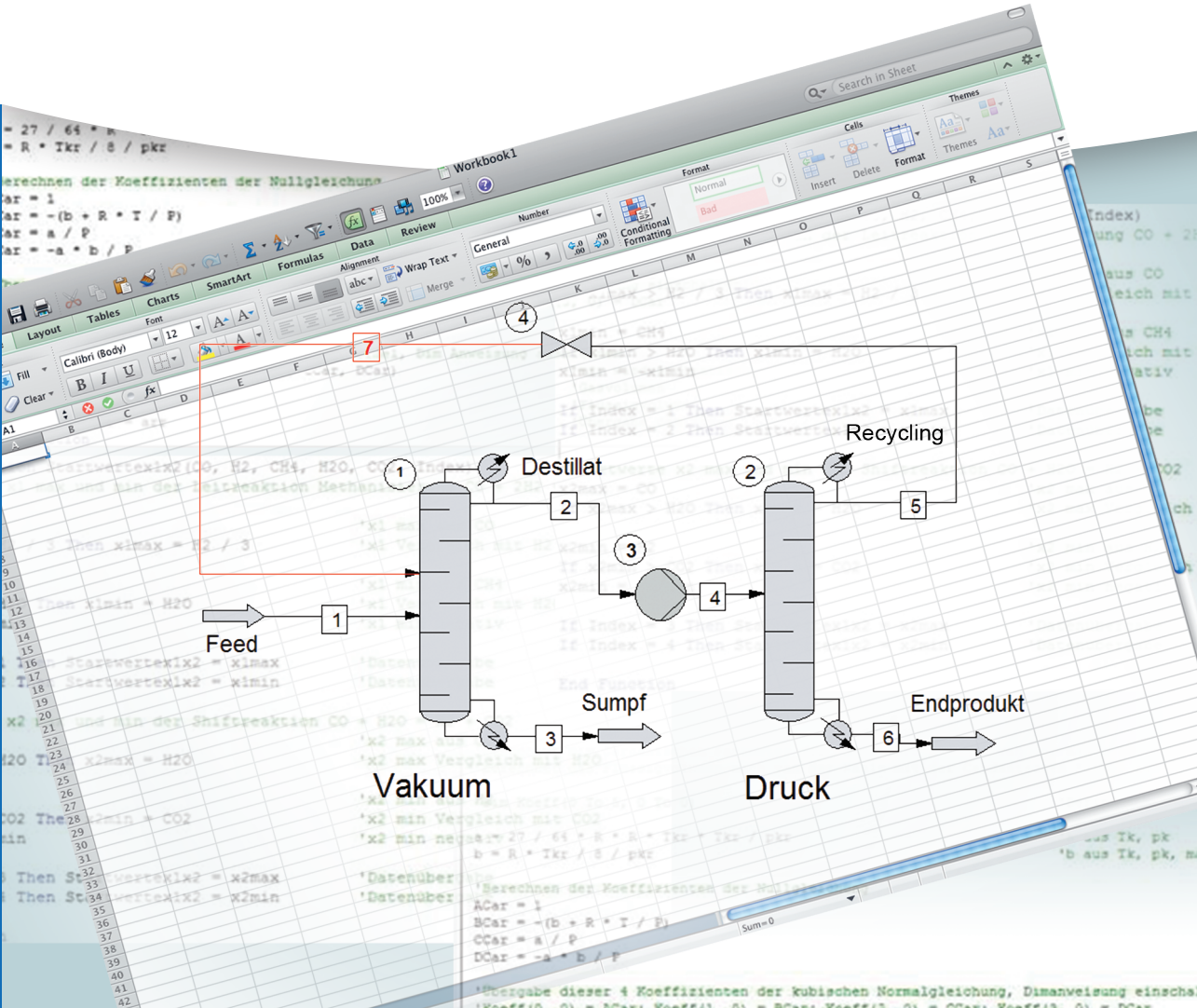
WILEY-VCH

Shichang Wang und Wolfgang Schmidt

Berechnungen in der Chemie und Verfahrenstechnik mit Excel und VBA



Shichang Wang
Wolfgang Schmidt

Berechnungen in der Chemie und Verfahrenstechnik mit Excel und VBA

Beachten Sie bitte auch weitere interessante Titel zu diesem Thema

Green, J., Bullen, S., Bovey, R., Alexander, M.

Excel 2007 VBA Programmer's Reference

2007
Print ISBN: 978-0-470-04643-2; also available in electronic formats

Billo, E.

Excel for Chemists

A Comprehensive Guide, Third Edition (with CD-ROM)

3. Auflage
2011
Print ISBN: 978-0-470-38123-6; also available in electronic formats

Billo, E.

Excel for Scientists and Engineers

Numerical Methods

2007
Print ISBN: 978-0-471-38734-3; also available in electronic formats

Mansfield, R.

Mastering VBA for Microsoft Office 2013

2013
ISBN: 978-1-118-69512-8

Shichang Wang
Wolfgang Schmidt

Berechnungen in der Chemie und Verfahrenstechnik mit Excel und VBA

WILEY-VCH
Verlag GmbH & Co. KGaA

Autoren
Prof. Dr. Shichang Wang
Hochschule Niederrhein
SWK-Energiezentrum E^2
Thermische Verfahrenstechnik
Reinarzstraße 49
47805 Krefeld
Germany

Dipl.-Ing. Wolfgang Schmidt
Südstr. 39
46562 Voerde
Germany

Print ISBN: 978-3-527-33716-3

ePDF ISBN: 978-3-527-68024-5

ePub ISBN: 978-3-527-68023-8

Mobi ISBN: 978-3-527-68025-2

oBook ISBN: 978-3-527-68022-1

1. Auflage 2015

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Umschlaggestaltung Bluesea Design, Simone Benjamin, McLeese Lake, Canada
Satz Reemers Publishing Services GmbH, Krefeld
Druck und Bindung Markono Print Media Pte Ltd, Singapore

Gedruckt auf säurefreiem Papier.

Für meinen Vater Zhixun Wang und meine Mutter Duolan Zheng aus dem Dorf Tengyuan, China – Shichang Wang

Für meine Kinder Annette und Christoph –
Wolfgang Schmidt

Berechnungen in der Chemie und Verfahrenstechnik mit Excel-VBA

Es gehört zu den täglichen Aufgaben von Chemikern und Ingenieuren, Berechnungen durchzuführen. Für einige spezielle Anwendungen stehen kommerzielle Softwareangebote zur Verfügung, weniger jedoch für die vielen kleineren, täglichen Aufgabenstellungen. In diesem Buch werden Lösungen in Excel und VBA genau dafür vorgestellt.

Schwerpunkt sind numerische Methoden und deren Anwendungen in der Chemie und Verfahrenstechnik. Dazu gehören numerische Integration und Differenzierung, Lösung von Differentialgleichungen, Lösungen linearer und nicht-linearer Gleichungen durch Matrixberechnung. Beispiele sind u.a. Volumenberechnungen von Behältern, Berechnung zwischenmolekularer Potentiale, Phasengleichgewichte, chemische Reaktionen, instationäre Wärmeleitung, Maßeinheiten, Zustandsgleichungen, Optimierung. Alle Berechnungen werden in Excel in Kombination mit der Programmierung VBA realisiert. In einer Schritt für Schritt Vorgehensweise erlernt der Leser nicht nur die Möglichkeiten der Numerik, sondern auch das Programmieren kennen und schätzen. Vorkenntnisse sind nicht erforderlich. Sowohl beim Studium als auch in der Praxis ist dieses Buch sehr hilfreich.

Vorwort

Berechnungen in der Chemie oder Verfahrenstechnik sind heute wichtiger denn je. Durch Berechnung und Simulation kann man Experimente im Voraus testen und später begleiten. Die Literatur ist voll von Berechnungsmethoden, deren Nutzung aber ohne entsprechende Hilfsmittel meist am Aufwand scheitern. Excel in Kombination mit VBA bietet sich geradezu an, die allgemein als schwierig empfundenen numerischen Methoden kennenzulernen.

Jedes Thema wird optimal in sich geschlossen behandelt, so dass sich der Leser mit jedem beliebigen Thema beschäftigen kann, ohne ein vorheriges Themen gelesen haben zu müssen.

Sowohl eine vollständige Einführung als auch eine Beschreibung der Möglichkeiten der Programmiersprache VBA würde den Rahmen dieses Buches sprengen. Diesbezüglich wird auf die allgemeine Literatur verweisen. Es seien hier die

Verlage Addison-Wesley, Hanser, Mikrosoft als auch M&T genannt, um nur einige zu nennen. Anregungen zu diesem Buch wurden insbesondere den Büchern von Ebert, Ederer: Computerberechnungen in der Chemie und Müller-Erlwein: Computeranwendungen in der chemischen Reaktionstechnik sowie einigen Lehrbüchern entnommen.

Die Excel-VBA-Geschichte, kurz erzählt.

Excel wie seine historischen Vorgänger Lotus Symphony waren ursprünglich für kaufmännische Anwendungen erstellt worden, die ersten Computer übrigens auch. Während Borland einen eigenen C-Compiler und eine Tabellenkalkulation im Quellcode lieferte, brachte Microsoft das erste Excel auf den Markt. Damit ließen sich tabellarische Daten leicht horizontal und vertikal summieren, was zur Kontrolle von Materialbilanzen z.B. einer Chemieproduktion außerordentlich hilfreich war. Für häufig wiederkehrende Vorgänge wurde eine Makrosprache eingeführt, die aber wenig intuitiv war. Bill Gates mochte Basic persönlich sehr. Deshalb befand sich stets ein Basic Interpreter auf jeder DOS-Diskette eines PCs.

Im Gegensatz zum langsamen Interpreter konnte man mit den Basic Compilern von Microsoft, Power Basic oder Turbo Basic Basic-Programme deutlich beschleunigen. Im Vergleich zum damals in der Technik häufig verwendeten FORTRAN war die Programmsprache Basic erheblich einfacher.

Als B. Gates Visual Basic 4 herausbrachte, begann der zweite Siegeszug von Basic. Sehr bekannt ist die Version 6, der VB6 Compiler. Damit werden industriell gefertigte Programme erstellt, die denen in C++ erstellten in nichts nachstehen.

Basic als VBA (Visual Basic for Application) ersetzte die alte Makrosprache in Excel. Die Stärke von VBA in Excel besteht heute darin, dass die Oberfläche von Excel mit VBA kombiniert werden kann. Dabei ist besonders erwähnenswert, dass aus einer Ausführung in Excel unmittelbar ein VBA-Programm, das sog. Makro erstellt werden kann, das sog. Makroaufzeichnen. Das ist auch zum Erlernen von VBA sehr hilfreich. Der Befehlsumfang von VBA ist im Vergleich zum alten Quick Basic enorm gewachsen. In Kombination mit DLL ist man fast so stark wie der VB6 Compiler. Programme in VB6 laufen allerdings ohne Excel.

