Kapitel 9

Die Funktionsfabrik: LAMBDA()

Die Funktion LAMBDA() als nächste Stufe der Excel-Evolution zu bezeichnen, ist keine Übertreibung. Sie kann nämlich etwas, was keine andere Funktion kann. Funktionen kalkulieren in der Regel Ergebnisse oder produzieren Arrays. LAMBDA() produziert Funktionen. Das war bisher nur mit der Makrosprache VBA möglich und setzte entsprechende Kenntnisse in der Excel-Programmiersprache voraus.

Jetzt hören wir schon die Nörgler im Hintergrund: Über 600 Funktionen an Bord, die meisten kennen maximal 20 davon, reicht das denn nicht langsam? Was bitte kann Excel mit seiner Funktionspalette nicht abdecken?

Zunächst mal: Die Welt wird sich auch ohne LAMBDA() weiterdrehen. Die Funktion ist nur für die Power-User und die Zauberer unter uns, zu denen die Leser dieses Buches ja gehören. LAMBDA() hat zudem einen anderen Problemlösungsansatz als andere Funktionen.

Wer komplexe Funktionskonstrukte entwirft, Funktionen schachtelt und Bedingungen kombiniert, erhöht nicht nur seinen eigenen Arbeits- und Zeitbedarf, sondern auch den seiner Mitmenschen, falls diese mit seinen Kunstwerken arbeiten müssen. Außerdem steigt das Risiko für Kalkulationsfehler proportional zur Länge der Formeln.

Mit LAMBDA() werden Funktionsmonster in verständliche Begriffe verpackt. Der Benutzer muss nicht den ganzen Weg zum Ergebnis verstehen, er bekommt eine Anleitung, welche Parameter er angeben muss, den Rest macht die LAMBDA()-Funktion.

9.1 Das Prinzip

Zunächst die Syntax. In der Funktionsklammer wird eine Reihe von maximal 255 Argumenten angegeben, die entweder Parameter für die Weiterberechnung oder die Berechnung selbst darstellen. Das letzte Argument ist immer die Kalkulation. Sie kann, muss aber nicht, alle zuvor bestimmten Parameter enthalten.

```
=LAMBDA(Parameter oder Kalkulation; ... Kalkulation)
```

Hier ein Beispiel: Für das Produkt aus zwei Zahlen genügt die Formel

```
=zahl1*zahl2
```

Stehen die beiden Parameter in A1 und A2, sieht das Ganze so aus:

```
A1: 100
A2: 3
A3: =A1*A2
```

Die LAMBDA()-Funktion bekommt zwei (willkürlich benannte) Parameter und anschließend die Kalkulation:

```
=LAMBDA(zahl1;zahl2;zahl1*zahl2)
```

Damit die Funktion richtig rechnet, werden ihr in einem weiteren Klammernpaar die Zahlen übermittelt:

```
=LAMBDA(zahl1;zahl2;zahl1*zahl2)(A1;A2)
```

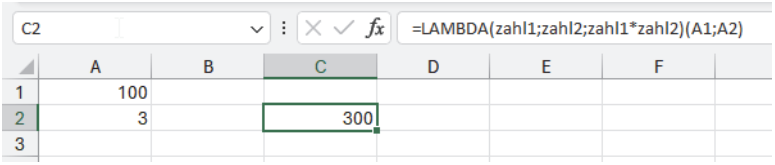


Abbildung 9.1: LAMBDA() berechnet einen Ausdruck mit zwei Variablen

Legen wir einen Bereichsnamen MULTI an und weisen diesem die LAMBDA()-Formel – ohne die zweite Klammer – als Bezug zu, lässt sich die Kalkulation anschließend so ausführen:

```
=MULTI(A1;A2)
```

9.2 Eine neue Funktion entsteht

In der Praxis sind die Formeln, die LAMBDA() übernimmt, natürlich komplexer. Entwickeln Sie eine weitere Funktion, die Excel so nicht hat: Sie berechnet, ob ein Suchkriterium in einem Textstring enthalten ist. Wir nennen die Funktion ENTHÄLT().

Beispieldaten: Die Liste in A1:B6 enthält Namen von Mitarbeitern und ihre Hobbys in einem Textstring mit dem Semikolon als Trennzeichen.

