Компьютерные вычисления (введение в систему *Mathematica*)

Лекция 1

Лекция 2

Лекция 3

Ввод и вывод

Вывести на печать результаты вычислений можно не только опустив “;” в конце строки. Можно еще пользоваться командой Print. Она печатает подряд без пробелов все свои аргументы. Чтобы разделять выводимые поля, их следует перемежать текстовыми строками (заключенными в двойные кавычки). В текстовые строки можно, в свою очередь, вставлять переменные при помощи команды StringForm.

In[1]:=
```
Do[Print["n = ", n, StringForm["   2`1` = `2`", n, 2^n]], {n, 5}];
```

```
n = 1,   2^1 = 2
n = 2,   2^2 = 4
n = 3,   2^3 = 8
n = 4,   2^4 = 16
n = 5,   2^5 = 32
```

Представление структур в Математике не всегда самое наглядное. Чтобы не преобразовывать при печати ответ всякий раз к требуемому виду, предусмотрена команда Format. Она не меняет значения выражения, а только лишь выводит выражение в требуемом виде. Предположим, что у нас имеется степенной ряд, коэффициенты которого нумеруются разбиениями (диаграммами Юнга).

In[2]:=
```
F = Sum_{n=1}^{5} Sum_{λ}^{IntegerPartitions[n]} Y[λ] Product_{i}^{λ} x_i
```

Out[2]=
$$x_1\,Y[\{1\}] + x_2\,Y[\{2\}] + x_3\,Y[\{3\}] + x_4\,Y[\{4\}] + x_5\,Y[\{5\}] + x_1^2\,Y[\{1,\,1\}] + x_1\,x_2\,Y[\{2,\,1\}] + x_2^2\,Y[\{2,\,2\}] + x_1\,x_3\,Y[\{3,\,1\}] + x_2\,x_3\,Y[\{3,\,2\}] + x_1\,x_4\,Y[\{4,\,1\}] + x_1^3\,Y[\{1,\,1,\,1\}] + x_1^2\,x_2\,Y[\{2,\,1,\,1\}] + x_1\,x_2^2\,Y[\{2,\,2,\,1\}] + x_1^2\,x_3\,Y[\{3,\,1,\,1\}] + x_1^4\,Y[\{1,\,1,\,1,\,1\}] + x_1^3\,x_2\,Y[\{2,\,1,\,1,\,1\}] + x_1^5\,Y[\{1,\,1,\,1,\,1,\,1\}]$$

Вот как можно сделать внешний вид разбиений более наглядным.

In[3]:=
```
Format[Y[λ_List]] := Y_Row[λ]
F
```

Out[4]=
$$x_1\,Y_1 + x_2\,Y_2 + x_3\,Y_3 + x_4\,Y_4 + x_5\,Y_5 + x_1^2\,Y_{11} + x_1\,x_2\,Y_{21} + x_2^2\,Y_{22} + x_1\,x_3\,Y_{31} + x_2\,x_3\,Y_{32} + x_1\,x_4\,Y_{41} + x_1^3\,Y_{111} + x_1^2\,x_2\,Y_{211} + x_1\,x_2^2\,Y_{221} + x_1^2\,x_3\,Y_{311} + x_1^4\,Y_{1111} + x_1^3\,x_2\,Y_{2111} + x_1^5\,Y_{11111}$$

Также для более приятного ввода и вывода в Математике имеется большое количество функций с сокращенными инфиксными и префиксными обозначениями, у которых никакие значения изначально никак не заданы. Это скобки разных видов, бинарные операции вида $\cdot,\circ,\wedge,\otimes,\oplus$ и многие другие. Их можно использовать, чтобы определять свои собственные

операции (подобно тому, как мы на прошлой лекции определяли операцию склейки деревьев). Но нужно понимать, что все равно за каждым сокращенным обозначением стоит соответствующая функция со своим заголовком и т.п.

