BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wirtschaftsinformatik 2
LE 01 – Wiederholung: Grundlagen von VBA und MS Access

Thomas Off

www.ThomasOff.de/lehre/beuth/wi2

Ziel

Ziel dieser Lehreinheit

- VBA für MS Access als Programmierumgebung innerhalb von MS Access kennenlernen
- Wenn Sie Kenntnisse anderer (prozeduraler) Programmiersprachen haben, sollen Sie diese auf VBA übertragen können
- Wenn Sie noch keine Kenntnisse in einer Programmiersprache haben, sollen Sie ausgewählte Konzepte von VBA kennenlernen und anwenden können
- nur die Konzepte vorgestellt, die in den weiteren Lehreinheiten notwendig sind

Inhalt

Einordnung

Einstieg in MS Access mit VBA

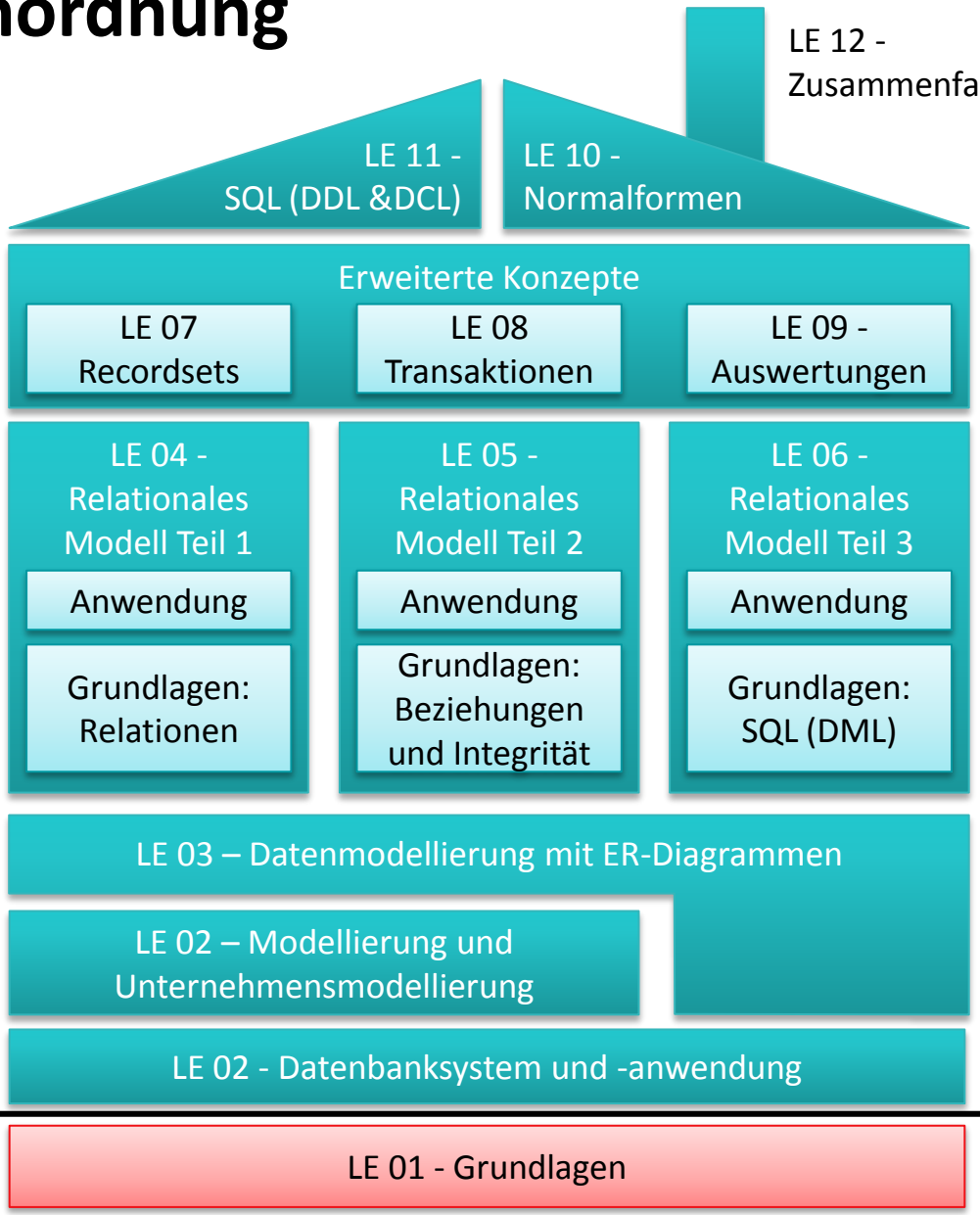
- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Einordnung


LE 12 - Zusammenfassung
LE 11 - SQL (DDL &DCL)
LE 10 - Normalformen
Erweiterte Konzepte
LE 07 Recordsets
LE 08 Transaktionen
LE 09 - Auswertungen
LE 04 - Relationales Modell Teil 1
Anwendung
Grundlagen: Relationen
LE 05 - Relationales Modell Teil 2
Anwendung
Grundlagen: Beziehungen und Integrität
LE 06 - Relationales Modell Teil 3
Anwendung
Grundlagen: SQL (DML)
LE 03 – Datenmodellierung mit ER-Diagrammen
LE 02 – Modellierung und Unternehmensmodellierung
LE 02 - Datenbanksystem und -anwendung
LE 01 - Grundlagen

Inhalt

Einordnung

Einstieg in MS Access mit VBA

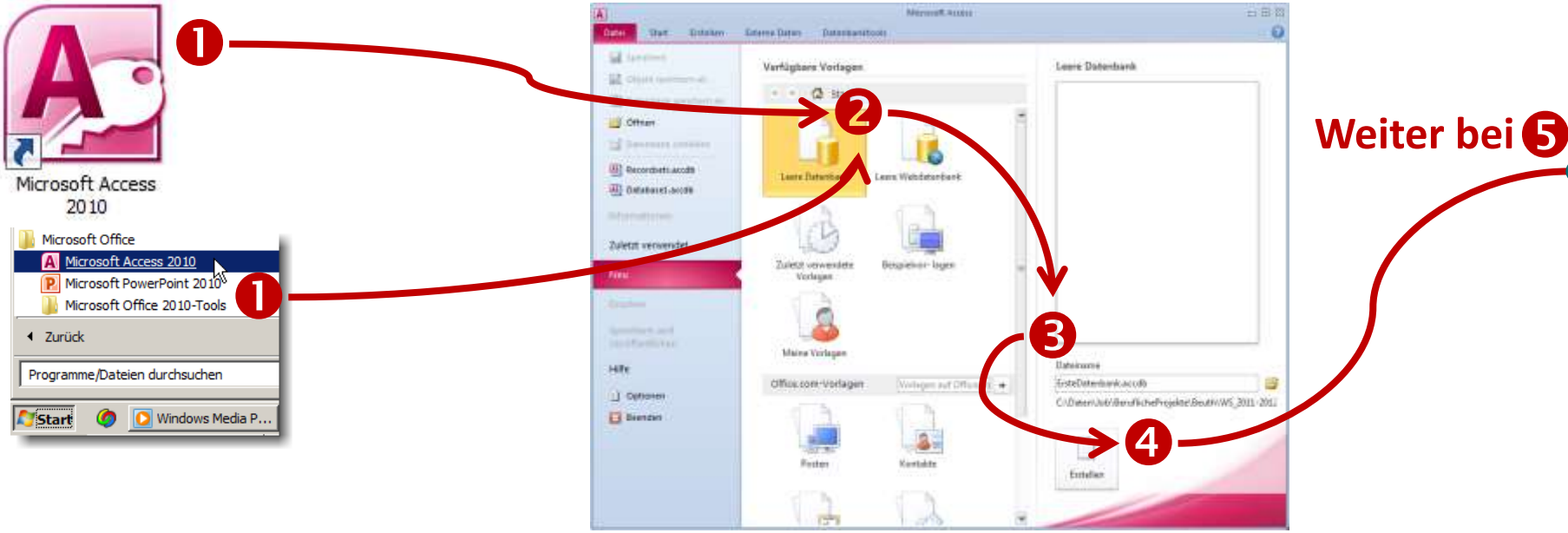
- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

MS Access mit VBA – Erste Schritte (1)

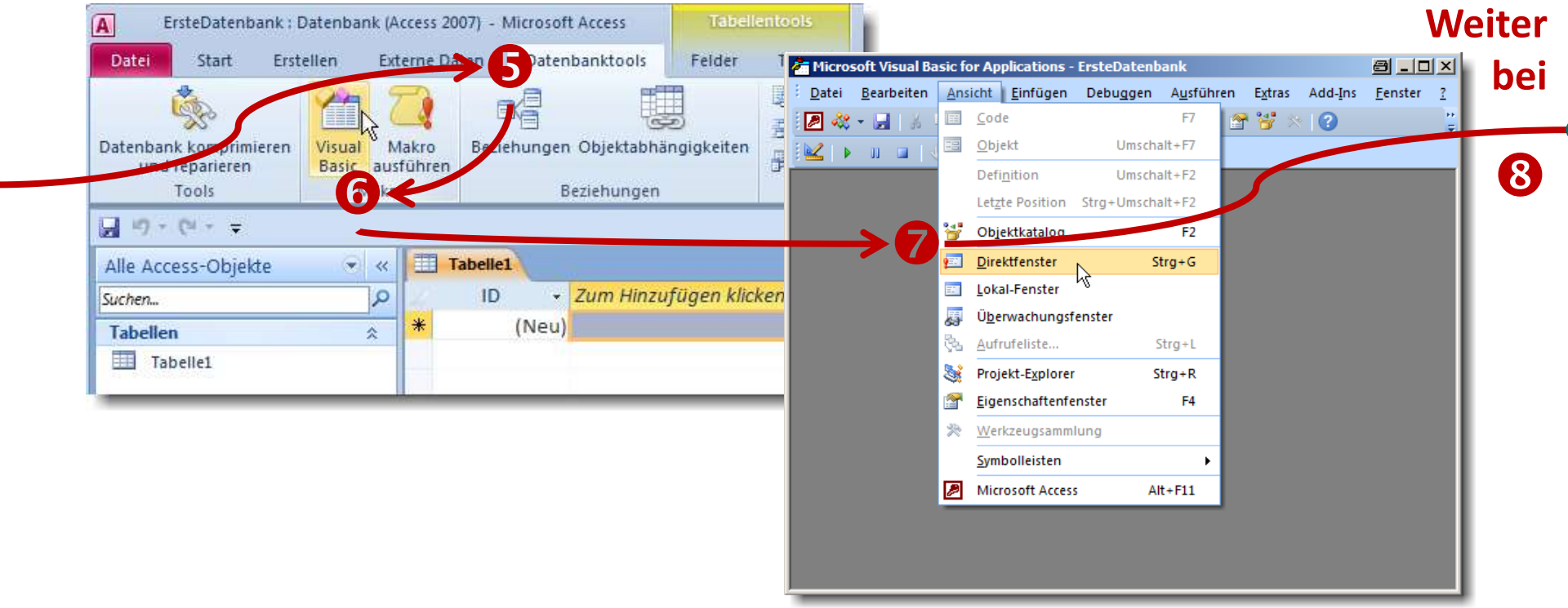


- **Starten von MS Access**
  - z.B. Desktop-Icon oder
  - z.B. über Menü Start>Alle Programme>Microsoft Office

Leere Datenbank anlegen

- bei verfügbaren Vorlagen "Leere Datenbank wählen"
- im rechten Fensterbereich im Feld "Dateiname" einen Dateinamen erfassen
- Schaltfläche "Erstellen" betätigen

MS Access mit VBA – Erste Schritte (2)

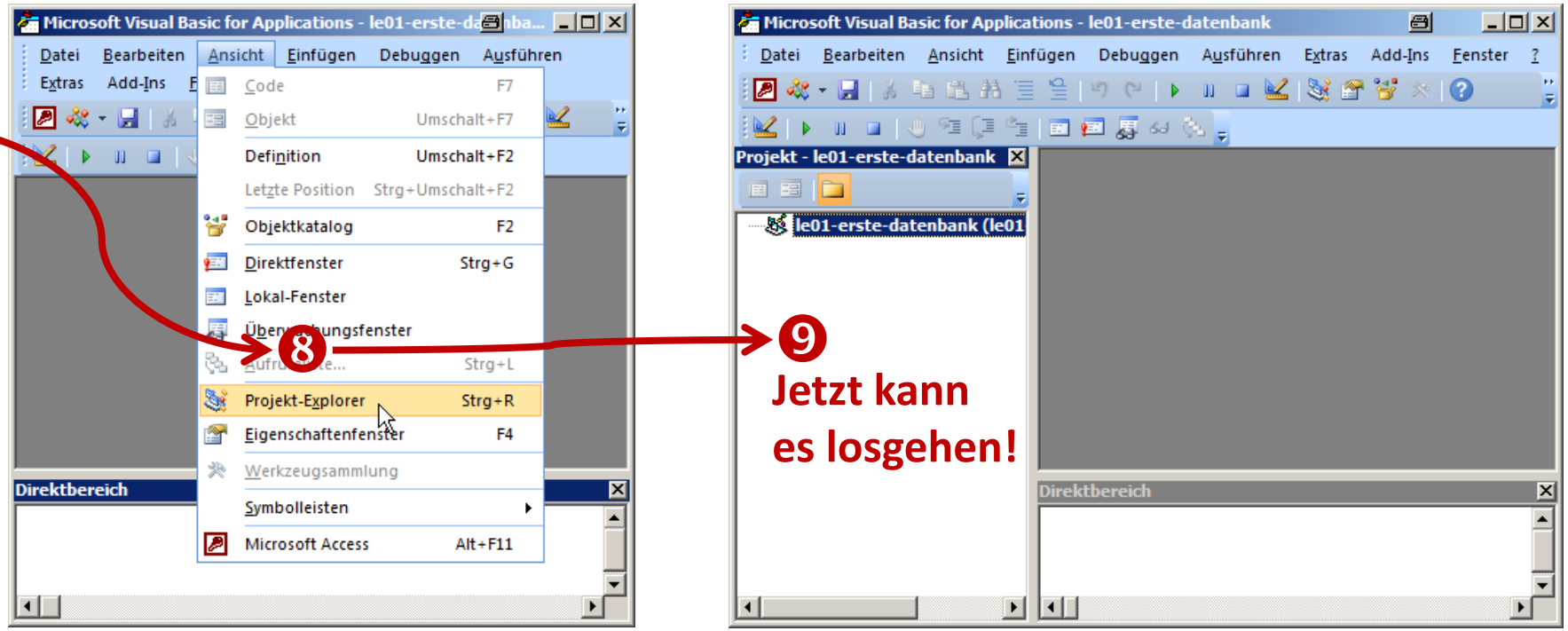


- **In neu angelegter, leerer Datenbank**
  - Registerkarte "Datenbanktools" auswählen
  - Icon "Visual Basic" wählen

Im geöffneten "Visual Basic for Applications"

- Menü Ansicht>Direktfenster einbleden (Strg+G)

MS Access mit VBA – Erste Schritte (3)



- **im Menü "Ansicht" Menüpunkt "Projekt-Explorer" wählen**

Jetzt kann die Programmierung mit dem Visual Basic-Editor beginnen

Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

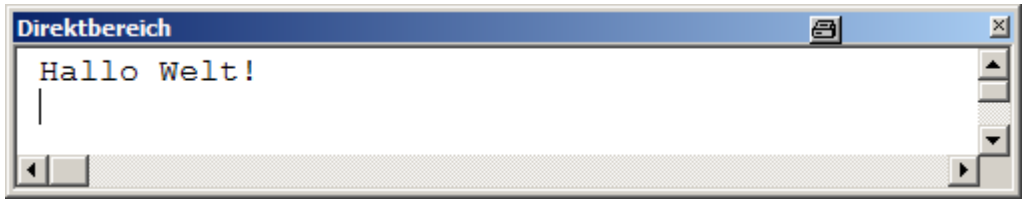
- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

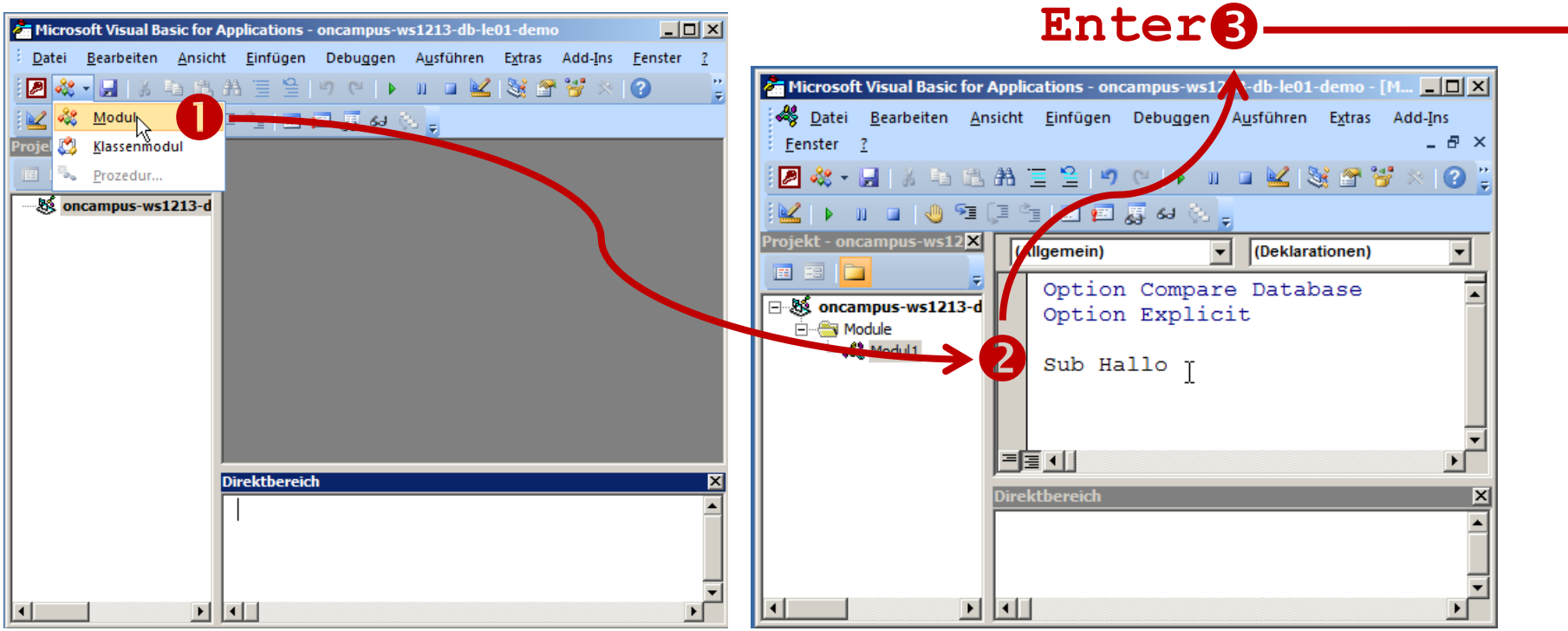
MS Access mit VBA – Hallo Welt! (1)

Demo 1.0

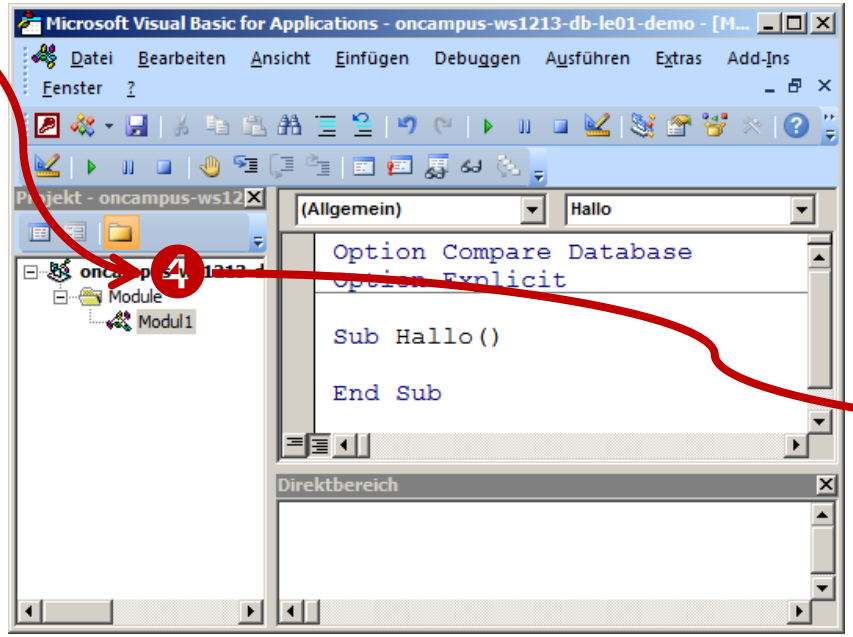
- Programm in VBA für MS-Access, das "Hallo Welt!" im Direktfenster ausgibt.
- Beispiel:

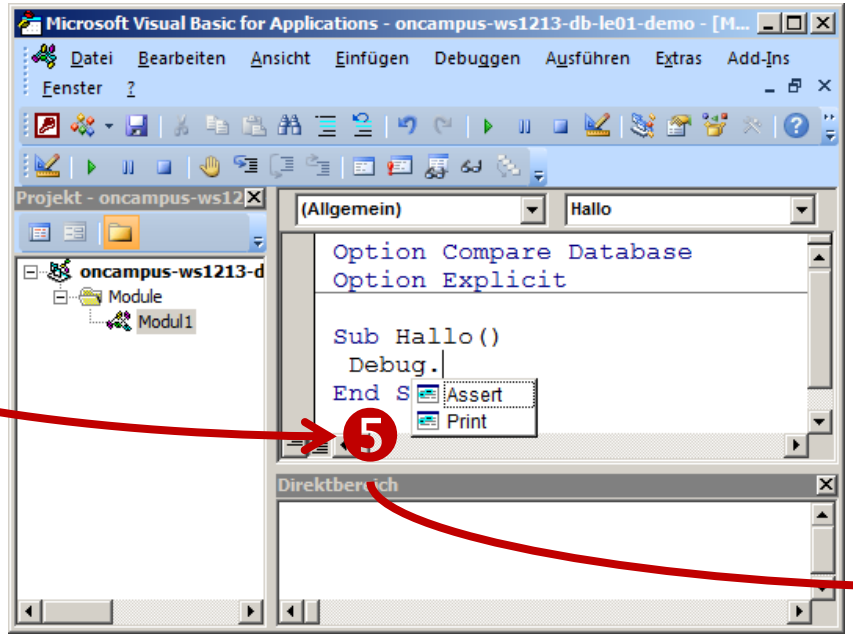


MS Access mit VBA – Hallo Welt! (2)

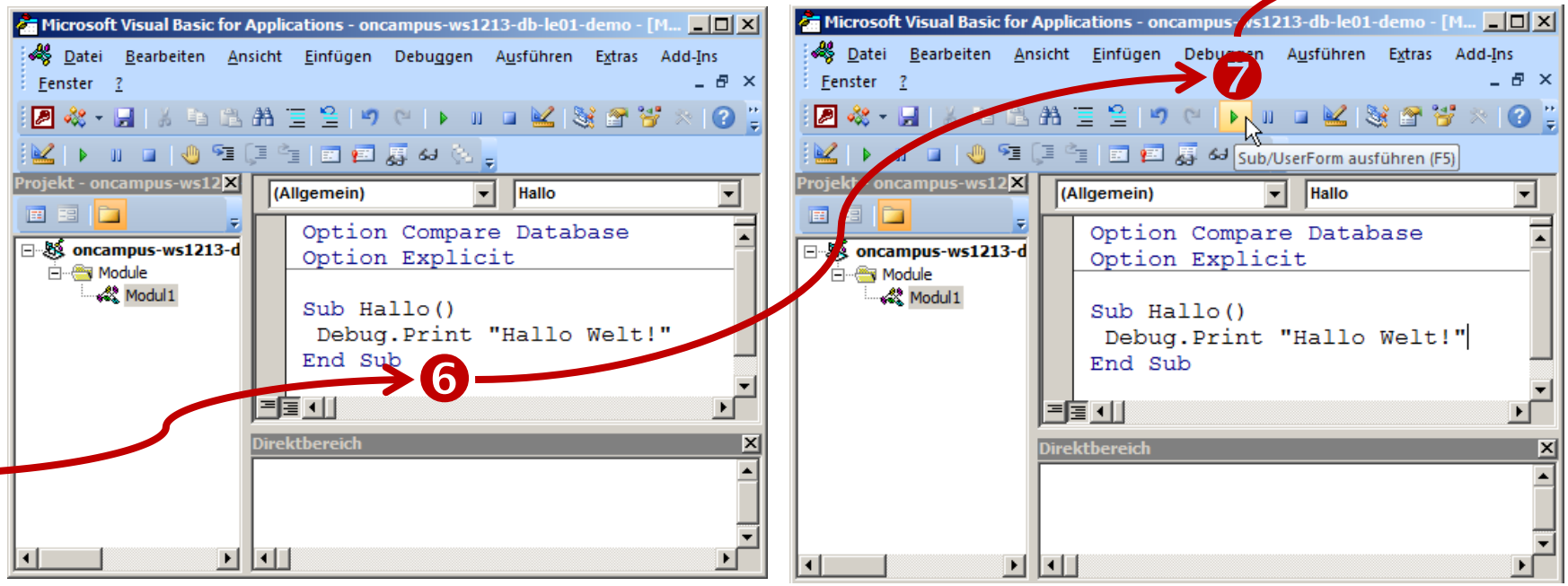


MS Access mit VBA – Hallo Welt! (3)

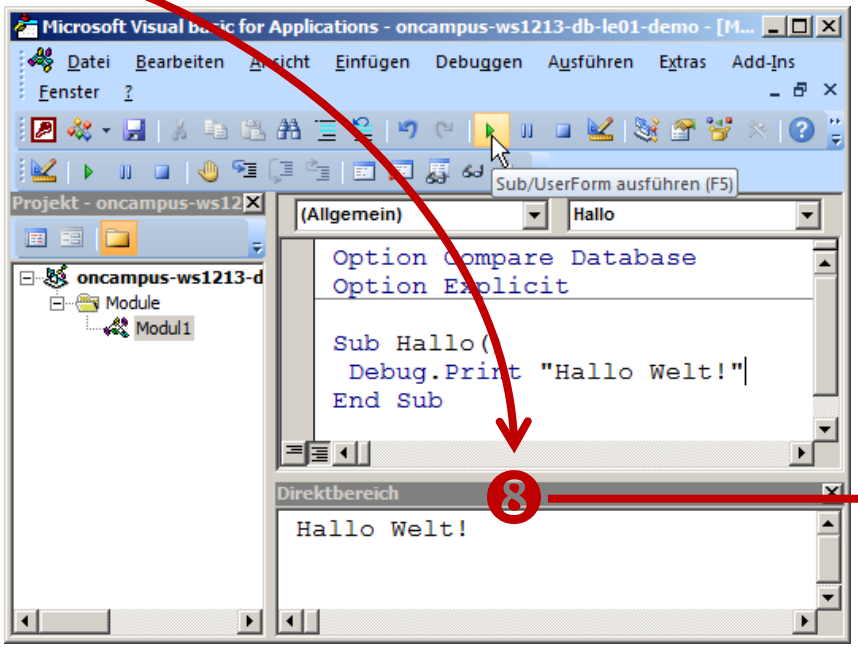




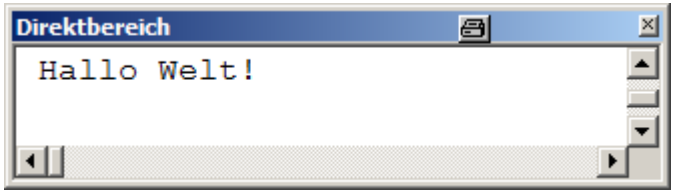
MS Access mit VBA – Hallo Welt! (4)



MS Access mit VBA – Hallo Welt! (5)



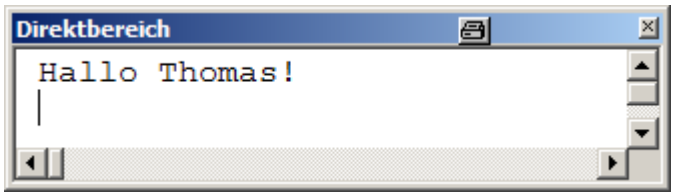
8 → 9 **Fertig!**



Übung: Hallo Welt!

Aufgabe 1.0

- Schreiben Sie ein Programm in VBA für MS-Access, das eine Begrüßung und Ihren Namen im Direktfenster ausgibt.
- Beispiel:



Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

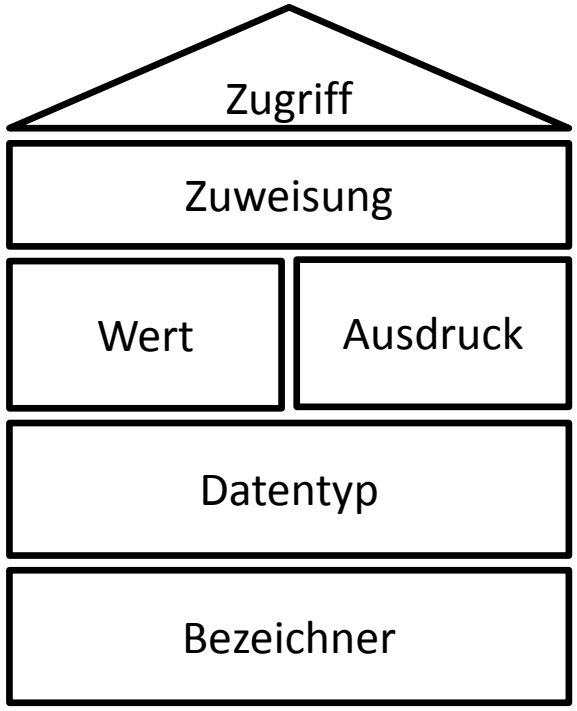
Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

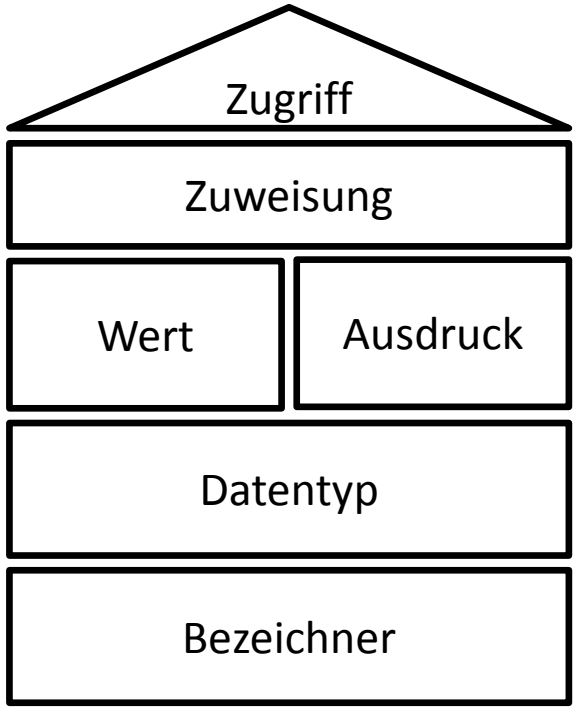
Ausblick

Variable

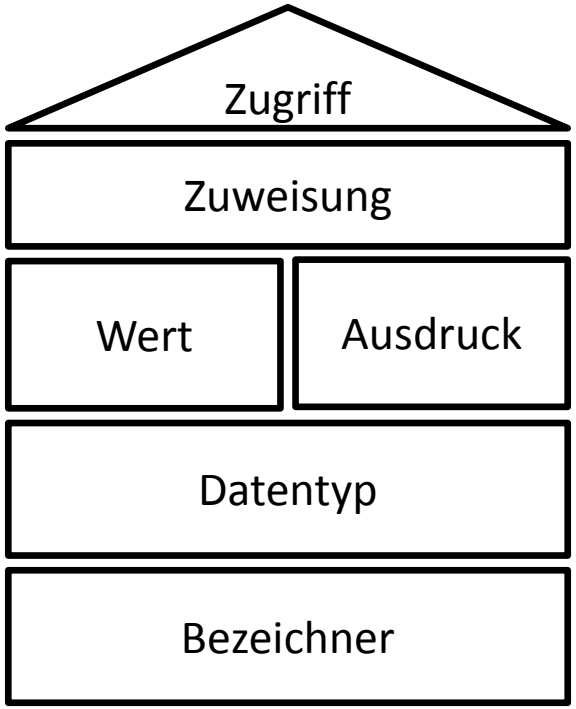
- hat einen **Bezeichner** (Namen)
- speichert Werte eines definierten **Datentyps**
- **Werte** oder **Ausdrücke** werden ihr **zugewiesen**
- bietet
  - lesenden **Zugriff** und
  - schreibenden/ändernden Zugriff auf gespeicherten Wert



Variable



Variable



Variable

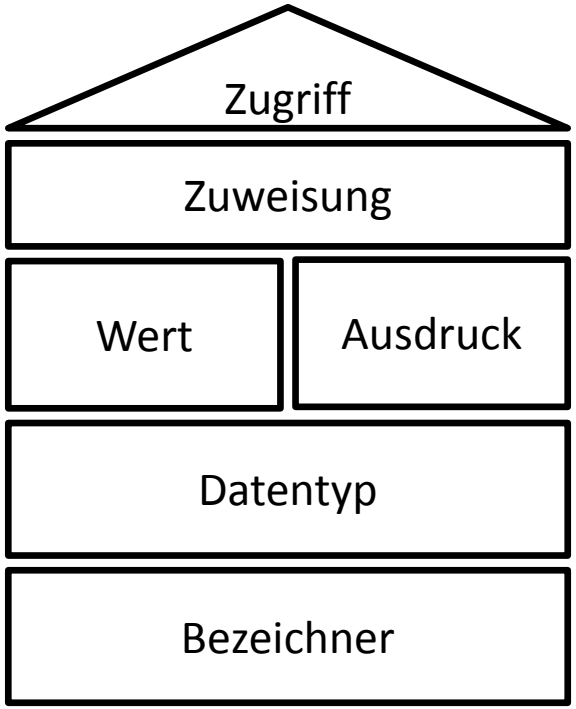
Bezeichner

- mit Buchstaben beginnen, dann Buchstaben, Ziffern oder Unterstriche
- keine Leerzeichen, Punkt, Ausrufezeichen, Sonderzeichen (wie @, &, $, #) und Operatoren (+, -, /, =)
- eindeutig (innerhalb ihre Gültigkeitsbereichs, siehe unten)
- Max. 255 Zeichen
- keine Reservierten Schlüsselworte
- sollten einer Konvention folgen (z.B. der Ungarischen Notation)

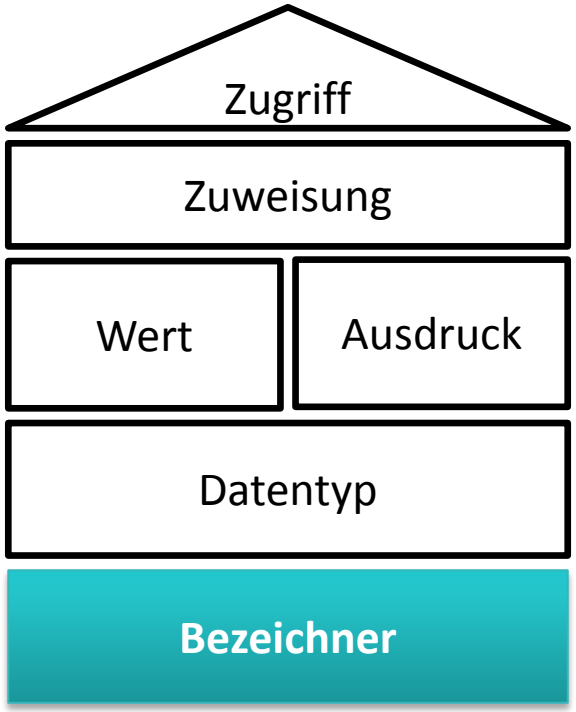
Beispiele

- **`i`**
- **`vorname`**
- **`strVorname`**
- **`intZahl1`**
- auch: **`c3po`** oder **`r2d2`** oder **`HAL_9000`**
- aber nicht: **`9000_HAL`** oder **`2d2r`**

