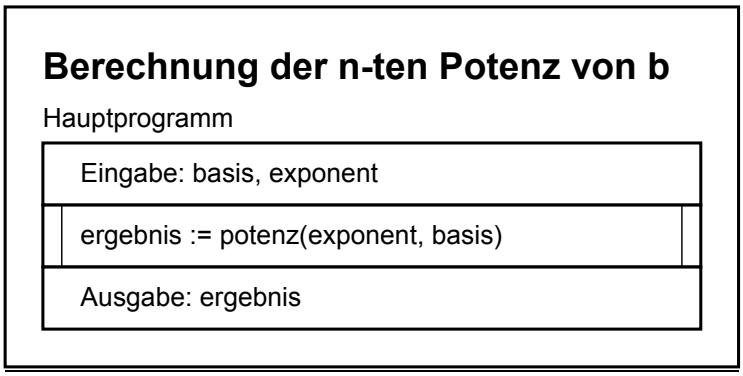
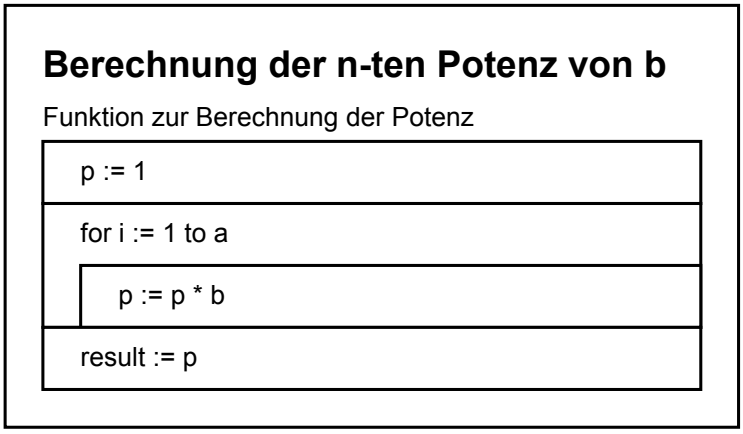
Aufgaben zu Funktionen

Aufgabe 1:
Ein Algorithmus ist zu entwickeln, welcher die n-te Potenz b^n (mit $n \geq 2$, $n \in N$) einer gegebenen natürlichen Zahl b berechnet.

Struktogramm:



Die Berechnung der Potenz erfolgt in einer eigenständigen Funktion.

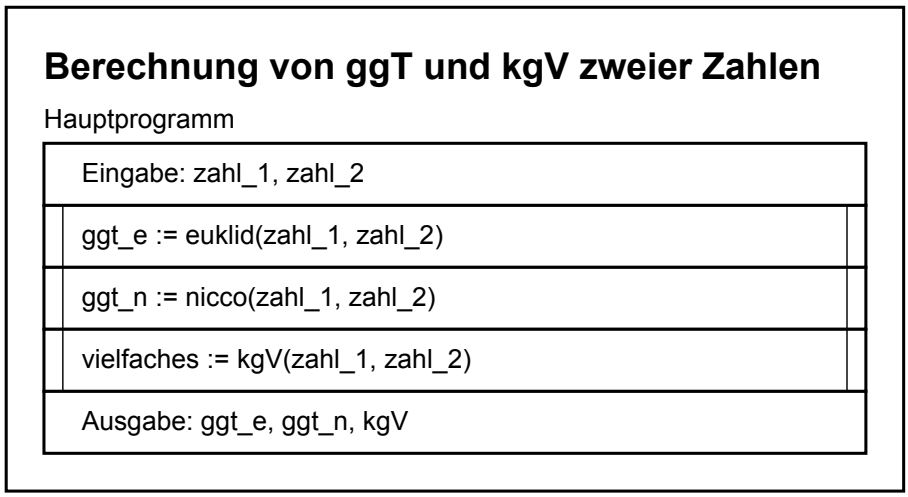


Quelltext:

```
function potenz(a, b: integer): integer;
var p, i: integer;
begin
  p := 1;
  for i := 1 to a do p := p * b;
  result := p;
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  basis, exponent, ergebnis: integer;
begin
  basis := StrToInt(Edit1.Text);
  exponent := StrToInt(Edit2.Text);
  ergebnis := potenz(exponent,basis);
  Edit3.Text := IntToStr(ergebnis);
end;
```

Aufgabe 2:
Entwickeln Sie einen Algorithmus zur Berechnung des ggT zweier Zahlen nach EUKLID.

Struktogramm: (Hauptprogramm)



