SQL & VBA in MS Access

SQL – Structured Query Language

Alle nicht vorhandenen Artikel unter 6€ :

```
----------------------------------------
SELECT [anr], [ware], [land], [menge], [preis]
FROM ARTIKEL
WHERE preis<=6 And Not vorhanden;
```

Teller mit Lieferfristen :

```
-------------------------
SELECT [ware], liefer-bestell AS Frist
FROM ARTIKEL
WHERE ware = 'Teller';
```

Vorhandene Teller & Tassen mit Lieferfristen:

```
---------------------------------------------
SELECT ARTIKEL.WARE, liefer-bestell AS Frist, vorhanden
FROM ARTIKEL
WHERE (ARTIKEL.WARE="Teller" OR ARTIKEL.WARE="Tasse") AND vorhanden;
```

Alle Artikel, welche vor dem 30.02.2003 geliefert wurden:

```
-----------------------------------------------------
SELECT ARTIKEL.WARE, liefer, liefer-bestell AS Frist, vorhanden
FROM ARTIKEL
WHERE ( ARTIKEL.liefer<#30,03,2003#);
```

Vertreter in Brandenburg:

```
-------------------------
SELECT VNR, Region
FROM VERKAUF
WHERE region='Brandenburg';
```

Anzahl von Artikeln in England:

```
------------------------------
SELECT COUNT(WARE) AS Sortiment
FROM Artikel
WHERE land="England" ;
```

Anzahl der Kategorien:

```
---------------------
SELECT COUNT(Kat) AS Anzahl_Kategorien
FROM Zuschlag;
```

Artikelmenge, kleinsten Preis, maximale Lieferfrist, durchschnittl. Lieferfrist :

```
-----------------------------------------------------
SELECT Sum([menge]) AS Summe, Min([Preis]) AS Minimalpreis,
Max([liefer]-[bestell]) AS [Maximale Lieferfrist],
Avg([liefer]-[bestell]) AS [Mittlere Lieferfrist]
FROM artikel;
```

Anzahl der Artikel, gruppiert nach Waren:

```
----------------------------------------
SELECT Ware, COUNT(ware) as Anzahl
FROM ARTIKEL
group by ware;
```

Anzahl der Waren, gruppiert nach Region:

```
----------------------------------------
SELECT Region, Count(anr) as Warenmenge
FROM verkauf
group by region;
```

Anzahl der Waren, gruppiert nach Land:

```
-------------------------------------
SELECT land, Count(ware) as Warenmenge
FROM artikel
group by land;
```

Anzahl der Positionen, gruppiert 1.nach Ort & 2.nach Region:

```
-------------------------------------------------------------
SELECT Count(*), Region , Ort
FROM verkauf
group by ort, region;
```

Gruppierung bzw. Sortierung nach Ware-Land-Preis:

```
-----------------------------------------------
SELECT Ware, Land, Preis
FROM artikel
GROUP BY ware, land, preis;
```

Länder, in denen die Menge über 3000 liegt:

```
-----------------------------------------
SELECT  Land
FROM artikel
GROUP BY land,menge
HAVING menge>=3000 ;
```

Länder, in denen die Menge über 3000 liegt (gruppiert):

```
-----------------------------------------------------------
SELECT  Land
FROM artikel
WHERE menge>=3000
GROUP BY land;
```

Länder, deren aufaddierte Menge 8000 übersteigt:

```
-------------------------------------------------
SELECT Land, SUM(menge) AS [Summierte Menge]
FROM artikel
GROUP BY land
HAVING SUM(menge)>=8000;
```

Regionen mit mindestens 2 Artikeln (gruppiert):

```
-------------------------------------------------
SELECT Region, Count(anr) AS Sortiment
FROM verkauf
GROUP BY region
HAVING COUNT(ANR)>=2;
```

siehe oben, aber innerhalb von Hessen & Brandenburg:

```
-----------------------------------------------------
SELECT [Region], Count([anr]) AS Sortiment
FROM verkauf
WHERE region IN("Hessen","Brandenburg")
GROUP BY [region]
HAVING COUNT(ANR)>=2;
```

