BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wirtschaftsinformatik 1
LE 03 – Variable, Konstante und Datentypen

Prof. Dr. Thomas Off

http://www.ThomasOff.de/lehre/beuth/wi1

Themen

06 – Zusammenfassung

05 – Debugger und Testen

04 – Fortgeschrittene Konzepte

04.A	04.B	04.C
Oberflächen (Teil 1) Elemente und Eigenschaften	Oberflächen (Teil 2) Ereignisverarbeitung	Zugriff auf Dateisystem und Anwendungen

03 – Grundkonzepte

03.A	03.B	03.C	03.D	03.E
Wert Ausdruck Variable Konstante Datentyp	Bedingte Ausführung/ Verzwei-gungen	Schleifen	Felder Mengen	Prozedur Funktion Modul

02 – Grundlagen der Programmierung

01 – Grundbegriffe der Wirtschaftsinformatik

Inhalt

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Rückblick

Algorithmus

- Definition: präzise, vollständige, eindeutig formulierte, endliche Verarbeitungsvorschrift, die Ausgangssituation in ein Ergebnis überführt, das zur Lösung einer Aufgabe dienen soll.
- Bestandteile: Anweisungen, Ablauf, Verzweigungen, Schleifen, Unterprogramme
- Beschreibung: Struktogramme, Programmablaufpläne, UML-Aktivitätsdiagramme, ...

Datenelement und -struktur

- einfache Datenelemente, zum Schreiben und Lesen eines Werts;
- komplexe Datenelemente, die aus einfachen aufgebaut sind und
- komplexe Datenstrukturen, die Datenelemente in bestimmter Form organisieren und außer schreibendem und lesendem Zugriff spezielle Aktionsmöglichkeiten bieten (z.B. Einfügen, Entfernen).
- Beispiele: Verkettete Liste, Stapel, Schlange, Baum

Rückblick

Programm

- mit den Sprachmitteln einer konkreten Programmiersprache ausgedrückter Algorithmus in Verbindung mit den ebenso ausgedrückten Datenstrukturen zur Ausführung in einem Computer
- kann vorliegen als
  - Quellcode: Darstellung in einer lesbaren und verständlichen Programmiersprache
  - Maschinencode: Darstellung mit Befehlen aus dem Befehlsvorrats des konkret verwendeten Computers

Programmiersprache: Formale Sprache zur Formulierung von Programmen mit präziser Syntax und eindeutiger Semantik

Maschinensprache: Binäre und ausführbare Darstellung des Programms, abhängig von der verwendeten Hardware

Rückblick

Compiler

- fertiger Programmcode wird vollständig dem Compiler übergeben und
- über ein Zwischenformat in ausführbares Programm übersetzt, das geladen und ausgeführt werden kann

Interpreter

- liest eine Anweisung aus dem Quellcode und übersetzt sie in Maschinencode, lädt die Anweisung und führt sie aus
- dann wird mit der nächsten Anweisung fortgefahren

Rückblick

Entwicklungs-umgebung

- stellt ein oder mehrere Werkzeuge zur Programmentwicklung zur Verfügung (z.B. zur Erfassung von Quellcode, den Compiler oder Interpreter, Debugger, Werkzeuge für die Gestaltung von Benutzeroberflächen)
- hier Verwendung von Microsoft Access mit der integrierten Programmiersprache Visual Basic for Applications (VBA)

MS Access/VBA als Entwicklungsumgebung

- VBA Editor in MS Access Datenbank verwenden
- In Modul werden unsere Anweisungen (zunächst) als Prozedur zwischen Sub und End Sub geschrieben
- Ausführung des Programms
  - Eingabemarkierung muss innerhalb der Prozedur positioniert sein
  - Per Menüeintrag Ausführen>Sub/User Form ausführen, per "Play"-Icon ▶ oder Funktionstaste F5

Rückblick

Wir haben in unserem ersten Programm bereits folgende Elemente kennengelernt

- Anweisungen: Debug.Print
- Schlüsselwörter (reservierte Wörter)
  - Sub
  - End Sub
- Bezeichner
  - Bsp1
- Literale, z.B. "Hallo Welt!"
- ...

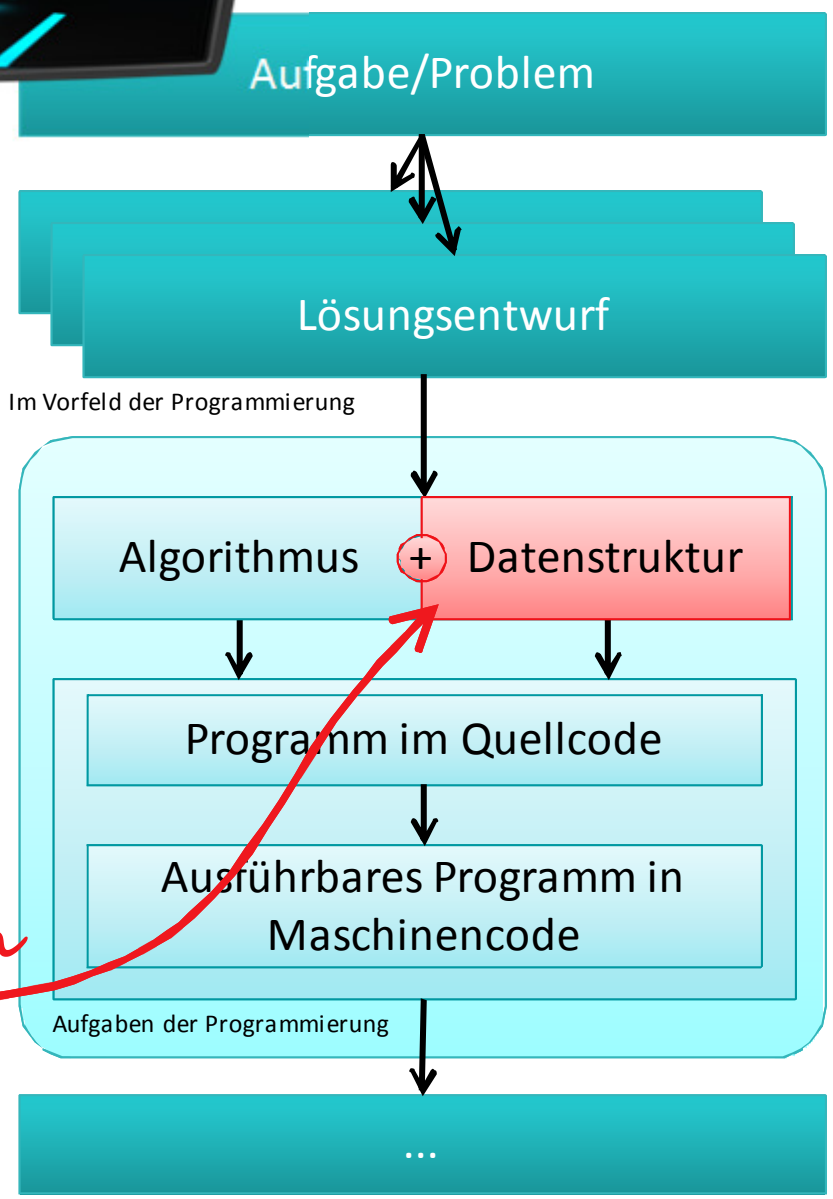
```
Sub Bsp1()

 Debug.Print "Hallo Welt!"

End Sub
```

Rückblick

Programmierung als stufenweise Umsetzung

- von Algorithmus und
- Datenstruktur
- in Quellcode einer Programmiersprache und
- der Überführung in ein ausführbares Programm im Maschinencode

Heute: Einfache Datenstrukturen



Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Kommentare in VBA-Programmen

Programm im Quellcode einer Programmiersprache soll gut lesbar und verständlich sein

- Struktur des Algorithmus klar erkennbar
- jede Anweisung verständlich
- Bedeutung der verwendeten Datenelemente erkennbar

Wichtiges Hilfsmittel ist der Kommentar

- als Hinzufügung einer Erläuterung (Annotation) zu einem Teil des Quellcodes
- ist zwar Teil des Quellcodes, aber keine Anweisung und auch kein Datenelement

Werden vom Interpreter/Compiler in VBA ignoriert, d.h. sind nicht Bestandteil des erzeugten Maschinencodes

Kommentare in VBA-Programmen

Syntax

- Einfaches Hochkomma leitet Kommentar bis zum Ende der Zeile ein

```
' <Kommentartext bis zum Ende der Zeile>
' <Kommentartext einer weiteren Zeile>
```

- Hinweis: Hochkomma ist das Zeichen über # - neben Enter!

```
' Kommentar zum Programm
' z.B. was es tut oder wer Autor ist
Sub Bsp()

 Debug.Print "Hallo Welt!" ' Ausgabe im Direktbereich

 ' Mehrere Ausgaben im Direktbereich (Block)
 Debug.Print "Hallo Thomas!"
 Debug.Print "Hallo Mike!"

End Sub
```

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Grundlagen

```
26062015
```

Was bedeutet diese Zahlenfolge?

Quelle: nach [1]

Grundlagen

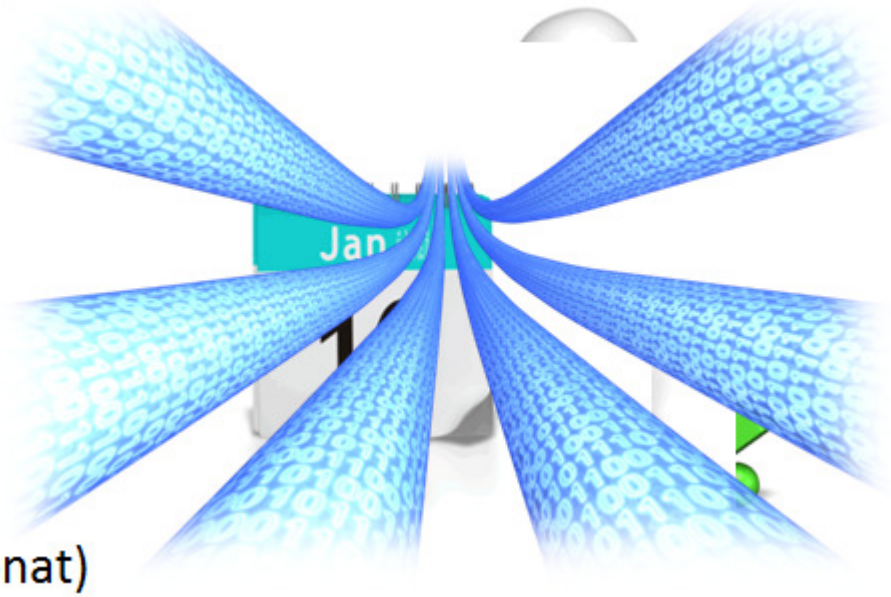
Verschiedene Deutungen der Folge 26062015

Quelle: nach [1]

Grundlagen

Verschiedene Deutungen der Folge 26062015

- Ganze Zahl
- Kalenderdatum
- Telefonnummer



Von Deutung abhängig

- erlaubte Operationen, z.B.
  - Addition ganzer Zahlen
  - Inkrement eines Kalenderdatums (z.B. Erhöhung um 1 Tag oder 1 Monat)
  - Erweiterung der Telefonnummer um internationale Vorwahl
- Speicherung der Daten ab
  - Ganze Zahl: Zahl in Binärdarstellung (als Folge von 0 und 1)
  - Kalenderdatum: 3 separate ganze Zahlen in Binärdarstellung
  - Telefonnummern: Folge von Ziffern (0 ... 9) und ggf. auch Zeichen ("+" oder "/")

Quelle: nach [1]

Grundlagen

Konsequenz: zur Beschreibung von Daten, die in einem Programm genutzt werden, ist folgendes festzulegen:

- Wie wird Datenart gedeutet, welchen Wertebereich hat sie?
- Wie werden Daten dargestellt, um Deutung zu unterstützen (31.12.2013 für Datum; (030) 1234568 für Telefonnummer)?
- Wie viel Speicher wird zur Darstellung benötigt?
- Welche Operationen sind auf Daten erlaubt?
- Wie kann die Nutzung erfolgen?

Quelle: [1]

Grundlagen

Aus den vorigen Ausführungen folgt,

- dass Daten im Programm genauer „bekannt zu machen“ sind; man sagt auch "zu deklarieren" sind
- dass dabei die Art der Daten angeben werden muss, d.h. der "Datentyp"
- dass dabei auch der Speicherbereich bezeichnet werden muss, wo die Daten abzulegen sind
- Daten müssen in den Speicherbereich "gelegt" und "herausgeholt" werden können

Quelle: [1]

Grundlagen

So ist eine Verarbeitung nicht möglich

So nicht!

```
10042014
```

So ist eine Verarbeitung möglich, denn es ist klar worum es sich handelt

Auftragsdatum	Datum
10.04.2014	

Kontonummer	Zahl
10042014	

Kontostand	Währungsbetrag
10042014	

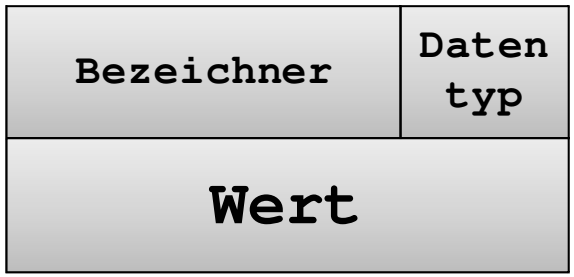
Quelle: nach [1]

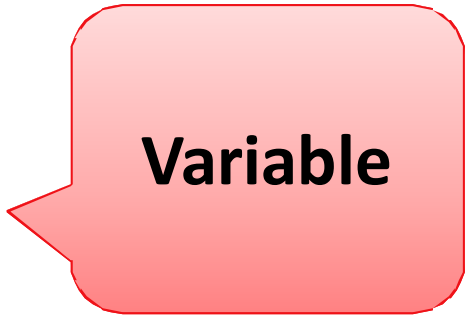
Überblick

Ein einfaches Datenelement, das

- eine festgelegte Datenart (Datentyp) und
- einen Namen (Bezeichner) hat und
- man nutzen kann, um Werte zu speichern, zu lesen und zu ändern

nennen wir Variable





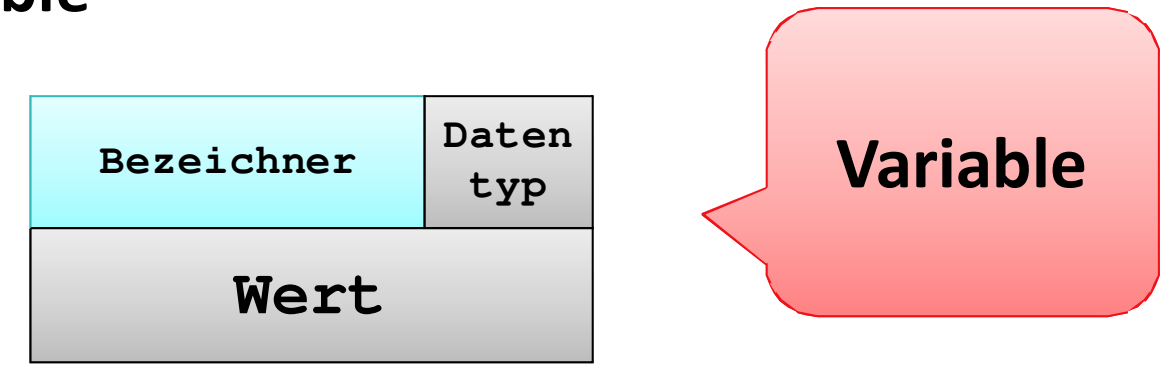
und müssen es dem Programm bekannt machen, um damit arbeiten zu können (Deklaration)

Bezeichner

Ein einfaches Datenelement, das

- eine festgelegte Datenart (Datentyp) und
- einen Namen (Bezeichner) hat und
- man nutzen kann, um Werte zu speichern, zu lesen und zu ändern

nennen wir Variable



und müssen es dem Programm bekannt machen, um damit arbeiten zu können (Deklaration)

