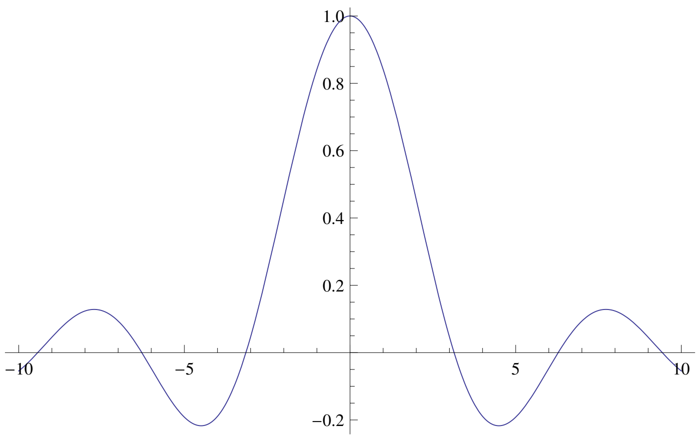
```
(*  Elemente der Programmierung,  WQ FS 2013 *)
```

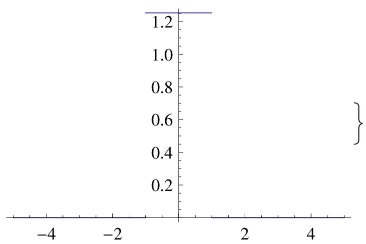
```
(* Beispiel  :=  gegen =  *)
z1 = Random[]
z2 := Random[]
Table[z1, {6}]
Table[z2, {6}]
```

```
0.261039

{0.261039, 0.261039, 0.261039, 0.261039, 0.261039, 0.261039}

{0.807207, 0.708977, 0.713343, 0.260002, 0.950567, 0.931403}
```

```
(* Beispiel  :=  gegen =  *)
Plot[Sinc[x], {x, -10, 10}]
```



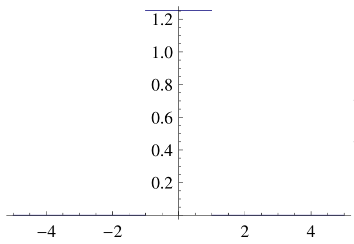

```
g[ω_] := FourierTransform[Sinc[t], t, ω]
Plot[g[x], {x, -5, 5}] // Timing
```

{1.109375,  }

```
h[ω_] = FourierTransform[Sinc[t], t, ω]
Plot[h[x], {x, -5, 5}] // Timing
```

$$\frac{1}{2}\sqrt{\frac{\pi}{2}}\ (\mathrm{Sign}[1-\omega]+\mathrm{Sign}[1+\omega])$$

{0.015625,



}

```
(* Beispiel  :=  gegen =  , sowie  :=  und =  *)
fib[n_Integer] := Switch[n, 0, 0, 1, 1, _, fib[n - 1] + fib[n - 2]]
```

```
Timing[fib[25]]
```

{1.062500, 75 025}

```
Timing[fib[29]]
```

{7.312500, 514 229}

```
Clear[fib]
```

```
fib[n_Integer] := fib[n] = Switch[n, 0, 0, 1, 1, _, fib[n - 1] + fib[n - 2]]
```