	A	B
1	Mitarbeiter	Hobbys
2	Marco	Golf;Tennis;Skifahren
3	Hennes	Tennis;Skateboarden;Kitesurfen
4	Stefan	Golf;Squash
5	Emma	Radfahren;Schwimmen
6	Kerstin	Tischtennis;Snowboarden;Tennis

Abbildung 9.2: Die Ausgangsbasis unseres Beispiels

Die Formel: Die erste Funktion, die zum Einsatz kommt, ist FINDEN(), sie sucht den Text (hier in $C$1) in der ersten Mitarbeiterzeile (B2) und liefert die Position des Textes oder #WERT.

Das Ergebnis wird mit ISTZAHL() in WAHR oder FALSCH umgewandelt und mit SUMME() aufsummiert:

```
=SUMME(--ISTZAHL(FINDEN($C$1;B2))>0)
```

Die doppelten Negativzeichen stellen sicher, dass die Summe WAHR und FALSCH in 1 und 0 umwandelt.

Das Ergebnis für die Liste sieht so aus:

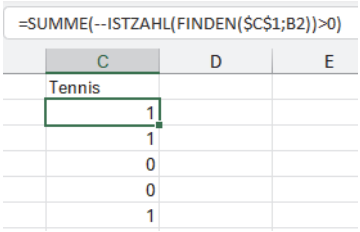


Abbildung 9.3: Ein Zwischenergebnis

LAMBDA()-Funktion anlegen: Die Formel wird in eine LAMBDA()-Funktion verpackt. Die startet mit zwei Variablen (*text* und *suchtext*) und verwendet diese in der Suchformel:

```
=LAMBDA(suchtext;text;SUMME(--ISTZAHL(FINDEN(suchtext;text))>0)
```

LAMBDA()-Funktion anwenden: Ohne zusätzliche Parameter würde die Funktion einen Fehlerwert ausgeben. Deshalb wird in einer weiteren Klammer Suchtext und Text angegeben:

```
=LAMBDA(text;suchtext;
  SUMME(--ISTZAHL(FINDEN(suchtext;text))>0))(B2;"Tennis")
```

Funktion ENTHÄLT() anlegen: Mit *Formel/Definierte Namen/Namen definieren* wird ein neuer Bereichsname ENTHÄLT angelegt. Als Bereich wird die Arbeitsmappe definiert, damit die Funktion für die gesamte Mappe gilt. Im Kommentar sind die Parameter erklärt, und die Bezugszeile enthält die LAMBDA()-Funktion ohne die Parameter in der zweiten Klammer.

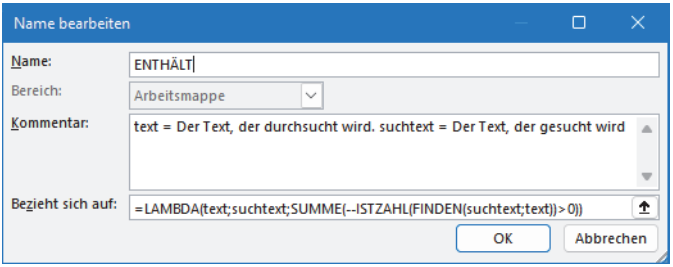


Abbildung 9.4: Ein neuer Bereichsname wird angelegt

Funktion ENTHÄLT() anwenden: Anstelle der LAMBDA()-Formel kann jetzt die neue Funktion verwendet werden, um die Liste nach einzelnen Hobbys zu durchsuchen.

C2 =ENTHÄLT($B2;C$1) =ENTHÄLT($B2;D$1)

	A	B	C	D
1	Mitarbeiter	Hobbys	Tennis	Golf
2	Marco	Golf;Tennis;Skifahren	1	1
3	Hennes	Tennis;Skateboarden;Kitesurfen	1	0
4	Stefan	Golf;Squash	0	1
5	Emma	Radfahren;Schwimmen	0	0
6	Kerstin	Tischtennis;Snowboarden;Tennis	1	0

Abbildung 9.5: Auf der Suche nach Hobbys

9.3 LAMBDA() in Tabellen

Für die Arbeit mit Tabellen bietet LAMBDA() den Vorteil, dass keine strukturierten Verweise in der Rechenformel nötig sind. Die »blasen« die Tabellenformeln in der Praxis nämlich ziemlich auf. Hier im Beispiel wird die Abfindung für die scheidenden Mitarbeiter berechnet. Die Parameter können wahlweise als Verweis oder als Bezug eingegeben werden.