Zu diesem Buch gehört selbstverständlich eine Excel-Datei: „Excel-VBA-Chemie.xlsm“ in der Version Office 2007. Diese Excel-Datei ist nicht geschützt und frei verwendbar. In den einzelnen Kapiteln wird die jeweils dazu gehörende Excel Tabelle angegeben, so dass diese leicht gefunden werden kann.

Einige Themen und Basic-Berechnungen wurden den Büchern von Müller-Erlwein und Ebert, Ederer entnommen.

Inhaltsverzeichnis

1 Funktionen für Excel und VBA

Sie lernen hier, wie man VBA in Excel aufruft und damit arbeitet. Sie erstellen eine Funktion und ein Makro und verwenden diese in Excel.

1.1 Erstellen einer VBA Funktion

Tabelle: Polynom

Modul: Polynom

VBA: Polynomf

Wir wollen hier eine Polynom-Funktion sowohl in Excel als auch in VBA erstellen und dann bewerten.

Starten Sie Excel 2007 und tragen Sie in Zelle B3 und B4 jeweils die Zahl 1 ein.

Nun schreiben Sie in die Zelle C3 1 + 2*B3+3*B3*B3:

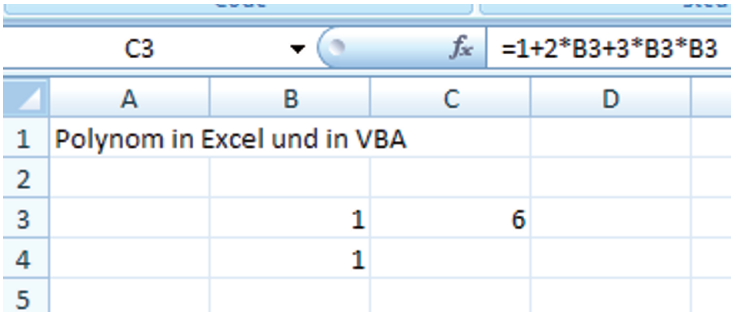


Abb. 1.1-1 Polynom Daten

Dies wird ebenfalls in die Bearbeitungsleiste (fx = …) übertragen. Das Ergebnis steht unmittelbar zur Verfügung.

Nun wenden wir uns der Alternative zu, nämlich die gleiche Berechnung in VBA durchzuführen.

Zum Start von VBA in Excel drücken Sie Alt + F11 oder wählen „Entwicklungstools“ , „VisualBasic“. Damit öffnet sich die Oberfläche von VBA.

Anm.: Wir gehen hier von einer leeren Excel-Mappe aus.

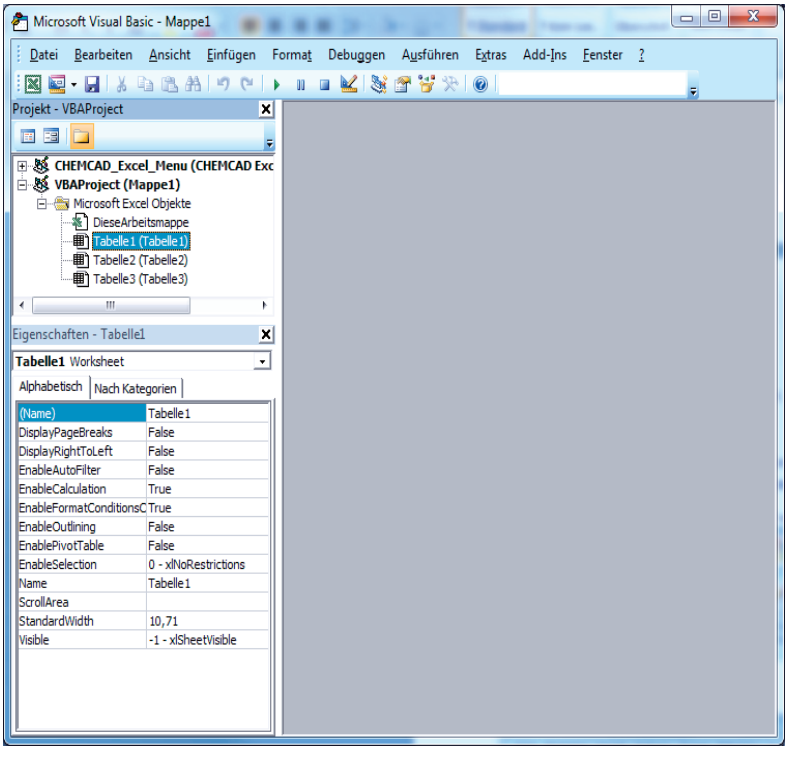


Abb. 1.1-2 Oberfläche Visual Basic (VBA)

Falls die rechte Fläche grau ist, klicken Sie doppelt auf „Tabelle1 (Tabelle1)“. Die rechte Fläche wird nun weiß. Wählen Sie nun „Einfügen“ im Hauptmenü und „Modul“.

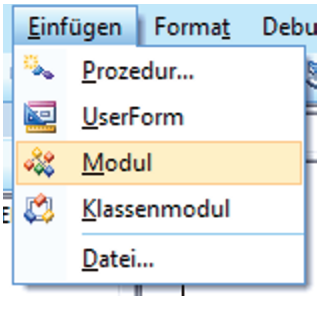


Abb. 1.1-3 Einfügen Modul

Excel hat das Modul links oben in das VBA-Projekt Fenster hinzugefügt:

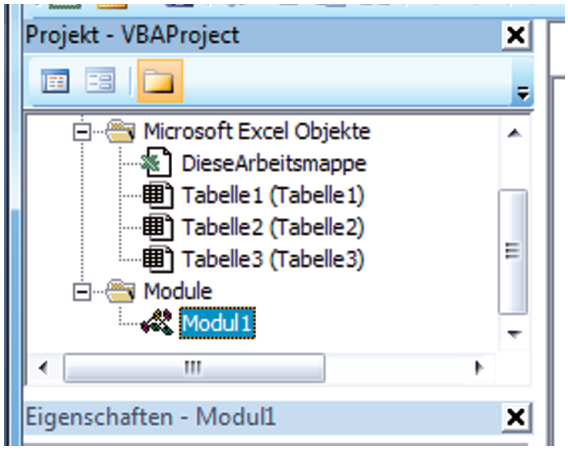


Abb. 1.1-4 Neues Modul

Im Eigenschaftsfenster tragen Sie den Namen „Polynom“ ein.

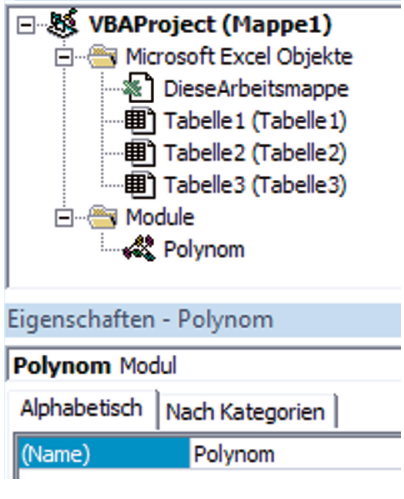


Abb. 1.1-5 Polynom als Modul

Klicken Sie doppelt auf „Polynom“ im Projektfenster. Der Cursor blinkt im rechten großen Fenster. Wählen Sie nun „Einfügen“ und „Prozedur“:

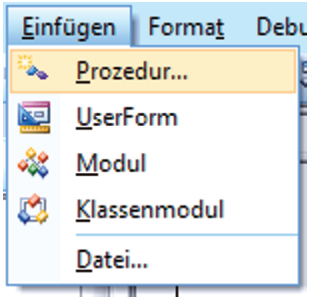


Abb. 1.1-6 Einfügen, Prozedur

Es erscheint ein neues Fenster.

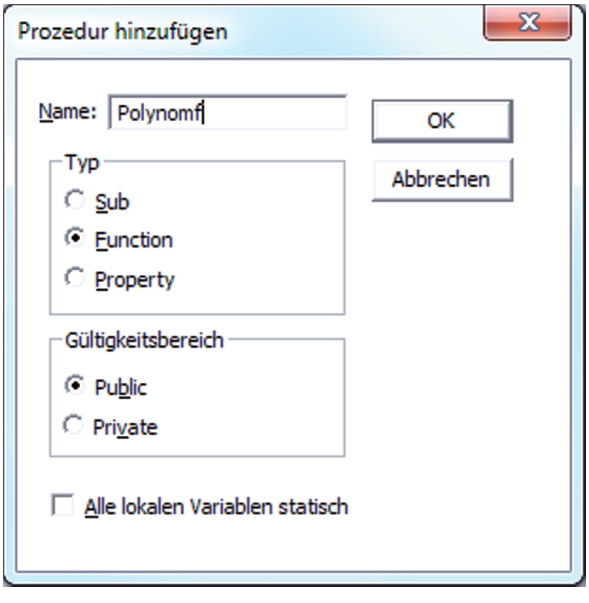


Abb. 1.1-7 Prozedur hinzufügen

Wählen Sie „Function“ und geben Sie den Namen der Funktion (Polynom) ein, OK.

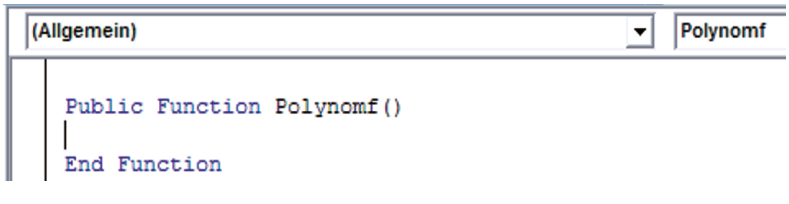


Abb. 1.1-8 Leere Funktion eröffnet

Die erste und letzte Zeile wird automatisch erstellt. Nun können Sie Ihr Programm schreiben.

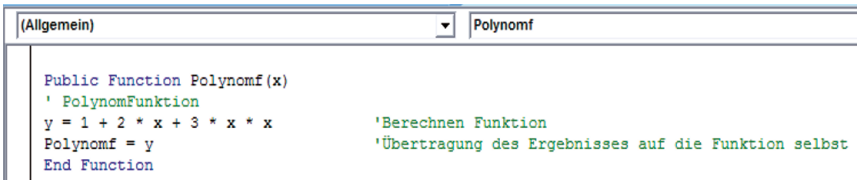


Abb. 1.1-9 Funktion Polynom

Die Zeile „PolynomFunktion“ ist ein Kommentar, weil sie mit einem Apostroph (Shift #) beginnt. Die Zeile

y = 1 + 2 * x + 3 * x * x

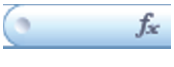
ist die eigentliche mathematische Funktion. Rechts davon steht ein Kommentar, der ebenfalls mit einem Apostroph beginnt.