Bezeichner

Zulässige sind

- Folge von 255 Zeichen (Unicode)
- erstes Zeichen Buchstabe
- nachfolgend auch _ und Ziffern möglich
- keine Leer- oder sonstigen Sonderzeichen
- kein reserviertes Schlüsselwort und
- Groß- und Kleinschreibung wird nicht unterschieden (Besonderheit von VBA!)
- innerhalb Gültigkeitsbereich (z.B. Prozedur) eindeutig

Beispiele zulässiger (gültiger) Bezeichner

```
i
meinAlter
strName
r2d2
c3po
```

Nicht zulässige (ungültige) Bezeichner

```
Sub
mein Alter
2r2d
fertig?
```

Bezeichner für Variablen (Teil 1)

Aussagekräftige Bezeichner

- Erhöhung der Lesbarkeit und Verständlichkeit
- Erleichtern Rückschlüsse auf Verwendungszweck
- Verbesserung der Wartungsfreundlichkeit
- sinnvolle und aussagekräftige Abkürzungen
- beginnend mit großem Buchstaben,
- dann in CamelCase-Schreibweise
- Beispiele: **`GebDat`**, **`PlzOrt`**, **`KreisDurchm`**

Deklaration (Teil 1): Bezeichner

Variablen werden dem Programm bekannt gemacht, indem Sie mit ihrem Bezeichner deklariert werden

- Generell

```
' Generelle Syntax
' Deklaration (Teil 1)
Dim <VariablenBezeichner>
```

- Beispiele

```
Dim i
Dim Name
Dim KundenNr
Dim KundeTel
Dim PlzOrt
Dim GebDat
```

Deklaration (Teil 1) Bezeichner: Beispiel 03.01

Ziel

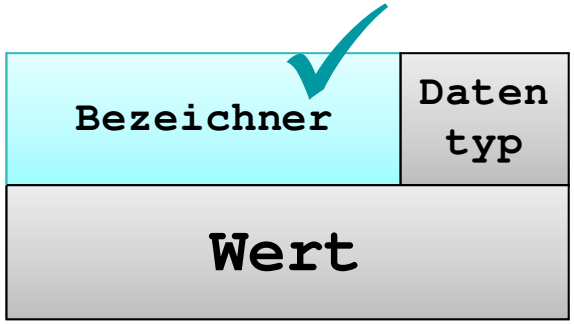
- Deklarieren Sie Variablen mit gültigen Bezeichnern für
  - Produktbeschreibung
  - Matrikelnummer
  - Verkaufspreis
  - Einkaufspreis
  - Aktenzeichen
  - Personalnummer
  - Bankleitzahl
  - Kontonummer
- Versuchen Sie Variablen mit ungültigen Bezeichnern zu deklarieren. Was passiert?

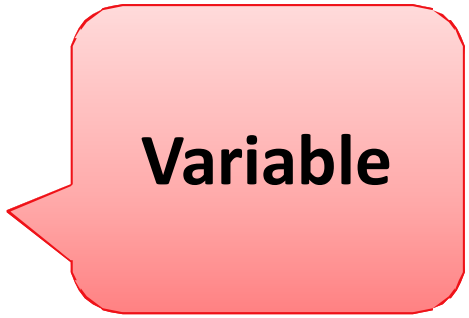
Deklaration (Teil 1) und Bezeichner

Ein einfaches Datenelement, das

- eine festgelegte Datenart (Datentyp) und
- einen Namen (Bezeichner) hat und
- man nutzen kann, um Werte zu speichern, zu lesen und zu ändern

nennen wir Variable





und müssen es dem Programm bekannt machen, um damit arbeiten zu können (Deklaration)

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

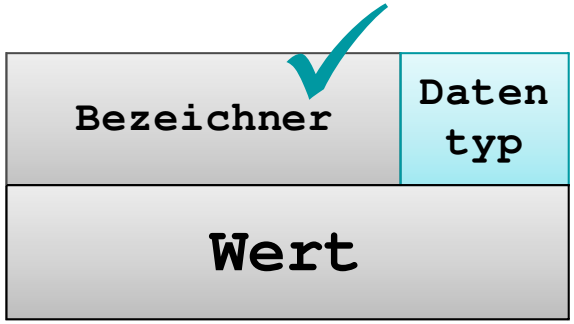
Abschluss und Ausblick

Datentypen

Ein einfaches Datenelement, das

- eine festgelegte Datenart (Datentyp) und
- einen Namen (Bezeichner) hat und
- man nutzen kann, um Werte zu speichern, zu lesen und zu ändern

nennen wir Variable



und müssen es dem Programm bekannt machen, um damit arbeiten zu können (Deklaration)

Datentypen in VBA

Einfache Datentypen für

- Natürliche und ganze Zahlen
- Reelle und rationale (gebrochene) Zahlen
  - Gleitkommazahlen
  - Festkommazahlen
- Wahrheitswerte
- Datum und Uhrzeit
- Zeichenketten

Komplexe Datentypen

- siehe LE06

Datentypen/-strukturen für Mengen und Felder

- siehe LE06

VBA-Datentypen für natürliche/ganze Zahlen

Typen

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
LongLong	8 Byte	-9.223.372.036.854.775.808	9.223.372.036.854.775.807

Beispielcode

```
Dim ganzeZahl As Integer
Dim auchGanzeZahl As Long
```

Operatoren

Operator	Benennung	Beispiel
+	Addition	5+6 ergibt 11
-	Subtraktion	9-3 ergibt 6
*	Multiplikation	9*8 ergibt 72

VBA-Datentypen für natürliche/ganze Zahlen

Operatoren (Fortsetzung)

Operator	Benennung	Beispiel
/	Division	3/4 ergibt 0,75
\	Ganzzahlen-Division	7\2 ergibt 3
Mod	Restwert (Modulo)	7 Mod 2 ergibt 1
-	Negation	-3 ergibt -3
^	Potenz	2^2 ergibt 4 3^3^3 ergibt 19.683

Vergleichsoperator	Benennung	Beispiel
<	Kleiner	3<5 ergibt True
<=	Kleiner gleich	3<=5 ergibt True
>	Größer	2>9 ergibt False
>=	Größer gleich	5>=3 ergibt True
=	Gleich	3=3 ergibt True
<>	Ungleich	3<>3 ergibt False

VBA-Datentypen für Gleitkommazahlen

Typen

Typ	Speicherbedarf	Negative Werte	Positive Werte
Single	4 Bytes	-3,402823E38 bis -1,401298E-45	1,401298E-45 bis 3,402823E38
Double	8 Bytes	-1,79769313486231E308 bis -4,94065645841247E-324	4,94065645841247E-324 bis 1,79769313486232E308

Beispielcode

```
Dim gleitkommaZahl As Single
Dim genauereGleitkommaZahl As Double
```

Operatoren

- Wie ganze Zahlen

Hinweis: Double etwa doppelte Genauigkeit von Single, z.B.

- als Double darstellbar: 3,14159265358979
- als Single darstellbar: 3,141593

Rundung

VBA-Datentypen für Festkommazahlen

Typen

Typ	Speicher-bedarf	Wertebereich
Currency	8 Bytes	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807

Beispielcode

```
Dim waehrungsBetrag As Currency
```

Operatoren

– Wie ganze Zahlen

Hinweis

– Sinnvolle Verwendung für Währungsbeträge!

VBA-Datentypen für gebrochene Zahlen

Formen der Zahlendarstellung

- Festkommazahlen
  - hat eine begrenzte Anzahl von Stellen
  - das Komma steht immer an der gleichen, festgelegten Stelle
- Gleitkommazahlen
  - hat eine begrenzte Anzahl von Stellen
  - Position des Kommas innerhalb der Zahl bedarfsgerecht festgelegt
  - Aufteilung der Zahl **x** in eine Mantisse **m** und einen Exponenten **e**

$$x = m * 10^{e}$$

  - Beispiel (aus Physik-Unterricht): Lichtgeschwindigkeit **c** im Vakuum

$$\begin{aligned} c &= 299792458\ m/s \\ &= 229792{,}458 * 10^{3}\ m/s \\ &= 2{,}99792458 * 10^{8}\ m/s \end{aligned}$$

VBA-Datentypen für gebrochene Zahlen

Beispiele

Zahl	Als Gleitkommazahl vom Typ Single	Als Festkommazahl vom Typ Currency
3.1415758483947	3.141576	3.1416
31.415758483947	31.41576	31.4158
314.15758483947	314.1576	314.1576
3141.5758483947	3141.576	3141.5758
31415.758483947	31415.76	31415.7585
3141575.8483947	3141576	3141575.8484
31415758.483947	3.141576E+07	31415758.4839
Fazit	**"Komma" gleitet durch die Zahl, Genauigkeit der Nachkomma-stellen hängt von der Position des Kommas ab**	**"Komma" steht immer an fester Position und garantiert eine feste Genauigkeit der Nachkommastellen**

VBA-Datentyp für Wahrheitswerte

Typ: Boolean

Typ	Speicherbedarf	Wertemenge
Boolean	2 Bytes	True oder False

Beispielcode

```
Dim ergebnisOk As Boolean
```

Wertemenge

Wert	Bedeutung
true	Wahr, Ja
false	Falsch, Nein

Hinweise

- Boolean repräsentiert eine Menge von Werten und keinen Bereich, denn es gibt "zwischen" true und false keine weiteren Werteausprägungen.
- Wir verwenden ausschließlich True und False und **nicht** deren Abbildung auf Integer-Werte (-1 für True bzw. 0 für False)

VBA-Datentyp für Wahrheitswerte

Operatoren

Operator	Bezeichnung	Beispiel
=	Gleich	True = False ergibt False
<>	Ungleich	True <> False ergibt True
And	Logisches Und	True And True ergibt True True And False ergibt False
Or	Logisches Oder	True Or True ergibt True True Or False ergibt True False Or False ergibt False
Xor	Exklusives Oder	False Xor True ergibt True True Xor False ergibt True True Xor True ergibt False False Xor False ergibt False
Not	Negation	Not True ergibt False Not False ergibt True

VBA-Datentyp für Datum und Zeit

Typ

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

Beispielcode

```
Dim gebDat As Date
' Im US-Datumsformat zuweisen mit #
Let gebDat = #10/13/1985# '13. Oktober 1985
' oder über den "Umweg" eines Strings (implizite Typumw.)
Let gebDat = "13.10.1985" 'auch 13. Oktober 1985
```

Operatoren

- Inkrement und Dekrement, d.h. tageweises Hochzählen bzw. Herunterzählen ("Addition und Subtraktion von Tagen")
- Vergleichsoperatoren

VBA-Datentyp für Datum und Zeit

Hilfsfunktionen (von VBA bereitgestellt)

Funktion	Par.-Typ	Aufgabe	Beispiel
Month	Date	Monat erfragen	`Month(gebDat)` ergibt `10`
Day	Date	Tag erfragen	`Day(gebDat)` ergibt `13`
Year	Date	Jahr erfragen	`Year(gebDat)` ergibt `1985`
Weekday	Date	Wochentag erfragen	`Weekday(gebDat)` ergibt `1` [für Sonntag]
TimeValue	Date	Tageszeit erfragen	`TimeValue(gebDat)` ergibt `17:05:33`
Now	-	Tagesdatum erfragen	`Now()` ergibt `10.04.2014 12:35:36`

DateDiff	String, 2x Date	Tage, Monate, Jahre usw. zwischen Datum 1 und Datum 2	`DateDiff("d", dat1, dat2)`
			`DateDiff("m", dat1, dat2)`
			`DateDiff("yyyy", dat1, dat2)`

VBA-Datentyp für Zeichenketten

Typ

Typ	Speicherbedarf	Wertebereich
String	1 Byte je Zeichen (und bei variabler Länge zzgl. 10 Bytes)	< 2,1 Mrd. Zeichen bei variabler Länge < ca. 65.400 Zeichen bei fester Länge

Beispielcode

```
Dim zeichenKette As String
```

Operatoren

Operator	Benennung	Beispiel
&	Verkettung	"Com" & "puter" ergibt "Computer"

VBA-Datentyp für Zeichenketten

Hilfsfunktionen (von VBA bereitgestellt)

– für einzelne Zeichen

Funktion	Par.-Typ	Aufgabe	Beispiel
Chr	Byte	Umwandlung des Zeichencodes in ein Zeichen	Chr(77) ergibt "M"
Asc	String	Ermittlung des Zeichencodes (des ersten Zeichens eines Strings)	Asc("a") ergibt 97 Asc("abc") ergibt 97

– für Zeichenketten

Funktion	Par.-Typ	Aufgabe	Beispiel
Len	String	Ermittlung Stringlänge	Len("Hallo Welt") ergibt 10
Left, Right	String, Byte	Ermittlung linker bzw. rechter Teil	Left("Hallo", 2) ergibt "Ha" Right("Welt", 2) ergibt "lt"
Mid	String, Von als Byte, Länge als Byte	Ermittlung mittlerer Teil	Mid("Hallo Du", 4, 2) ergibt "lo"

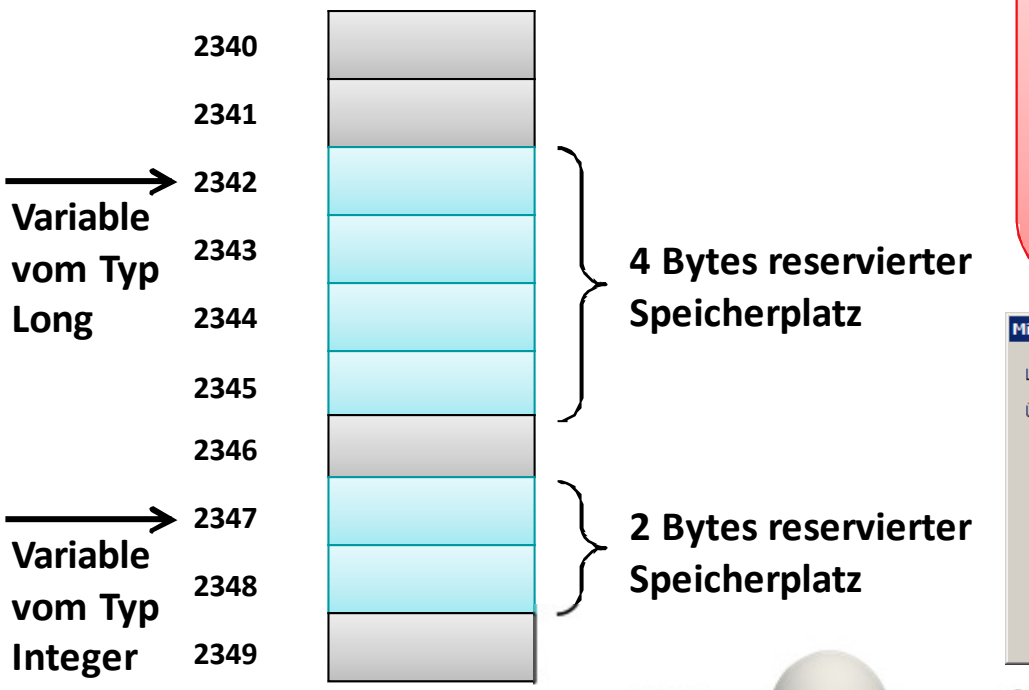
VBA-Datentyp für Zeichenketten

Hilfsfunktionen (von VBA bereitgestellt)

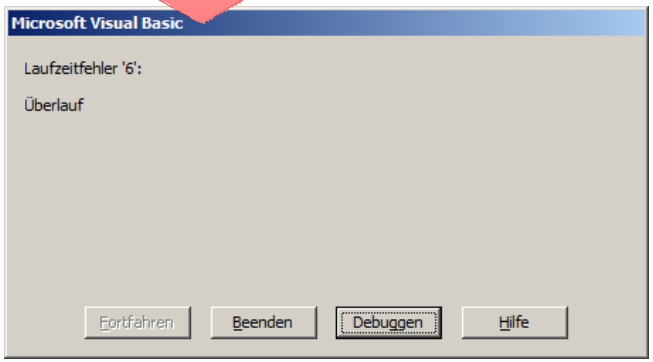
– für Zeichenketten (Fortsetzung)

