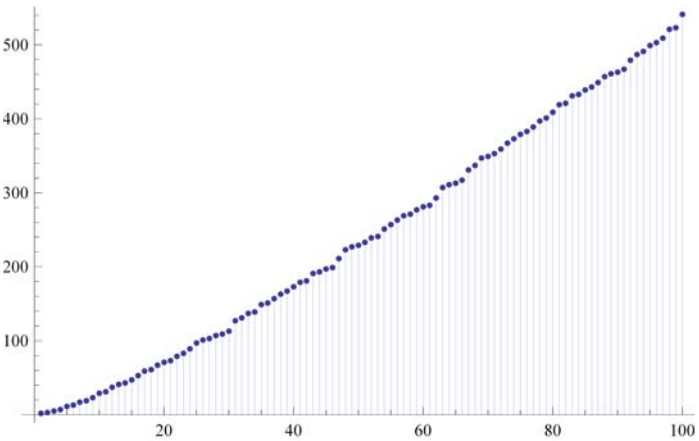


Для вывода списка значений проще всего использовать ListPlot. Обратите внимание на две опции — Joined –> True (позволяет соединить точки линией) и Filling → Axis (заполнение ниже графика). Например, построим график первых 100 простых чисел.

In[117]:=
```
Table[Prime[i], {i, 100}]
ListPlot[%, Filling → Axis]
```

Out[117]=
```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163,
 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251,
 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439,
 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541}
```

Out[118]=



Графика 3D

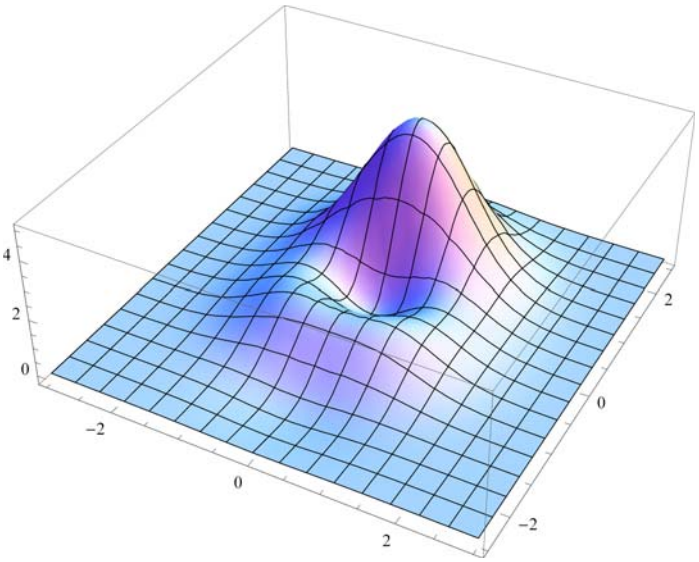
Plot3D, ParametricPlot3D

Построение трехмерных графиков схоже с построением их двухмерных собратьев — только к названиям функций добавляется 3D + надо задавать интервалы не для одной переменной, а для двух.
Кроме того, получившуюся картинку можно двигать и поворачивать при помощи мыши.

In[119]:=
```
graph[x_, y_] := (x^2 + 3 y^2 + 2 y^3) e^(1-x^2-y^2);
```

In[120]:=
```
Plot3D[graph[x, y], {x, -3, 3}, {y, -2.5, 2.5}, PlotRange → All]
```

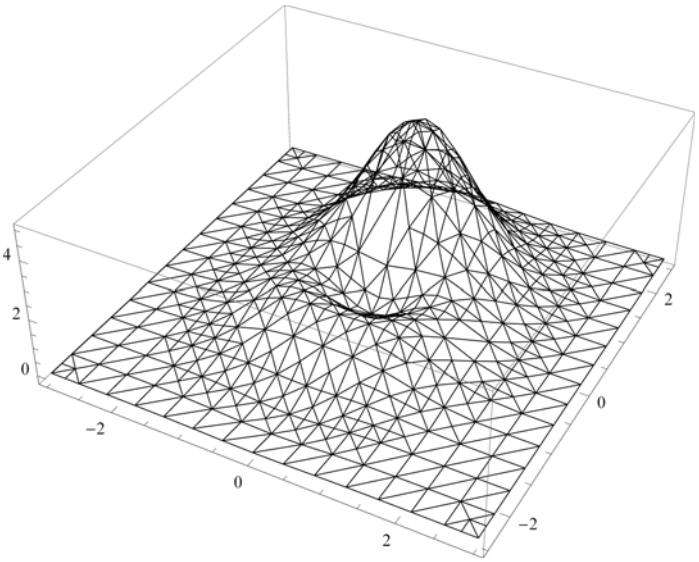
Out[120]=



Однако опций в трехмерной графике гораздо больше. Например, уберем все кроме сетки.

In[121]:=
```
Plot3D[graph[x, y], {x, -3, 3}, {y, -2.5, 2.5},
 PlotRange → All, Mesh → All, PlotStyle → None, MaxRecursion → 1]
```

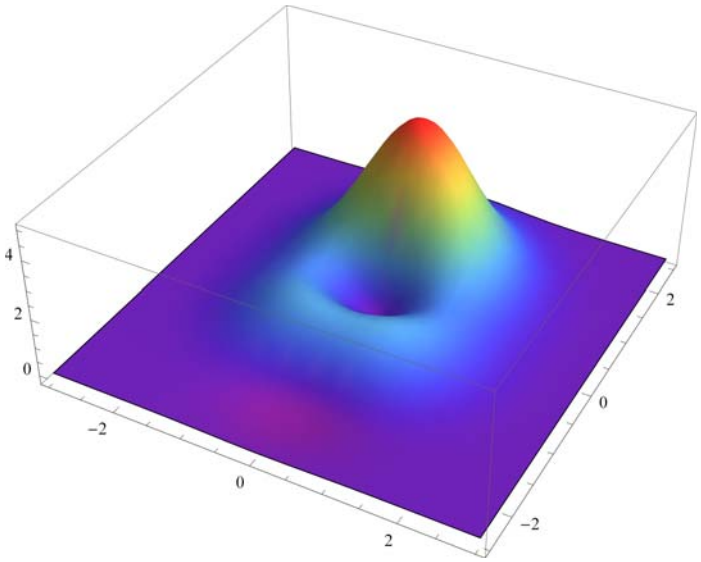
Out[121]=



А теперь уберем сетку, но раскрасим все используя “радужную” схему (схем раскраски очень много, см. ColorFunction в хелпе).

