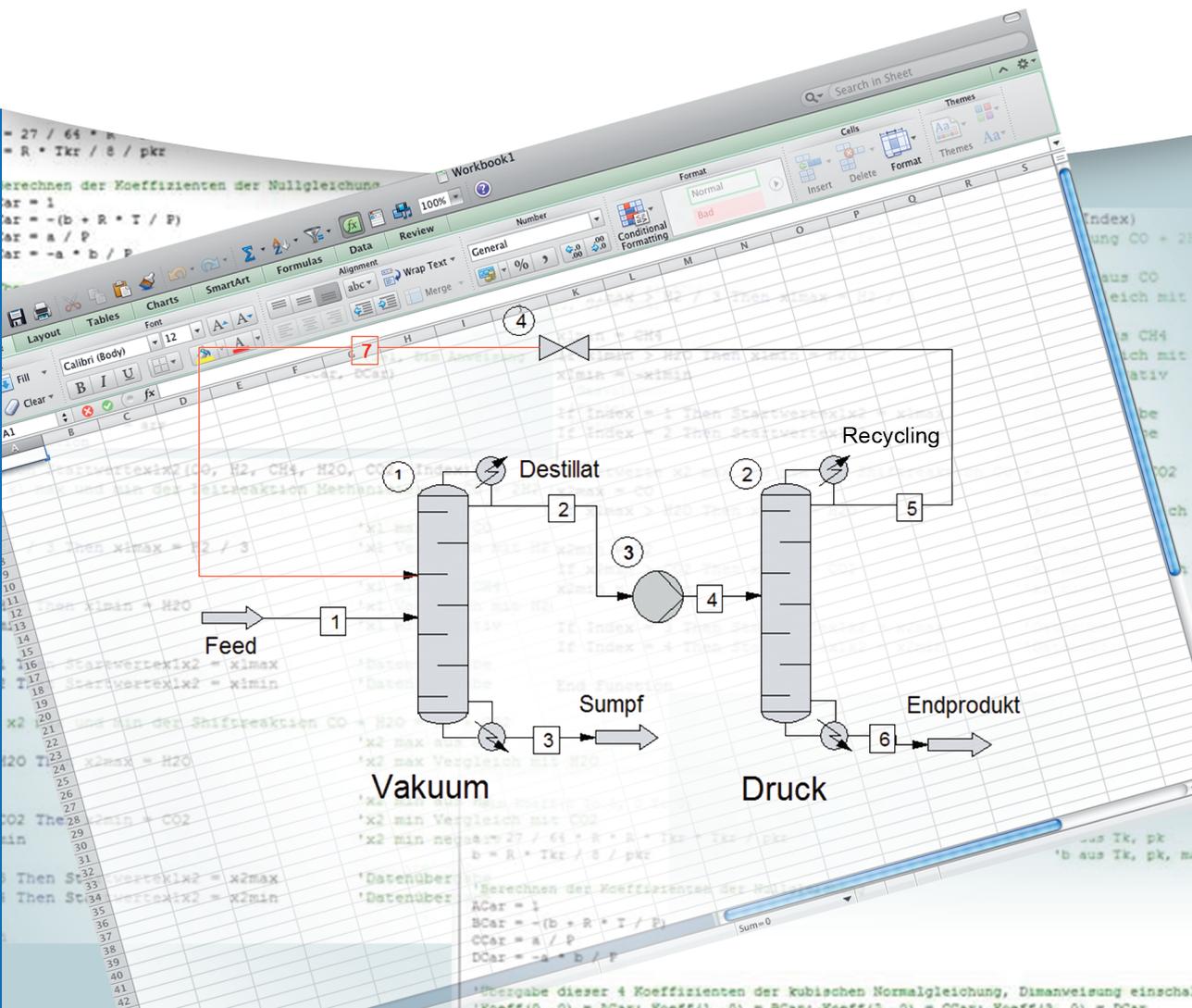


Shichang Wang und Wolfgang Schmidt

# 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<sup>2</sup>  
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

Alle Bücher von Wiley-VCH werden sorgfältig erarbeitet. Dennoch übernehmen Autoren, Herausgeber und Verlag in keinem Fall, einschließlich des vorliegenden Werkes, für die Richtigkeit von Angaben, Hinweisen und Ratschlägen sowie für eventuelle Druckfehler irgendeine Haftung