Die Programmzeile

Polynomf = y

überträgt das Ergebnis in die Funktion selbst. Unter diesem Namen finden Sie diese Funktion in Excel automatisch. Das werden wir jetzt in Excel ausführen.

Auf dem PC sind in der unteren Leiste 2 Fenster von Excel offen, zwischen denn Sie hin und her springen können. Dies ist einerseits das Fenster der Excel-Tabelle, andererseits das VBA-Fenster. Wählen Sie nun das Excel-Fenster, das VBA Fenster wird aber nicht geschlossen. Geben Sie in Zelle B4 die Zahl 1 ein. Klicken Sie auf die Zelle C4. und auf „fx“ links von der Bearbeitungssleiste.



Es erscheint das Menü „Funktion suchen“. Wählen Sie „Benutzerdefiniert“ und darin „Polynomf“.

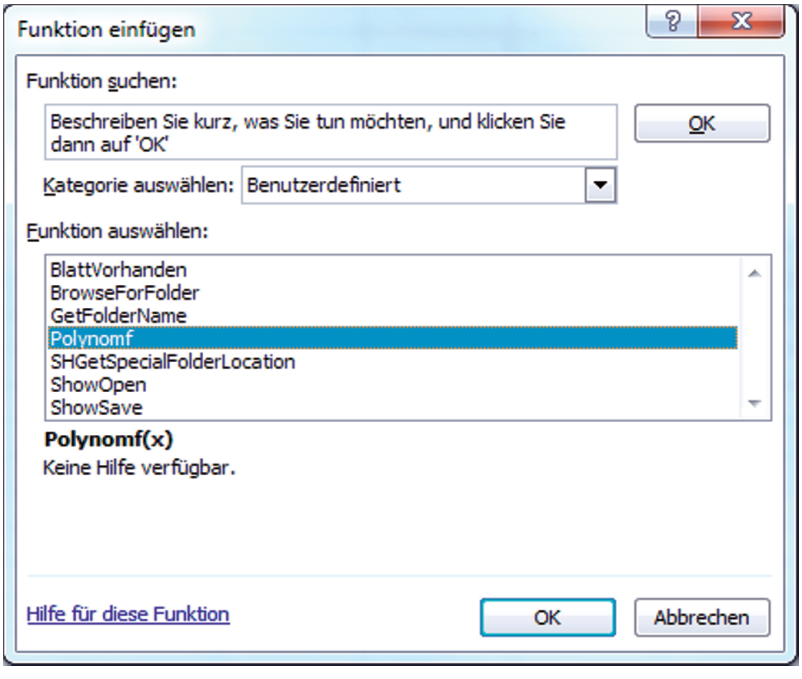


Abb. 1.1-10 Funktion einfügen

Anm.: die weiteren hier angezeigten Funktionen befinden sich (noch) nicht auf Ihrem PC.

Es erscheint:

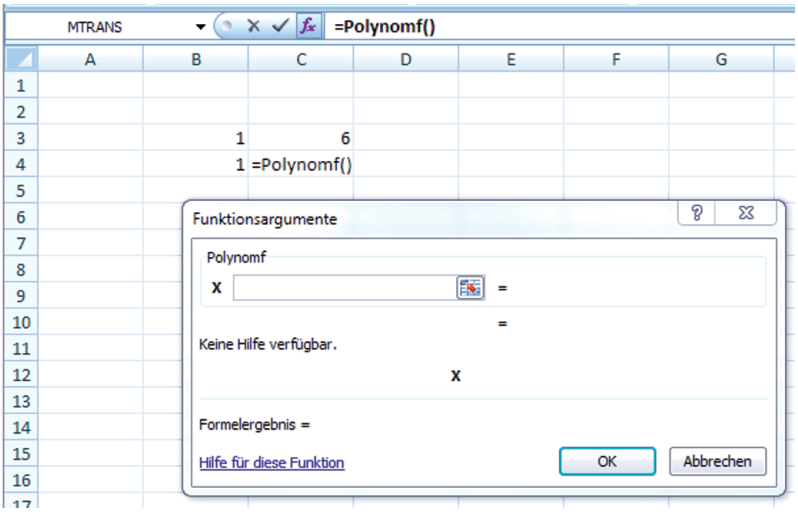


Abb. 1.1-11 Funktionsargumente

Der Mauscursor sollt im Menü „Funktionsargumente“ neben dem X stehen und blinken. Falls nicht dort anklicken. Nun klicken Sie die Zelle B4 an und dann OK.

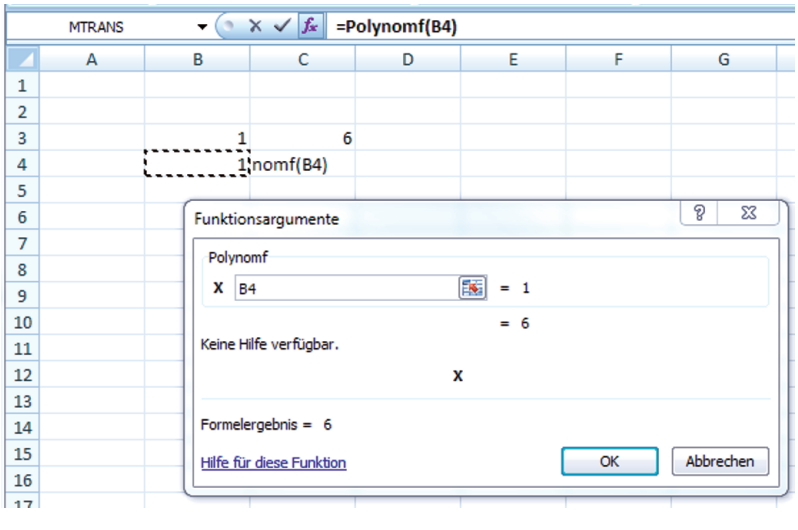


Abb. 1.1-12 Zelle zuweisen

Das Ergebnis wird bereits im Menü angezeigt: Formelergebnis = 6.

In der Exceltabelle erscheint:

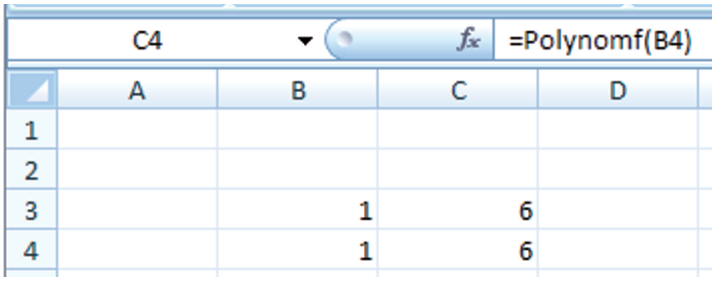


Abb. 1.1-13 Ergebnis

Sowohl die Excel-Formel in C3 als auch die VBA-Funktion in C4 ergeben dasselbe Ergebnis. Klicken Sie zur Kontrolle zwischen den Zellen C3 und C4 hin und her.

Wenn Sie nun in B3 und B4 eine andere Zahl eingeben, z.B. 10, erscheint:

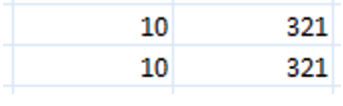


Abb. 1.1-14 Neues Ergebnis

Dies finden Sie in der Excel Datei „Excel-VBA-Chemie.xlsm“ in der Tabelle „Polynom“.

Im Vergleich beider Methoden ergibt sich folgendes. Eine Formel in Excel ist zunächst weniger aufwändig als diese in VBA zu erstellen. Ist die Formel allerdings komplexer und muss sie z.B. iterativ berechnet werden, ist VBA eindeutig im Vorteil. Für die Excel-Formel steht nämlich nur 1 Zeile zur Verfügung, während in VBA die Zahl der Zeilen unbegrenzt ist. Auch viele Möglichkeiten in Excel wie z.B. „Wenn“ und „Verweis“ lassen sich auch in VBA verwenden. So lautet z.B. der VBA Befehl für „Wenn“ schlicht „if“, der für „Verweis“ lautet „Lookup“. Dazu bietet die VBA Literatur viele Beispiele. Durch Makroaufzeichnen kann man dies leicht nachvollziehen.

Die hier beschriebene Funktion kann standardmäßig nur einen einzigen Wert übertragen. Mit der nachstehend beschriebenen Methode lässt sich das aber erweitern. Öffnen Sie dazu die Tabelle „Funktionen“.

Es wird zunächst nur ein Ergebnis angezeigt, nämlich der erste Wert einer Mehrfachvariablen, dem Array. Mathematisch betrachtet ist dies eine eindimensionale Matrix, oder auch Vektor genannt. Sie hat die Bezeichnung arr(x). Um nun tatsächlich alle 6 Daten dieses Arrays zu erhalten, müssen Sie vor Aufruf dieser Funktion die Zellen K25:K30 auswählen, d.h. markie-

ren. Dann steht in Zelle K25 dasselbe Ergebnis wie zuvor. Drücken Sie nun die F2 Taste und dann die Dreierkombination Strg+Shift+Enter.

Dadurch erhält die Funktion geschweifte Klammern und in allen markierten Zellen finden Sie nun die 6 Ergebnisse. Das Löschen erfordert einige Schritte. Markieren Sie die zu löschenden Zellen und wählen mit der rechten Maustaste „Inhalte löschen“. Dabei werden alle Zellen komplett gelöscht.

Die jetzt übertragenen Daten sind eindimensional. Sie können aber auch mehrdimensionale Daten übertragen. Dazu verwenden Sie die Funktion „arrTest“. Verfahren Sie wie zuvor, markieren Sie aber die Zellen R24:S26.

1.2 Makros, aufnehmen und bearbeiten

Tabelle: Makro

Modul: Modul1

Makro: Makro1

Ein Makro ist ein selbständig laufendes Programm. Es gibt zwei Möglichkeiten, ein Makro zu erstellen. Entweder man programmiert es selber oder man nimmt es auf. Sie lernen in den nächsten Schritten wie Sie ein Makro aufnehmen können. Dazu öffnen Sie eine Tabelle und benennen diese „Makro“. Tragen Sie in die Zellen B3 = 2 und in die Zelle C3 = 3 ein. Nun wollen wir ein Programm (Makro) erstellen, welches diese beiden Zellen multipliziert und das Ergebnis in die Zelle D3 schreibt.

Wählen Sie im Hauptmenü „Entwicklungstools“. Klicken Sie die Zelle D3 an und wählen Sie „Makro aufzeichnen“.