In[122]:=
```
Plot3D[graph[x, y], {x, -3, 3}, {y, -2.5, 2.5},
 PlotRange → All, ColorFunction → "Rainbow", Mesh → None]
```

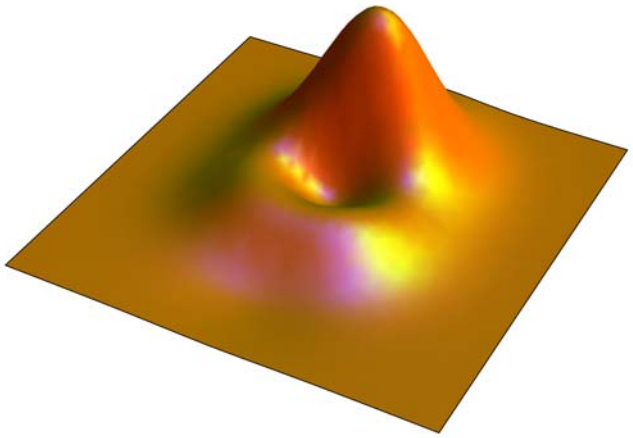
Out[122]=



Можно добавить эффект освещения (заодно убрав обрамляющий бокс и оси координат).

In[123]:=
```
Plot3D[graph[x, y], {x, -3, 3}, {y, -2.5, 2.5}, PlotRange → All, Boxed → False,
 Axes → False, Mesh → None, PlotStyle → {Orange, Specularity[White, 30]}]
```

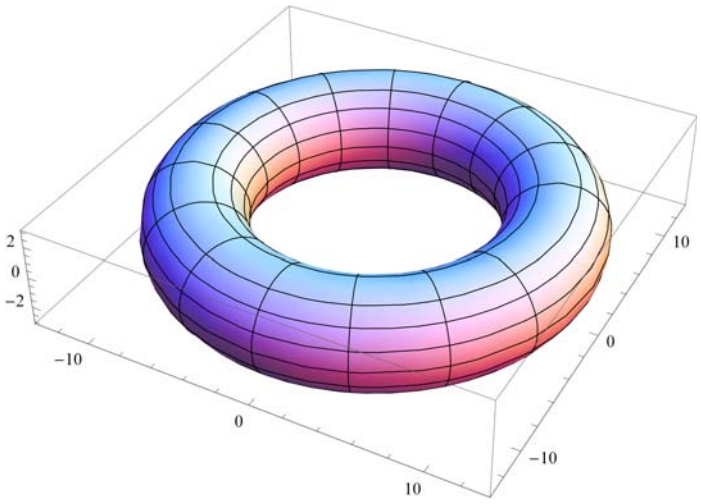
Out[123]=



ParametricPlot3D позволяет рисовать очень эффектные поверхности, например, тор (бублик). Подробнее про параметризацию этой поверхности вы можете прочитать в Википедии.

In[124]:=
```
R1 = 10;
R2 = 3;
ParametricPlot3D[
 {Cos[t] (R1 + R2 Sin[u]), Sin[t] (R1 + R2 Sin[u]), R2 Cos[u]}, {t, 0, 2 π}, {u, 0, 2 π}]
```

Out[126]=



“Примитивная” графика

Graphics, Line, Point, Circle, цвета Red, Blue, Green, Hue, опции PointSize, Thickness

Кроме графиков функций в *Mathematica* есть и графические примитивы (то есть базовые объекты) — точка, линия, полигон, окружность, куб, цилиндр и прочие.
Отобразить их можно при помощи функции Graphics (Graphics3D для трехмерных объектов).

In[127]:=
```
Graphics[Point[{3, 3}]]
```

Out[127]=

Несколько объектов нужно объединять в список.

In[128]:=
```
Graphics[{Point[{0, 0}], Point[{2, 0}], Line[{{0, 0}, {2, 0}}]}]
```

Out[128]=

Кроме того, в таком списке можно указать дополнительные опции для изменения стиля объектов — цвет (Hue или именнованные цвета), толщина линий (Thickness), толщина точки (PointSize). Опция будет применена ко всем объектам, которые заданы после нее.

In[129]:=
```
Graphics[{
  PointSize[0.02], Point[{0, 0}], Point[{2, 0}],
  Red, Line[{{0, 0}, {2, 0}}]
 }]
```

Out[129]=

Объединить разнородные типы графиков (функции и примитивы) можно при помощи Show.

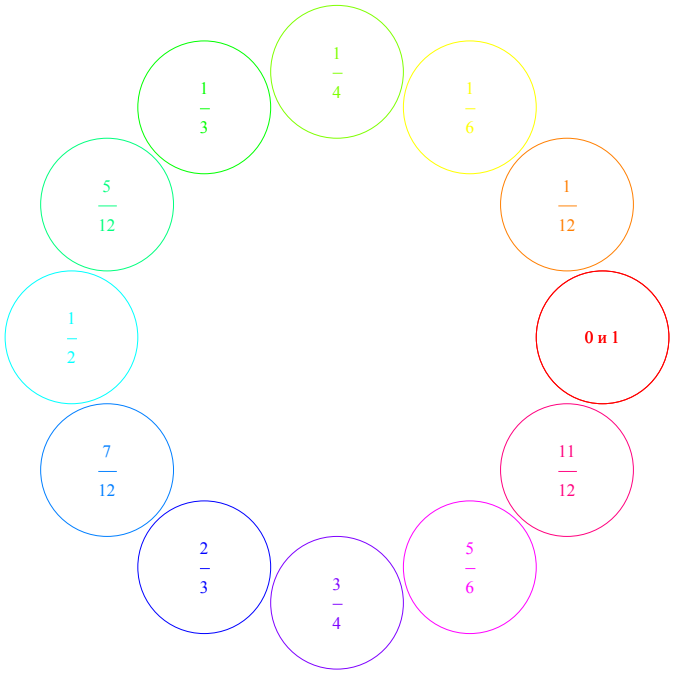
In[130]:=
```
Show[
 Graphics[{Orange, PointSize[Large], Point[{-1, 0}], Point[{1, 0}]},
  PlotRange → {{-2, 2}, {-2, 1}}],
 Plot[(x/1.5)^2 - 1.5, {x, -1, 1}, PlotStyle → {Orange, Thickness[0.02]}]
]
```

Out[130]=

Никто не мешает использовать внутри графики списки объектов, например, кругов. Цвет кругов задается при помощи Hue — для наглядности я поместил значения Hue в центры кругов. Как видно, Hue устроен следующим образом — 0 (красный), 1/3 (зеленый), 2/3 (синий), 1 (снова красный).