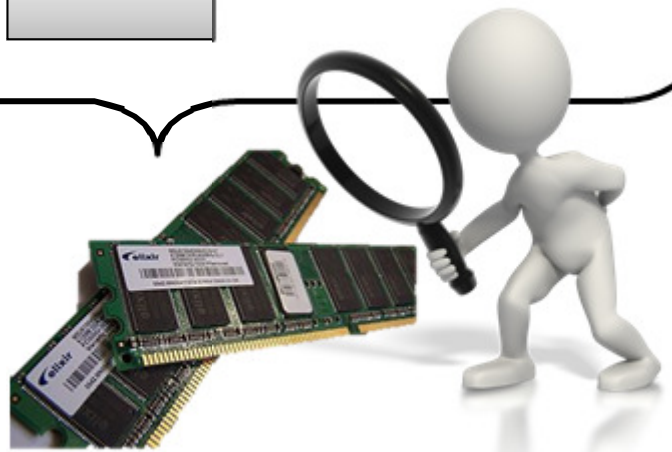
Funktion	Par.-Typ	Aufgabe	Beispiel
Ltrim	String	Führende Leerzeichen entfernen	Ltrim(" Welt ") ergibt "Welt "
Rtrim	String	Endende Leerzeichen entfernen	Rtrim(" Welt ") ergibt " Welt"
Trim	String	Führende und endende Leerzeichen entfernen	Trim(" Welt ") ergibt "Welt"
Instr	Basis als String, Teil als String	Position von Teil in Basis	Instr("Hallo Welt", "lo") ergibt 4
Val	String	Liefert den numerischen Wert der Zeichenkette	Val("42") ergibt 42 Val("0815") ergibt 815 Val("0.07") ergibt 0,07 Val("") ergibt 0 Val("Hallo") ergibt 0

VBA-Datentypen und Speicher



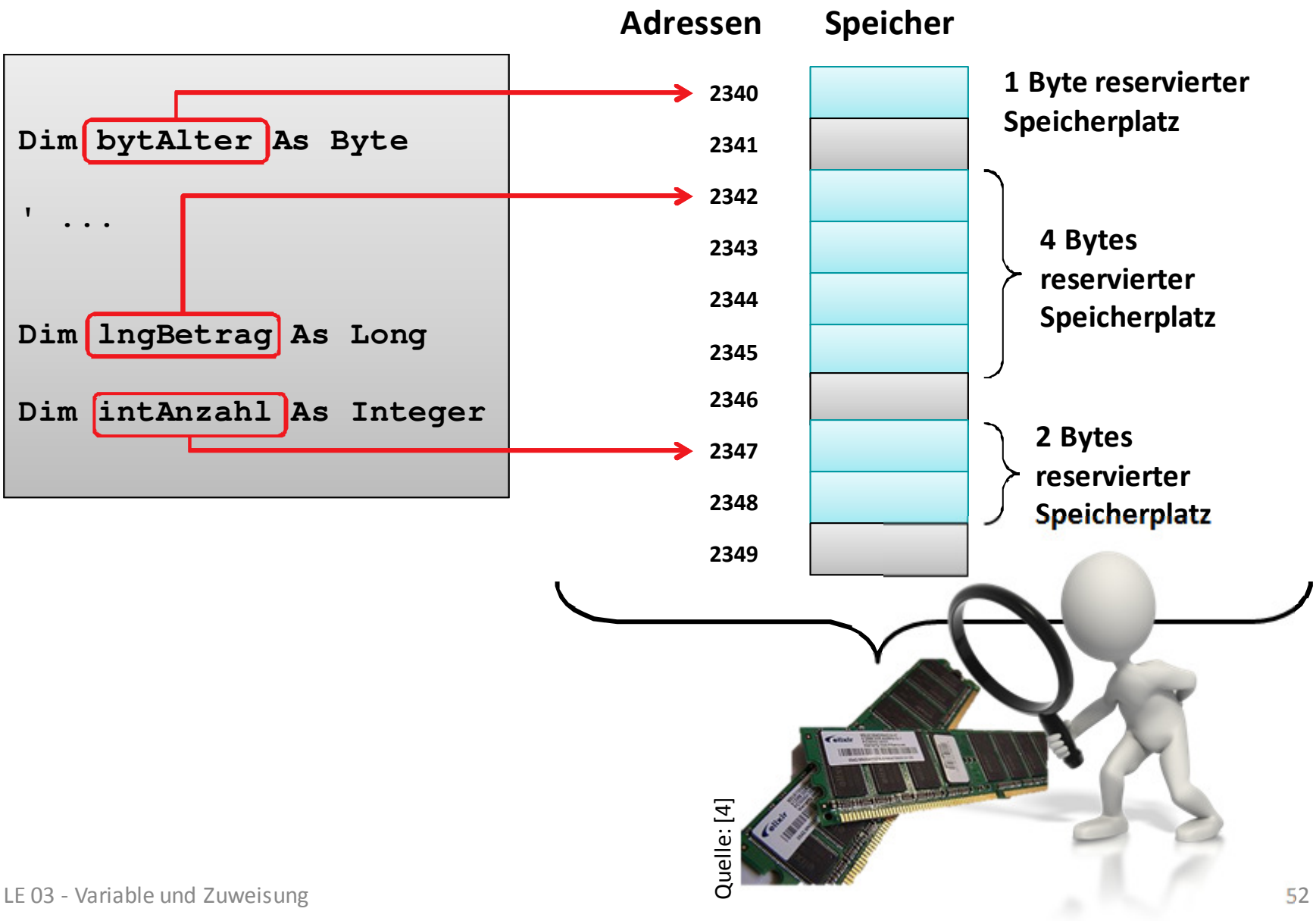
Ein "Überlauf" tritt auf, wenn Wert zugewiesen wird, der mehr Speicherplatz benötigt, als für Variable vorgesehen ist.





Quelle: [4]

Bezeichner und Datentyp



Quelle: [4]

Deklaration (Teil 2): Bezeichner und Datentyp

Variablen werden dem Programm bekannt gemacht, indem Sie mit ihrem Bezeichner und Datentyp deklariert werden

- Generell

```
' Generelle Syntax
' Deklaration
Dim <VariablenBezeichner> As <DatentypBezeichner>
```

- Beispiele

```
Dim i As Integer
Dim Name As String
Dim KundenNr As Long
Dim KundeTel As String
Dim PlzOrt As String
Dim GebDat As Date
```

Namenskonvention für Variablenbezeichner

Im Rahmen dieses Kurses Anlehnung an "Ungarische Notation"

- In der Praxis sind Konventionen für die Bezeichnung von Variablen verbreitet
- Aufbau: ***<Präfix><Variablenbezeichner>***
- Präfix: 3 Kleinbuchstaben geben Auskunft über Typ (z.B. **`int`** für **`Integer`**, **`str`** für **`String`**)

Beispiele

```
Dim strName As String
Dim lngKundenNr As Long
Dim strKundeTel As String
Dim strPlzOrt As String
Dim datGebDatum As Date
' Ausnahmen (für Hilfsvariablen, Details später):
Dim i As Integer
```

Namenskonvention für Variablenbezeichner

Beispiele für Variablen mit VBA-Datentypen

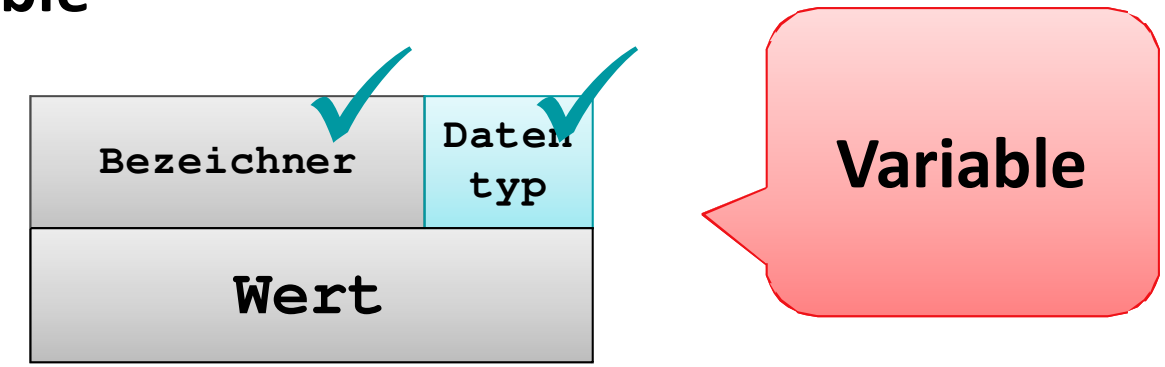
Präfix	Datentyp	Beispiele
int	Integer	intMonatsGehalt, intZaehler
lng	Long	lngJahresGehalt
byt	Byte	bytAlter
sgl	Single	sglMonatsGehalt
dbl	Double	dblBreite
cur	Currency	curUmsatz
dec	Decimal	decStart
dat	Date	datStart, datEnde, datGeb
bol	Boolean	bolBestanden, bolBrille
str	String	strName, strVorname

Datentypen

Ein einfaches Datenelement, das

- eine festgelegte Datenart (Datentyp) und
- einen Namen (Bezeichner) hat und
- man nutzen kann, um Werte zu speichern, zu lesen und zu ändern

nennen wir Variable



und müssen es dem Programm bekannt machen, um damit arbeiten zu können (Deklaration)

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

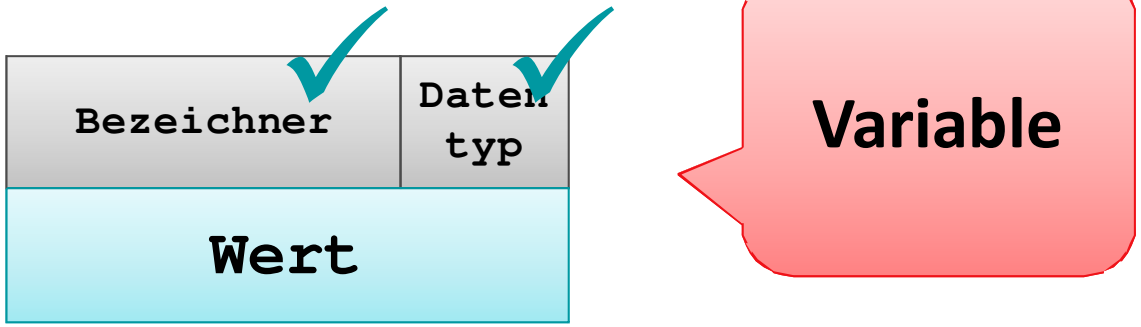
Inhalt

Wert, Ausdruck und Zuweisung

Ein einfaches Datenelement, das

- eine festgelegte Datenart (Datentyp) und
- einen Namen (Bezeichner) hat und
- man nutzen kann, um Werte zu speichern, zu lesen und zu ändern

nennen wir Variable



und müssen es dem Programm bekannt machen, um damit arbeiten zu können (Deklaration)

Wert, Ausdruck und Zuweisung

Wert

- Ausdruck, der nicht weiter ausgewertet werden kann
- Literal genau ein Wert (Element) aus der Wertemenge eines Datentyps
- Beispiele
  - 42 z.B. Wert für eine ganze Zahl
  - "zweiundvierzig" z.B. Wert für eine Zeichenkette
  - "42" z.B. Wert für ein Zeichen einer Zeichenkette
  - #12/24/2016#, z.B. Weihnachten

Wert, Ausdruck und Zuweisung

Wert

- Gelegentlich ist unklar, von welchem Typ ein Wert sein soll
  - Ist 42 Byte, Integer oder Long?
  - Soll 3.14 als Single oder Double verwendet werden?
- ... dann Verwendung von Typkennzeichen notwendig

Datentyp	Kennzeichen	Beispiele
Byte	- Kein -	
Integer (Standard)	% und ohne	42%, 42
Long	&	42&
LongLong	^	42^
Single	!	3.1!
Double (Standard)	# und ohne	3.1#, 3.1
Currency	@	3.1@

```
Direktbereich
? TypeName(42)
Integer
? TypeName(42%)
Integer
? TypeName(42&)
Long
? TypeName(3.1)
Double
? TypeName(3.1#)
Double
? TypeName(3.1!)
Single
? TypeName(3.1@)
Currency
```

Wert, Ausdruck und Zuweisung

Ausdruck

- Kombination aus Werten, Operatoren, Variablen oder Funktionen
- kann nach einer definierten Berechnungsregel ausgewertet werden (d.h. sein Wert kann berechnet werden)
  - setzt voraus, dass Operanden und Operatoren sinnvoll zu einander passen
- kann ausgewertet und ausgegeben werden,
  - mit der der Debug.Print-Anweisung oder
  - testweise umittelbar im Direktbereich mittels "**?**"-Anweisung und abschließender Enter-Taste
- kann innerhalb von Zuweisungen verwendet werden

Wert und Ausdruck: Beispiel 03.03

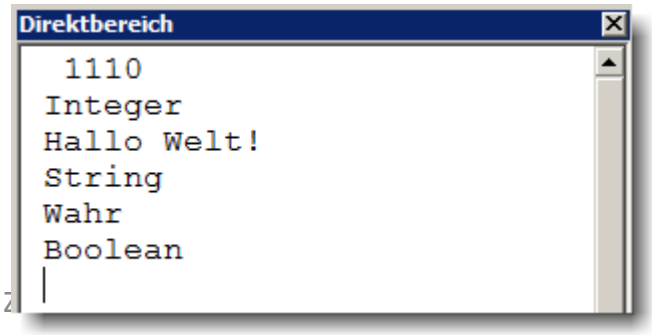
Ziel

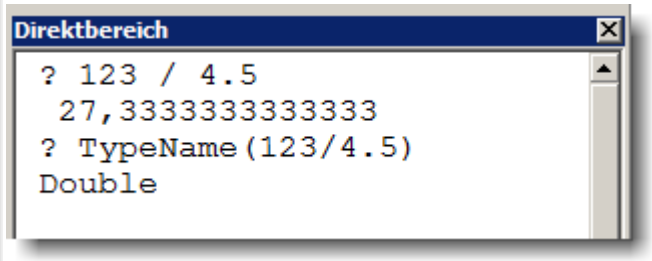
- Auswertung von Ausdrücken und
- Ausgabe des Ergebniswertes im Direktbereich

