

```

[ >
> # Plan:
# Umsetzung der Algorithmen der vergangenen Doppelstunden
# - (Erweiterter) Euklidischer Algorithmus
# - Divide and conquer-Version eines Multiplik.-Algorithmus
# - Bestimmung einer Quadratwurzel
# - Ggf. Maple-Funktionen zu Grafik, Analysis

> # Euklidischer Algorithmus:
# Gegeben: a,b von Null verschiedene
# Berechnet Folge von Resten r_1,r_2,...
# Erste Idee: Daten r[i]
# Beobachtung: Speicherung aller Reste nicht notwendig,
# da jeweils nur die beiden letzten benoetigt werden.
euclid := proc(a,b) # Annahme: a,b von Null verschieden
  local j, r1, r2, m, h, s1, s2, t1, t2; # aktuell letzten Reste
  r1 := abs(a); r2 := abs(b); s1 := 1; s2 := 0; t1 := 0; t2 := 1;
  j := 0;
  while (r2 <> 0) do
    j := j+1;
    m := iquo(r1,r2); # Ganzzahlanteil des Quotienten
    printf("%3d %4d %4d %4d %4d %4d %4d %4d \n",
      j,r1,r2,m,s1,s2,t1,t2);
    h := r1 - m*r2; r1 := r2; r2 := h;
    h := s1 - m*s2; s1 := s2; s2 := h;
    h := t1 - m*t2; t1 := t2; t2 := h;
  end:
  printf("ggT : %10d \n", r1);
  printf("Ganzzahlige Lin.-Komb: %4d * %4d + %4d * %4d \n",
s1,a,t1,b);
  return(r1);
end proc:

euclid(9876, 3456);
euclid(42,125);
euclid(-42, 125);

>
1  9876 3456    2    1    0    0    1
2  3456 2964    1    0    1    1   -2
3  2964  492    6    1   -1   -2    3
4   492   12   41   -1    7    3   -20
ggT : 12
Ganzzahlige Lin.-Komb: 7 * 9876 + -20 * 3456

1  42 125    0    1    0    0    1
2  125 42    2    0    1    1    0
3  42 41    1    1   -2    0    1
4  41  1   41   -2    3    1   -1
ggT : 1
Ganzzahlige Lin.-Komb: 3 * 42 + -1 * 125

1  42 125    0    1    0    0    1

```

```

  2   125   42   2   0   1   1   0
  3   42   41   1   1  -2   0   1
  4   41   1   41  -2   3   1  -1
ggT : 1
Ganzzahlige Lin.-Komb: 3 * -42 + -1 * 125
                        1
> 7*9876 - 20*3456;
                        12
[ > help(iquo);
[ > help("if");
[ > a:=1; if (a=1) then print("ja"); fi;
                        a := 1
                        "ja"
[ > a := [27,48,12,39];
gr := 1; for i from 2 to 4 do
  if (a[1] < a[i]) then
    gr := 0;
  end if;
end do;
if (gr = 1) then
  print("Erste Zahl groesste Zahl.");
else
  print("Nicht so.");
end if;
                        a := [27, 48, 12, 39]
                        gr := 1
                        "Nicht so."