In[131]:=
```
Graphics[
  Table[{
    Hue[i],
    Circle[4 {Cos[2 π i], Sin[2 π i]}],
    Text[If[i == 0 ⋁ i == 1, "0 и 1", i], 4 {Cos[2 π i], Sin[2 π i]}]
  }, {i, 0, 1, 1/12}]]
```

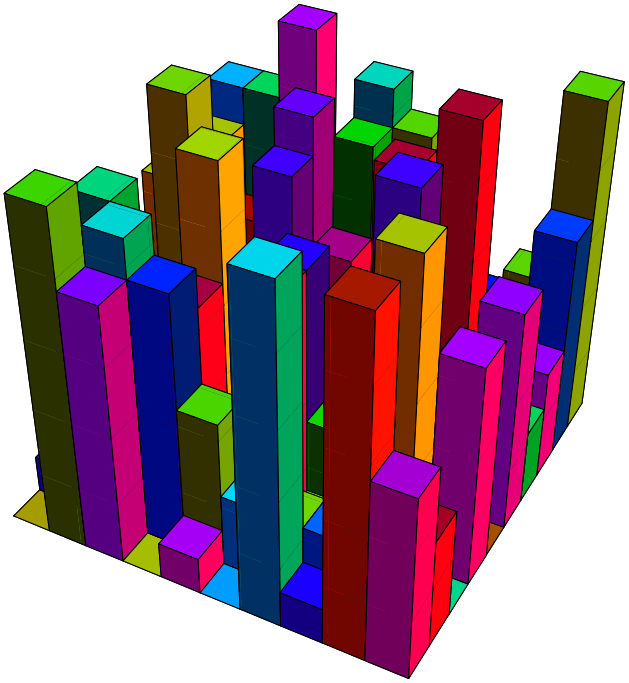
Out[131]=



Использование 3D объектов ничем сильно не отличается. Вот так можно построить кубоиды разной высоты и случайного цвета на площадке 10x10.

In[132]:=
```
Graphics3D[
 Table[{Hue[RandomReal[]], Cuboid[{i, j, 0}, {i + 1, j + 1, RandomInteger[9]}]},
  {i, 0, 9}, {j, 0, 9}], Boxed → False]
```

Out[132]=



Функции и программирование

Образцы (_), If, Which, Do, While, Break, Block, анонимные функции (#&)

Собственные функции

Создать свою функцию в *Mathematica* очень просто, например, вот функция $\mathrm{sq}(x) = x^2$:

In[133]:=
```
sq[x_] := x^2;
```

Проверим как она работает:

In[134]:=
```
{sq[x], sq[1], sq[-3.5], sq[π]}
```

Out[134]=
$$\{x^2,\ 1,\ 12.25,\ \pi^2\}$$

Важно:

1) аргументы объявляются при помощи знака подчеркивания (x_)

2) знак подчеркивания не используется в теле функции (x^2)

3) название функции не должно совпадать с уже существующей встроенной функцией (просто проверьте в хелпе) + не надо начинать ее название с цифр

4) между определением и телом ставится знак :=

5) в конце лучше ставить точку с запятой (;)

6) нужно вычислить функцию перед использованием (и после каждого обновления)

Знак подчеркивания (он же образец, он же шаблон, он же pattern) нужен для того, чтобы показать *Mathematica* переменную какого типа вы собираетесь использовать. Простой знак

подчеркивания обозначает любое выражение — действительно функцию у можно использовать и с числом, и со списком, и со строкой (и с чем угодно).

In[135]:=
```
{sq[1], sq[{1, 2, 3}], sq["строка"]}
```

Out[135]=
```
{1, {1, 4, 9}, строка^2}
```

Тип переменной можно уточнить, используя более сложные образцы/шаблоны. Вот несколько примеров:

f[x_List] — x должен быть быть списком (вместо List может стоять любая “голова”)
f[x_Integer] — x должен быть быть целым числом
f[x_Integer?Positive] — x должен быть быть целым + положительным числом (после ? может стоять любая функция одного параметра, возвращающая True/False)
f[x_?OddQ] — x должен быть быть нечетным числом

Ветвления

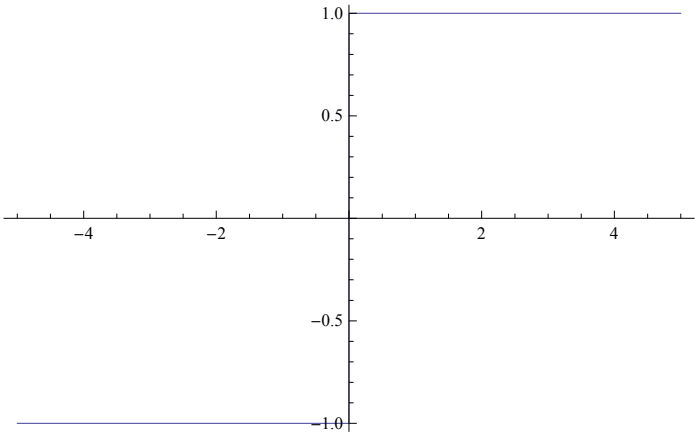
Ветвление можно реализовать при помощи If (две альтернативы) или Which (несколько альтернатив). Например, известная математическая функция сигнум принимает три значения:
-1 если $x < 0$
0 если $x = 0$
1 если $x > 0$
Так как здесь три альтернативы, то лучше использовать Which.

Важно: Логическое сравнение вводится как двойное равно (==).

In[136]:=
```
sgn[x_] := Which[
   x < 0, -1,
   x == 0, 0,
   x > 0, 1
  ];
```

In[137]:=
```
Plot[sgn[x], {x, -5, 5}]
```

Out[137]=



Для простых случаев конечно проще использовать If. Например, построим матрицу из 0 и 1 в шахматном порядке. Для этого достаточно сложить номер строки и столбца элемента и проверить, является ли сумма четной или нечетной.

In[138]:=
```
Table[If[Mod[i + j, 2] == 0, 1, 0], {i, 5}, {j, 5}]
MatrixForm[%]
```

Out[138]=
```
{{1, 0, 1, 0, 1}, {0, 1, 0, 1, 0}, {1, 0, 1, 0, 1}, {0, 1, 0, 1, 0}, {1, 0, 1, 0, 1}}
```

Out[139]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Циклы

В *Mathematica* есть стандартные циклы For и While, а также свой цикл Do. Мне кажется, что Do полностью заменяет For, поэтому достаточно знать Do и While.
Do — это способ организации цикла, когда вы заранее знаете количество шагов
While — здесь вы заранее знаете условие остановки цикла, но количество шагов заранее неизвестно

Например, сгенерируем список из 10 чисел и посчитаем сумму этих элементов. Для этого подойдет цикл Do, так как заранее известно, что нужно обойти все элементы с 1 по 10. Результат мы сохраняем в переменную result, которая изначально равна 0.