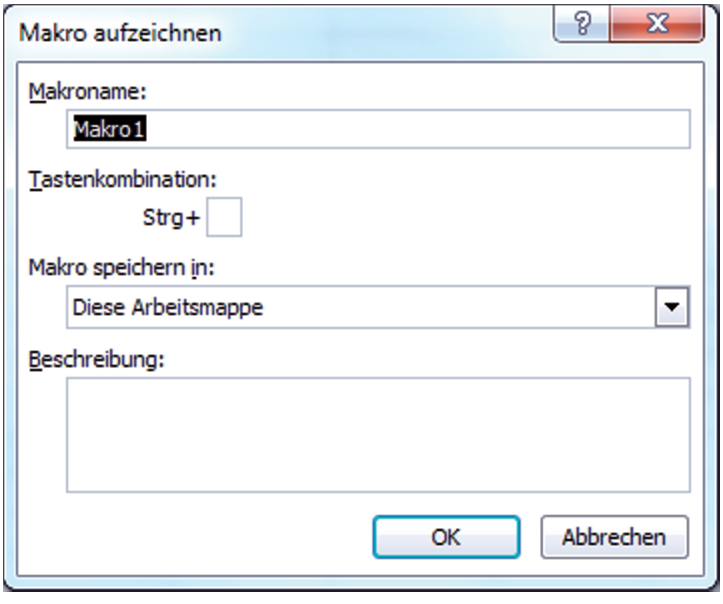


Abb. 1.2-1 Makro aufzeichnen

In Zelle D3 geben Sie die Formel ein =B3*C3..

Ergebnis:

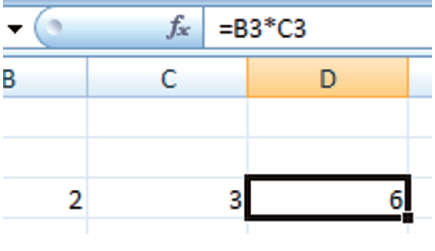


Abb. 1.2-2 Formel eingeben

Mit der Enter Taste abschließen. Dann das Makro schließen, indem Sie auf „Aufzeichnen beenden“ klicken.

Nun löschen Sie D3. Starten Sie das Makro. Dazu wählen Sie „Makro“.

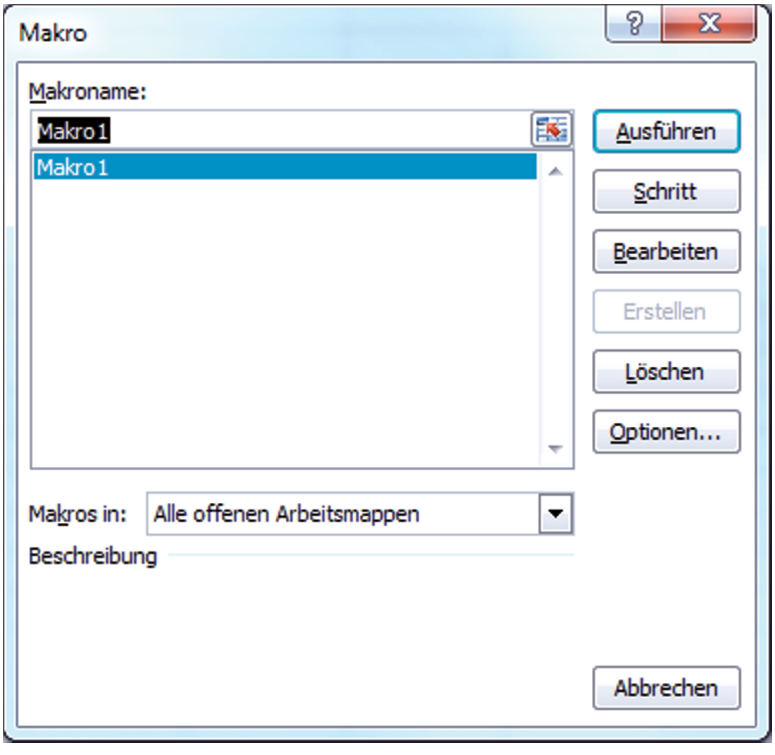


Abb. 1.2-3 Makro auswählen und Ausführen

Klicken Sie nun auf „Ausführen“. In Zelle D3 erscheint jetzt das Ergebnis.

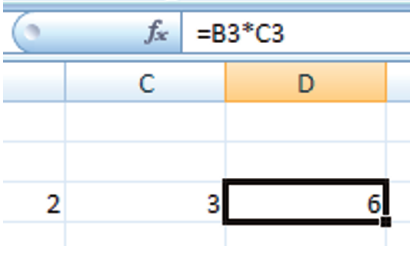


Abb. 1.2-4 Makro ausgeführt

Das Makro schreibt also in die Zelle D3 genau das hinein, was Sie zuvor auch geschrieben hatten. Nun schauen wir uns das Makro etwas näher an. Dazu klicken Sie wieder auf „Makros“.

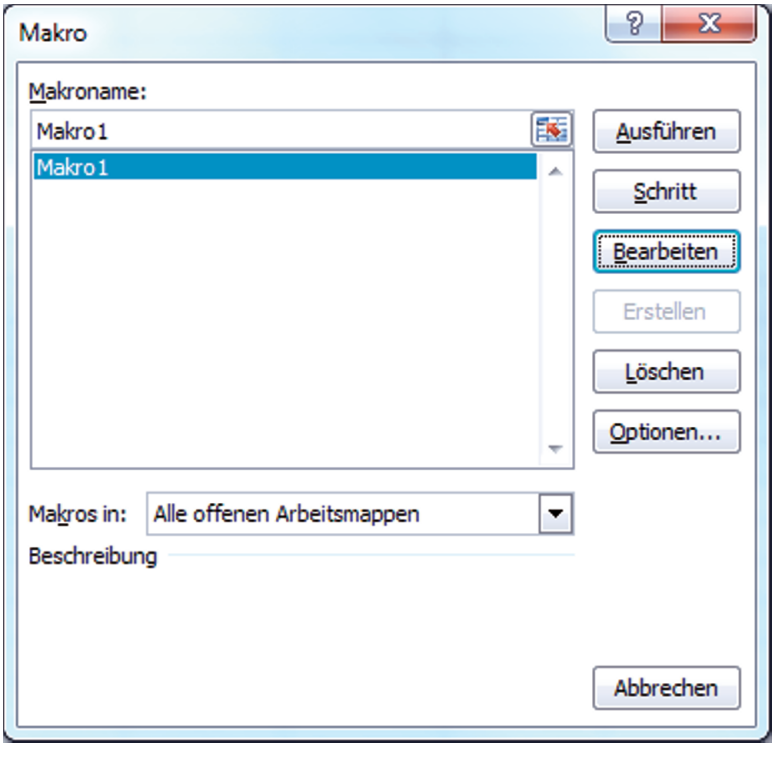


Abb. 1.2-5 Makro auswählen und bearbeiten

Jetzt klicken Sie auf „Bearbeiten“. Das VBA-Fenster wird automatisch geöffnet. Das Makro sieht möglicherweise etwas anders. Ändern Sie es wie folgt:

```
Sub Makro1()
'
' Makro1 Makro
'
    ActiveCell.FormulaR1C1 = "=RC[-2]*RC[-1]"
End Sub
```

Abb. 1.2-6 Der generierte VBA Codes des Makros

Excel generiert aus der Aufzeichnung einen VBA-Code und speichert diesen als ein Makro. Auf diese Weise lernt man den entsprechenden VBA Befehle kennen.

Die einzige wichtige Programmzeile lautet: ActiveCell.FormulaR1C1 = "=RC[-2]*RC[-1]" und bedeutet, daß in der selektierten Zelle eine Formel eingegeben wurde.

Die Schreibweise RC ist erklärungsbedürftig. R bedeutet Row (Reihe), C Column (Spalte). R1C1 ist die zuletzt markierte Zelle. RC(-2) bedeutet 2 Columns links, (-2) steht neben C, nicht neben R. C(-1) bedeutet daher 1 Zelle links von der ursprünglichen. Damit sind die Zelle **relativ** zur Ausgangsposition D3 gekennzeichnet. Welchen Vorteil hat das?

Schreiben Sie die Zahl 5 in die Zelle B2, 4 in die Zelle C6. Klicken Sie die Zelle D6 an, sodass sie markiert ist und und starten Sie das Makro erneut mit „Ausführen“. Die Berechnung wird ausgeführt. Das Ergebnis steht in Zelle D6.

Wenn Sie wollen, dass das Makro eine Berechnung nur der Zellen B4:D4 ausführt, statt eine Berechungsformel zu schreiben, dann sollten Sie es wie folgt ändern:

```
Sub Makro2()
'
    [D4] = [B4] * [C4]
End Sub
```

Abb. 1.2-7 Absolute Zellen in VBA

und den Makronamen in Makro2 ändern.

Füllen Sie die Zellen wie folgt aus und starten Sie das Makro2. Ergebnis:

	A	B	C	D
1	Makro			
2				
3		2	3	6
4		4	5	20

Abb. 1.2-8 Ergebnis Makro 2

Das Makro funktioniert jetzt nur mit den 3 Zellen B4, C4 und D4. Damit haben Sie zwei Makros, die unterschiedlich funktionieren.

Das Aufzeichnen von Makros ist eine sehr sinnvolle Einrichtung in Excel-VBA und erleichtert in vielen Fällen das Programmieren. Man lernt dabei oft VBA-Befehle schneller und einfacher kennen als durch Suchen in Fachbüchern.

Etwas umständlich ist das Starten eines Makros, so wie bisher durchgeführt. Bequem ist das Starten mit einem Start-Knopf, d.h. Command Button.