Aufgabe

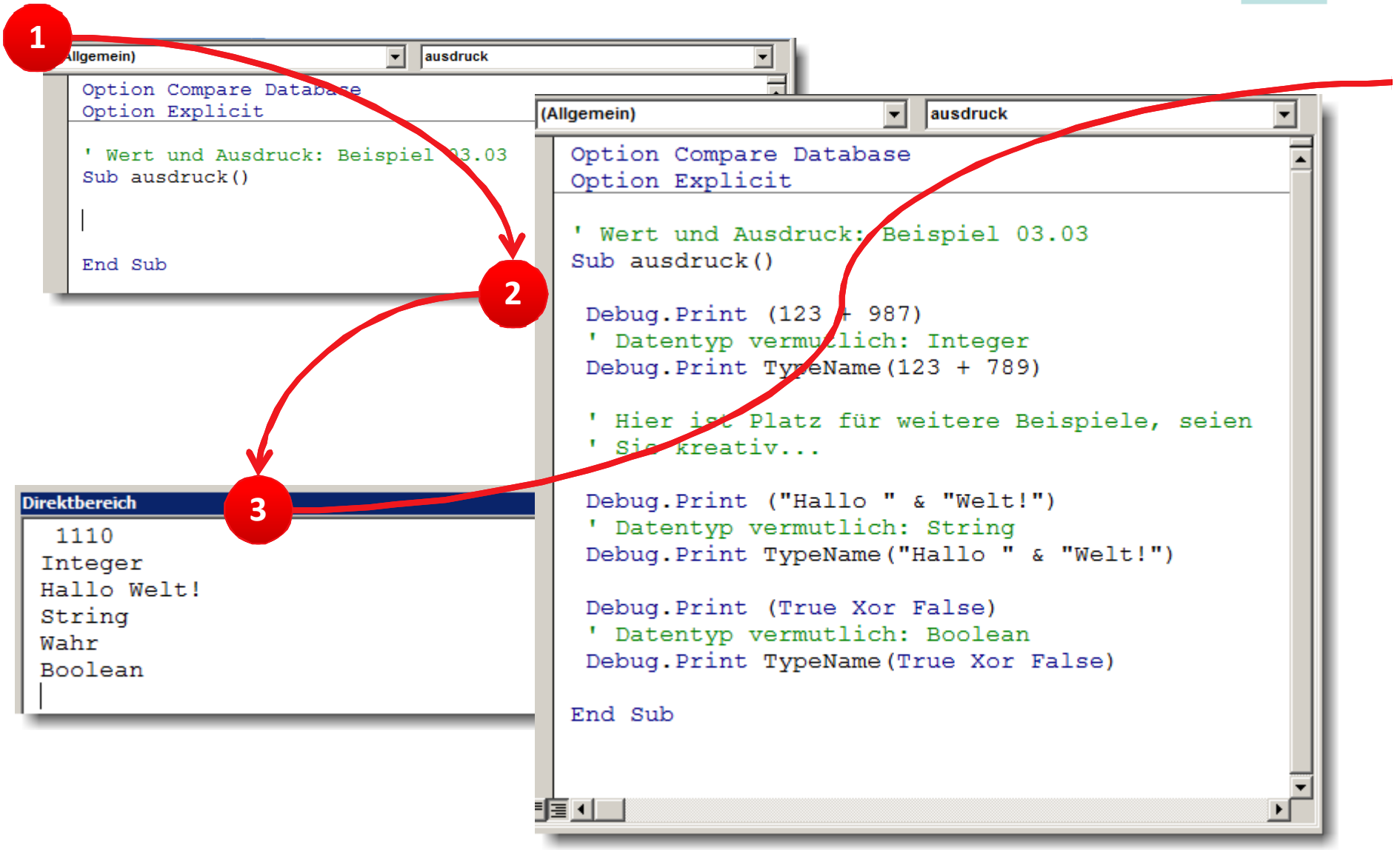
- Ausdrücke mit Debug.Print auswerten und ausgeben und Ausdrücke im Direktfenster mittels "**?**"-Operator auswerten
  - Mathematische Operationen auf ganzen Zahlen, z.B.: **123** + **987**
  - Verkettung von Zeichenketten, z.B. "Hallo " und "Welt!"
  - Logische Operationen, z.B. **true xor false**
- Überlegungen zum Datentyp des Ergebnisses und Prüfung der Überlegung mit VBA-Funktion **TypeName(*<Ausdruck>*)**

Beispiel

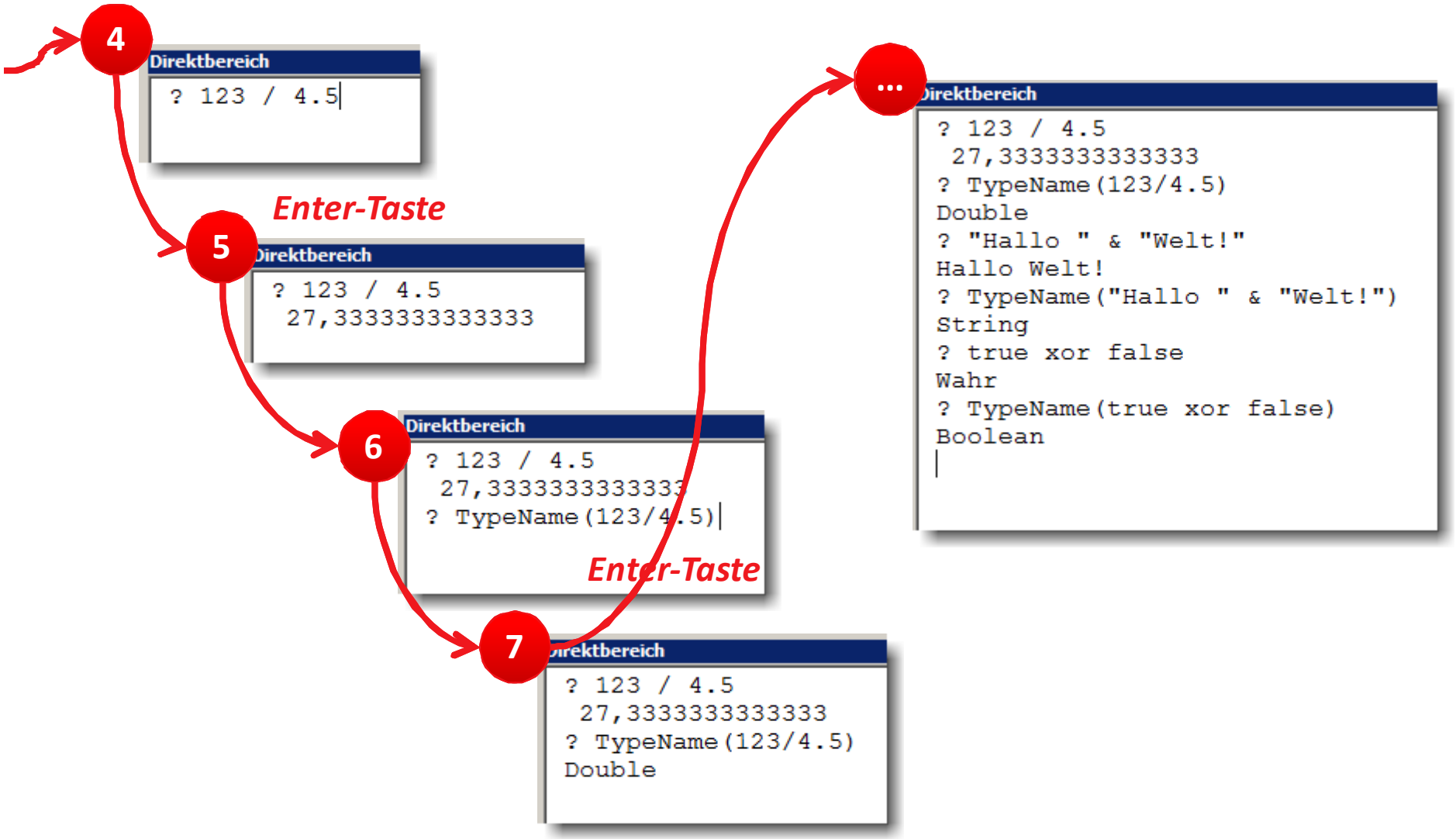




Wert und Ausdruck: Beispiel 03.03



Wert und Ausdruck: Beispiel 03.03



Wert und Ausdruck: Beispiel 03.03

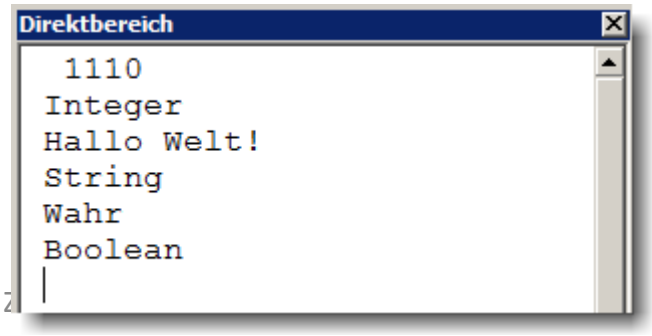
Ziel

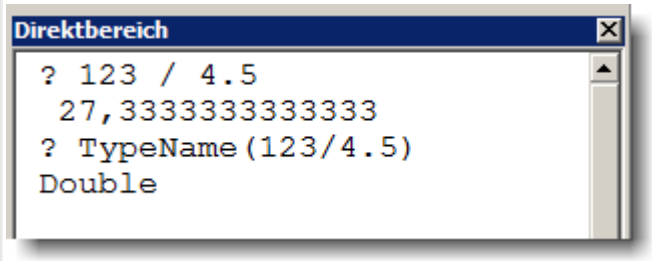
- Auswertung von Ausdrücken und
- Ausgabe des Ergebniswertes im Direktbereich

Aufgabe

- Ausdrücke mit Debug.Print auswerten und ausgeben und Ausdrücke im Direktfenster mittels "**?**"-Operator auswerten
  - Mathematische Operationen auf ganzen Zahlen, z.B.: **123 + 987**
  - Verkettung von Zeichenketten, z.B. "Hallo " und "Welt!"
  - Logische Operationen, z.B. **true xor false**
- Überlegungen zum Datentyp des Ergebnisses und Prüfung der Überlegung mit VBA-Funktion **TypeName(*<Ausdruck>*)**

Beispiel





Arten von Operatoren

Operatoren durch Datentypen festgelegt und benutzt, um Werte zu berechnen, z.B.

- Arithmetische Operatoren (z.B. `+, -`)
- Vergleichsoperatoren (z.B. `>, <, =`) liefern als Ergebnis Wahrheitswert
- Logische Operatoren (z.B. `or`, `and`, `xor`) verknüpfen Wahrheitswerte)

Zuweisungsoperatoren (für alle Datentypen), um Ergebnis einer Berechnung in einem Datenelement (Variable) zu speichern

- Operatorsymbol **=**
- Verwechselungsgefahr mit Vergleichsoperator!

Merkmale von Operatoren

Anzahl der Operanden

- in der Regel 1 oder 2
- Beispiele: Addition mit zwei Operanden: 2 + 3 und Negation mit einem Operanden: -1

Notation

- Präfix-, Infix- und Postfixnotation (wo steht der Operator)
- Beispiele: Multiplikation als Infixnotation: 2 * 3 und Negation als Präfixnotation: -2

Assoziativität

- Richtung der Auswertung in VBA immer von links nach rechts, d.h. treten in einem Ausdruck gleiche Operatoren auf, wird von links nach rechts ausgewertet
- Ausnahme: Zuweisungsoperator (immer rechts auswerten, dann links zuweisen)

Priorität

- welche Operationen werden zuerst ausgeführt (analog zur Regel „Punktrechnung vor Strichrechnung“)
- Ausdrücke in Klammern werden vor Anwendung der Operatorprioritäten ausgewertet

Operatorprioritäten

Operatoren	Operator-symbole	Auswertungs-reihenfolge (Priorität)	Operatoren	Operator-symbole	Auswertungs-reihenfolge (Priorität)
Potenz	^	1	Konkatination von Zeichenketten	&	7
Negation	-	2	Vergleich	=, >, <, <=, >=, <>	8
Multiplikation, Division	*, /	3	Logische Negation	Not	9
Ganzzahlige Division	\	4	Logisches Und	And	10
Modulo	Mod	5	Logisches Oder	Or	11
Addition, Subtraktion	+, -	6	Exklusives Oder	Xor	12
			Zuweisung	=	99

Quelle: [2], S. 11 f. erweitert um [7]

Zuweisung

Syntax

- Optionale Anweisung **`Let`**
- Symbol **=** als Zuweisungsoperator

```
Let <Variable> = <Ausdruck oder Wert>
```

- Ausdruck und Wert werden abhängig vom Datentyp dargestellt und entsprechend zugewiesen
  - String: in Anführungszeichen, z.B. "Hallo Welt"
  - Zahlen: ohne Anführungszeichen usw., z.B. 42, 23
  - Gebrochene Zahlen: wie Zahlen, aber mit Dezimalpunkt anstelle Komma, z.B. 123.45
  - Datum
    - im amerikanischen Format und in #-Zeichen, z.B. #12/14/2014#
    - über den Umweg eines Strings im deutschen Format, z.B. "14.12.2014"

Zuweisung

Beispiele

```
' Deklaration
Dim l As Long
Dim i As Integer
Dim curKontostand AS Currency
Dim curBrutto As Currency
Dim curNetto As Currency
'...

' Zuweisung von Werten
Let i = 42 ' Zuweisung einer ganzen Zahl
Let curKontoStand = 123.56 ' Zuweisung einer Kommazahl
Let strVorname = "Paul" ' Zuweisung eines String
Let datGebDat = #08/23/1992# ' Zuweisung eines Datums
' Zuweisung eines Ausdrucks
Let l = i * 3
Let curBrutto = curNetto * 1.19
```

Zuweisung

Anweisung mit zwei Seiten

- Linke Seite: gibt an, welcher Variablen der Wert zugewiesen wird
- Rechte Seite gibt an, welcher Wert zugewiesen wird oder welcher Ausdruck ausgewertet und dann zugewiesen wird
- Reihenfolge ist zu beachten (Operator ist rechts assoziativ)
  - Zuerst wird die rechte Seite vollständig ausgewertet (d.h. berechnet); erst dann wird Ergebnis der linken Seite zugewiesen
  - wichtig für Zuweisungen, die Variable der linken Seite auf der rechten Seite enthalten
  - Beispiel: Welcher Wert von a wird ausgegeben?

```
Let a = 21
Let a = a * 2
Debug.Print a
```

Linke Seite

Rechte Seite

```
42
```

Zuweisung

Erstmalige Zuweisung: Initialisierung

- 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)

Jede nach der Initialisierung folgende Zuweisung ist einfach nur eine Zuweisung.

Zuweisung

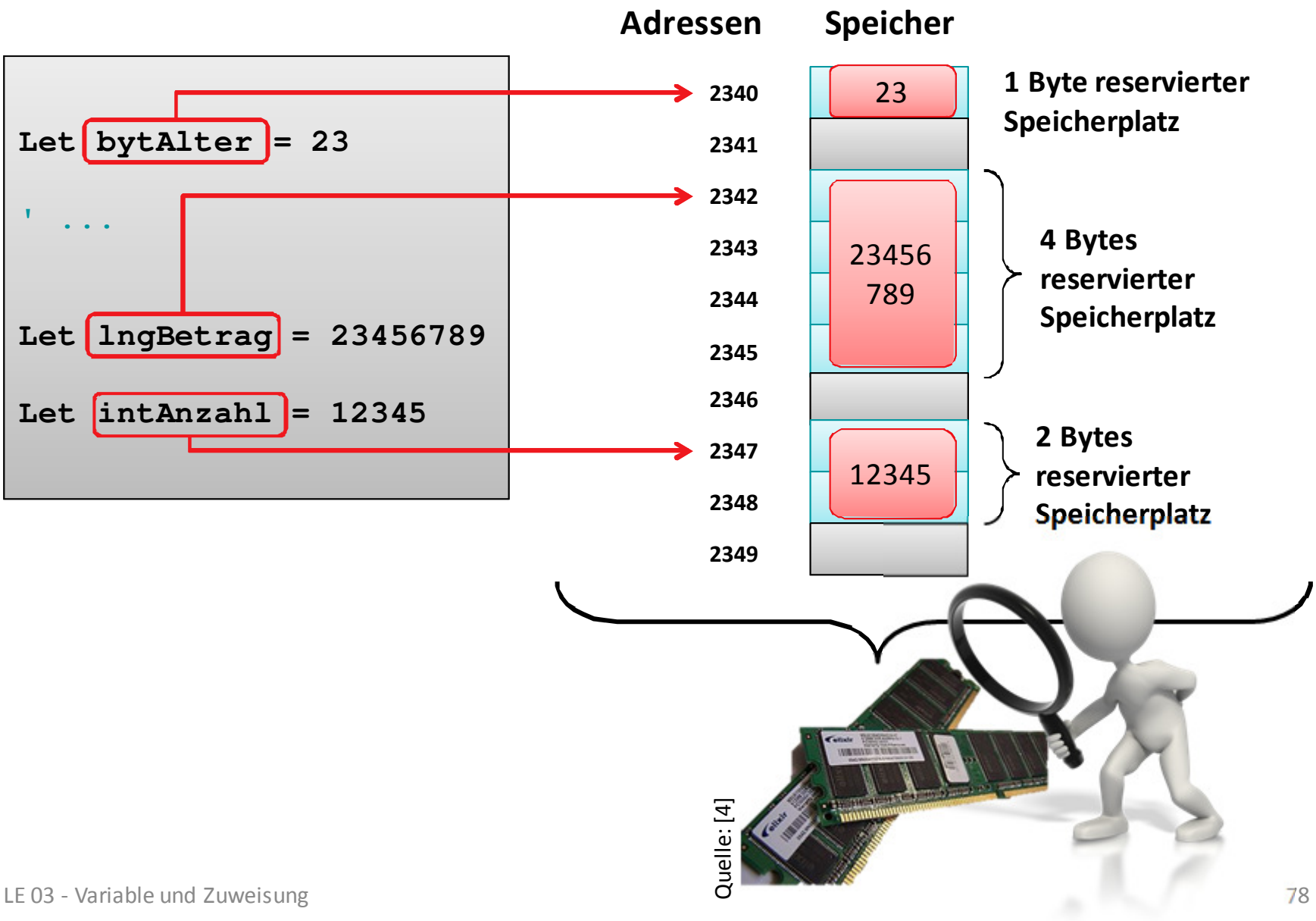
Schreibender/Ändernder Zugriff auf die Variable