Funktion Abfindung:

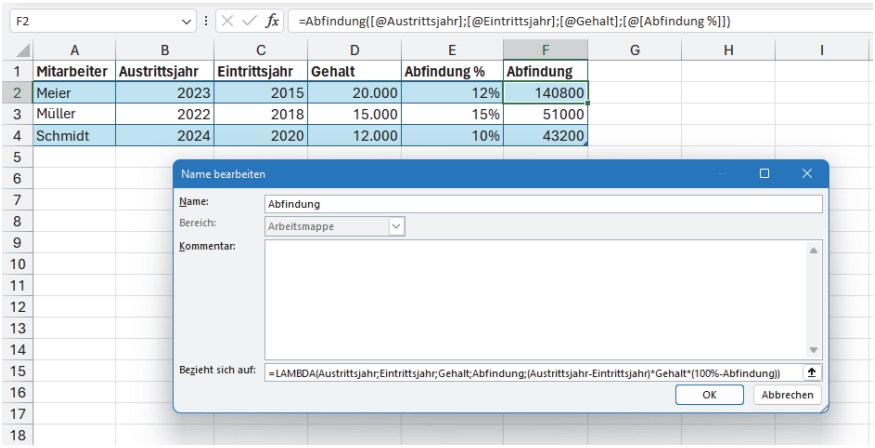
```
=LAMBDA(Austrittsjahr;Eintrittsjahr;Gehalt;Abfindung;(Austrittsjahr-
Eintrittsjahr)*Gehalt*(100%-Abfindung))
```

In der Tabelle:

```
=Abfindung([@Austrittsjahr];[@Eintrittsjahr];[@Gehalt];[@[Abfindung %]])
```

Oder:

```
=Abfindung(B2;C2;D2;E2)
```



	A	B	C	D	E	F
1	Mitarbeiter	Austrittsjahr	Eintrittsjahr	Gehalt	Abfindung %	Abfindung
2	Meier	2023	2015	20.000	12%	140800
3	Müller	2022	2018	15.000	15%	51000
4	Schmidt	2024	2020	12.000	10%	43200

Abbildung 9.6: LAMBDA()-Funktion für die Tabelle

Strukturierte Verweise lassen sich als Parameter in Textform in der LAMBDA()-Funktion verwenden. Werden sie über Bezüge oder Bereichsnamen geliefert, müssen sie mit INDIREKT() vorbehandelt werden.

Hier ein Beispiel:

Die Tabelle enthält Umsätze über Produkte, in J1 steht der Name der Tabelle, in J2 die auszuwertende Spalte. Die sortierte Liste mit eindeutigen Werten aus der bezeichneten Spalte liefert diese Funktionsschachtel:

```
=SORTIEREN(EINDEUTIG(INDIREKT(J1&"["&J2&"]")))
```

Die LAMBDA()-Funktion führt Tabellenname und Spalte als Parameter.

```
=LAMBDA(tbl;spalte;SORTIEREN(EINDEUTIG(INDIREKT(tbl&spalte))))(J1;"["&J2&"]")
Bereichsname: OBST
=OBST(J1;J2)
```

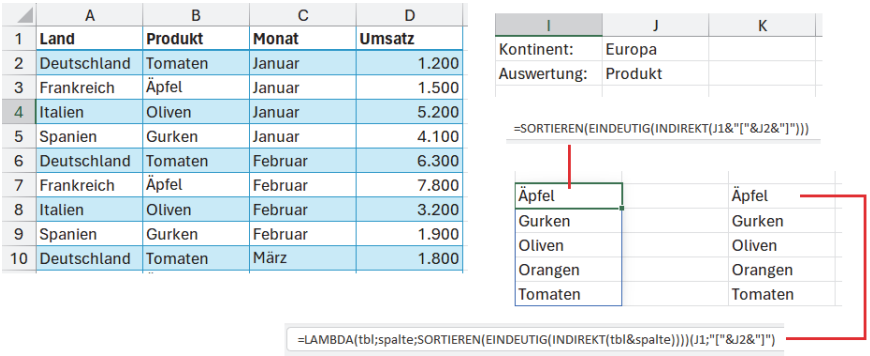


Abbildung 9.7: Tabelle und Tabellenspalte als LAMBDA()-Parameter

9.4 LAMBDA() für Diagramme

Die Entscheidung, das eigene Reporting oder Berichtswesen von PivotCharts auf Standard-Diagramme auf Basis von dynamischen Arrays umzustellen, ist nicht die falsche. Die Vorteile liegen auf der Hand: PivotCharts müssen händisch oder per Intervall aktualisiert werden und brauchen PivotTables als Basis.