In[5]:=
```
a ∧ b // FullForm
```

Out[5]//FullForm=
```
Wedge[a, b]
```

Наконец, если и этого не хватает, в Математике предусмотрена возможность вводить свои собственные обозначения. Для этого нужно подключить дополнительно пакет Notation

In[6]:=
```
Needs["Notation`"]
```

Пример собственного обозначения для предела

In[7]:=
```
Notation[ lim f_ ⟺ Limit[f_, x_ → a_] ]
```

In[8]:=
```
lim (Sin[Tan[x]] - Tan[Sin[x]]) / (ArcSin[ArcTan[x]] - ArcTan[ArcSin[x]])
x→0
```

Out[8]=
```
1
```

А вот пример собственной операции возведения в квадрат

In[9]:=
```
Notation[ ■x_ ⟺ x_^2 ]
```

In[10]:=
```
■5
(a + b)^3 // Expand
```

Out[10]=
```
25
```

Out[11]=
```
a^3 + 3 (■a) b + 3 a (■b) + b^3
```

Отладка и повышение быстродействия

Если программа “зависла” или слишком долго считает, можно прервать вычисления, набрав одновременно “ALT,” или “ALT.” В самом крайнем случае можно в меню выбрать Evaluation→Quit Kernel. Чтобы к этим кардинальным средствам прибегать как можно реже, полезно контролировать ход вычислений. Простейший прием - вставлять в код отладочные печати. Более хитрый состоит в использовании динамических выражений

In[12]:=
```
Dynamic[n]
```

Out[12]=
```
10
```

Значение такого выражения обновляется автоматически в реальном времени по мере его изменения

In[13]:=
```
n = 0
Do[++n; Pause[0.2], {20}]
```

Out[13]=
```
0
```

Приведу еще несколько приемов, повышающих эффективность работы Математики (это лишь те приемы, которые встречались в моей практике. В действительности их наверняка гораздо больше).
В комбинаторике часто приходится иметь дело с матрицами больших размеров, большинство компонент которых равны нулю. Матрицы такого вида лучше всего релизовывать при помощи команды SparseArray. В дальнейшем с ними можно обращаться как с обычными матрицами - складывать, перемножать, искать собственные числа, собственные векотры, вычислять определитель и т.п.

In[15]:=
```
M = SparseArray[{{i_, i_} → 1, {2, 5} → 5}, {6, 6}]
```
Out[15]=
```
SparseArray[<7>, {6, 6}]
```

In[16]:=
```
M // MatrixForm
M.M // MatrixForm
```

Out[16]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Out[17]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 10 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Если результаты сложных и объемных вычислений хранятся в виде списка замен большого размера, то для повышения эффективности обращения к такому списку можно применить к нему команду Dispatch. Эта команда упаковывает список замен, то есть приводит к виду, удобному для эффективного обращения к нему ядра Математики.

In[18]:=
```
repl = Table[x_i → i^2, {i, 10 000}];
```

In[19]:=
```
Short[repl]
```
Out[19]//Short=
```
{x_1 → 1, <<9998>>, x_10 000 → 100 000 000}
```

In[20]:=
```
Timing[Sum[x_i /. repl, {i, 10 000}]]
```
Out[20]=
```
{4.015625, 333 383 335 000}
```

In[21]:=
```
disp = Dispatch[repl];
```

In[22]:=
```
Timing[Sum[x_i /. disp, {i, 10 000}]]
```
Out[22]=
```
{0.015625, 333 383 335 000}
```

Кстати, мы использовали здесь команду Timing - еще одно средство контроля эффективности выполняемых расчетов.