#### **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.

© 2015 Wiley-VCH Verlag & Co. KGaA, Boschstr. 12, 69469 Weinheim, Germany

Alle Rechte, insbesondere die der Übersetzung in andere Sprachen, vorbehalten. Kein Teil dieses Buches darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form – durch Photokopie, Mikroverfilmung oder irgendein anderes Verfahren – reproduziert oder in eine von Maschinen, insbesondere von Datenverarbeitungsmaschinen, verwendbare Sprache übertragen oder übersetzt werden. Die Wiedergabe von Warenbezeichnungen, Handelsnamen oder sonstigen Kennzeichen in diesem Buch berechtigt nicht zu der Annahme, dass diese von jedermann frei benutzt werden dürfen. Vielmehr kann es sich auch dann um eingetragene Warenzeichen oder sonstige gesetzlich geschützte Kennzeichen handeln, wenn sie nicht eigens als solche markiert sind.

**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 verwiesen. Es seien hier die

Verlage Addison-Wesley, Hanser, Mikrossoft 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 Programmiersprache 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

|          |  |            |
|----------|--|------------|
| <b>1</b> | <b>Funktionen für Excel und VBA</b>                                    | <b>1</b>   |
| 1.1      | Erstellen einer VBA Funktion   | 1          |
| 1.2      | Makros, aufnehmen und bearbeiten                                       | 9          |
| 1.3      | Einführung in die VBA Programmierung                                   | 17         |
| 1.3.1    | Daten in Tabellen und in VBA verbinden                                 | 17         |
| 1.4      | Eigenes Programm schreiben   | 34         |
| 1.5      | Berechnungen in eigener Benutzeroberfläche ausführen                   | 36         |
| 1.6      | Menüs programmieren  | 49         |
| 1.7      | Grafische Darstellungen in 3D  | 54         |
| 1.8      | Dreiecksdiagramme  | 64         |
| 1.9      | Datenaustausch mit Dateien   | 74         |
| <br>     |  |            |
| <b>2</b> | <b>Mathematische Methoden</b>  | <b>84</b>  |
| 2.1      | Funktionen und ihre grafische Darstellung                              | 84         |
| 2.2      | Berechnen von Reihen   | 92         |
| 2.3      | Steigung und Minimum einer Funktion                                    | 100        |
| 2.4      | Nullstellensuche   | 110        |
| 2.5      | Lösen von kubischen Gleichungen, die Cardanische Formel                | 114        |
| 2.6      | Lösen von Gleichungssystemen, die Gauß-Jordan Methode                  | 128        |
| 2.7      | Numerische Integration nach Simpson                                    | 135        |
| 2.8      | Numerische Lösung von Differentialgleichungen, die Runge-Kutta-Methode | 142        |
| 2.9      | Partielle Differentialgleichungen                                      | 147        |
| 2.10     | Lineare Regression   | 155        |
| <br>     |  |            |
| <b>3</b> | <b>Anwendungen in Chemie und Verfahrenstechnik</b>                     | <b>165</b> |
| 3.1      | Maßeinheiten und deren Umrechnung                                      | 165        |
| 3.2      | Berechnung von Gemischen   | 173        |
| 3.3      | Molgewicht eines Moleküls aus der Summenformel                         | 182        |
| 3.4      | Füllstandsberechnung von Behältern                                     | 188        |
| 3.5      | Reale Gasgleichung nach van der Waals und Soave-Redlich-Kwong          | 198        |
| 3.6      | Kompression und Expansion eines Gases                                  | 216        |

|          |  |            |
|----------|--|------------|
| 3.7      | Kompression realer Gase                              | 229        |
| 3.8      | Die barometrische Höhenformel der Atmosphäre         | 230        |
| 3.9      | Molekularpotentiale nach Coulomb                     | 232        |
| 3.10     | Chemisches Gleichgewicht nach van 't Hoff und Gibbs  | 253        |
| 3.11     | Methanisierung-Shift nach van t'Hoff                 | 264        |
| 3.12     | Reaktion nach Gibbs                                  | 287        |
| 3.13     | Chemische kinetische Reaktion nach Arrhenius         | 291        |
| 3.14     | Verbrennungsrechnung                                 | 302        |
| 3.15     | Polymerisation                                       | 315        |
| 3.16     | Elektrochemische Reaktion, Brennstoffzelle           | 321        |
| 3.17     | Wärme- und Stoffaustausch, stationär und instationär | 330        |
| 3.17.1   | Wärmeaustausch                                       | 330        |
| 3.17.2   | Stoffaustausch                                       | 339        |
| 3.18     | Dampf-Flüssiggleichgewicht, McCabe-Thiele-Diagramm   | 346        |
| 3.19     | Flüssig-Flüssiggleichgewicht                         | 356        |
| 3.20     | Fest-Flüssiggleichgewicht                            | 373        |
| 3.21     | Batchdestillation nach Rayleigh und Schlünder        | 383        |
| 3.22     | Das Biot-Savart-Gesetz und dessen Anwendung          | 399        |
| <b>4</b> | <b>Anhang</b>  | <b>412</b> |
| 4.1      | Auswahlmenü in Excel                                 | 412        |
| 4.2      | Kopieren von Excel Tabellen                          | 414        |
| 4.3      | Inhaltsverzeichnis                                   | 421        |
| 4.3.1    | Querverweis Literaturverzeichnis                     | 428        |
| 4.3.2    | Bildunterschriften                                   | 430        |
| 4.4      | Formelnummerierung                                   | 433        |
| 4.4.1    | Stichwortverzeichnis                                 | 437        |
| 4.5      | Tastenkombination                                    | 440        |
| <b>5</b> | <b>Literaturverzeichnis</b>                          | <b>443</b> |
|          | <b>Index</b>   | <b>446</b> |

