



Формулы

Символы

(:= esc)

:a: → α

:b: → β

:γ: → γ

...

:f: → φ :j: → ϕ

:e: → ε :ce: → ε

:w: → ω :o: → ω :om: → o

:deg: → °

In[1]:= α β φ ϕ

Out[1]= α β φ ϕ

Набор формул

Ctrl + 6 → □⁶

Ctrl + 2 → √□

... + Ctrl + 5 → ∛□

Ctrl + / → □
□

Ctrl + Space → end subexpression

Alt + / → (*comment*)

Alt +),], } → (), [], {}

Ctrl + , → □ □ □

□

Ctrl + Enter → □

□

... → $\begin{pmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{pmatrix}$

Выражения

Everything is an Expression

```
Head[a1, a2, a3,...]
```

In[2]:=

```
In[3]:= FullForm[a + b]
```

```
Out[3]/FullForm= Plus[a, b]
```

In[4]:= F[x, y]

```
Out[4]= F[x, y]
```

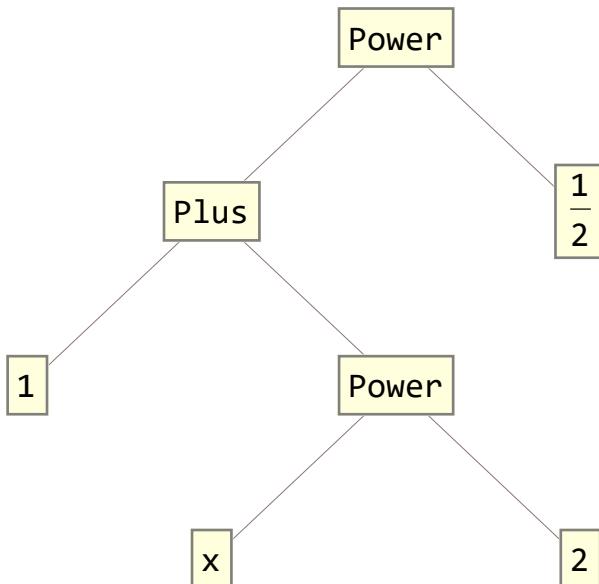
In[5]:=

Expressions

```
Head []
FullForm []
TreeForm []
```

In[6]:= TreeForm[$\sqrt{1 + x^2}$]

Out[6]/TreeForm=



In[7]:=

```
In[8]:= FullForm[{1, 2, 3}]
```

```
Out[8]//FullForm= List[1, 2, 3]
```

Для вызова справки выделить имя и нажать F1

Part[] - [] (* [] < - [[,]] < - *)

Range[]

Table[]

Subdivide[]

Правила преобразования

```
In[9]:= 2 + 2
```

```
Out[9]= 4
```

```
In[10]:= 2 + x
```

```
Out[10]= 2 + x
```

Задание правил

```
In[11]:= x = 4;
```

```
In[12]:= ? x
```

Global`x

x = 4

```
In[13]:=
```

```
In[14]:= 2 + x
```

```
Out[14]= 6
```

Удаление всех правил, ассоциированных с символом

```
In[15]:= Clear[x]
```

```
In[16]:= ? x
```

Global`x

```
In[17]:=
```

Set[] (=)

SetDelayed[] (:=)

Простое задание правила через `=` не подходит для определения функций, поскольку вычисляет правую часть до создания правила

```
In[18]:= x = 2;
F[x_] = x^2;
F[3]
```

```
Out[20]= 4
```

правило с отложенным вычислением

```
In[21]:= F[x_] := x^2;
F[3]
```

```
Out[22]= 9
```

```
In[23]:= F[0] := 5;
F[x_] := x^2;
F[x_, __] := x^3;
```

```
In[26]:= ? F
```

Global`F

```
F[0] := 5
```

```
F[x_] := x^2
```

```
F[x_, __] := x^3
```

```
In[27]:= F[0]
```

```
Out[27]= 5
```

```
In[28]:= F[3]
```

```
Out[28]= 9
```

```
In[29]:= F["kjhsdgfkjh"]
```

```
Out[29]= kjhsdgfkjh2
```

```
In[30]:= F[0] := 0;
F[1] := 1;
F[n_] := F[n - 1] + F[n - 2]; (*определение с использованием рекурсии*)
{F[0], F[1], F[2], F[3], F[4], F[5], F[6], F[7], F[8], F[9]}
```

```
Out[33]= {0, 1, 1, 2, 3, 5, 8, 13, 21, 34}
```