При рекурсивном определении функций можно запоминать уже вычисленные значения, чтобы не заставлять рекурсию всякий раз работать заново. Это повышает скорость вычислений на несколько порядков. Вот определение многочленов Чебышева

In[23]:=
```
Clear[T];
T /: T1 = x;
T /: Tn_ := x Tn-1 - (1 - x^2)/(n - 1) ∂x Tn-1 // Expand;
Timing[T20]
```

Out[26]=
$$\{16.140625,\ 1 - 200\ (\blacksquare x) + 6600\ x^4 - 84\,480\ x^6 + 549\,120\ x^8 - 2\,050\,048\ x^{10} + 4\,659\,200\ x^{12} - 6\,553\,600\ x^{14} + 5\,570\,560\ x^{16} - 2\,621\,440\ x^{18} + 524\,288\ x^{20}\}$$

А вот как это считается, если сохранять промежуточные вычисления

In[27]:=
```
Clear[T];
T /: T1 = x;
T /: Tn_ := T /: Tn = x Tn-1 - (1 - x^2)/(n - 1) ∂x Tn-1 // Expand;
Timing[T20]
```

Out[30]=
$$\{0.,\ 1 - 200\ (\blacksquare x) + 6600\ x^4 - 84\,480\ x^6 + 549\,120\ x^8 - 2\,050\,048\ x^{10} + 4\,659\,200\ x^{12} - 6\,553\,600\ x^{14} + 5\,570\,560\ x^{16} - 2\,621\,440\ x^{18} + 524\,288\ x^{20}\}$$

Здесь “/:” используется для того, чтобы определение T_n было связано именно с символом T, а не с заголовком Subscript, и чтобы командой Clear[*T*] можно было очищать определения в процессе отладки.

Если проделанные в Математике вычисления хочется сохранить до следующей сессии работы с программой, вот как это можно сделать

In[31]:=
```
SetDirectory[NotebookDirectory[]]
```

Out[31]=
```
C:\Users\maxim_2\Documents\maxim\WNMATH\Разное
```

In[32]:=
```
DumpSave["tmp.mx", {T}]
```

Out[32]=
```
{T}
```

Без указания директории будет использоваться системная (которая, ко всему прочему, скорее всего, защищена от записи). Вместо DumpSave можно использовать Save. Отличие в том, что Save сохраняет в текстовом виде, а DumpSave - в бинарном внутреннем формате Математики, что, конечно же, эффективнее. Теперь можно выйти из программы или остановить ядро, а потом загрузить сохраненный файл

In[1]:=
```
SetDirectory[NotebookDirectory[]];
<< tmp.mx
```

Все определения, связанные с символом T (как вычисленные многочлены Чебышева, так и рекурсивная формула для них) восстановились полностью.