Regionen und Vertreternamen, gruppiert und abwärts geordnet:

```
-------------------------------------------------------------
SELECT Region,  VName, Count(*) AS Positionen
FROM verkauf
GROUP BY Region, VName
ORDER BY Region DESC, VName DESC;
```

Warenerlöse und durchschnittl. Lieferfristen aller Artikel:

```
-------------------------------------------------------------
SELECT Ware AS Sortiment, sum(preis*menge) AS [Umsatzsumme in €],
AVG(liefer-bestell) AS [Durchschnitt Lieferfrist in Tagen]
FROM artikel
GROUP BY Ware
HAVING sum(preis*menge)>10000;
```

Gruppierung nach Regionen mit mehr als einem Artikel, rückwärts geordnet:

```
-----------------------------------------------------------------------
SELECT [REGION], Count([ANR]) AS [Anzahl von ANR]
FROM verkauf
GROUP BY [REGION]
HAVING Count(ANR)>1
ORDER BY [REGION] DESC;
```

Hier macht MS-Access einen Fehler (wegen 'distinct'):

```
-----------------------------------------------------
SELECT REGION, Count(distinct ANR) AS [Anzahl von ANR]
FROM verkauf
GROUP BY REGION
HAVING Count(ANR)>1
ORDER BY REGION DESC;
```

>UNTERABFRAGEN<

Artikel mit tiefstem Preis:

```
--------------------------
SELECT ANR AS [Artikel-Nr], Ware AS [Artikel mit tiefsten Preis], Preis AS [Preis in €], Land
```

```
FROM ARTIKEL
WHERE (SELECT MIN (preis) FROM artikel) = preis;
```

Land mit der kürzesten Lieferfrist:

```
----------------------------------
SELECT  Land, liefer-bestell AS Lieferfrist
FROM ARTIKEL
WHERE (liefer-bestell) = (SELECT MIN (liefer-bestell) FROM artikel);
```

Länder mit größere Mengen eines Artikels als Argentinien:

```
-----------------------------------------------------------
SELECT Land, sum(menge) AS Gesamtmenge
FROM ARTIKEL
GROUP BY land
HAVING sum(menge) > (SELECT sum(menge) FROM Artikel WHERE land = "Argentinien");
```

Welche Vertreter verteiben welche Waren:

```
SELECT ware, vname
FROM artikel, verkauf;

SELECT *
FROM artikel, verkauf;

SELECT ware, vname
FROM artikel, verkauf
WHERE artikel.anr=verkauf.anr;
```

Welche vertreter mit Waren ab 2500 Stück aus England und China:

```
SELECT vname, ware, menge, land
FROM artikel, verkauf
WHERE artikel.land IN("England","China")
AND artikel.menge>=2500
AND artikel.anr=verkauf.anr;
--------------------------------
SELECT vname, ware, menge, land
FROM artikel, verkauf
WHERE artikel.land IN("England","China")
AND artikel.anr=verkauf.anr
GROUP BY vname,  ware, artikel.menge, land
HAVING artikel.menge>=2500;
```

Verterter mit Waren mit einem Preis unter 8€:

- aufsteigend sortiert nach Waren
- absteigend sortiert nach PReisen
- gruppiert nach Waren

```
-------------------------------------------
SELECT  ware, vname, preis
FROM artikel, verkauf
WHERE artikel.anr=verkauf.anr
GROUP BY ware, vname, artikel.preis
HAVING artikel.preis<8.00
ORDER BY ware,  preis DESC;
```

Warenmengen in Regionen ab 5000:

```
--------------------------------
SELECT region, sum(menge) AS Summenmenge
FROM artikel, verkauf
WHERE artikel.anr=verkauf.anr
GROUP BY  region
HAVING SUM(menge)>=5000;
```