Patterns

Specifying Types (`F[x_type] := ...`)

Optional (`F[x_: x0] := ...`)

PatternTest (`F[x_?test] := ...`)

Optional with PatternTest (`F[x : (_?test) : x0] := ...`)

Condition (`F[x_ /; test] := ...`)

Sequence of variables (`F[x__] := ...`)

```
Parameterized function (F[α_][x_] := ...)
```

```
...
```

Применение функции к списку аргументов = замена заголовка List именем функции:

```
In[34]:= Apply[F, {s, 5}]
```

```
Out[34]= s^3
```

краткая запись

```
In[35]:= F @@ {s, 5}
```

```
Out[35]= s^3
```

```
In[36]:= ArcTan @@ (a + b)
```

```
Out[36]= ArcTan[a, b]
```

```
In[37]:= Clear[F];
```

Три эквивалентных способа записи применения функции к одному аргументу

```
In[38]:= F@ (a)
```

```
F[a]
```

```
(a) // F
```

```
Out[38]= F[a]
```

```
Out[39]= F[a]
```

```
Out[40]= F[a]
```

(порядок применения операторов при преобразовании к полной форме)

```
In[41]:=
```

Безымянные (анонимные) функции

```
In[42]:= Function[{x, y, z}, z + y^2]
```

```
Out[42]= Function[{x, y, z}, z + y^2]
```

```
In[43]:= Function[{x, y, z}, z + y^2][4, 5, 6]
```

```
Function[{x, y, z}, z + y^2] @@ {4, 5, 6}
```

```
Out[43]= 31
```

```
Out[44]= 31
```

краткая запись с использованием безымянных аргументов

```
In[45]:= (#^2) &
```

```
(#1^2 + #2) &
```

```
Out[45]= #1^2 &
```

```
Out[46]= #1^2 + #2 &
```

```
In[47]:= (#2) &@ 5
Out[47]= 25

In[48]:= (#12 + #2) & @@ {5, 3}
Out[48]= 28

In[49]:= (*использование анонимной функции в качестве аргумента другой функции*)
SortBy[{1, 2, 3, 4, 5, 6}, N[Sin[#]] &]
Out[49]= {5, 4, 6, 3, 1, 2}
```

Функциональное программирование

Многие функции по умолчанию применяются к спискам поэлементно

```
In[50]:= Sin[{1, 2, 3}]
Out[50]= {Sin[1], Sin[2], Sin[3]}
```

За это отвечает атрибут `Listable`

```
In[51]:= Attributes[Sin]
Out[51]= {Listable, NumericFunction, Protected}
```

В частности это позволяет складывать числа со списками или использовать общий множитель

```
In[52]:= 1 + {1, 2, 3}
2 * {1, 2, 3}
Out[52]= {2, 3, 4}
Out[53]= {2, 4, 6}

In[54]:= Attributes[Plus]
Out[54]= {Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}
```

Поэлементное применение функций без атрибута `Listable` (что, в частности, необходимо для анонимных функций)

```
In[55]:= Map[(#2) &, {1, 2, 3}]
Out[55]= {1, 4, 9}
```