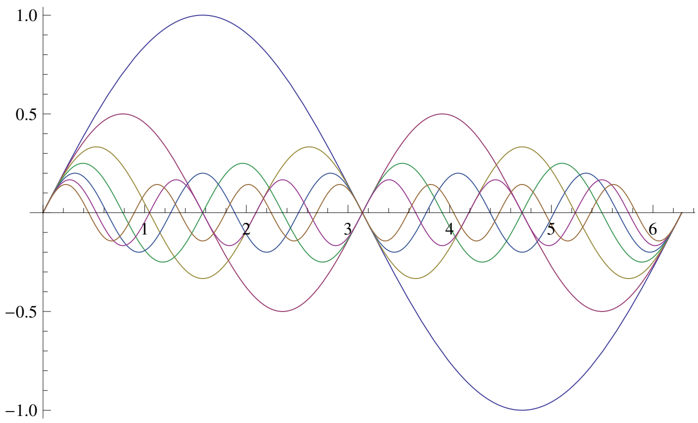
```
Timing[fib[15]]
```

{0., 610}

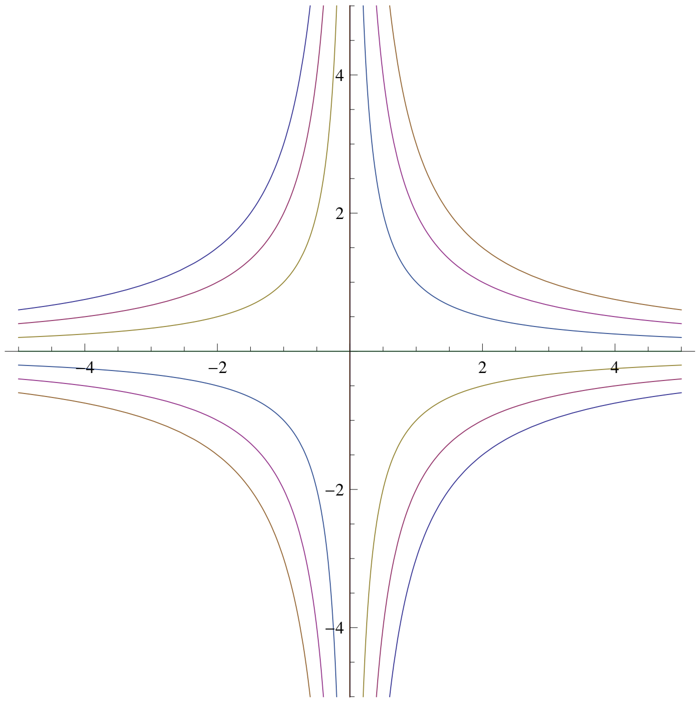
```
? fib
```

```
Global`fib

fib[0] = 0

fib[1] = 1

fib[2] = 1

fib[3] = 2

fib[4] = 3

fib[5] = 5

fib[6] = 8

fib[7] = 13

fib[8] = 21

fib[9] = 34

fib[10] = 55

fib[11] = 89

fib[12] = 144

fib[13] = 233

fib[14] = 377

fib[15] = 610

fib[n_Integer] := fib[n] = Switch[n,
   0, 0,
   1, 1,
   _, fib[n - 1] + fib[n - 2]]

Timing[fib[111]]

{0., 70 492 524 767 089 125 814 114}

(* aber Rekursionstiefe noch beschraenkt,
im Modul geht es noch besser: beachte Buendelzuweisung!  *)

fibM[n_Integer] := Module[{a = 0, b = 1}, Do[{a, b} = {b, a + b}, {n}]; a]

Timing[fibM[1000]]

{0.,
 43 466 557 686 937 456 435 688 527 675 040 625 802 564 660 517 371 780 402 481 729 089 536 ⋱
  555 417 949 051 890 403 879 840 079 255 169 295 922 593 080 322 634 775 209 689 623 239 873 ⋱
  322 471 161 642 996 440 906 533 187 938 298 969 649 928 516 003 704 476 137 795 166 849 228 ⋱
  875}

(*  noch 'schneller' ist die MatrixPotenz - Rechnung,
wie in vorangegangener Ueb *)
```

```
(* Funktionen mit eigenen Optionen *)

multPlot[func_, list1_List, list2_List] :=
 Plot[Evaluate[Table[func, list1]], list2]

multPlot[Sin[n x] / n, {n, 1, 7}, {x, 0, 2 Pi}]
```




```
multPlot2[func_, list1_List, list2_List, Opts___?OptionQ] :=
 Plot[Evaluate[Table[func, list1]], list2, Opts]

multPlot2[c / x, {c, -3, 3}, {x, -5, 5}, PlotRange → {-5, 5}, AspectRatio → 1]
```



```
allgMat[localOpts___?OptionQ] :=
 myallgMat[{x_{#1,#2} &, zeilen, spalten} /. {localOpts }
   /. {zeilen → 2, spalten → 2}]
myallgMat[{f_, z_, s_}] := Table[f[i, j], {i, 1, z}, {j, 1, s}]
```

```
MatrixForm /@ {allgMat[], allgMat[spalten → 5]}
```

$$\left\{\begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix}, \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} \end{pmatrix}\right\}$$

```
sehrallgMat[localOpts___?OptionQ] :=
 myallgMat[{f, zeilen, spalten} /. {localOpts}
   /. {f → (x_{#1,#2} &) , zeilen → 2, spalten → 2}]
(* Beispiel: Matrix mit Zufallszahlen *)
sehrallgMat[f → Function[{x, y}, Random[]], zeilen → 3] // MatrixForm
```

$$\begin{pmatrix} 0.162918 & 0.623117 \\ 0.545559 & 0.60417 \\ 0.398436 & 0.321387 \end{pmatrix}$$

```
(*  Funktionen mit optionalen Parametern *)

potReihe[f_, x_: x, xo_: 0, order_: 6] := Series[f, {x, xo, order}]
potReihe[Sin[x], x]
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} + O[x]^7$$

```
potReihe[Sin[x], x, Pi, 3]
```

$$-(x - \pi) + \frac{1}{6}(x - \pi)^3 + O[x - \pi]^4$$

```
(* 'bessere' Programmierung ist aber,
Standardwerte als Optionen zu formulieren *)

Clear[potReihe]
Options[potReihe] := {val → x, value → 0, order → 6};
potReihe[f_, Opts___?OptionQ] :=
 Series[f, {var, value, order} /. {Opts} /. Options[potReihe] ]
potReihe[Sin[y], var -> y, order → 7]
```

$$y - \frac{y^3}{6} + \frac{y^5}{120} - \frac{y^7}{5040} + O[y]^8$$