Warenpositionen pro Vertreter

- nur Vertreter mit mehr als einer Position
- Mengen ab 8000

```
----------------------------------------
SELECT vname, COUNT(*) AS Warenpositionen, SUM(menge) AS Mengensumme
FROM artikel, verkauf
WHERE artikel.anr=verkauf.anr
GROUP BY vname
HAVING SUM(menge)>=8000
AND COUNT(*)>1;
```

In welchen Vereinen sind die Vertreter:

```
SELECT vname, verein
FROM verkauf, mitglied
WHERE verkauf.vnr=mitglied.vnr
GROUP BY vname, verein;
```

Vertreter mit Funktionen:

```
SELECT vname,funktion
FROM verkauf, mitglied
WHERE verkauf.vnr=mitglied.vnr
GROUP BY vname, funktion
HAVING funktion<>"";
--------------------------------
SELECT vname,funktion
FROM verkauf, mitglied
WHERE verkauf.vnr=mitglied.vnr
AND  funktion
GROUP BY vname, funktion;
```

Vertreter ohne Funktionen:

```
---------------------------
SELECT vname,funktion
FROM verkauf, mitglied
WHERE verkauf.vnr=mitglied.vnr
AND  funktion IS NULL
GROUP BY vname, funktion;
```

Umsätze der Vertreter:

```
----------------------
SELECT vname, SUM(menge*preis) AS Umsatz
FROM verkauf,artikel
WHERE artikel.anr=verkauf.anr
GROUP BY vname;
```

Vertreter mit Umsatz ab 20000:

```
SELECT vname, SUM(menge*preis) AS Umsatz
FROM verkauf,artikel
WHERE artikel.anr=verkauf.anr
GROUP BY vname
HAVING  SUM(menge*preis)>=20000;
```

Dynamische Abfrage

```
SELECT vname, SUM(menge*preis) AS Umsatz
FROM verkauf,artikel
WHERE artikel.anr=verkauf.anr AND vname=Eingabe_Vertretername
GROUP BY vname;
```

Vertreter mit höchstem Einzelumsatz:

```
SELECT vname, (menge*preis) AS Einzelumsatz
FROM verkauf,artikel
WHERE artikel.anr=verkauf.anr
GROUP BY vname, (menge*preis)
HAVING (menge*preis)  = (SELECT MAX(menge*preis) FROM artikel);
---------------------------------
SELECT vname, (menge*preis) AS Einzelumsatz
FROM verkauf,artikel
WHERE artikel.anr=verkauf.anr
AND  (menge*preis)  = (SELECT  MAX(menge*preis) FROM ARTIKEL);
```

Vertreter mit höchstem Umsatz:

```
------------------------------
SELECT vname, SUM(menge*preis) AS Einzelumsatz
FROM verkauf,artikel
WHERE artikel.anr=verkauf.anr
GROUP BY vname
HAVING SUM(menge*preis)  >= ALL (SELECT  SUM(menge*preis)
FROM artikel,verkauf
WHERE artikel.anr=verkauf.anr
GROUP BY vname);
```

Welche Vertreter können nicht liefern, weil Ware nicht vorhanden:

```
-------------------------------------
SELECT vname, ware
FROM verkauf, artikel
WHERE artikel.anr=verkauf.anr
AND NOT vorhanden;
```

Welche Kategorien haben die Vertreter:

```
---------------------------------------
SELECT vname,kat
FROM vertreter, verkauf
WHERE vertreter.vnr = verkauf.vnr
GROUP BY vname, kat;
```

Bei welchen Vertretern sind Arbeits- & Wohnort identisch:

```
-----------------------------------------------------------
```

```
SELECT vname,ort,vort
FROM vertreter, verkauf
WHERE vertreter.vnr = verkauf.vnr
AND vort=ort
GROUP BY vname, ort, vort;
```