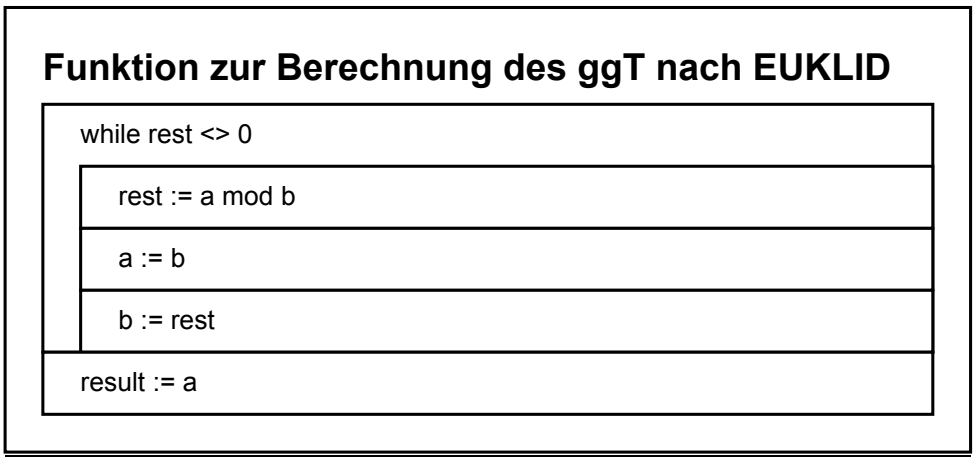
Quelltext: (Hauptprogramm)

```
procedure TForm1.Button1Click(Sender: TObject);
var
  zahl_1, zahl_2, ggt_e, ggt_n, vielfaches: integer;
begin
  zahl_1 := StrToInt(Edit1.Text);
  zahl_2 := StrToInt(Edit2.Text);

  ggt_e := euklid(zahl_1, zahl_2);
//Berechnung des ggT nach EUKLID
  ggt_n := nicco(zahl_1,zahl_2);
//Berechnung des ggT nach NICCOMACHUS
  vielfaches := kgv(zahl_1,zahl_2);
//Berechnung des kgV

  Edit3.Text := IntToStr(ggt_e);
  Edit4.Text := IntToStr(ggt_n);
  Edit5.Text := IntToStr(vielfaches);
end;
```

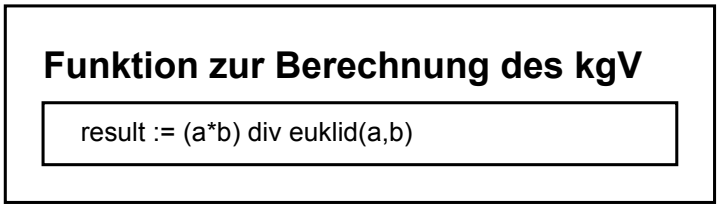
Die Berechnung des ggT nach EUKLID erfolgt in einer eigenständigen Funktion.




```
function euklid(a,b: integer): integer;
var rest: integer;
begin
  while rest <> 0 do begin
    rest := a mod b;
    a := b;
    b := rest;
  end;
  result := a;
end;
```

Zusatzaufgabe 1:

Entwickeln Sie einen Algorithmus zur Berechnung des kgV zweier Zahlen

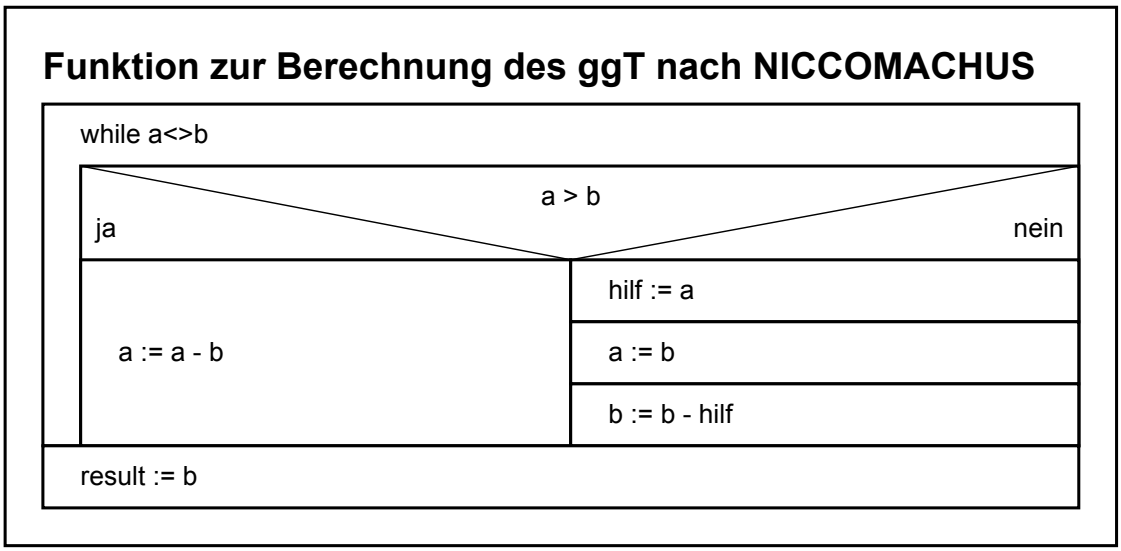


Achtung! Wenn Sie den kgV so berechnen wollen, muss die Funktion `euklid` **vorher** deklariert sein, da sie in der Funktion `kgv` verwendet wird.

```
function kgV(a,b: integer): integer;
begin
  result := ((a*b) div euklid(a,b));
end;
```

Zusatzaufgabe 2:

Entwickeln Sie einen Algorithmus zur Berechnung des ggT zweier Zahlen nach NICCOMA-CHUS.




```
function nicco(a,b: integer): integer;
var
  hilf: integer;
begin
  while a <> b do begin
    if a > b then a := a - b
    else begin
      hilf := a;
      a := b;
      b := b - hilf;
    end;
  end;
  result := b;
end;
```