Variable



Variable



Variable

Datentyp

- definiert einen Bereich oder eine Menge von Werten, die in einer Variable dieses Typs gespeichert werden können und
- definiert Operationen, die auf den Werten möglich sind
- bestimmt den zur Speicherung von Werten benötigten Platz
- hat einen eindeutigen Bezeichner

Beispiele

- **`Integer`**: Ganze Zahlen, z.B. mit Operationen Addition, Division
- **`Single`**: Gebrochene Zahlen, z.B. mit Operationen Addition, Division
- **`String`**: Text, z.B. mit Operationen zum Zusammenfügen, aber nicht zur Division
- **`Boolean`**: Wahrheitswerte, z.B. mit Operationen zum Vergleich

Wichtige VBA-Datentypen

Ganze Zahlen

Gebrochene Zahlen

Wahrheitswerte

Datum und Zeit

Zeichenketten

Wichtige VBA-Datentypen

Ganze Zahlen

Typ	Speicherbedarf	Kleinster Wert	Größter Wert
Byte	1 Byte	0	255
Integer	2 Byte	-32.768	32.767
Long	4 Byte	-2,147.483.648	2,147.483.647

Gebrochene Zahlen

Wahrheitswerte

Datum und Zeit

Zeichenketten

Wichtige VBA-Datentypen

Ganze Zahlen

Gebrochene Zahlen

Typ	Speicherbedarf	Wertebereich
Single	4 Bytes	+/-3,402823E38 bis +/-1,401298E-45
Double	8 Bytes	-1,79769313486231E308 bis -4,94065645841247E-324
Currency	8 Bytes	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807

Wahrheitswerte

Datum und Zeit

Zeichenketten

Wichtige VBA-Datentypen

Ganze Zahlen

Gebrochene Zahlen

Wahrheitswerte

Typ	Speicherbedarf	Wertemenge
Boolean	1 Byte	True, False

Datum und Zeit

Zeichenketten

Wichtige VBA-Datentypen

Ganze Zahlen

Gebrochene Zahlen

Wahrheitswerte

Datum und Zeit

Typ	Speicherbedarf	Wertebereich
Date	8 Bytes	Datum vom 01. Januar 0100 bis zum 31. Dezember 9999 Uhrzeit von 0:00:00 bis 23:59:59

Zeichenketten

Wichtige VBA-Datentypen

Ganze Zahlen

Gebrochene Zahlen

Wahrheitswerte

Datum und Zeit

Zeichenketten

Typ	Speicherbedarf	Wertebereich
String	10 Bytes + 1 Byte je Zeichen	< 2,1 Mrd. Zeichen

Wichtige VBA-Datentypen

Ganze Zahlen

Gebrochene Zahlen

Wahrheitswerte

Datum und Zeit

Zeichenketten

Variable

Deklaration

- Festlegung eines Bezeichners und eines Datentyps für eine Variable
- Aufbau: **`Dim`** <*bezeichner*> **`As`** <*Datentyp*>
- führt dazu, dass eine ausreichend große Speicherstelle im Arbeitsspeicher über den festgelegten Bezeichner ansprechbar ist
- diese Speicherstelle hat per Konvention noch keinen gültigen Wert (dieser muss noch zugewiesen werden, siehe folgende Folien)

Variable

Beispiele

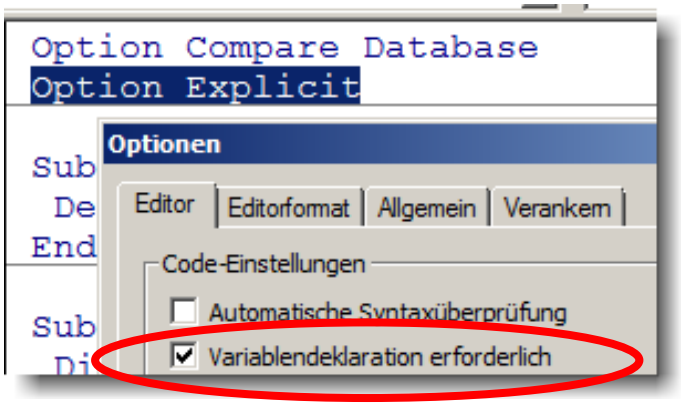
```
Dim i As Integer

Dim vorname As String

Dim strVorname As String
```

Variable

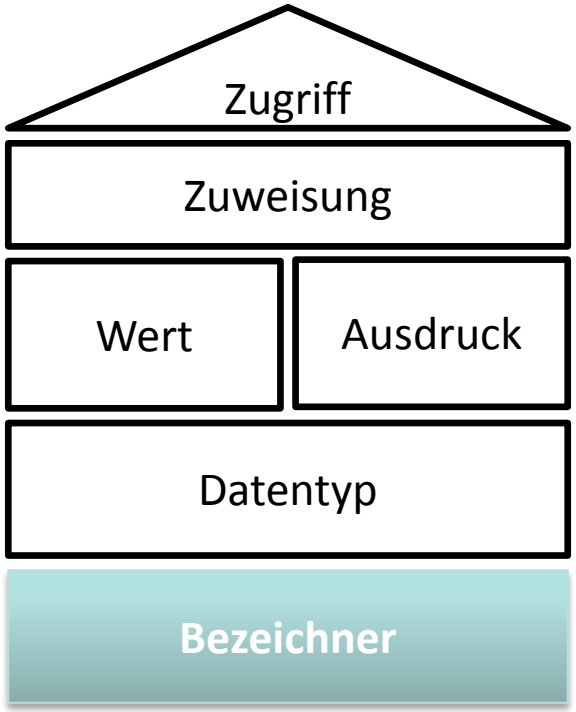
Hinweis

- VBA deklariert Variablen standardmäßig, ohne dass wir dies explizit tun müssen
- dies ist eine sehr häufige Quelle von Fehlern! Deshalb sollte diese Option in Access unbedingt deaktiviert werden
  - Menü "Extras" > "Optionen" > Registerkarte "Editor" > "Variablendeklaration erforderlich"

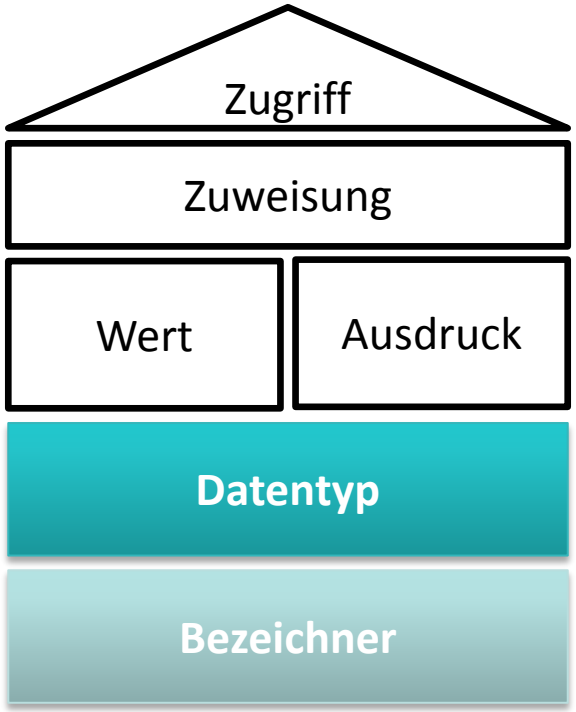
    oder
  - **`Option Explicit`** manuell zum Beginn des Moduls einfügen



Variable



Variable



Variable

Wert

- Ausdruck, der nicht weiter ausgewertet werden kann
- Literal genau ein Wert (Element) aus der Wertemenge eines Datentyps

Beispiele

- **`7`** z.B. Wert für eine ganze Zahl
- **`"sieben"`** z.B. Wert für eine Zeichenkette
- **`"7"`** z.B. Wert für ein Zeichen einer Zeichenkette
- **`true`** als Wert der Wertemenge **`{true, false}`** für einen Wahrheitswert (Boolean)

Variable

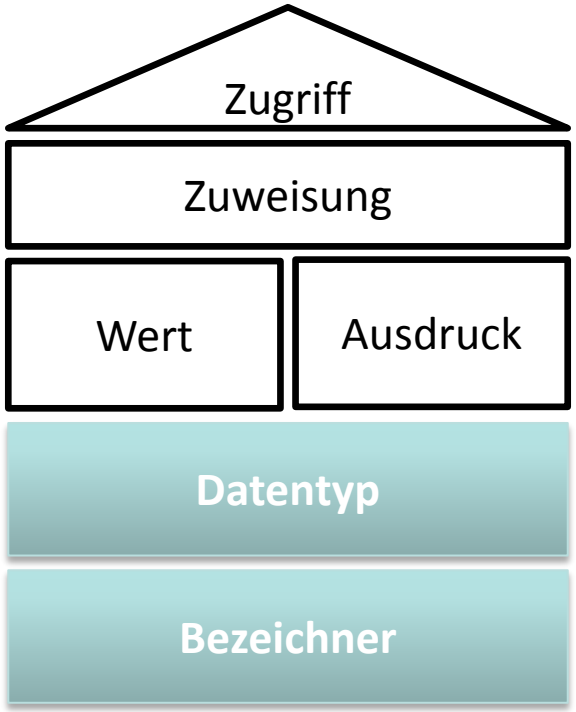
Ausdruck

- Kombination aus Werten, Operatoren, Variablen oder Funktionen
- nach definierter Berechnungsregel auswertbar (d.h. sein Wert kann berechnet werden)
  - Operanden und Operatoren passen sinnvoll
  - Operatorprioritäten werden berücksichtigt
- testweise im Direktbereich mittels "**?**"-Anweisung auswertbar
- innerhalb von Zuweisungen verwendbar

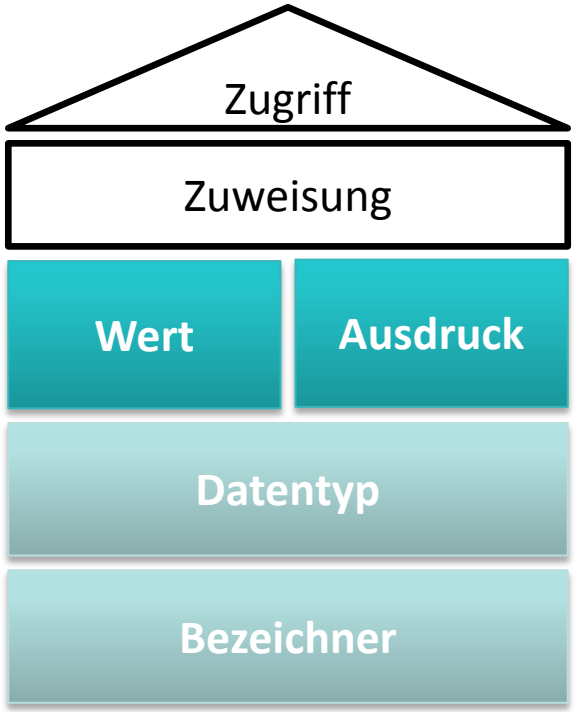
Beispiele

- `1 + 1`
- `Not(false)`
- `2 * 4 = 8`

Variable



Variable



Variable

Zuweisung

- Optionale Anweisung **`Let`**
- Symbol **=** als Zuweisungsoperator (nicht zu verwechseln mit Prüfung auf Gleichheit)
- Aufbau: **`Let`** *<variable>* **=** *<wert-oder-ausdruck>*
- Anweisung mit zwei Seiten
  - Links: gibt an, welcher Variablen der Wert zugewiesen wird
  - Rechts: welcher Wert/Ausdruck zugewiesen wird
- Zuerst rechte Seite vollständig ausgewertet (d.h. berechnet); erst dann wird Ergebnis der linken Seite zugewiesen
- Reihenfolge wichtig für Zuweisungen, die Variable der linken Seite auf der rechten Seite enthalten

Beispiele (auf der nächsten Folie)

Variable

Beispiele für Zuweisungen

```
Let intVarA = 1

Let intVarB = 2

Let intErg = intVarA + intVarB

Let intVarA = intVarA + 1
```

Variable

Initialisierung (erstmalige Zuweisung)

- nach Deklaration einer Variable hat sie (per Definition) noch keinen gültigen Wert
- erstmalige Zuweisung eines Wertes zu einer Variable (oder Konstante, siehe unten) wird als Initialisierung bezeichnet
- Initialisierung erfolgt
  - für Variablen als eigenständige Anweisung nach der Deklaration oder
  - für Konstanten zusammen mit der Deklaration (siehe unten)

Variable

Beispiele für Zuweisungen

```
' Deklarationen
Dim a As Byte
Dim strName As String

' Initialisierung von a
Let a = 1

' Initialisierung von strName
Let strName = "Max Mustermann"
```

Variable

Typ-Umwandlung (implizit)

- Werte von Variablen mit unterschiedlichen Datentypen in Ausdruck
- häufige Problemstellung bei Auswertung von Ausdrücken und bei Zuweisungen
- (unsichtbare) Typ-Umwandlung während der Auswertung von Ausdrücken als implizit bezeichnet
- Implizite Typumwandlung geht ohne Informationsverlust einher, z.B.
  - Alle Datentypen nach String
  - Byte nach Integer, Long, Decimal
  - Currency nach Decimal

Variable

Beispiele für Typumwandlungen (implizit)

```
Dim a As Byte
Dim b As Integer
Dim strName As String

Let a = 1
Let strName = "Max Mustermann"

' Implizite Typumwandlung a (Byte in String)
Let strName = strName & a

' Implizite Typumwandlung a (Byte in Integer)
Let b = a + 2
```

Variable

Typ-Umwandlung (explizit)

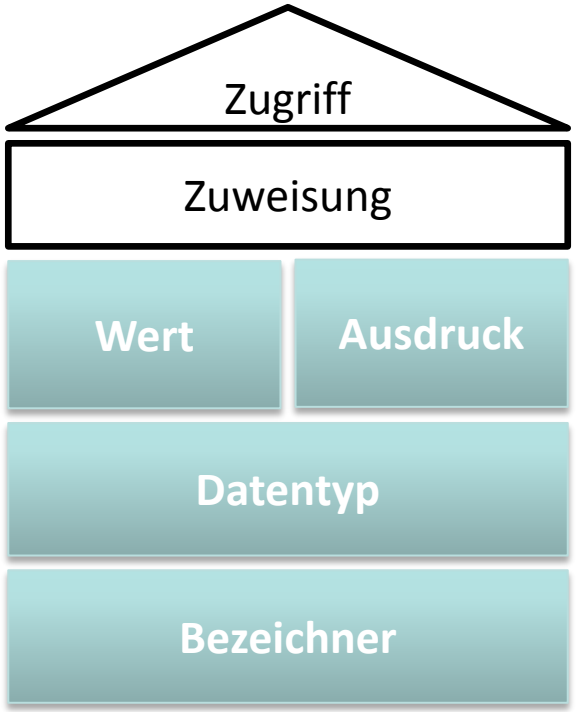
- explizite notwendige Umwandlungen können mit (gewolltem) Informationsverlust einhergehen, z.B.
  - Double nach Integer unter Verlust der Kommastellen
- Verwendung von Cast-Operatoren, um Umwandlung zu steuern
  - CBool, CInt, CDec, CCur
  - CStr, CDate, CLng, CDbl

Variable

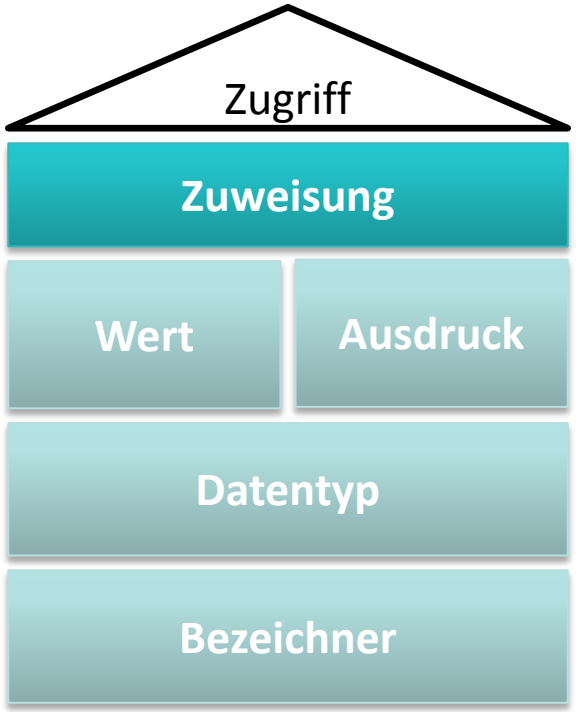
Beispiele für Typumwandlungen (explizit)

```
Dim dblZahl As Double
Dim intZahl As Integer
Dim strNummer As String

Let dblZahl = 1.2345
Let strNummer = "123"

' Explizite Typumwandlung (Double in Integer)
Let intZahl = CInt(dblZahl)

' Explizite Typumwandlung (String in Integer)
Let intZahl = intZahl + CInt(strNummer)
```

Variable



Variable



Variable

Lesender Zugriff

- Bezeichner der Variable ist ein Zeiger auf einen Speicherbereich
- Beim Lesen wird der referenzierte Speicherbereich ermittelt
- anschließend wird der in diesem Speicherbereich vorhandene Wert gelesen

Schreibender/ändernder Zugriff

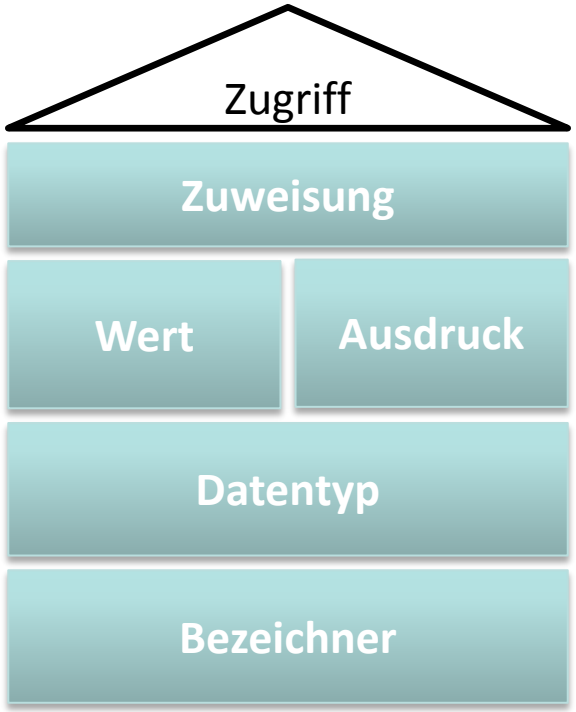
- Zuweisungsoperator verwenden, um an dem durch die Variable referenzierten Speicherbereich einen Wert zu schreiben

Variable

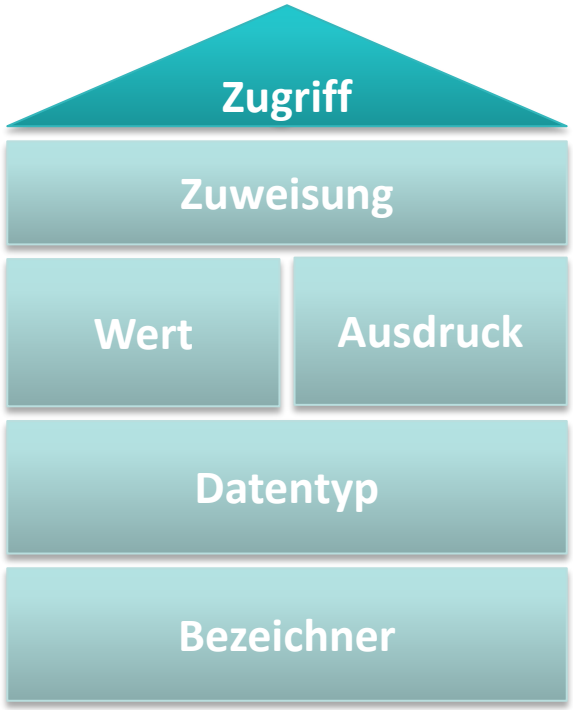
Beispiele für Zugriffe

```
' Lesender Zugriff auf a
Debug.Print a

' Schreibender Zugriff auf i
Let i = 5

' Lesender und schreibender Zugriff
Let b = i + a
```

Variable

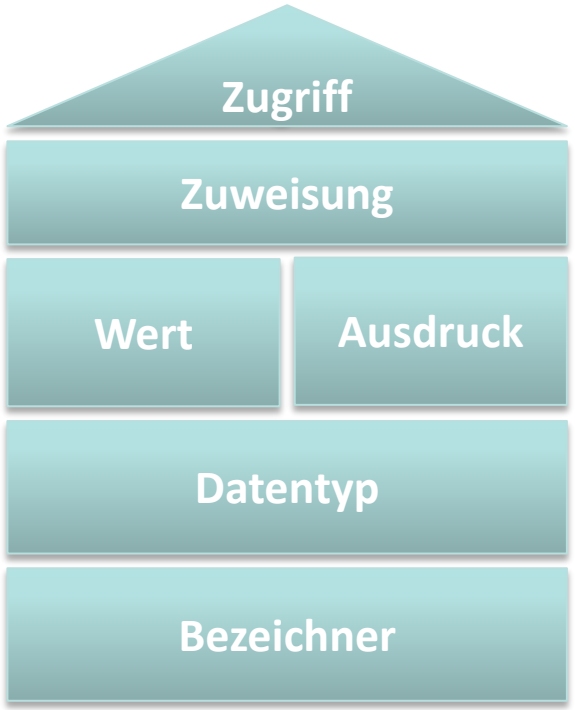


Variable



Variable

- hat einen **Bezeichner** (Namen)
- speichert Werte eines definierten **Datentyps**
- **Werte** oder **Ausdrücke** werden ihr **zugewiesen**
- bietet
  - lesenden **Zugriff** und
  - schreibenden/ändernden Zugriff auf gespeicherten Wert



Demo: Variablen

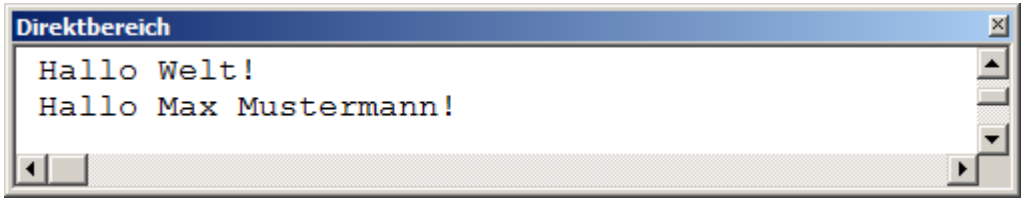
Demo 1:

- Deklarieren von drei Variablen
  - Integer
  - Single
  - String
- Initialisieren der Variablen
- Zuweisungen
  - der Integer-Variable die Summe aus eigenem Wert und dem Wert der Single-Variable zuweisen
  - dem String den eigenen Wert verknüpft mit dem Wert der Integer-Variable zuweisen
- Ergebnis im Direktfenster ausgeben

Übung: Variable

Aufgabe 1.1:

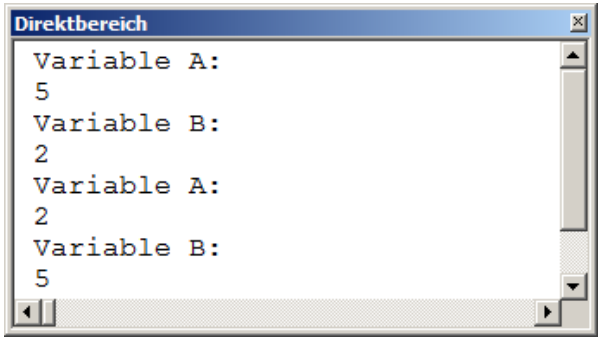
- Deklarieren Sie zwei Variablen vom Typ String
- Initialisieren Sie eine Variable mit Ihrem Vornamen, die andere mit Ihrem Nachnamen
- Geben Sie erst "Hallo Welt!" und dann "Hallo " gefolgt von den Variablenwerten für Vorname und Nachname aus
- Ihr Ergebnis könnte etwa so aussehen:



Übung: Variable

Aufgabe 1.2

- Deklarieren Sie zwei Variablen vom Typ Integer
- Initialisieren Sie die Variablen mit unterschiedlichen Werten
- Implementieren Sie einen Wertetausch der beiden Variablen
  - Hat Variable A den Wert W1 hat und Variable B den Wert W2,
  - soll nach dem Wertetausch A den Wert W2 und B den Wert W1 haben
- Geben Sie die Variablen vor und nach dem Wertetausch im Direktbereich aus
- Ihr Ergebnis könnte so aussehen:



Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

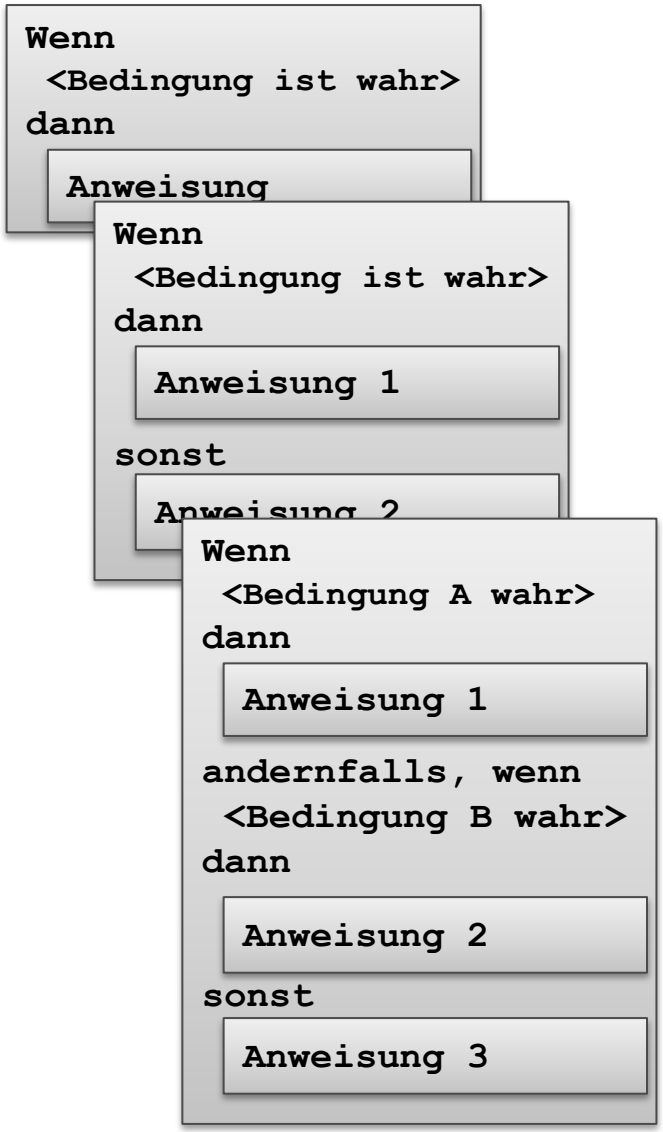
Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Verzweigungen

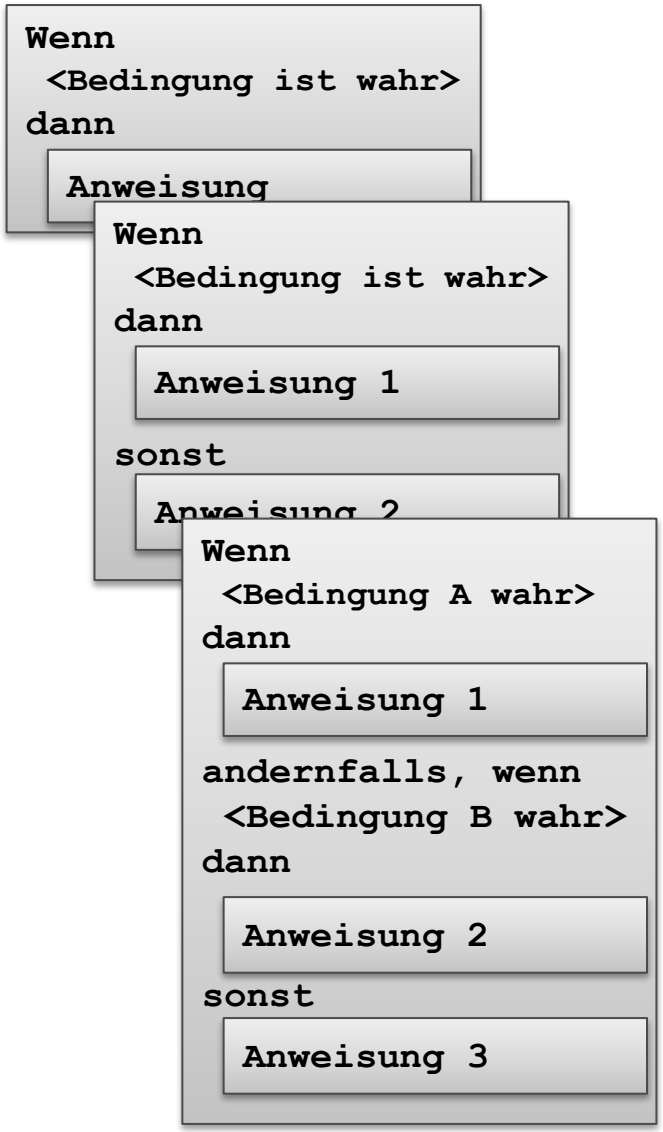
- im Programmablauf Bedingungen prüfen
- Bedingung
  - Ausdruck der Wahrheitswert liefert (Wahr oder Falsch)
  - nutzt immer Variable und häufig Boolsche Operatoren (AND, OR, XOR)
- Anweisungen, für den Fall
  - der erfüllten Bedingung
  - der nicht erfüllten Bedingung



Verzweigungen

Variante 1...

Variante 2...

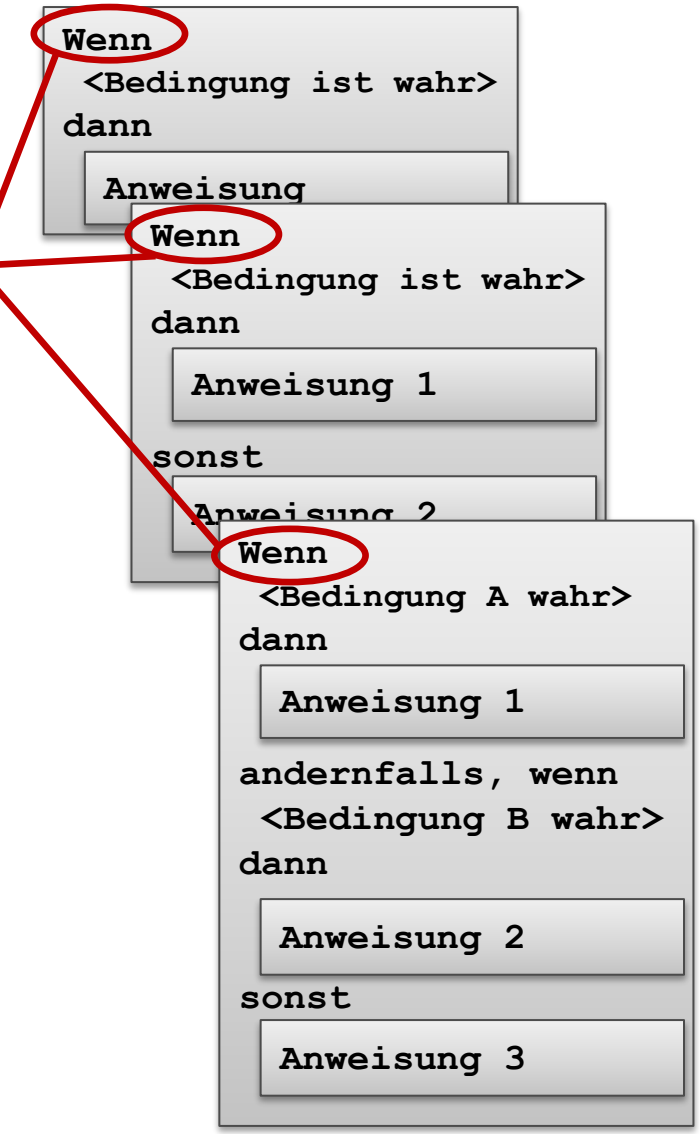


Verzweigungen

Variante 1 mit insgesamt fünf Worten

- `If`
- `Then`
- `Else`
- `ElseIf`
- `End If`

Variante 2...

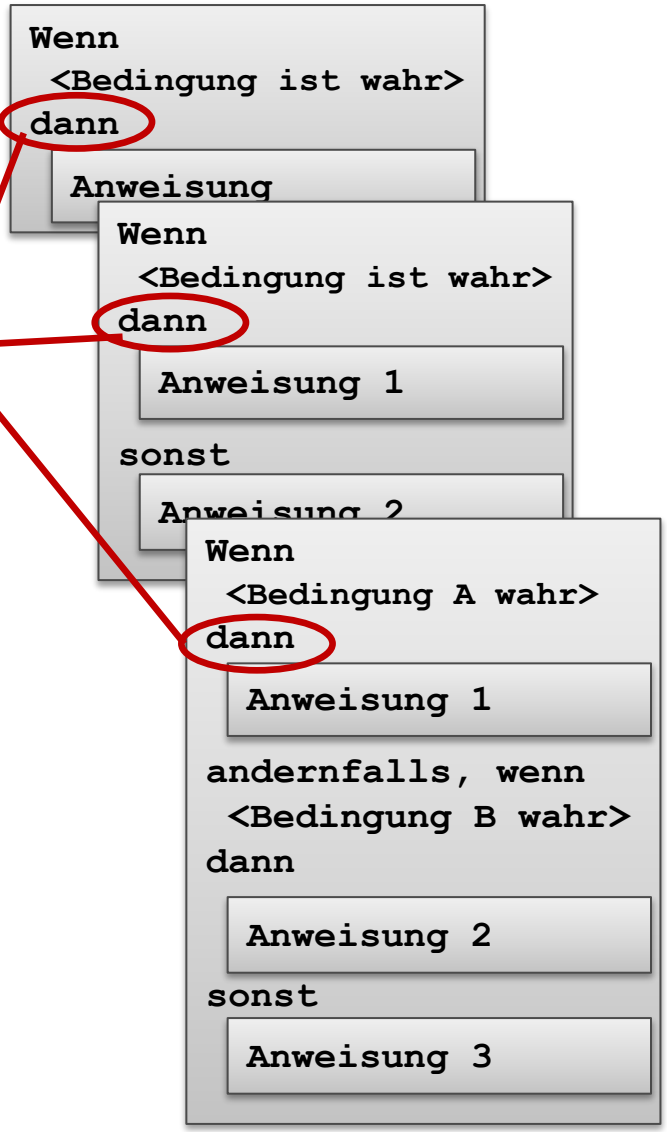


Verzweigungen

Variante 1 mit insgesamt fünf Worten

- `If`
- `Then`
- `Else`
- `ElseIf`
- `End If`

Variante 2...

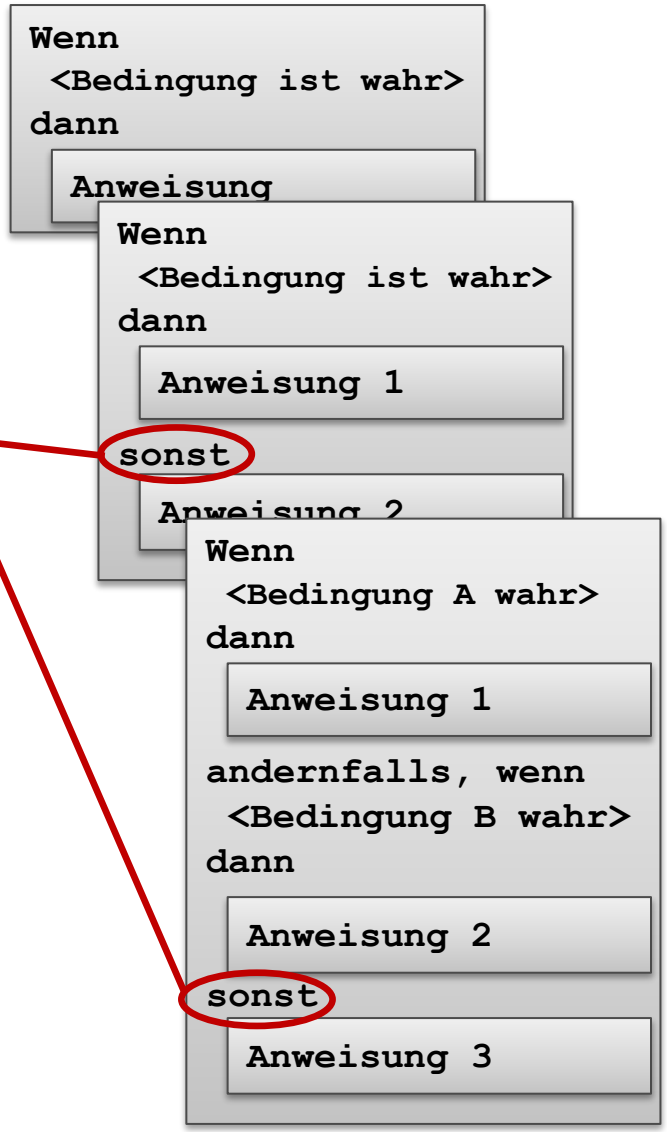


Verzweigungen

Variante 1 mit insgesamt fünf Worten

- `If`
- `Then`
- `Else`
- `ElseIf`
- `End If`

Variante 2...