Welcher Vertreter liefert am schnellsten:

```
----------------------------------------
SELECT vname, ware, (liefer-bestell) AS Lieferfrist
FROM artikel, verkauf
WHERE artikel.anr = verkauf.anr
GROUP BY vname, ware,  (liefer-bestell)
HAVING  (liefer-bestell) = (SELECT MIN(liefer-bestell) FROM artikel);
```

siehe oben - Lösungs-Alternative:

```
---------------------------------
SELECT vname, ware, (liefer-bestell) AS Lieferfrist
FROM artikel, verkauf
WHERE artikel.anr = verkauf.anr
AND  (liefer-bestell) = (SELECT MIN(liefer-bestell) FROM artikel) ;
```

Welche Zuschläge haben die Vertreter:

```
-------------------------------------
SELECT [vname], [zuschlag], zuschlag.kat
FROM verkauf, zuschlag, vertreter
WHERE [verkauf].[vnr]=[vertreter].[vnr]
And [vertreter].[kat]=[zuschlag].[kat]
GROUP BY [vname], [zuschlag], zuschlag.kat;
```

Welche Zuschläge haben die Vertreter in einem Verein:

```
-----------------------------------------------------
SELECT DISTINCT vname, zuschlag, zuschlag.kat, verein
FROM verkauf, zuschlag, vertreter, mitglied
WHERE verkauf.vnr=vertreter.vnr
And vertreter.kat=zuschlag.kat
AND verkauf.vnr=mitglied.vnr;
```

Beziehungen erstellen !!!

View

View v1 aus Tabelle Artikel mit den Spalten Ware, Menge, Liefer, Land,

wobei Lieferermine vor dem 30.03.2002 liegen:

```
-----------------------------------------------------------------------------------------
SELECT ware, menge, liefer, land
INTO v1
FROM artikel
WHERE liefer < #30,03,2003#;
```

Kopie von Artikel

```
-------------------------
Select * INTO v2 FROM artikel;
```

View Umsatz mit den Vertreternamen und den Umsätzen der Vertreter:

```
-----------------------------------------------------------------------------------
SELECT vname, SUM(menge*preis) AS Umsatz
INTO Umsatz
```

```
FROM verkauf,artikel
WHERE artikel.anr=verkauf.anr
GROUP BY vname;
```

aus Tabelle Umsatz die Vertreter mit Umsatz ab 20000:

```
SELECT vname, Umsatz
FROM Umsatz
WHERE Umsatz >= 20000 ;
```

Alias-Namen

Waren, Vertreter und Vereine - für die in Frage kommenden Tabellen Alias-Namen:

```
SELECT ware, vname, verein
FROM artikel a, verkauf v, mitglied m
WHERE a.anr = v.anr AND m.vnr = v.vnr;
```

Theta-Join

```
SELECT vname, ware,  artikel.anr,  verkauf.vnr
FROM artikel, verkauf
WHERE artikel.anr > verkauf.vnr;
```

Auto-Join (mit Theta-Join)

Welche Liefertermine für Tassen liegen vor den Lieferterminen von Tassen:

```
SELECT tasse.ware,tasse.liefer, teller.ware, teller.liefer
FROM artikel tasse, artikel teller
WHERE tasse.ware ="Tasse" AND teller.ware="Teller"
AND tasse.liefer < teller.liefer;
```

Welche englischen Waren sind preiswerter als die chinesischen Waren:

```
SELECT  englische.ware, englische.preis,chinesische.ware, chinesische.preis
FROM artikel chinesische, artikel englische
WHERE chinesische.land="China"
And englische.land="England"
And englische.preis <chinesische.preis;
```

Vertreter, Wohnort (vort), Vereine & Funktionen im Verein:

```
SELECT Distinct vname, vertreter.vort, verein, funktion
FROM verkauf, mitglied, vertreter
WHERE verkauf.vnr=mitglied.vnr
 AND verkauf.vnr=vertreter.vnr;
```

OUTER JOIN

```
SELECT vname, verein, funktion
FROM mitglied RIGHT JOIN verkauf ON verkauf.vnr=mitglied.vnr;

SELECT vname, verein, funktion
FROM verkauf LEFT JOIN mitglied ON verkauf.vnr=mitglied.vnr;
```