краткая запись

```
In[56]:= (#2) & /@ {1, 2, 3}
Out[56]= {1, 4, 9}
```

Functional Operations

```
Map[] ( /@ )
Nest[]
Fold[]
```

```
In[57]:= Nest[1 + 1 &, y, 7]
Out[57]= 1 + 1
           1 + 1
           1 + 1
           1 + 1
           1 + 1
           1 + 1
           1 + 1
           y

In[58]:= TexForm[1 + 1
                  1 + 1
                  1 + 1
                  1 + 1
                  1 + 1
                  1 + 1
                  1 + 1
                  y]

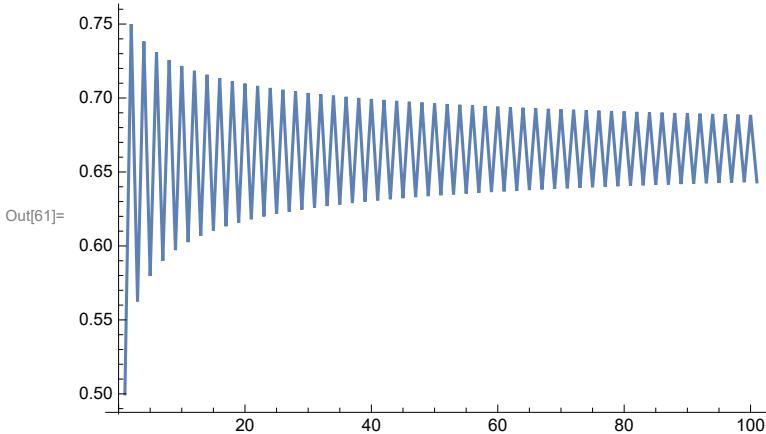
Out[58]//TeXForm=
\frac{1}{\frac{1}{\frac{1}{\frac{1}{\frac{1}{\frac{1}{\frac{1}{y}}}}}}}
```

Бифуркационная диаграмма

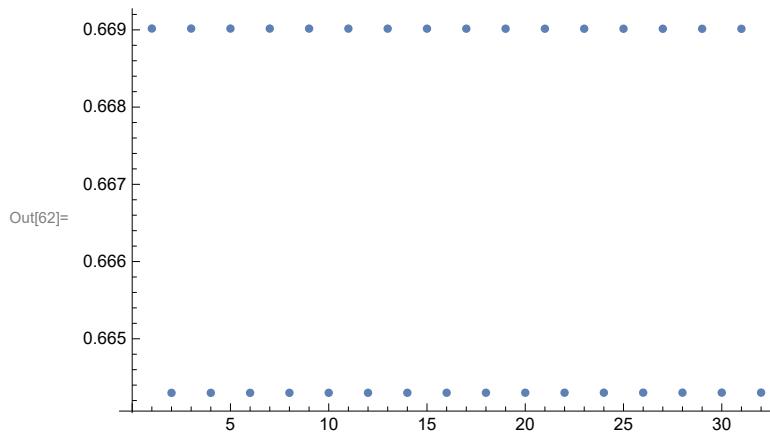
Логистическое отображение

$$x_{n+1} = k x_n (1 - x_n)$$

```
In[59]:= k = 3;
k # (1 - #) &;
(*динамика системы*)
NestList[k # (1 - #) &, 0.5, 100] // ListLinePlot
```

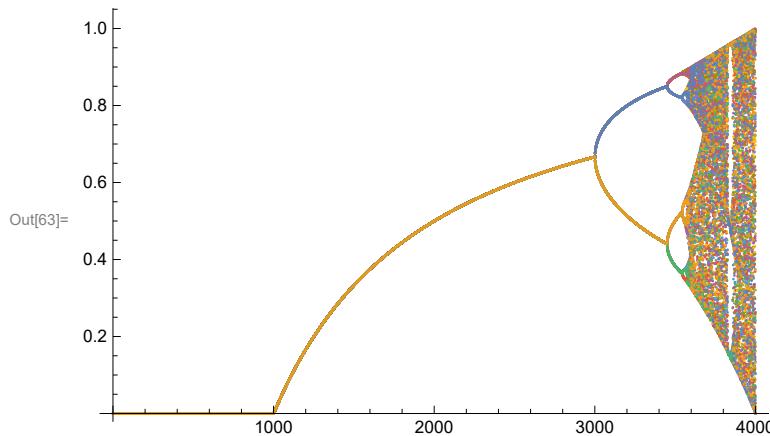


```
In[62]:= (*предельные точки*)
NestList[k # (1 - #) &, 0.5, 10000] [[-32 ;;]] // ListPlot
```



In[63]:= (*полный код для построения бифуркационной диаграммы*)

```
ListPlot[Table[NestList[k # (1 - #) &, 0.5, 10000][[-32 ;;]], {k, 0, 4, .001}]^T]
```



In[64]:= ListLinePlot[Table[Outer[List, {k}, Sort[NestList[k # (1 - #) &, .5, 10000][[-32 ;;]]]]^T, {k, 0, 4, .001}]^T, AxesLabel → {"k", "x"}, LabelStyle → Large, ImageSize → Large]