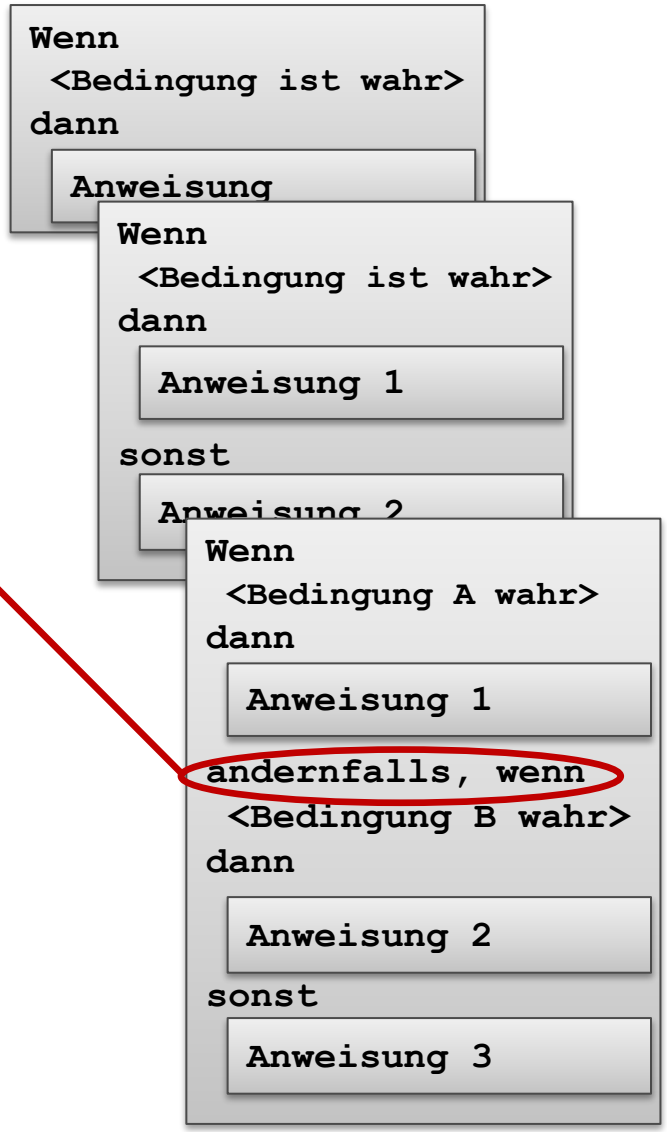


Verzweigungen

Variante 1 mit insgesamt fünf Worten

- `If`
- `Then`
- `Else`
- `ElseIf`
- `End If`

Variante 2...

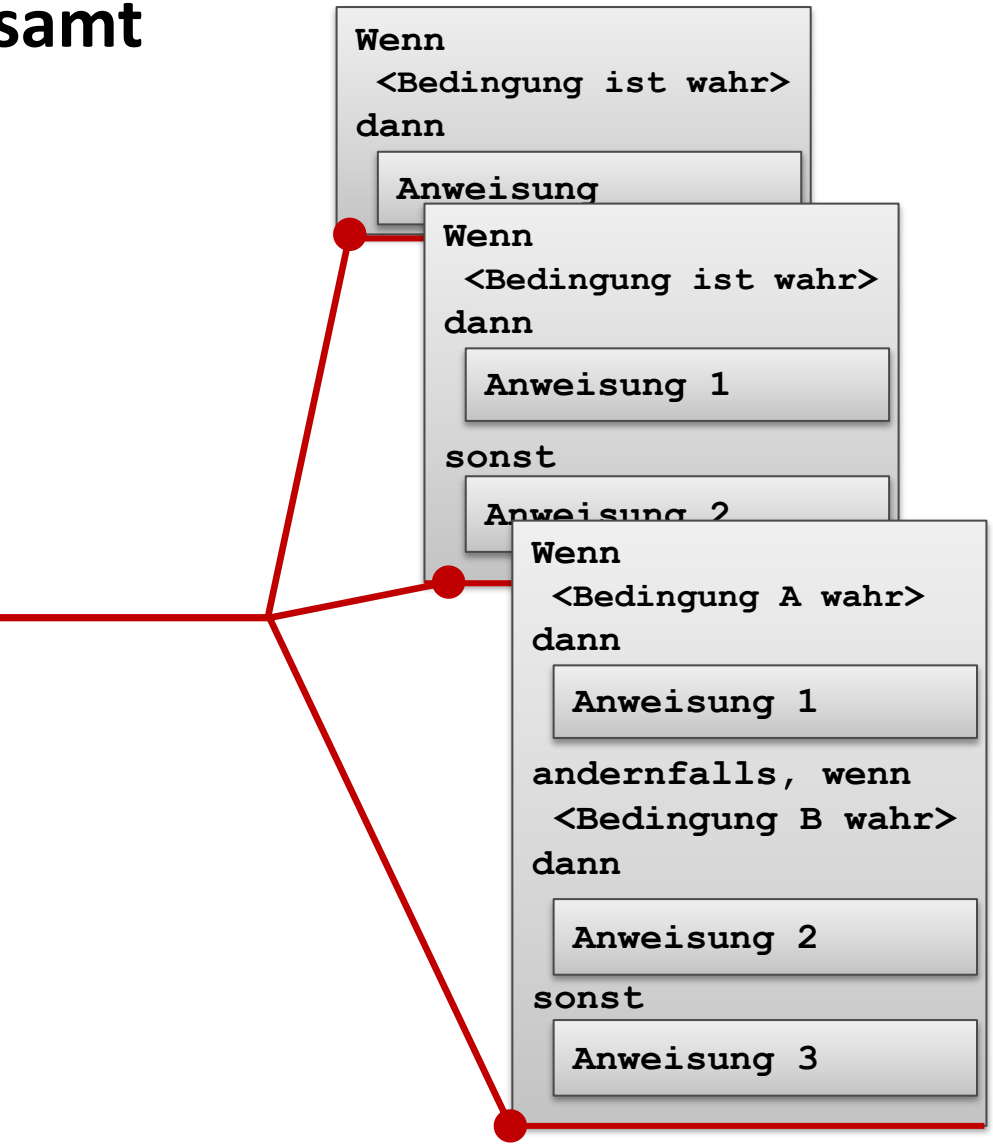


Verzweigungen

Variante 1 mit insgesamt fünf Worten

- `If`
- `Then`
- `Else`
- `ElseIf`
- `End If`

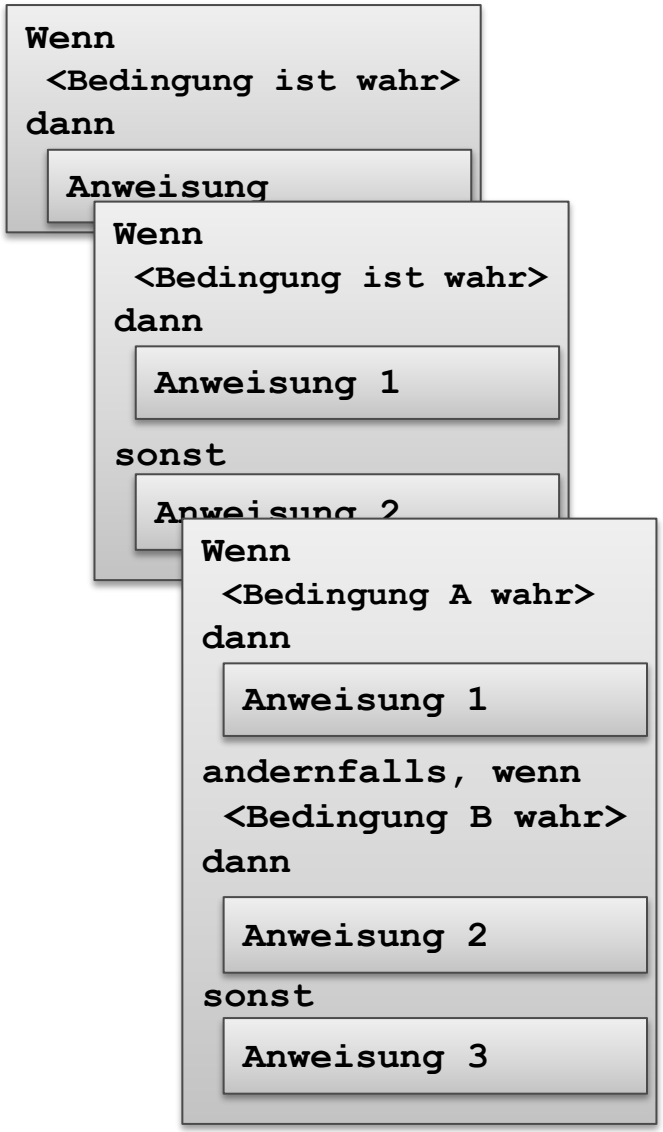
Variante 2...



Verzweigungen

Variante 1...

Variante 2...



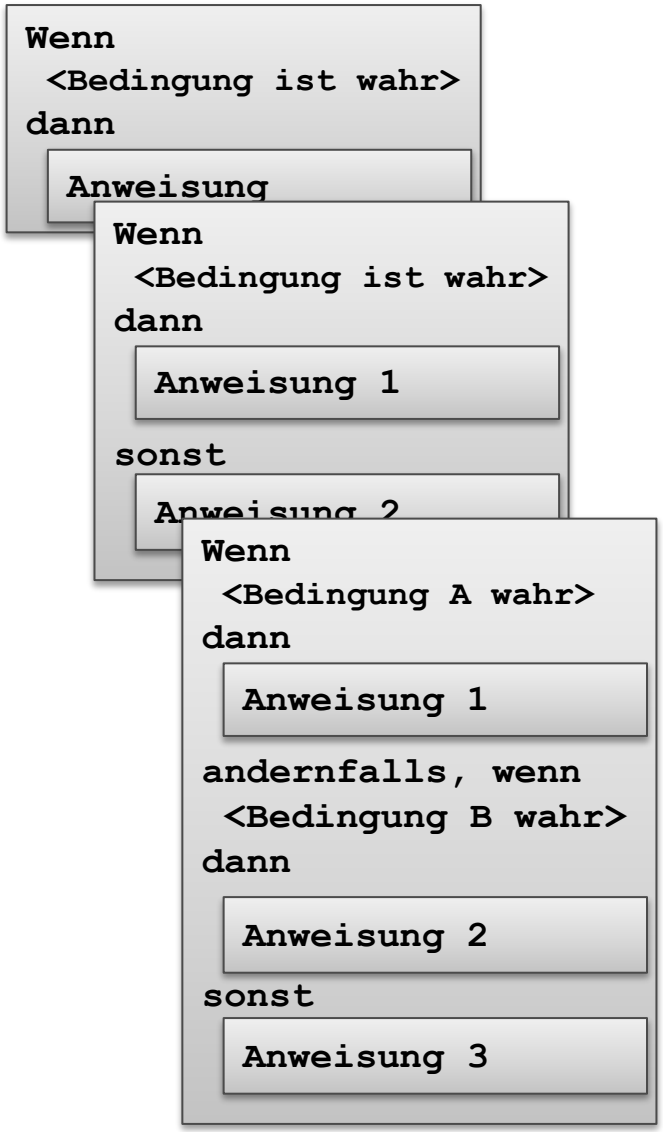
Verzweigungen

Variante 1...

Variante 2 mit insgesamt vier Worten

- `Select Case`
- `Case`
- `Case Else`
- `End Select`

und zusätzlichen Ergänzungen.



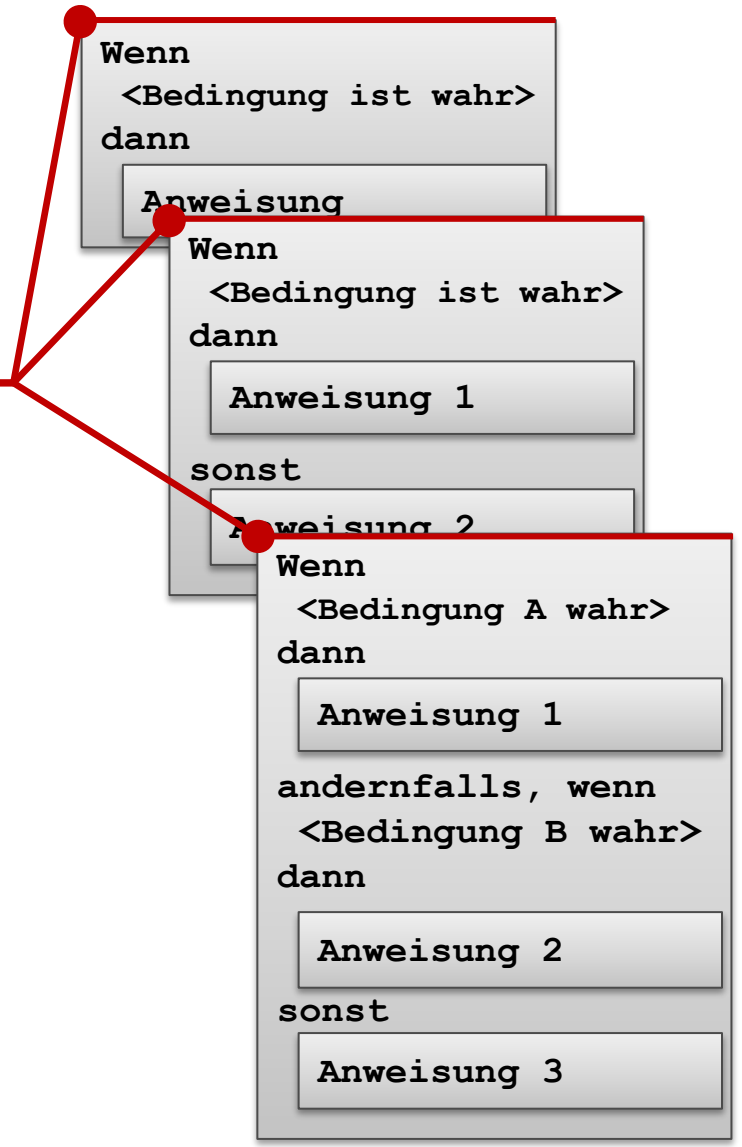
Verzweigungen

Variante 1...

Variante 2 mit insgesamt vier Worten

- `Select Case`
- `Case`
- `Case Else`
- `End Select`

und zusätzlichen Ergänzungen.



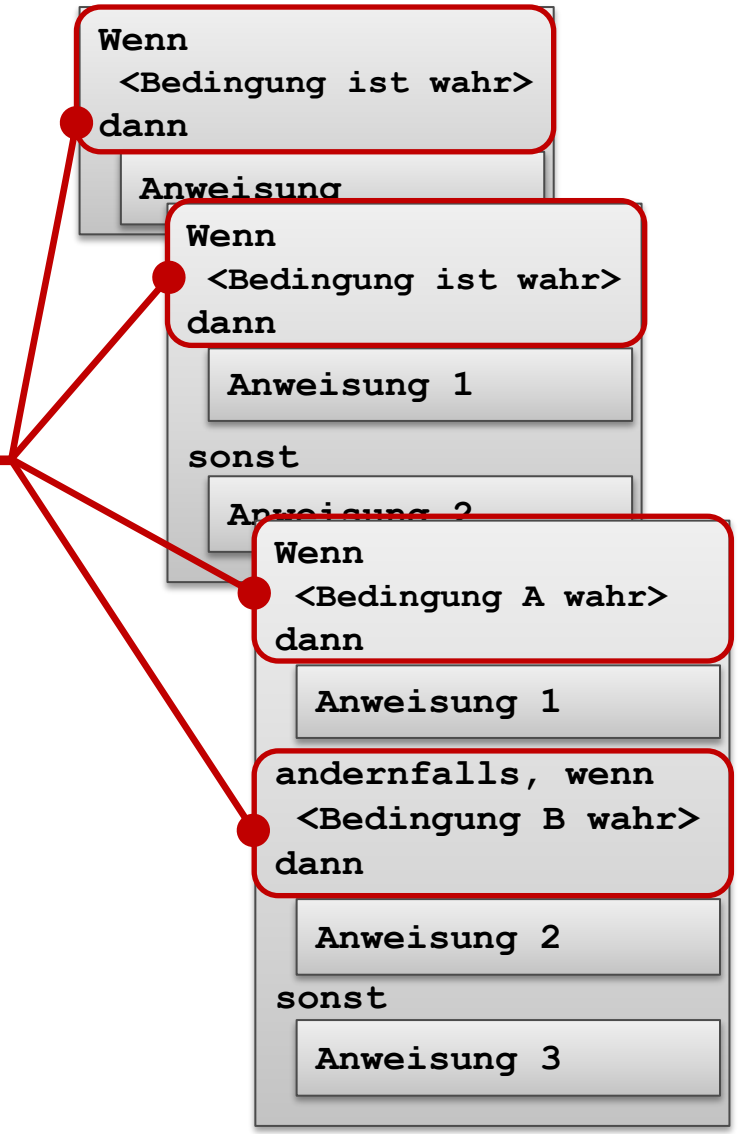
Verzweigungen

Variante 1...

Variante 2 mit insgesamt vier Worten

- `Select Case`
- `Case`
- `Case Else`
- `End Select`

und zusätzlichen Ergänzungen.



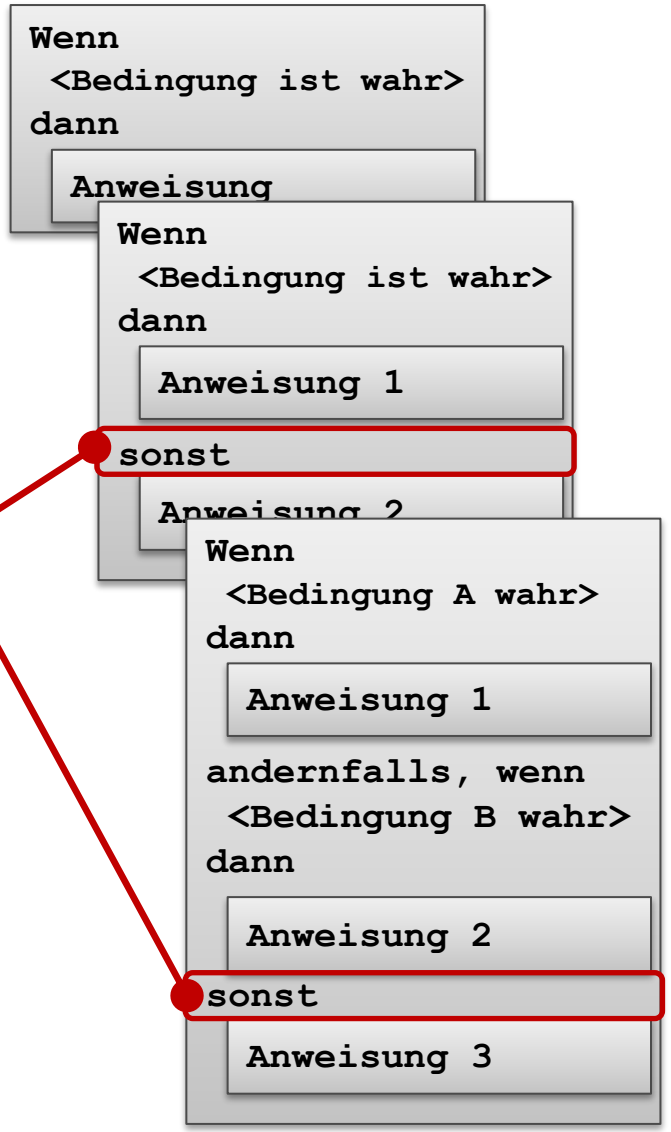
Verzweigungen

Variante 1...

Variante 2 mit insgesamt vier Worten

- `Select Case`
- `Case`
- `Case Else`
- `End Select`

und zusätzlichen Ergänzungen.



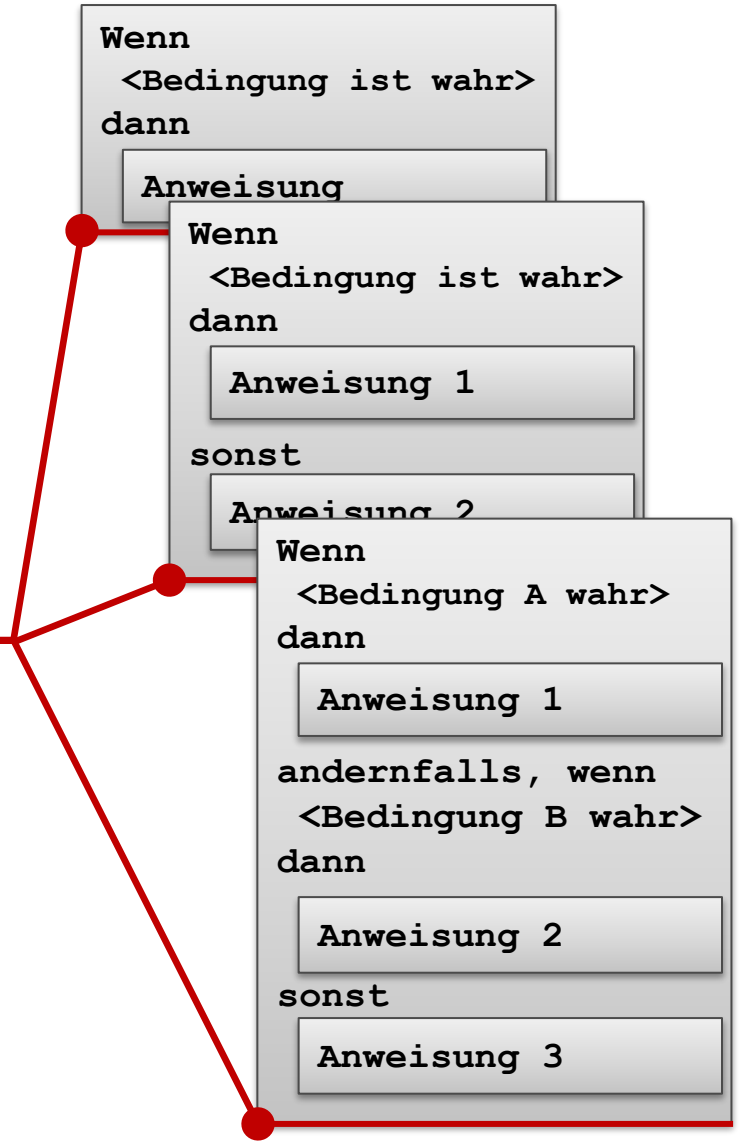
Verzweigungen

Variante 1...

Variante 2 mit insgesamt vier Worten

- `Select Case`
- `Case`
- `Case Else`
- `End Select`

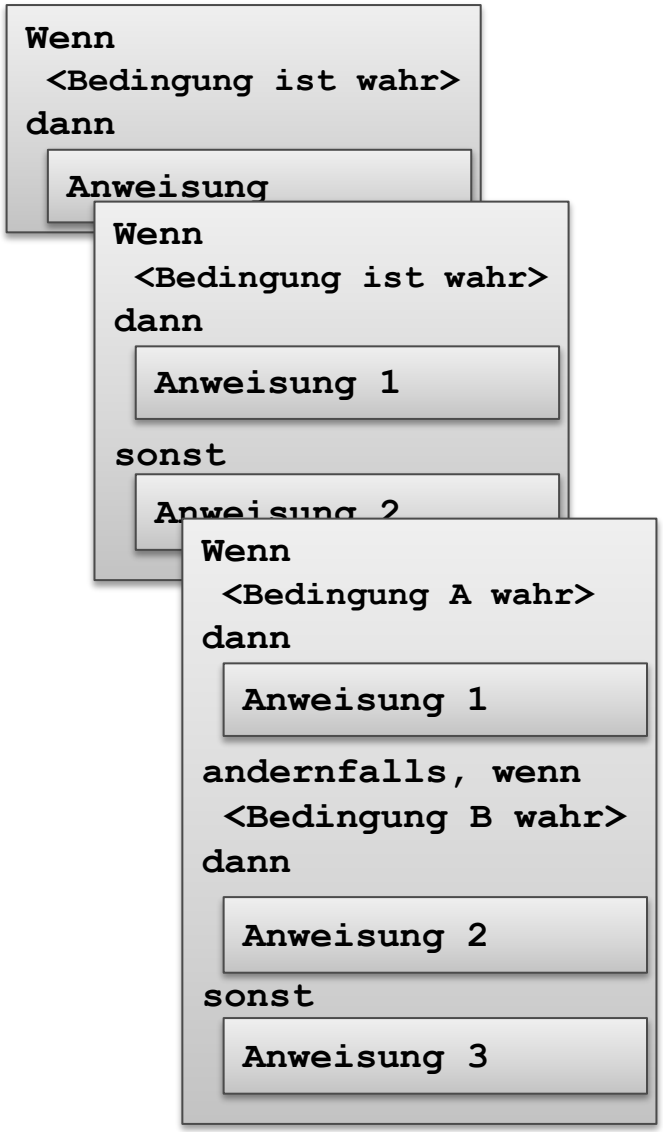
und zusätzlichen Ergänzungen.



Verzweigungen

Variante 1...

Variante 2...



Demo: Verzweigungen (1)

Demo 1.1

- Deklarieren Sie zwei Variablen
  - bolBestanden vom Typ Boolean
  - bytNote vom Typ Byte
- Initialisieren Sie die Variablen
  - bolBestanden soll mit dem Wert False initialisiert werden
  - bytNote soll mit dem Wert 2 initialisiert werden
- Schreiben Sie eine Verzweigung,
  - die den Wert der Variablen bolBestanden immer dann auf True setzt,
  - wenn der Wert von bytNote kleiner oder gleich 4 ist
- Geben Sie den Wert von bolBestanden im Direktbereich aus

Demo: Verzweigungen (2)

Demo 1.2

- Deklarieren Sie zwei Variablen
  - bolBestanden vom Typ Boolean
  - bytNote vom Typ Byte
- Initialisieren Sie die Variable bytNote mit dem Wert 2
- Schreiben Sie eine Verzweigung,
  - die den Wert der Variablen bolBestanden immer dann auf True setzt,
  - wenn der Wert von bytNote gleicher oder gleich 4 ist,
  - sonst den Wert der Variablen bolBestanden auf False setzt
- Geben Sie den Wert von bolBestanden im Direktbereich aus

Demo: Verzweigungen (3)

Demo 1.3

- Deklarieren Sie zwei Variablen
  - strBewertung vom Typ String
  - bytNote vom Typ Byte
- Initialisieren Sie die Variable bytNote mit dem Wert 2
- Schreiben Sie eine If-Then-ElseIf-Else-Verzweigung,
  - die den Wert der Variablen strBewertung auf "sehr gut" setzt,
  - wenn der Wert von bytNote gleich 1 ist,
  - andernfalls, den Wert der Variablen strBewertung auf "gut" setzt,
  - wenn der Wert von bytNote gleich 2 ist,
  - ...
  - sonst den Wert von strBewertung auf "Ungültige Note" setzt.
- Geben Sie den Wert von strBewertung im Direktbereich aus

Demo: Verzweigungen (4)

Demo 1.4

- Deklarieren Sie zwei Variablen
  - strBewertung vom Typ String
  - bytNote vom Typ Byte
- Initialisieren Sie die Variable bytNote mit dem Wert 2
- Schreiben Sie eine **Select-Case**-Verzweigung,
  - die den Wert der Variablen strBewertung auf "sehr gut" setzt,
  - wenn der Wert von bytNote gleich 1 ist,
  - andernfalls, den Wert der Variablen strBewertung auf "gut" setzt,
  - wenn der Wert von bytNote gleich 2 ist,
  - ...
  - sonst den Wert von strBewertung auf "Ungültige Note" setzt.
- Geben Sie den Wert von strBewertung im Direktbereich aus

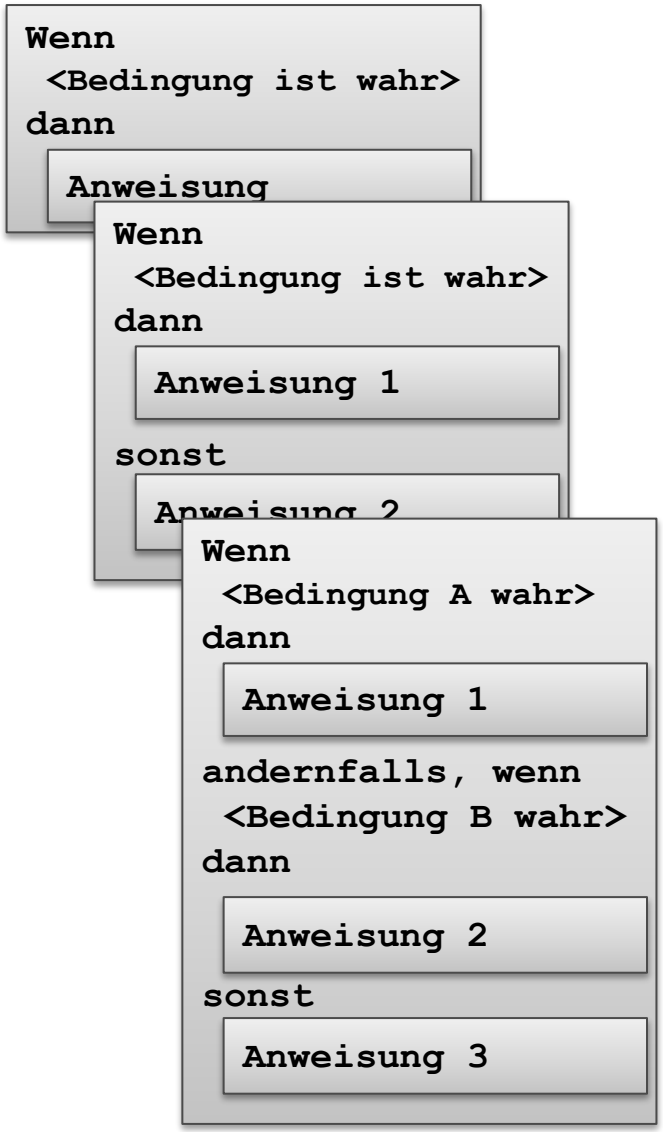
Übung: Verzweigungen

Aufgabe 1.3

- Deklarieren Sie eine Variable vom Typ Byte für den Wochentag
  - der Wert 1 soll Montag entsprechen, der Wert 2 Dienstag, ...
- Initialisieren Sie die Variable mit einer beliebigen Zahl
- Implementieren Sie eine Verzweigung die den Name des Wochentags ausgibt
- Wenn die Variable einen Wert > 7 hat, soll "Ungültiger Wochentag" ausgegeben werden

Verzweigungen

- im Programmablauf Bedingungen prüfen
- Bedingung
  - Ausdruck der Wahrheitswert liefert (Wahr oder Falsch)
  - nutzt immer Variable und häufig Boolsche Operatoren (AND, OR, XOR)
- Anweisungen, für den Fall
  - der erfüllten Bedingung
  - der nicht erfüllten Bedingung



Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Schleifen

- Zählerschleife

  [...]

- Vorprüfend/Kopfgesteuert

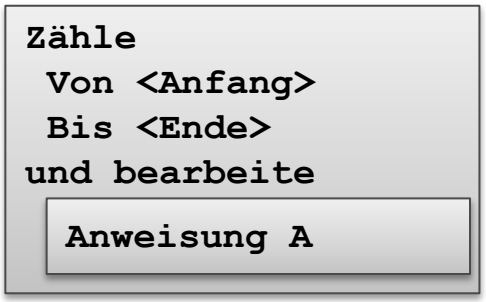
  [...]

- Nachprüfend/Fußgesteuert

  [...]

Schleifen

- Zählerschleife: Wiederholung einer bekannten Anzahl von Durchläufen
  - Beispiel: Es sollen die Zahlen 1 bis 10 addiert werden.
- Vorprüfend/Kopfgesteuert
  [...]
- Nachprüfend/Fußgesteuert
  [...]



Schleifen

- Zählerschleife

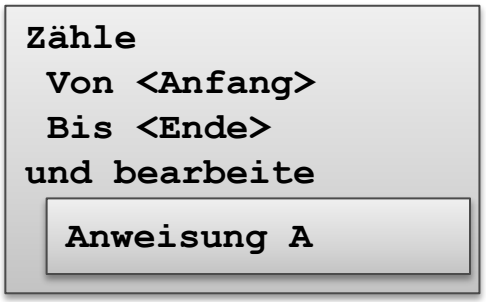
  [...]

- Vorprüfend/Kopfgesteuert

  [...]

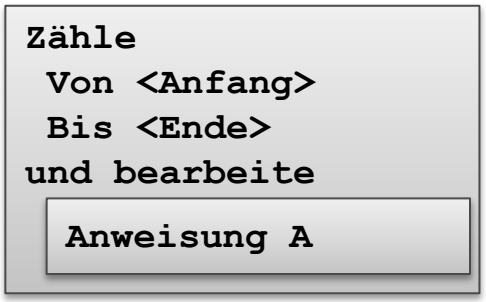
- Nachprüfend/Fußgesteuert

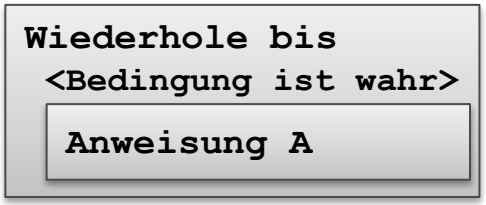
  [...]



Schleifen

- Zählerschleife
  [...]
- Vorprüfend/Kopfgesteuert: Wiederholung solange, bis eine Bedingung wahr/falsch wird
  - Beispiel: Jeder Kunde soll eine Nachricht über das neue Produkt erhalten.
- Nachprüfend/Fußgesteuert
  [...]





Schleifen

- Zählerschleife

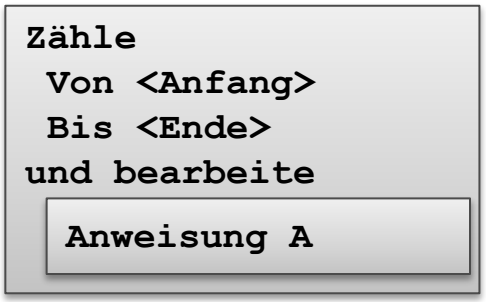
  [...]

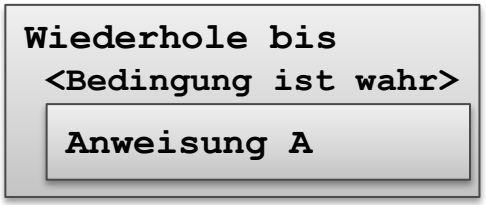
- Vorprüfend/Kopfgesteuert

  [...]

- Nachprüfend/Fußgesteuert

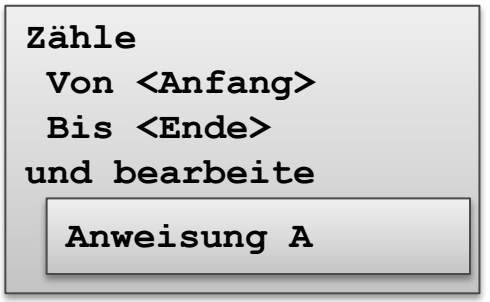
  [...]