Alle Mitglieder, die nicht in einem Verein sind:

```
SELECT Distinct vname AS Vertreter_ohne_Club
FROM verkauf
WHERE vnr NOT IN (SELECT vnr FROM mitglied);
```

UNION

Zur Vorbereitung zwei Tabellen erzeugen …

- View Club1 mit allen Vertretern, die im Skatclub sind:

```
SELECT DISTINCT vname, verein
INTO Club1
FROM verkauf, mitglied
WHERE verkauf.vnr = mitglied.vnr
AND verein="Skatclub";
```

- View Club2 mit allen Vertretern, die im Kaninchenzüchterverein sind:

```
SELECT DISTINCT vname, verein
INTO Club2
FROM verkauf, mitglied
WHERE verkauf.vnr = mitglied.vnr
AND verein="Kaninchenzuechter";
```

Eigentliche UNION:

```
SELECT * FROM Club1
UNION
SELECT * FROM Club2;
```

Kombi-Aufgabe:

1) Vertreter mit Region und Wohnort (Vort) aus Bayern und Hessen in View Süd
2) Vertreter mit Region und Wohnort (Vort) aus Brandenburg und Sachsen in View Nord
3) UNION mit Bayern und Sachsen

---

1)

```
SELECT vname
INTO Süd
FROM vertreter, verkauf
WHERE vertreter.vnr = verkauf.vnr
AND (verkauf.region="Bayern" OR verkauf.region="Hessen");
```

2)

```
SELECT vname, region, vort
INTO Nord
FROM vertreter, verkauf
WHERE vertreter.vnr = verkauf.vnr
AND region IN ("Brandenburg","Sachsen");
```

3)

```
SELECT* FROM Süd Where region="Bayern"
UNION
SELECT* FROM Nord Where region="Sachsen";
```

Vereinigung Orte und Wohnorte (vort) der Vertreter:

```
SELECT ort FROM verkauf
UNION
SELECT vort FROM vertreter;
```

Komplexe Aufgabe - Serienbrief (MULIT-JOIN)

```
SELECT DISTINCT vname,ort, vort, artikel.anr, ware, land, liefer, region,zuschlag.kat  INTO Serie
FROM artikel, verkauf, zuschlag, vertreter
WHERE verkauf.vnr=vertreter.vnr AND vertreter.kat=zuschlag.kat AND verkauf.anr = artikel.anr
ORDER BY vname;
```

SERIENBRIEF:

Frau/Herrn
«vname»
«vort»

Filiale: «ort»

Sehr geehrte/r Frau/Herr «vname»,
wir freuen uns, ihnen mitteilen zu können, dass Sie in «region» mit Arbeitsort «ort» für das Sortiment «ware» mit der ANR «anr» aus «land» verantwortlich sind. Der Liefertermin ist «liefer». Ihrer Zuschlagskategorie ist «kat».

Zu Ihrer Information sehen Sie nachstehend die Gesamtumsätze aller Vertreter:

##

Umsätze der Vertreter

```
SELECT  verkauf.vname,SUM(preis*menge)  INTO Vertreter_Serie
FROM verkauf, artikel
WHERE artikel.anr=verkauf.anr
GROUP BY vname;
```

##

VBA - Visual Basic for Applications

==========

LinearProgramm

```
Dim x, y, z As Single
x = CSng(InputBox("Anfangssparbetrag eingeben: "))
y = CSng(InputBox("Zinswert in % eingeben: "))
z = x + (x * y / 100)
z = Format(z, "#,##0.00")
Debug.Print "Der Endwert beträgt: "; z; "€"
```

Alternative-einfach

```
Dim x As Single
x = InputBox("Temperaturen eingeben")
If x < o Then
   Debug.Print "Frost bei "; x; "Grad"
Else
   Debug.Print "Frostfrei bei "; x; "Grad"
End If
```

