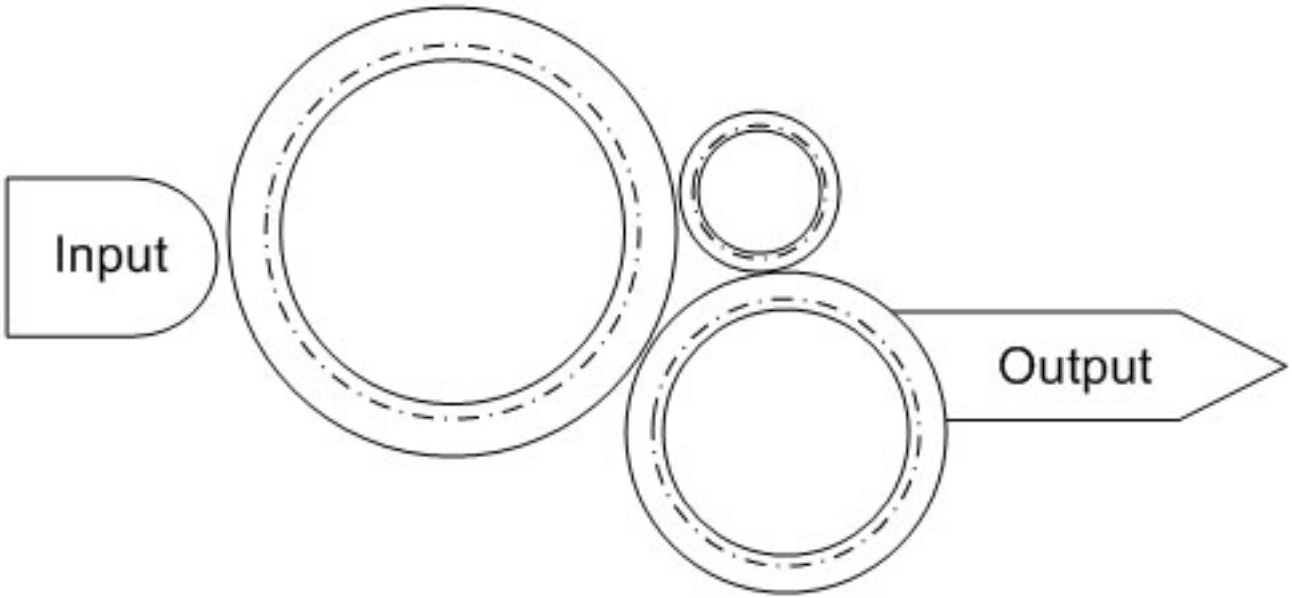
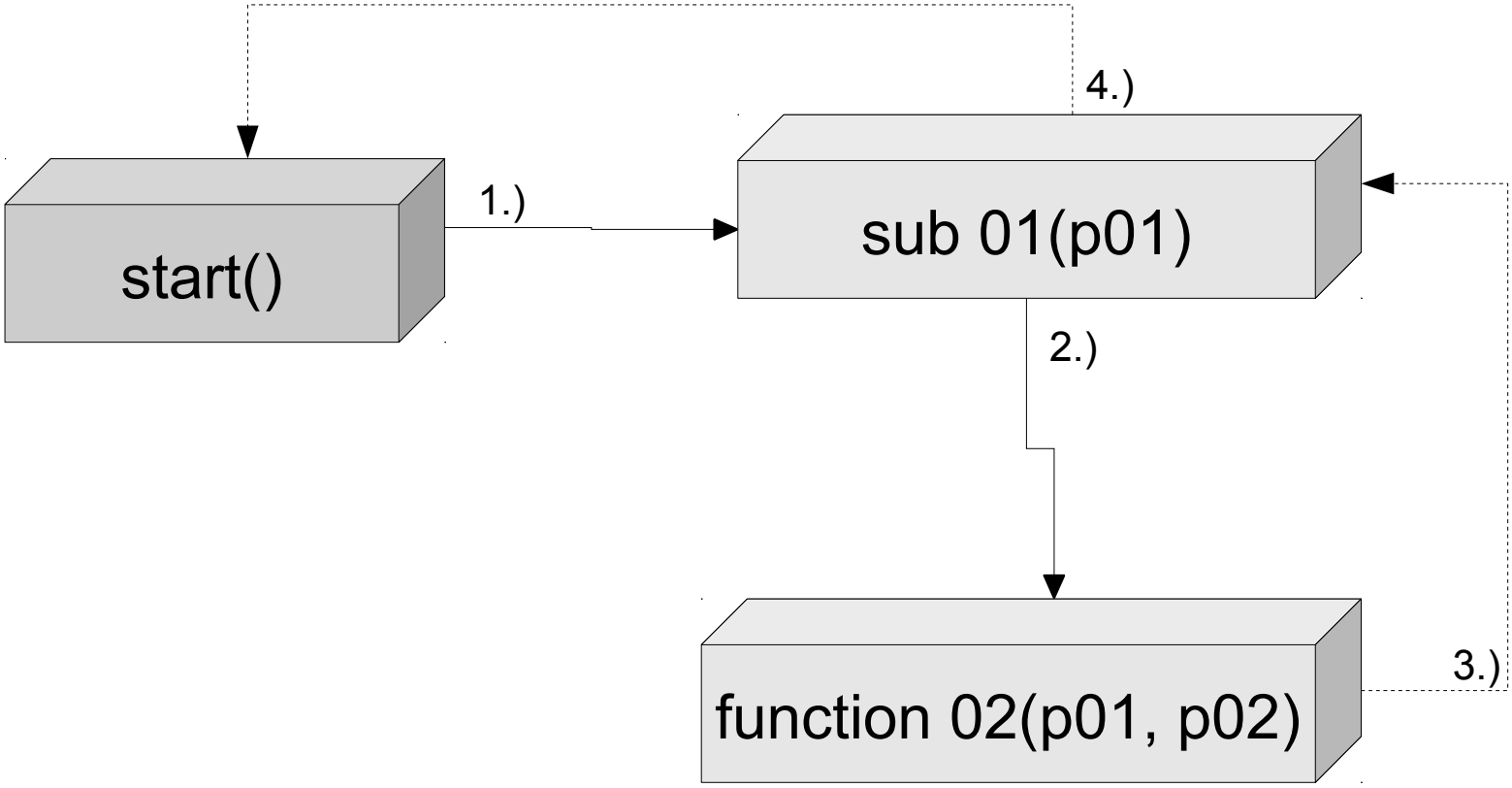
V(isual) B(asic for) A(pplication)
Prozeduren und Funktionen



Subroutine (Unterprogramm)

- Zerlegung eines Projekts in verschieden Aufgaben.
- Implementierung einer Aufgabe mit Hilfe von Anweisungen.
- Beschreibung einer bestimmten Funktionalität mit Hilfe einer Programmiersprache.
- Strukturierung von großen Programmen.