In[140]:=
```
M = RandomInteger[{-5, 5}, 10]
result = 0;
Do[result = result + M[[i]], {i, 1, 10}]
result
```

Out[140]=
```
{5, -2, -4, -1, 2, 4, 3, 5, -1, 5}
```

Out[143]=
```
16
```

Пусть теперь нам понадобилось найти сумму не всех элементов, а только до первого отрицательного элемента. Мы заранее не знаем, где тот находится, поэтому лучше использовать While. Индекс *i* нужно определить заранее. Цикл прекратится как только заданное условие даст False — в нашем случае это условие будет таким “текущее число > 0”. Но мы же легко можем выйти за границу списка, поэтому введем двойное условие “индекс не вышел за границу списка И текущее число > 0”. Логическое И вводится как ESC and ESC.

Замечание: Условие про границу поставлено первым, чтобы оно и проверялось первым. Тогда если оно не выполнится, то второе условие не будет проверяться. Это важно, так как по коду *i* может выйти за границу списка — тогда *M*[[*i*]] мог выдать ошибку.

In[144]:=
```
result = 0; i = 1;
While[i ≤ Length[M] ⋀ M[[i]] > 0, result = result + M[[i]]; i = i + 1];
result
```

Out[146]=
```
5
```

Иногда бывает полезно досрочно выйти из цикла. Для этого используется функция Break. Например, проверим есть ли в списке число 0, и если есть то выдадим True, иначе False.

In[147]:=
```
result = False;
Do[If[M[[i]] == 0, result = True; Break[]], {i, 1, 10}]
result
```

Out[149]=
```
False
```

Важно: Как вы надеюсь заметили несколько функций в коде должно разделяться точкой с

запятой (;).

Анонимные функции

Анонимная функция (или чистая функция, или pure function) — это просто функция, у которой аргумент не имеет названия (x или y), а просто обозначается как #. В конце такой функции нужно **обязательно** поставить знак &. Например, функция $y = x^2 + 1$ может быть введена так (в примере посчитано ее значение при $x = 3$).

```
In[150]:= #^2 + 1 &[3]
Out[150]= 10
```

Зачем нужны анонимные функции? Хотя бы для того, что не нужно плодить новых функций, ведь вот как тот же код выглядел бы в традиционном исполнении.

```
In[151]:= F[x_] := x^2 + 1;
In[152]:= F[3]
Out[152]= 10
```

Если у функции не одна переменная, а несколько, то их можно просто нумеровать — #1, #2, #3, ...

```
In[153]:= #1^2 + #2^2 &[3, 4]
Out[153]= 25
```

Анонимность особенно полезна при применении функций к спискам.

Применение функций к спискам

Map, Select, Nest, Fold, Apply, Flatten, Thread,

Очень мощным инструментом *Mathematica*, отличающим его от многих других систем программирования, является функциональное программирование, когда решение задачи получается в результате “композиции” функций, т.е. последовательного применения нескольких функций. Как правило, при таком подходе промежуточные данные хранятся в виде списка + активно используются анонимные функции.

Map

Основной функцией при таком подходе является Map — по сути это отображение заданного множества при помощи заданного закона. Простейший пример:

```
In[154]:= Map[h, {1, 2, 3}]
Out[154]= {h[1], h[2], h[3]}
```

В качестве h может быть использовано любое выражение, в том числе анонимная функция. Например, используем функцию $\#^2 + 1$ & из предыдущего параграфа.

```
In[155]:= Map[#^2 + 1 &, {1, 2, 3, 4, 5}]
Out[155]= {2, 5, 10, 17, 26}
```

Следующий пример: сгенерируем список пар точек, применим к ним функцию Point и нарисуем результат.

In[156]:=
```
RandomReal[{-1, 1}, {30, 2}]
Map[Point, %]
Graphics[%]
```

Out[156]=
```
{{-0.773252, 0.231161}, {0.542821, -0.275362}, {-0.013566, -0.467074},
 {-0.374887, -0.14483}, {-0.885875, -0.152739}, {-0.271308, 0.937039},
 {0.0204681, 0.11149}, {0.330576, -0.917368}, {0.86178, -0.947988},
 {0.606045, -0.354812}, {0.902751, -0.981338}, {0.223877, 0.318162},
 {-0.134551, 0.63323}, {0.625729, 0.742037}, {-0.255227, 0.385136},
 {0.331275, 0.524281}, {-0.0311763, -0.695876}, {-0.0557596, 0.263797},
 {0.951732, -0.839082}, {0.815479, -0.757813}, {0.579184, -0.102011},
 {-0.318934, -0.561957}, {0.428009, 0.926572}, {0.0797593, -0.872797},
 {0.837167, 0.806329}, {0.465436, -0.673829}, {0.578789, -0.798904},
 {0.848728, 0.369006}, {-0.296103, -0.947244}, {-0.850732, -0.761482}}
```

Out[157]=
```
{Point[{-0.773252, 0.231161}], Point[{0.542821, -0.275362}],
 Point[{-0.013566, -0.467074}], Point[{-0.374887, -0.14483}],
 Point[{-0.885875, -0.152739}], Point[{-0.271308, 0.937039}],
 Point[{0.0204681, 0.11149}], Point[{0.330576, -0.917368}],
 Point[{0.86178, -0.947988}], Point[{0.606045, -0.354812}],
 Point[{0.902751, -0.981338}], Point[{0.223877, 0.318162}],
 Point[{-0.134551, 0.63323}], Point[{0.625729, 0.742037}],
 Point[{-0.255227, 0.385136}], Point[{0.331275, 0.524281}],
 Point[{-0.0311763, -0.695876}], Point[{-0.0557596, 0.263797}],
 Point[{0.951732, -0.839082}], Point[{0.815479, -0.757813}],
 Point[{0.579184, -0.102011}], Point[{-0.318934, -0.561957}],
 Point[{0.428009, 0.926572}], Point[{0.0797593, -0.872797}],
 Point[{0.837167, 0.806329}], Point[{0.465436, -0.673829}],
 Point[{0.578789, -0.798904}], Point[{0.848728, 0.369006}],
 Point[{-0.296103, -0.947244}], Point[{-0.850732, -0.761482}]}
```