Alternative-mehrfach

```
--------------------
Dim x As Single
x = InputBox("Temperatur eingeben:")
If x < -10 Then
   Debug.Print "Strenger Frost bei "; x; "Grad."
ElseIf x < 0 Then
   Debug.Print "Frost bei "; x; "Grad."
ElseIf x < 10 Then
   Debug.Print "Kalt bei"; x; "Grad."
ElseIf x < 20 Then
   Debug.Print "Mild bei"; x; "Grad."
ElseIf x < 30 Then
   Debug.Print "Warm bei"; x; "Grad."
Else
   Debug.Print "Heiß bei"; x; "Grad."
End If
```

Bool, Elself(NOT)

```
-----------------
Dim x As Single, y As Boolean
y = InputBox("Schnee TRUE/FALSE eingeben:")
x = InputBox("Temperatur eingeben:")
If (y And x <= 0) Then
   Debug.Print "Schnee und Frost bei "; x; "Grad."
   Debug.Print "Schnee fegen und Streuen!"
ElseIf (y And x > 0) Then
   Debug.Print "Kein Frost, aber Schnee bei"; x; "Grad."
   Debug.Print "Schneematsch beseitigen!"
ElseIf (Not y) Then
   Debug.Print "Weiterschlafen!"
End If
```

Maximumzahlen

```
-------------
If (x > y And x > z) Then
   Debug.Print "Das Maximum der Zahlen ist x --> "; x
ElseIf (y > z And y > x) Then
   Debug.Print "Das Maximum der Zahlen ist y --> "; y
ElseIf (z > y And z > x) Then
   Debug.Print "Das Maximum der Zahlen ist z --> "; z

ElseIf (x > y And x = z) Then
   Debug.Print "Das Maximumzahlen sind x & z --> "; z
ElseIf (y > x And y = z) Then
   Debug.Print "Das Maximumzahlen sind y & z --> "; z
ElseIf (y > z And y = x) Then
   Debug.Print "Das Maximumzahlen sind x & y --> "; x

ElseIf (y = z And y = x) Then
   Debug.Print "Alle sind gleich --> "; x
End If
```

Select-Case

```
-----------
Dim x As String
x = InputBox("Zeichen von a bis e eingeben")
Select Case x
Case Is = "a"
```

```
   Debug.Print "Zeichen: a"
Case Is = "b"
   Debug.Print "Zeichen: b"
Case Is = "c"
   Debug.Print "Zeichen: c"
Case Is = "d"
   Debug.Print "Zeichen: d"
Case Is = "e"
   Debug.Print "Zeichen: e"
Case Else
   Debug.Print "Zeichen nicht vordefiniert!"
End Select
```

Mahnwesen zu Select-Case

```
------------------------
Dim x As Integer
x = InputBox("Tage ab Eingang der Zahlunsaufforderung:")
Debug.Print "Das war Ihre Eingabe -->"; x
Select Case x
Case Is <= 15
   Debug.Print "Alles im grünen Bereich."
Case Is <= 30
   Debug.Print "1.Mahnung, weil"; x - 15; "Tage im Verzug."
Case Is <= 45
   Debug.Print "2.Mahnung, weil"; x - 15; "Tage im Verzug."
Case Is <= 60
   Debug.Print "3.Mahnung, weil"; x - 15; "Tage im Verzug."
Case Else
   Debug.Print "Rechtsabteilung,weil "; x - 15; "Tage im Verzug."
End Select
```

Schleifen

```
---------
Dim x, start, ende, schritt As Integer
start = InputBox("Startwert:")
ende = InputBox("Endwert:")
schritt = InputBox("Schritt:")
For x = start To ende Step schritt
 Debug.Print x
 Next x
Debug.Print "Habe von "; start; " bis "; ende; " gezählt! Schrittweite war"; schritt; "!"
```