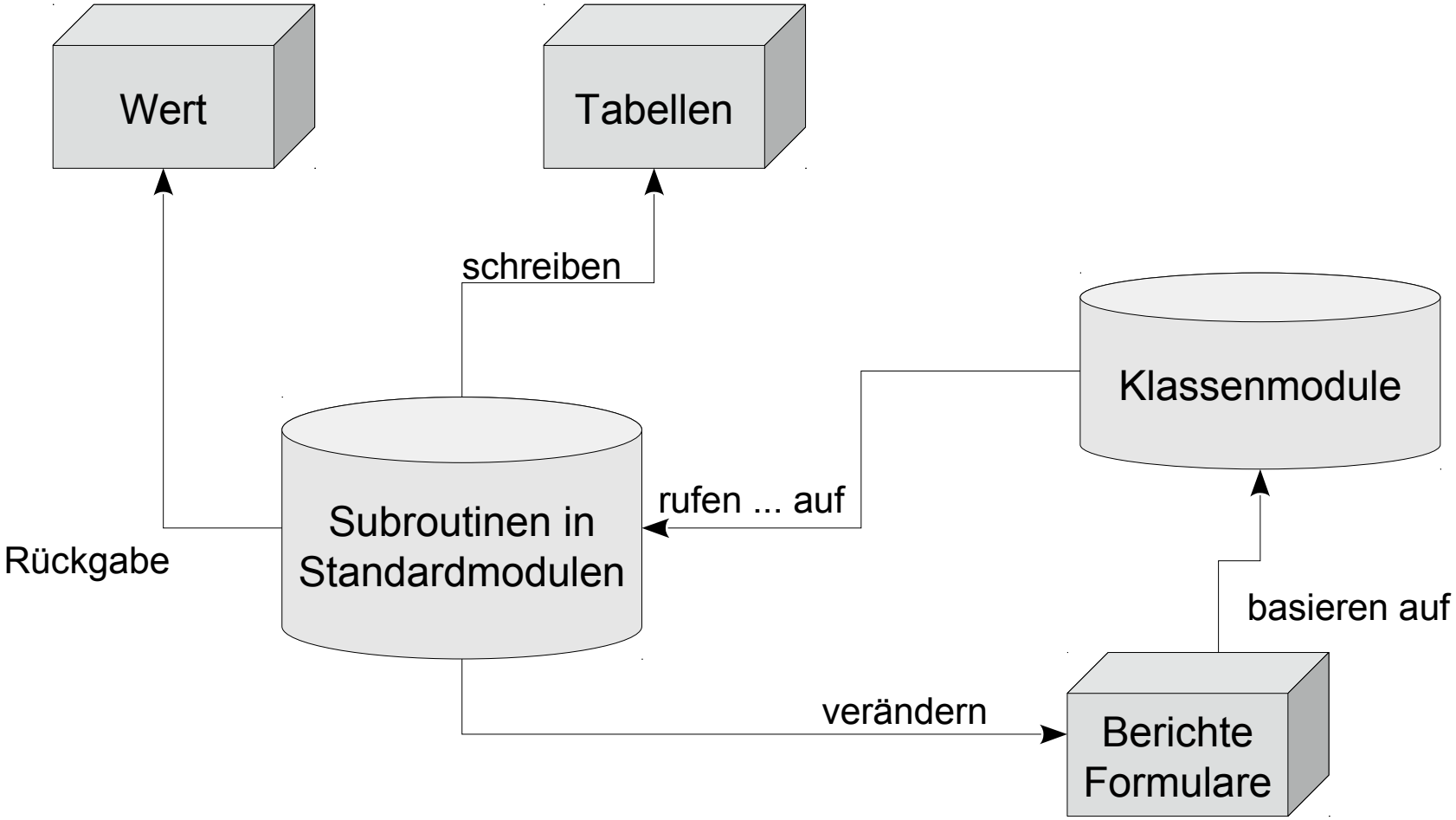
Aufruf von Subroutinen



Zusammenfassung mit Hilfe von Modulen

- Module sammeln Subroutinen zu einem Projekt.
- Module fassen Beschreibungen eines Objektes zusammen.
- Automatische Anlage für ein Microsoft Office Objekt.
- Manuelle Anlage vom Entwickler, um Subroutinen zusammen zu fassen und wieder zu verwenden.

… in Microsoft Office



Prozeduren

- Blackbox von Anweisungen, die eine Aufgabe in einem Projekt widerspiegeln.
- Prozeduren können Parameter zur Initialisierung von Variablen übergeben werden.
- Prozeduren geben keinen Wert an den Aufrufer zurück.

Aufbau einer Prozedur

Sub Starten()	Prozedur-Kopf
Dim wertL As Integer Dim wertR As Integer wertL = 5 wertR = 3	Anweisungen in der Prozedur
End Sub	Prozedur-Fuß

Prozedur-Kopf

Sub	Starten	(						)
Sub	Addieren	(	opL As Integer	,	opR As Integer	,	...	)
Sub	[Name]	(	[Parameterliste]					)

- Jede Prozedur beginnt mit dem Schlüsselwort Sub.
- Der Name der Prozedur ist eindeutig in dem Modul.
- Dem Prozedur-Namen folgt in runde Klammern die Parameterliste. Die Parameterliste definiert die Startwerte der Prozedur.

Bezeichner

- Namen für Funktionen, Konstanten, Variablen etc.
- Die Namen sind in ihrem Gültigkeitsbereich eindeutig.
- Schlüsselwörter der Programmiersprache sind als benutzerdefinierte Namen nicht erlaubt.
- Keine Unterscheidung zwischen Groß- und Kleinschreibung. Der Name „start“ und „Start“ sind Platzhalter für eine Prozedur.

Erlaubte Zeichen

- Buchstaben A...Z und a...z.
- Zahlen 0...9.
- Der Unterstrich.

Konventionen

- Benutzerdefinierte Namen beginnen mit einem Buchstaben.
- Leerzeichen, Punkt, Ausrufezeichen und die Zeichen @, &, $ und das Hash-Zeichen dürfen nicht in einem Namen genutzt werden.
- Der Name ist maximal 255 Zeichen lang.

Konventionen für Prozeduren

- Der Name einer Prozedur beginnt mit einem Großbuchstaben.
- Für Prozeduren werden häufig Verben genutzt.
- Der Name spiegelt die implementierte Aufgabe wieder.

Anweisungen in einer Prozedur

- Deklaration von Variablen und Konstanten.
- Anweisungen zur mathematischen Berechnung von Variablen etc.
- Aufruf von Prozeduren.

Parameterliste

Sub	Starten	(						)
Sub	Addieren	(	opL As Integer	,	opR As Integer	,	...	)
Sub	[Name]	(	[Parameterliste]					)

- Dem Prozedur-Namen folgt direkt die Parameterliste.
- Die Parameterliste beginnt und endet mit den runden Klammern.
- Die Parameterliste kann leer sein. Der Prozedur werden keine Start-Werte übergeben.

Parameter in der Liste

Sub	Starten	(						)
Sub	Addieren	(	opL As Integer	,	opR As Integer	,	...	)
Sub	[Name]	(	[Parameterliste]					)

- Die Parameter werden durch Kommata in der Parameterliste getrennt.
- Die Parameterliste kann beliebig viele Parameter enthalten.

Deklaration von Parametern

Sub	Addieren	(	opL As Integer	,	opR As Integer	,	...	)
Sub	[Name]	(	[var] As [Typ]	,	[var] As [Typ]	,	...	)

- Jeder Parameter hat einen eindeutigen benutzerdefinierten Namen.
- Mit Hilfe des Schlüsselwortes As wird dem Parameter ein Datentyp zugewiesen.