[ > # Naive divide and conquer-Multiplikation
# Multipliziere zwei Zahlen a,b der Länge n=2^k
# Annahme: Mult. mit Zweier-/Zehnerpotenz effizient zur
#   Verfuegung stehend

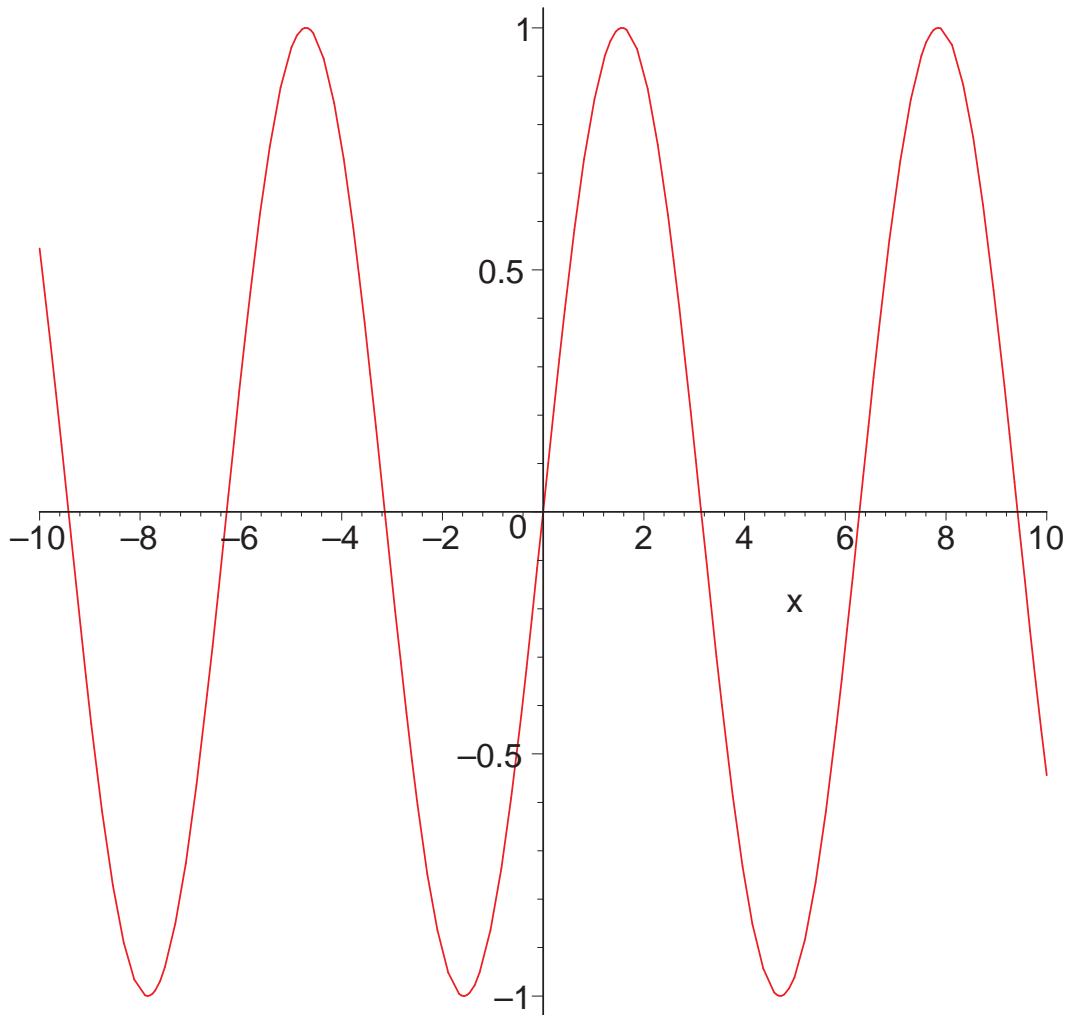
mymult := proc(n,a,b)
local u,v,w,x;
  if (n=1) then
    return(a*b);
  else # Zerlegung: a in u,v , b in w,x
    u := iquo(a,10^(n/2));
    v := irem(a,10^(n/2));
    w := iquo(b,10^(n/2));
    x := irem(b,10^(n/2));
    print("u,v,w,x: ", u,v,w,x);
    return(mymult(n/2,u,w)*10^n +
           (mymult(n/2,u,x)+mymult(n/2,v,w))*10^(n/2) +
           mymult(n/2,v,x));
  end if;
end proc:

```

```

> mymult(4,1111,1111);