- Bei Zuweisung wird in dem durch die Variable referenzierten Speicherbereich ein Wert abgelegt
- Ein bereits vorhandener Wert wird überschrieben (weil stets nur ein Wert durch die Variable gespeichert werden kann)

Lesender Zugriff auf den Wert der Variable

- Beim Lesen einer Variable wird der durch sie referenzierte Speicherbereich ermittelt
- Anschließend wird der in diesem Speicherbereich vorhandene Wert gelesen

Zuweisung



Quelle: [4]

Zuweisung: Beispiel 03.04

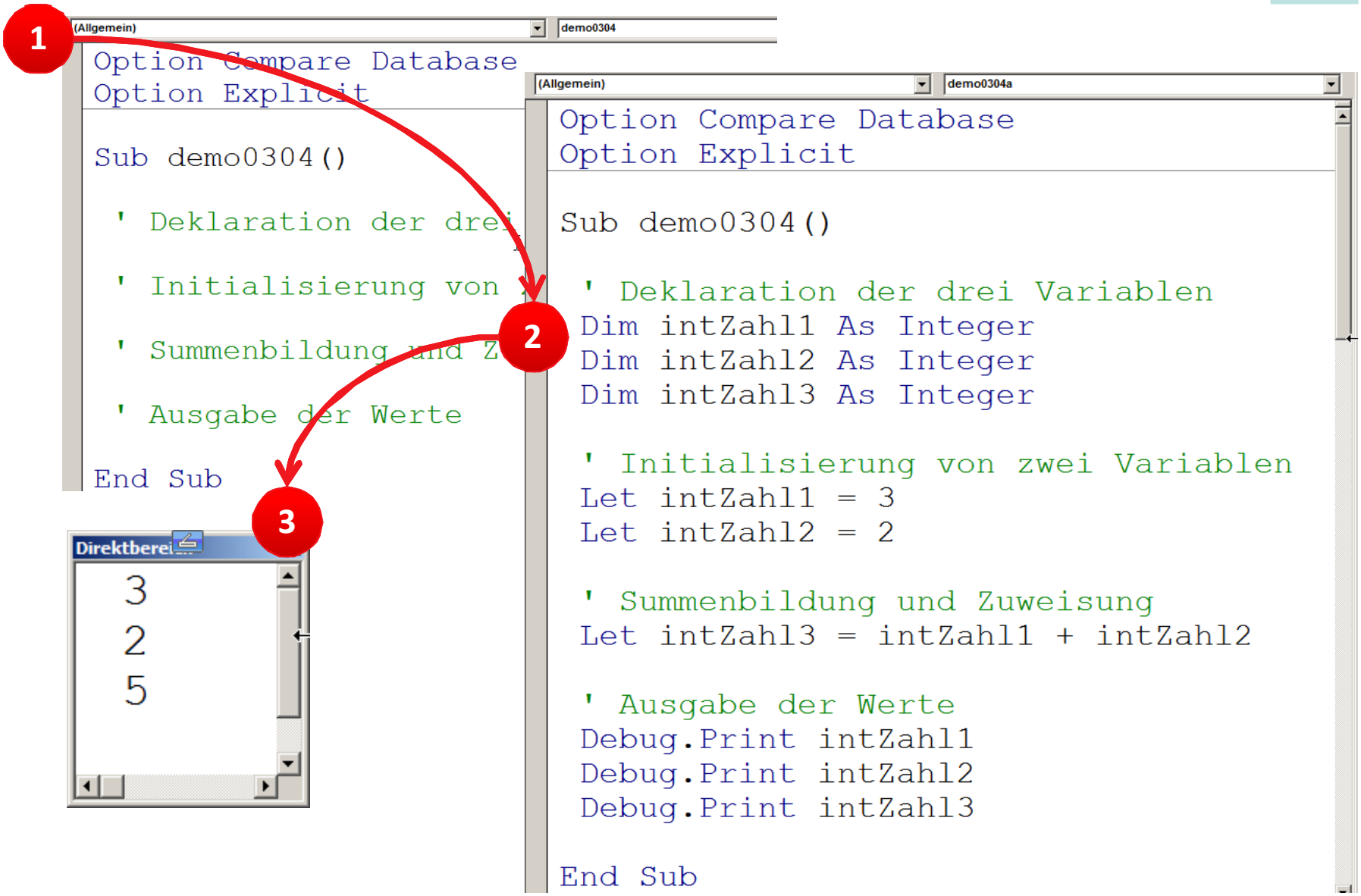
Ziel

- Deklaration, Initialisierung und Zuweisung unterscheiden

Aufgabe

- Deklaration intZahl1, intZahl2 und intZahl3 (alle Integer)
- Initialisierung von intZahl1 mit 3 und intZahl2 mit 2
- Berechnen der Summe aus intZahl1 und intZahl2
- Zuweisung des Ergebnisses der Berechnung zu intZahl3
- Ausgabe von intZahl1, intZahl2 und intZahl3

Zuweisung: Beispiel 03.04



Zuweisung: Beispiel 03.04

Ziel

- Deklaration, Initialisierung und Zuweisung unterscheiden

Aufgabe

- Deklaration von intZahl1, intZahl2 und intZahl3 (alle Integer)
- Initialisierung von intZahl1 mit 3 und intZahl2 mit 2
- Berechnen der Summe aus intZahl1 und intZahl2
- Zuweisung des Ergebnisses der Berechnung zu intZahl3
- Ausgabe von intZahl1, intZahl2 und intZahl3

Zuweisung

Verwechselungsgefahr mit Vergleichsoperator

- Operatorsymbol für Vergleich und Zuweisung ist identisch
- Negativ-Beispiel: Was ist was?

```
Sub nichtverwechseln()

 Dim a As Boolean
 Dim b As Boolean
 Dim c As Boolean

 Let a = False
 Let b = False
 Let c = False

 Let a = b = c

 Debug.Print a
 Debug.Print b
 Debug.Print c

End Sub
```

Zuweisungsoperator

Vergleichsoperator

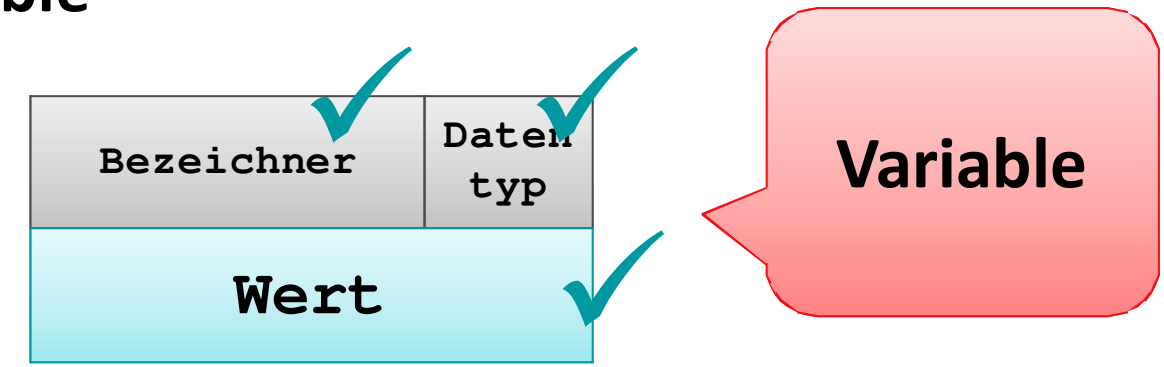
```
Direktbereich
Wahr
Falsch
Falsch
```

Wert, Ausdruck und Zuweisung

Ein einfaches Datenelement, das

- eine festgelegte Datenart (Datentyp) und
- einen Namen (Bezeichner) hat und
- man nutzen kann, um Werte zu speichern, zu lesen und zu ändern

nennen wir Variable



und müssen es dem Programm bekannt machen, um damit arbeiten zu können (Deklaration)

Inhalt

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Variable

- wird deklariert mit Schlüsselwort **Dim**
- hat einen **Bezeichner**
- und ist von einem definierten **Datentyp**
- **Werte** oder **Ausdrücke** werden ihr **zugewiesen**
- erstmalige Zuweisung heißt **Initialisierung**
- bietet **Zugriff** auf gespeicherten Wert
  - lesend
  - schreibend/ändernd

Syntax

```
Dim <VarBezeich> As <Datentyp>

Let <VarBezeich> = <Wert/Ausdr>
```

Beispiele

```
' Weniger gute Bezeichner
Dim i As Integer
Dim s As String
' Aussagekräftige Bezeichner
Dim bytAlter As Byte
Dim sglBeitrag As Single
' Initialisierung mit Wert
Let bytAlter = 20
' Initialisierung mit Ausdruck
Let sglBeitrag = bytAlter * 3
' Lesender Zugriff
Debug.Print bytAlter
' Ändernder Zugriff
Let bytAlter = bytAlter + 1
Let sglBeitrag = 42
```

Variable: Beispiel 03.01

Ziel

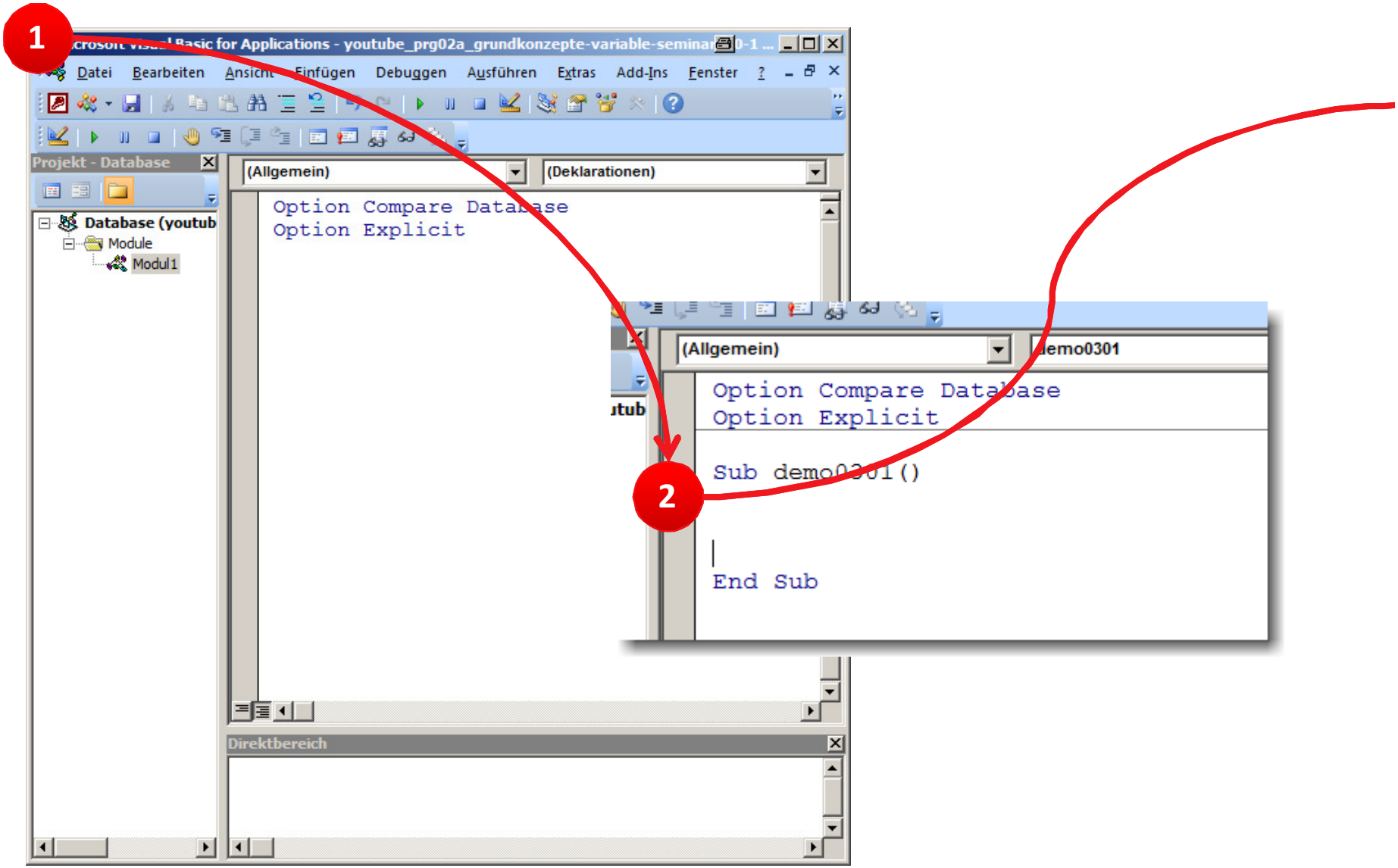
- Variablen, Datentypen, Deklaration, Zuweisung und Zugriff demonstrieren