## 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  $1 + 2*B3 + 3*B3*B3$ :

|   | A                           | B | C | D |
|---|-----------------------------|---|---|---|
| 1 | Polynom in Excel und in VBA |   |   |   |
| 2 |                             |   |   |   |
| 3 |                             | 1 | 6 |   |
| 4 |                             | 1 |   |   |
| 5 |                             |   |   |   |

**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.

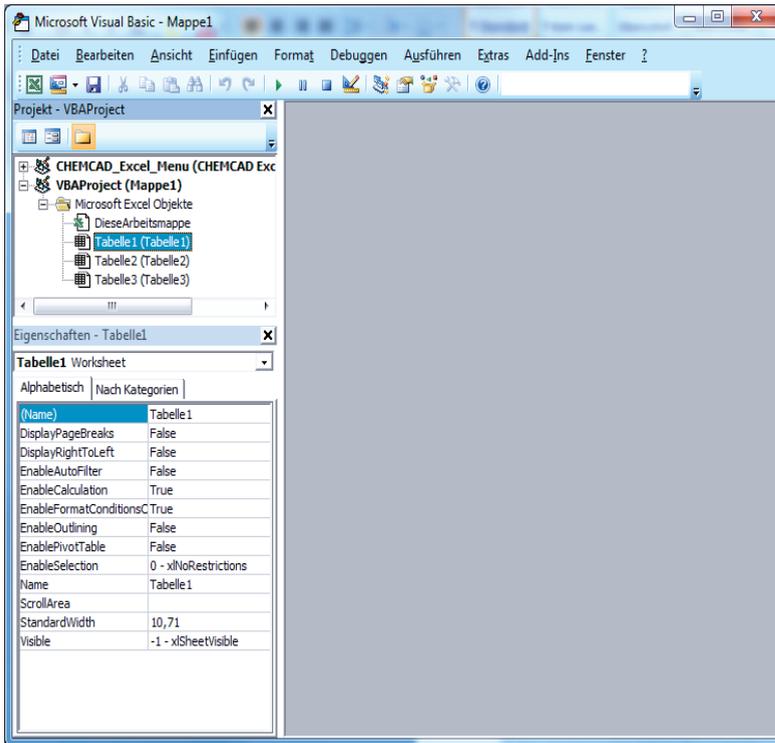


Abb. 1.1-2 Oberfläche Visual Basic (VBA)

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“.

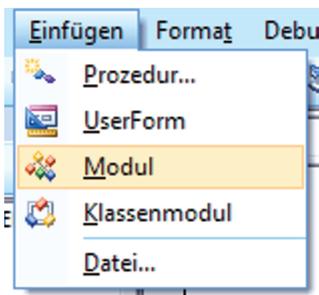


Abb. 1.1-3 Einfügen Modul

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

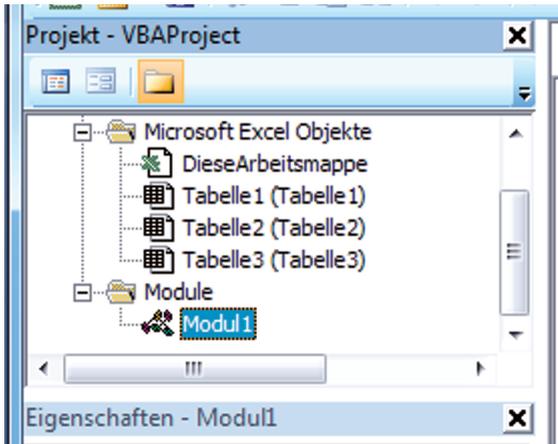


Abb. 1.1-4 Neues Modul

Im Eigenschaftsfenster tragen Sie den Namen „Polynom“ ein.