                                "u,v,w,x: ", 11, 11, 11, 11
                                "u,v,w,x: ", 1, 1, 1, 1
                                "u,v,w,x: ", 1, 1, 1, 1
                                "u,v,w,x: ", 1, 1, 1, 1
                                "u,v,w,x: ", 1, 1, 1, 1
                                1234321
[ > with(LinearAlgebra):
[ > # Wurzelbestimmung mit Babylonischem Verfahren
# Bestimme Quadratwurzel von a näherungsweise
babylonisch := proc(a, x0)
# Startwert x0
local f, i, xx;
# Iterationsvorschrift
f := x-> 1/2 *(x+a/x);
xx := x0;
for i from 1 to 10 do
xx := evalf(f(xx));
end;
evalf(xx,50);
end proc:
[ > babylonisch(2,10);
print(evalf(% - sqrt(2),50));
                                1.414213562
                                -0.3730950488016887242096980785696718753769 10-9
[ >
[ > # Funktionen und grafische Darstellungen
[ > plot(sin(x),x);

```



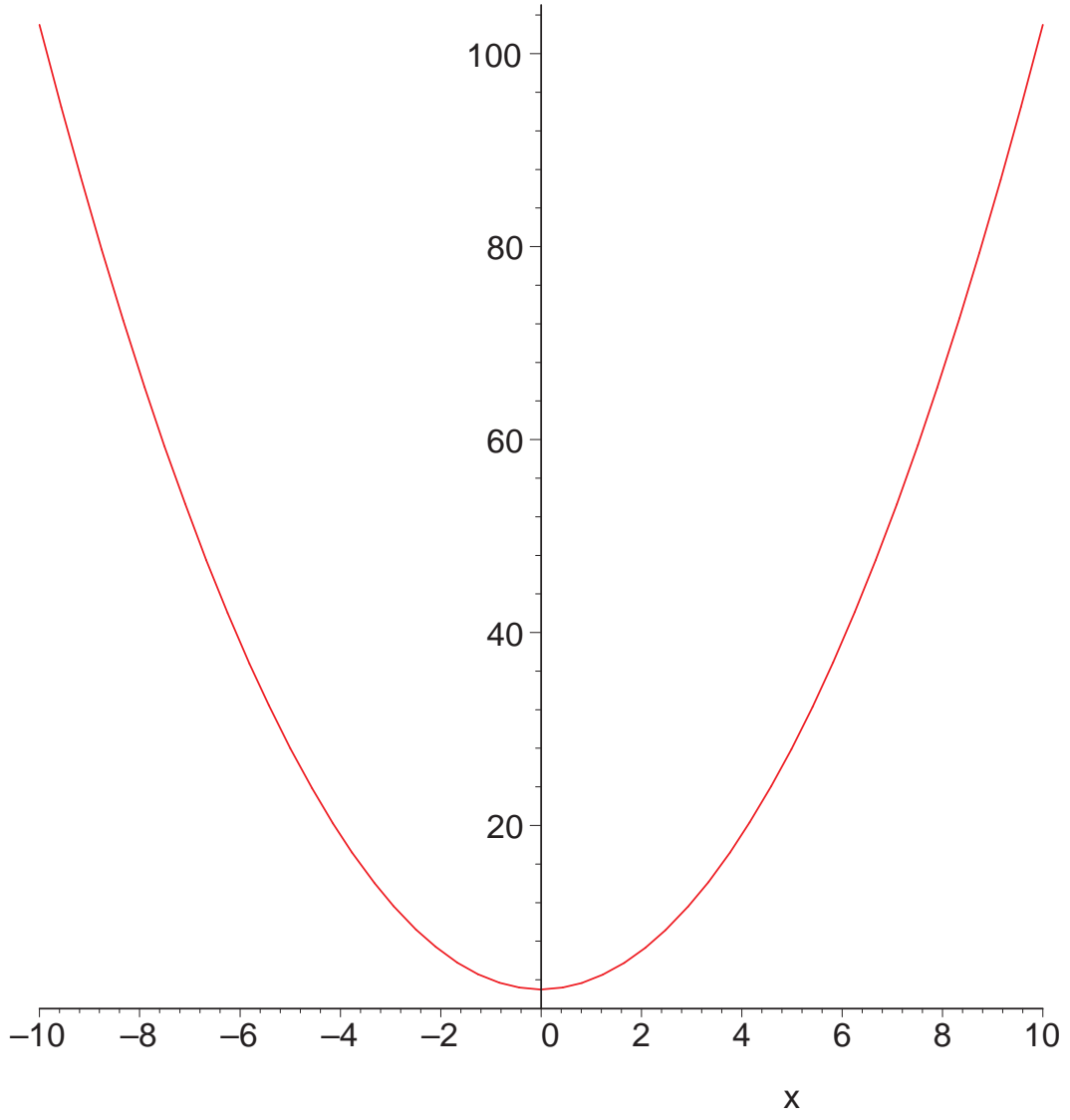
```

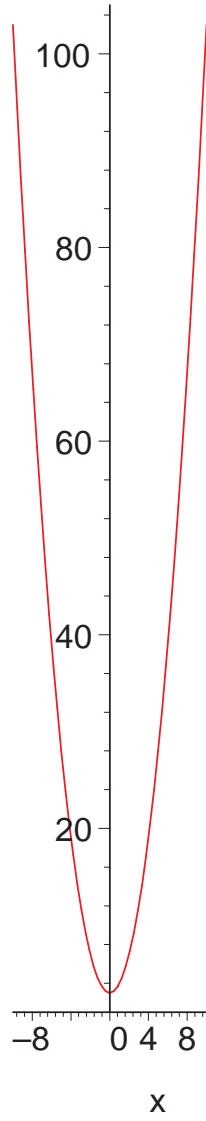
> # Funktionen:
  # Definition einer Funktion:
  f := proc(x) x^2+3 end proc:
  f := x -> x^2+3;

                                     f:=x→x2+3
> f(3);

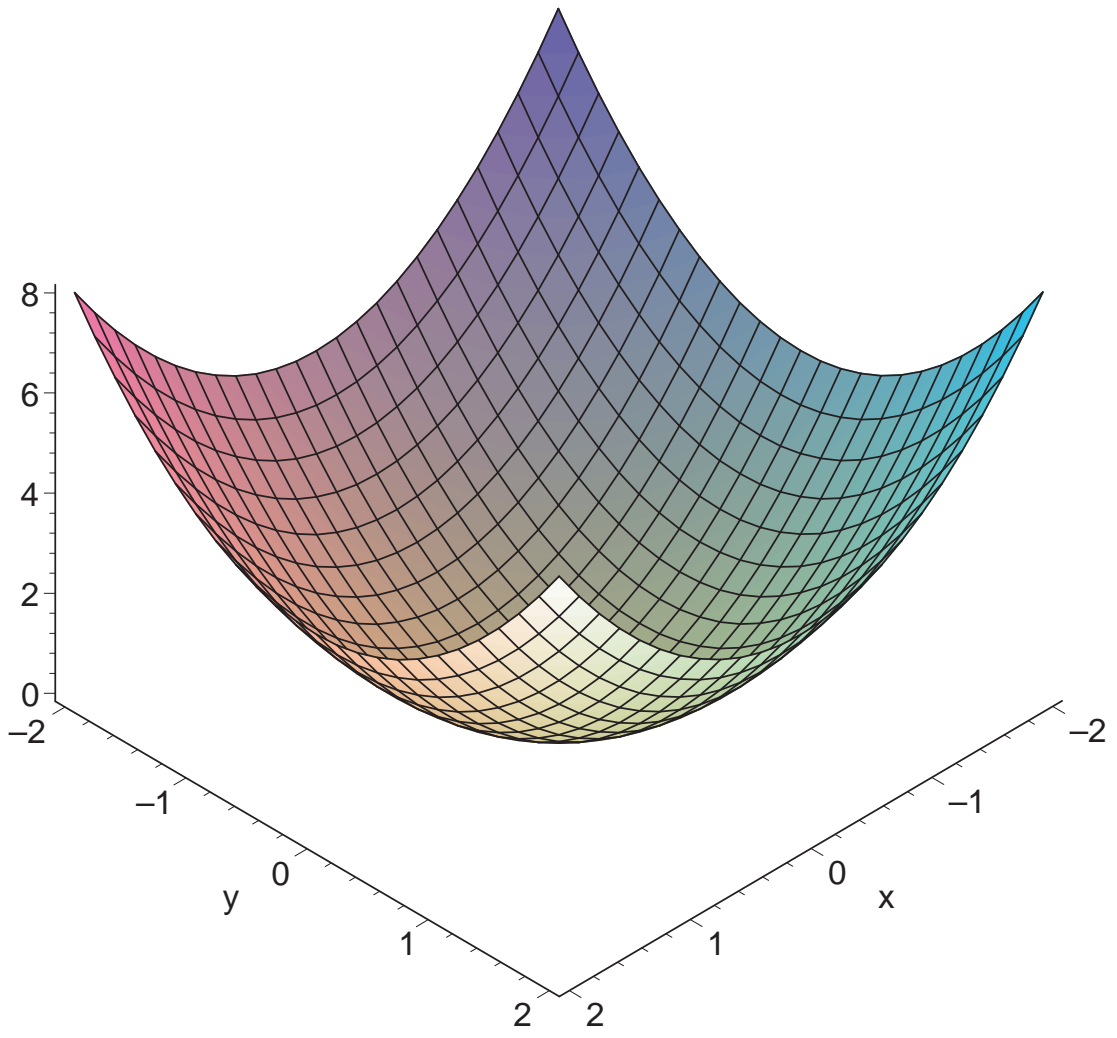
                                     12
> plot(f(x));
  plot(f(x), scaling = CONSTRAINED);

```





```
> plot3d(x^2+y^2,x=-2..2,y=-2..2, axes=FRAME);
```



[>