In[3]:=
```
? T
```

```
Global`T
```

```
T /: Subscript[T, 1] = x
```

```
T /: Subscript[T, 2] = -1 + 2 x^2
```

```
T /: Subscript[T, 3] = -3 x + 4 x^3
```

```
T /: Subscript[T, 4] = 1 - 8 x^2 + 8 x^4
```

```
T /: Subscript[T, 5] = 5 x - 20 x^3 + 16 x^5
```

```
T /: Subscript[T, 6] = -1 + 18 x^2 - 48 x^4 + 32 x^6
```

```
T /: Subscript[T, 7] = -7 x + 56 x^3 - 112 x^5 + 64 x^7
```

```
T /: Subscript[T, 8] = 1 - 32 x^2 + 160 x^4 - 256 x^6 + 128 x^8
```

```
T /: Subscript[T, 9] = 9 x - 120 x^3 + 432 x^5 - 576 x^7 + 256 x^9
```

```
T /: Subscript[T, 10] = -1 + 50 x^2 - 400 x^4 + 1120 x^6 - 1280 x^8 + 512 x^10
```

```
T /: Subscript[T, 11] = -11 x + 220 x^3 - 1232 x^5 + 2816 x^7 - 2816 x^9 + 1024 x^11
```

```
T /: Subscript[T, 12] = 1 - 72 x^2 + 840 x^4 - 3584 x^6 + 6912 x^8 - 6144 x^10 + 2048 x^12
```

```
T /: Subscript[T, 13] = 13 x - 364 x^3 + 2912 x^5 - 9984 x^7 + 16640 x^9 - 13312 x^11 + 4096 x^13
```

```
T /: Subscript[T, 14] = -1 + 98 x^2 - 1568 x^4 + 9408 x^6 - 26880 x^8 + 39424 x^10 - 28672 x^12 + 8192 x^14
```

```
T /: Subscript[T, 15] =
 -15 x + 560 x^3 - 6048 x^5 + 28800 x^7 - 70400 x^9 + 92160 x^11 - 61440 x^13 + 16384 x^15
```

```
T /: Subscript[T, 16] =
 1 - 128 x^2 + 2688 x^4 - 21504 x^6 + 84480 x^8 - 180224 x^10 + 212992 x^12 - 131072 x^14 + 32768 x^16
```

```
T /: Subscript[T, 17] =
 17 x - 816 x^3 + 11424 x^5 - 71808 x^7 + 239360 x^9 - 452608 x^11 + 487424 x^13 - 278528 x^15 + 65536 x^17
```

```
T /: Subscript[T, 18] = -1 + 162 x^2 - 4320 x^4 + 44352 x^6 -
   228096 x^8 + 658944 x^10 - 1118208 x^12 + 1105920 x^14 - 589824 x^16 + 131072 x^18
```

```
T /: Subscript[T, 19] = -19 x + 1140 x^3 - 20064 x^5 + 160512 x^7 -
   695552 x^9 + 1770496 x^11 - 2723840 x^13 + 2490368 x^15 - 1245184 x^17 + 262144 x^19
```

```
T /: Subscript[T, 20] = 1 - 200 x^2 + 6600 x^4 - 84480 x^6 + 549120 x^8 -
   2050048 x^10 + 4659200 x^12 - 6553600 x^14 + 5570560 x^16 - 2621440 x^18 + 524288 x^20
```

```
Subscript[T, n_] ^:=
 T /: Subscript[T, n] = Expand[x Subscript[T, n - 1] - ((1-x^2) ∂_x Subscript[T,n-1])/(n-1)]
```

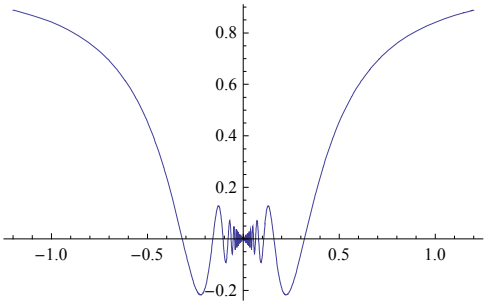
Графика

Возможности работы с графикой в Математике поистине безграничны. Все аспекты описать невозможно. Я приведу лишь несколько начальных простых примеров, совершенствовать и усложнять которые слушатели смогут самостоятельно.

Простейшая графическая команда - Plot, для рисования графика функции

In[4]:=
```
Plot[x Sin[1 / x], {x, -1.2, 1.2}]
```

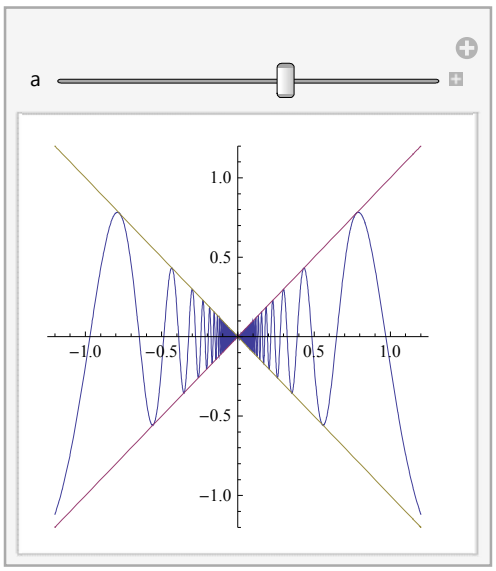
Out[4]=



Сделаем несколько усовершенствований. 1) добавим еще две функции x и -x для наглядности; 2) добавим в конце команды опции PlotRange для задания границ выводимых значений и AspectRatio для согласования масштабов по осям; 3) добавим параметр и простой манипулятор для изменения его значений