Aufgabe

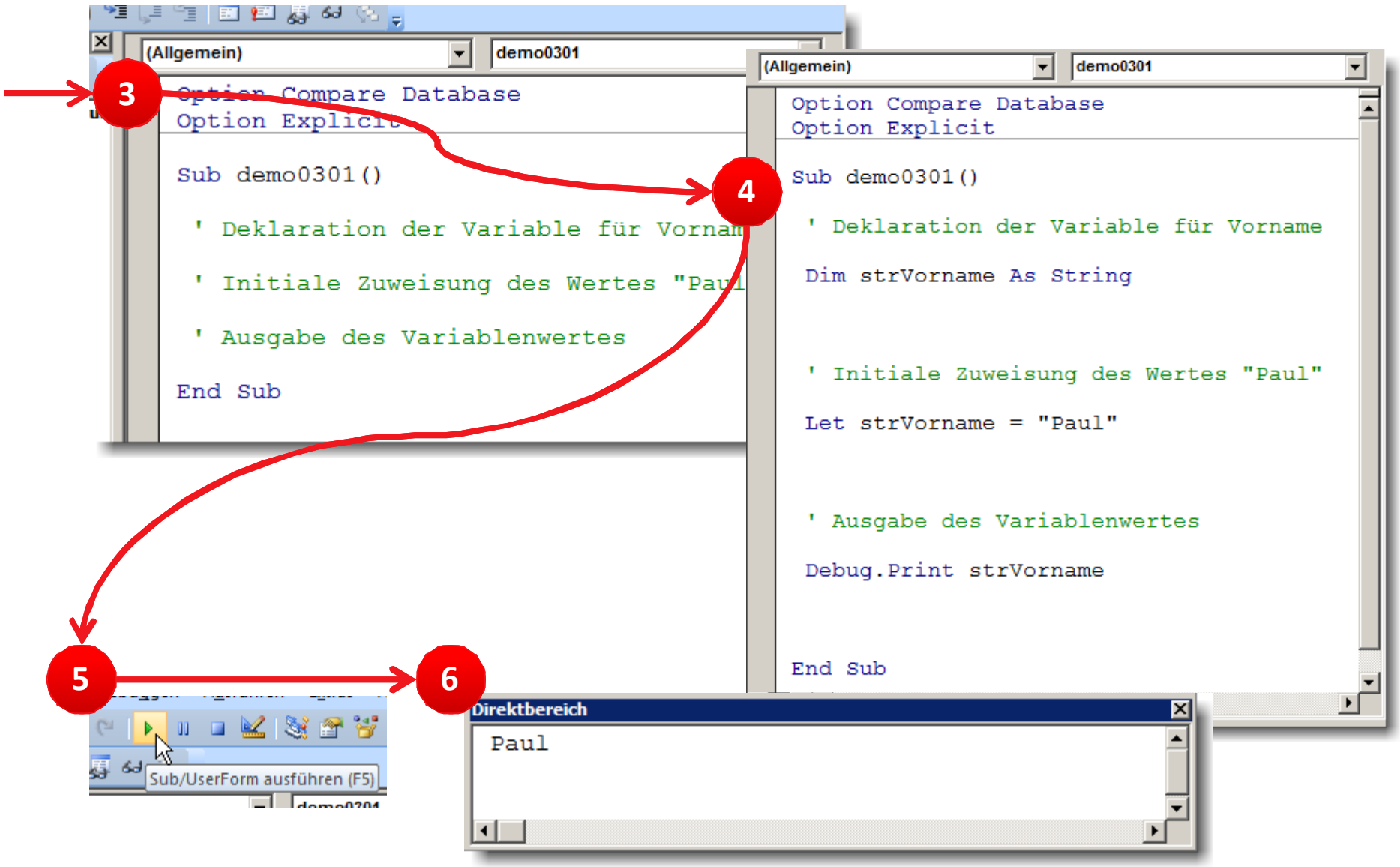
- Deklarieren einer Variable mit dem Bezeichner **`strVorname`** vom Datentyp **`String`**
- initiale Zuweisung des Wertes "Paul" zur Variable **`strVorname`**
- Lesender Zugriff auf den Wert der Variable **`strVorname`** und Ausgabe des Wertes im Direktbereich

Ergebnis

Überblick: Beispiel 03.01



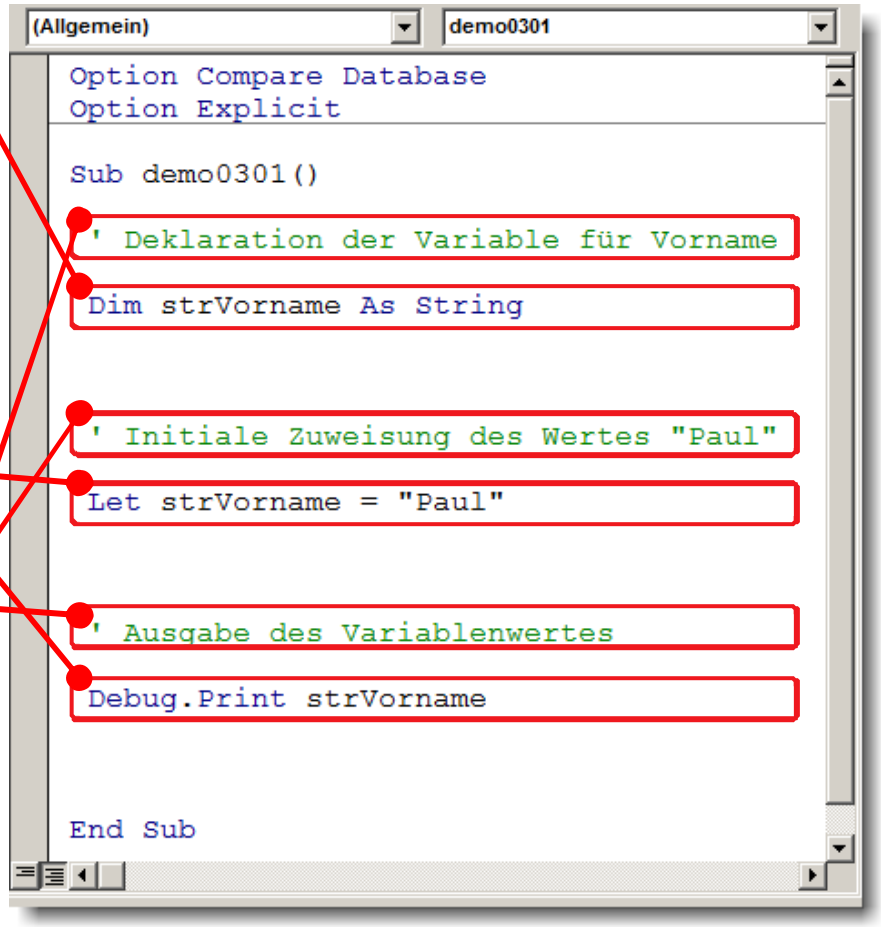
Überblick: Beispiel 03.01



Überblick: Beispiel 03.01

Bestandteile des Programms

- Deklaration einer Variable
  - mit Bezeichner **`strVorname`**
  - Schlüsselwort **`Dim`**
  - gefolgt von **`As <Datentyp>`**
- Anweisungen
  - Initiale Zuweisung eines Wertes zur Variable **`strVorname`**
  - Ausgabe des Variablenwertes
- Kommentare
  - ohne Wirkung auf den Programmablauf
  - Unterstützen Verständnis des Algorithmus
- Hinweis: Auch die Prozedur (Unterprogramm) **`demo0301`** wird hier deklariert

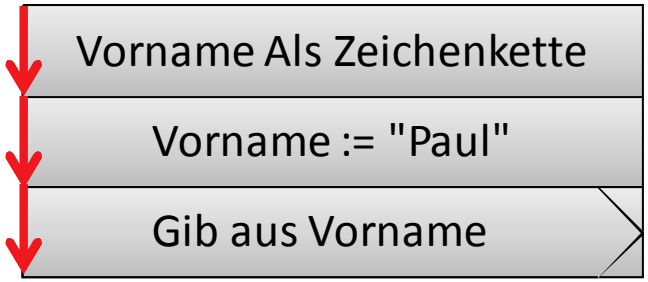


Überblick: Beispiel 03.01

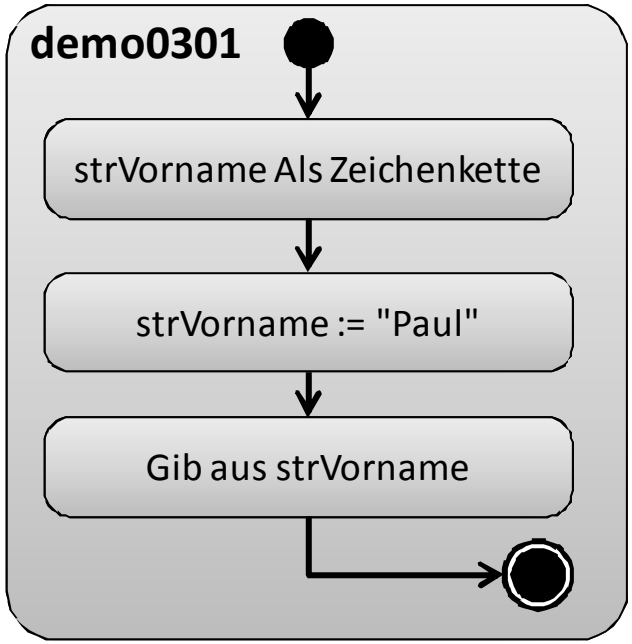
Darstellung des Programms

- Deklaration einer Variable
  - mit Bezeichner **`strVorname`**
  - Schlüsselwort **`Dim`**
  - gefolgt von **`As <Datentyp>`**
- Anweisungen
  - Initiale Zuweisung eines Wertes zur Variable **`strVorname`**
  - Ausgabe des Variablenwertes
- Kommentare
  - ohne Wirkung auf den Programmablauf
  - Unterstützen Verständnis des Algorithmus
- Hinweis: Auch die Prozedur (Unterprogramm) **`demo0301`** wird hier deklariert

Struktogramm:



Aktivitätsdiagramm:



Überblick: Beispiel 03.01

Ziel

- Variablen, Datentypen, Deklaration, Zuweisung und Zugriff demonstrieren

Aufgabe

- Deklarieren einer Variable mit dem Bezeichner **`strVorname`** vom Datentyp **`String`**
- initiale Zuweisung des Wertes "Paul" zur Variable **`strVorname`**
- Lesender Zugriff auf den Wert der Variable **`strVorname`** und Ausgabe des Wertes im Direktbereich

Ergebnis

VBA-Datentypen: Beispiel 03.02

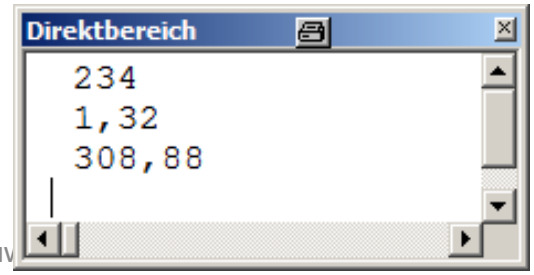
Ziel

- Nutzung von Variablen verschiedener Typen und deren Operationen

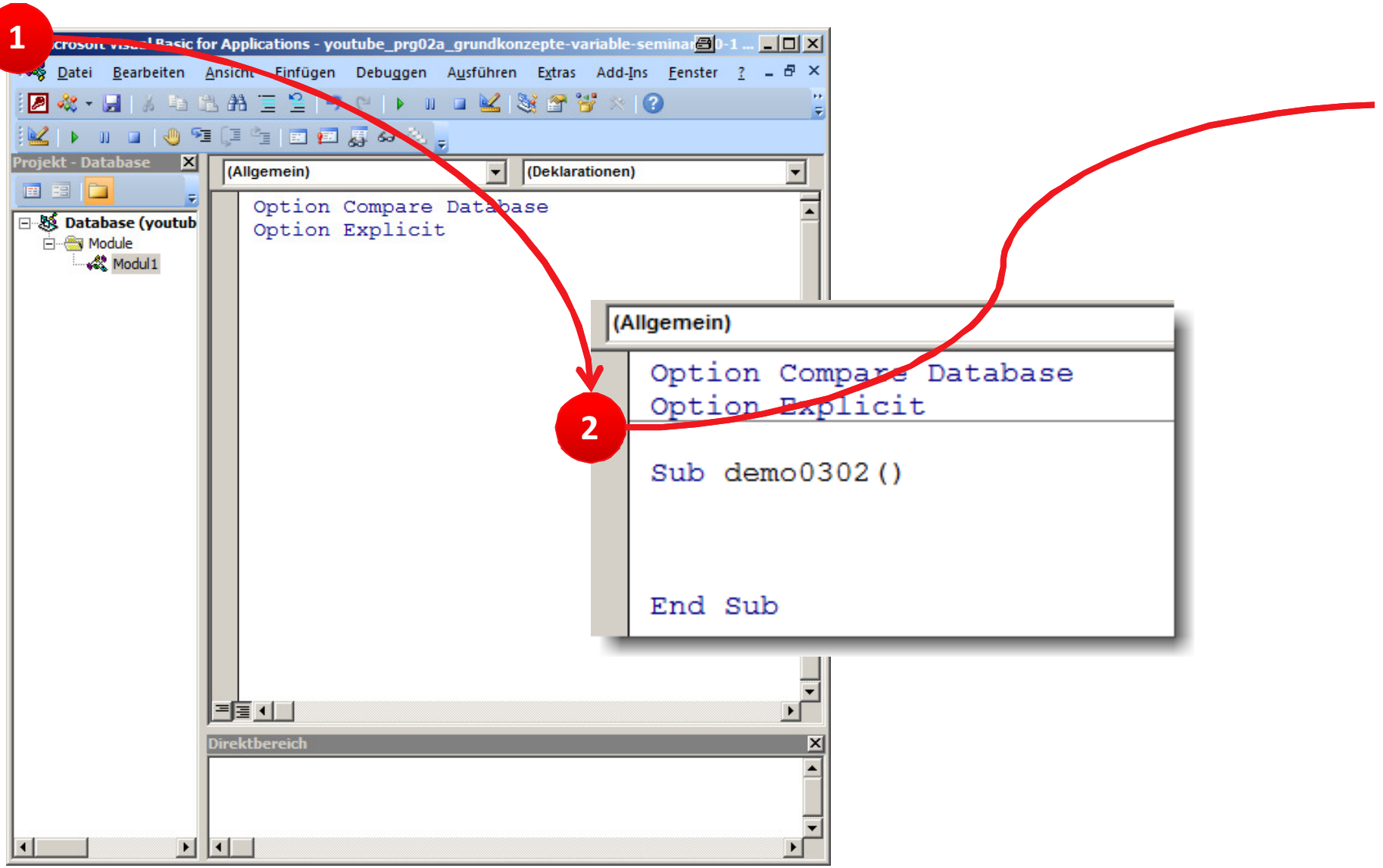
Aufgabe: Währungsumrechnung

- Deklarieren Sie zwei Variablen **`curEuro`** und **`curDollar`** beide vom Typ **`Currency`** und eine Variable **`sglKursEurUsd`** vom Typ **`Single`**
- Weisen Sie **`curEuro`** den initialen Wert **`234`** und **`sglKursEurUsd`** den initialen Wert **`1.32`** zu
- Geben Sie die Werte im Direktbereich aus
- Berechnen Sie den Wert in US-Dollar und speichern Sie ihn in der Variable **`curDollar`**
- Geben Sie den Wert im Direktbereich aus