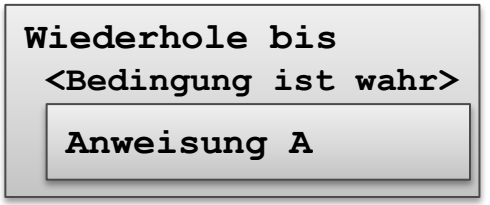


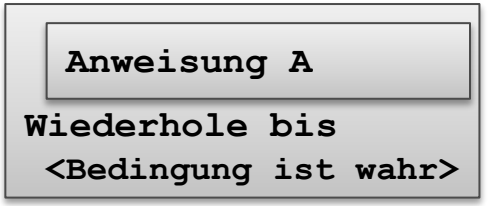


Schleifen

- Zählerschleife
  [...]
- Vorprüfend/Kopfgesteuert
  [...]
- Nachprüfend/Fußgesteuert: Wiederholung solange, bis eine Bedingung wahr/falsch wird, aber mindestens einmal
  - Beispiel: Gesamtpreis aus allen Artikeln der Bestellung ermitteln.







Schleifen

- Zählerschleife

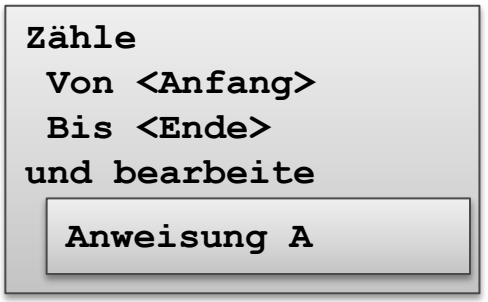
  [...]

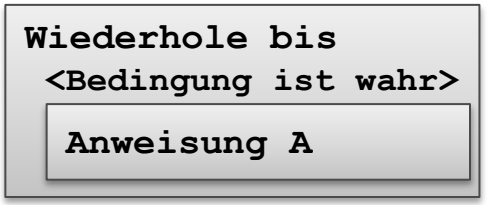
- Vorprüfend/Kopfgesteuert

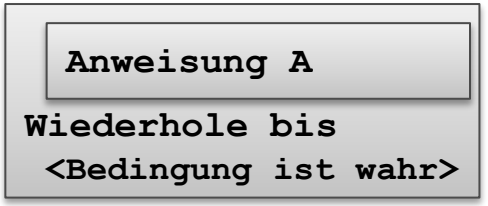
  [...]

- Nachprüfend/Fußgesteuert

  [...]





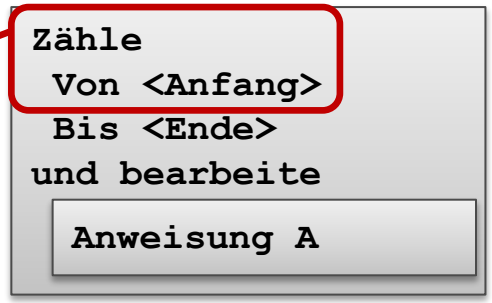


Zählerschleife

Bekannte Anzahl von Durchläufen

Mit Schlüsselworten

- **`For`**
- **`To`**
- **`Next`**
- optional: **`Step`**



Hinweis: Eine zuvor deklarierte Variable wird zum Zählen benötigt.

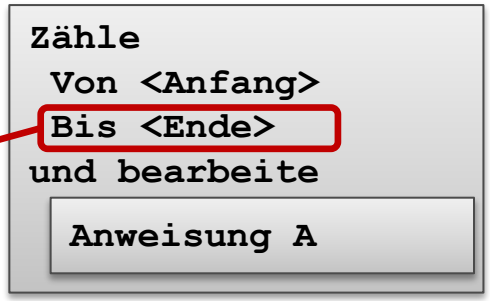
Zählerschleife

Bekannte Anzahl von Durchläufen

Mit Schlüsselworten

- **`For`**
- **`To`**
- **`Next`**
- optional: **`Step`**

Hinweis: Eine zuvor deklarierte Variable wird zum Zählen benötigt.



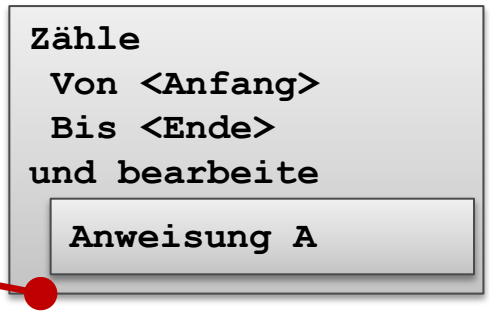
Zählerschleife

Bekannte Anzahl von Durchläufen

Mit Schlüsselworten

- **`For`**
- **`To`**
- **`Next`**
- optional: **`Step`**

Hinweis: Eine zuvor deklarierte Variable wird zum Zählen benötigt.



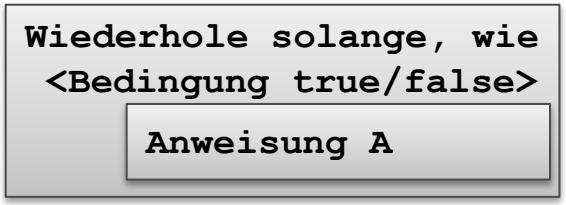
Demo: Zählerschleife

Demo 1.5

- Addieren Sie die Zahlen von 0 bis 5 mit Hilfe einer Zählerschleife
- Deklarieren Sie
  - eine Variable vom Typ Integer für die Summe
  - eine Variable vom Typ Byte als Zählervariable
- Schreiben Sie eine For-Schleife, die von 0 bis 5 läuft
- innerhalb der Schleife
  - soll der Variable Summe der Wert der Variable Summe plus den aktuellen Wert der Zählervariable zugewiesen werden
  - soll der aktuelle Wert der Summe ausgegeben werden

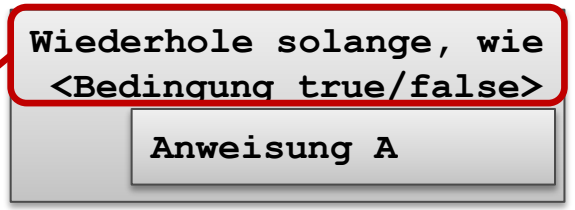
Vorprüfende Schleife

- Vorprüfend bzw. Kopfgesteuert, weil vor der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird
- Mit Schlüsselworten
  - **`Do While`** bzw. **`Do Until`**
  - **`Loop`**



Vorprüfende Schleife

- Vorprüfend bzw. Kopfgesteuert, weil vor der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird
- Mit Schlüsselworten
  - **`Do While`** bzw. **`Do Until`**
  - **`Loop`**

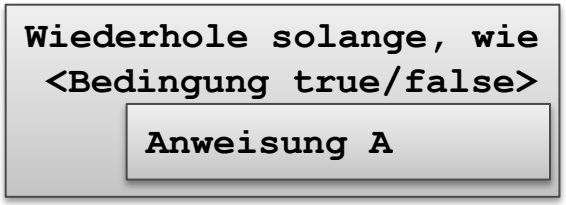


Vorprüfende Schleife

- Vorprüfend bzw. Kopfgesteuert, weil vor der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird
- Mit Schlüsselworten
  - **`Do While`** bzw. **`Do Until`**
  - **`Loop`**

Vorprüfende Schleife

- Vorprüfend bzw. Kopfgesteuert, weil vor der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird
- Mit Schlüsselworten
  - **`Do While`** bzw. **`Do Until`**
  - **`Loop`**



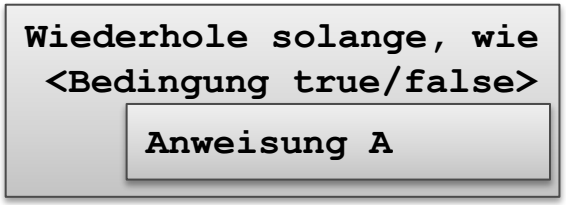
Demo: Kopfgesteuerte, vorprüfende Schleife

Demo 1.6

- Addieren Sie die Zahlen von 2 bis 7 mit Hilfe einer vorprüfenden Schleife
- Deklarieren Sie
  - eine Variable vom Typ Integer für die Summe
  - zwei Variablen vom Typ Integer für Beginn und Ende
- Initialisieren Sie Beginn und Ende
- Schreiben Sie eine Do While-Schleife, die solange läuft, wie der Variablenwert von Beginn kleiner gleich dem Wert von Ende ist
- innerhalb der Schleife
  - soll der Variable Summe der Wert der Variable Summe plus den aktuellen Wert der Beginn-Variable zugewiesen werden
  - soll der Wert der Beginn-Variable erhöht werden
  - soll der aktuelle Wert der Summe ausgegeben werden

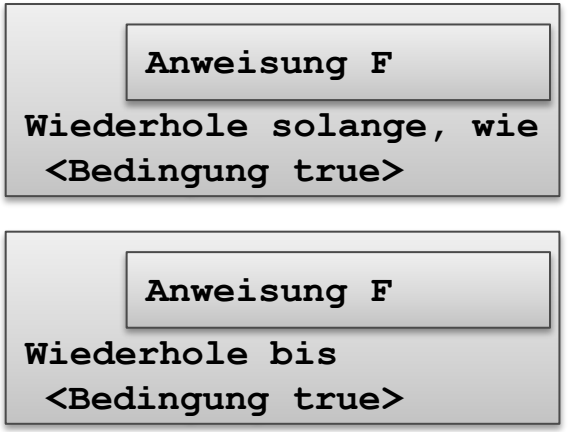
Vorprüfende Schleife

- Vorprüfend bzw. Kopfgesteuert, weil vor der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird
- Mit Schlüsselworten
  - **`Do While`** bzw. **`Do Until`**
  - **`Loop`**



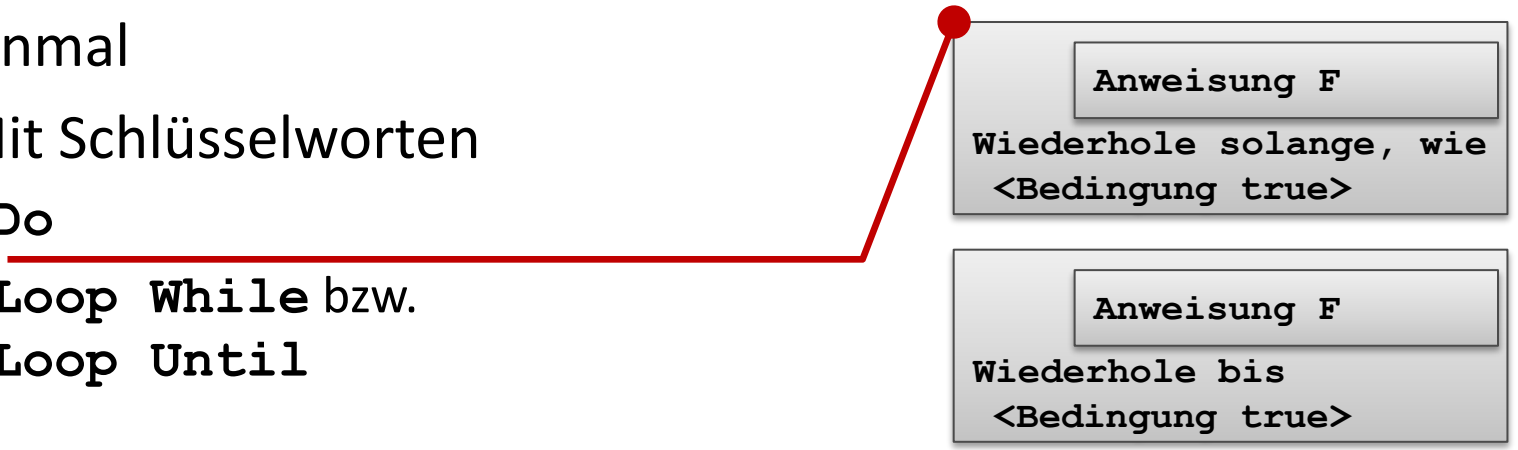
Nachprüfende Schleife

- Nachprüfend bzw. fußgesteuert, weil nach der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird; aber mind. einmal
- Mit Schlüsselworten
  - **`Do`**
  - **`Loop While`** bzw. **`Loop Until`**



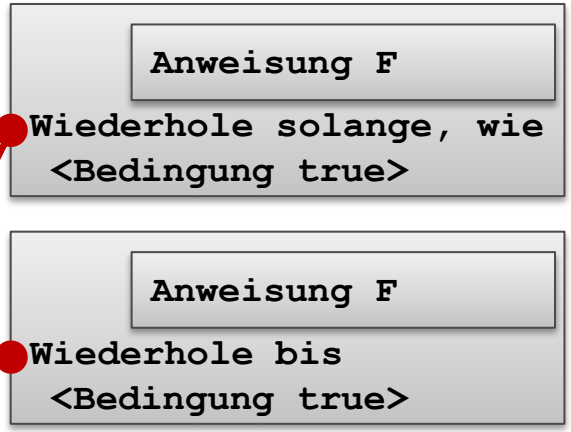
Nachprüfende Schleife

- Nachprüfend bzw. fußgesteuert, weil nach der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird; aber mind. einmal
- Mit Schlüsselworten
  - **`Do`**
  - **`Loop While`** bzw. **`Loop Until`**



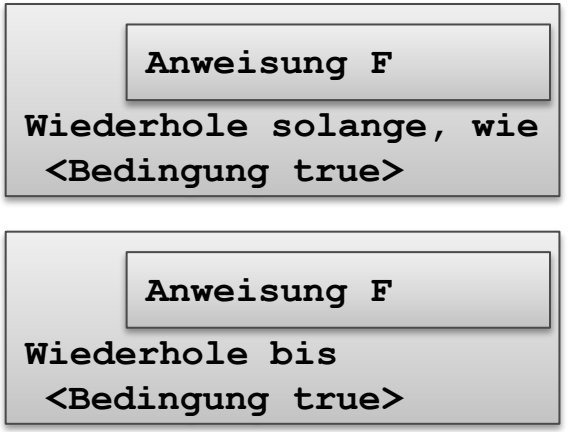
Nachprüfende Schleife

- Nachprüfend bzw. fußgesteuert, weil nach der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird; aber mind. einmal
- Mit Schlüsselworten
  - **`Do`**
  - **`Loop While`** bzw. **`Loop Until`**



Nachprüfende Schleife

- Nachprüfend bzw. fußgesteuert, weil nach der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird; aber mind. einmal
- Mit Schlüsselworten
  - **`Do`**
  - **`Loop While`** bzw. **`Loop Until`**



Demo: Nachprüfende, fußgesteuerte Schleife

Demo 1.7

- Addieren Sie die Zahlen von 2 bis 7 mit Hilfe einer vorprüfenden Schleife
- Deklarieren Sie
  - eine Variable vom Typ Integer für die Summe
  - zwei Variablen vom Typ Integer für Beginn und Ende
- Initialisieren Sie Beginn und Ende
- Schreiben Sie eine Do Loop While-Schleife, die solange läuft, wie der Variablenwert von Beginn kleiner gleich dem Wert von Ende ist
- innerhalb der Schleife
  - soll der Variable Summe der Wert der Variable Summe plus den aktuellen Wert der Beginn-Variable zugewiesen werden
  - soll der Wert der Beginn-Variable erhöht werden
  - soll der aktuelle Wert der Summe ausgegeben werden
- Finden Sie heraus, unter welcher Bedingung sich diese Schleife anders verhält, als die Do While-Schleife

Übung: Schleifen

Aufgabe 1.4

- Schreiben Sie eine Zählerschleife, alle geraden Zahlen im Bereich von 0 bis 20 addiert
- Geben Sie die Summe im Direktbereich aus

Aufgabe 1.5

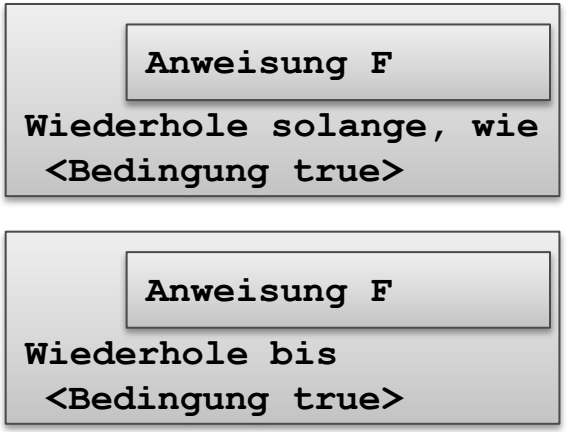
- Implementieren Sie die Aufgabenstellung aus Aufgabe 1.4 mit einer vorprüfenden/kopfgesteuerten Schleife anstelle der Zählerschleife

Aufgabe 1.6

- Passen Sie Ihre Lösung aus Aufgabe 1.5 so an, dass die Schleife rückwärts von 20 bis 0 läuft
- Verwenden Sie eine nachprüfende/fußgesteuerte Schleife

Nachprüfende Schleife

- Nachprüfend bzw. fußgesteuert, weil nach der Anweisung die Bedingung geprüft wird
- Durchlaufen, bis eine Bedingung erfüllt/nicht erfüllt wird; aber mind. einmal
- Mit Schlüsselworten
  - **`Do`**
  - **`Loop While`** bzw. **`Loop Until`**



Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Module, Prozeduren und Funktionen

Modul [...]

Prozedur [...]

Funktion [...]

Module, Prozeduren und Funktionen

Modul

- Strukturierung großer Anwendungen
- umfasst Deklarationen (z.B. Typen, Variablen), Prozeduren und Funktionen als Bestandteile
- kann seine Bestandteile anderen Modulen zur Verfügung stellen
  - Sichtbarkeit von Deklarationen und Prozeduren/Funktionen festlegen
  - Standardmäßig alles sichtbar

Prozedur [...]

Funktion [...]

Online Shop mit
Kundenverwaltung,
Produktkatalog
Bestellungsabwicklung

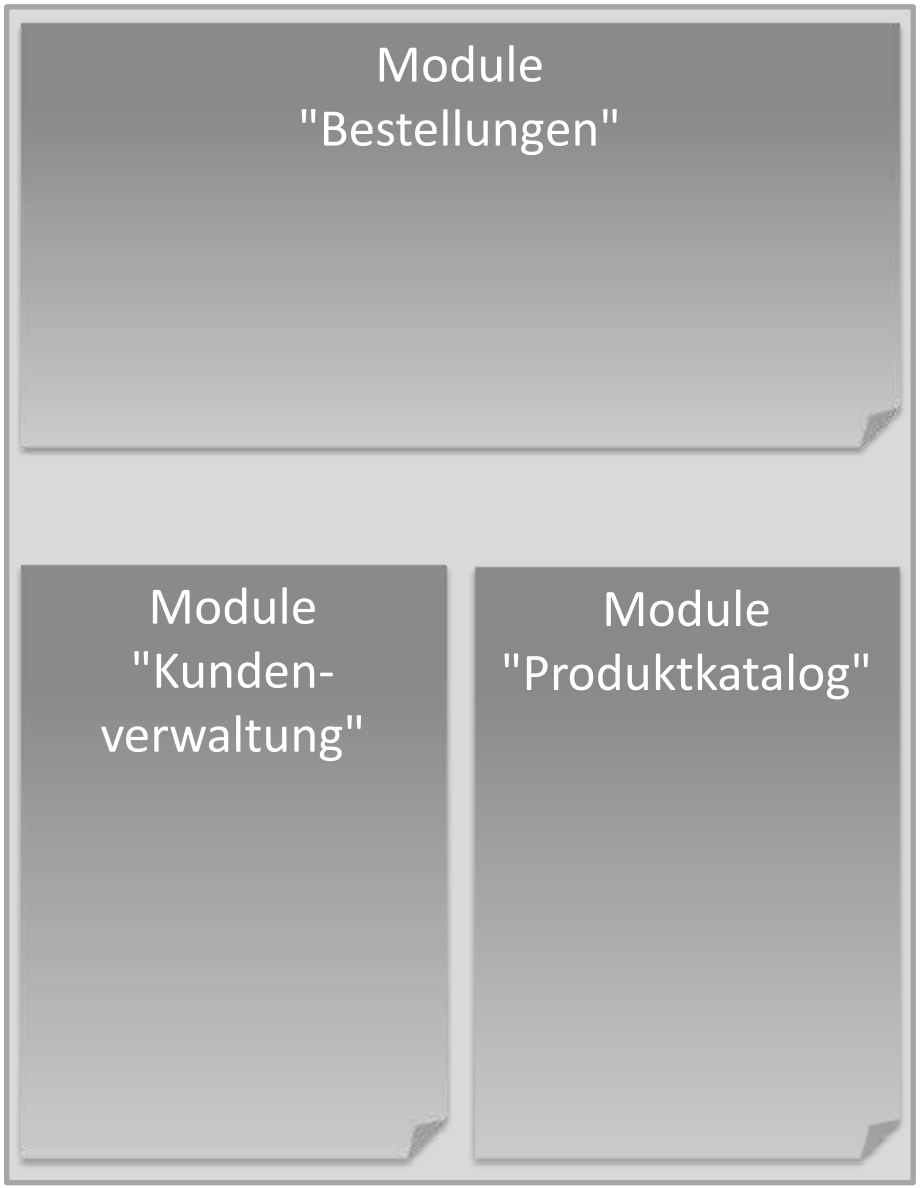
Module, Prozeduren und Funktionen

Modul

- Strukturierung großer Anwendungen
- umfasst Deklarationen (z.B. Typen, Variablen), Prozeduren und Funktionen als Bestandteile
- kann seine Bestandteile anderen Modulen zur Verfügung stellen
  - Sichtbarkeit von Deklarationen und Prozeduren/Funktionen festlegen
  - Standardmäßig alles sichtbar

Prozedur [...]

Funktion [...]



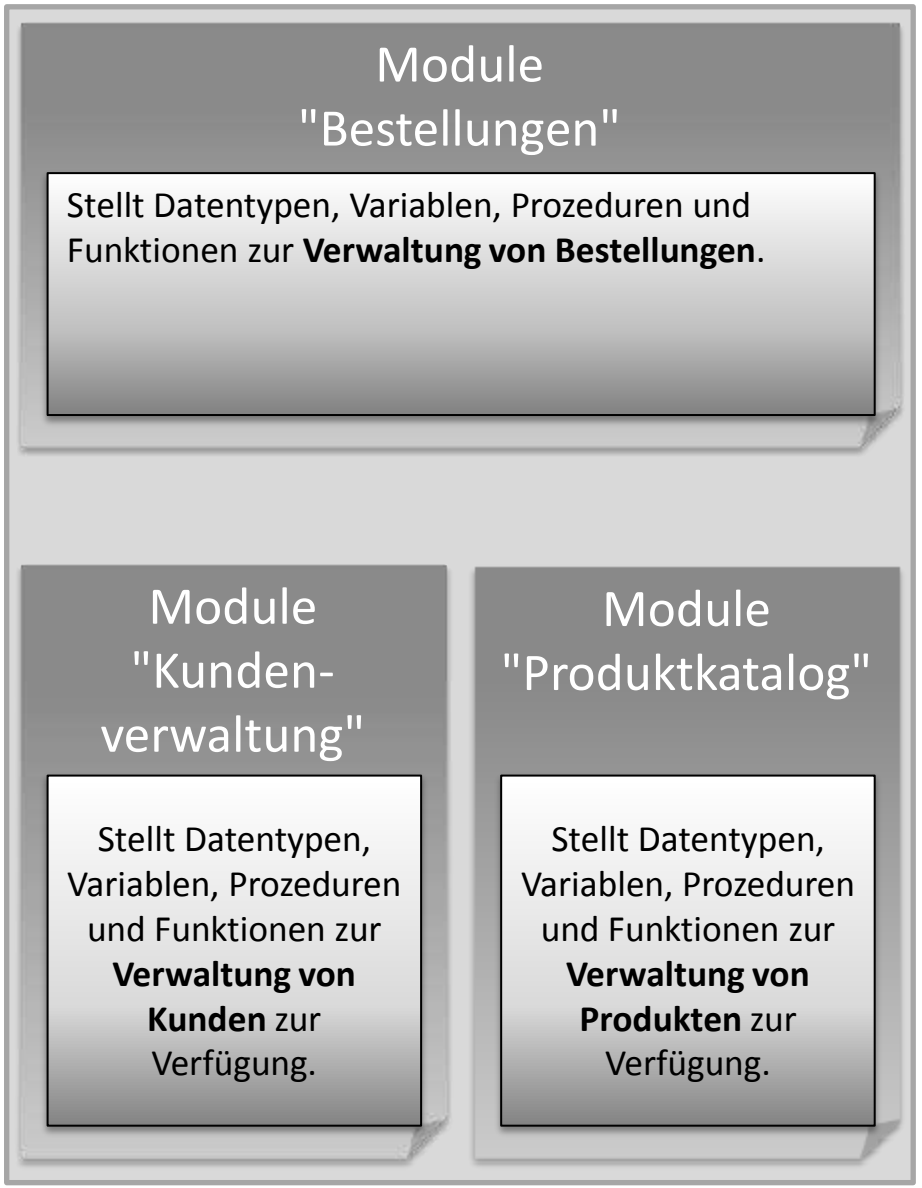
Module, Prozeduren und Funktionen

Modul

- Strukturierung großer Anwendungen
- umfasst Deklarationen (z.B. Typen, Variablen), Prozeduren und Funktionen als Bestandteile
- kann seine Bestandteile anderen Modulen zur Verfügung stellen
  - Sichtbarkeit von Deklarationen und Prozeduren/Funktionen festlegen
  - Standardmäßig alles sichtbar

Prozedur [...]

Funktion [...]



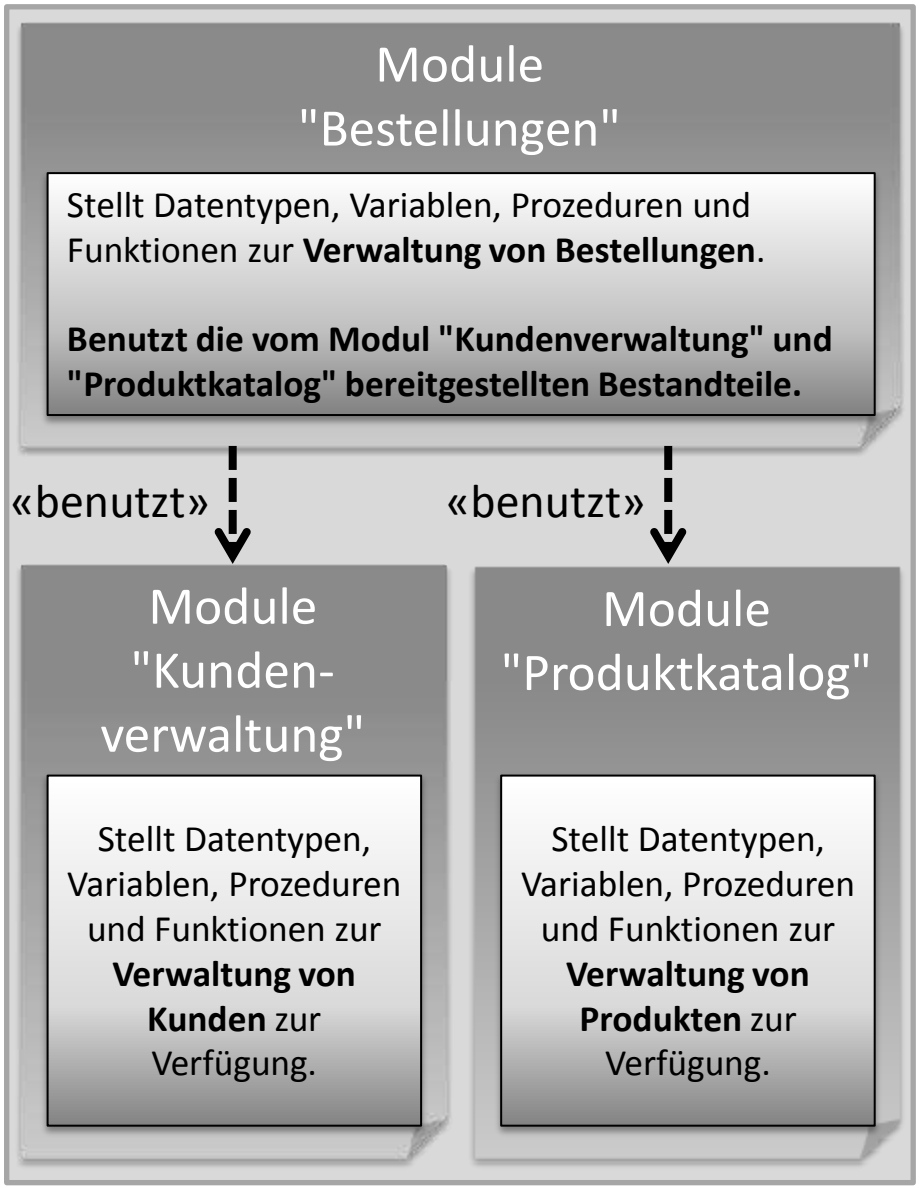
Module, Prozeduren und Funktionen

Modul

- Strukturierung großer Anwendungen
- umfasst Deklarationen (z.B. Typen, Variablen), Prozeduren und Funktionen als Bestandteile
- kann seine Bestandteile anderen Modulen zur Verfügung stellen
  - Sichtbarkeit von Deklarationen und Prozeduren/Funktionen festlegen
  - Standardmäßig alles sichtbar

Prozedur [...]

Funktion [...]

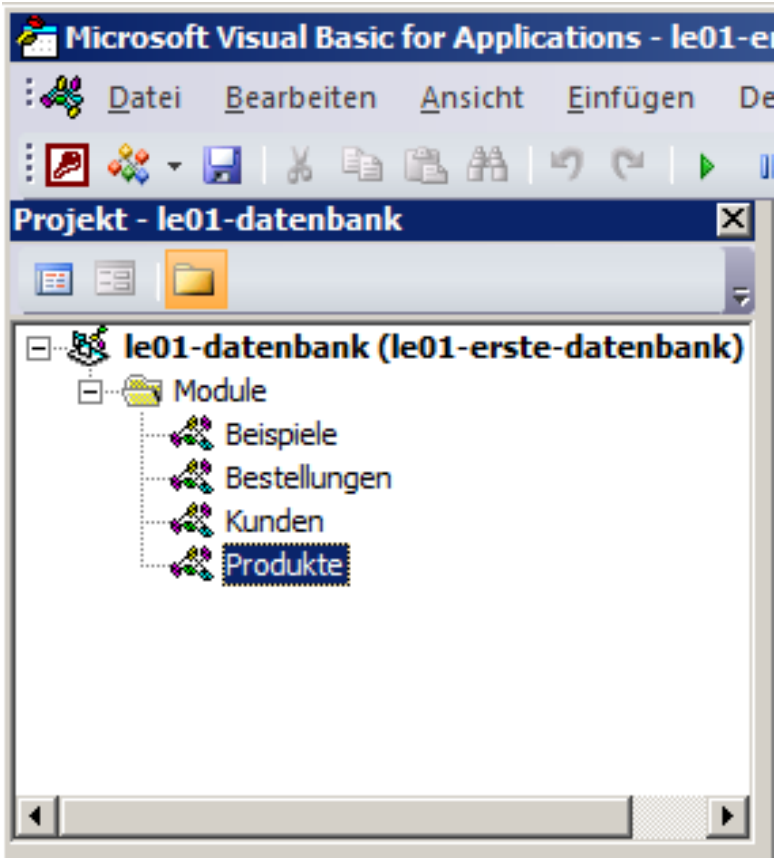


Demo - Module

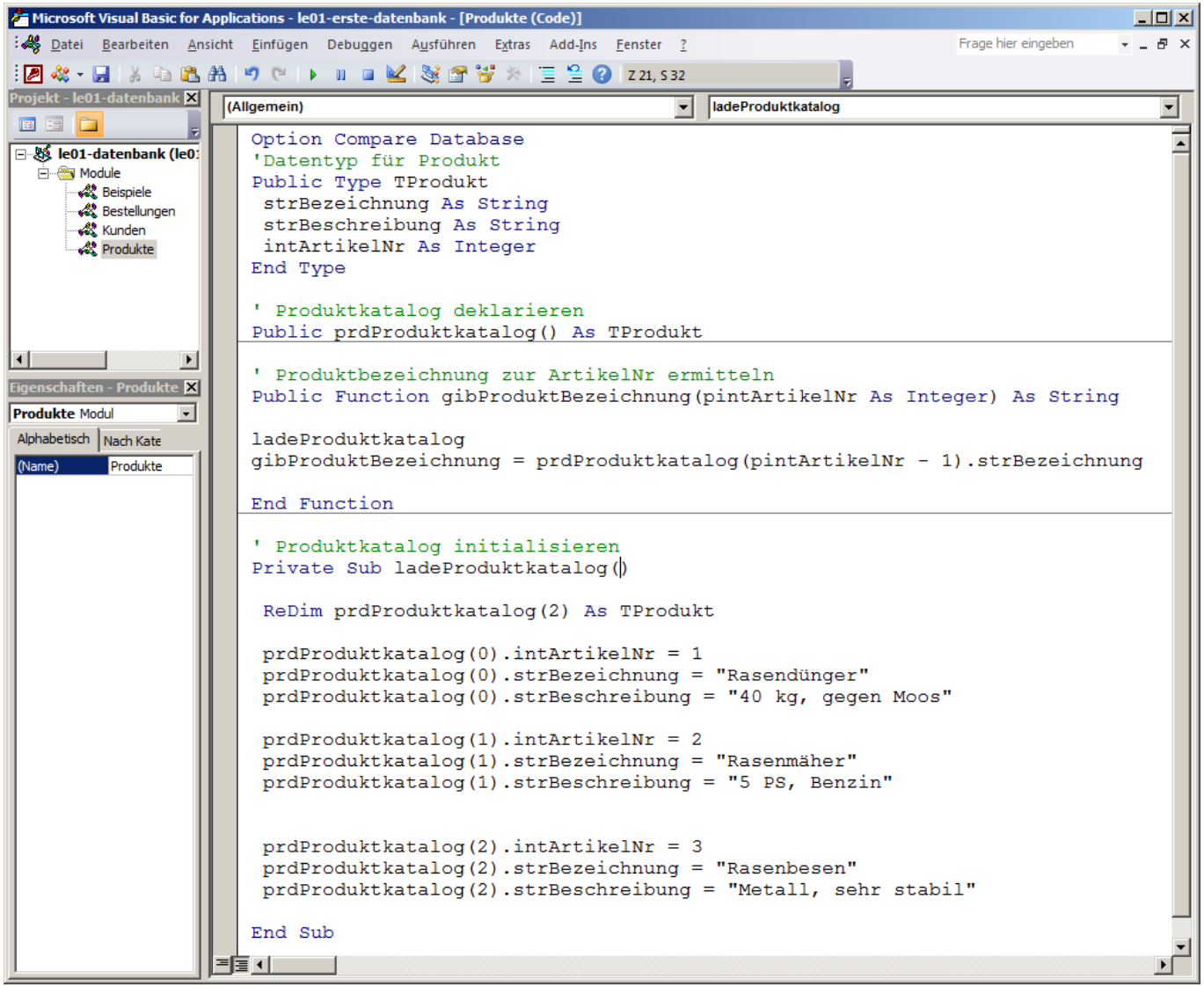
Demo 1.8

- Module in Visual Basic anlegen und bearbeiten
- Beispiel-Module anlegen
  - Kunden
  - Bestellungen
  - Produkte