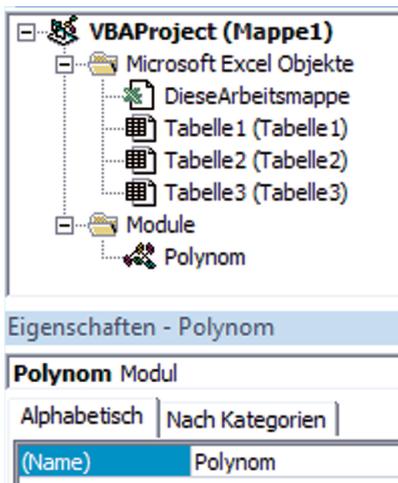


Abb. 1.1-5 Polynom als Modul

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

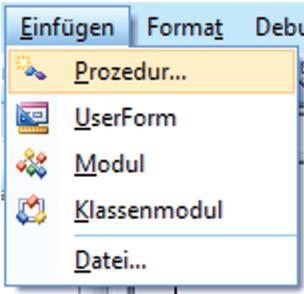


Abb. 1.1-6 Einfügen, Prozedur

Es erscheint ein neues Fenster.

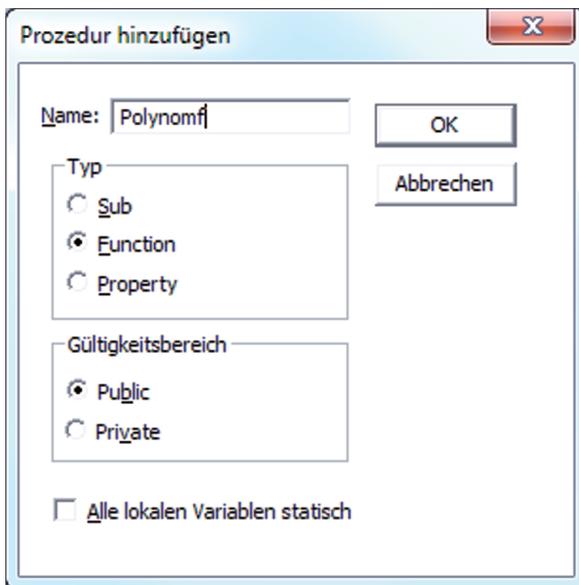


Abb. 1.1-7 Prozedur hinzufügen

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

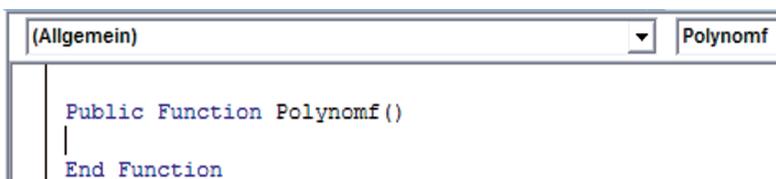
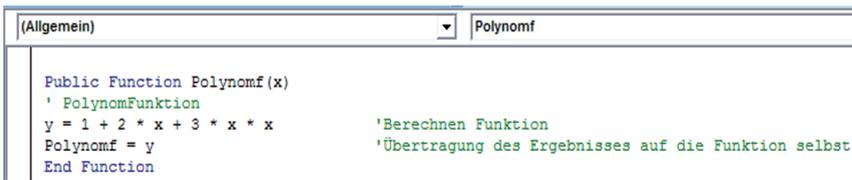


Abb. 1.1-8 Leere Funktion eröffnet

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



```

(Allgemein) Polynomf
Public Function Polynomf(x)
' Polynomfunktion
y = 1 + 2 * x + 3 * x * x      'Berechnen Funktion
Polynomf = y                  'Übertragung des Ergebnisses auf die Funktion selbst
End Function

```

**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

$$\text{Polynomf} = y$$

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 denen 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 Bearbeitungsleiste.