Aufruf einer Prozedur

```
Call Addieren(wertL, wertR)
```

```
Sub | Addieren | ( | opL As Integer | , | opR As Integer | , | ... | )
```

- Der Aufruf einer Prozedur beginnt mit dem Schlüsselwort Call (Aufruf).
- Dem Schlüsselwort folgt der Name der Prozedur. Der Name ist ein eindeutiger Platzhalter für die Prozedur.
- Dem Namen folgt die Parameterliste. Die Parameterliste beim Aufruf entspricht der Liste im Prozedur-Kopf.

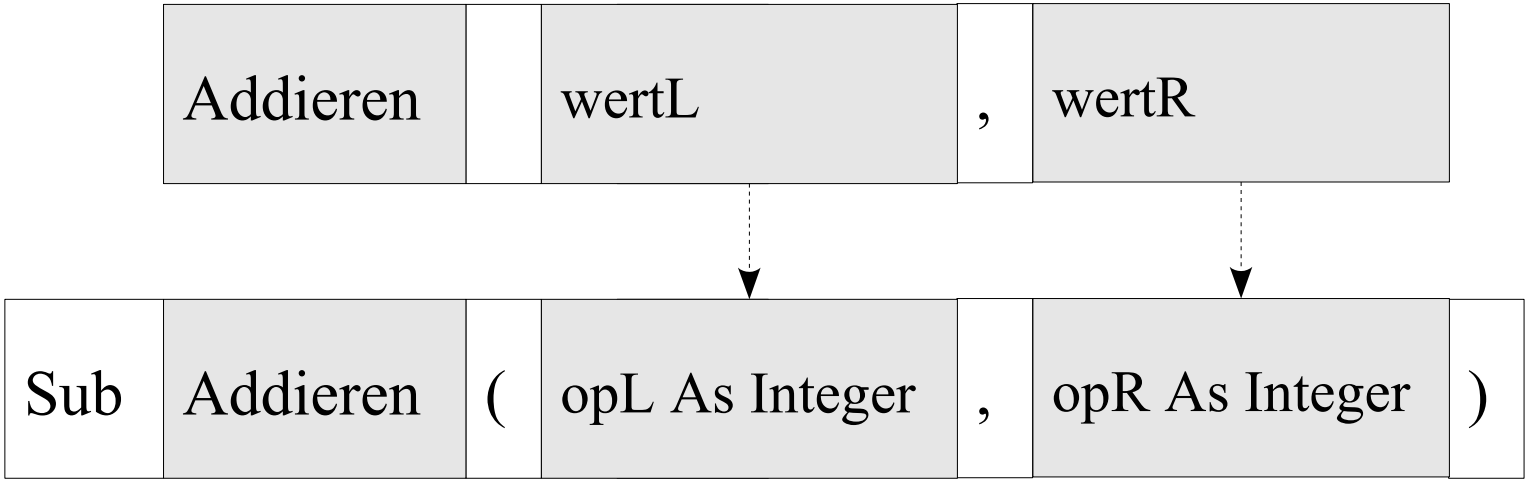
Andere Möglichkeit

Addieren wertL, wertR

Sub	Addieren	(	opL As Integer	,	opR As Integer	,	...	)

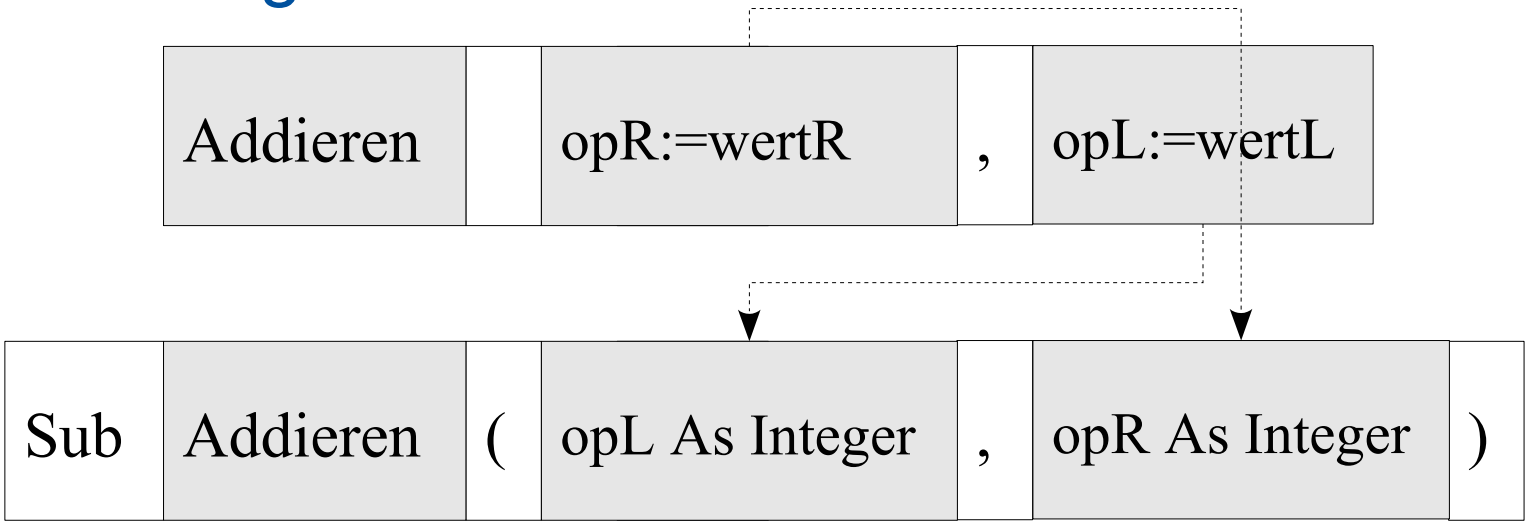
- Die Prozedur wird mit dem Namen aufgerufen.
- Dem Namen folgen die zu übergebenen Parameter. Die Parameter werden durch ein Kommata getrennt.
- Die Parameterliste darf nicht durch die runden Klammern begrenzt werden.

Übergabe von Parametern



- Für jeden Parameter in der Liste im Prozedur-Kopf wird ein Wert übergeben.
- Die Parameter des Aufrufs werden von links nach rechts dem Parametern im Prozedur-Kopf zugeordnet.
- Der Parameter im Prozedur-Kopf und der Wert im Aufruf sollten den gleichen Datentypen haben.

Nutzung von benannten Parametern



- Dem Parameter wird mit Hilfe des Operators := ein Wert zugewiesen.
- Der Name des Parameters im Kopf der Subroutine wird beim Aufruf ein Wert entsprechend des Datentyps übergeben.

Benannte Parameter im Aufruf

wertR	:=	wertR
[parameter]	:=	[ausdruck]

- Der Operator := wird aus dem Doppelpunkt und dem Gleichheitszeichen zusammengesetzt.
- Links vom Operator steht ein Name eines Parameters, der in der aufzurufende Prozedur in der Parameterliste deklariert ist.
- Rechts vom Operator kann ein beliebiger Ausdruck oder eine Variable stehen, der den zu übergebenen Wert berechnet oder gespeichert hat.