Wählen Sie „Entwicklungstools“, „Entwurfsmodus“. Links davon befindet sich „Einfügen“. Dort finden Sie die Symbole der Formularsteuerelemente und der ActiceX-Steuerelemente:

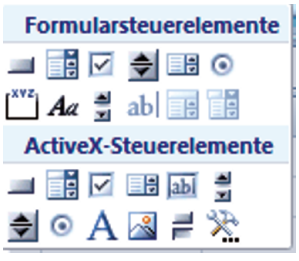


Abb. 1.2-9 Formularsteuerelemente und ActiveX-Steuerelemente

Im Bereich Formularsteuerelemente finden Sie oben links ein kleines Symbol. Halten Sie die Maus darauf und es erscheint „Schaltfläche (Formularsteuerelement)“. Klicken Sie dieses Symbol an und ziehen Sie das Symbol auf die gewünschte Größe:

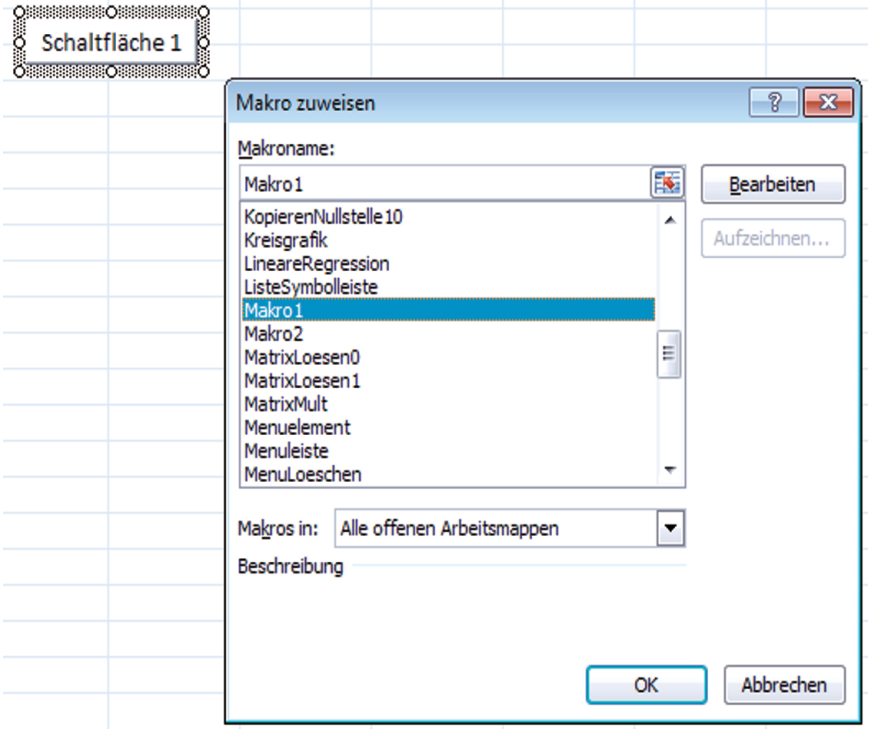


Abb. 1.2-10 Schaltfläche und Makro zuweisen

In dem angezeigten Menü wählen Sie Makro1 und dann OK. Klicken Sie den Knopf mit der rechten Maustaste an und ändern Sie den Text in „Makro1“. Klicken Sie danach erneut auf „Entwurfsmodus“. Das Symbol erhält nun wieder seine normale Ansicht.

Wählen Sie die Zelle D3 und klicken Sie auf den Knopf „Makro1“. Dadurch wird das Makro1 automatisch ausgeführt. In D3 erscheint das Rechenergebnis.

Wählen Sie wieder den Entwurfsmodus. Wenn Sie jetzt den Knopf „Makro1“ mit der Rechten Maustaste anklicken, erscheint:

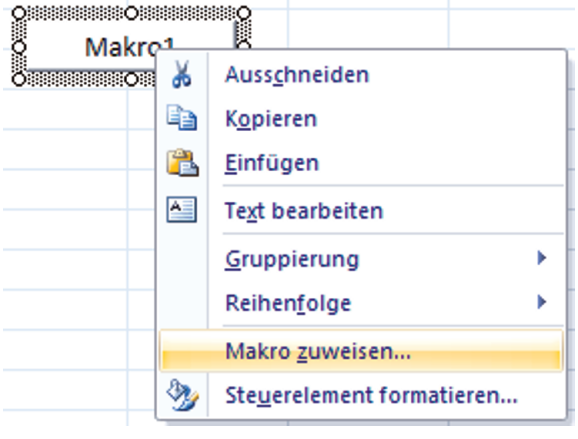


Abb. 1.2-11 Makro erneut zuweisen

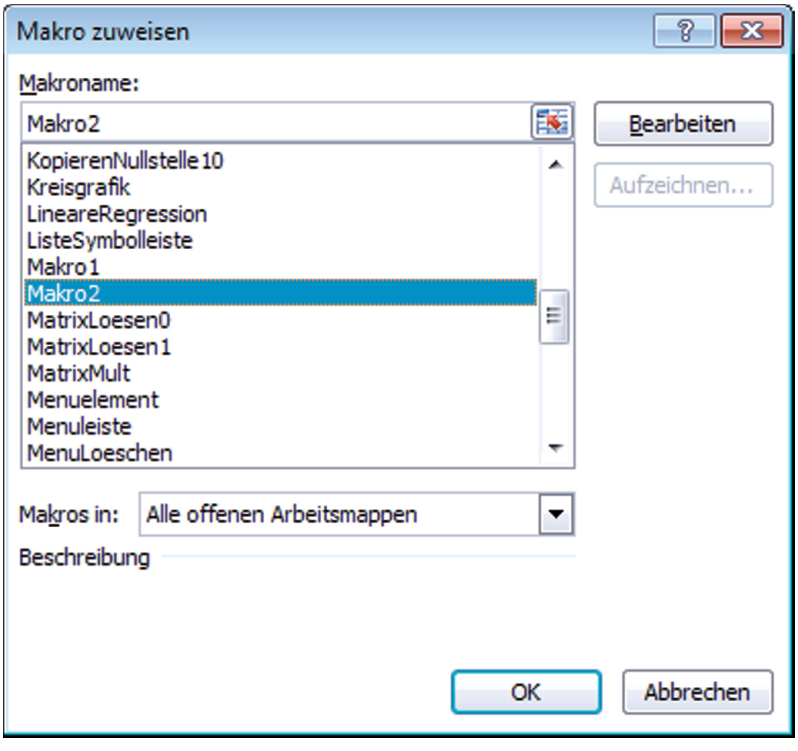


Abb. 1.2-12 Makro2 zuweisen

Wählen Sie „Makro2“. Ändern Sie den Namen des Knopfes auf „Makro2“, dann die Zahlen in B4 bzw. D4 und klicken Sie auf den Knopf „Makro2“.

In D4 erscheint das Ergebnis. Damit haben Sie die wichtigsten Möglichkeiten eines Makros kennengelernt.

1.3 Einführung in die VBA Programmierung

1.3.1 Daten in Tabellen und in VBA verbinden

Tabelle: Daten

Modul: Daten

Funktion: Beispiel1

Nachdem Sie im vorherigen Kapitel bereits gelernt haben, eine Funktion in VBA zu erstellen, lernen Sie hier an 4 Beispielen, wie Sie die Zellen in Excel mit den Variablen in VBA verbinden können. Diese Technik wird sowohl bei der Dateneingabe als auch bei der Datenausgabe benötigt. Damit wiederholen wir einen Teil des vorherigen Kapitels. Zunächst behandeln wir die Datenübertrageung selbst an Hand von 3 Beispielen, anschließend eine Datengenerierung zur grafische Darstellung eines Kreises.

Im Beispiel 1 erstellen wir die folgende mathematische Polynomfunktion:

$$y = a + b * x + c * x\text{^}2$$

Die nachfolgenden Schritte wurden grundsätzlich z.T. bereits im vorangegangenen Kapitel besprochen. Deren Anwendung wird hier an dem vorliegenden Beispiel vertieft.

Öffnen Sie mit Alt F11 das VBA Fenster und erweiteren Sie „Module“ um „Daten“. Klicken Sie dazu im Projekt-Fenster links auf „Module“ und dann auf „Einfügen“ im Hauptmenü. Wählen Sie dort „Modul“. Im Eigenschaftenfenster links unter dem Projektfenster ändern Sie den Modulnamen in „Daten“.

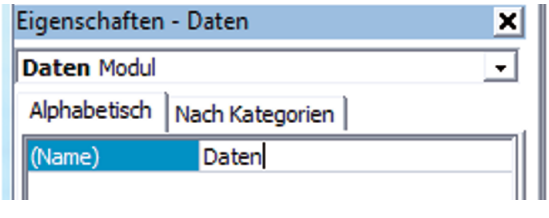


Abb. 1.3-1 Modul Daten benennen

Jetzt befindet sich das Modul „Daten in der Modulliste:

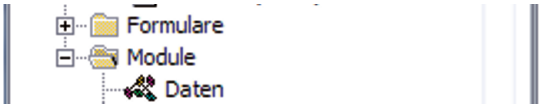


Abb. 1.3-2 Modul Daten eingefügt

Klicken Sie doppelt auf „Daten“, dann doppelt auf „Einfügen“ im Hauptmenü und wählen Sie „Prozedur“ und dann „Function“. Geben Sie als Namen „Beispiel1“ ein. VBA trägt nun automatisch das Grundgerüst ein:

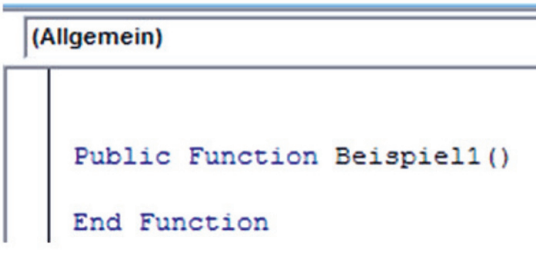


Abb. 1.3-3 VBA Code Grundmodell

Das füllen Sie nun wie folgt aus. Zunächst tragen Sie in die Funktion die folgenden 3 Variablen in die Klammern ein:

```
Public Function Beispiel1(a, b, c, x)

End Function
```

Abb. 1.3-4 Übergabe der Variablen

Damit ist die Funktion in der Lage, diese 4 Variablen einzulesen. Nun schreiben Sie die Funktion in die leere Zeile darunter:

```
Public Function Beispiel1(a, b, c, x)
y = a + b * x + c * x * x
End Function
```

Abb. 1.3-5 Vollständige Funktion

Statt x^2 müsste man schreiben x^2. Schneller rechnet VBA aber, wenn man x*x schreibt. Nun muß die Variable noch mit der Funktion verbunden werden. Schreiben Sie daher: Beispiel1 = y.

```
Public Function Beispiel1(a, b, c, x)
y = a + b * x + c * x * x
Beispiel1 = y
End Function
```

Abb. 1.3-6 Übergabe des Ergebnisses an den Funktionsnamen

Wenn Sie einen Kommentar einfügen wollen, schreiben Sie diesen in Apostroph.
Die Variablen lassen sich auch in große Buchstaben ändern.

```
Public Function Beispiel1(A, B, C, X)
'Datentransfer aus Tabelle "Daten" als
Y = A + B * X + C * X * X
Beispiel1 = Y
End Function
```

Abb. 1.3-7 Kommentar einfügen

Damit ist die Funktion fertig. Nun wechseln Sie in die Excel Tabelle „Daten“ und geben folgendes ein:

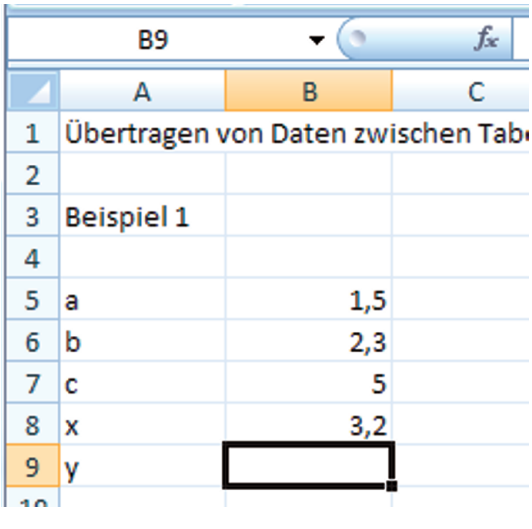


Abb. 1.3-8 Beispiel 1 Durchführung der Berechnung

Klicken Sie in die Zelle B9 und dann auf das Funktionszeichen fx

Wählen Sie „Benutzerdefiniert“

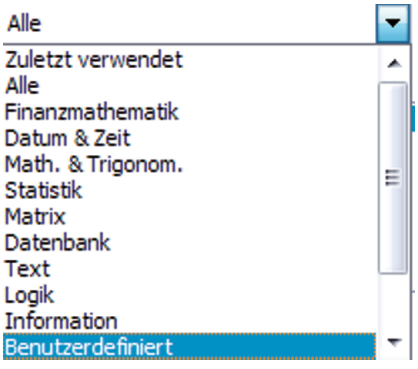


Abb. 1.3-9 Benutzerdefiniert

und dann „Beispiel1“:

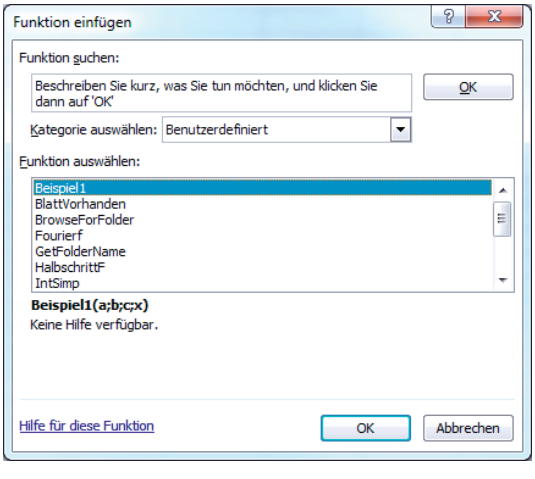


Abb. 1.3-10 Funktion einfügen

Es öffnet das Fenster „Funktionsargumente“.

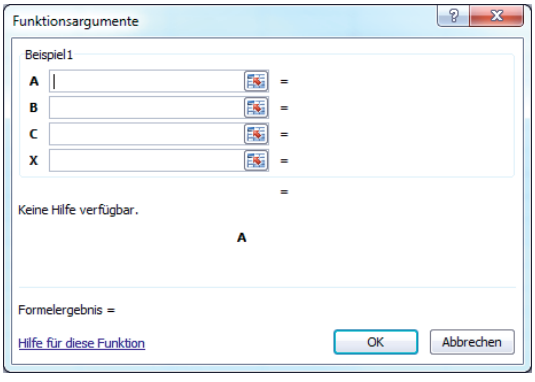


Abb. 1.3-11 Funktionsargumente

Klicken Sie in dieses Fenster in die obere Zeile „A“ und dann in Excel auf B5. Dann klicken Sie in die Zeile „B“ und auf B6, in die Zeile „C“ und auf B7 und schließlich in die Zeile „X“ und in die Zelle B8.

Ergebnis:

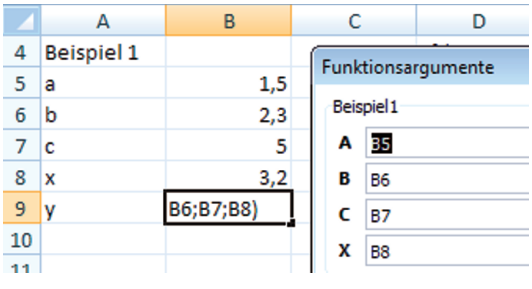


Abb. 1.3-12 Funktionsargumemnte und Excelzellen

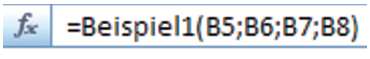
In dem Menü „Funktionsargumente“ finden Sie neben den Zeilen „A“ bis „X“ die eingegebenen Werte.

sowie unten das Ergebnis „Formelergebnis = 60,06“. Wenn Sie nun OK klicken wird das Menü geschlossen und es erscheint das Ergebnis in der Zelle B9.

4	Beispiel 1	
5	a	1,5
6	b	2,3
7	c	5
8	x	3,2
9	y	60,06

Abb. 1.3-13 Beispiel 1 Ergebnis

In der Bearbeitungsleiste finden Sie:



Es ist zwar sehr komfortabel, die Zelle B9 wie oben beschrieben mit Hilfe des Menüs „Funktionsargumente“ auszufüllen, alternativ kann man auch die Zelle B9 anklicken und die Bearbeitungsleiste direkt Feld direkt ausfüllen. Korrekturen des Berechungsalgorithmus sind kein Problem, wohl aber Änderungen der Variablen. In dem Fall muß die Funktion neu gestartet und die Änderungen aktiviert werden.

Nun wollen wir uns den Datentransfer von den Excelzellen in die VBA-Variablen näher ansehen. Dazu wechseln wir wieder in das VBA Fenster (Alt F11). Klicken Sie die erste Funktionszeile an und drücken Sie F9.

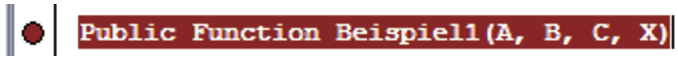


Abb. 1.3-14 Programmzeile mit F9 markieren

Wechseln Sie jetzt zurück in die Tabelle „Daten“ und klicken Sie auf die Zelle B9, dann auf das Funktionszeichen und es erscheint:

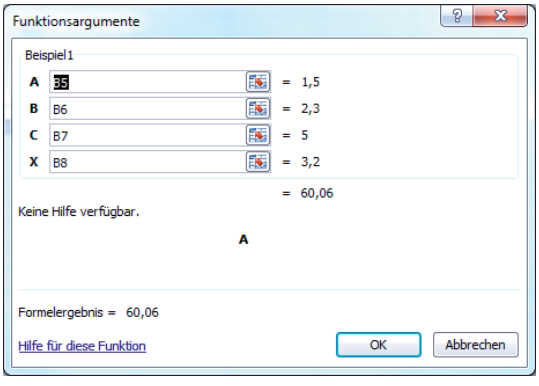


Abb. 1.3-15 Funktionsargumente

Wenn Sie jetzt OK drücken, springt Excel automatisch in VBA.

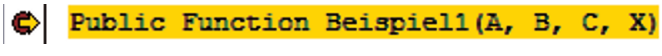


Abb. 1.3-16 Programmstop an der markierten Stelle

Die erste Zeile ist jetzt gelb markiert. Das bedeutet, dass diese Zeile nun ausgeführt wird.

Drücken Sie nun die F8 Taste.

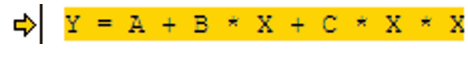


Abb. 1.3-17 Der 1. Schritte mit F8 nach dem Stopp

und halten Sie den Mauszeiger in der ersten Zeile auf A. Es erscheint A = 1,5. Wiederholen Sie das mit den anderen Variablen B, C und X. Sie finden dieselben Daten wie in der Excel Tabelle „Daten“. Drücken Sie nun F8. Der gelbe Balken wandert ein Zeile tiefer.

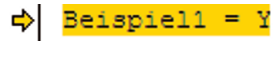


Abb. 1.3-18 Der 2. Schritt mit F8 nach dem Stopp

Halten Sie nun den Mauszeiger auf y in der vorherigen Zeile. Es wird y = 60,06 angezeigt. Sie können auch hier den Mauszeiger auf jede Variable der Formel halten, um dessen Inhalt anzuzeigen. Wenn Sie noch einmal F8 drücken, erscheint

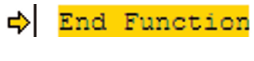


Abb. 1.3-19 Der 3. Schritt mit F8 nach dem Stopp

Nun wird das Ergebnis von y in die Funktionsvariable „Beispiel1“ übertragen. Wenn Sie den Mauszeiger in der Zeile davor auf y oder Beispiel1 halten, wird wieder 60,06 angezeigt.

Drücken Sie erneut F8 und Sie werden beobachten, dass das Ergebnis in die Zelle B6 geschrieben wird. Um einen Wert innerhalb der Funktion anzuzeigen, können Sie auch wie folgt verfahren:

Wiederholen Sie die Prozedur bis zu diesem Bild:

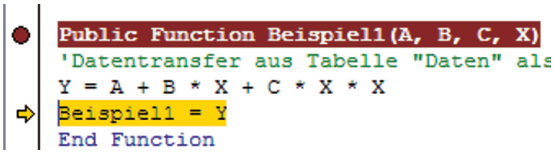


Abb. 1.3-20 Stopp an der Übergabe

Dann wählen Sie im Hauptmenü „Debuggen“ und „ Aktuellen Wert anzeigen“. Dort befindet sich ein Brillensymbol.

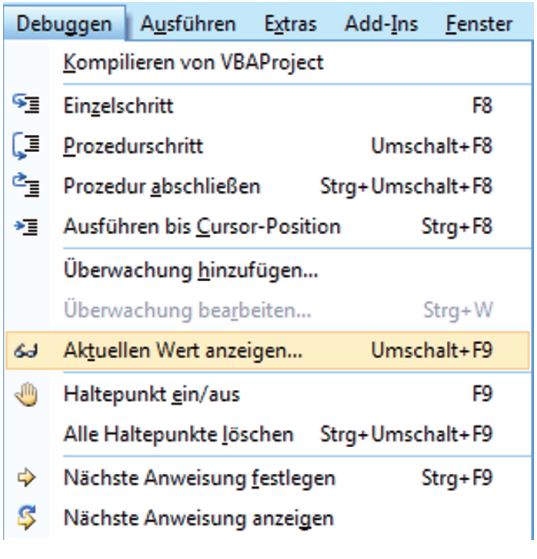


Abb. 1.3-21 Debuggen und Aktuellen Wert anzeigen

Es erscheint:

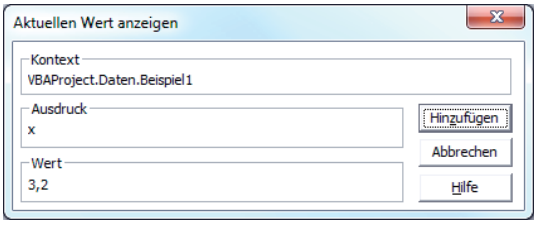


Abb. 1.3-22 Aktuellen Wert anzeigen und Hinzufügen

Hier sehen Sie, das x = 3,2 ist. Wenn Sie jetzt „Hinzufügen“ wählen, erscheint unten eine neue Zeile.