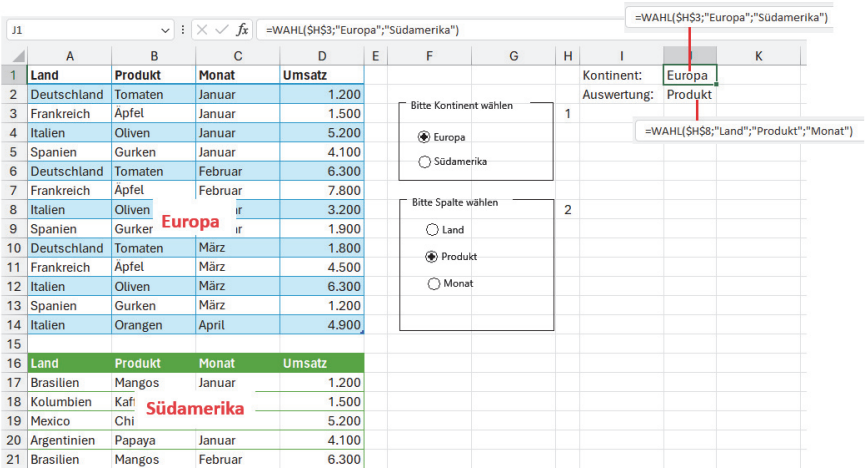
LAMBDA() kann hier den Arbeitsaufwand drastisch reduzieren. Aufwendige Formeln werden in LAMBDA()-Funktionen geschrieben und für die Diagrammdaten abgerufen.

Hier ein Beispiel mit Formularelement-Unterstützung:

Das Tabellenblatt *Obstimport* enthält zwei Tabellen, *Europa* und *Südamerika*, mit Importdaten und Umsätzen für Obst. Für die Auswahl des Kontinents wird über *Entwicklertools/Steuerelemente einfügen* ein Optionsfeldrahmen mit zwei Optionen gezeichnet, die gemeinsame Verknüpfung ist die Zelle $H$3.

Die zweite Optionsfeldgruppe bietet drei Optionen für die Spaltenwahl an, Ausgabeverknüpfung ist $H$8.

Das gewählte Ergebnis liefert je eine WAHL-Funktion in $J$1 und $J$2:



	Land	Produkt	Monat	Umsatz
2	Deutschland	Tomaten	Januar	1.200
3	Frankreich	Äpfel	Januar	1.500
4	Italien	Oliven	Januar	5.200
5	Spanien	Gurken	Januar	4.100
6	Deutschland	Tomaten	Februar	6.300
7	Frankreich	Äpfel	Februar	7.800
8	Italien	Oliven		3.200
9	Spanien	Gurker		1.900
10	Deutschland	Tomaten	März	1.800
11	Frankreich	Äpfel	März	4.500
12	Italien	Oliven	März	6.300
13	Spanien	Gurken	März	1.200
14	Italien	Orangen	April	4.900
15				
16	Land	Produkt	Monat	Umsatz
17	Brasilien	Mangos	Januar	1.200
18	Kolumbien	Kaf		1.500
19	Mexico	Chi		5.200
20	Argentinien	Papaya	Januar	4.100
21	Brasilien	Mangos	Februar	6.300

Abbildung 9.8: Auswahl der Tabelle und Spalte über Optionsfelder

Für Diagramme müssen alle Daten, also Rubrikenachse und Datenreihen, in Form von Bereichsnamen vorliegen. Schreiben Sie die komplexen Formeln als LAMBDA()-Funktionen und verwenden Sie diese für die beiden Bereiche *Obst_Rubrik* und *Obst_Umsatz*. Die Bereichsnamen sollten lokal sein, d. h. sich auf das Tabellenblatt beziehen.

Name	Formel
OBST	`=LAMBDA(tbl;spalte; SORTIEREN(EINDEUTIG(INDIREKT(tbl&"["&spalte&"]"))))`
Obst_Rubrik	`=OBST($J$1;$J$2)`
Obst_Daten	`=LAMBDA(produkt;SUMMEWENN( INDIREKT(Obstimport!$J$1&"["&Obstimport!$J$2&"]"); produkt;INDIREKT(Obstimport!$J$1&"[Umsatz]")))`
Obst_Umsatz	`=Obst_Daten(Obstimport!$H$11#)`

Zeichnen Sie ein leeres Säulendiagramm in das Tabellenblatt und aktivieren Sie *Diagrammentwurf/Daten/Daten auswählen*. Der Rubrikenbereich wird aus *Obst_Rubrik* gebildet, die erste und einzige Datenreihe bekommt den Namen *Umsatz* und den Bezug auf *Obst_Umsatz*.