In[5]:=
```
Manipulate[
 Plot[{x Sin[a / x], x, -x}, {x, -1.2, 1.2},
  PlotRange → {-1.2, 1.2}, AspectRatio → 1],
 {a, 0, 10}]
```

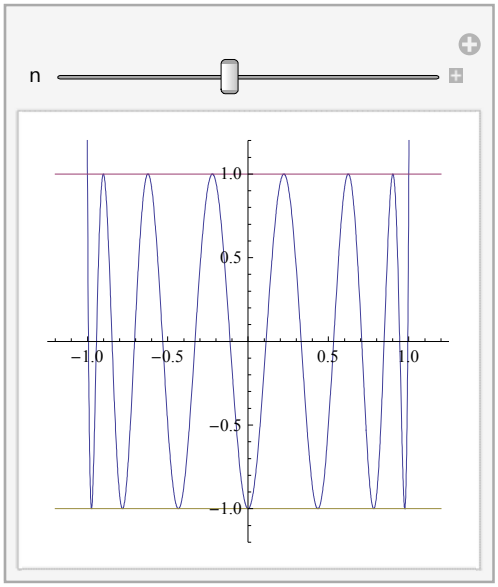
Out[5]=



А вот знакомые нам многочлены Чебышева. У них все критические точки вещественны, половина из них имеет критическое значение 1, а остальные -1. Для команыд Plot “лучше”, если выводимая функция вычислена заранее, а не вычисляется в процессе построения графика.

```
Manipulate[y = Tn;
 Plot[{y, 1, -1}, {x, -1.2, 1.2},
  PlotRange → {-1.2, 1.2}, AspectRatio → 1],
 {n, 1, 30, 1}]
```

Out[6]=

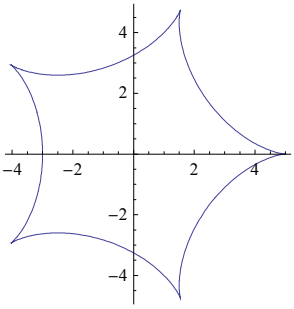


Здесь я использовал команду ParametricPlot, чтобы нарисовать циклоиду.

In[7]:=

```
r[ϕ_] := {Cos[ϕ], Sin[ϕ]};
n = 4;
ParametricPlot[n r[ϕ] + r[-n ϕ], {ϕ, 0, 2 π}]
```

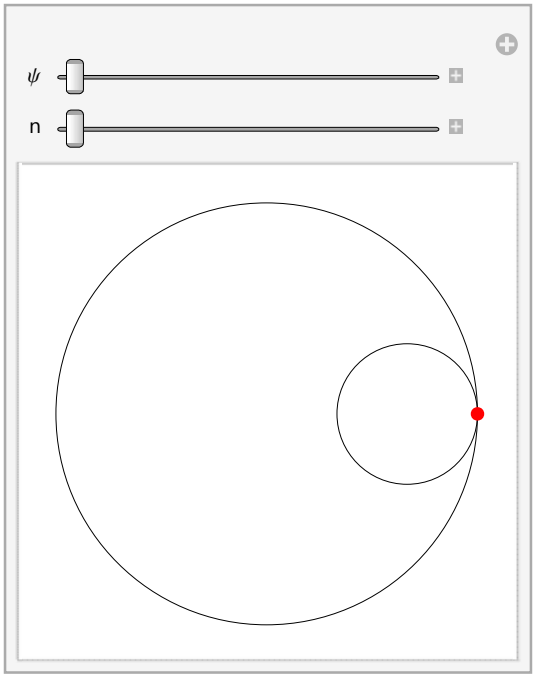
Out[9]=



Чтобы нарисовать что-нибудь вручную, нужно использовать окружение Graphics. Графика состоит из последовательности “Примитив” и “Директив”, сгрупированных в фигурных скобках. Примитивы - это “что мы рисуем” - точка, круг, линия, прямоугольник и т.д. Дирекивы - это “как мы рисуем” - цвет, заливка, толщина линий и т.д. Вот простейшая графика, состоящая из двух окружностей и точки.