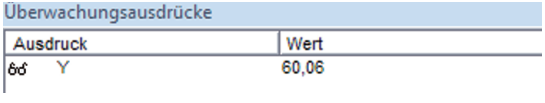


Abb. 1.3-23 Überwachungsfenster

Dies ist eine Überwachungszeile. Immer wenn Sie X in Zelle B8 ändern, wird dies hier angezeigt.

Nun heben wir die Markierung mit F9 auf. Klicken Sie die 1. Programmzeile an und drücken Sie F9. Falls Sie das Programm abbrechen müssen, wählen Sie den kleinen rechteckigen Knopf „Zurücksetzen“ im VBA Fenster .

Sie können soviele Überwachungsausdrücke einfügen wie Sie wollen. Wenn Sie im Überwachungsfenster mit der rechten Maustaste auf das Brillensymbol klicken, erscheint:

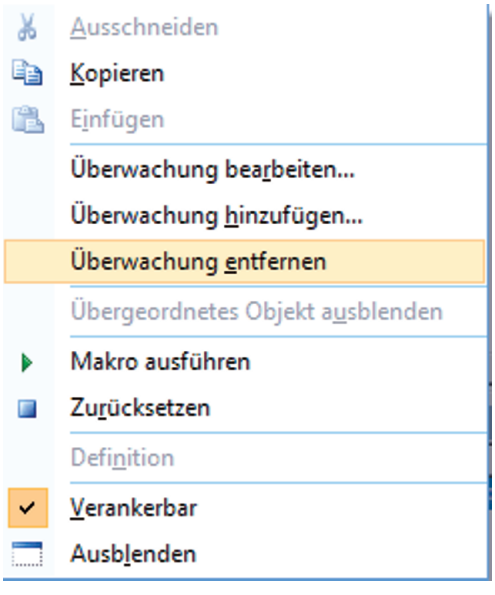


Abb. 1.3-24 Überwachung entfernen

Wenn Sie auf „Überwachen entfernen“ klicken wird die Überwachunsgzeile entfernt. Im nächsten Beispiel wollen wir die Datenübetragung in einem Makro betrachten.

Nun behandeln wir das Beispiel2. In der Tabelle „Daten“ finden Sie:

11		
12	Beispiel2	
13		
14	a	0,12
15	b	1,2
16	c	23,8
17	x	15
18	y	
19		

Abb. 1.3-25 Beispiel 2 Daten eintragen

Im VBA Fenster öffnen Sie nun das Makro „Beispiel1“. Dazu wechseln Sie in das VBA Fenster mit Alt F11 und wählen das Modul „Daten“.

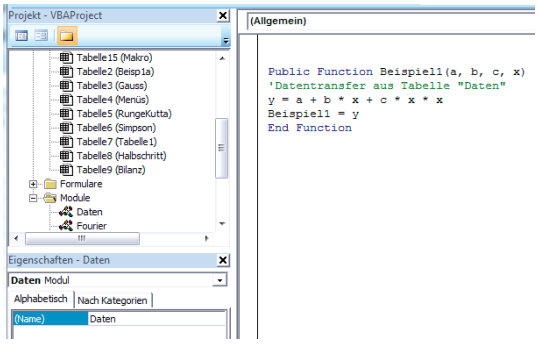


Abb. 1.3-26 Modul Daten

Dort finden Sie den VBA Code für Beispiel2. Es handelt sich nicht um eine Funktion, sondern um ein Makro. Dieses wird wie folgt erstellt:

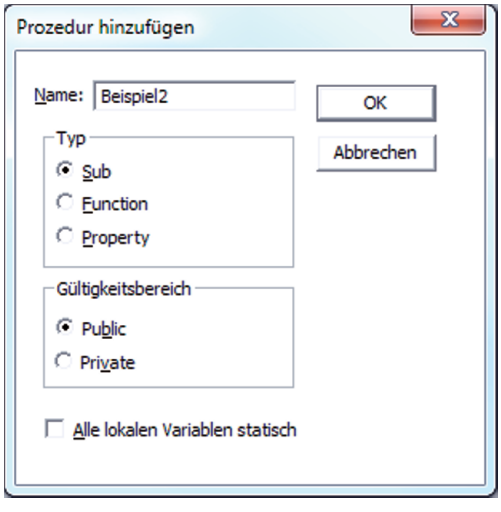


Abb. 1.3-27 Makro Beispiel2

VBA erstellt nun automatisch den Rahmen zum Makro. Fertig sieht das wie folgt aus:

```
Public Sub Beispiel2()
'Datentransfer aus Tabelle "Daten" als Makro
A = [B14]: B = [B15]: C = [B16]: X = [B17]
Y = A + B * X + C * X * X
[B18] = Y
End Sub
```

Abb. 1.3-28 Makro Beispiel2

Das Makro erkennt man an der 1. Zeile, dort steht Sub statt Function. Ein Makro funktioniuert nicht wie eine Funktion. Ein Makro arbeitet eigenständig wie ein Programm und wird nicht einer Zelle zugeordnet. Die Datenübertargeung funktioniert daher anders als bei eienr Funktion. Ein Makro, dessen Klammern leer sind, kann in Excel über „Entwicklertools“ „Beispiel2“ und „Ausführen“ gestartet werden. Wenn in der Klammer jedoch Variablen stehen, wie das bei der Funktion übkliche ist, kann dieses Makro nicht eigenständig gestartet werden, sondern muß von einem übergeordneten Makro gestartet, welches über die Variablen Daten übergibt werden.

In der Zeile

```
A = Cells(23, 2): B = Cells(24, 2): C = Cells(25, 2): X = Cells(17, 2)
```

werden Mit den Klammern […] kennzeichen Sie die Zelle in der aktuellen Tabelle. Diese aktuelle Tabelle muß die Tabelle „Daten“ sein, d.h. die muß angewählt sein. Wenn Sie jetzt das Makro „Beispiel2“ erneut starten, funktioniert es.

12	Beispiel2	
13		
14	a	0,12
15	b	1,2
16	c	23,8
17	x	15
18	y	5373,12

Abb. 1.3-29 Ergebnis

Wir lernen eine weitere Eigenschaft in „Debuggen“ kennen. Klicken Sie irgend eine Zeile des Makros „Beispiel2“ an und wählen Sie dann „Debuggen“.

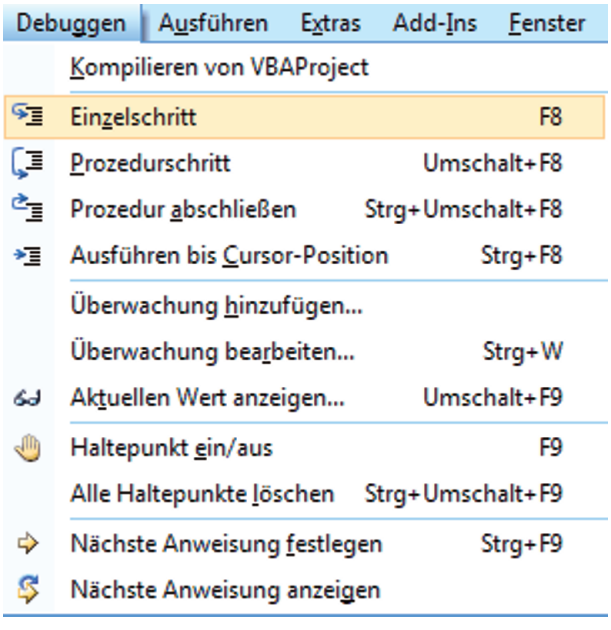
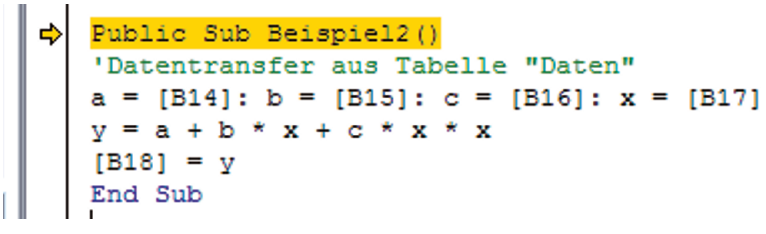


Abb. 1.3-30 Debuggen im Einzelschritt

Die erste Programmzeile wird automatisch gelb.



```
Public Sub Beispiel2()
'Datentransfer aus Tabelle "Daten"
a = [B14]: b = [B15]: c = [B16]: x = [B17]
y = a + b * x + c * x * x
[B18] = y
End Sub
```

Abb. 1.3-31 Programmstart

Mit der Taste F8 geht das auch. Drücken Sie nun die Taste F8 und Sie sehen, dass die gelbe Zeile um eins tiefer wandert. Auch hier könenn Sie die Werte der Variablen durch einfaches Berühren mit dem Mauszeiger oder dem Debug Befehl „Aktuellen Wert anzeigen“ sehen.

Im Beispiel 3 lernen Sie eine andere, flexible Art des Datentransfers kennen. Dazu schreiben Sie zunächst in Excel.

21	Beispiel3	
22		
23	a	0,12
24	b	1,2
25	c	23,8
26	x	15
27	y	

Abb. 1.3-32 Daten in Excel

Nun richten Sie sich wieder ein Makro „Beispiel3“ ein, wie oben beschrieben. Darin kopieren Sie zunächst den Inhalt von „Beispiel2“. Das sieht dann so aus:

```
Public Sub Beispiel3()
'Datentransfer aus Tabelle "Daten"
a = [B14]: b = [B15]: c = [B16]: x = [B17]
y = a + b * x + c * x * x
[B18] = y
End Sub
```

Abb. 1.3-33 Beispiel3

Ändern Sie dies wie folgt:

```
Public Sub Beispiel3()
'Datentransfer aus Tabelle "Daten"
a = Cells(14, 2): b = Cells(15, 2): c = Cells(16, 2): x = Cells(17, 2)
y = a + b * x + c * x * x
Cells(18, 2) = y
End Sub
```

Abb. 1.3-34 Datenübergabe mit dem Befehl Cells

Es ist leicht zu erkennen, dass aus [B14] cells(14,2) wird. Dabei steht die 14 für die Zeile (Row) und die 2 für die Soalte (Column). Sobald Sie Cells geschrieben haben, erscheint:

```
Default([RowIndex], [ColumnIndex])
```

Das bedeutet, das in der Klammer zuerst die Reihe, dann die Spalte geschrieben wird. Damit wäre Beispiel3 identisch mit Beispiel2. Jetzt wollen wir die Adressen aber so setzen, wie es der Darstellung in Excel entspricht.

```
Public Sub Beispiel3()
'Datentransfer aus Tabelle "Daten" als Makro
A = Cells(23, 2): b = Cells(24, 2): C = Cells(25, 2): x = Cells(26, 2)
Y = A + b * x + C * x * x
Cells(27, 2) = Y
End Sub
```

Abb. 1.3-35 Fertiges Makro

Starten Sie dieses Makro von VBA aus, indem Sie irgendwo in Beispiel3 reinklicken und F8 drücken. Wieder wird die 1. Zeile gelb markiert.