Zugriff auf die Prozedur

Private	Sub	getFlaeche	(	radius As Double	)
Public	Sub	getFlaeche	(	radius As Double	)
[Zugriff]	Sub	[Name]	(	[Parameterliste]	)

Öffentlicher Zugriff

Public	Sub	[Name]	(	[Parameterliste]	)

- Die Subroutine ist im gesamten Projekt sichtbar.
- Ein Aufruf kann aus allen Prozeduren im Projekt, egal in welchem Modul erfolgen.

Aufruf der Subroutine

SubKreis_Berechnung	.	getFlaeche	(	radius	)
[Modul]	.	[Subroutine]	(	[parameterliste]	)

- Modulnamen werden mit Prozeduren durch den Punktoperator verbunden.
- Die Angabe rechts vom Punkt beschreibt das Objekt / Modul, in dem die Subroutine links vom Punkt definiert ist.

Öffentlicher Zugriff

Private	Sub	[Name]	(	[Parameterliste]	)

- Die Subroutine ist nur in dem Modul sichtbar, in dem sie definiert ist.
- Ein Aufruf kann aus allen Prozeduren im Modul, in dem die Subroutine definiert ist, erfolgen.

Aufruf der Subroutine

SubKreis_Berechnung	.	getUmfang	(	radius	)
		getUmfang	(	radius	)
[Modul]	.	[Subroutine]	(	[parameterliste]	)

- Bei einem Aufruf einer privaten Prozedur kann der Modulname weggelassen werden.

Funktionen

- Blackbox von Anweisungen, die eine Aufgabe in einem Projekt widerspiegeln.
- Funktionen können Parameter zur Initialisierung von Variablen übergeben werden.
- Funktionen geben einen Wert von einem beliebigen Datentyp an den Aufrufer zurück.

Vordefinierte Funktionen

- Mathematische Berechnungen.
- Datums- und Zeitfunktionen.
- Bearbeitung von Strings.
- Konvertierung von Datentypen.
- Auflistung nach Kategorien: https://msdn.microsoft.com/en-us/library/office/ff836861.aspx
- Alphabetische Auflistung: https://msdn.microsoft.com/en-us/library/office/ff823033.aspx

Aufbau einer Funktion

Function Subtrahieren (operandL As Integer, operandR As Integer) As Integer	Funktionskopf
Dim ergebnis As Integer ergebnis = operandL - operandR Subtrahieren = ergebnis	Anweisungen in der Funktion
End Function	Funktionsfuß

Funktionskopf

Function	Einlesen	(		)	As	String
Function	Subtrahieren	(	[Parameterliste]	)	As	Integer
Function	[Name]	(	[Parameterliste]	)	As	[Datentyp]

- Jede Funktion beginnt mit dem Schlüsselwort Function.
- Der Name der Funktion ist eindeutig in dem Modul.
- Dem Namen der Funktion folgt die Parameterliste.
- Mit Hilfe von As wird der Funktion ein Datentyp zugewiesen.

Konventionen für Funktionen

- Funktionsnamen beginnen häufig mit Verben, die die Aufgabe widerspiegeln. Zum Beispiel „BerechneKreisflaeche“ für eine Funktion, die die Fläche eines Kreises berechnet.
- Funktionen, die mit „is“ beginnen, liefern einen boolschen Wert zurück. Zum Beispiel „isEqual“ überprüft, ob die übergebenen Werte gleich sind.
- Funktionen, die mit „set“ beginnen, setzen eine Variable.
- Funktionen, die mit „get“ beginnen, liefern einen beliebigen Wert zurück.

Parameterliste

Function	Subtrahieren	(	opL As Integer	,	opR As Integer	)
Function	[Name]	(	[Parameterliste]			)

- Dem Funktionsnamen folgt direkt die Parameterliste.
- Die Parameterliste beginnt und endet mit den runden Klammern.
- Die Parameterliste kann leer sein. Der Funktion werden keine Start-Werte übergeben.

Parameter in der Liste

Function	Subtrahieren	(	opL As Integer	,	opR As Integer	)
Function	[Name]	(	[Parameterliste]			)

- Die Parameter werden durch Kommata in der Parameterliste getrennt.
- Die Parameterliste kann beliebig viele Parameter enthalten.

Deklaration von Parametern

Function	Subtrahieren	(	opL As Integer	,	opR As Integer	)
Function	[Name]	(	[Parameterliste]			)

- Jeder Parameter hat einen eindeutigen benutzerdefinierten Namen.
- Mit Hilfe des Schlüsselwortes As wird dem Parameter ein Datentyp zugewiesen.

Datentyp einer Funktion

Function	Subtrahieren	(	[Parameterliste]	)	As	Integer
Function	[Name]	(	[Parameterliste]	)	As	[Datentyp]

- Jeder Funktion wird im Funktionskopf ein Datentyp zugewiesen.
- Der Parameterliste folgt das Schlüsselwortes As. Mit Hilfe des Schlüsselwortes wird der Funktion ein beliebiger Datentyp zugewiesen.
- Die Funktion ist vom Datentyp ...

Anweisungen in einer Funktionen

- Deklaration von Variablen und Konstanten.
- Anweisungen zur mathematischen Berechnung von Variablen etc.
- Aufruf von Prozeduren und Funktionen.
- Übergabe eines Wertes an die Funktion.

Rückgabewert der Funktion

Subtrahieren	=	ergebnis
[Name]	=	[ausdruck]

- Dem Namen einer Funktion wird mit Hilfe des Zuweisungsoperator ein Wert zugewiesen. Der zugewiesene Wert sollte zum Datentyp der Funktion passen.
- Der Funktionsname ist ein Platzhalter für den Rückgabewert der Funktion.
- Der Rückgabewert kann durch einen Ausdruck berechnet oder direkt angegeben werden.

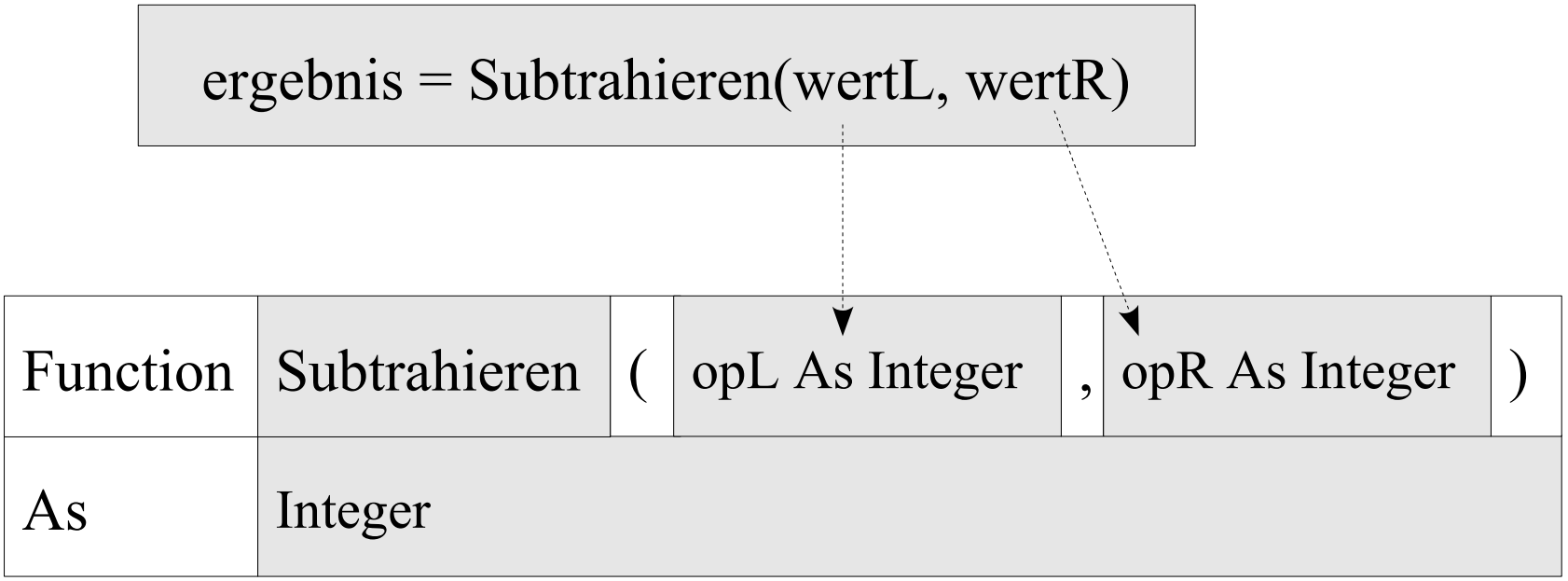
Aufruf einer Funktion

```
ergebnis = Subtrahieren(wertL, wertR)
```