In[10]:=
```
Manipulate[
 Graphics[{Circle[{0, 0}, n + 1], Circle[n r[ψ], 1],
   {PointSize[Large], Red, Point[n r[ψ] + r[-n ψ]]}}],
 {ψ, -2 π, 2 π}, {n, 2, 10, 1}]
```

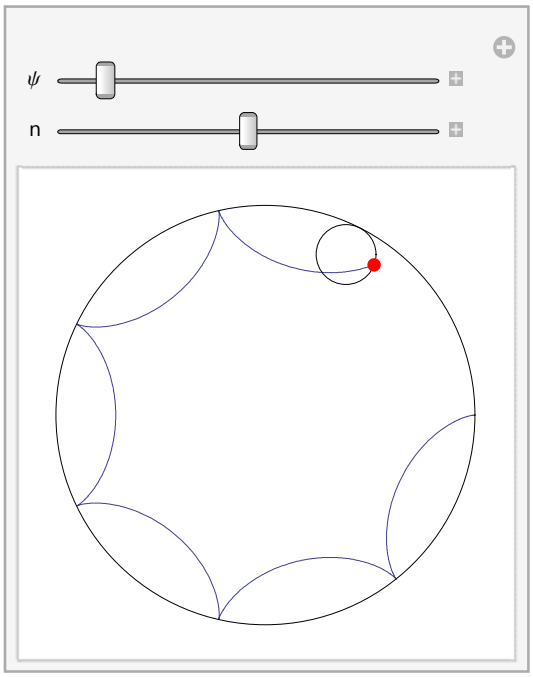
Out[10]=



Построенная выше циклоида - это след точки на маленькой окружности, катящейся по большой. А как наложить два изображения друг на друга? Это можно сделать с помощью команды Show. Кроме того, в Show можно добавить графические опции, которые будут применены ко всем использованным внутри графическим командам

In[11]:=
```
Manipulate[
 Show[
  ParametricPlot[n r[ϕ] + r[-n ϕ], {ϕ, 0, ψ}],
  Graphics[{Circle[{0, 0}, n + 1], Circle[n r[ψ], 1],
    {PointSize[Large], Red, Point[n r[ψ] + r[-n ψ]]}
   }],
  Axes → False, PlotRange → (n + 1) {{-1, 1}, {-1, 1}}],
 {ψ, -2 π, 2 π}, {n, 2, 10, 1}]
```

Out[11]=



В следующем примере добавим немного интерактивности. Пусть имеется матрица, состоящая из нулей и единиц.

In[12]:=
```
n = 10;
A = RandomInteger[1, {n, n}];
MatrixForm[A]
```

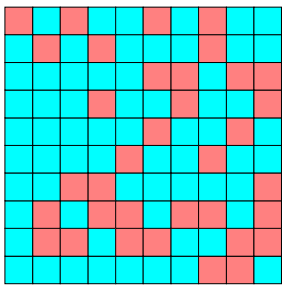
Out[14]//MatrixForm=

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

Изобразим эту матрицу в виде шахматной доски, поля которой раскрашены в два цвета в зависимости от значений компонент матрицы. Заметим, что координаты в графике имеют привычное обозначение (первая слева направо, вторая снизу вверх). Для согласованности с традиционной нумерацией строк и столбцов матрицы координаты придется повернуть на 90 градусов.