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

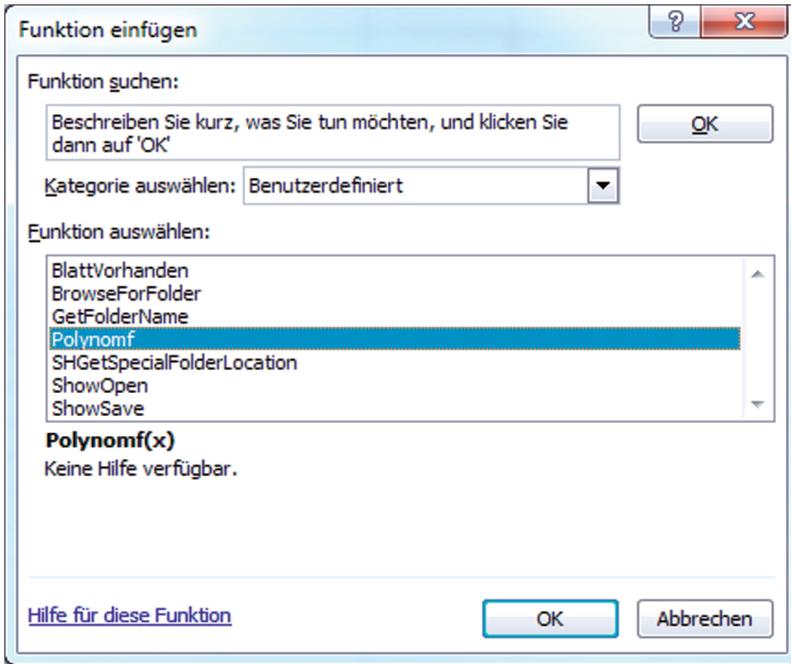


Abb. 1.1-10 Funktion einfügen

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

Es erscheint:

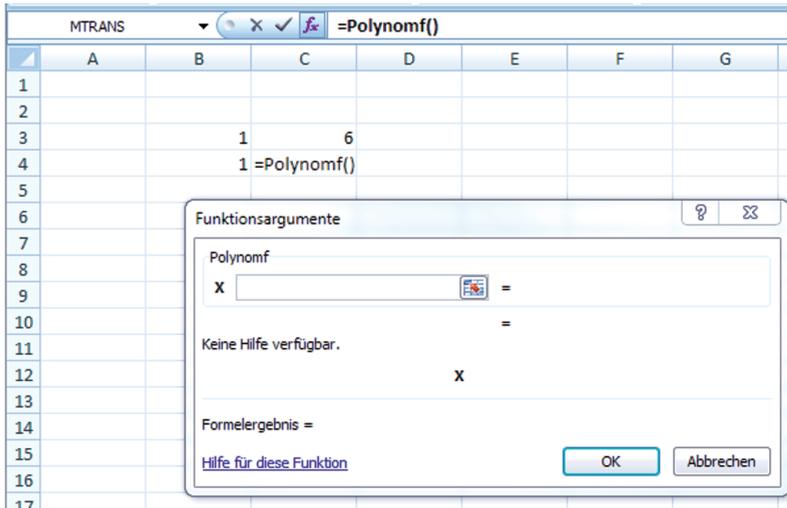


Abb. 1.1-11 Funktionsargumente

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.

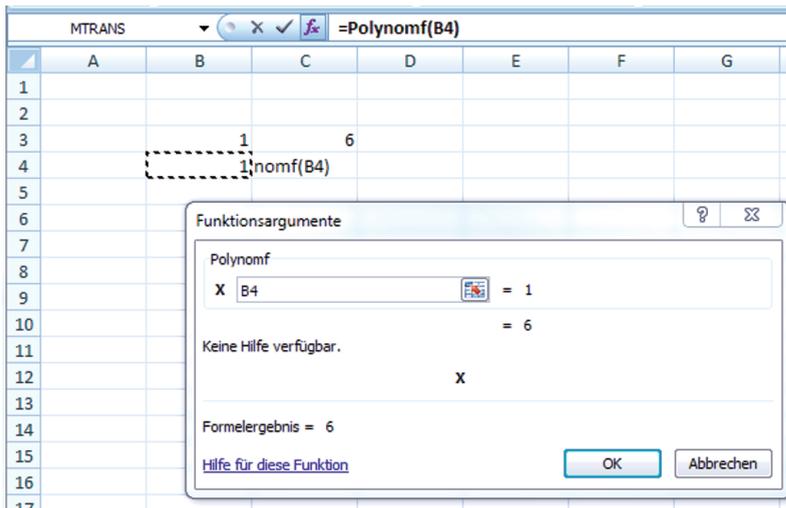


Abb. 1.1-12 Zelle zuweisen

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

In der Exceltabelle erscheint:

| C4 |   | fx =Polynomf(B4) |   |   |
|----|---|------------------|---|---|
|    | A | B                | C | D |
| 1  |   |                  |   |   |
| 2  |   |                  |   |   |
| 3  |   | 1                | 6 |   |
| 4  |   | 1                | 6 |   |