Function	Subtrahieren	(	opL As Integer	,	opR As Integer	)
As	Integer					

- Die Funktion wird mit dem Namen aufgerufen.
- Dem Namen folgen die zu übergebenen Parameter. Die Parameter werden durch ein Kommata getrennt.
- Die Parameterliste wird durch die runden Klammern begrenzt.

Übergabe von Parametern



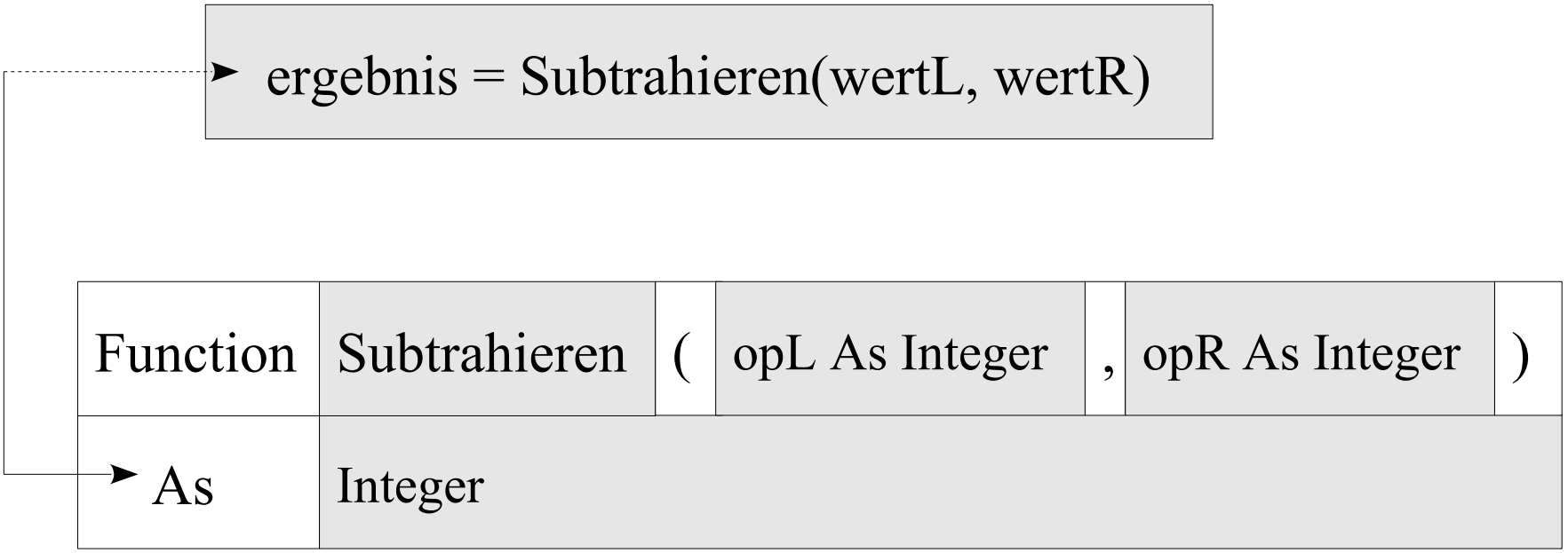
- Für jeden Parameter in der Liste im Funktionskopf wird ein Wert übergeben.
- Die Parameter des Aufrufs werden von links nach rechts dem Parametern im Funktionskopf zugeordnet.

Aufruf und Rückgabewert

```
Sub Starten()

    rueck = Subtrahieren(wertL, wertR)

End Sub
```

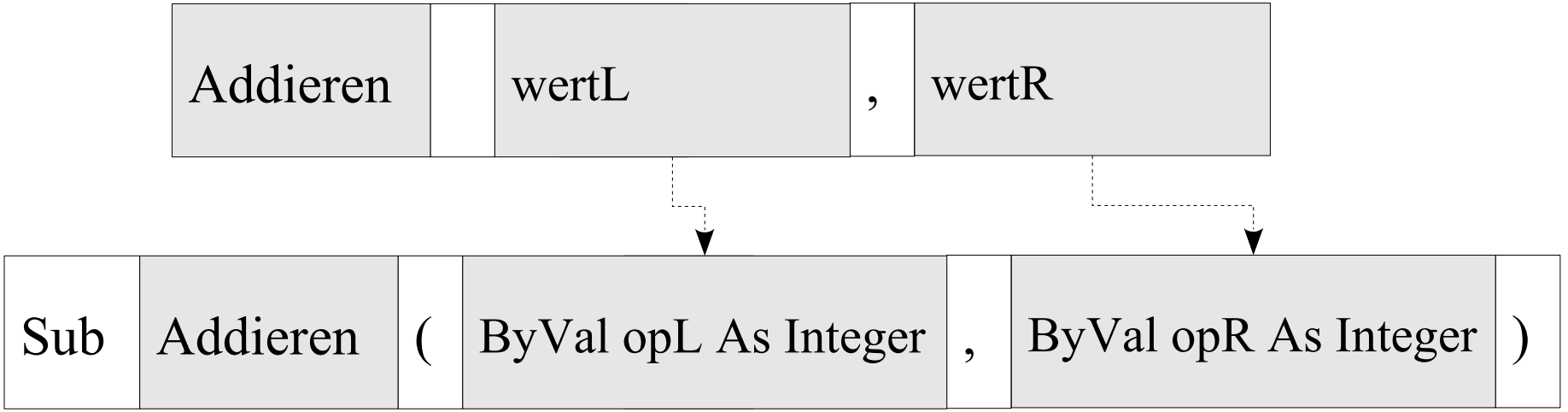
```
Function Subtrahieren(operandL As Integer, operandR As Integer)
As Integer

    ergebnis = operandL - operandR

    Subtrahieren = ergebnis
End Function
```

Datentyp des Rückgabewertes



- Der Rückgabewert entspricht dem Datentyp der Funktion.
- Die Variable, die den Rückgabewert speichert, sollte den gleichen Datentyp wie die Funktion besitzen.