Schleifen-Rekursion

```
-------------------
Dim x, start, ende, summe As Integer
start = 1
ende = 100
summe = 0
For x = start To ende
 Debug.Print "Durchlauf:"; x; "--> Summe ="; summe; "+"; x; "="; (summe + x)
 summe = summe + x
 Next x
Debug.Print "FERTIG!"
```

Schleifen mit Abbruchbedingung

```
-----------------------------
Dim x, start, ende, summe As Integer
start = 1
ende = 100
```

```
summe = 0
For x = start To ende
 Debug.Print "Durchlauf:"; x; "--> Summe ="; summe; "+"; x; "="; (summe + x)
 summe = summe + x
 If summe >= 1000 Then Exit For
 Next x
Debug.Print "FERTIG!"
```

Monatsumsätze mit Abbruchbedingung

```
Dim x, monate, umsatz, zielwert, summe As Single
monate = 12
summe = 0
Debug.Print "PROTOKOLL"
zielwert = InputBox("Zielwert eingeben:")
For x = 1 To monate
 umsatz = InputBox("Umsatz:")
 Debug.Print "Umsatz in Monat"; x; " : "; umsatz
 summe = summe + umsatz
 If summe >= zielwert Then Exit For
 Next x
Debug.Print "Gesamtumsatz:"; summe; ">= Zielwert: "; zielwert
```

WHILE-Schleife

```
Dim x, y As Integer
x = 0
y = InputBox("Endwert:")
Debug.Print
Debug.Print ("START!")
While x < y
   x = x + 1
   Debug.Print x
Wend
Debug.Print ("FERTIG!")
```

DO-WHILE-Schleife

```
Dim x, y As Integer
x = 0
y = InputBox("Endwert:")
Debug.Print
Debug.Print ("PROGRAMMSTART!")
Do While x < y
   x = x + 1
   Debug.Print x
Loop
Debug.Print ("PROGRAMMENDE!")
```

DO-LOOP-WHILE-Schleife

```
Dim x, y As Integer
x = 0
y = InputBox("Abbruch bei:")
```

```
Debug.Print
Debug.Print ("PROGRAMMSTART!")
Do
   x = x + 1
   Debug.Print x
Loop While x < y
Debug.Print ("PROGRAMMENDE!")
```

DO-UNTIL-Schleife

```
Dim x, y As Integer
x = 0
y = InputBox("Abbruch bei:")
Debug.Print
Debug.Print ("PROGRAMMSTART!")
Do Until x < y
   x = x + 1
   Debug.Print x
Loop
Debug.Print ("PROGRAMMENDE!")
```

DO-LOOP-UNTIL-Schleife

```
Dim x, y As Integer
x = 0
y = InputBox("Abbruch bei:")
Debug.Print
Debug.Print ("PROGRAMMSTART!")
Do
   x = x + 1
   Debug.Print x
Loop Until x < y
Debug.Print ("PROGRAMMENDE!")
```

Feld (ARRAY)

```
Debug.Print "PROGRAMMSTART!"

For i = 1 To 100
   Liste(i) = i
   Next i
For i = 1 To 100
   Debug.Print Liste(i)
   Next i

x = InputBox("Welches Feld?")

Debug.Print "Feldinhalt: "; Liste(x)

Debug.Print "PROGRAMMENDE!"
```

Preiserhöhung

```
Debug.Print "P-START!"

Dim Preis_Alt(1 To 4) As Single
```

```
Preis_Alt(1) = 2.87
Preis_Alt(2) = 6.89
Preis_Alt(3) = 7.56
Preis_Alt(4) = 3.78

Dim Preis_Neu(1 To 4) As Single
Dim prozent As Single
Dim i As Integer
prozent = InputBox("Preiserhöhung in %:")

For i = 1 To 4
   Debug.Print
   Debug.Print "Altpreis Artikel"; i; ": "; Preis_Alt(i)
   Preis_Neu(i) = Preis_Alt(i) * prozent / 100 + Preis_Alt(i)
   Debug.Print "Neupreis Artikel"; i; "nach"; prozent; "% Erhöhung:"; Preis_Alt(i)
   Next i

Debug.Print
Debug.Print "P-ENDE!"
```