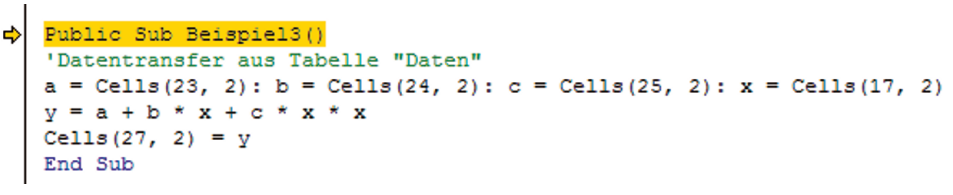


Abb. 1.3-36 Einzelschrittberechnung

Drücken Sie nun mehrfach F8, bis die Gelbmarkierung verschwunden ist. Zwischendurch prüfen Sie die Variablen wie oben gezeigt. Wenn alles ok ist, müßte in Zelle B27 die Zahl 5373,12 stehen. Natürlich können Sie dieses Makro auch über den Makro Befehl aus dem Hauptmenü unter „Entwicklertools“ starten.

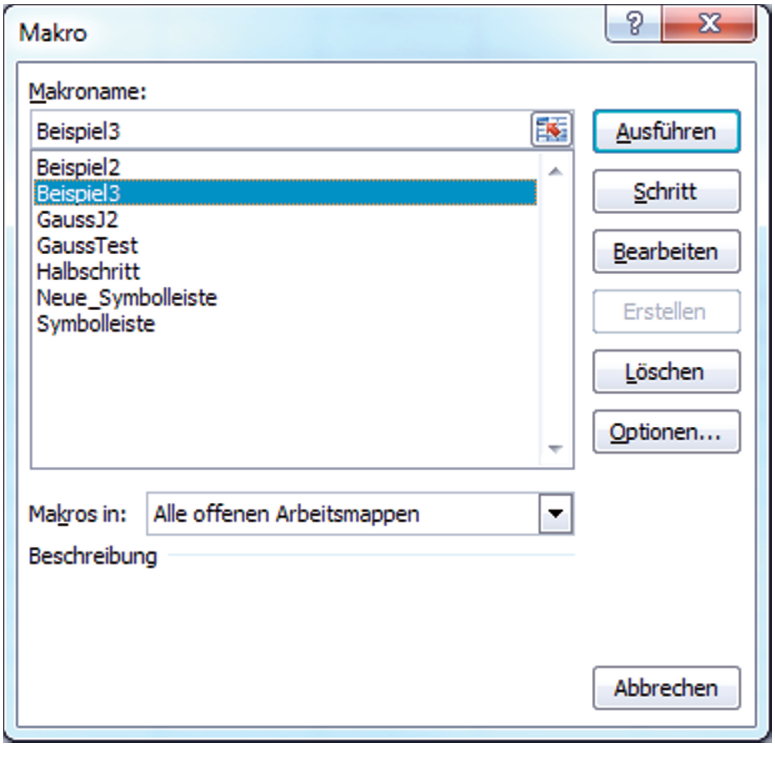


Abb. 1.3-37 Beispiel3 Bearbeiten

In nächsten Beispiel4 wollen wir die indizierte bzw. tabellarische Datenübertragung kennenlernen. Dazu legen wir keine neuen Daten in Excel an, sondern verwenden die von Beispiel3. Durch Kopieren erstellen wir uns wie zuvor erst einmal das Grundgerüst in VBA.

```
Public Sub Beispiel3()
'Datentransfer aus Tabelle "Daten" als Makro
A = Cells(23, 2): B = Cells(24, 2): C = Cells(25, 2): X = Cells(26, 2)
Y = A + B * X + C * X * X
Cells(27, 2) = Y
End Sub
```

Abb. 1.3-38 Beispiel4

Markieren Sie die beiden ersten Programmzeilen mit einem Apostroph (Shift #), sodass daraus ein Text wird und schreiben Sie die neuen Programmzeilen darunter, so dass es wie folgt ausieht:

```
Public Sub Beispiel4()
'Datentransfer aus Tabelle "Daten" als Makro
'a = Cells(23, 2): b = Cells(24, 2): c = Cells(25, 2): x = Cells(26, 2)
'ersetzt durch
Dim k(10)
For i = 1 To 4                      'Array definition
k(i) = Cells(22 + i, 2)             'Lesen der Zellen
Next i                              'Schleifenende

'y = a + b * x + c * x * x ersetzt durch
Y = k(1) + k(2) * k(4) + k(3) * k(4) * k(4)             'Formel
Cells(27, 2) = Y                    'Rückgabe an die Funktion
End Sub
```

Abb. 1.3-39 Datenübertragung als For Next Schleife

Starten Sie dieses Makro wie gewohnt in VBA mit F8 und testen Sie die Variable k(i). Sie werden die Daten der Zellen B23 bis B26 sehen. Neu ist der Befehl Dim k(10). Das bedeutet, dass eine indizierte Variable deklariert wird, die maximal 10 Indice haben kann. Hier werden nur 4 benötigt.

Neu ist auch der Befehl

for i = 1 to 4

Das bedeutet, daß die Variable i der Reihe nach die Zahlenwerte 1,2,3, 4 annimmt und jedesmal danach der Befehl

k(i) = Cells(22+i,2) ausgeführt wird. Dabei wird dasselbe erreicht wie zuvor mit a = Cells(23,2). Nur wird die Zahl 23 aus 22+1 bzw. 22+2 usw. ersetzt und a durch k(1) und b durch k(2) usw. Der Vorteil besteht darin, dass nur eine Zeile zum Datenlesen erforderlich ist.

Mit „next i“ wird diese Schleife erneut ausgeführt bis i = 4 ist. Dann erfolgt die eigentliche Formelberechnung.

Insgesamt erscheint diese Art Programm etwas umständlich und weniger leicht lesbar als Beispiel3. Das stimmt, jedoch ist sie bei größerer Datenmenge im Vorteil. Man denke nur daran, dass eine Tabelle aus 30 Zeilen und 20 Spalten in Variable gelesen werden muss. Allein dafür hätte man nicht genug Buchstaben zur Verfügung.

Es ist hier besonders empfehlenswert, die Programmschritte mit F8 zu prüfen.

1.4 Eigenes Programm schreiben

Tabelle: Daten

Modul: Daten

Makro: Kreisgrafik

In diesem einfachen Makro wollen wir die Koordinaten eines Kreises berechnen, diese in die Tabelle „Daten“ schreiben und grafisch als Kreis darstellen.

Erstellen Sie ein Makro mit dem Namen „Kreisgrafik“. Das fertige Programm sieht wie folgt aus:

```
Public Sub Kreisgrafik()
Dim x(100), y(100)                      'Array Definition
Pi = 4 * Atn(1)                         'Berechnen von Pi
z = 1                                   'Zeilenzähler
[D4] = "x"                              'schreibt x in die Zelle D4
[E4] = "y"                              'schreibt y in die Zelle E4
For i = 0 To 2 * Pi Step 2 * Pi / 100   'Schleifenbeginn
x(z) = Sin(i): y(z) = Cos(i)            'Winkelfunktionen zur Kreisberechnung
Cells(z + 4, 4) = x(z)                  'Sinus wird in die x-Spalte geschrieben
Cells(z + 4, 5) = y(z)                  'Coisinus wird in die y-Spalte geschrieben
z = z + 1                               'Zeile eins tiefer setzen
Next i                                  'Schleifenende
End Sub
```

Abb. 1.4-1 Makro Kreisgrafik

Mit der Variablen z berechen wir die Zeile, in die die Tabellendaten geschrieben werden. Nach jeder Berechnung wird der Wert von z um 1 erhöht und damit die Zeile um eins tiefer gesetzt. Damit ist die Zeilenzahl von i unabhängig.

Die Programmzeile

For i = 0 To 2 * Pi Step 2 * Pi / 100

ist in dieser Form neu. Der Winkel i im Bogenmass beginnt mit 0, schreitet aber dann nicht in den Schritten 1, 2, 3 usw. fort, sondern in Teilen des Kreises und der ist insgesamt 2π. Der Kreis soll in 100 Teilen berechnet werden. Daher muß die Schrittweite 2π/100 sein. Das erfüllt der Befehl Step, was Schrittweite bedeutet.

Das Ergebnis in Excel mit der Wahl des Hauptgitters sieht wie folgt aus.

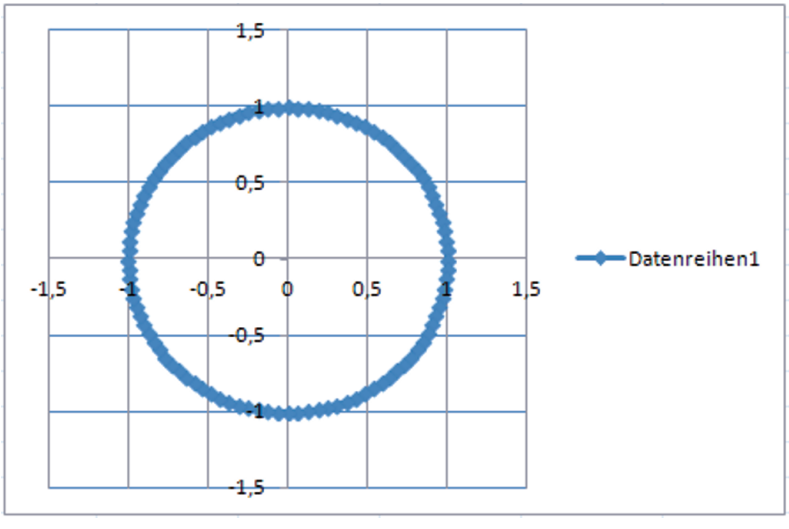


Abb. 1.4-2 Darstellung des Kreises

Hier sehen Sie den oberen, positiven Teil der 100 zeiligen Tabelle:

Kreisgrafik	
x	y
0	1
0,06279052	0,99802673
0,12533323	0,9921147
0,18738131	0,98228725
0,24868989	0,96858316
0,30901699	0,95105652
0,36812455	0,92977649
0,42577929	0,90482705
0,48175367	0,87630668
0,53582679	0,84432793
0,58778525	0,80901699
0,63742399	0,77051324
0,68454711	0,72896863
0,72896863	0,68454711

Abb. 1.4-3 Tabelle Kreisgrafik