**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:

|  |    |     |
|--|----|-----|
|  | 10 | 321 |
|  | 10 | 321 |

**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  $\text{arr}(x)$ . 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

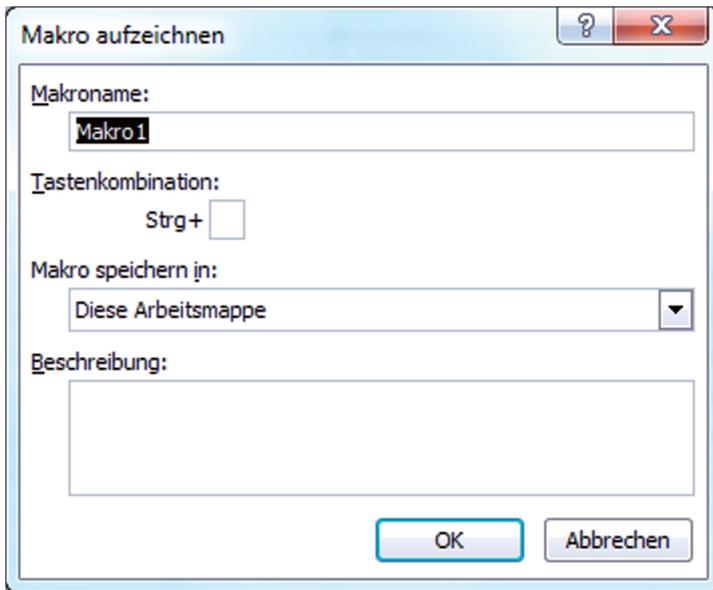
Tabelle: Makro

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`.

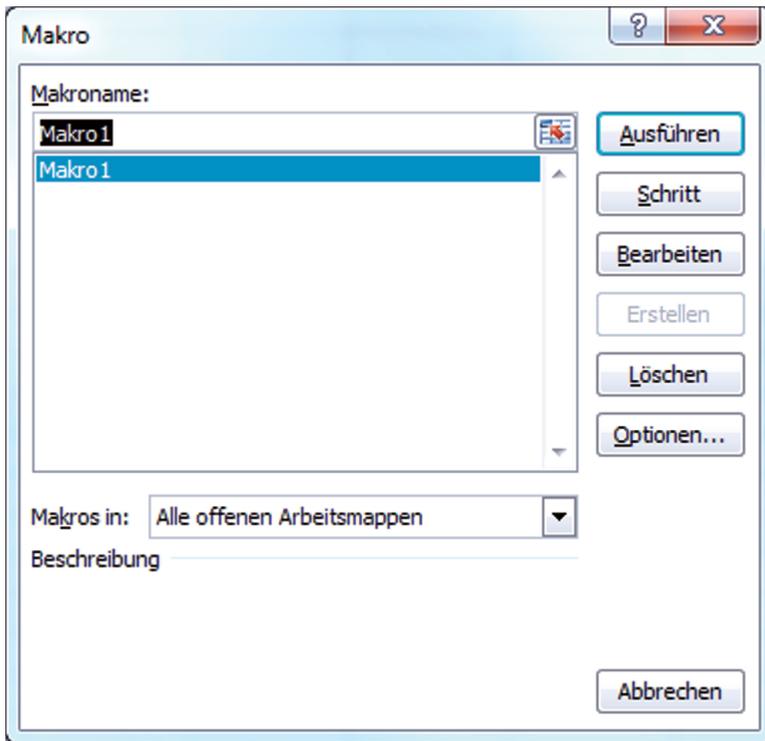
Ergebnis:

| fx =B3*C3 |   |   |
|-----------|---|---|
| B         | C | D |
|           |   |   |
| 2         | 3 | 6 |