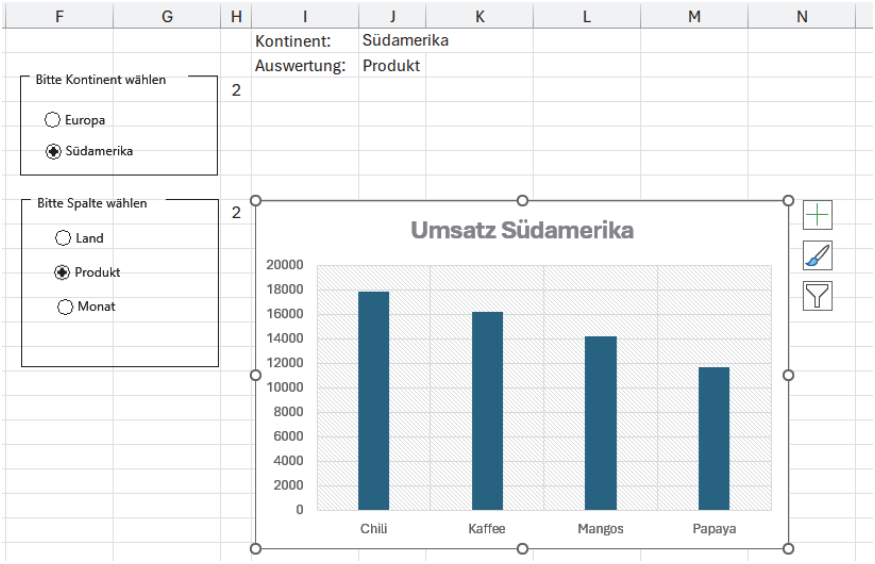


Abbildung 9.9: Das Diagramm mit Rubrik und Daten aus den Bereichsnamen

9.5 LAMBDA-Funktionen

Kurze Zeit nach der Einführung von LAMBDA() als Funktion tauchten im Microsoft-365-Update die ersten LAMBDA-Funktionen auf. Das sind Funktionen, die Aufgaben unter Verwendung einer LAMBDA-Funktion lösen. Hier eine Übersicht:

Funktion	Bedeutung
MAP()	Erstellt ein Array, in dem jedes Element über eine LAMBDA()-Funktion neu berechnet ist.
MATRIXERSTELLEN()	Generiert ein Array unter Verwendung einer LAMBDA()-Funktion.

Funktion	Bedeutung
NACHSPALTE()	Erstellt ein Spaltenarray unter Verwendung einer LAMBDA()-Funktion.
NACHZEILE()	Erstellt ein Zeilenarray unter Verwendung einer LAMBDA()-Funktion.
REDUCE()	Reduziert ein Array auf eine berechnete Größe unter Verwendung einer LAMBDA()-Funktion.
SCAN()	Scannt ein Array und berechnet jedes einzelne Element davon über eine LAMBDA()-Funktion

9.5.1 MAP()

Mit dieser Funktion wird ein Array Wert für Wert analysiert. Die LAMBDA()-Formel im zweiten Argument berechnet den Wert und gibt das Ergebnis in einem dynamischen Array aus. So werden zum Beispiel Wertebereiche mit logischen Bedingungen ermittelt: In E2 und E3 stehen die Grenzwerte für die Zahlen von A2 bis A10.

```
=MAP(A2:A10;LAMBDA(b;UND(b>E2;b<E3)))
```

Wie viele Werte ermittelt wurden, liefert die SUMME()-Funktion. Da diese aber WAHR und FALSCH nicht zählt, wird der Bezug noch mit Doppelminus berechnet.

```
=SUMME(--B2#)
```

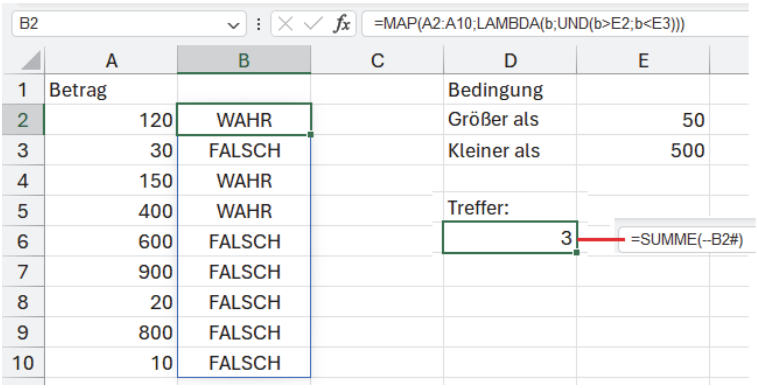


Abbildung 9.10: Das Diagramm mit Rubrik und Daten aus den Bereichsnamen