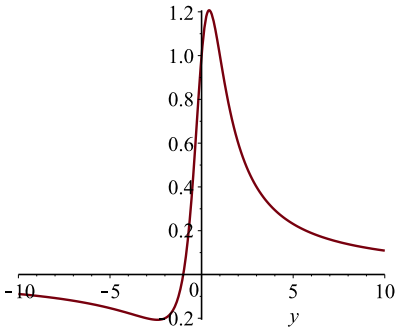
Непосредственно после запуска системы компьютерной алгебры Maple для построения простейших графиков можно использовать процедуру **plot**. Эта процедура локализована в ядре системе и не требует загрузки каких-либо вспомогательных средств.

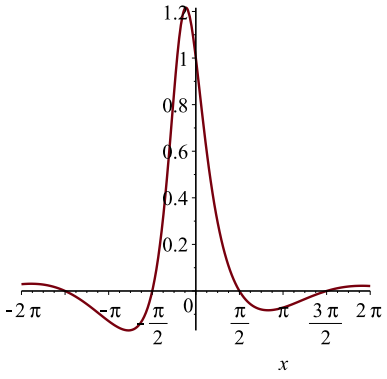
У этой процедуры есть только один обязательный параметр, представляющий собой синтаксически корректное выражение (с точки зрения Maple) той зависимости, которую мы хотели бы отобразить. Выражение должно зависеть только от *одной* независимой переменной. В этом случае система сама догадается как называется независимая переменная.

```
> plot((1+y)/(1+y^2));
```



При вызове процедуры в такой *спартанской* форме считается, что аргумент изменяется в замкнутом диапазоне от -10 до 10. Однако, если выражение выражение содержит тригонометрические функции, то нижняя граница диапазона изменения аргумента принимается равной $-2\,\pi$. Соответственно верхняя граница полагается равной $2\,\pi$.

```
> plot(cos(x)/(1+x+x^2));
```



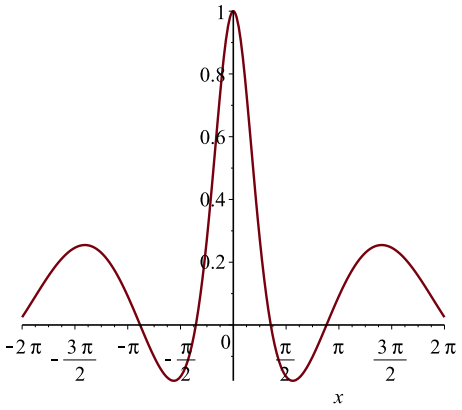
Частным случаем выражения являются вызовы функций. Maple допускает использование в выражениях вещественных функций от одной вещественной переменной.

Проиллюстрируем эту возможность на простом примере:

```
> f:=x->(1-x*sin(x))/(1+x^2);
```

$$f := x \to \frac{1 - x\sin(x)}{1 + x^2}$$

```
> plot(f(x));
```

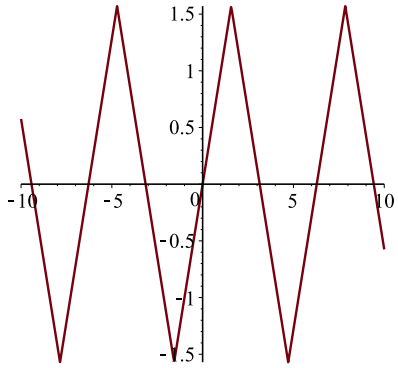


Если выражение содержит *только* вызов какой-то функции и больше ничего, то аргумент этой функции *можно и не указывать*!

```
> g:=t->arcsin(sin(t));
```

$$g := t \to \arcsin(\sin(t))$$

```
> plot(g);
```



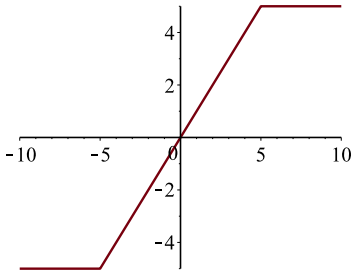
Система компьютерной алгебры Maple обладает достаточно развитым языком программирования, позволяя записывать не только простейшие оператор-функции,

показанные выше, но и гораздо более сложные зависимости. Такие зависимости удобно оформлять в виде процедур.

```
> R:=proc(x) if abs(x)<5 then x else 5*signum(x) fi end;
```

$$R := \mathbf{proc}(x)\ \mathbf{if}\ \mathrm{abs}(x) < 5\ \mathbf{then}\ x\ \mathbf{else}\ 5 * \mathrm{signum}(x)\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

```
> plot(R);
```



Обратите внимание, что в этом случае **нельзя** указывать параметр процедуры!

```
> plot(R(x));
Error, (in R) cannot determine if this expression is true or false: abs(x) < 5
```

В выражениях можно использовать ссылки на другие переменные. Однако этим переменным должно быть присвоено некоторое численное значение.