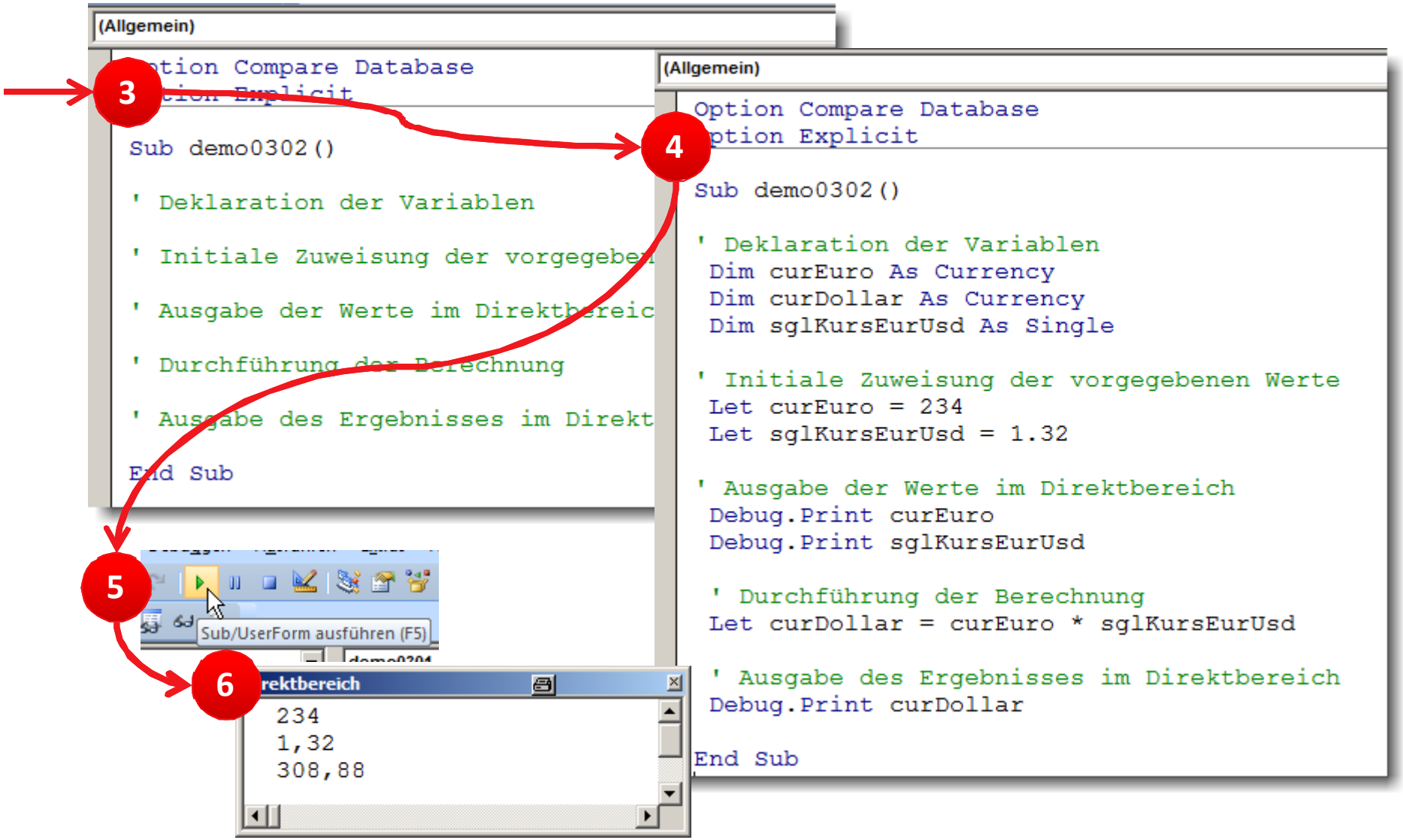
Ergebnis



VBA-Datentypen: Beispiel 03.02



VBA-Datentypen: Beispiel 03.02



VBA-Datentypen: Beispiel 03.02

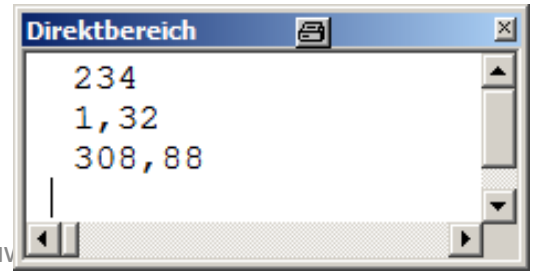
Ziel

- Nutzung von Variablen verschiedener Typen und deren Operationen

Aufgabe: Währungsumrechnung

- Deklarieren Sie zwei Variablen **`curEuro`** und **`curDollar`** beide vom Typ **`Currency`** und eine Variable **`sglKursEurUsd`** vom Typ **`Single`**
- Weisen Sie **`curEuro`** den initialen Wert **`234`** und **`sglKursEurUsd`** den initialen Wert **`1.32`** zu
- Geben Sie die Werte im Direktbereich aus
- Berechnen Sie den Wert in US-Dollar und speichern Sie ihn in der Variable **`curDollar`**
- Geben Sie den Wert im Direktbereich aus

Ergebnis



Konstante

ist der Variable ähnlich

- hat einen Bezeichner
- speichert Werte eines definierten Datentyps
- Werte oder Ausdrücke werden ihr zugewiesen
- bietet lesenden Zugriff auf gespeicherten Wert

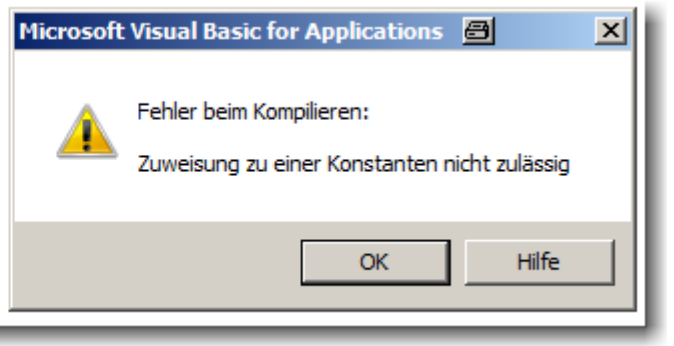
unterscheidet sich von der Variable

- Schlüsselwort zur Deklaration ist **`Const`**
- es folgen: Bezeichner, Datentyp und Zuweisung des Wertes
- Bezeichner per Konvention in Großbuchstaben
- zugewiesener Wert kann während der Programmausführung nicht verändert werden

Konstante

Veränderung des einmal zugewiesenen Wertes nicht mehr möglich

```
Sub Raumzeit()

 Const PI As Double = 3.14159265359

 ' Veränderung im Raum-Zeit-Kontinuum oder
 ' Versuch einer versehentlichen Änderung
 PI = 4.23

End Sub
```



Konstante

Einsatzmöglichkeiten

- gute Bezeichner für Konstanten unterstützen die Verwendung vordefinierter Werte
- Konstante in Verbindung mit unveränderbaren Werten, z.B.

```
Const PI As Double = 3.14159265359
```

- Konstante als symbolischer Namen
  - für selbst definierte Werte, die an nur an einer Stelle festgelegt und an mehreren Stellen verwendet werden sollen

```
Const MWST As Single = 0.19
```

  - für vordefinierte Werte, die in Verbindung mit Anweisungen eine besondere Bedeutung haben (nächste Folie)

Konstante

Beispiele vordefinierter Konstanten

- Wochentage

Konstante	Wert	Beschreibung
vbSunday	1	Sonntag
vbMonday	2	Montag
…	…	…
vbSaturday	7	Samstag

- Dateieigenschaften

Konstante	Wert	Beschreibung
vbReadOnly	1	Schreibgeschützte Datei/Verzeichnis
vbHidden	2	Versteckte Datei/Verzeichnis
vbSystem	4	Systemdatei
…	…	…

Konstante

- wird deklariert mit Schlüsselwort Const
- hat einen **Bezeichner**
- und ist von einem definierten **Datentyp**
- **Werte** oder **Ausdrücke** können einmalig **zugewiesen** werden
- erstmalige Zuweisung (**Initialisierung**) erfolgt gemeinsam mit **Deklaration**
- bietet lesenden **Zugriff** auf gespeicherten Wert

```
' Generelle Syntax
' Deklaration und Initialisierung
' Const <Bezeichner> As <Datentyp> = <WertOderAusdruck>

' Beispiel
Const PI As Double = 3.14159265359
```

Inhalt

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Typumwandlung: Beispiel 03.05

Ziel

- Erkennen, dass VBA richtig rechnet

Aufgabe

- Umrechnung eines gegebenen Jahresgehalts von 64800 EUR in das Monatsgehalt (12 Gehälter)
- Gehaltserhöhung auf 65000 EUR und erneute Berechnung des Monatsgehalts
- Ausgabe des Jahres und Monatsgehalts

Ergebnis

- ?

Typumwandlung

Definition: Eine Typumwandlung (syn. Casting) ist die Umwandlung

- eines Wertes w1 eines Typs T1
- in einen Wert w2 des Typs T2.

Beispiele für Typumwandlungen:

– Integer	→	Double:	17	→	17.0
– Double	→	Integer:	3.75	→	3
– Byte	→	Integer:	17	→	17

Typumwandlung

Bei Typumwandlung besteht häufig die Notwendigkeit für Werte (Literale) anzugeben, von welchem Typ sie sind, z.B.

- Ist 42 Byte, Integer oder Long?
- Soll 3.14 als Single oder Double verwendet werden?

Verwendung von Typkennzeichen für Umwandlungen

Datentyp	Kennzeichen	Beispiele
Byte	- Kein -	
Integer (Standard)	% und ohne	42%, 42
Long	&	42&
LongLong	^	42^
Single	!	3.1!
Double (Standard)	# und ohne	3.1#, 3.1
Currency	@	3.1@

```
Direktbereich
? TypeName(42)
Integer
? TypeName(42%)
Integer
? TypeName(42&)
Long
? TypeName(3.1)
Double
? TypeName(3.1#)
Double
? TypeName(3.1!)
Single
? TypeName(3.1@)
Currency
```

Typumwandlungen

Unterscheidungsmöglichkeit 1

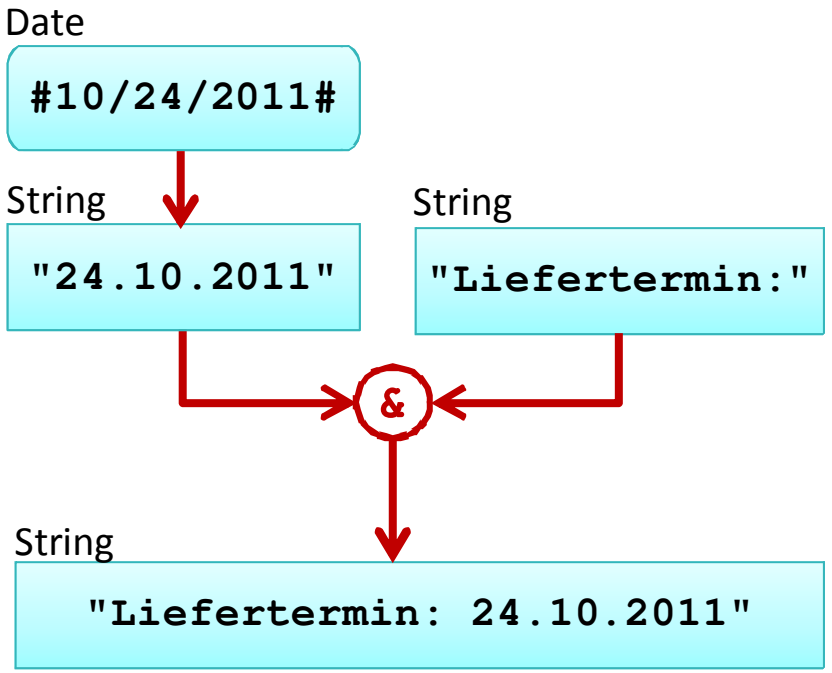
- zwischen nah verwandten Typen (automatisch von VBA ausgeführt, implizit)
- zwischen beliebigen Typen (explizite Nutzung einer Funktion zur Typumwandlung erforderlich)

Hier: Nah verwandte Typen, z.B. die numerischen Typen

Typumwandlungen

Implizite Umwandlung

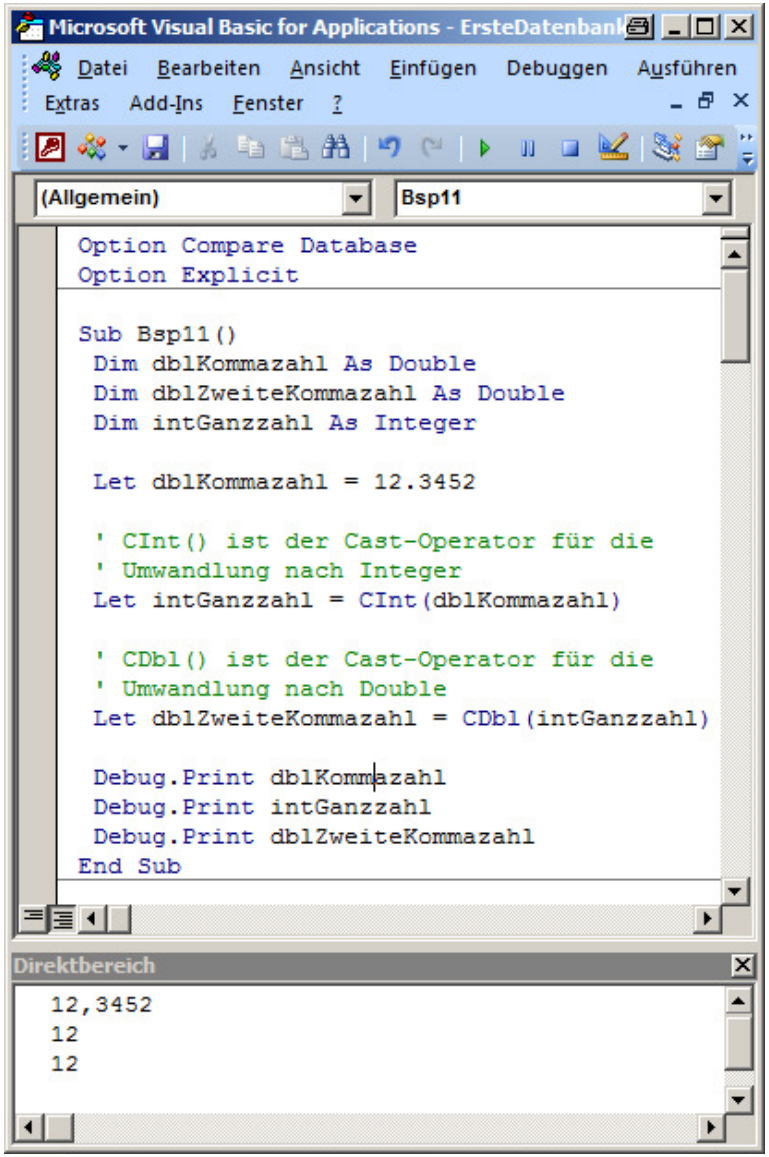
- häufige Problemstellung bei Auswertung von Ausdrücken und bei Zuweisungen
- Werte von Variablen mit unterschiedlichen Datentypen in Ausdruck
- Implizite Typumwandlung geht ohne Informationsverlust einher, z.B.
  - Alle Datentypen nach String
  - Byte nach Integer, Long, Decimal
  - Currency nach Decimal



Typumwandlungen

Explizite Umwandlung

- können mit (gewolltem) Informationsverlust einhergehen, z.B.
  - Double nach Integer unter Verlust der Kommastellen
- Verwendung von Cast-Operatoren, um Umwandlung zu steuern
  - **`CBool, CInt, CLng, CStr, CDate, CDbl`**
  - bei **`CInt`** und **`CLng`** in Verbindung mit Rundung
  - Details auf nächster Folie



Typumwandlungsfunktionen[1]

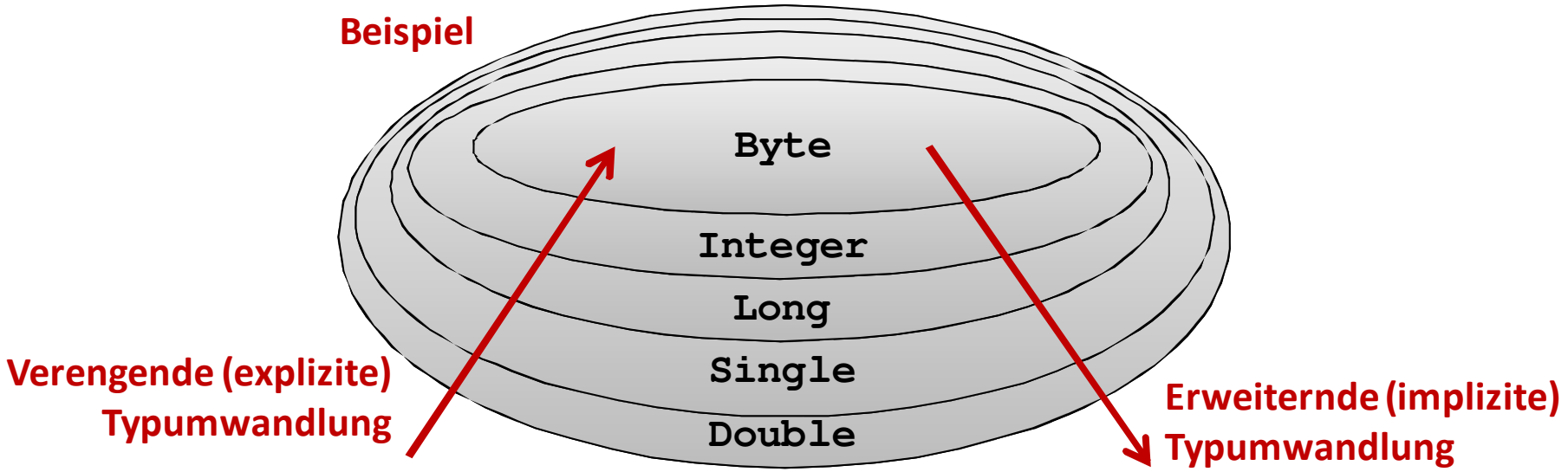
Funktion	Rückgabetyp	Bereich des Arguments *Ausdruck*
CBool	Boolean	Eine gültige Zeichenfolge oder ein gültiger numerischer Ausdruck.
CByte	Byte	0 bis 255.
CCur	Currency	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807.
CDate	Date	Ein beliebiger gültiger Datumsausdruck.
CDbl	Double	-1,79769313486231E308 bis -4,94065645841247E-324 für negative Werte; 4,94065645841247E-324 bis 1,79769313486232E308 für positive Werte.
CInt	Integer	-32.768 bis 32.767; Nachkommastellen werden gerundet.
CLng	Long	-2.147.483.648 bis 2.147.483.647; Nachkommastellen werden gerundet.
CLngLng	LongLong	-9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807; Dezimalen werden gerundet (nur auf 64-Bit-Plattformen zulässig)
CSng	Single	-3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte.
CStr	String	Rückgabe für CStr hängt vom Argument *Ausdruck* ab.