Übergabe von Parametern als Wert

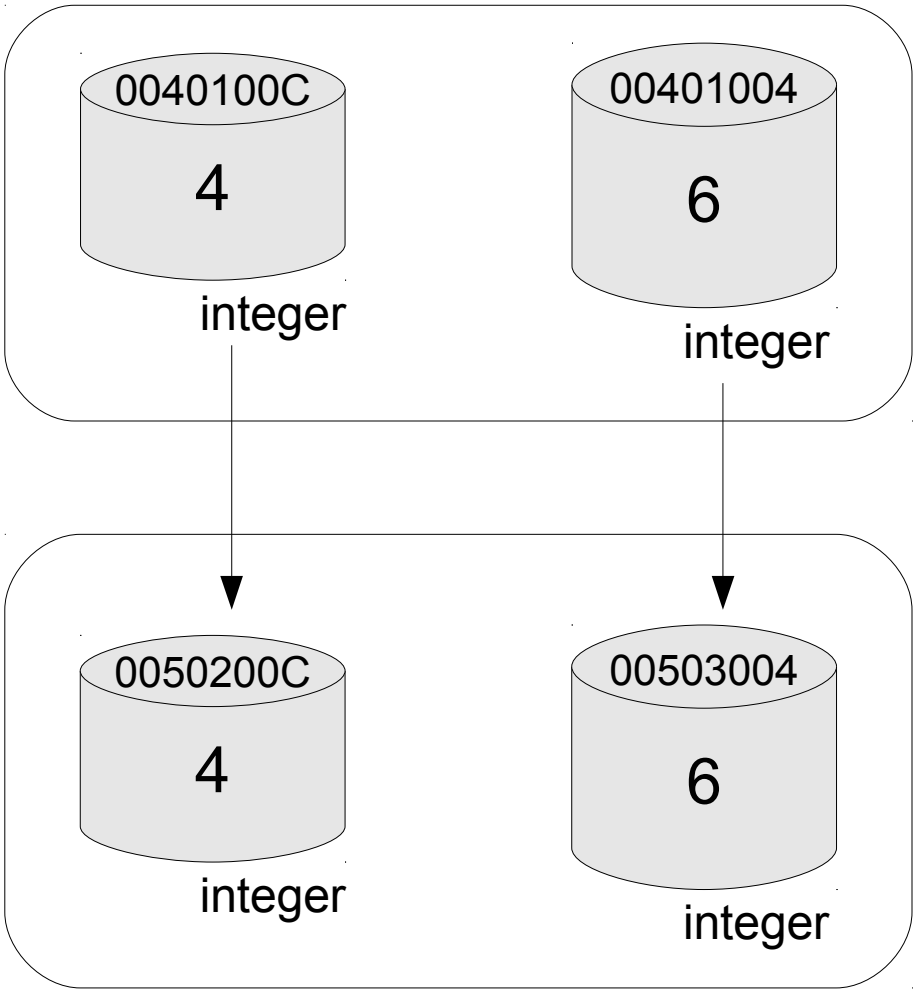


- Kennzeichnung mit dem Schlüsselwort ByVal vor dem Namen des Parameters.
- In dem Parameter der Subroutine wird eine Kopie des übergebenen Wertes gespeichert.
- Die Variablen in der aufrufenden Funktion werden nicht verändert.

„call by Value“

Der Wert der zu übergebenen Variablen wird beim Aufruf

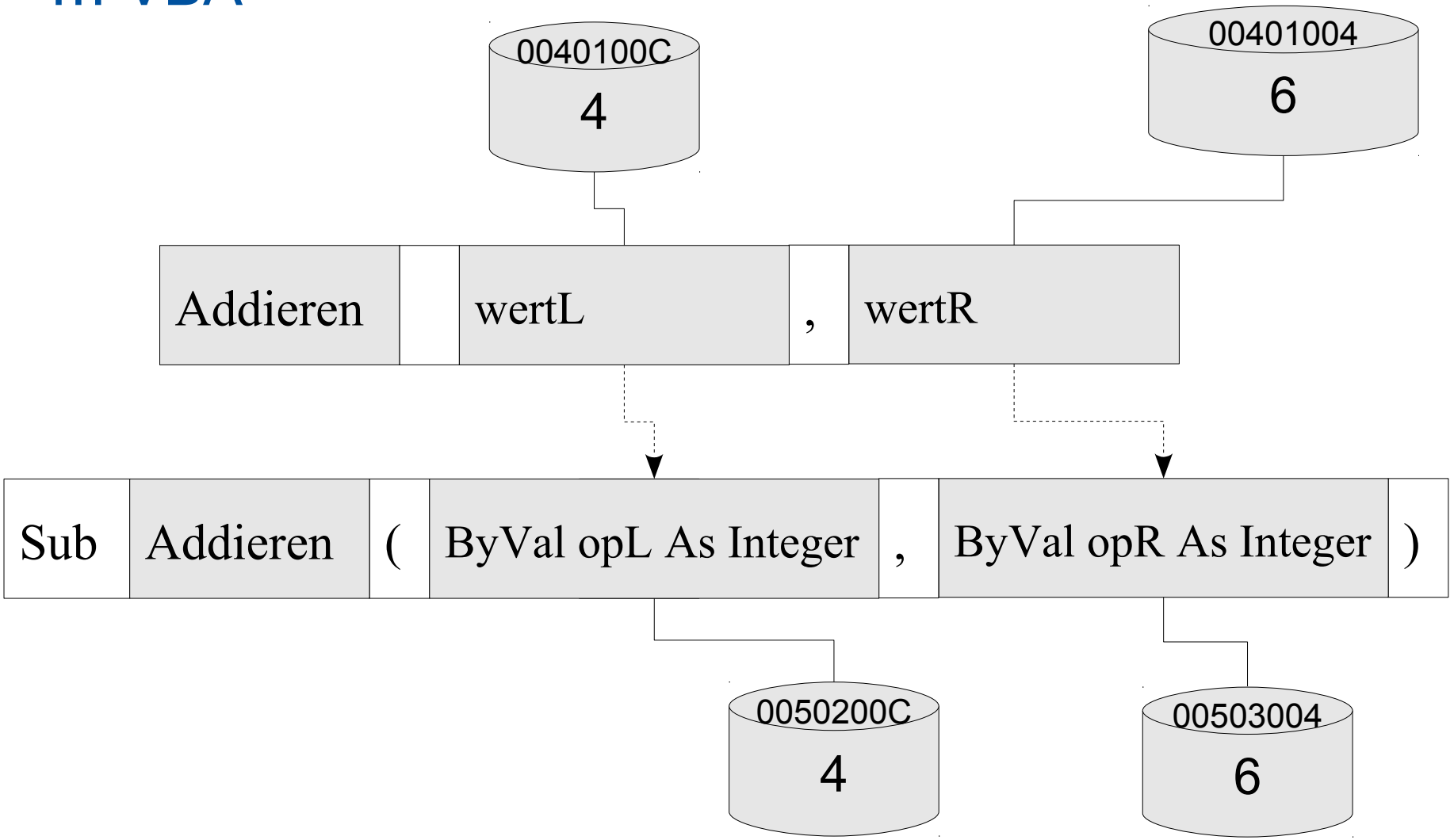
… in den dazugehörigen Parameter gespeichert.