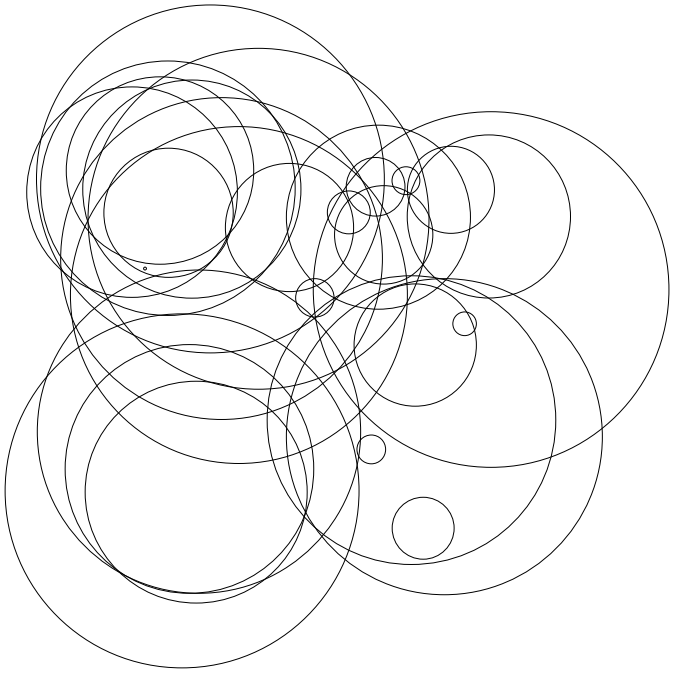
Out[158]=

Аналогично можно получить список кругов (только теперь я запишу это в одну строчку и не буду использовать %).

Важно: В этом примере Circle вызывается не просто так, а с решеткой и амперсандом (Circle[⌗, ...] &). Это потому, что у функции Circle не один параметр (список-пара чисел), а два (второй – это радиус). Мы же подаем на вход только пару чисел, поэтому нужно явно описать, что пойдет на вход вторым параметром (я использую случайное число от 0 до 1).

In[159]:=
```
Graphics[Map[Circle[#, RandomReal[1]] &, RandomReal[{-1, 1}, {30, 2}]]]
```

Out[159]=



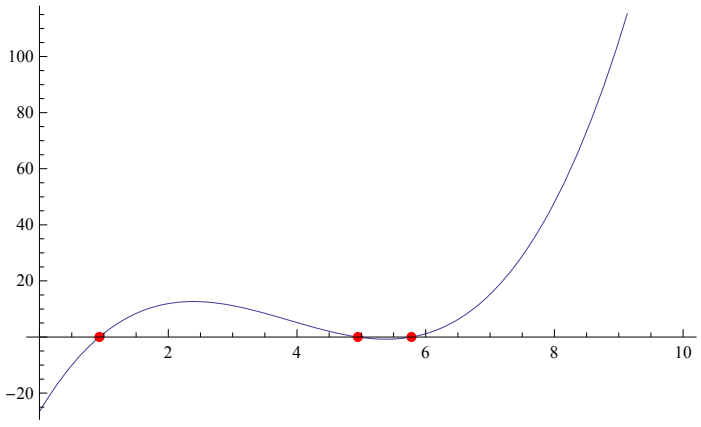
Еще пример — построим график случайного многочлена с тремя корнями на отрезке [0,10] и его корней. Такой многочлен будет иметь вид $(x - a)(x - b)(x - c)$, где a, b, c — случайные числа, которые мы предварительно сохраним для того, чтобы вывести их на график. Сам многочлен реализуем используя функцию Product (произведение).

In[160]:=
```
p = RandomReal[{0, 10}, 3]
polynom = Product[x - p[[i]], {i, 3}]
Show[
 Plot[polynom, {x, 0, 10}],
 Graphics[{PointSize[0.015], Red, Map[Point[{#, 0}] &, p]}]
]
```

Out[160]=
```
{0.929848, 5.77843, 4.94568}
```

Out[161]=
```
(-5.77843 + x) (-4.94568 + x) (-0.929848 + x)
```

Out[162]=



Select

Еще одна мощная функция — Select — позволяет выбирать из списка элементы согласно выбранного критерия. Например, выберем из списка чисел от 1 до 100 только простые числа.

In[163]:=
```
Select[Range[100], PrimeQ]
```

Out[163]=
```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}
```

В качестве критерия может служить любая функция с одним параметром, возвращающая True/False. Например, из списка чисел от 1 до 100 только те числа, сумма цифр которых делится на 9 без остатка.

In[164]:=
```
Select[Range[100], Mod[Total[IntegerDigits[#]], 9] == 0 &]
```

Out[164]=
```
{9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99}
```

Пример посложнее: набросаем много точек на квадрат $x, y \in [-1, 1]$ и выберем только те, что попали в круг радиуса 1 с центром (0,0). Выведем эти точки на экран и посчитаем их количество.

In[165]:=
```
Select[RandomReal[{-1, 1}, {10 000, 2}], #[[1]]^2 + #[[2]]^2 < 1 &];
Graphics[{Yellow, Map[Point, %]}]
num = Length[%%]
```

Out[166]=

Out[167]=
```
7907
```

Площадь квадрата равна 2 x2 = 4. Ей соответствует общее количество точек 1000, количество точек круга мы вычислили — соответственно можем по пропорции посчитать площадь круга. По идее она должна быть равна $\pi r^2 = \pi \approx 3.14$.

In[168]:=
```
N[4 num / 10 000]
```

Out[168]=
```
3.1628
```

Nest и Fold

Функции Nest и Fold используются в тех случаях, когда нужно применить одну и ту же функцию несколько раз. Например, мы положили в банк 1000 рублей на 5 лет под 15 процентов годовых. Вычислим сумму депозита по истечении срока (пусть проценты добавляются раз в год).

In[169]:=
```
Nest[# + 0.15 # &, 1000, 5]
```

Out[169]=
```
2011.36
```

Чтобы увидеть промежуточные вычисления нужно применить не Nest, а NestList.

In[170]:=
```
NestList[# + 0.15 # &, 1000, 5]
```

Out[170]=
```
{1000, 1150., 1322.5, 1520.88, 1749.01, 2011.36}
```

Действительно, в самом начале имеем 1000. В конце года к этой тысяче добавляем 15% — получаем 1000+0.15·1000=1150. Еще через год имеем 1150+0.15·1150=1322.5. И так далее. Общая формула для каждого года N+0.15·N — ее я использовал, только записав в виде анонимной функции.

Функция Fold (FoldList) отличается только тем, что на каждом шаге можно использовать дополнительный параметр. Например, пусть процент вклада каждый год меняется — 15%, 16%, 17%, 18%, 19%. Тогда код будет выглядеть так:

In[171]:=
```
FoldList[#1 + #2 #1 &, 1000, {0.15, 0.16, 0.17, 0.18, 0.19}]
```

Out[171]=
```
{1000, 1150., 1334., 1560.78, 1841.72, 2191.65}
```

Здесь #1 — это первый параметр (текущая сумма), а #2 — второй параметр, который берется из списка.