Wochentage

```
----------
Debug.Print "P-START!"

Dim weekdays(1 To 7) As String
Dim i As Integer

For i = 1 To 7
   weekdays(i) = InputBox("Eingabe Tag")
   Next i

For i = 1 To 7
   Debug.Print "Tag"; i; ": "; weekdays(i)
   Next i

Debug.Print "P-ENDE!"
```

ToDo-Liste

```
----------
Debug.Print "P-START!"

Dim Liste(1 To 10) As String
Dim i As Integer
Dim aus As Boolean
i = 0

For i = 1 To 10
   Liste(i) = InputBox("Eintrag:")
   aus = InputBox("Liste Fertig? TRUE/FALSE :")
   If aus Then Exit For
   Next i

For i = 1 To 10
   Debug.Print "Eintrag"; i; ": "; Liste(i)
   Next i

Debug.Print "P-ENDE!"
```

Call-by-Wert

```
--------------------------
Public Function test()
Debug.Print
Debug.Print "P-START!"
Dim y, x As Single
x = 10
y = 20
Debug.Print ("Hauptprogramm x: "); x
Debug.Print ("Hauptprogramm y: "); y
Call wert((x), (y))
Debug.Print ("Hauptprogramm x: "); x
Debug.Print ("Hauptprogramm y: "); y
Debug.Print "P-ENDE!"
End Function

Public Sub wert(x As Single, y As Single)
x = 0
y = 1
Debug.Print ("Nebenprogramm x: "); x
Debug.Print ("Nebenprogramm y: "); y
End Sub
```

Call-By-Referenz

```
----------------
Public Function UP_2()
Debug.Print
Debug.Print "P-START!"
Dim y As Single, z As Single, preis As Single
preis = InputBox("Brutto-Preis:")
Call Referenz(preis, y, z)
Debug.Print "Bruttopreis:"; preis
Debug.Print "MwSt.:"; y
Debug.Print "Nettopreis:"; z
Debug.Print "P-ENDE!"
End Function

Public Sub Referenz(Brutto As Single, MWst As Single, Netto As Single)
MWst = Brutto / 119 * 19
Netto = Brutto - MWst
End Sub
```

DB_Zugriff

```
----------
Debug.Print "P-START!"
Dim db As Database, rs As Recordset
Set db = CurrentDb
Set rs = db.OpenRecordset("SELECT vname, region, ort FROM verkauf WHERE vname = 'Keller';")
rs.MoveFirst
Do Until rs.EOF
   Debug.Print rs![vname]; ": "; rs![region]; ", "; rs![ort]
   rs.MoveNext
Loop
db.Close
Debug.Print "P-ENDE!"
```

DB_Editieren

```
------------
Debug.Print
```

```
Debug.Print "P-START!"
Dim db As Database, rs As Recordset
Set db = CurrentDb
Set rs = db.OpenRecordset("Artikel")
rs.MoveFirst
Do Until rs.EOF
   Debug.Print
   Debug.Print ("Alt:"); rs![menge]
   rs.Edit
   rs![menge] = rs![menge] * 2
   Debug.Print ("Neu:"); rs![menge]
   rs![menge] = rs![menge] / 2
   Debug.Print ("Rück:"); rs![menge]
   rs.MoveNext
Loop
db.Close
Debug.Print
Debug.Print "P-ENDE!"
```

DB_Suche

```
Debug.Print
Debug.Print "P-START!"
Dim db As Database, rs As Recordset
   Set db = CurrentDb
   Set rs = db.OpenRecordset("Verkauf", dbOpenDynaset)
   s = InputBox("Gesuchter Name")
   rs.FindFirst "[vname]= " + "'" + s + "'"
   Do Until rs.NoMatch
   Debug.Print rs![vname], rs![region], rs![ort]
   rs.FindNext "[vname]= " + "'" + "'"
   Loop
Debug.Print "P-ENDE!"
```