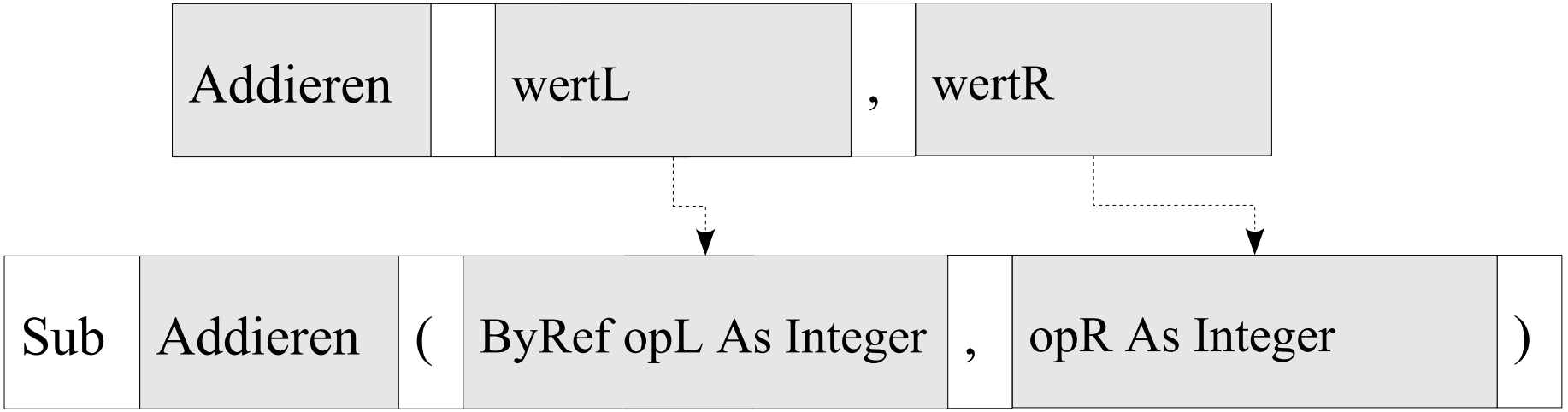
Erläuterung

- Der Wert einer Variablen, einer Konstanten oder eines berechneten Ausdrucks wird an ein Parameter im Kopf einer Subroutine übergeben.
- Der Wert der Quelle und der Wert des Parameters / des Ziels werden an unterschiedlichen Stellen im Speicher abgelegt.
- Die Quelle kann durch die Subroutine nicht verändert werden. Der Wert der Quelle ist schreibgeschützt. Nur der Wert im Parameter kann durch die Anweisungen in der dazugehörigen Subroutine verändert werden.

… VBA


0040100C
4
00401004
6
Addieren
wertL
,
wertR
Sub
Addieren
(
ByVal opL As Integer
,
ByVal opR As Integer
)
0050200C
4
00503004
6

Übergabe von Parametern als Referenz

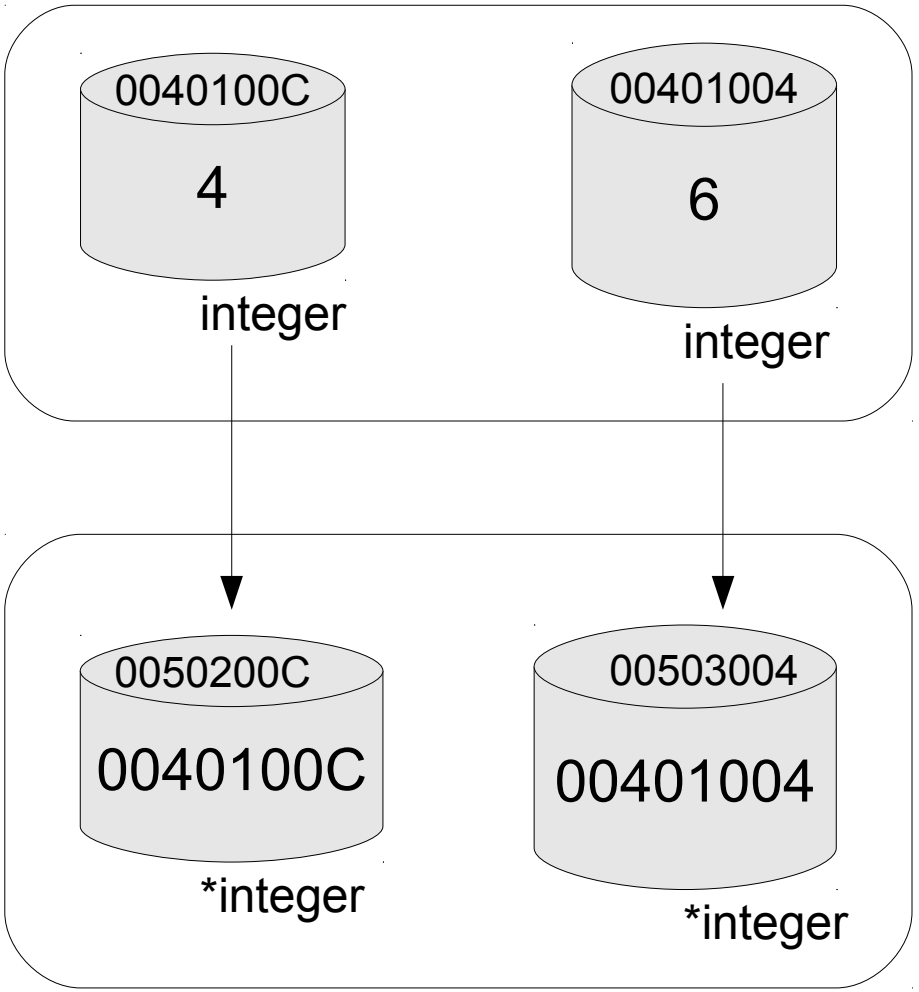


- Kennzeichnung mit dem Schlüsselwort ByRef vor dem Namen des Parameters.
- Standard-Übergabe an den Parameter. Das Schlüsselwort ByRef muss nicht angegeben werden.
- Dem Parameter wird eine Referenz auf das Argument übergeben.