Вот еще пара примеров использования Nest и Fold.

In[172]:=
```
NestList[#^2 &, a, 10]
```

Out[172]=
$$\{a, a^2, a^4, a^8, a^{16}, a^{32}, a^{64}, a^{128}, a^{256}, a^{512}, a^{1024}\}$$

In[173]:=
```
FoldList[1 / (#2 + #1) &, x, {a, b, c, d}]
```

Out[173]=
$$\left\{x, \frac{1}{a+x}, \frac{1}{b+\frac{1}{a+x}}, \frac{1}{c+\frac{1}{b+\frac{1}{a+x}}}, \frac{1}{d+\frac{1}{c+\frac{1}{b+\frac{1}{a+x}}}}\right\}$$

Apply

Apply позволяет поменять голову у выражения. Например, в старых версиях *Mathematica* не было функции Total, поэтому для вычисления суммы элементов списка мы просто меняли голову списка List на сумму Plus.

In[174]:=
```
Apply[Plus, {1, 2, 3, 4, 5}]
```

Out[174]=
```
15
```

Например, в примере с графиком многочлена получить многочлен можно не только при помощи Product, но и при помощи связки Map + Apply. Сначала при помощи Map получить список вида *x* – *a*, а потом применить к этому списку голову “умножение” (Times).

In[175]:=
```
Map[x - # &, RandomReal[{0, 10}, 3]]
Apply[Times, %]
```

Out[175]=
```
{-2.87102 + x, -7.23031 + x, -2.12119 + x}
```

Out[176]=
```
(-7.23031 + x) (-2.87102 + x) (-2.12119 + x)
```

Преобразование выражений

Expand, Factor, TrigExpand, TrigFactor, Together, Cancel, Apart, Collect, CoefficientList, Simplify, Rule (->), ReplaceAll (/.)

В *Mathematica* есть несколько функций для преобразования (а иногда упрощения) выражений.

Expand и Factor

Эти используются для раскрытия скобок (Expand) и разложения на множители (Factor).

In[177]:=
```
Expand[(x + y)^5]
```

Out[177]=
$$x^5 + 5\,x^4\,y + 10\,x^3\,y^2 + 10\,x^2\,y^3 + 5\,x\,y^4 + y^5$$

In[178]:=
```
Factor[x^10 - 1]
```

Out[178]=
$$(-1 + x)\,(1 + x)\,\left(1 - x + x^2 - x^3 + x^4\right)\,\left(1 + x + x^2 + x^3 + x^4\right)$$

In[179]:=
```
Expand[(x + y)^3 (x - 2 y) (5 x + 6 y) (x + y^2)]
Factor[%]
```

Out[179]=
$$5\,x^6 + 11\,x^5\,y - 9\,x^4\,y^2 + 5\,x^5\,y^2 - 43\,x^3\,y^3 + 11\,x^4\,y^3 - 40\,x^2\,y^4 - 9\,x^3\,y^4 - 12\,x\,y^5 - 43\,x^2\,y^5 - 40\,x\,y^6 - 12\,y^7$$

Out[180]=
$$(x - 2\,y)\,(x + y)^3\,(5\,x + 6\,y)\,\left(x + y^2\right)$$

PowerExpand

Сократим дробь $\dfrac{\left(\sqrt[5]{\sqrt{a^{\frac{4}{3}}}}\right)^{\frac{3}{2}}\left(\sqrt{a\sqrt[3]{a^2 b}}\right)^4}{\left(\sqrt[5]{a^4}\right)^3\left(\sqrt[4]{a\sqrt{b}}\right)^6}$

In[181]:=
```
PowerExpand[(a^2 (a^(4/3))^(3/10) (a^2 b)^(2/3)) / ((a^4)^(3/5) (a Sqrt[b])^(3/2))]
```

Out[181]=
$$\frac{1}{a^{1/6}\,b^{1/12}}$$

TrigExpand и TrigFactor

При работе с тригонометрическими выражениям лучше использовать TrigExpand (раскрыть) и TrigFactor (свернуть).

In[182]:=
```
TrigExpand[Sin[2 ArcTan[t]]]
```

Out[182]=
$$\frac{2\,t}{1+t^2}$$

In[183]:=
```
TrigExpand[Cos[8 x]]
```

Out[183]=
$$\text{Cos}[x]^8-28\,\text{Cos}[x]^6\,\text{Sin}[x]^2+70\,\text{Cos}[x]^4\,\text{Sin}[x]^4-28\,\text{Cos}[x]^2\,\text{Sin}[x]^6+\text{Sin}[x]^8$$

In[184]:=
```
TrigFactor[Cos[x + y] + Sin[x] Sin[y]]
```

Out[184]=
```
Cos[x] Cos[y]
```

Together, Cancel и Apart

Эти функции помогут при работе с дробями. Together приведет к общему знаменателю:

In[185]:=
```
Together[a/b + b/a]
```

Out[185]=
$$\frac{a^2+b^2}{a\,b}$$

Cancel сократит одинаковые множители в числителе и знаменателе.

In[186]:=
```
Cancel[(x^2 - 1)/(x^3 - 1)]
```

Out[186]=
$$\frac{1+x}{1+x+x^2}$$

Apart разобьет выражение на неприводимые дроби.

In[187]:=
```
Apart[1/((x + 1) (x - 2) (x + 4))]
```

Out[187]=
$$\frac{1}{18\,(-2+x)}-\frac{1}{9\,(1+x)}+\frac{1}{18\,(4+x)}$$

Collect и CoefficientList

При работе с полиномами будут полезны функции Collect и CoefficientList. Collect соберет все коэффициенты по степеням заданной переменной. Например:

In[188]:=
```
Collect[1 + x y + x + y + x^2 y + x y^2 + x^2 y^2, x]
Collect[1 + x y + x + y + x^2 y + x y^2 + x^2 y^2, y]
```

Out[188]=
$$1+y+x^2\left(y+y^2\right)+x\left(1+y+y^2\right)$$

Out[189]=
$$1+x+\left(1+x+x^2\right)y+\left(x+x^2\right)y^2$$

Функция CoefficientList позволяет получить список коэффициентов (чтобы потом уже с ним работать отдельно). Первый коэффициент соответствует степени 0, второй — степени 1 и т.д.

In[190]:=
```
CoefficientList[(x + 1) (x - 3) (x + 5) (x - 7), x]
```

Out[190]=
```
{105, 76, -34, -4, 1}
```

Simplify

Это самая общая функция для упрощения выражений. Она знает очень много методов, хотя

иногда никакого заметного результата не приносит.