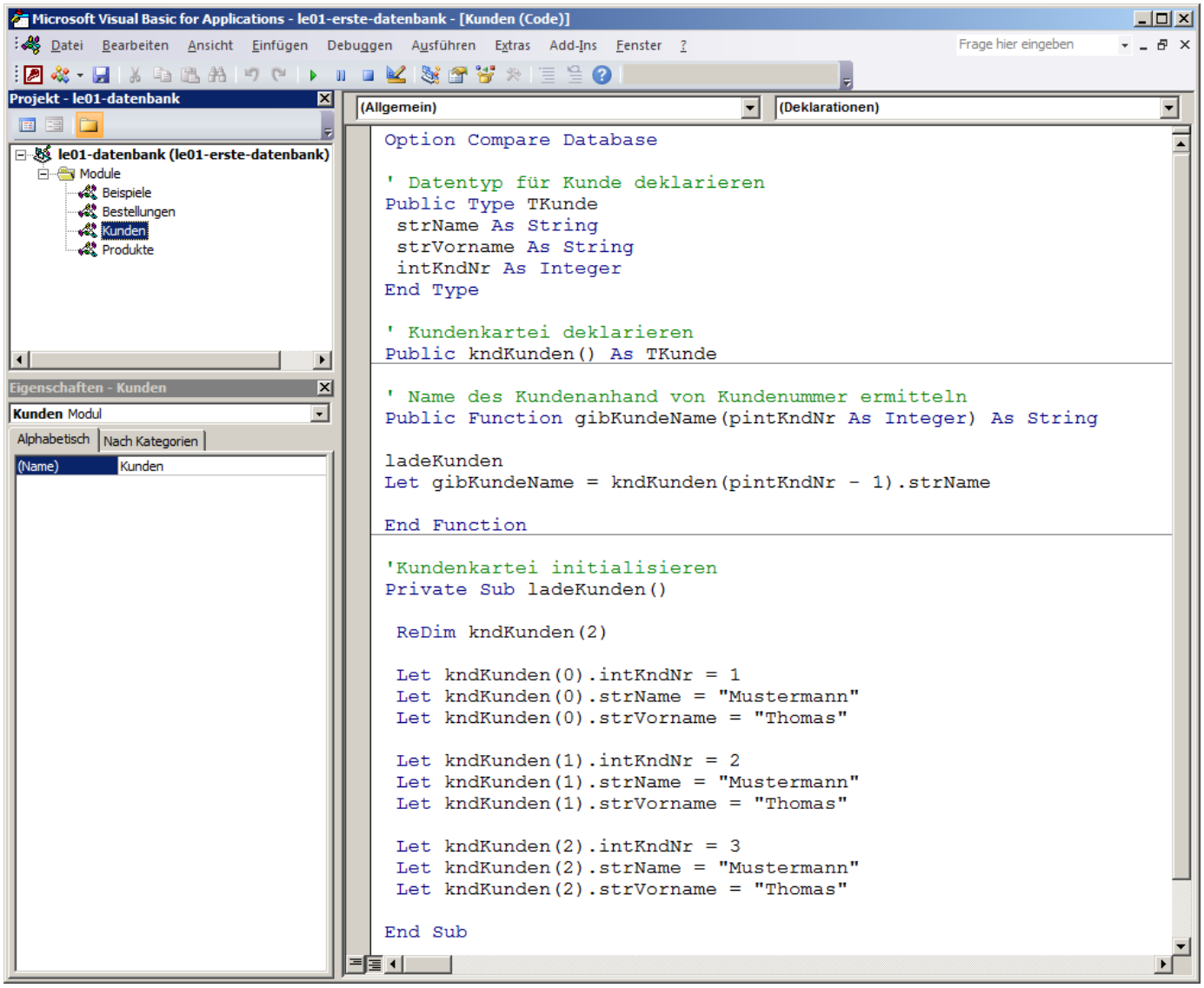
Beispiel


Microsoft Visual Basic for Applications - le01-e
Datei
Bearbeiten
Ansicht
Einfügen
De
Projekt - le01-datenbank
le01-datenbank (le01-erste-datenbank)
Module
Beispiele
Bestellungen
Kunden
Produkte

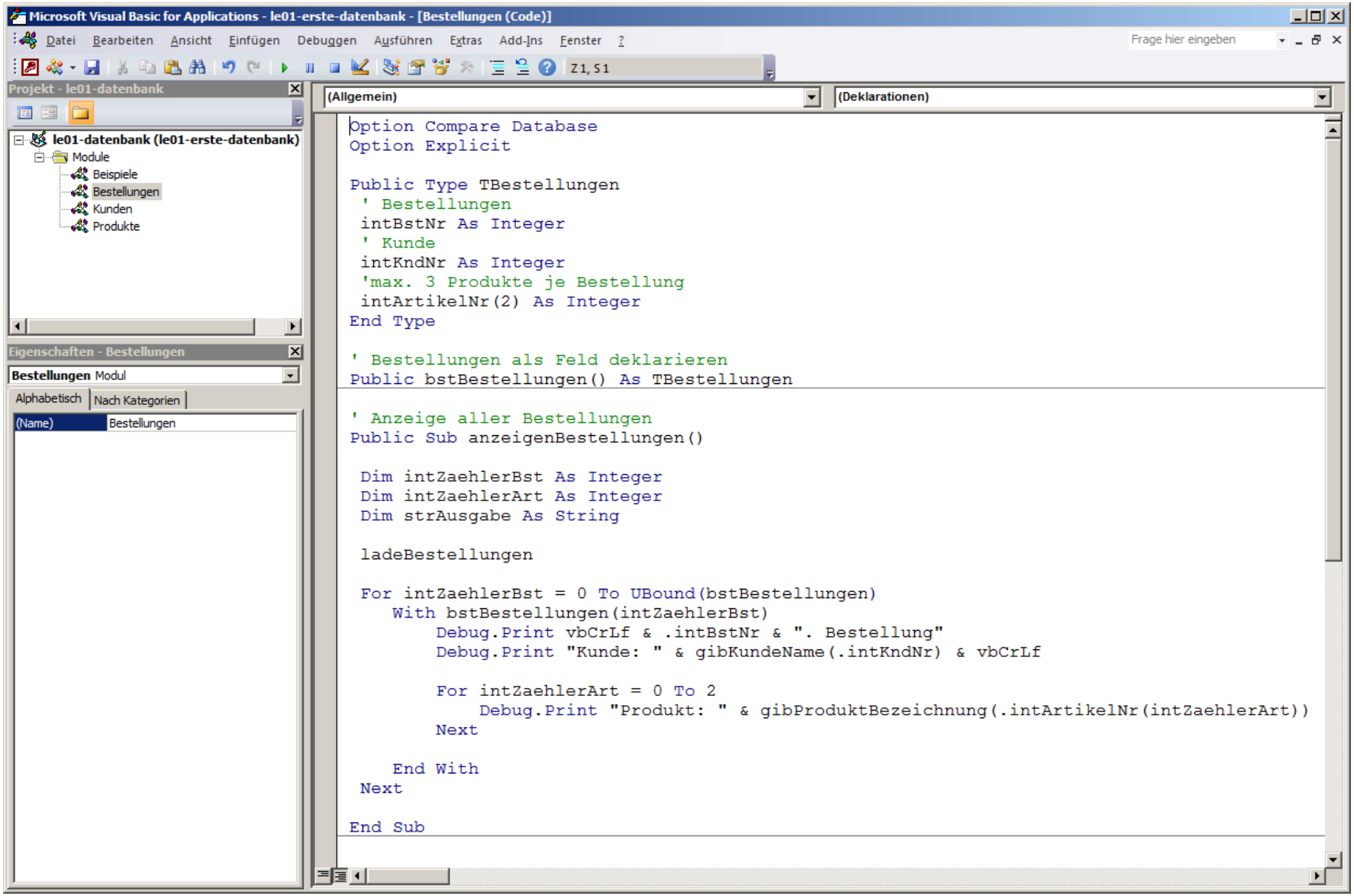
Beispiel: Module - Produkte



Beispiel: Module - Kunden



Beispiel: Module - Bestellungen



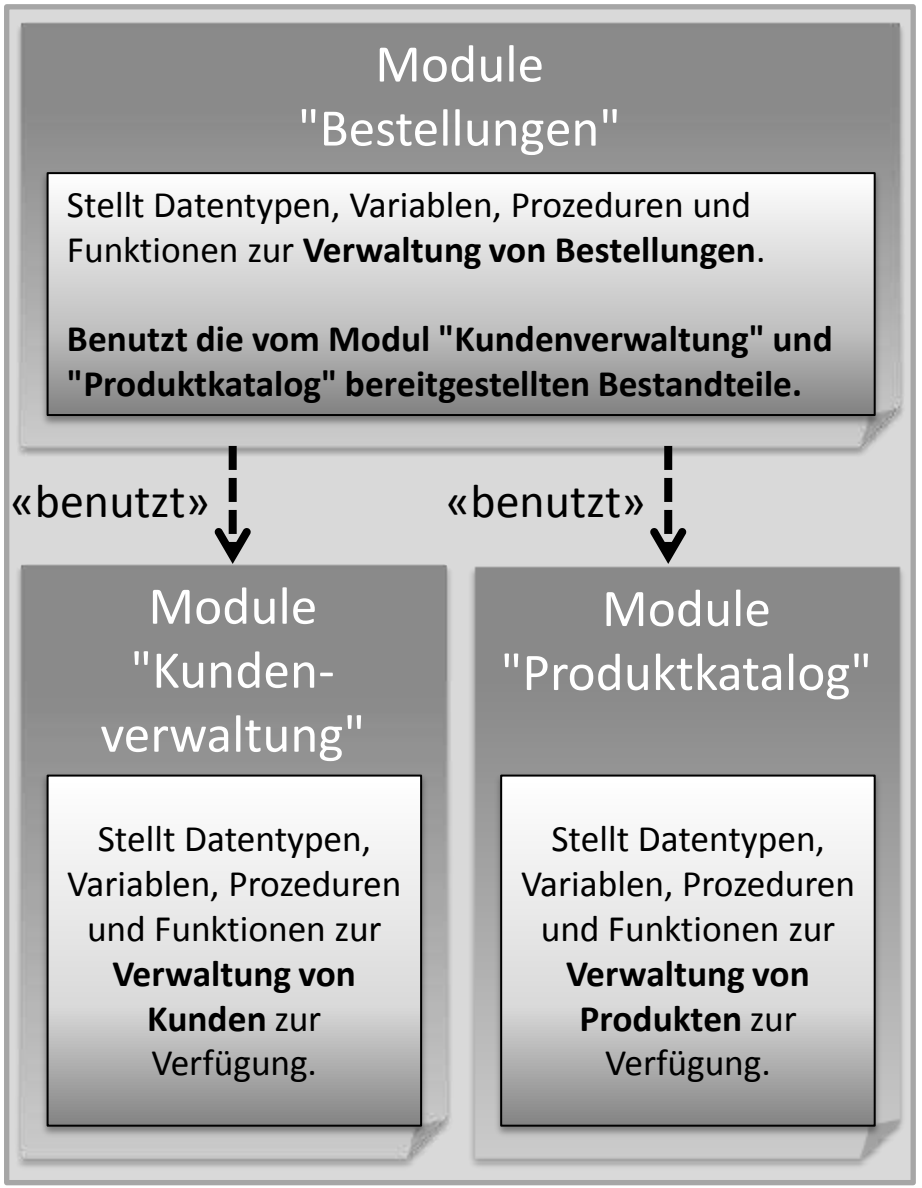
Module, Prozeduren und Funktionen

Modul

- Strukturierung großer Anwendungen
- umfasst Deklarationen (z.B. Typen, Variablen), Prozeduren und Funktionen als Bestandteile
- kann seine Bestandteile anderen Modulen zur Verfügung stellen
  - Sichtbarkeit von Deklarationen und Prozeduren/Funktionen festlegen
  - Standardmäßig alles sichtbar

Prozedur [...]

Funktion [...]



Module, Prozeduren und Funktionen

Modul [...]

Prozedur [...]

Funktion [...]

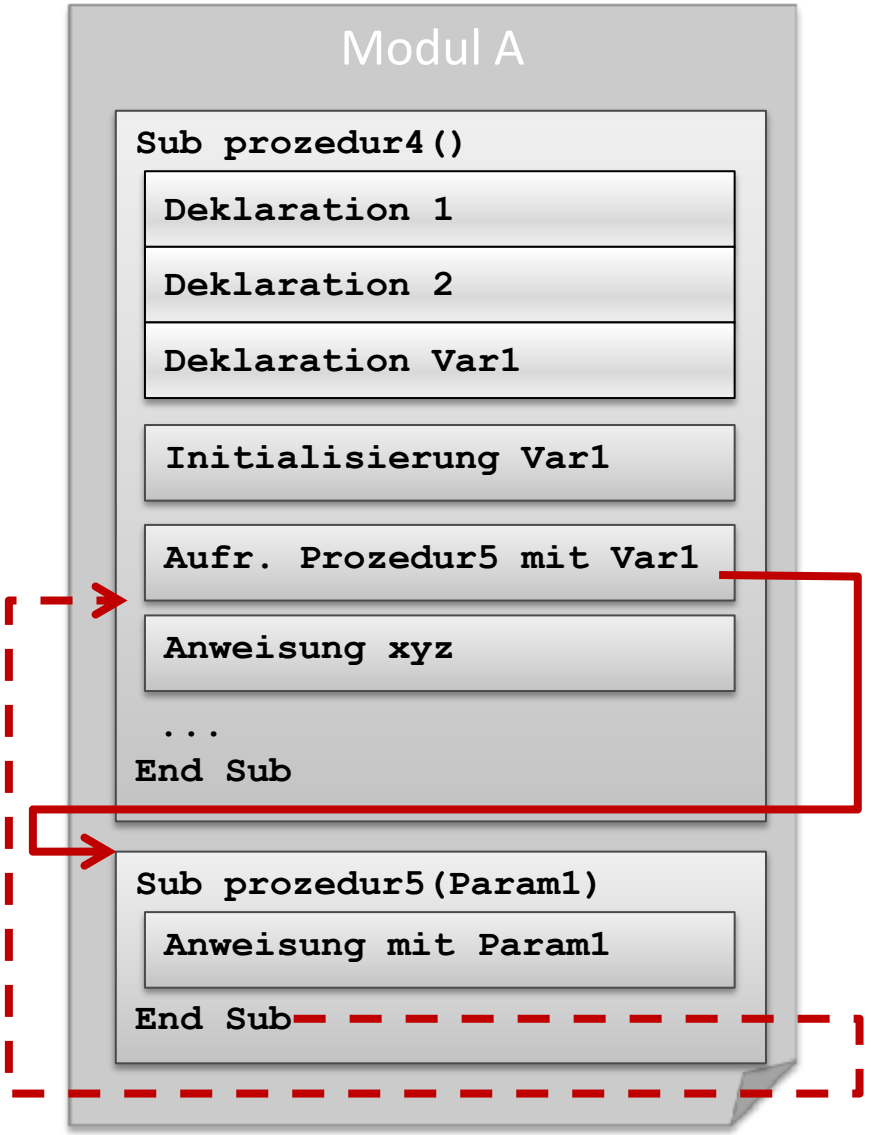
Module, Prozeduren und Funktionen

Modul [...]

Prozedur

- Zusammenfassung von Deklarationen und Anweisungen, die ausgeführt werden
- kann
  - z.B. aus anderen Prozeduren/Funktionen aufgerufen werden
  - andere Prozeduren/Funktionen aufrufen
- Parameter
  - können der Prozedur beim Aufruf übergeben werden
  - im Kopf der aufgerufenen Prozedur (Signatur) deklariert
  - ...
- liefert keinen Ergebniswert zurück

Funktion [...]



Beispiel - Prozeduren

```
Sub Bsp_Prozedur()

 gibAusAddition 2, 3
 gibAusAddition 4, 5

End Sub


Sub gibAusAddition(pintZahl1, pintZahl2)

 Debug.Print pintZahl1 & "+" & pintZahl2 & _
      "=" & pintZahl1 + pintZahl2

End Sub
```

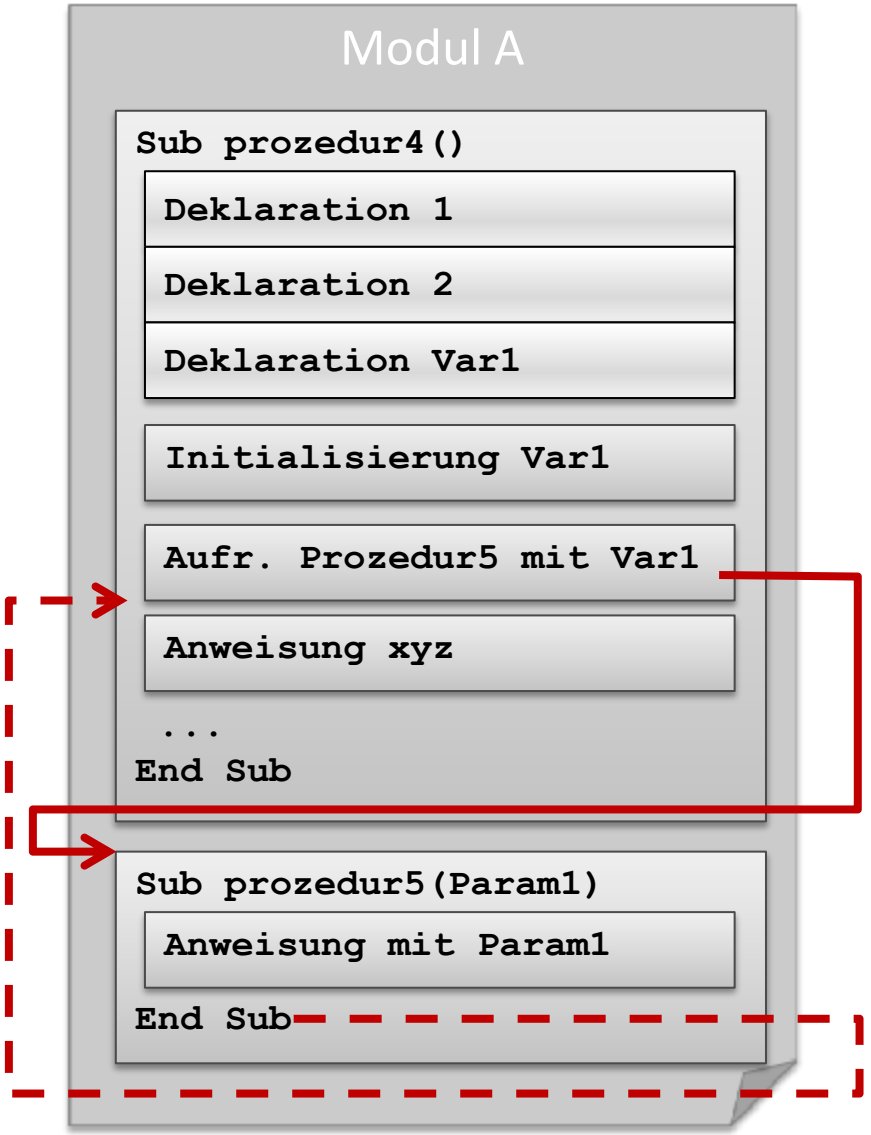
Module, Prozeduren und Funktionen

Modul

Prozedur

- Zusammenfassung von Deklarationen und Anweisungen, die ausgeführt werden
- kann
  - z.B. aus anderen Prozeduren/Funktionen aufgerufen werden
  - andere Prozeduren/Funktionen aufrufen
- Parameter
  - können der Prozedur beim Aufruf übergeben werden
  - im Kopf der aufgerufenen Prozedur (Signatur) deklariert
  - ...
- liefert keinen Ergebniswert zurück

Funktion



Demo - Prozeduren

Demo 1.9

- Konstante im Modul Bestellungen mit Umrechnungskurs (1 US-Dollar = 0,767341928 Euro)
- Prozedur im Modul Bestellungen zur Umrechnung von US-Dollar (USD) in Euro (EUR)
- Prozedur bekommt als Parameter einen Dollarbetrag übergeben und gibt im Direktbereich den Eurobetrag und den verwendeten Umrechnungskurs aus

Übung: Prozedur

Aufgabe 1.7: Schreiben Sie

- in einem Modul Bestellungen
- eine Prozedur, die einen Währungsbetrag als Netto übergeben bekommt,
- die Steuer (19 Prozent) und den Brutto-Betrag errechnet und
- Netto, Steuer und Brutto-Betrag im Direktbereich ausgibt
- Rufen Sie die Prozedur aus einer anderen Prozedur mit verschiedenen Beispielwerten auf

Module, Prozeduren und Funktionen

Modul [...]

Prozedur [...]

Funktion [...]

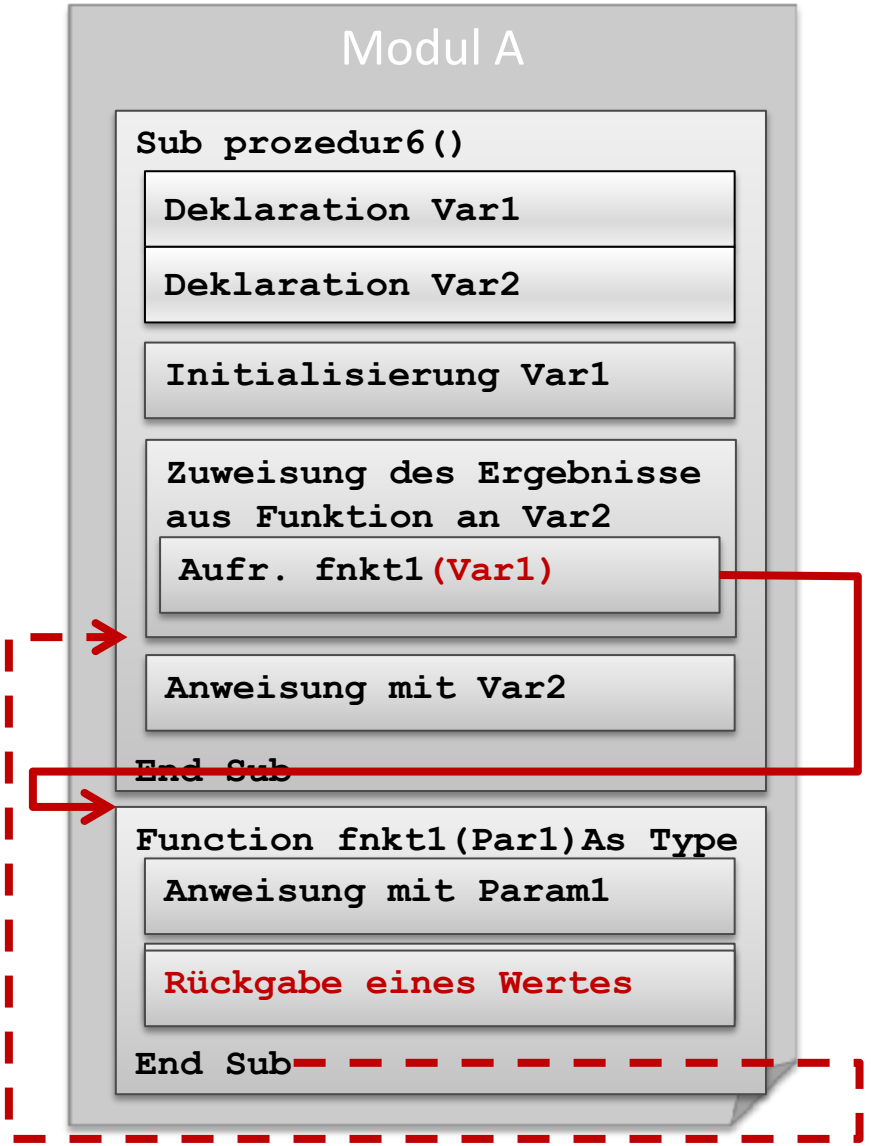
Module, Prozeduren und Funktionen

Modul

Prozedur

Funktion

- Zusammenfassung von Deklarationen und Anweisungen, die ausgeführt werden
- kann
  - z.B. aus anderen Prozeduren/Funktionen aufgerufen werden
  - andere Prozeduren/Funktionen aufrufen
- Parameter in Klammern
  - können der Funktion beim Aufruf übergeben werden
  - im Kopf der aufgerufenen Funktion (Signatur) deklariert
  - …
- liefert einen Ergebniswert zurück



Beispiel - Funktionen

```
Sub Bsp_Funktion()
 Dim intErgebnis As Integer

 Let intErgebnis = addiere(2, 3)
 Debug.Print intErgebnis

 Debug.Print addiere(5, 2)

End Sub

Function addiere(pintZahl1, pintZahl2) As Integer

 Let addiere = pintZahl1 + pintZahl2

End Function
```

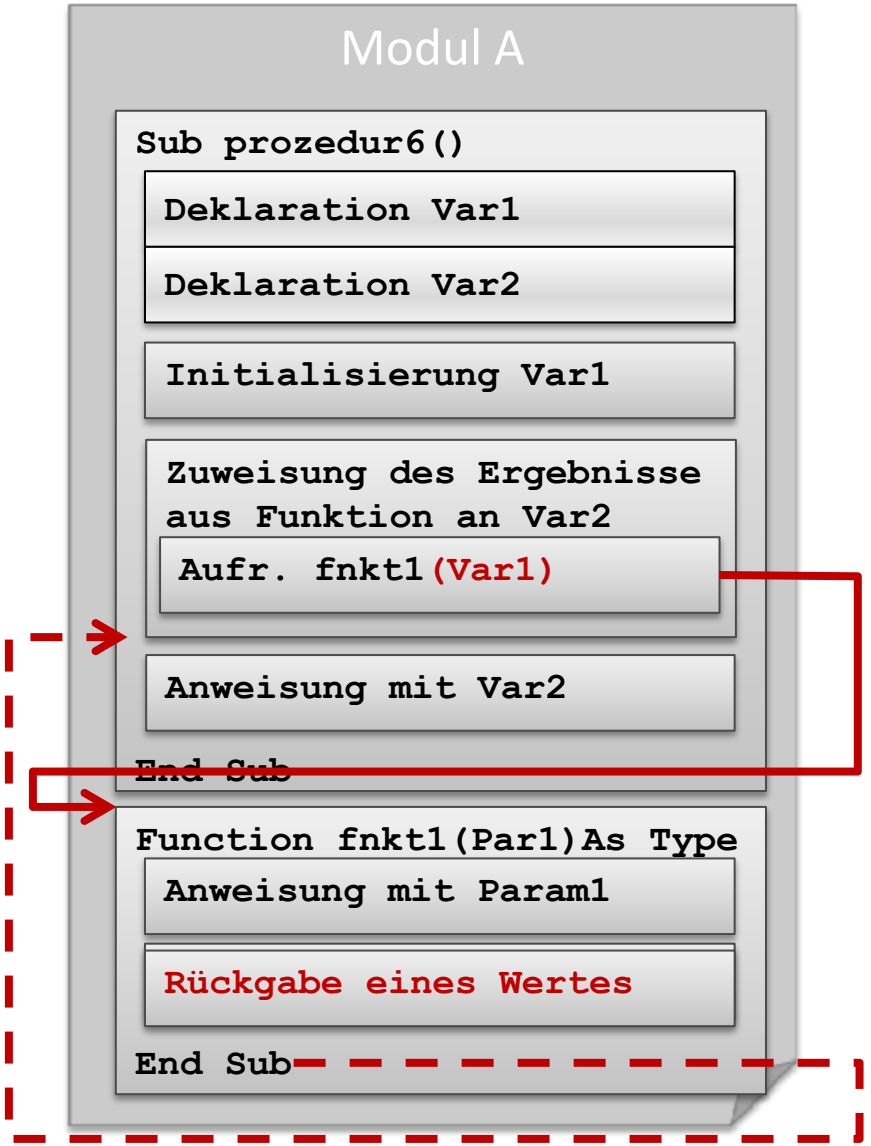
Module, Prozeduren und Funktionen

Modul

Prozedur

Funktion

- Zusammenfassung von Deklarationen und Anweisungen, die ausgeführt werden
- kann
  - z.B. aus anderen Prozeduren/Funktionen aufgerufen werden
  - andere Prozeduren/Funktionen aufrufen
- Parameter in Klammern
  - können der Funktion beim Aufruf übergeben werden
  - im Kopf der aufgerufenen Funktion (Signatur) deklariert
  - …
- liefert einen Ergebniswert zurück



Demo - Funktion

Demo 1.10

- Konstante im Modul Bestellungen mit Umrechnungskurs (1 Euro = 1,3032 US-Dollar)
- Funktion im Modul Bestellungen zur Umrechnung von US-Euro (EUR) in US-Dollar (USD)
- Funktion bekommt als Parameter einen Eurobetrag übergeben, berechnet den Dollarbetrag und gibt ihn als Ergebnis zurück
- In der aufrufenden Prozedur wird das Ergebnis der Funktion im Direktbereich ausgegeben

Übung: Funktion

Aufgabe 1.8: Schreiben Sie

- in einem Modul Bestellungen
- eine Funktion, die einen Währungsbetrag als Netto übergeben bekommt,
- die Steuer (19 Prozent) errechnet und als Ergebnis zurückgibt
- verwenden Sie in der aufrufenden Prozedur dieses Ergebnis, um Netto, Steuer und Brutto-Betrag im Direktbereich auszugeben

Module, Prozeduren und Funktionen

Modul [...]

Prozedur [...]

Funktion [...]

Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Oberflächenelemente

Felder[...]

Schaltflächen[...]

Gliederungselemente[...]

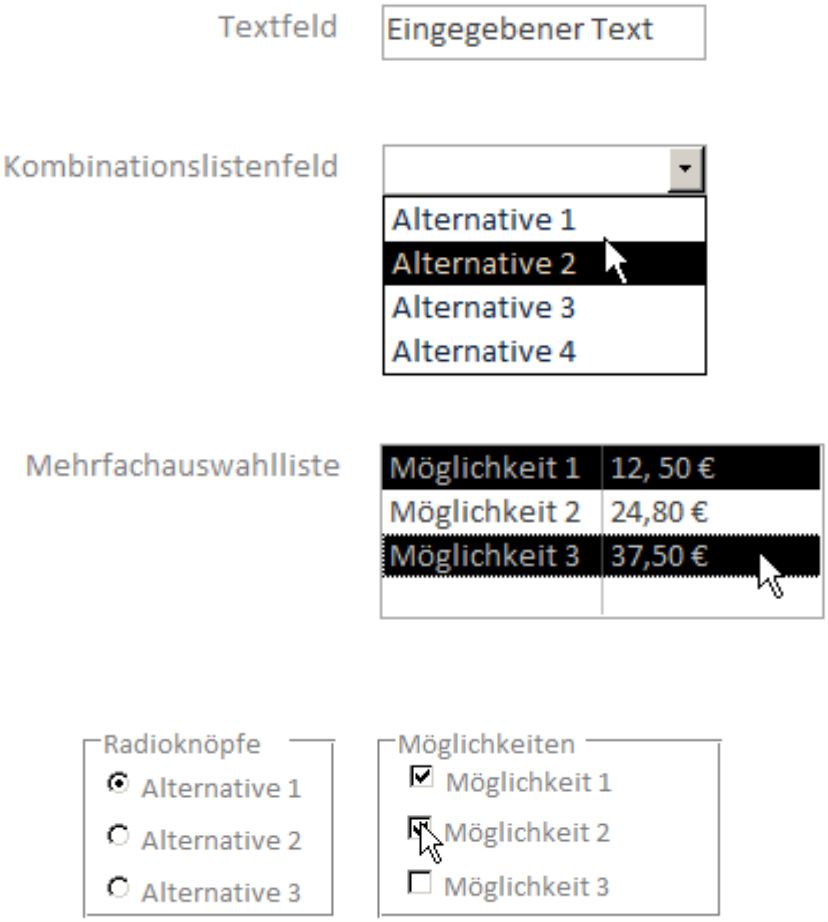
Oberflächenelemente

Felder

- Textfelder
- Aufklappliste/ Kombinationsfeld
- Mehrfachauswahllisten
- Radioknöpfe (Optionsfeld)
  - zusammengefasst in Gruppen (mit Rahmen)
- Kontrollkästchen (Checkbox)
  - häufig zusammengefasst in Gruppen (mit Rechteck anstelle von Rahmen)

Schaltflächen [...]

Gliederungselemente [...]



Oberflächenelemente

Felder[...]

Schaltflächen[...]

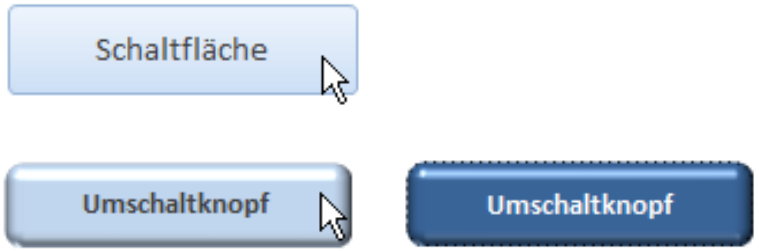
Gliederungselemente[...]

Oberflächenelemente

Felder[...]

Schaltflächen

- Schaltfläche (einfach)
- Schaltfläche (Umschaltknopf/Troggle)



Gliederungselemente[...]

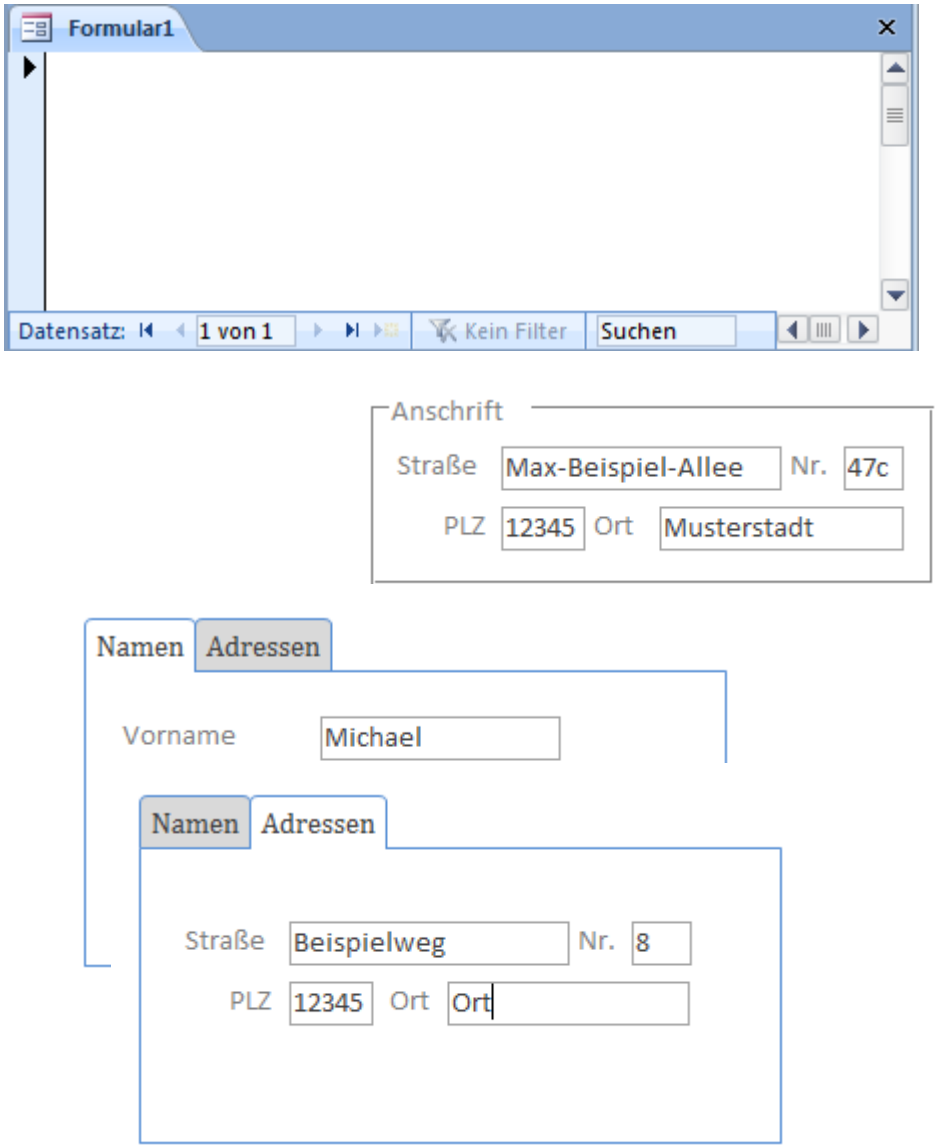
Oberflächenelemente

Felder[...]

Schaltflächen[...]

Gliederungselemente

– Formular, in der Regel mit
  - Inhalt
  - Datensatzmarkierer
  - Navigationsleiste
– Rahmen/Gruppen
– Registerkartensatz mit Registerkarten



Oberflächenelemente

Felder[...]

Schaltflächen[...]

Gliederungselemente[...]