Вот что бывает, если мы забываем это сделать.

```
> plot(sin(a*x));
Error, (in plot) cannot determine plotting variable
```

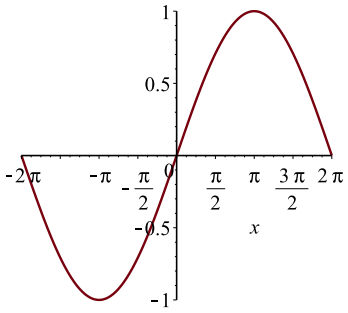
Проинициализируем переменную *a*

```
> a:=1/2;
```

$$a := \frac{1}{2}$$

Теперь всё хорошо.

```
> plot(sin(a*x));
```

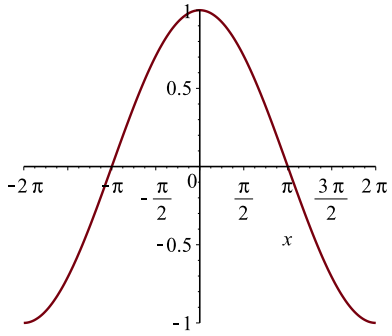


Таких ссылок может быть несколько.

```
> b:=Pi/2;
```

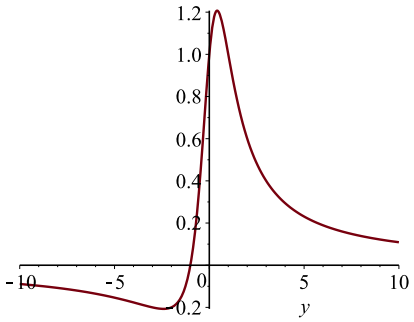
$$b := \frac{1}{2}\,\pi$$

```
> plot(sin(a*x+b));
```



Второй параметр процедуры **plot** является необязательным. Если обратиться к справочной системе, то, на первый взгляд, его функциональность кажется весьма ограниченной. Действительно, он позволяет указать имя независимой переменной. Например, самый первый пример из этого документа можно записать и в таком виде:

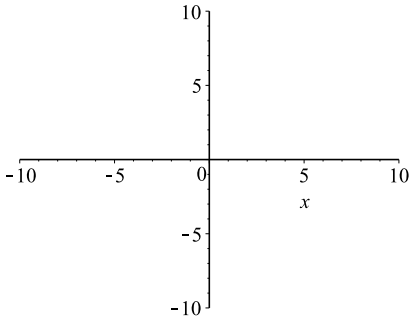
```
> plot((1+y)/(1+y^2),y);
```



Однако, как мы уже видели ранее, Maple и так в состоянии определить имя этой переменной. Более того, невнимательность или различного рода опечатки могут привести к весьма неожиданному результату

```
> plot((1+y)/(1+y^2),x);
```

Warning, expecting only range variable x in expression (1+y)/(y^2+1) to be plotted but found name y



Выведенное синим цветом предупреждение поясняет причину появления пустого изображения.

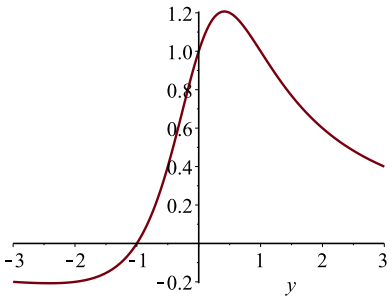
Основное назначение второго параметра состоит в том, что с его помощью *можно указать диапазон изменения независимой переменной.*

В системе Maple диапазон задаётся нижней и верхней границами, между которыми поставлены *без пробелов* две точки. Границы диапазона должны представлять собой вещественные числа. С точки зрения Maple это означает, что границы принадлежат типу *numeric.*

Формально второй параметр **plot** должен представлять собой либо имя независимой переменной, либо уравнение, в левой части которого стоит имя независимой переменной. Правая часть уравнения будет являться диапазоном изменения этой переменной.

Поскольку принимаемые по умолчанию границы диапазона, как правило, неудобны, то с помощью второго параметра мы имеем возможность отображать только те части графика, которые нам интересны.

```
> plot((1+y)/(1+y^2),y=-3..3);
```



В частности, границы диапазона по умолчанию могут находиться за пределами области допустимых значений. Тогда значительная часть графика просто пустая.

```
> plot(sqrt(x)/(1+x));
```

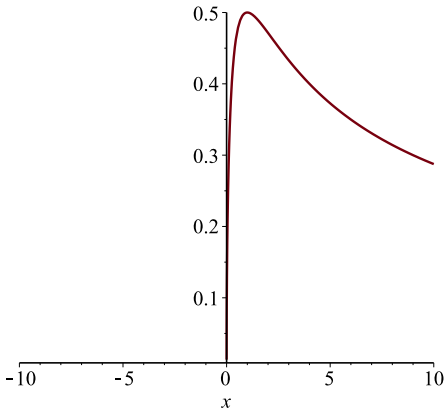
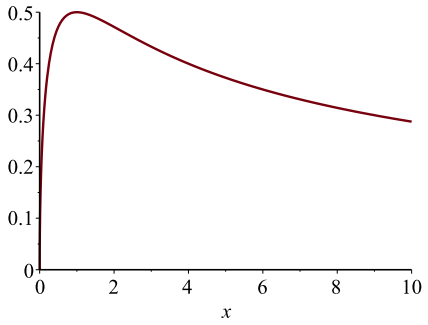


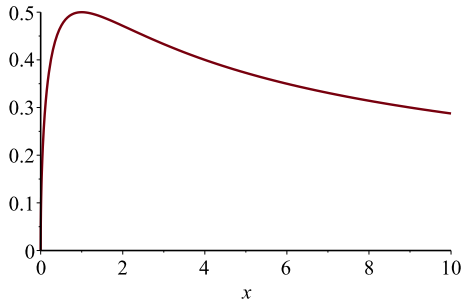
Рисунок станет более содержательным, если то же самое записать в виде

```
> plot(sqrt(x)/(1+x),x=0..10);
```



Границы диапазона могут быть перепутаны. В большинстве случаев график всё равно нарисуется. Однако это порочная практика программирования.

```
> plot(sqrt(x)/(1+x),x=10..0);
```



А вот пробелов между точками лучше не допускать!

```
> plot(sqrt(x)/(1+x),x=0. .10);
```

```
Error, (in plot) unexpected option: x = 0.
```

Границы диапазона могут быть выражениями, которые ссылаются на инициализированные ранее переменные.

```
> a:=1;
```

$$a := 1$$

```
> plot(sqrt(x)/(1+x),x=a..2*a+3);
```