In[191]:=
```
Simplify[(x^2 - 1)/(x^3 - 1)]
```

Out[191]=
$$\frac{1+x}{1+x+x^2}$$

In[192]:=
```
Simplify[Sin[x]^2 + Cos[x]^2]
```

Out[192]=
```
1
```

In[193]:=
```
Simplify[(2 Tan[x])/(1 + Tan[x]^2)]
```

Out[193]=
```
Sin[2 x]
```

Правила подстановки

Правило подстановки (Rule) — это выражение вида *a* → *b*. Стрелка вводится как ESC –> ESC. *Mathematica* очень часто использует подстановки, например, решение системы уравнений выводится как список правил.

In[194]:=
```
Solve[{x^2 - y^2 == 10, x + y == 1}, {x, y}]
```

Out[194]=
$$\left\{\left\{x \to \frac{11}{2},\ y \to -\frac{9}{2}\right\}\right\}$$

Это удобно, так как потом это правило можно подставить в любое выражение, для чего используется функция ReplaceAll.

In[195]:=
```
ReplaceAll[x, x → 1]
```

Out[195]=
```
1
```

In[196]:=
```
ReplaceAll[x^2 + y^2, {x → 3, y → 4}]
```

Out[196]=
```
25
```

Математические вычисления

Solve, Eliminate, Reduce, FindRoot, LinearSolve, D, Integrate, Limit, Series, Minimize, Maximize

Решение уравнений и систем

Для аналитического решения уравнений и систем используется Solve. Обратите внимание, что результат выводится как список списков, где каждый подсписок — это отдельное решение.

In[197]:=
```
Solve[x^3 + a x + b == 0, x]
```

Out[197]=
$$\left\{\left\{x \to -\frac{\left(\frac{2}{3}\right)^{1/3} a}{\left(-9\,b+\sqrt{3}\,\sqrt{4\,a^3+27\,b^2}\right)^{1/3}}+\frac{\left(-9\,b+\sqrt{3}\,\sqrt{4\,a^3+27\,b^2}\right)^{1/3}}{2^{1/3}\,3^{2/3}}\right\},\right.$$
$$\left\{x \to \frac{\left(1+i\sqrt{3}\right) a}{2^{2/3}\,3^{1/3}\left(-9\,b+\sqrt{3}\,\sqrt{4\,a^3+27\,b^2}\right)^{1/3}}-\frac{\left(1-i\sqrt{3}\right)\left(-9\,b+\sqrt{3}\,\sqrt{4\,a^3+27\,b^2}\right)^{1/3}}{2\times 2^{1/3}\,3^{2/3}}\right\},$$
$$\left.\left\{x \to \frac{\left(1-i\sqrt{3}\right) a}{2^{2/3}\,3^{1/3}\left(-9\,b+\sqrt{3}\,\sqrt{4\,a^3+27\,b^2}\right)^{1/3}}-\frac{\left(1+i\sqrt{3}\right)\left(-9\,b+\sqrt{3}\,\sqrt{4\,a^3+27\,b^2}\right)^{1/3}}{2\times 2^{1/3}\,3^{2/3}}\right\}\right\}$$

Конечно, аналитическое решение возможно не всегда, поэтому в таких случаях ничего не остается кроме как найти приближенное решение.

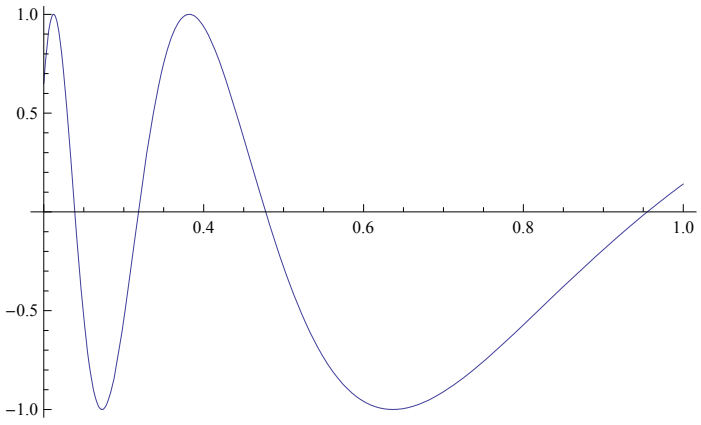
In[198]:=
```
Solve[x^5 + x - 1 == 0, x];
% // N
```

Out[199]=
```
{{x → 0.5 + 0.866025 i}, {x → 0.5 - 0.866025 i}, {x → 0.754878},
 {x → -0.877439 + 0.744862 i}, {x → -0.877439 - 0.744862 i}}
```

У сложных функций найти все корни на отрезке довольно сложно, поэтому их ищут численным методом по одному, указывая начальное приближение. Для этого используется функция FindRoot. Посмотрите, какие корни функции Sin[3/x] находятся при разных начальных данных.

In[200]:=
```
Plot[Sin[3 / x], {x, 0.2, 1}]
```

Out[200]=



In[201]:=
```
FindRoot[Sin[3 / x], {x, 0.1}]
FindRoot[Sin[3 / x], {x, 0.2}]
FindRoot[Sin[3 / x], {x, 0.3}]
FindRoot[Sin[3 / x], {x, 0.4}]
FindRoot[Sin[3 / x], {x, 0.5}]
```

Out[201]= {x → 0.0795775}

Out[202]= {x → 0.190986}

Out[203]= {x → 0.31831}

Out[204]= {x → 0.477465}

Out[205]= {x → 0.477465}

Иногда нужно не решить систему, а исключить некоторые переменные из нее, получив из двух уравнений одно и т.п.

```
In[206]:= Eliminate[{x + y == 1, 5 x - 2 y == 3}, x]
Out[206]= 7 y == 2
```

```
In[207]:= Eliminate[{x + y - z == 0, 3 x + y + 2 z == 7, 3 y - z == 5}, {y, z}]
Out[207]= 13 x == -1
```

Решение неравенств возможно при помощи функции Reduce.

```
In[208]:= Reduce[(x - 2) (x + 3) (x - 1) > 0 && x > 0]
Out[208]= 0 < x < 1 || x > 2
```

В качестве параметра этой функции можно задавать область допустимых значений, например, Integers (целые числа), Reals (вещественные), Complexes (комплексные).

```
In[209]:= Reduce[x^4 + 2 x^3 - x - 2 == 0, x, Integers]
Out[209]= x == -2 || x == 1
```

Решение линейных систем

Для решения линейных систем существуют специальные функции, например, LinearSolve. Для решения нужно задать в качестве параметров функции матрицу и столбец свободных членов.

```
In[210]:= LinearSolve[{{1, 1, 1}, {2, 4, 6}, {3, 9, 27}}, {1, 2, 3}]
Out[210]= {1, 0, 0}
```

Проверим правильность решение путем прямого вычисления $A^{-1} \cdot b$

```
In[211]:= Inverse[{{1, 1, 1}, {2, 4, 6}, {3, 9, 27}}].{1, 2, 3}
Out[211]= {1, 0, 0}
```

Производные и интегралы

Операции нахождения производной и интеграла осуществляются при помощи D и Integrate соответственно. Проверим, являются ли эти операции обратными. После применения Simplify видно, что это так.