1) Quelle: [3]

Typumwandlungen

Unterscheidungsmöglichkeit 2

- Verengende Typumwandlung
- Erweiternde Typumwandlung



Typumwandlungen

Unterscheidungsmöglichkeit 2

- Verengende Typumwandlung
  - Konvertierung in Typ mit kleinerem Wertebereich
  - Verlust von Information (z.B. Genauigkeit)
  - müssen explizit befohlen werden, z.B.:
  - Beispiele

```
Dim i As Integer, s As Single
Let s = 3.14
Let i = CInt(s)
```

- Erweiternde Typumwandlung
  - Konvertierung in Typ mit größerem Wertebereich
  - Erweiternde Typumwandlungen implizit, d.h. automatisch vorgenommen
  - Beispiele

```
Dim i As Integer, b As Byte
Let b = 3
Let i = b
```

Inhalt

Inhalt

Rückblick

Kommentare in VBA

Grundlagen und Überblick

Datentypen

Wert, Ausdruck und Zuweisung

Variable und Konstante

Typumwandlungen

Abschluss und Ausblick

Inhalt

Abschluss

Variable

- speichert Wert eines definierten Datentyps
- Wert oder Ausdruck wird i zugewiesen
- bietet
  - lesenden Zugriff und
  - schreibenden/ ändernden Zugriff auf gespeicherten Wert
- hat einen Bezeichner (Namen)

Abschluss

Variable

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

IKEA FAKTUM
Küchenunterschrank

IKEA SKÄR
Schuhschrank

IKEA
ASPVIK
Akten-
schrank

Hier und im Folgenden Quelle: [6]

Abschluss

Schubladen in Möbeln

– bieten Platz für Gegenstände eines bestimmten Typs

IKEA FAKTUM
Küchenunterschrank

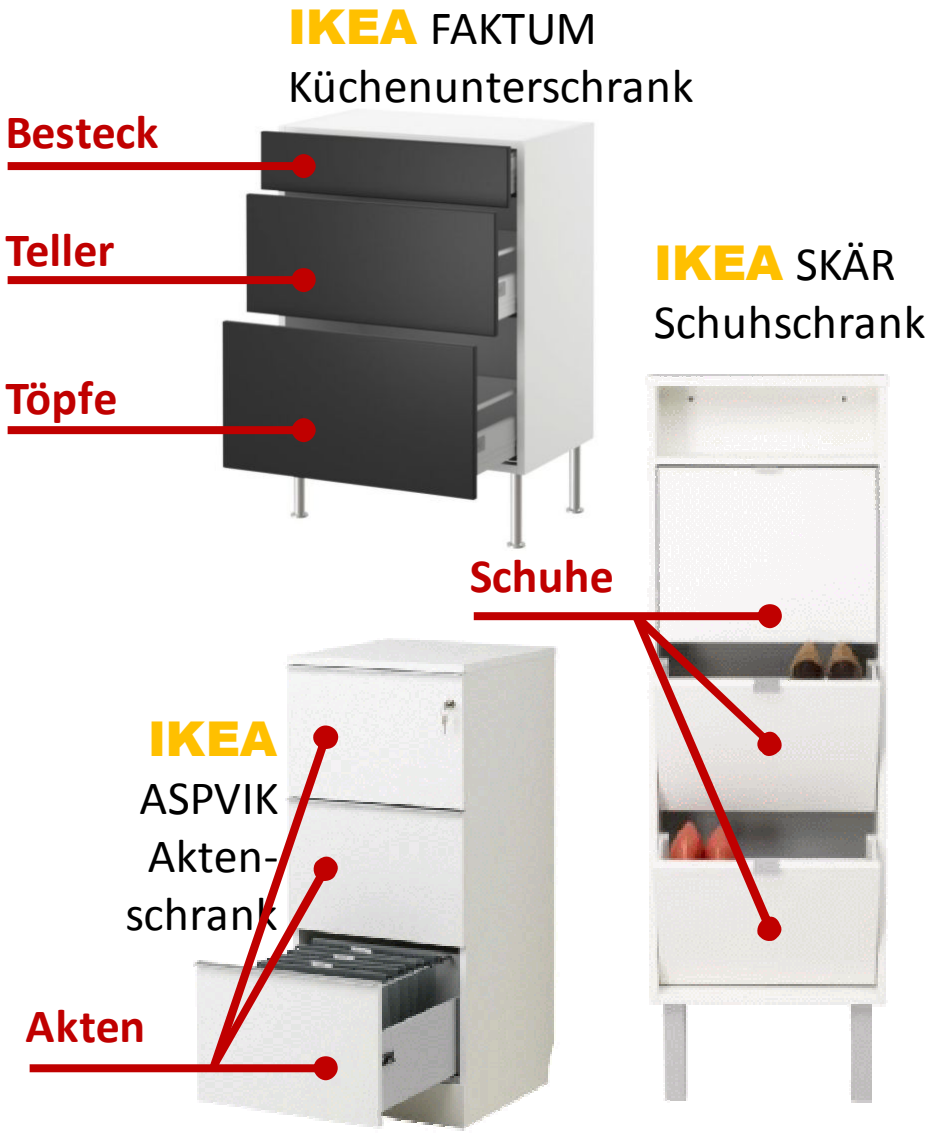
IKEA SKÄR
Schuhschrank

IKEA
ASPVIK
Akten-
schrank

Abschluss

Schubladen in Möbeln

– bieten Platz für Gegenstände eines bestimmten Typs



Abschluss

Schubladen in Möbeln

- bieten Platz für Gegenstände eines bestimmten Typs
- Gegenstände werden in ihnen abgelegt

IKEA FAKTUM
Küchenunterschrank

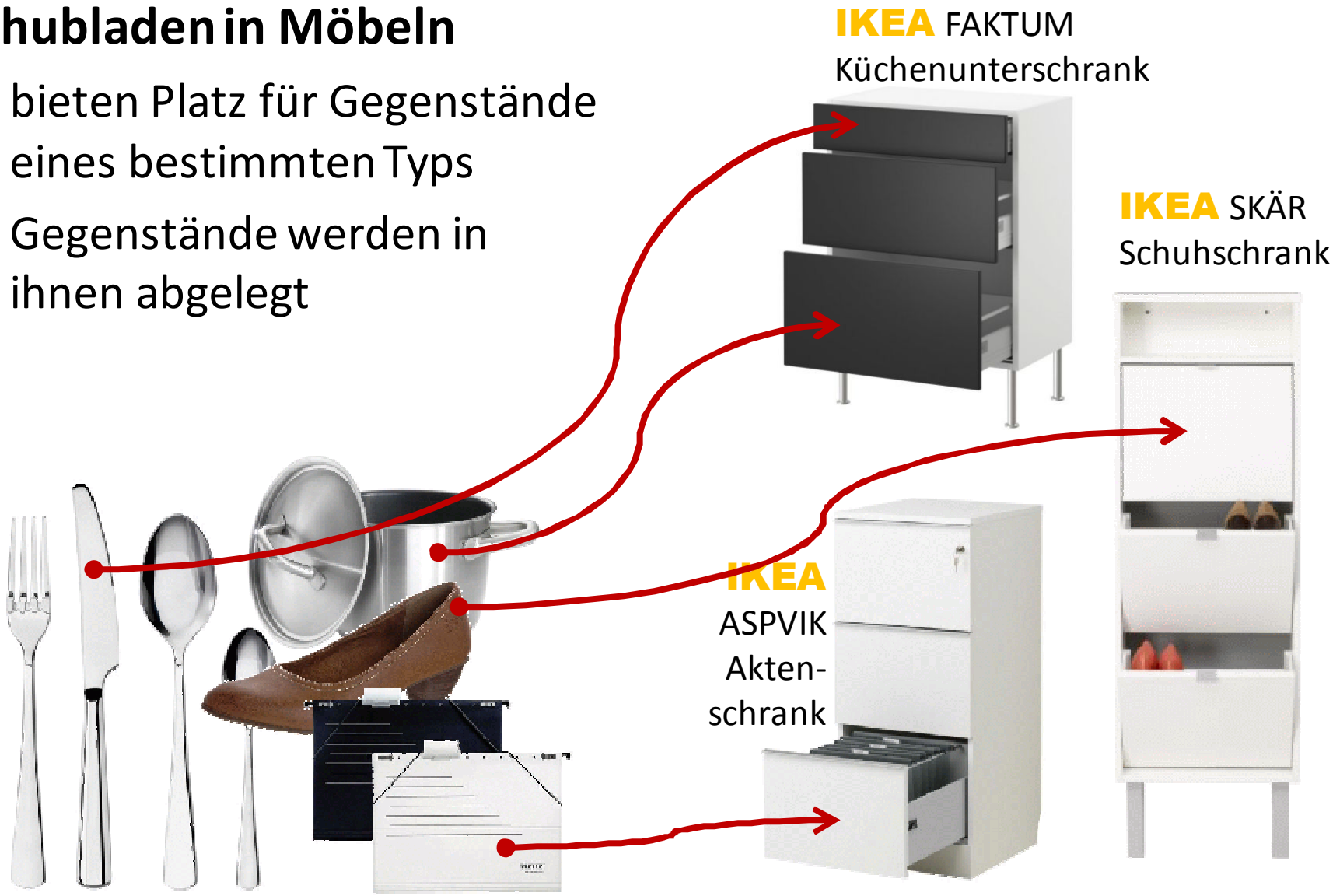
IKEA SKÄR
Schuhschrank

IKEA
ASPVIK
Akten-
schrank

Abschluss

Schubladen in Möbeln

- bieten Platz für Gegenstände eines bestimmten Typs
- Gegenstände werden in ihnen abgelegt



Abschluss

Schubladen in Möbeln

- bieten Platz für Gegenstände eines bestimmten Typs
- Gegenstände werden in ihnen abgelegt
- es kann
  - nachgesehen werden was vorhanden ist
  - vorhandener Gegenstand durch anderen ersetzt werden

Abschluss

Schubladen in Möbeln

- bieten Platz für Gegenstände eines bestimmten Typs
- Gegenstände werden in ihnen abgelegt
- es kann
  - nachgesehen werden was vorhanden ist
  - vorhandener Gegenstand durch anderen ersetzt werden
- haben nur selten eine Bezeichnung

IKEA FAKTUM
Küchenunterschrank

IKEA SKÄR
Schuhschrank

IKEA
ASPVIK
Akten-
schrank

Abschluss

Schubladen in Möbeln

- bieten Platz für Gegenstände eines bestimmten Typs
- Gegenstände werden in ihnen abgelegt
- es kann
  - nachgesehen werden was vorhanden ist
  - vorhandener Gegenstand durch anderen ersetzt werden
- haben nur selten eine Bezeichnung

Abschluss

Variable	Schubladen in Möbeln
– speichert Werte eines definierten Datentyps	– bieten Platz für Gegenstände eines bestimmten Typs
– Werte oder Ausdrücke werden ihr zugewiesen	– Gegenstände werden in ihnen abgelegt
– bietet • lesenden Zugriff und • schreibenden/ändernden Zugriff auf gespeicherten Wert	– es kann • nachgesehen werden was vorhanden ist • vorhandener Gegenstand durch anderen ersetzt werden
– hat einen Bezeichner (Namen)	– haben nur selten eine Bezeichnung

Abschluss

Konstante

- ist der Variable ähnlich, unterscheidet sich aber
  - Schlüsselwort zur Deklaration ist **`Const`**
  - zugewiesener Wert kann während der Programmausführung nicht verändert werden
- hat kaum etwas mit **IKEA**-Möbeln zu tun.

Variable

- wird deklariert mit Schlüsselwort Dim
- hat einen **Bezeichner**
- und ist von einem definierten **Datentyp**
- **Werte** oder **Ausdrücke** werden ihr **zugewiesen**
- erstmalige Zuweisung heißt **Initialisierung**
- bietet **Zugriff** auf gespeicherten Wert
  - lesend
  - schreibend/ändernd

Syntax

```
Dim <VarBezeich> As <Datentyp>

Let <VarBezeich> = <Wert/Ausdr>
```

Beispiele

```
' Weniger gute Bezeichner
Dim i As Integer
Dim s As String
' Aussagekräftige Bezeichner
Dim bytAlter As Byte
Dim sglBetrag As Single
' Initialisierung mit Wert
Let bytAlter = 20
' Initialisierung mit Ausdruck
Let sglBetrag = bytAlter * 3
' Lesender Zugriff
Debug.Print bytAlter
' Ändernder Zugriff
Let bytAlter = bytAlter + 1
Let sglBeitrag = 42
```

Konstante

- wird deklariert mit Schlüsselwort Const
- hat einen **Bezeichner**
- und ist von einem definierten **Datentyp**
- **Werte** oder **Ausdrücke** können einmalig **zugewiesen** werden
- erstmalige Zuweisung (**Initialisierung**) erfolgt gemeinsam mit **Deklaration**
- bietet lesenden **Zugriff** auf gespeicherten Wert

```
' Generelle Syntax
' Deklaration und Initialisierung
' Const <Bezeichner> As <Datentyp> = <WertOderAusdruck>

' Beispiel
Const PI As Double = 3.14159265359
Const PI As Double = 3.14159265359
```

Ausblick

06 – Zusammenfassung

05 – Debugger und Testen

04 – Fortgeschrittene Konzepte

04.A	04.B	04.C
Oberflächen (Teil 1) Elemente und Eigenschaften	Oberflächen (Teil 2) Ereignisverarbeitung	Zugriff auf Dateisystem und Anwendungen

03 – Grundkonzepte

03.A	03.B	03.C	03.D	03.E
Wert Ausdruck Variable Konstante Datentyp	Bedingte Ausführung/ Verzwei-gungen	Schleifen	Felder Mengen	Prozedur Funktion Modul

02 – Grundlagen der Programmierung

01 – Grundbegriffe der Wirtschaftsinformatik

Quellen

[1] Heike Ripphausen-Lipa: Folien zur Vorlesung "Programmierung 1", Fachbereich 6, Beuth Hochschule für Technik Berlin, 2010.

[2] Otto Rauh: Skript: Einführung in VBA mit Excel. Hochschule Heilbronn, Fassung vom August 2012, Online: http://www.orauh.de/app/download/5682302164/ExcVBA01.pdf?t=1346409095

[3] VBA-Online Hilfe für Entwickler: Typ-Umwandlungsfunktionen. Microsoft, 2010

[4] Urheber: Cyberdex, Lizenz: public domain (gemeinfrei), Quelle: http://en.wikipedia.org/wiki/File:Memory_module_DDRAM_20-03-2006.jpg

[5] Stackoverflow.com: Why can't variable names start with numbers? Online: http://stackoverflow.com/questions/342152/why-cant-variable-names-start-with-numbers

[6] IKEA: http://www.ikea.com

[7] Microsoft: Operator Precedence in Visual Basic. Online: https://docs.microsoft.com/de-de/dotnet/visual-basic/language-reference/operators/operator-precedence

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wirtschaftsinformatik 1
LE 03 – Variable, Konstante und Datentypen

Prof. Dr. Thomas Off

http://www.ThomasOff.de/lehre/beuth/wi1