**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“.



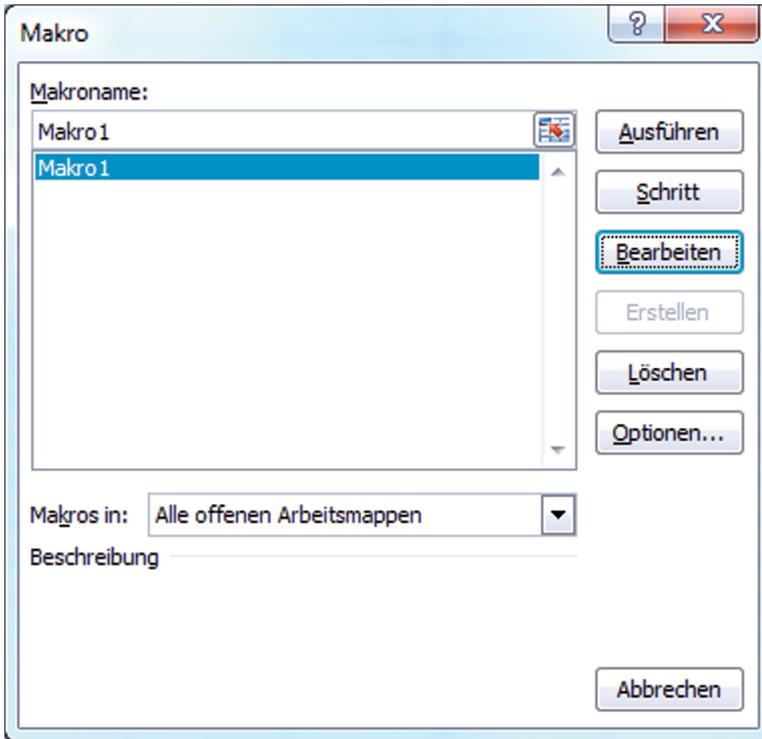
**Abb. 1.2-3 Makro auswählen und Ausführen**

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

|   |   | fx =B3*C3 |  |
|---|---|-----------|--|
|   | C | D         |  |
| 2 | 3 | 6         |  |

**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 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 Berechnungsformel 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 ActiveX-Steuerelemente:

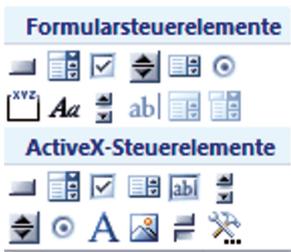


Abb. 1.2-9 Formularsteuerelemente und ActiveX-Steuerelemente

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:

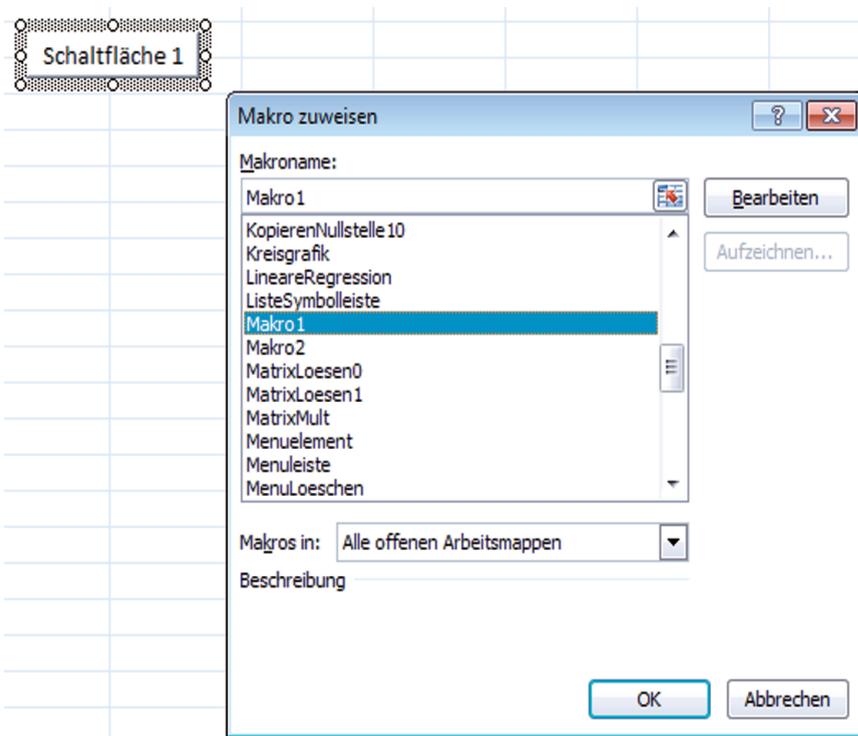


Abb. 1.2-10 Schaltfläche und Makro zuweisen

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:

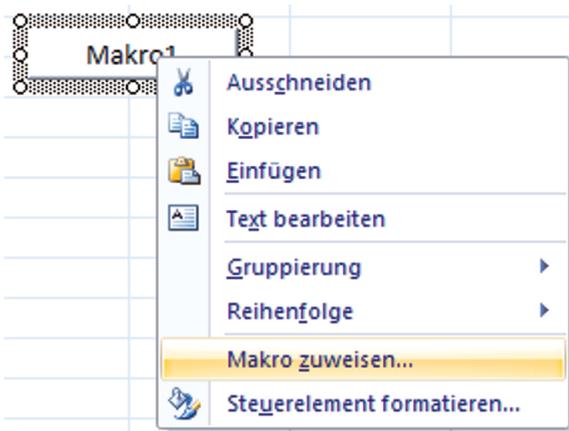
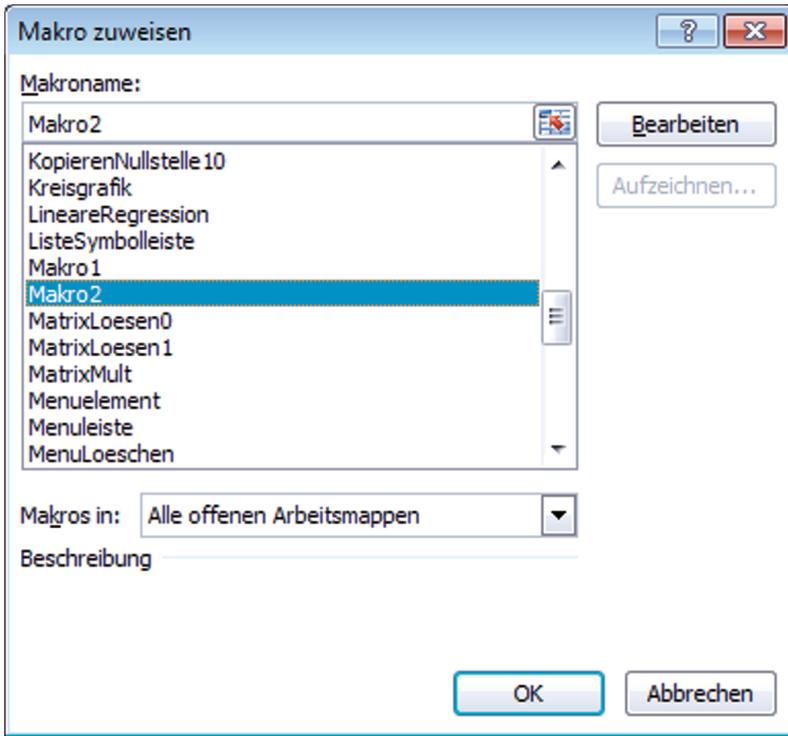


Abb. 1.2-11 Makro erneut zuweisen



**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

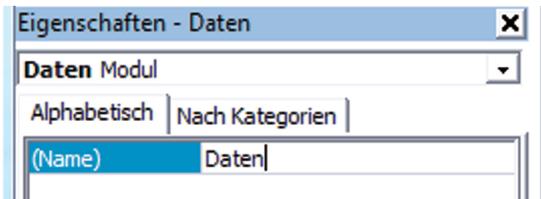
Nachdem Sie im vorherigen Kapitel bereits gelernt haben, eine Funktion in VBA zu erstellen, lernen Sie hier an 4 Beispielen, wie Sie die Zellen in Excel mit den Variablen in VBA verbinden können. Diese Technik wird sowohl bei der Dateneingabe als auch bei der Datenausgabe benötigt. Damit wiederholen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

Im Beispiel 1 erstellen wir die folgende mathematische Polynomfunktion:

$$y = a + b * x + c * x^2$$

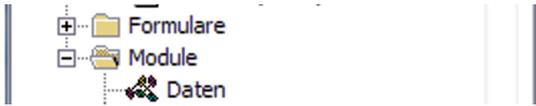
Die nachfolgenden Schritte wurden grundsätzlich z.T. bereits im vorangegangenen Kapitel besprochen. Deren Anwendung wird hier an dem vorliegenden Beispiel vertieft.

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



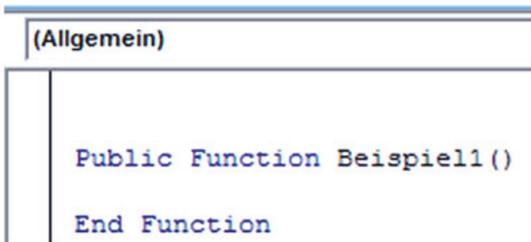
**Abb. 1.3-1 Modul Daten benennen**

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  $x^{\wedge}2$ . Schneller rechnet VBA aber, wenn man  $x*x$  schreibt. Nun muß die Variable noch mit der Funktion verbunden werden. Schreiben Sie daher: `Beispiel1 = y`.