Oberflächenelemente

Vorschläge für Namenskonventionen

Element	Namenskonvention	Beispiel
Textfeld	txt	txtName
Aufklappliste (Combo)	cmb	cmbAnrede
Liste (Mehrfachauswahl)	lst	lstProduktart
Radioknöpfe (Option)	opt	optAdresstyp
Ankreuzfelder (Checkbox)	chk	chkIstGültig
Gruppen	grp	grpAdresse
Registerkarte (Tab)	tab	tabName
Formular	frm	frmPersonen
...		

Oberflächenelemente

Felder[...]

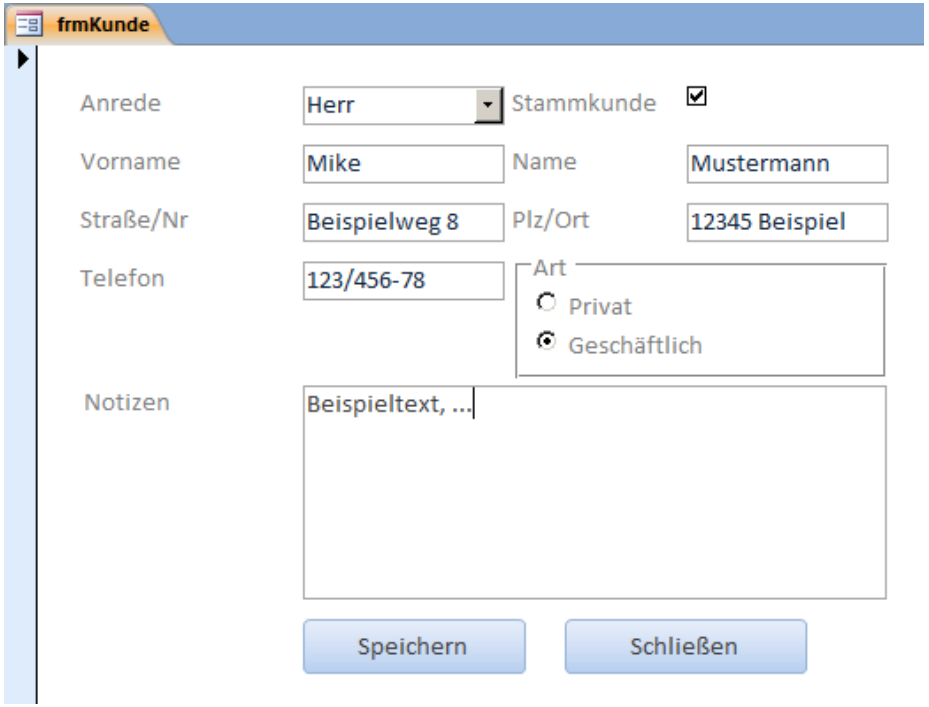
Schaltflächen[...]

Gliederungselemente[...]

Demo: Oberflächenelemente

Demo 1.11

- Formular zur Erfassung von Personenangaben mit
  - Name, Vorname als Textfeld (einzeilig)
  - Ankreuzfeld mit Kennzeichen ob Stammkunde
  - Aufklappliste für Anrede (Herr, Frau, Familie)
  - Straße, Hausnummer als Textfeld
  - PLZ und Ort als Textfeld
  - Telefonnummer
  - Radioknöpfe, ob Telefon privat oder geschäftlich
  - Notizen als mehrzeiliges Textfeld
  - Schaltflächen



Übung: Formulare und Oberflächenelemente

Aufgabe 1.9

– Erstellen Sie das folgende Formular eines kleinen Taschenrechners aus geeigneten Oberflächenelementen

Oberflächenelemente

Felder[...]

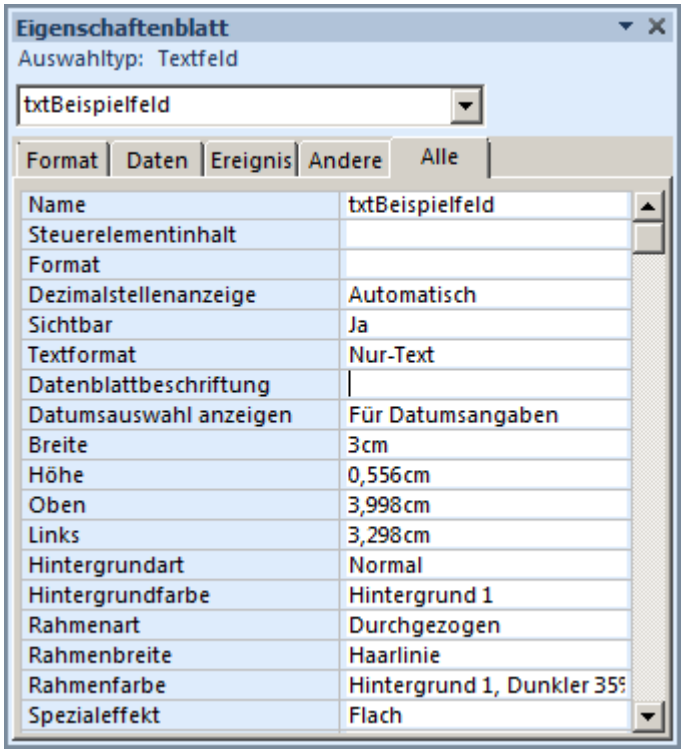
Schaltflächen[...]

Gliederungselemente[...]

Eigenschaften von Oberflächenelemente

Oberflächenelemente haben Eigenschaften,

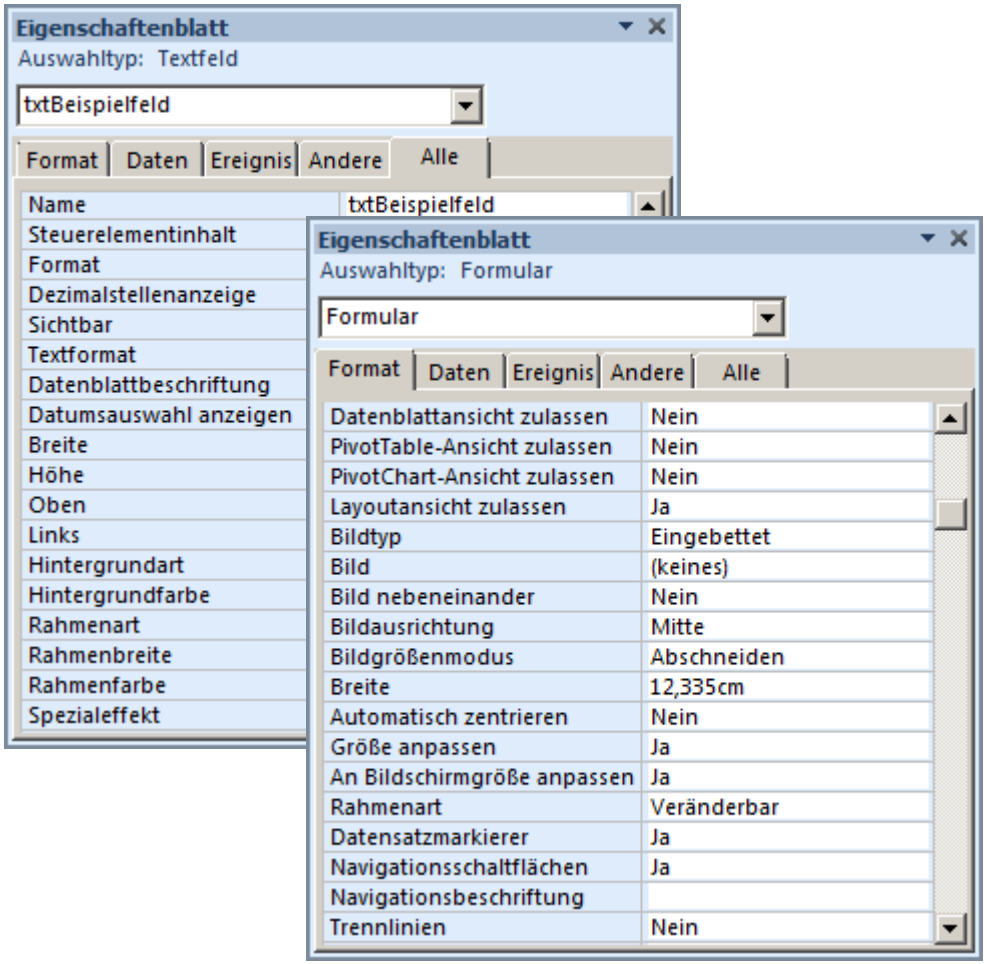
- die im Eigenschaftenblatt bearbeitet werden können, z.B.
  - Textfeld
    - Bezeichnung
    - Sichtbarkeit
    - Änderbarkeit
    - Position
    - aktueller Wert
    - ...
  - Formular
    - Bezeichnung
    - Sichtbarkeit
    - ...



Eigenschaften von Oberflächenelemente

Oberflächenelemente haben Eigenschaften,

- die im Eigenschaftenblatt bearbeitet werden können, z.B.
  - Textfeld
    - Bezeichnung
    - Sichtbarkeit
    - Änderbarkeit
    - Position
    - aktuellen Wert
    - ...
  - Formular
    - Bezeichnung
    - Sichtbarkeit
    - ...



Eigenschaften von Oberflächenelemente

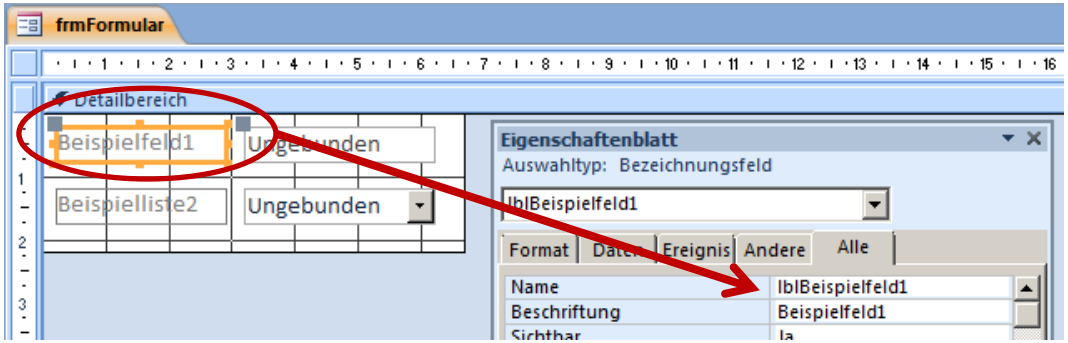
Oberflächenelemente haben Eigenschaften

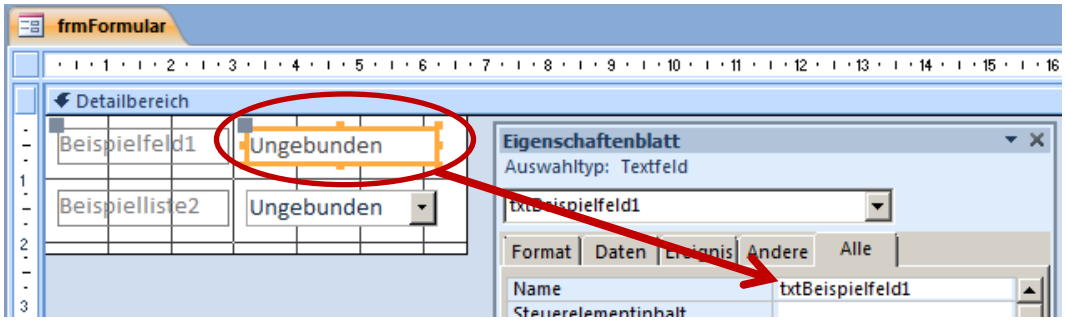
- die im Eigenschaftenblatt bearbeitet werden können
- und auch durch Programmierung verändert werden können

Eigenschaften von Oberflächenelemente

Beispiel 1 - Ausgangssituation

- Textfeld "Beispielfeld1" hat
  - Bezeichnung (links) mit Namen "lblBeispielfeld1"
  - Feld (rechts) mit Namen "txtBeispielfeld1"





Eigenschaften von Oberflächenelemente

Beispiel 1 - Ziel

- Textfeld "Beispielfeld1" soll durch Programmierung den Text "Hallo Welt" als Wert bekommen

Eigenschaften von Oberflächenelemente

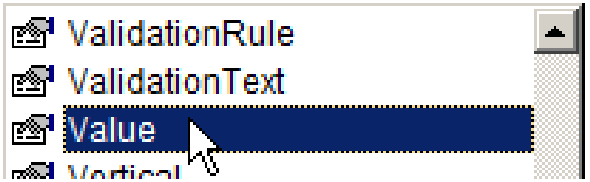
Beispiel 1 - Umsetzung

- Nutzung von Me (Referenz auf aktuelles Fenster)

```
Me.tx
```

- gefolgt vom Namen des Feldes "txtBezeichungsfeld1"

```
Me.txtBeispielfeld1.V
```



- gefolgt von der Eigenschaft, auf die Zugegriffen werden soll (hier Value)

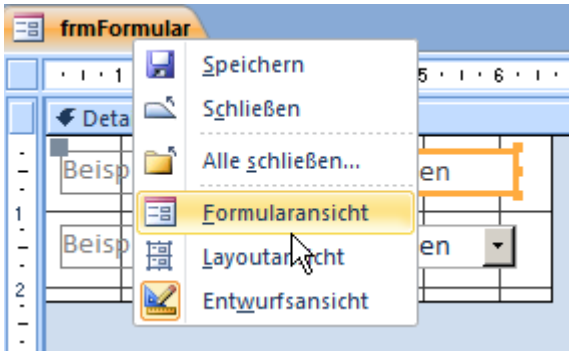
```
Me.txtBeispielfeld1.Value = "Hallo Welt!"
```

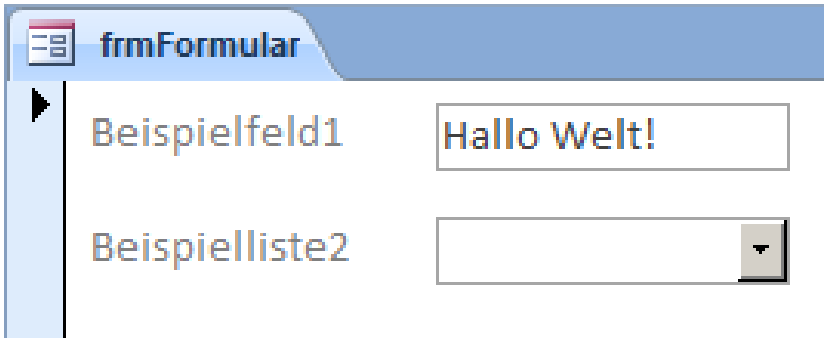
- jeweils getrennt durch Punkt "."

Eigenschaften von Oberflächenelemente

Beispiel 1 - Ergebnis

– Wird das Formular von der Entwurfsansicht in die Formularansicht geschaltet, steht der Text "Hallo Welt" im Textfeld



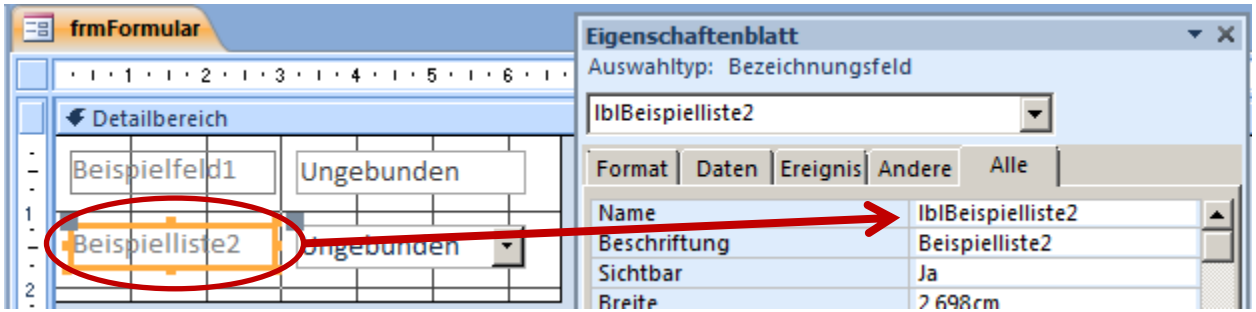


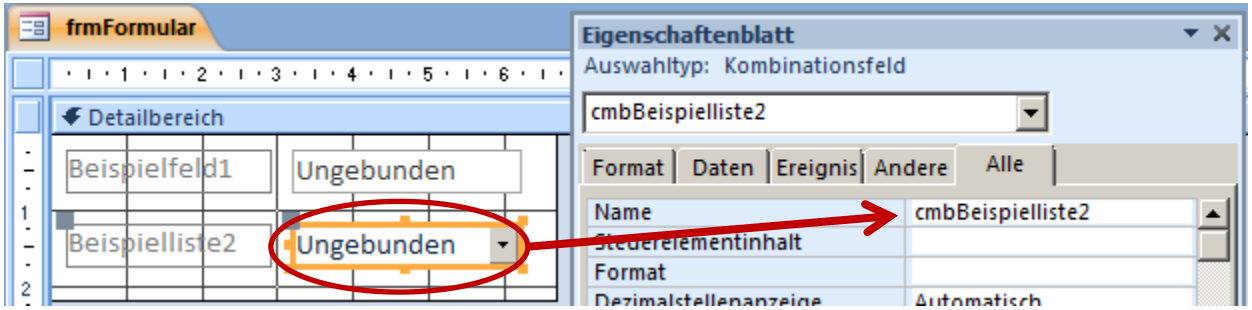
Hinweis: Details, wie auf das Umschalten reagiert werden kann, folgen auf den nächsten Folien.

Eigenschaften von Oberflächenelemente

Beispiel 2 - Ausgangssituation

- Textfeld "Beispielliste2" hat
  - Bezeichnung (links) mit Namen "lblBeispielliste2"
  - Liste (rechts) mit Namen "cmbBeispielliste2"





Eigenschaften von Oberflächenelemente

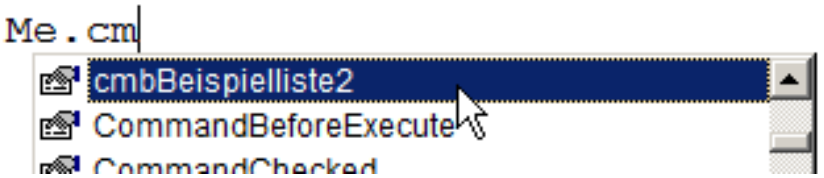
Beispiel 2 - Ziel

- Liste "Beispielliste" (rechts) und Bezeichnung (links) soll durch Programmierung unsichtbar werden

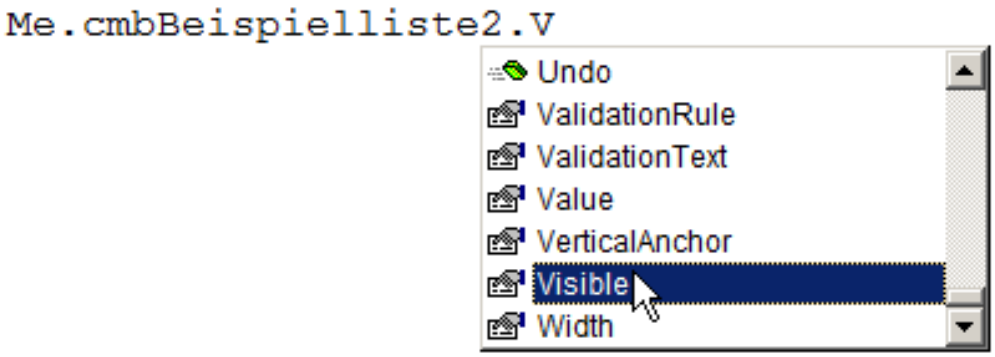
Eigenschaften von Oberflächenelemente

Beispiel 2 - Umsetzung

– Nutzung von Me (Referenz auf aktuelles Fenster)



– gefolgt vom Namen des Feldes "cmbBeispielliste2" und



– der Eigenschaft, auf die Zugegriffen werden soll (hier Visible)

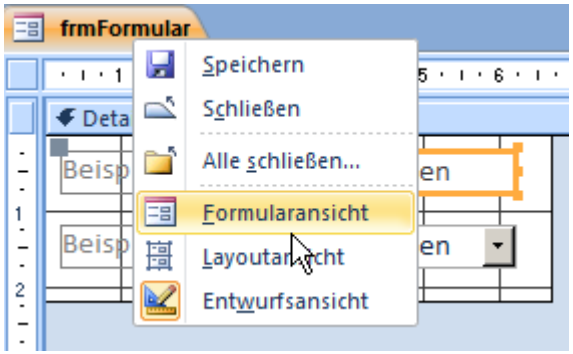
```
Me.cmbBeispielliste2.Visible = False
```

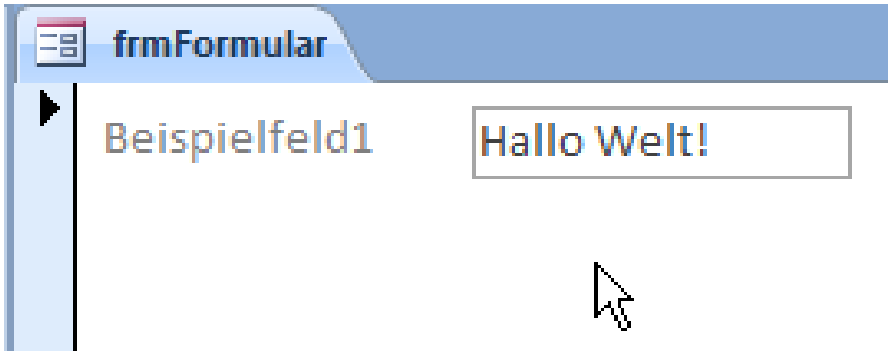
– jeweils getrennt durch Punkt "."

Eigenschaften von Oberflächenelemente

Beispiel 2 - Ergebnis

– Wird das Formular von der Entwurfsansicht in die Formularansicht geschaltet, ist die Aufklappliste unsichtbar





Hinweis: Details, wie auf das Umschalten reagiert werden kann, folgen auf den nächsten Folien.

Eigenschaften von Oberflächenelemente

Oberflächenelemente haben Eigenschaften

- die im Eigenschaftenblatt bearbeitet werden können
- und auch durch Programmierung verändert werden können

Eigenschaften von Oberflächenelemente

Oberflächenelemente haben Eigenschaften

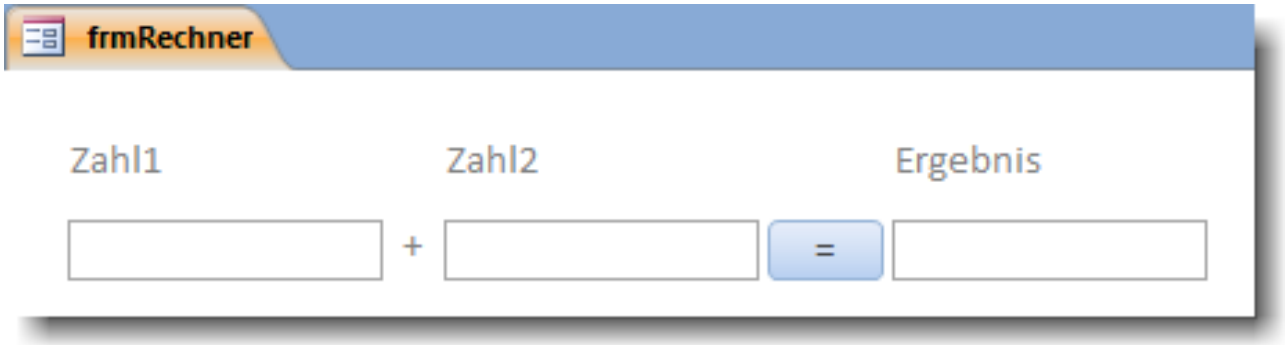
- die im Eigenschaftenblatt bearbeitet werden können
- und auch durch Programmierung verändert werden können
  - Nutzung der Referenzvariable "Me" (aktuelles Formular)
  - Punktnotation
    - zum Zugriff auf Elemente des Formulars (über deren aussagekräftige Namen)
    - zum Zugriff auf Eigenschaften der Elemente
  - Beispiele

```
' Wertzuweisung
Me.txtBeispielfeld1.Value = "Hallo Welt!"
' Deaktivieren eines Feldes
Me.txtBeispielfeld1.Enabled = False
' Liste unsichtbar machen
Me.cmbBeispielliste2.Visible = False
```

Übung: Oberflächenelement Eigenschaften

Aufgabe 1.10

- Setzen Sie das Feld "Ergebnis" des Taschenrechner-Formulars aus der vorherigen Aufgabe 1.9 auf inaktiv (gesperrt), so dass ein dort eingegebener Wert nicht verändert werden kann
- Entfernen Sie im Formular den Datensatznavigierer und die Navigationsschaltflächen
- Beispiel:



Eigenschaften von Oberflächenelemente

Oberflächenelemente haben Eigenschaften

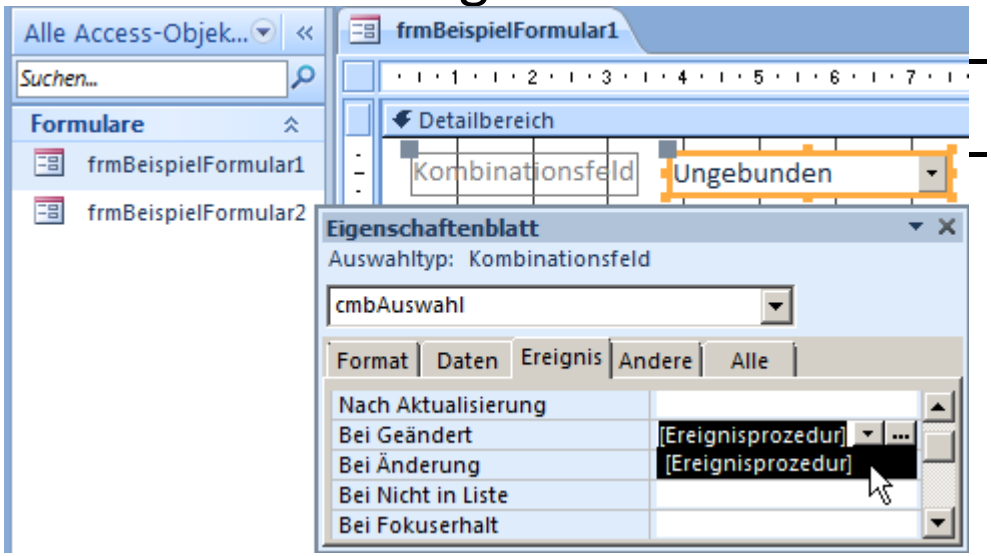
- die im Eigenschaftenblatt bearbeitet werden können
- und auch durch Programmierung verändert werden können
  - Nutzung der Referenzvariable "Me" (aktuelles Formular)
  - Punktnotation
    - zum Zugriff auf Elemente des Formulars (über deren aussagekräftige Namen)
    - zum Zugriff auf Eigenschaften der Elemente
  - Beispiele

```
' Wertzuweisung
Me.txtBeispielfeld1.Value = "Hallo Welt!"
' Deaktivieren eines Feldes
Me.txtBeispielfeld1.Enabled = False
' Liste unsichtbar machen
Me.cmbBeispielliste2.Visible = False
```

Ereignisse als Reaktion auf Benutzeraktionen

Benutzer führt Aktionen auf der Oberfläche aus

- Klicken auf Schaltflächen
- Eingeben von Daten in Felder
- Auswahl von Listeneinträgen
- Verlassen von Feldern
- Mouse bewegen



Als Reaktion werden Ereignisprozeduren aufgerufen

- btnSchalflaeche**_Click()**
- txtTextfeld**_KeyPress()**
- cmbAuswahl**_Change()**
- txtTextfeld**_Exit()**
- Form**_MouseMove()**
- ...

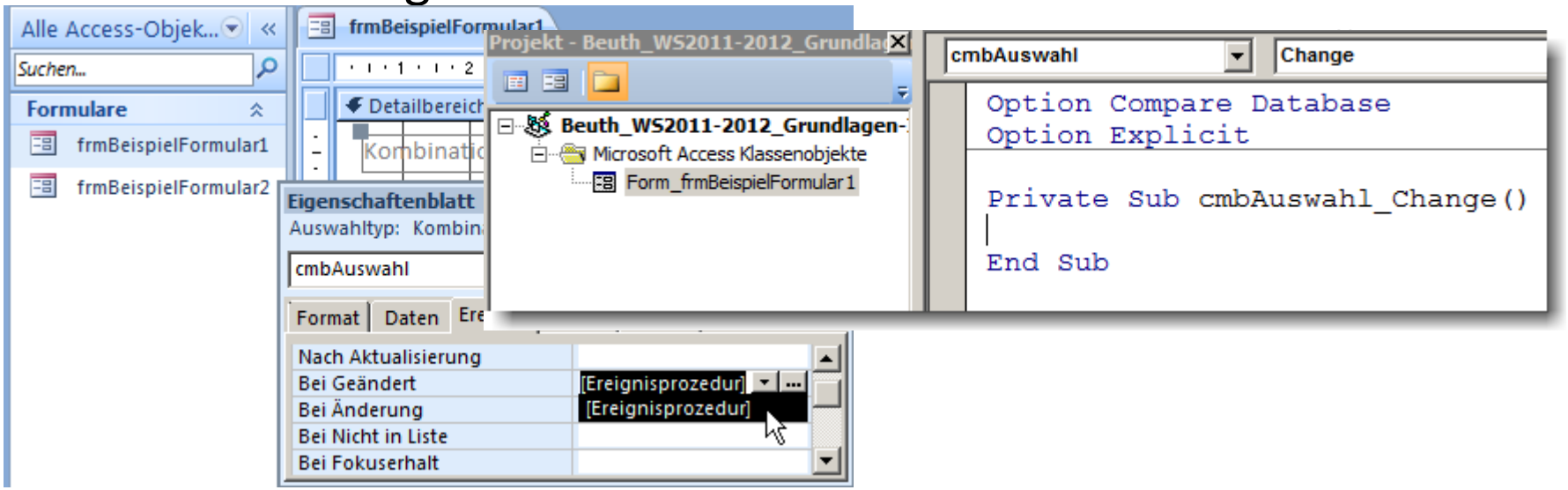
Ereignisse als Reaktion auf Benutzeraktionen

Benutzer führt Aktionen auf der Oberfläche aus

- Klicken auf Schaltflächen
- Eingeben von Daten in Felder
- Auswahl von Listeneinträgen
- Verlassen von Feldern
- Mouse bewegen

Als Reaktion werden Ereignisprozeduren aufgerufen

- btnSchalflaeche_**Click()**
- txtTextfeld_**KeyPress()**
- cmbAuswahl_**Change()**
- txtTextfeld_**Exit()**



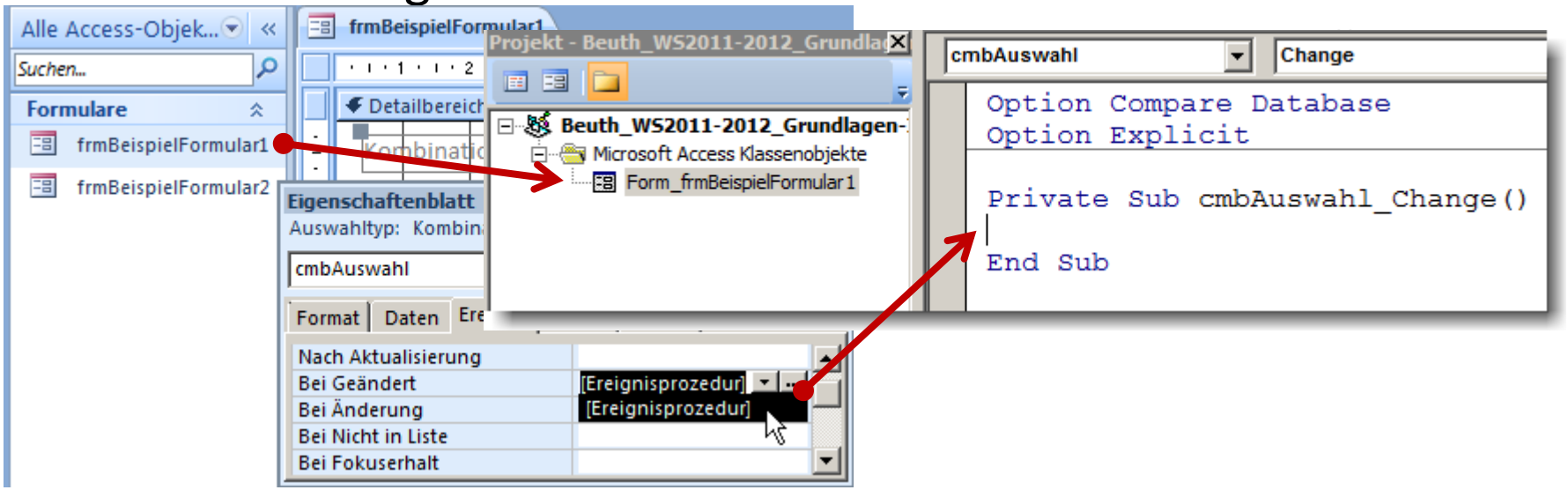
Ereignisse als Reaktion auf Benutzeraktionen

Benutzer führt Aktionen auf der Oberfläche aus

- Klicken auf Schaltflächen
- Eingeben von Daten in Felder
- Auswahl von Listeneinträgen
- Verlassen von Feldern
- Mouse bewegen

Als Reaktion werden Ereignisprozeduren aufgerufen

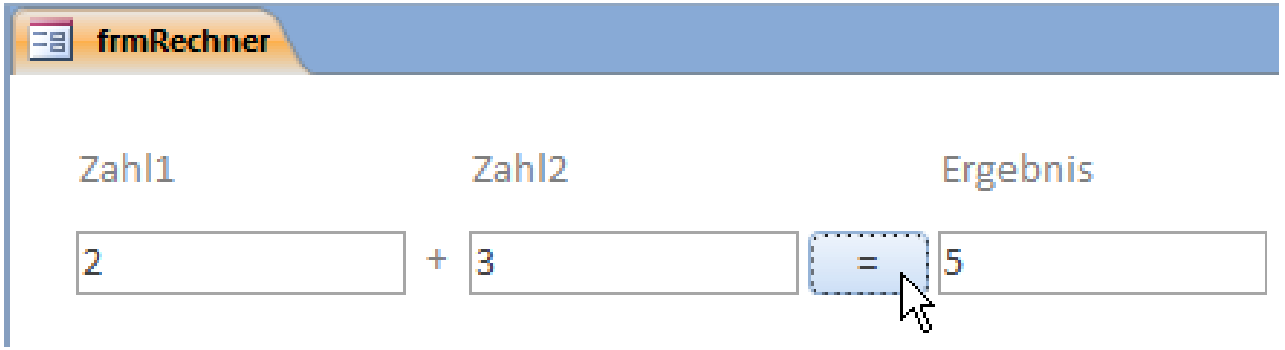
- btnSchalflaeche**_Click()**
- txtTextfeld**_KeyPress()**
- cmbAuswahl**_Change()**
- txtTextfeld**_Exit()**



Demo: Formulare und Ereignisse

Demo 1.12

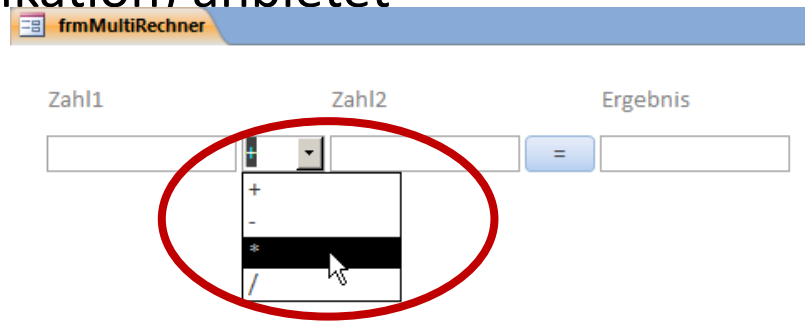
– Ereignisprozedur, die beim Klick auf die Schaltfläche "=" das Ergebnis der Addition in das Ergebnisfeld schreibt