„call by reference“

Die Adresse der zu übergebenen Variable im Speicher

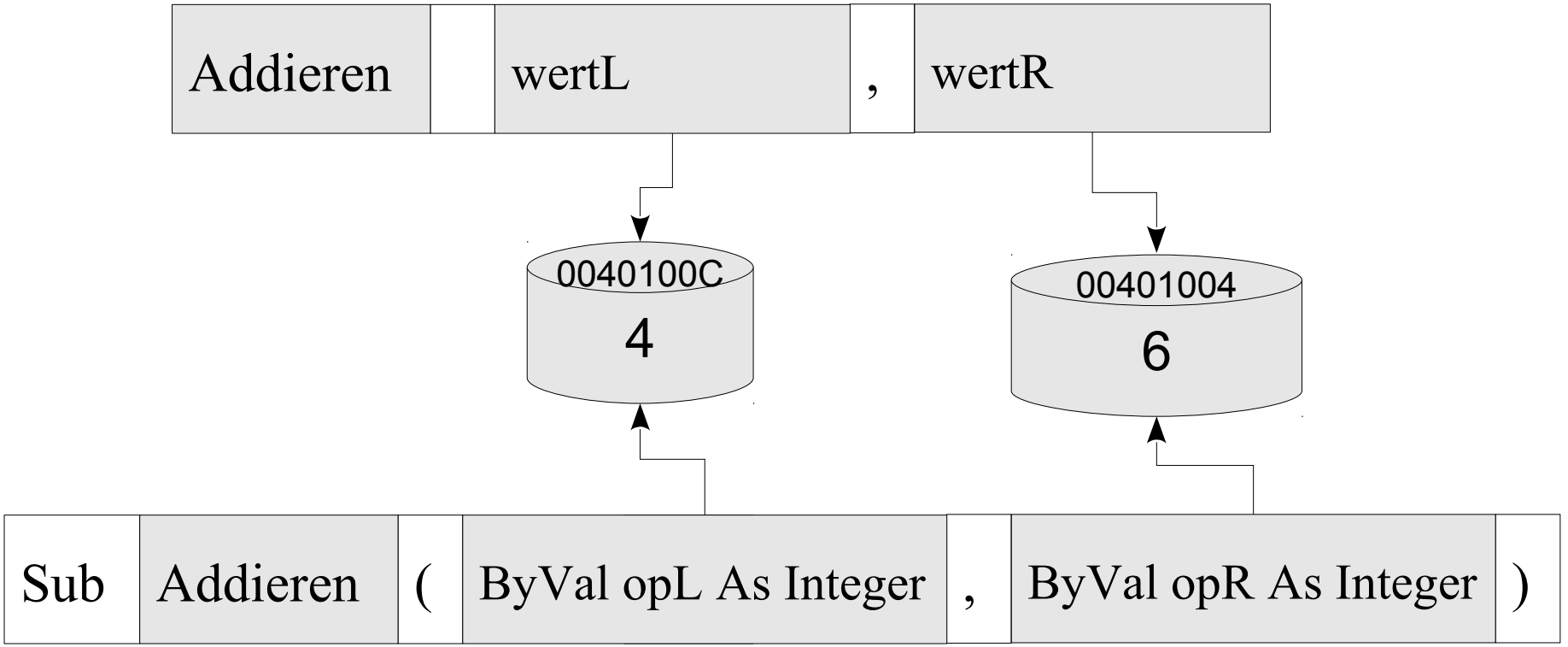
… wird an den dazugehörigen Parametern übergeben.



Erläuterung

- Der Wert einer Variablen, einer Konstanten oder eines berechneten Ausdrucks wird an einer Speicherstelle abgelegt.
- An den Parameter wird ein Verweis auf den Beginn des Speicherbereichs der Quelle übergeben
- Die Quelle sowohl als auch der Zielparameter greifen auf die gleiche Stelle im Speicher zu. Aufrufer sowohl als auch die Anweisungen in der dazugehörigen Subroutine können den Wert an der Position im Speicher ändern.

… in VBA


Addieren
wertL
,
wertR
0040100C
4
00401004
6
Sub
Addieren
(
ByVal opL As Integer
,
ByVal opR As Integer
)

Lebensdauer einer Variablen

- Wie lange existiert eine Variable?
- Die Lebensdauer kann sich auf das Modul (die Datei) oder die Subroutine beziehen.
- Bei der Geburt wird Speicher für die Variable reserviert, der beim Tod freigegeben wird.

Globale Variablen am Anfang eines Moduls

- Globale Variablen sind für das Modul (die Datei), in der sie deklariert sind, gültig.
- Konstanten, die im gesamten Programm benötigt werden, werden global am Anfang der Datei deklariert.
- Auf globale Variablen sollte verzichtet werden.

```
Option Compare Database
Option Explicit

Const dblPi As Double = 3.1415
Public umfang As Double

    Sub KreisFlaeche()

    End Sub
```

Öffentliche Variablen

- Variablen oder Konstanten sind mit dem Zugriffsspezifikator Public gekennzeichnet.
- Die Variablen und Konstanten können von allen Modulen in dem Projekt genutzt werden.
- Standard-Deklaration.

```
Option Compare Database
Option Explicit

Private Const dblPi As Double = 3.1415
Public flaeche As Double

Sub KreisFlaeche()
   modKreis.umfang = radius * dblPI
End Sub
```

Nutzung der öffentlichen Variablen

```
Option Compare Database
Option Explicit

Private Const dblPi As Double = 3.1415
Public flaeche As Double

    Sub KreisFlaeche()
        flaeche = radius * dblPI
    End Sub
```

```
Option Compare Database
Option Explicit

    Sub Starten
      Debug.Print modKreis.flaeche

    End Sub
```

Aufruf: [Modulname].[Variable]

Private Variablen

- Variablen oder Konstanten sind mit dem Zugriffsspezifikator Private gekennzeichnet.
- Die Variablen und Konstanten können nur von Prozeduren in diesem Modul genutzt werden.
- Ein Zugriff von außen (von anderen Modulen) ist nicht möglich.

```
Option Compare Database
Option Explicit

Private Const dblPi As Double = 3.1415
Private flaeche As Double

    Sub KreisFlaeche()
        flaeche = radius * dblPI
    End Sub
```

Nutzung der privaten Variablen

```
Option Compare Database
Option Explicit

Private Const dblPi As Double = 3.1415
Private flaeche As Double

    Sub KreisFlaeche()
        flaeche = radius * dblPI
    End Sub
```

```
Option Compare Database
Option Explicit

    Sub Starten
      Debug.Print modKreis.flaeche

    End Sub
```

Fehler: Methode oder Datenbankobjekt nicht gefunden

Lokale Variablen in einer Subroutine

```
Sub Starten()
    Const zaehlerBeginn As Integer = 1
    Dim zaehler As Integer
    Dim abstand As Integer
```

- Lokale Konstanten beginnen wie alle Konstanten mit dem Schlüsselwort Const. Ein weiterer Zugriffsspezifikator ist nicht nötig.
- Lokale Variablen beginnen mit dem Schlüsselwort Dim.

Erläuterung

- Lokale Variablen und Parameter sind in der Subroutine gültig und sichtbar, in der sie deklariert sind.
- Sobald die Subroutine aufgerufen wird, wird Speicherplatz für die Variable bereitgestellt.
- Sobald die Subroutine verlassen wird, wird der Speicherplatz wieder freigegeben. Ein Zugriff auf die Variable ist nicht möglich. Die Variable wird gelöscht.

Statische lokale Variablen in einer Subroutine

```
Sub GetZaehler()
    Static zaehler As Integer
```

- Statische lokale Variablen beginnen mit dem Schlüsselwort Static.
- Statische lokale Variablen werden immer mit dem Standardwert initialisiert.

Erläuterung

- Statische Variablen sind in der Subroutine gültig, in der sie deklariert sind.
- Sobald die Subroutine erstmalig aufgerufen wird, wird Speicherplatz für die statische Variable bereitgestellt.
- Sobald die Subroutine verlassen wird, wird der Speicherplatz bei einer statischen Variablen nicht freigegeben. Beim nächsten Aufruf der Subroutine ist ein Zugriff auf den gültigen Wert möglich.
- Sobald das Modul verlassen wird, wird der Speicherplatz für eine statische Variable freigegeben.