In[15]:=
```
Graphics[Table[{EdgeForm[Thin], If[A[[n+1-j,i]] == 0, Cyan, Pink], Rectangle[{i, j}]},
    {i, n}, {j, n}], ImageSize → 150] //
 Row[{#, "  A = ", MatrixForm[A]}] &
```

Out[15]=

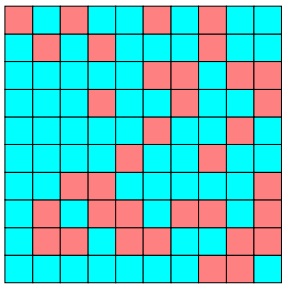


$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0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

А теперь сделаем так, чтобы по клику мышки цвет ячейки (и соответствующая компонента матрицы) менялись. Для этого используем команду ClickPane. Ее первый аргумент - сама графика (к которой применен Dynamic), а второй аргумент - функция, которая выполняется при нажатии на мышку. Аргументом функции служат координаты мышки. В нашем случае мы округляем координаты до целых и меняем соответствующую компоненту матрицы.

In[16]:=
```
ClickPane[Dynamic[
  {Graphics[
     Table[{EdgeForm[Thin], If[A[[n+1-j,i]] == 0, Cyan, Pink], Rectangle[{i, j}]},
      {i, n}, {j, n}], ImageSize → 150], "  A = ", MatrixForm[A]} // Row],
 ({i, j} = Round[# - 1/2]; If[1 ≤ i ≤ n && 1 ≤ j ≤ n, A[[n+1-j,i]] = 1 - A[[n+1-j,i]]]) &]
```

Out[16]=



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

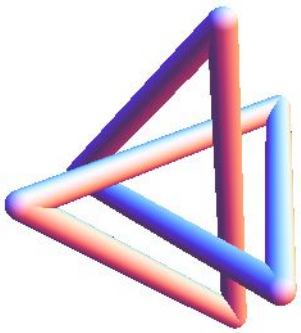
Помимо двумерной графики, в Математике реализована трехмерная. Такие команды как Plot3D и ParametricPlot3D рисуют поверхности, с помощью Graphics3d можно создавать трехмерные объекты вручную - многогранники, сферы, и т.п. Трехмерные изображения можно вращать при помощи мышки.

In[17]:=
```
Clear[pp];
pp[a_] := Join @@ Table[{Cos[2 π (i + a z) / 3], Sin[2 π (i + a z) / 3], z},
     {i, 0, 2}, {z, {-1, 1}}] // Append[#, #[[1]]] &;
pp[0]
```

Out[19]=
$$\left\{\{1, 0, -1\}, \{1, 0, 1\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}, -1\right\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}, 1\right\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}, -1\right\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}, 1\right\}, \{1, 0, -1\}\right\}$$

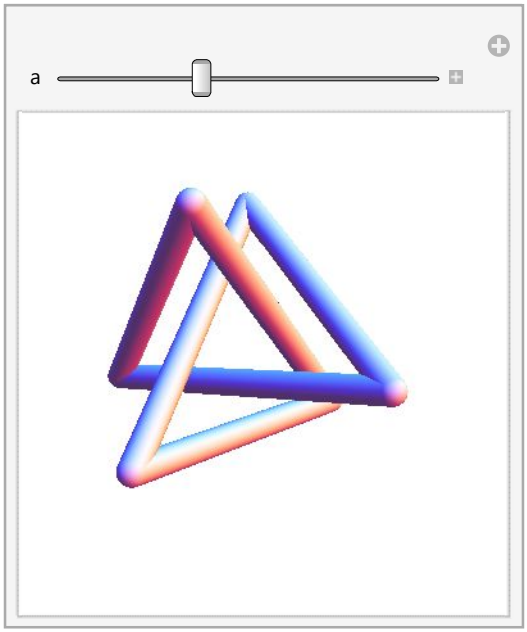
In[20]:=
```
Graphics3D[{Tube[pp[1.1], 0.1]}, Boxed → False]
```

Out[20]=



In[21]:=
```
Manipulate[
 Graphics3D[{Tube[pp[a], .1]}, Boxed → False], {{a, 1.1}, 0, 3}]
```

Out[21]=



Заключение

Мы рассмотрели лишь малую часть возможностей, предоставляемых Математикой. Больше можно узнать, читая Help'ы и рассматривая всевозможные примеры, собранные в проекте Demonstrations на сайте Wolfram (сылка имеется в пункте меню Help).