Übung: Formulare und Ereignisse

Aufgabe 1.11

– Ändern Sie die Oberfläche des Taschenrechners so, dass anstelle des "+"-Operators eine Aufklappliste mehrere Operationen (z.B. Addition, Subtraktion, Multiplikation) anbietet



– Erstellen Sie eine Ereignisprozedur,
  - die beim Klick auf die Schaltfläche "=" zunächst prüft, welchen Wert die Aufklappliste hat
  - das Ergebnis der Operation berechnet und in das Ergebnisfeld schreibt

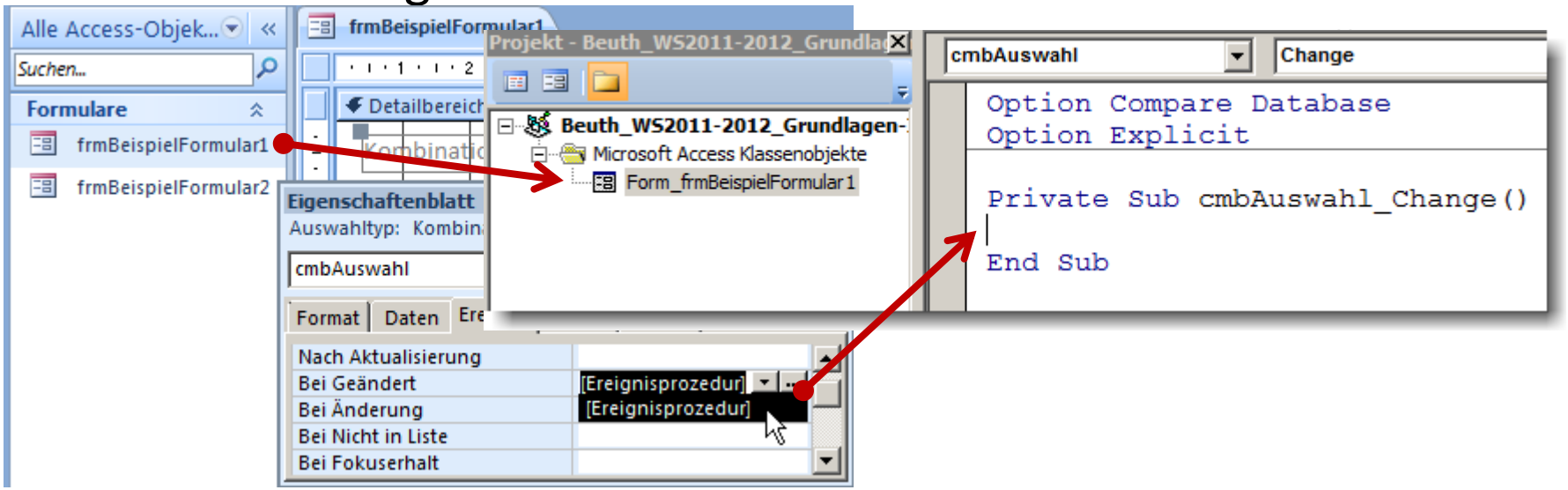
Ereignisse als Reaktion auf Benutzeraktionen

Benutzer führt Aktionen auf der Oberfläche aus

- Klicken auf Schaltflächen
- Eingeben von Daten in Felder
- Auswahl von Listeneinträgen
- Verlassen von Feldern
- Mouse bewegen

Als Reaktion werden Ereignisprozeduren aufgerufen

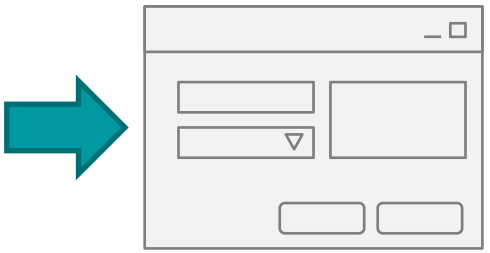
- btnSchalflaeche**_Click()**
- txtTextfeld**_KeyPress()**
- cmbAuswahl**_Change()**
- txtTextfeld**_Exit()**



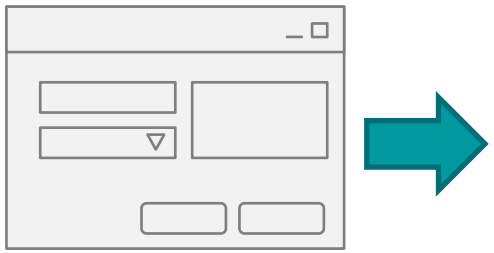
Navigation zwischen Formularen

Möglichkeiten zur Navigation zwischen Formularen

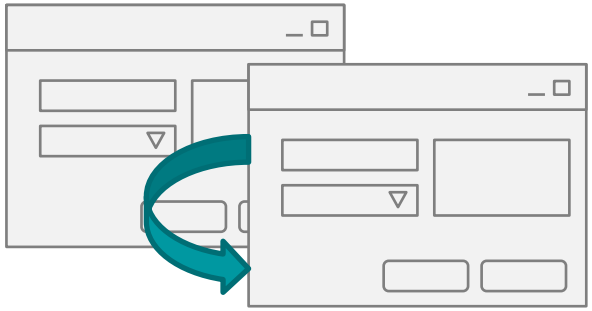
- Fenster öffnen
  - per Name öffnen
  - Parameterübergabe an Fenster



- Fenster schließen
  - aktuelles Fenster schließen
  - ein anderes Fenster schließen



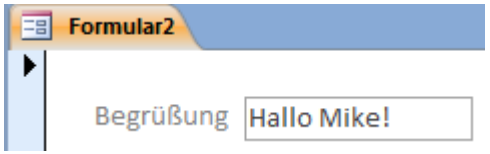
- Von einem Fenster zum nächsten navigieren



Demo: Navigation zwischen Fenstern

Demo 1.12

- Formular1
  - mit Textfeld "Name" und
  - Schaltfläche "Weiter"
- Formular2 mit Textfeld "Begrüßung"
- Bei Klick auf Weiter
  - soll Formular2 geöffnet werden
  - im Textfeld soll "Hallo" und der Name angezeigt werden, der in Fenster 1 eingegeben wurde



Demo: Navigation zwischen Fenstern

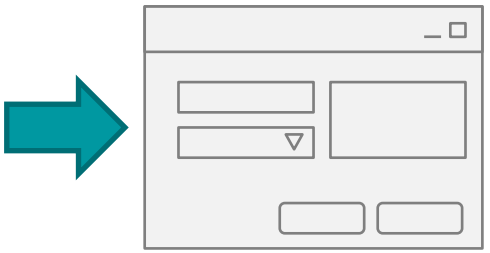
Demo 1.12 - Lösungsansatz:

- Kommando **`DoCmd`** stellt Funktionen zur Verfügung, z.B.
  - **`OpenForm`** *<Formularname>*
  - **`OpenForm`** *<Formularname>*`,,,,,,`*<Übergabeparameter>*

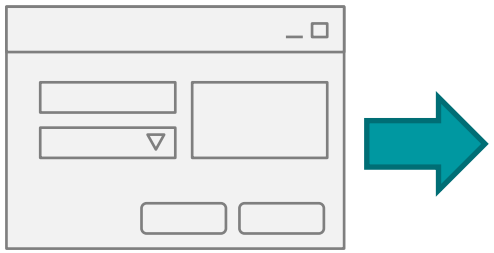
Navigation zwischen Formularen

Möglichkeiten zur Navigation zwischen Formularen

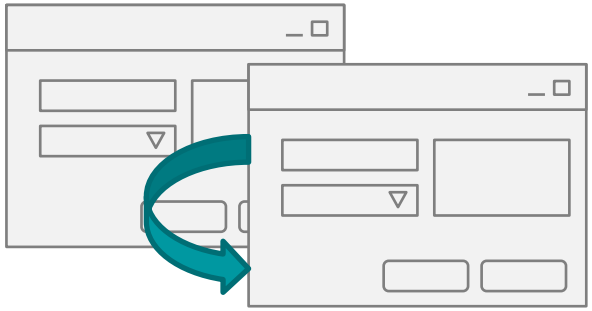
- Fenster öffnen
  - per Name öffnen
  - Parameterübergabe an Fenster



- Fenster schließen
  - aktuelles Fenster schließen
  - ein anderes Fenster schließen

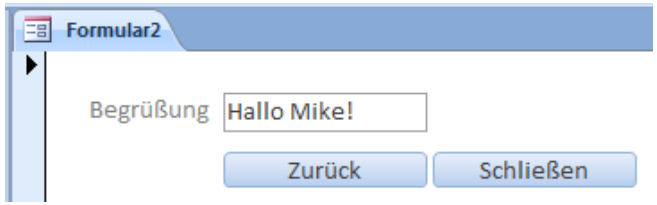


- Von einem Fenster zum nächsten navigieren



Demo: Navigation zwischen Fenstern

Demo 1.13



- Formular1 unverändert
- Formular2 soll
  - eine Zurück- und Schließen-Schaltfläche bekommen
- unverändert bei Klick auf Weiter
- Beim Klick auf Schließen Formular2 und Formular1 schließen
- Bei Klick auf Zurück zunächst noch nichts

Demo: Navigation zwischen Fenstern

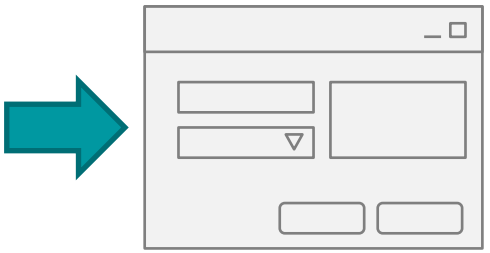
Demo 1.13 - Lösungsansatz:

- Kommando **`DoCmd`** stellt Funktionen zur Verfügung, z.B.
  - **`Close`** *<Typ-des-zu-schließenden-Objekts>, <Name>*

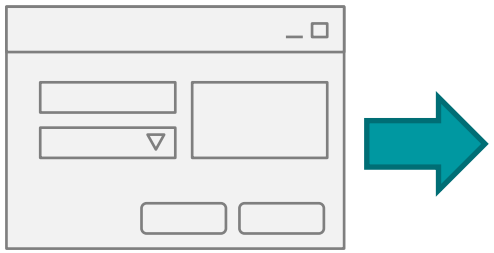
Navigation zwischen Formularen

Möglichkeiten zur Navigation zwischen Formularen

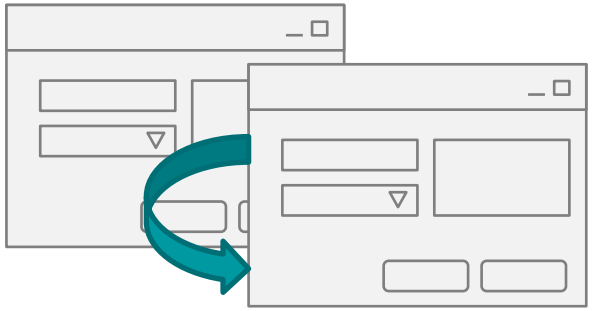
- Fenster öffnen
  - per Name öffnen
  - Parameterübergabe an Fenster



- Fenster schließen
  - aktuelles Fenster schließen
  - ein anderes Fenster schließen



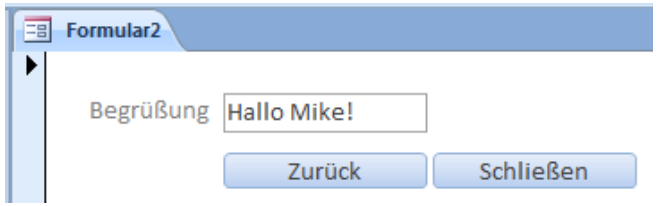
- Von einem Fenster zum nächsten navigieren



Demo: Navigation zwischen Fenstern

Demo 1.14

- Formular1 unverändert
- Formular2 unverändert
- Bei Klick auf Zurück
  von Formular 2 zu Formular 1 navigieren



Demo: Navigation zwischen Fenstern

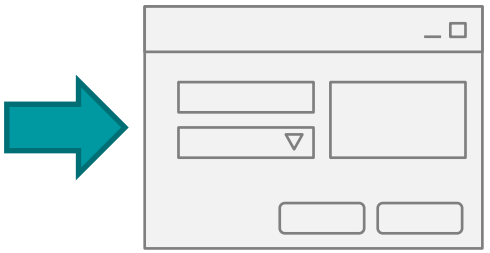
Demo 1.14 - Lösungsansatz:

- Kommando **`DoCmd`** stellt Funktionen zur Verfügung, z.B.
  - **`BrowseTo`** *<Typ-des-Objekts>, <Name>*

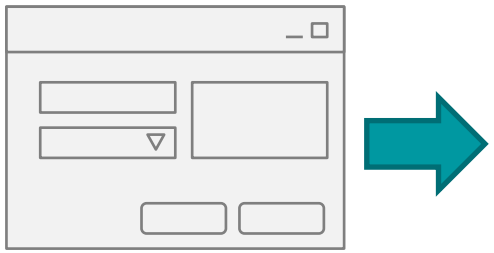
Navigation zwischen Formularen

Möglichkeiten zur Navigation zwischen Formularen

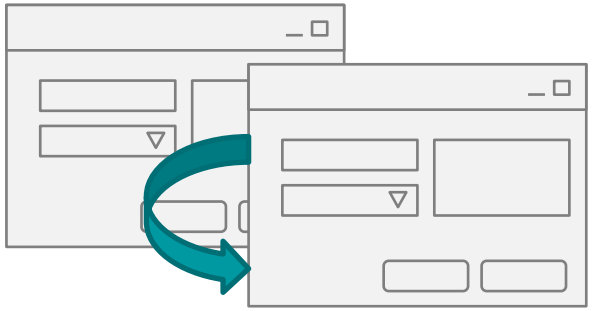
- Fenster öffnen
  - per Name öffnen
  - Parameterübergabe an Fenster



- Fenster schließen
  - aktuelles Fenster schließen
  - ein anderes Fenster schließen



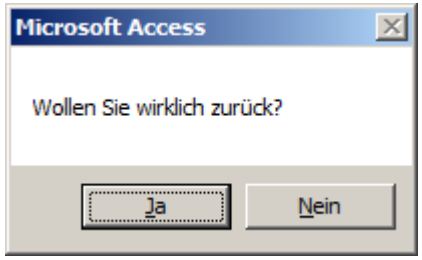
- Von einem Fenster zum nächsten navigieren

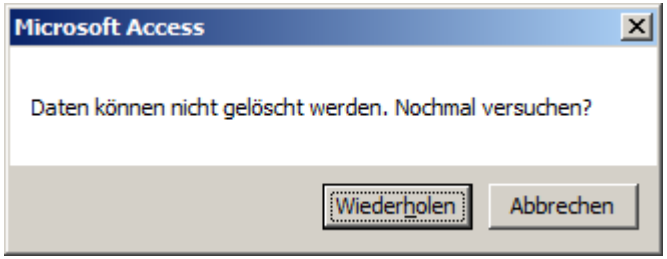


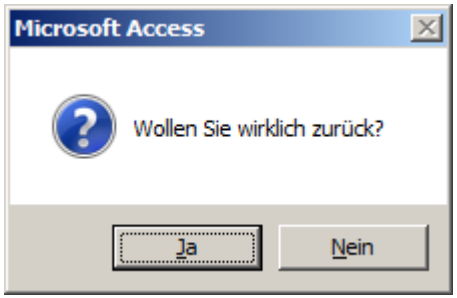
Meldungen

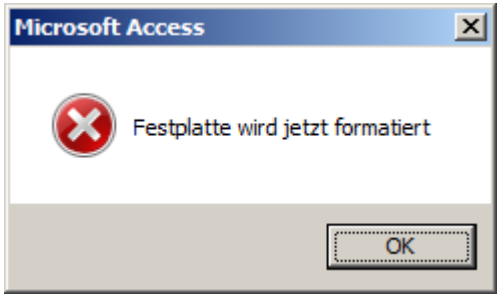
Meldungsfenster

- dienen zur Ausgabe einer Informationsmeldung oder
- sollen eine Benutzerentscheidung ermöglichen
- umfassen
  - Meldungstext
  - Schaltflächen, die zur Meldung passen
  - Symbol
- Rückgabe des Wertes der gedrückten Schaltfläche









Demo: Meldungsfenster

Demo 1.15

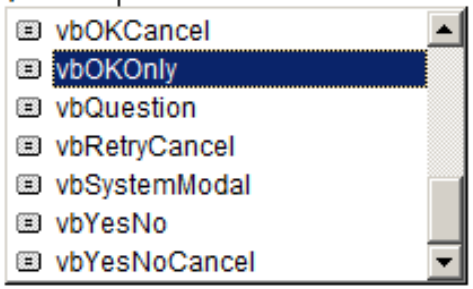
- Beim Klick auf Zurück (im Formular2 aus Demo 1.14) soll die folgende Meldung erscheinen
- Nur wenn der Benutzer auf "Ja" klickt, soll zum Formular1 navigiert werden

Demo: Meldungsfenster

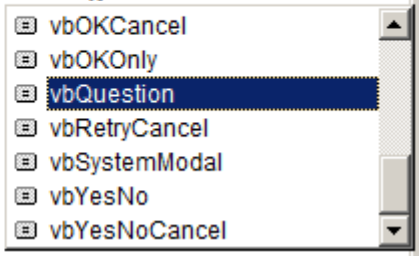
Demo 1.15 – Lösungsansatz

- Befehl MsgBox bietet die Möglichkeit ein Meldungsfenster zu öffnen, z.B.
  - MsgBox <Melundungstext>, <Art-der-Schaltflächen-Icons>
- Beispiel

```
MsgBox "Wollen Sie wirklich zurück?", vbO
```



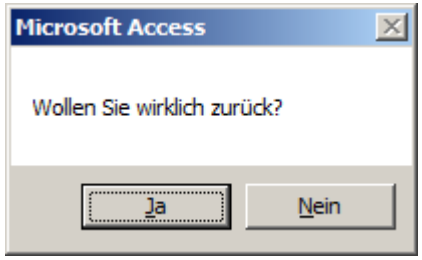

```
Dim intAntw As Integer
Let intAntw = _
 MsgBox("Wollen Sie wirklich zurück?", vbYesNo + vbQus
```

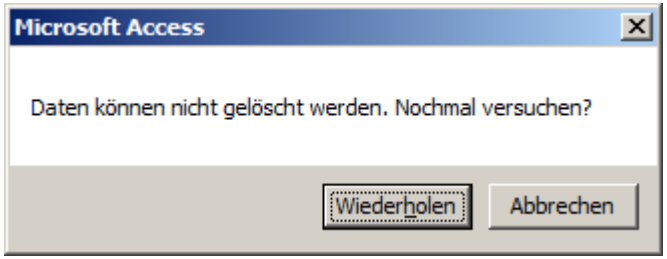


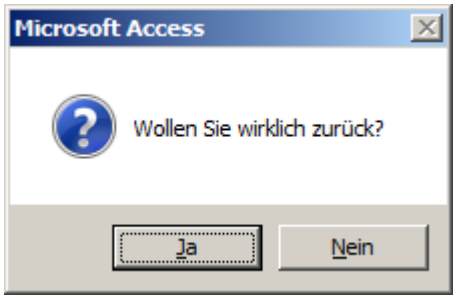
Meldungen

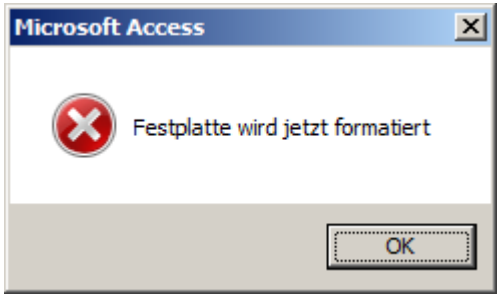
Meldungsfenster

- dienen zur Ausgabe einer Informationsmeldung oder
- sollen eine Benutzerentscheidung ermöglichen
- umfassen
  - Meldungstext
  - Schaltflächen, die zur Meldung passen
  - Symbol
- Rückgabe des Wertes der gedrückten Schaltfläche









Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Dokumentation

Dokumentation von Programmen

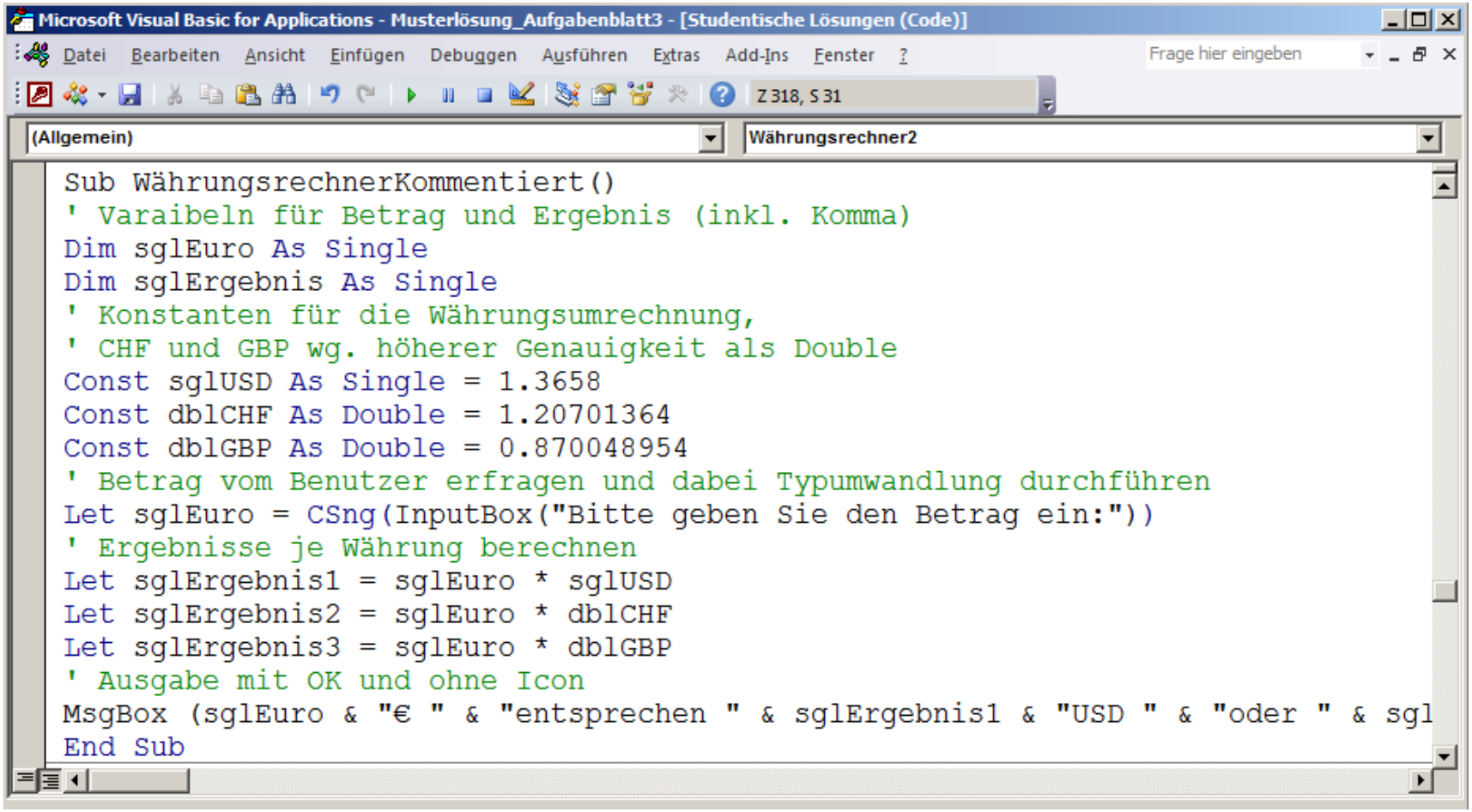
- Nachvollziehen der Programme und Abläufe innerhalb des Programms (z.B. bei Verzweigungen)
- Bedeutung von Variablen und Parametern
- Spezifikation von Prozeduren/Funktionen (was tun sie?)

im Quellcode in Form von Kommentaren

- die oberhalb des kommentierten Elementes stehen
- die bei der Programmausführung keine Auswirkung haben

Dokumentation

Beispiel



Inhalt

Einordnung

Einstieg in MS Access mit VBA

- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...

Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

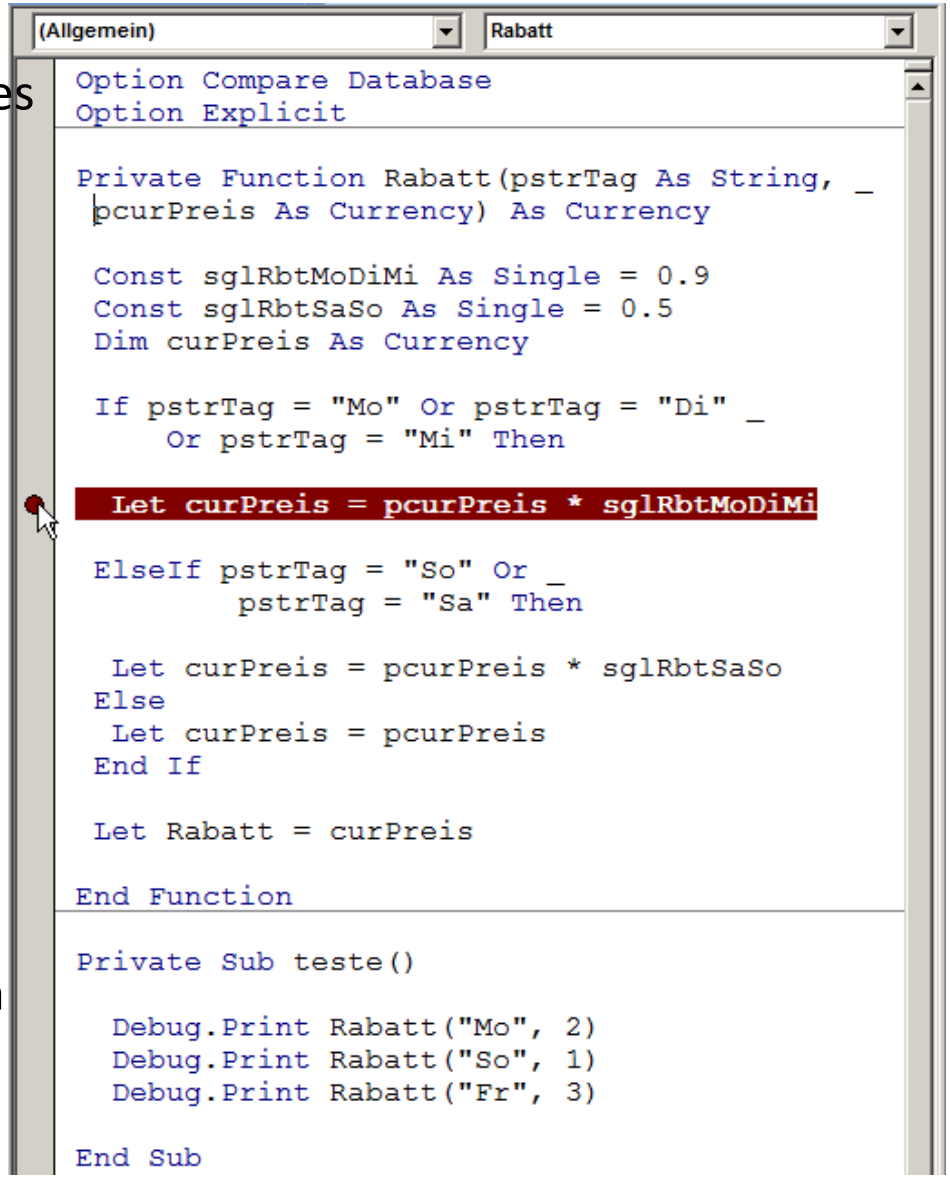
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

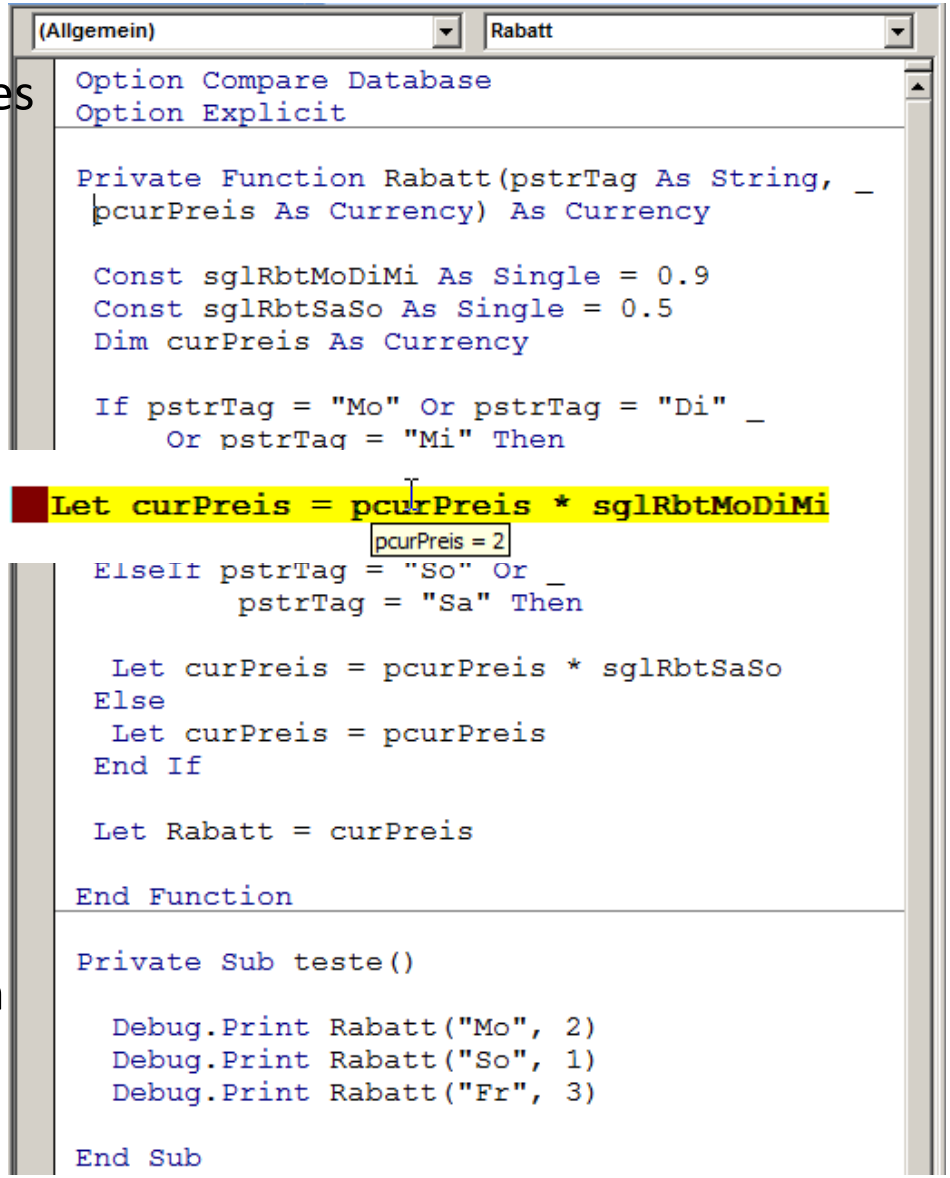
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

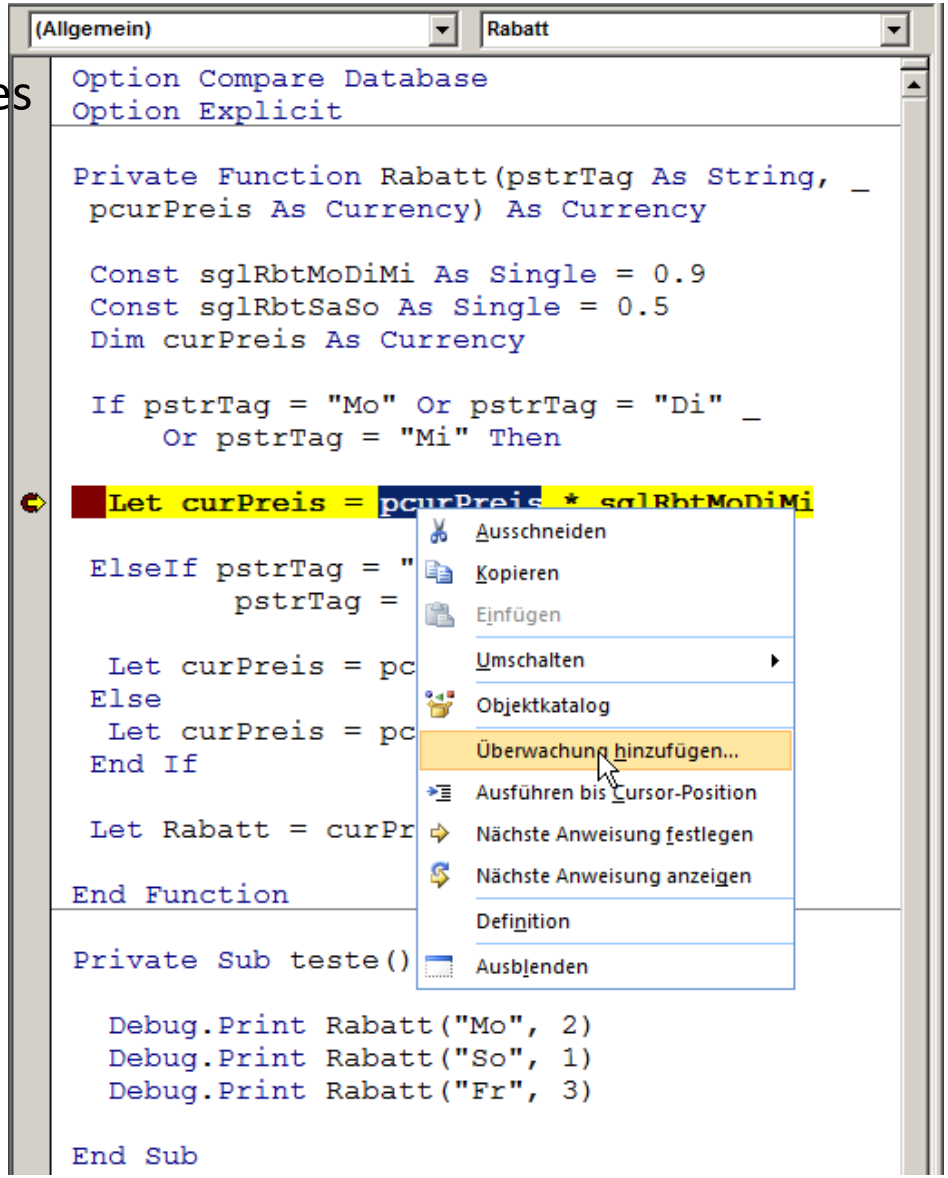
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