In[212]:=
```
Integrate[x^5/Sqrt[x^3 - 1], x]
D[%, x]
Simplify[%]
```

Out[212]= $\frac{2}{9}\sqrt{-1+x^3}\,\left(2+x^3\right)$

Out[213]= $\frac{2}{3}x^2\sqrt{-1+x^3}+\frac{x^2\,(2+x^3)}{3\sqrt{-1+x^3}}$

Out[214]= $\frac{x^5}{\sqrt{-1+x^3}}$

Однако, нужно помнить, что *Mathematica* — это сборник алгоритмов, и мыслить пока она не умеет, поэтому случаются казусы. В примере при дифференцировании была “съедена” константа, которая затем не была добавлена при интегрировании — в результате ответ получился верный с точностью до константы.

In[215]:=
```
(x - 3)^3/3
Expand[D[%, x]]
Integrate[%, x]
```

Out[215]= $\frac{1}{3}\,(-3+x)^3$

Out[216]= $9-6\,x+x^2$

Out[217]= $9\,x-3\,x^2+\frac{x^3}{3}$

Система позволяет находить производные любого порядка.

In[218]:=
```
D[Sin[Cos[x]], {x, 5}]
```

Out[218]=
```
-Cos[Cos[x]] Sin[x] + 15 Cos[x]^2 Cos[Cos[x]] Sin[x] - 10 Cos[Cos[x]] Sin[x]^3 -
 Cos[Cos[x]] Sin[x]^5 + 15 Cos[x] Sin[x] Sin[Cos[x]] + 10 Cos[x] Sin[x]^3 Sin[Cos[x]]
```

Суммы и произведения

Mathematica также умеет находить аналитические суммы отрезков рядов.

In[219]:=
```
Sum[k^2, {k, n}]
```

Out[219]= $\frac{1}{6}\,n\,(1+n)\,(1+2\,n)$

И произведения

In[220]:=
```
Product[i^2, {i, 1, n}]
```

Out[220]= $(n!)^2$

Пределы

И, конечно, же *Mathematica* знает основные пределы.

In[221]:=
```
Limit[(1 + 1/n)^n, n → ∞]
```

Out[221]= e

Кроме того, система позволяет находить односторонние пределы.

In[222]:=
```
Limit[1/(1 - e^(x/(1-x))), x → 0, Direction → 1]
Limit[1/(1 - e^(x/(1-x))), x → 0, Direction → -1]
```

Out[222]= ∞

Out[223]= −∞

Разложение в ряд Тейлора

Для разложения функции в ряд Тейлора нужно использовать Series. Результат дополняется “O-большим”, который можно убрать при помощи Normal.

In[224]:=
```
Series[x Sin[x], {x, 0, 10}]
Normal[%]
```

Out[224]= $x^2 - \frac{x^4}{6} + \frac{x^6}{120} - \frac{x^8}{5040} + \frac{x^{10}}{362\,880} + O[x]^{11}$

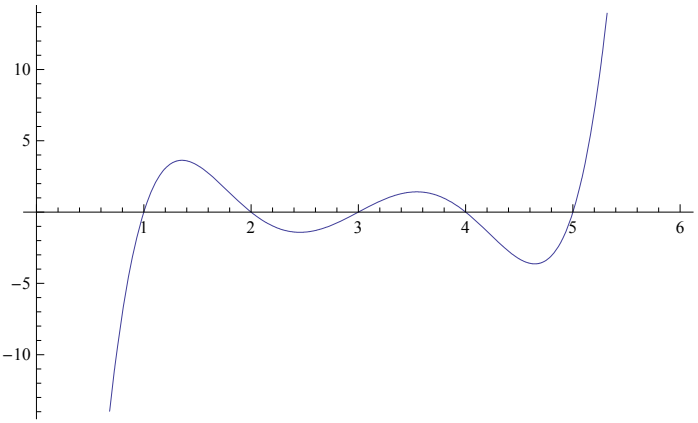
Out[225]= $x^2 - \frac{x^4}{6} + \frac{x^6}{120} - \frac{x^8}{5040} + \frac{x^{10}}{362\,880}$

Нахождение минимального/максимального значения функции

Найдем минимальное и максимальное значение функции на отрезке [1, 5].

In[226]:=
```
f = (x - 1) (x - 2) (x - 3) (x - 4) (x - 5);
Plot[f, {x, 0, 6}]
NMinimize[{f, x ≥ 1, x ≤ 5}, x]
NMaximize[{f, x ≥ 1, x ≤ 5}, x]
```

Out[227]=



Out[228]= {−3.63143, {x → 4.64443}}

Out[229]= {3.63143, {x → 1.35557}}

Мини-справка

Сокращенная форма функций

В *Mathematica* все есть выражение, поэтому конструкции вида $a + b$, $x \to 1$, $y + y^2 /. y \to 2$ все равно на “математическом” языке выглядят как *f*[arg1, arg2, ...]. Но знать некоторые сокращения все равно не помешает. Приведу сокращения, которые применяются чаще всего.

Part[*h*, 1, 2]	сокращается как	*h*[[1, 2]]
Map[*f*, {*a*, *b*, *c*}]	==	*f* /@ {*a*, *b*, *c*}
Apply[*f*, expr]	==	*f* @@ expr
Rule[*a*, *b*]	==	$a \to b$
ReplaceAll[*x*, $x \to 1$]	==	$x /. x \to 1$

Полезные пункты меню

Показать подсказку при наборе функций — Edit > Complete Selection (Ctrl+K)
Вставить заголовок — Format > Style > Title (Alt+1)
Вставить секцию — Format > Style > Section (Alt+4)
Вставить текстовую ячейку — Format > Style > Text (Alt+7)
Удалить все выводы из файла (для сокращения размера) — Cell > Delete All Output
Выполнить весь файл — Evaluation > Evaluate Notebook
Остановить вычисление — Evaluation > Abort Evaluation (Alt+точка)
Закрыть ядро — Evaluation > Quit Kernel > Local
Основная палитра — Palettes > Basic Math Assistant