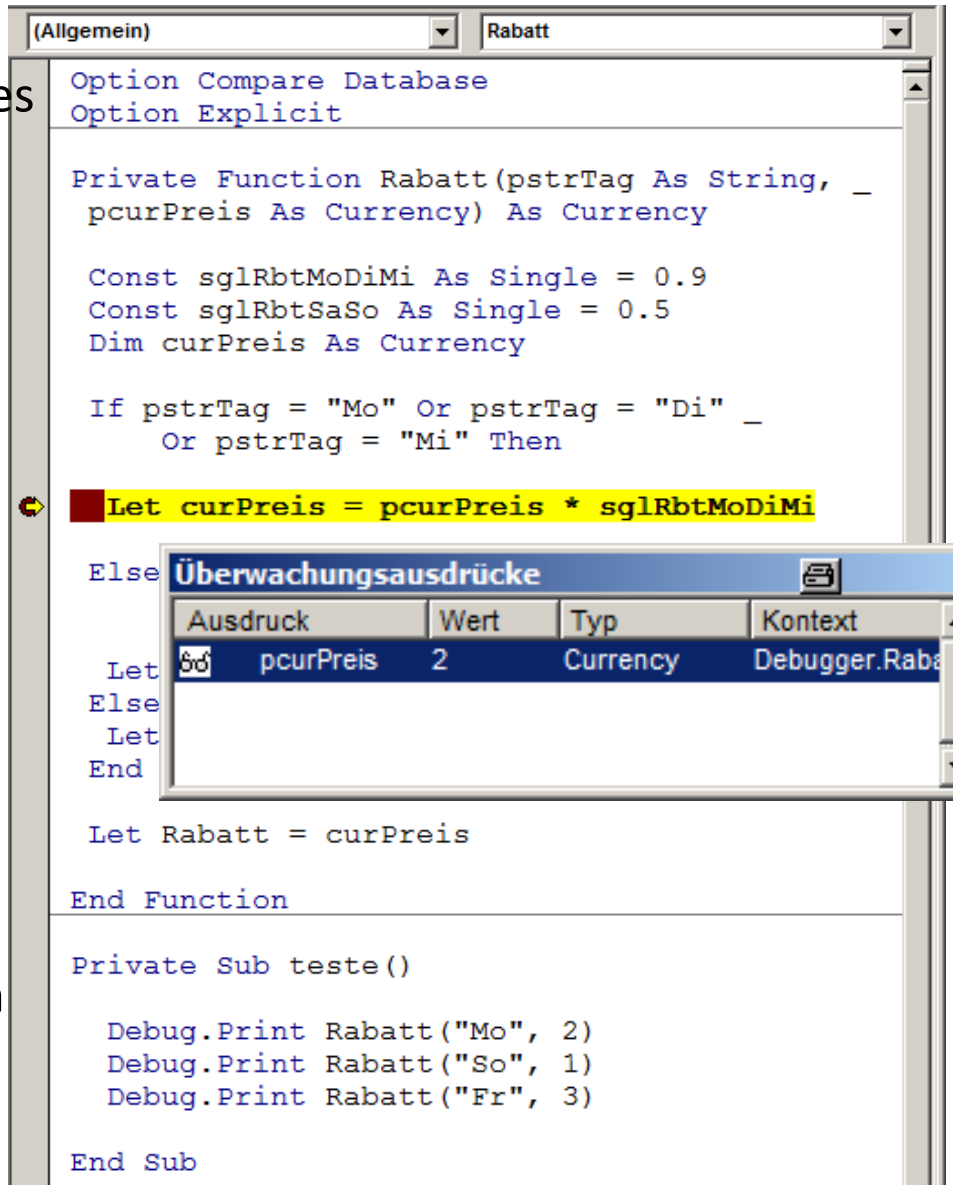
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

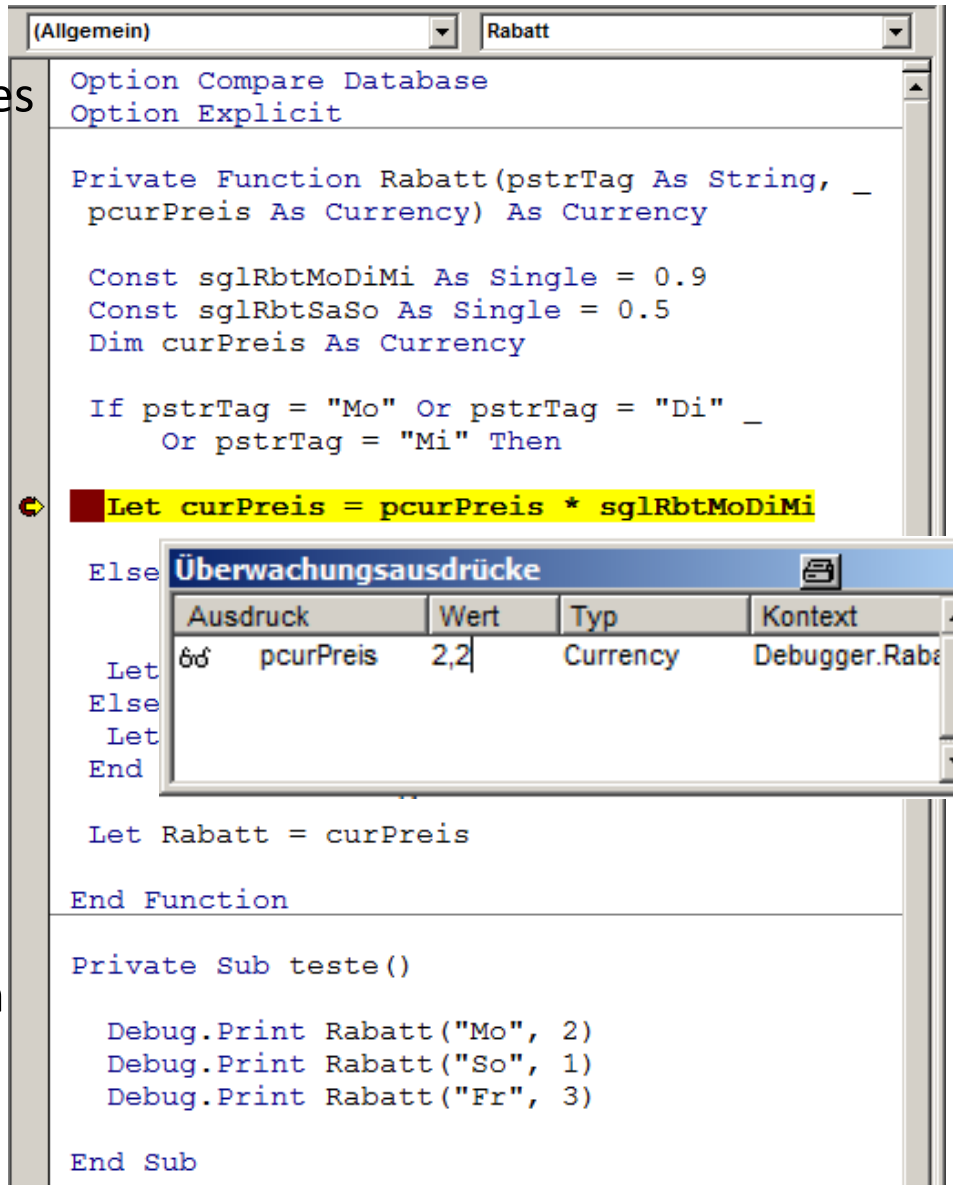
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

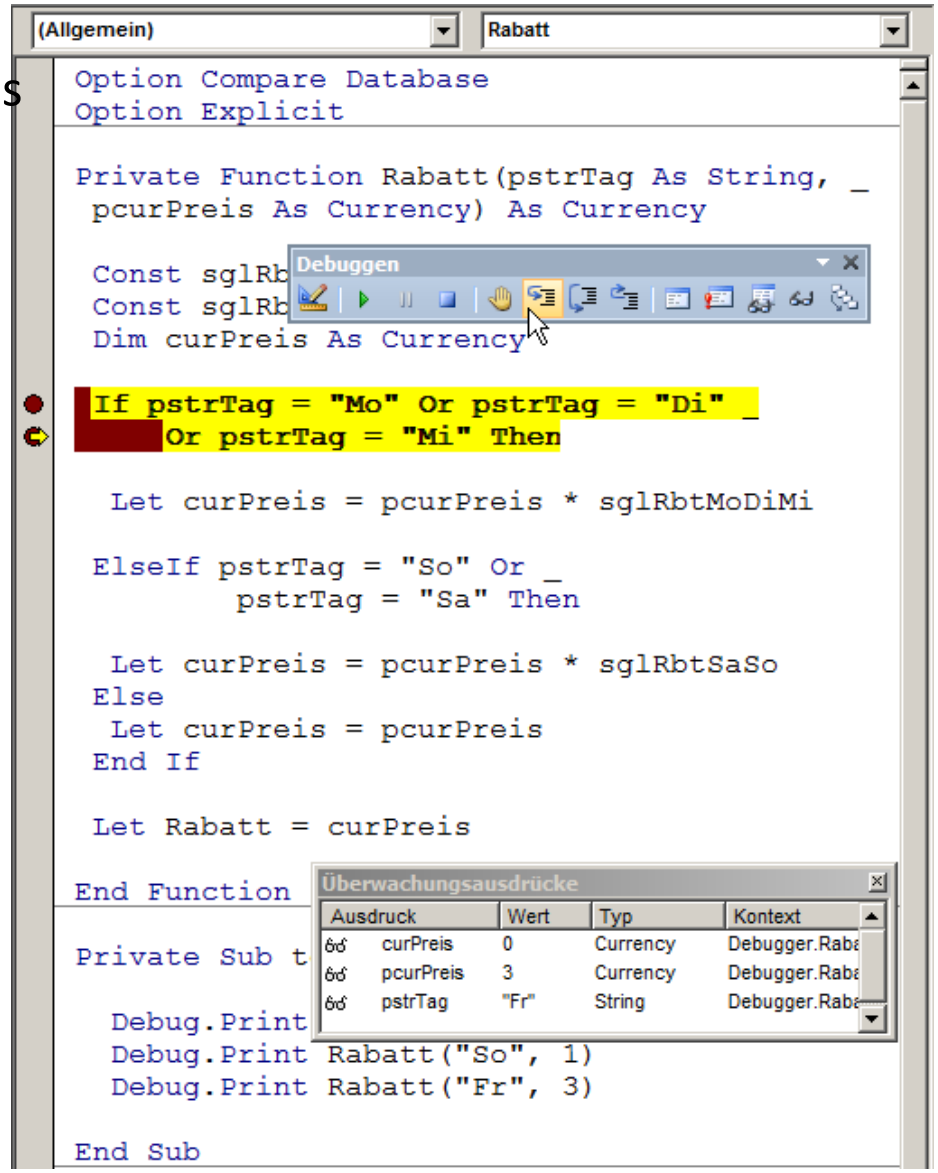
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

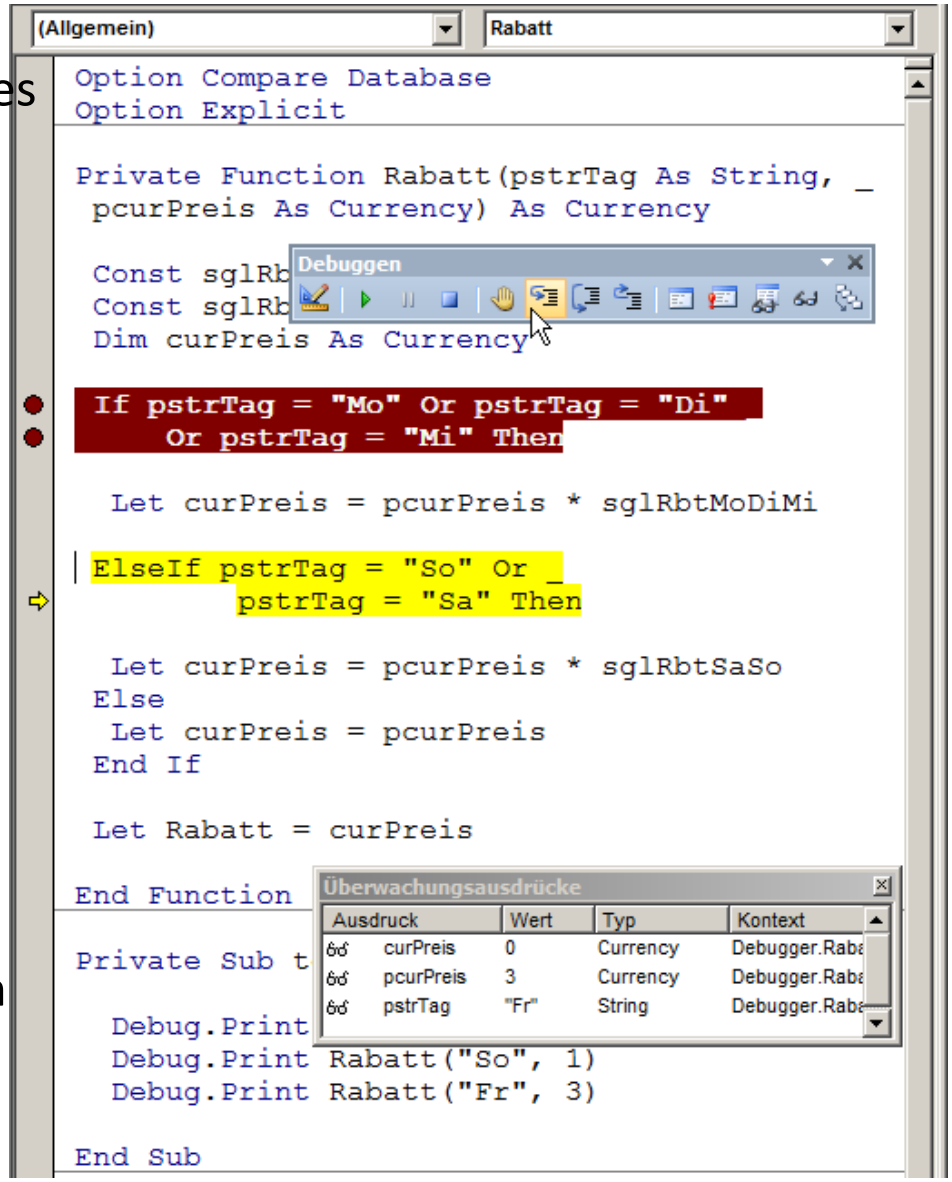
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

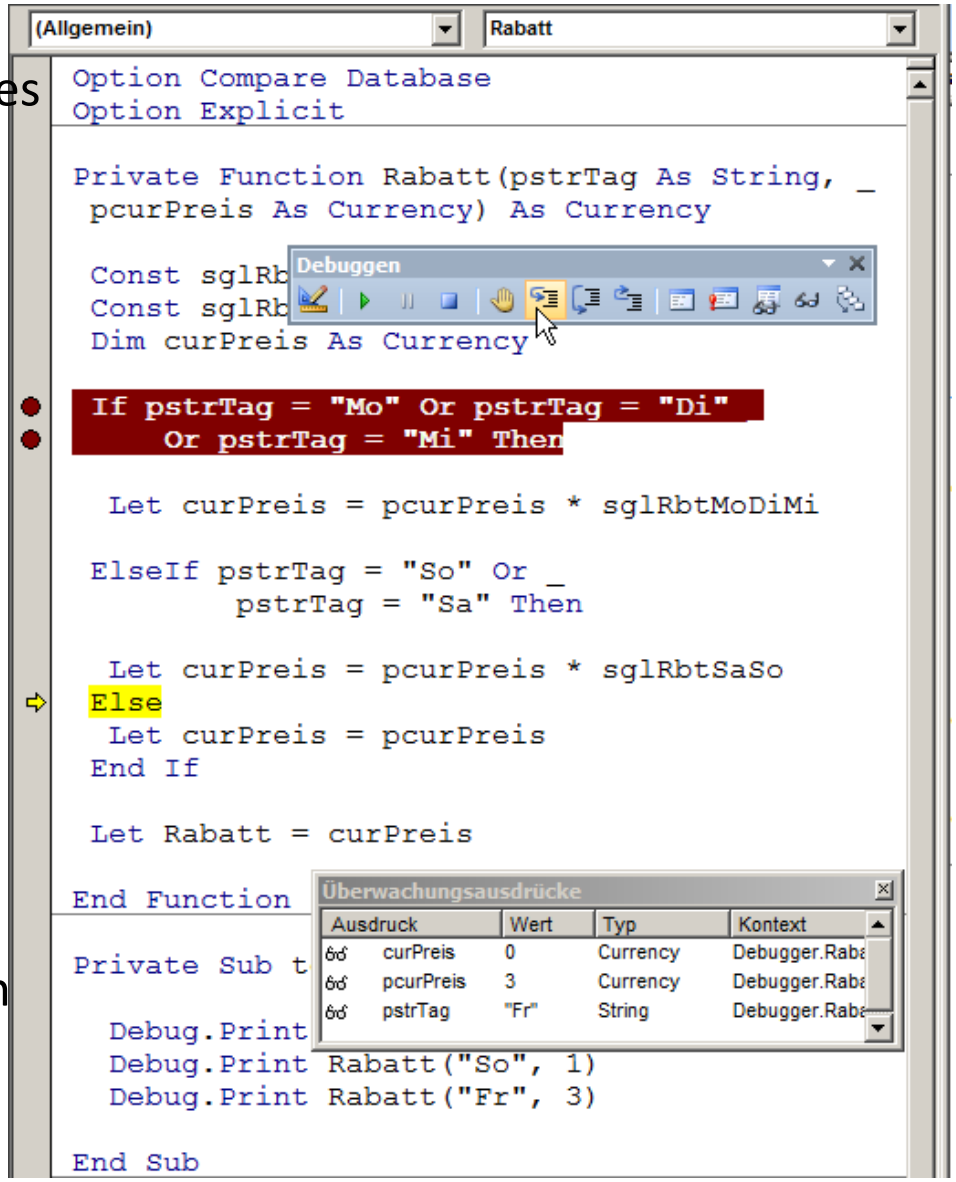
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

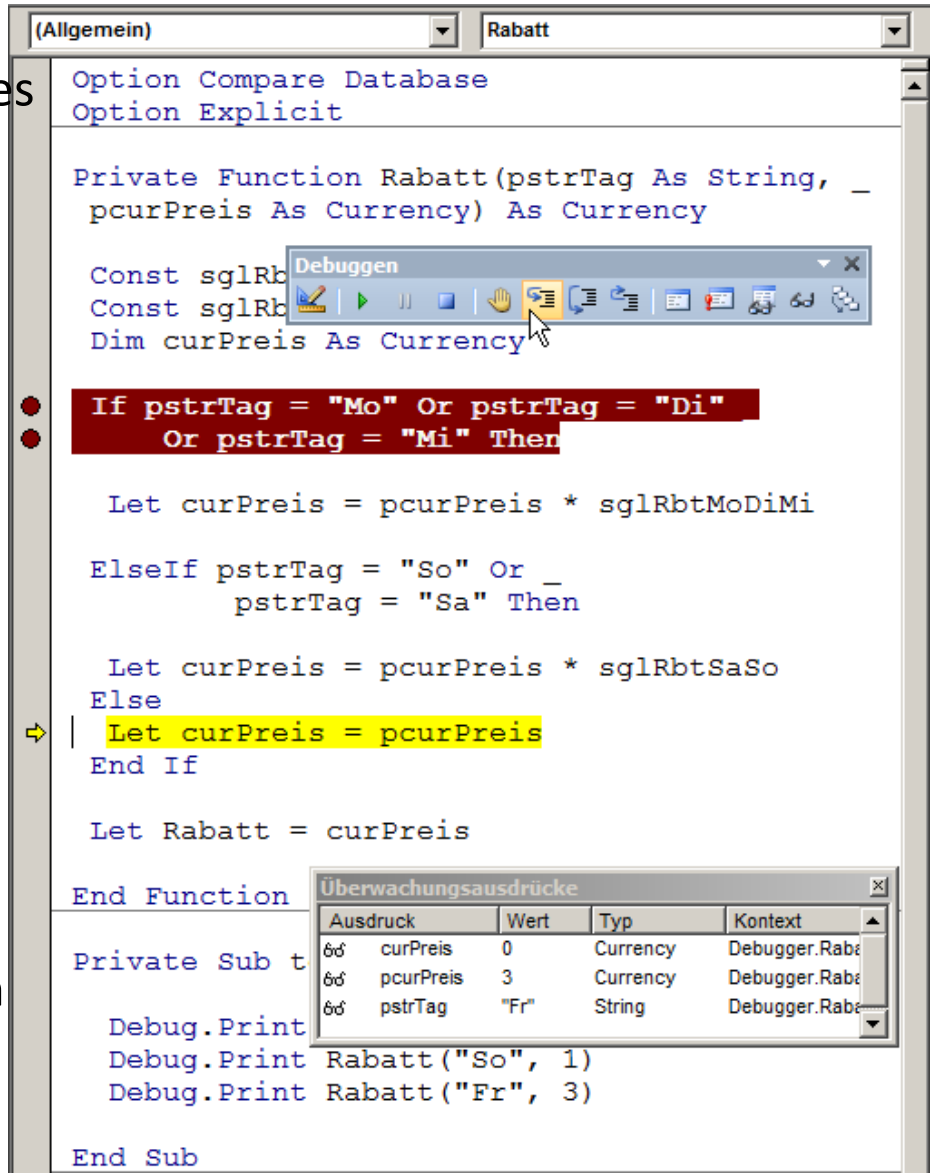
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Debugger

Zweck

- Nachvollziehen und Erklärung eines Programmablaufs
  - der Ausführungsreihenfolge von Anweisungen
  - dient zur Erklären der Wertebelegung von Variablen
- Fehlerbeseitigung

Haltepunkte

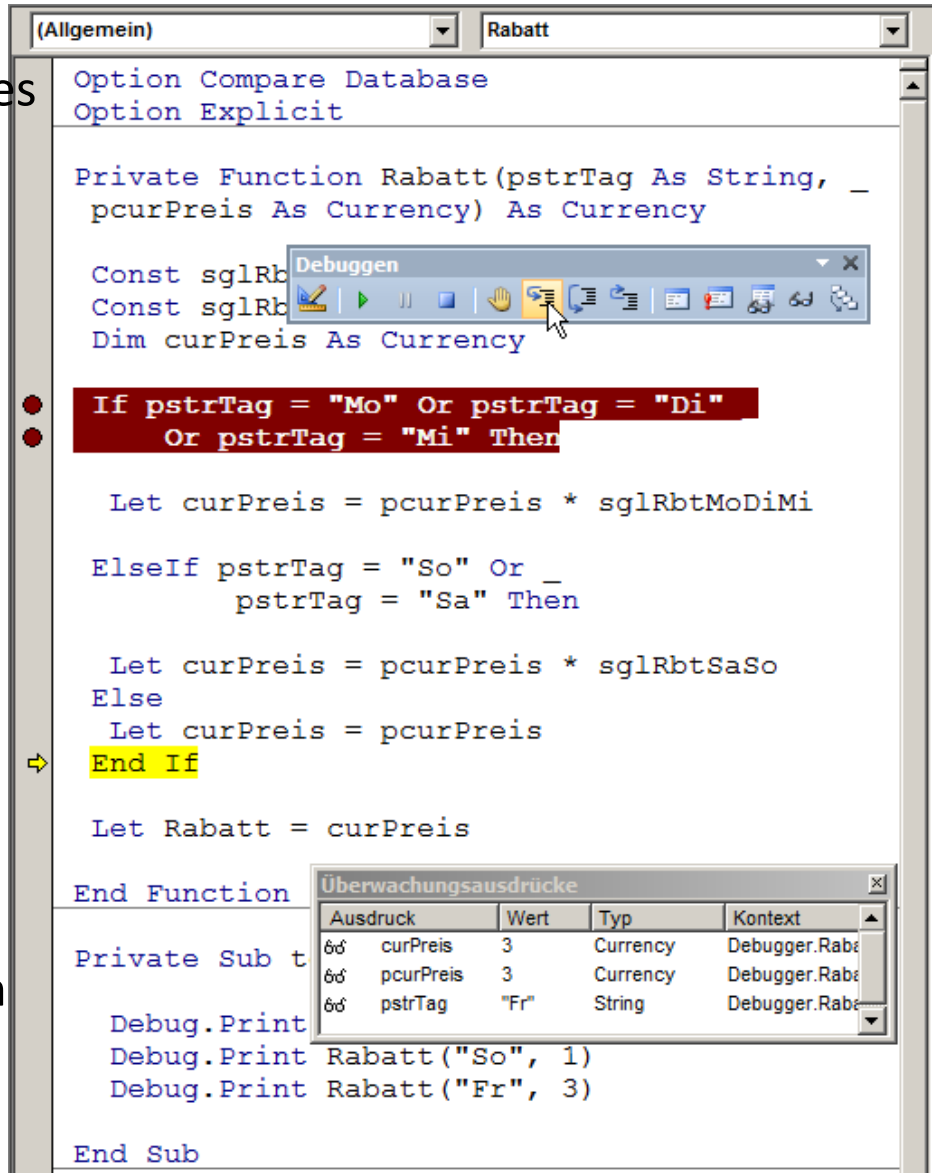
- Markieren Zeilen (Anweisung) in der der Programmablauf angehalten werden soll

Überwachung

- Betrachtung von Variablenwerten
- Änderung von Variablenwerten

schrittweise Ausführung

- Ausführen einzelner Anweisungen
- Ausführen einer ganzen Prozedur
- ...



Übung: Alles zusammen!

Aufgabe 1.12

- Login-Formular mit Textfeldern für Benutzername und Passwort, einer Abbrechen- und einer Login-Schaltfläche
- Erstellen Sie Ereignisprozeduren für den
  - Klick auf Abbrechen: Formular schließen (Me.Close())
  - Klick auf Login
    - Deklarieren Sie sich zwei Variablen für Benutzername und Passwort
    - Initialisieren Sie die Variablen mit Ihrem Namen und einem geheimen Passwort
    - Vergleichen Sie die in den Feldern eingegebenen Werte mit den Werten, die Sie in den Variablen gespeichert haben
      - Stimmen sie nicht überein, zeigen Sie eine Fehlermeldung
      - Stimmen sie überein, zeigen Sie eine Willkommensmeldung und schließen das Formular

Inhalt

Einordnung

Einstieg in MS Access mit VBA

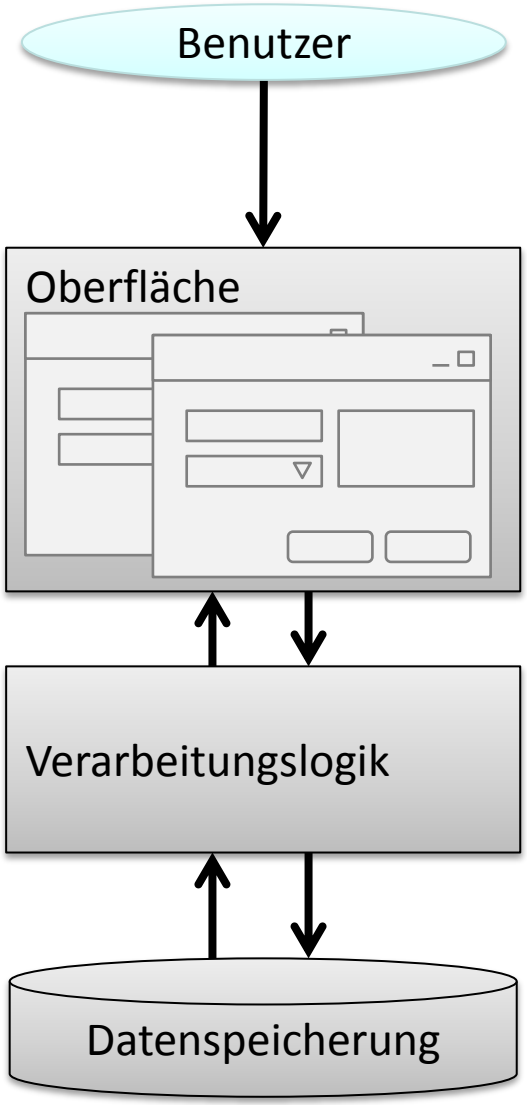
- Erste Schritte
- Hallo Welt-Programm

Grundlagen von VBA und MS Access

- Variable mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse
- Dokumentation
- Debugger

Ausblick

Ausblick


Benutzer
Oberfläche
Verarbeitungslogik
Datenspeicherung

Ausblick

Benutzeroberfläche: kann umgesetzt werden mit

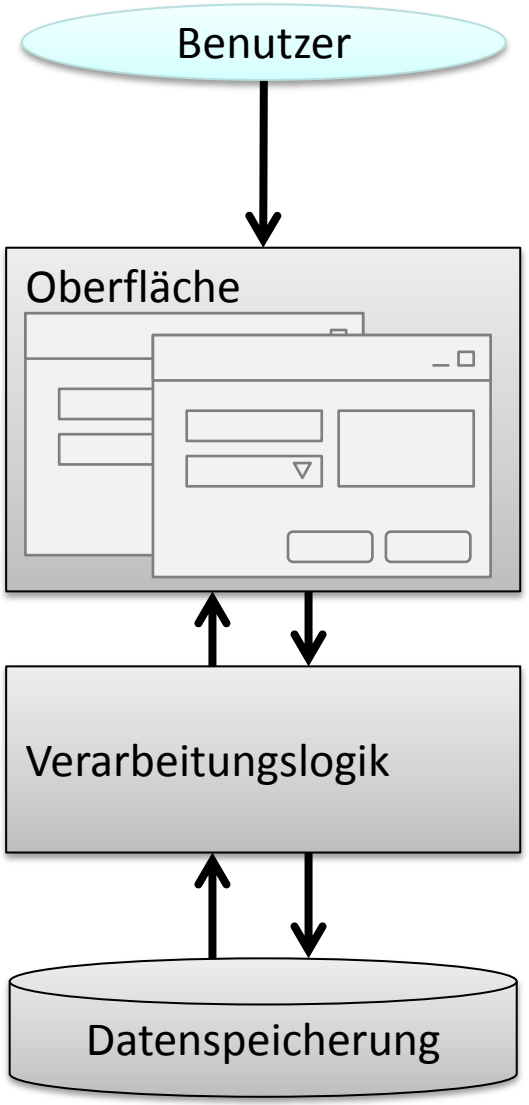
- Formulare mit ihren Klassenobjekten,
- Oberflächenelemente, z.B. Schaltflächen, Felder
- Ereignisse und Ereignisprozeduren

Verarbeitungslogik: kann umgesetzt werden mit

- Modulen mit Prozeduren, Funktionen
- Schleifen, Verzweigungen
- Datentypen, Variablen

Datenspeicherung

- noch offen



Ausblick

Benutzeroberfläche: kann umgesetzt werden mit

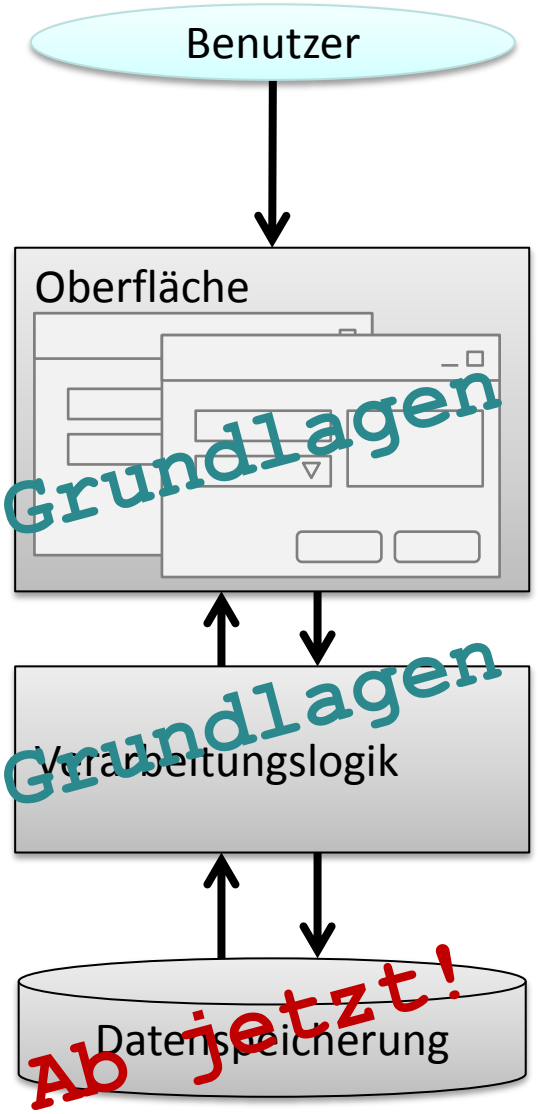
- Formulare mit ihren Klassenobjekten,
- Oberflächenelemente, z.B. Schaltflächen, Felder
- Ereignisse und Ereignisprozeduren

Verarbeitungslogik: kann umgesetzt werden mit

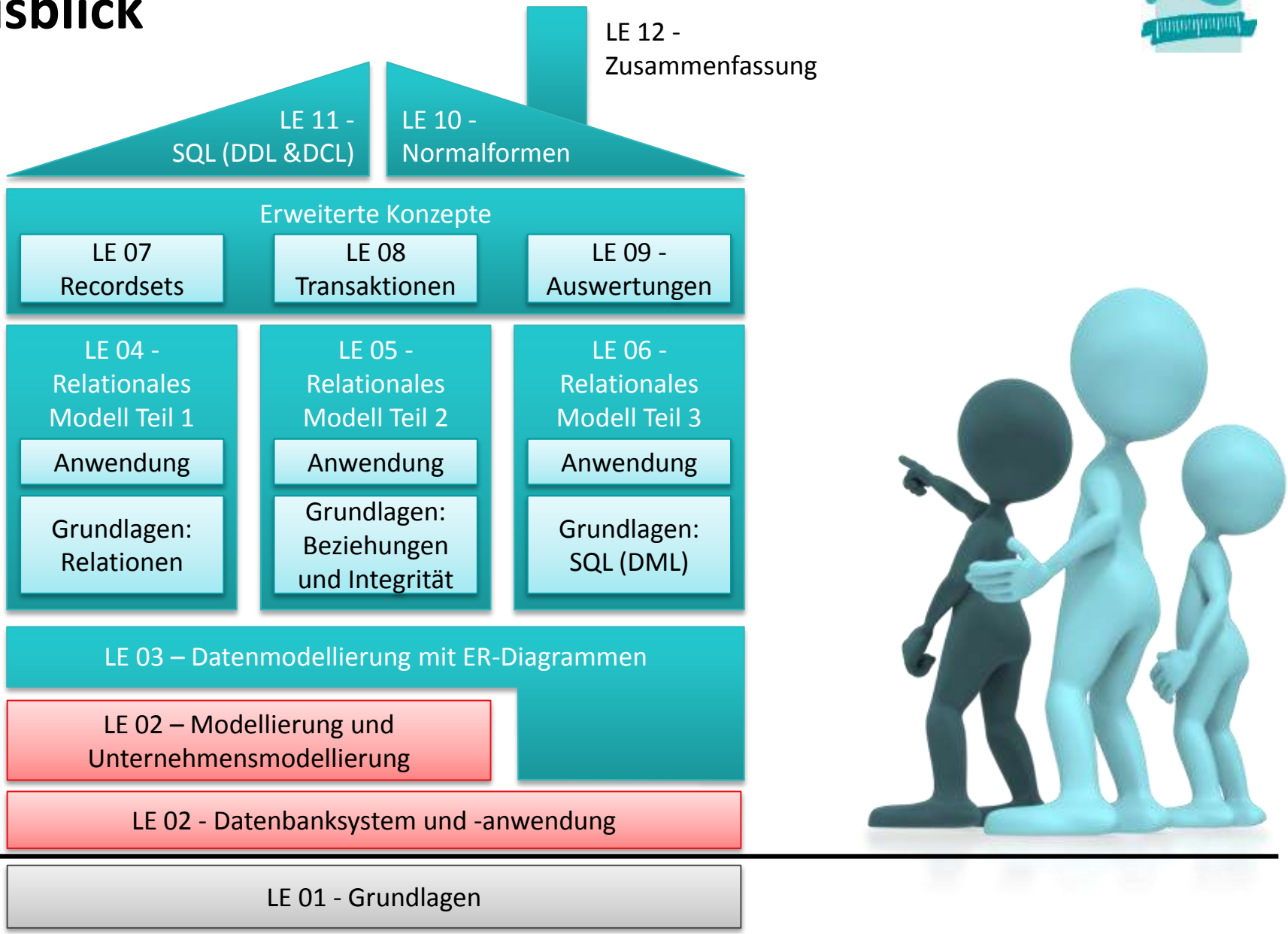
- Modulen mit Prozeduren, Funktionen
- Schleifen, Verzweigungen
- Datentypen, Variablen

Datenspeicherung

- noch offen



Ausblick


LE 12 - Zusammenfassung
LE 11 - SQL (DDL &DCL)
LE 10 - Normalformen
Erweiterte Konzepte
LE 07 Recordsets
LE 08 Transaktionen
LE 09 - Auswertungen
LE 04 - Relationales Modell Teil 1
Anwendung
Grundlagen: Relationen
LE 05 - Relationales Modell Teil 2
Anwendung
Grundlagen: Beziehungen und Integrität
LE 06 - Relationales Modell Teil 3
Anwendung
Grundlagen: SQL (DML)
LE 03 – Datenmodellierung mit ER-Diagrammen
LE 02 – Modellierung und Unternehmensmodellierung
LE 02 - Datenbanksystem und -anwendung
LE 01 - Grundlagen

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wirtschaftsinformatik 2
LE 01 – Wiederholung: Grundlagen von VBA und MS Access

Thomas Off

www.ThomasOff.de/lehre/beuth/wi2